

ASP.NET 3.5

סדנת לימוד

בשפות C# ו-VB

קרא בהקדמה על קבצי קוד המקור

תרגום: צבי אלרואי

עריכה לשונית ועיצוב: שרה עמיהוד

עיצוב עטיפה: שרון רז

תודה ליצחק עמיהוד ולמאיר קרודו על הייעוץ המקצועי

שמות מסחריים

שמות המוצרים והשירותים המוזכרים בספר הינם שמות מסחריים רשומים של החברות שלהם. הוצאת הוד-עמי והוצאות Wiley ו-Wrox עשו כמיטב יכולתן למסור מידע אודות השמות המסחריים המוזכרים בספר זה ולציין את שמות החברות, המוצרים והשירותים. שמות מסחריים רשומים (registered trademarks) המוזכרים בספר צוינו בהתאמה.

הודעה

ספר זה מיועד לתת מידע אודות מוצרים שונים. נעשו מאמצים רבים לגרום לכך שהספר יהיה שלם ואמין ככל שניתן, אך אין משתמעת מכך כל אחריות שהיא. המידע ניתן "כמות שהוא" ("as is"). הוצאת הוד-עמי והוצאות Wiley ו-Wrox אינן אחראיות כלפי יחיד או ארגון עבור כל אובדן או נזק אשר ייגרם, אם ייגרם, מהמידע שבספר זה, או מהקבצים שבאתר (או מתקליטור/דיסקט/קבצי מחשב שעשויים להיות מצורפים לו).

Limit of Liability/Disclaimer of Warranty: The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Website is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Website may provide or recommendations it may make. Further, readers should be aware that Internet Websites listed in this work may have changed or disappeared between when this work was written and when it is read.

לשם שטף הקריאה כתוב ספר זה בלשון זכר בלבד. ספר זה מיועד לגברים ונשים כאחד ואין בכוונתנו להפלות או לפגוע בציבור המשתמשים/ות.

ASP.NET 3.5

סדנת לימוד

בשפות C# ו-VB

Imar Spaanjaars

ייעוץ מקצועי: מאיר קרודו dev@krudo.net



Beginning ASP.NET 3.5: In C# and VB

By Imar Spaanjaars

Copyright © 2008 by Wiley Publishing, Indianapolis, Indiana.
ISBN 978-0-470-18759-3

This Hebrew translation published under license by John Wiley & Sons, Inc.

Copyright: All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Legal Department, Wiley Publishing, Inc., 10475 Crosspoint Blvd., Indianapolis, IN 46256, (317) 572-3447, fax (317) 572-4355, or online at <http://www.wiley.com/go/permissions>

The Wiley Brand trade dress is a trademark of Wiley Publishing, Inc. in the United States and/or other countries. Used by permission: The WROX Brand trade dress is a trademark of Wrox, Inc. in the United States and/or other countries. Used by permission.

כל הזכויות שמורות © הוצאת הוד-עמי בע"מ

www.hod-ami.co.il info@hod-ami.co.il

אין לעשות שימוש מסחרי ו/או להעתיק, לשכפל, לצלם, לתרגם, להקליט, לשדר, לקלוט ו/או לאחסן במאגר מידע בכל דרך ו/או אמצעי מכני, דיגיטלי, אופטי, מגנטי ו/או אחר – בחלק כלשהו מן המידע ו/או התמונות ו/או האיוורים ו/או התוכניות ו/או כל תוכן אחר הכלולים ו/או שצורפו לספר זה. כל שימוש החורג מציטוט קטעים קצרים במסגרת של ביקורת ספרותית אסור בהחלט, אלא ברשות מפורשת בכתב מהמוציא לאור. הוראות אלו משלימות את הוראות הזכויות באנגלית לעיל.

הודפס בישראל

מהדורה ראשונה 10/2009

מסת"ב 978-965-361-400-0 ISBN

תוכן עניינים מקוצר

xxi	הקדמה
1	פרק 1: ASP.NET 3.5 מתחילים
34	פרק 2: בניית אתר אינטרנט בסביבת ASP.NET
66	פרק 3: עיצוב דפי האינטרנט שלך
107	פרק 4: עבודה עם פקדי ASP.NET
141	פרק 5: תכנות דפי Web ב-ASP.NET
203	פרק 6: יצירת אתרי Web בעלי מראה עקבי
248	פרק 7: ניווט באתר
282	פרק 8: פקדי משתמש
308	פרק 9: בדיקת אימות הקלט מהמשתמש
340	פרק 10: ASP.NET AJAX
373	פרק 11: מבוא לבסיסי נתונים
405	פרק 12: הצגה ועדכון של נתונים
450	פרק 13: LINQ
501	פרק 14: הצגת נתונים - נושאים מתקדמים
545	פרק 15: אבטחת אתר אינטרנט בסביבת ASP.NET 3.5
588	פרק 16: התאמה אישית של אתרי אינטרנט
624	פרק 17: טיפול בחריגים, ניפוי ומעקב (Exception Handling, Debugging, and Tracing)
666	פרק 18: פריסת אתר אינטרנט (Deploying Your Web Site)

703	 נספח א': תשובות לתרגילים
724	 נספח ב': קביעת התצורה של SQL Server 2005/2008
744	 נספח ג': תכונות חדשות (SP1)
747	 נספח ד': Silverlight
759	 אינדקס

תוכן עניינים

xxi	הקדמה
xxii	למי מיועד ספר זה
xxii	במה עוסק הספר
xxvi	מבנה הספר
xxvi	מה נדרש כדי להשתמש בספר
xxvi	מוסכמות
xxviii	קוד מקור
xxix	טעויות דפוס או קוד
xxx	p2p.Wrox.com
xxx	SP1
xxx	על המחבר
xxx	על העורך המקצועי
xxx	על מחבר נספח ד'

1	פרק 1: ASP.NET 3.5 מתחילים
2	כלי הפיתוח Visual Web Developer של מיקרוסופט
3	הורדה של Visual Web Developer
3	התקנת Visual Web Developer Express Edition
4	כיצד זה פועל
5	יצירת אתר Web הראשון שלך בסביבת ASP.NET 3.5
9	היכרות עם ASP.NET 3.5
10	הבנת HTML
10	אלמנטים של HTML
12	מאפיינים של HTML
13	ההבדל בין HTML לבין XHTML
13	תמיד סגור את האלמנטים
13	תמיד השתמש באותיות "קטנות" עבור התגיות ושמות המאפיינים
13	תמיד הקף את ערך המאפיין בגרשיים
14	קנן את התגיות כהלכה
14	תמיד הוסף הצהרת DOCTYPE לדף שלך
14	מבט ראשון ב- ASP.NET Markup
15	היכרות עם סביבת הפיתוח (IDE)
15	שטח הפיתוח הראשי
15	התפריט הראשי (Main Menu)
16	אזור סרגל הכלים (Toolbar Area)
16	תיבת הכלים (Toolbox)
17	סייר הפתרון (Solution Explorer)

17	 סייר בסיס הנתונים (Database Explorer)
18	 טבלת המאפיינים (Properties Grid)
18	 חלון המסמך (Document Window)
18	 דף הפתיחה (Start Page)
21	 חלונות שמכילים מידע
22	 רשימת השגיאות (Error List)
22	 חלון הפלט (Output Window)
22	 חלון הסימנייה (Bookmark Window)
22	 החלון Find Results
23	 התאמה אישית של סביבת הפיתוח (IDE)
23	 סידור מחדש של החלונות
23	 התאמת תיבת הכלים
26	 התאמת חלון המסמך (Document Window)
26	 התאמת סרגלי כלים
27	 עריכת סרגלי כלים קיימים
27	 יצירת סרגלי כלים משלך
27	 התאמת קיצורי דרך למקלדת
28	 איפוס (reset) השינויים
28	 איפוס פריסת החלונות
28	 איפוס ארגז הכלים
28	 איפוס כל ההגדרות
30	 היישום לדוגמה
32	 עצות מעשיות לגבי Visual Web Developer
32	 סיכום
33	 תרגילים
34	 פרק 2: בניית אתר אינטרנט בסביבת ASP.NET
35	 יצירת אתרים באמצעות VWD 2008
35	 סוגים שונים של פרויקטים
35	 Web Site Projects
36	 Web Application Projects
36	 בחירת תבנית האתר הנכונה
37	 ASP.NET Web Site
37	 ASP.NET Web Service
37	 WCF Service
38	 Empty Web Site
38	 יצירה ופתיחה של אתר אינטרנט חדש
38	 יצירת אתרים חדשים
40	 פתיחת אתרי אינטרנט קיימים
41	 עבודה עם קבצים באתר האינטרנט שלך
41	 סוגי הקבצים הרבים של אתר אינטרנט בסביבת ASP.NET 3.5
42	 קבצי אינטרנט
44	 קבצי נתונים

45	הוספת קבצים קיימים
46	ארגון האתר שלך
48	סוגים מיוחדים של קבצים
49	לעבוד עם טפסי אינטרנט (Web Forms)
49	אפשרויות תצוגה שונות בטפסי אינטרנט
51	בחירה בין קוד ברקע לבין דפים עם קוד כלול
56	הוספת סימני עריכה לדף
56	הוספה ועיצוב טקסט
59	הוספת טבלאות וסימני עריכה אחרים
62	קישור בין דפים
64	עצות מעשיות לעבודה עם טפסי אינטרנט
65	סיכום
65	תרגילים

פרק 3: עיצוב דפי האינטרנט שלך

66	מדוע אתה צריך CSS?
67	בעיות בעריכה עם HTML
68	כיצד פותרת CSS את בעיות העיצוב
69	מבוא ל-CSS
72	שפת CSS
73	גיליון סגנון (Style Sheet)
73	Selectors (בוררים)
74	Universal selector
74	Type selector
74	ID Selector
75	Class Selector
75	הקבצה ושילוב של Selectors
76	מאפיינים (Properties)
77	ערכים (Values)
78	שימוש בהגדרה מקוצרת
85	הוספת CSS לדפים שלך
85	בחירת האפשרות לקישור גיליונות סגנון
87	עבודה עם CSS ב- Visual Web Developer
87	יצירת סגנונות חדשים בגיליונות סגנון חיצוני
93	יצירת גיליונות סגנון
99	החלת סגנונות
101	ניהול סגנונות
104	עצות מעשיות לעבודה עם CSS
105	סיכום
106	תרגילים

פרק 4: עבודה עם פקדי ASP.NET

107	מבוא לפקדי שרת
112	מבט מקרוב על שרתי פקד של ASP.NET

112	הגדרת פקדים בדפים שלך
112	מאפיינים משותפים לכל הפקדים
115	סוגי פקדים
115	פקדים בקטגוריה Standard
116	פקדים פשוטים
116	פקדי רשימה
121	פקדי מכולה
122	מבט מקרוב על הפקד Panel
124	הפקד Wizard
128	פקדים נוספים בקטגוריה Standard
128	ImageButton ו-LinkButton
128	ImageMap ו-Image
128	Calendar
128	FileUpload
128	Substitute ו-Localize, Literal
129	AdRotator
129	HiddenField
129	XML
129	Table
129	פקדי HTML
130	איך לבחור בין פקדי Standard לבין פקדי HTML
130	פקדי נתונים (Data)
130	פקדי בדיקת תקפות (Validation)
130	פקדי ניווט (Navigation)
131	פקדי הזדהות - כניסה למערכת (Login)
131	הרחבות Ajax
131	מנוע המצב (State Engine) של ASP.NET
132	מה זה מצב (State) ומדוע הוא חשוב?
133	כיצד פועל מנוע המצב
137	לא כל הפקדים סומכים על ViewState
137	הערה לגבי ViewState וביצועים
138	עצות מעשיות לגבי עבודה עם פקדים
139	סיכום
140	תרגילים

פרק 5: תכנות דפי Web ב-ASP.NET 141

142	מבוא לתכנות
142	סוגי נתונים ומשתנים
146	המרת סוגי נתונים
148	שימוש במערכים (Arrays) ואוספים (Collections)
148	הגדרה ועבודה עם מערכים (Arrays)
151	הגדרה ועבודה עם אוספים (Collections)
152	מבוא לתבניות הקוד Generics

153	משפטים (Statements)
154	אופרטורים (Operators)
154	אופרטורי הצבה
154	אופרטורים אריתמטיים
156	אופרטורי השוואה
158	אופרטורי שרשור
159	אופרטורים לוגיים
161	קבלת החלטות
162	המבנים If Else, If Else-If
164	המבנים Switches / Select Case
169	לולאות
169	הלולאה For
171	הלולאות For Each / foreach
171	הלולאות While ו-Until
173	ארגון הקוד
173	שיטות: פונקציות ושגרות
176	התיקייה App_Code
180	ארגון קוד בעזרת מרחבי שמות (namespaces)
182	יבוא מרחבי שמות
182	כתיבת הערות
183	הערות כלולות בקוד
184	כתיבת הערות XML
185	יסודות תכנות מוכון אובייקטים
185	מונחי OO חשובים
185	אובייקטים
186	מחלקות
187	מאפיינים
189	יצירת מאפיינים לקריאה-בלבד ולכתיבה-בלבד
190	שיטות
191	פונקציות בנייה (בנאים)
193	הורשה
196	מגבילי גישה (Access Modifiers)
197	אירועים
199	עצות מעשיות לתכנות נכון
200	סיכום
201	תרגילים

פרק 6: יצירת אתרי Web בעלי מראה עקבי 203

203	עקביות עריכה ועיצוב של דף בעזרת דפים ראשיים
205	יצירת דפים ראשיים
208	יצירת דפי תוכן
211	מבט מקרוב בדפים ראשיים
212	קיבון של דפים ראשיים

212	אזהרות לגבי דפים ראשיים
213	שימוש בדף בסיס מרכזי
214	מבוא למחזור החיים של דף ASP.NET
216	יישום של דף הבסיס
221	יצירת תבניות דף לשימוש חוזר
225	ערכות נושא
226	סוגים שונים של ערכות נושא
226	בחירה בין Theme לבין StyleSheetTheme
227	החלת ערכות נושא
231	הרחבת ערכות נושא
233	החלפת ערכות נושא באופן דינמי
240	Skins
242	יצירת קובץ Skin
243	Named Skins
244	הערה אחרונה על Skins
245	עצות מעשיות ליצירת דפים עקביים
245	סיכום
247	תרגילים
248	פרק 7: ניווט באתר
249	התמצאות ברחבי האתר
249	מהם URL מוחלט ו-URL יחסי
249	URL יחסי
250	URL יחסי מבוסס-שורש
251	URL יחסי בפקד צד שרת
253	URL מוחלט
254	הבנת דפי ברירת המחדל
255	שימוש בפקדי הניווט
255	ארכיטקטורת פקדי הניווט
256	בחינת הקובץ Web.sitemap
256	היבטים חשובים של הקובץ Web.sitemap
258	שימוש בפקד Menu
259	דברים שיש לשקול כשמתמשים ב-Menu
260	יצירת גרסה בסיסית של הפקד Menu
264	עיצוב הפקד Menu
268	שימוש בפקד TreeView
272	שימוש בפקד SiteMapPath
274	הכוונה מחדש על ידי תכנות
274	הכוונת הלקוח לדף אחר על ידי תכנות
277	הכוונה מחדש מצד השרת
279	עצות מעשיות לניווט
280	סיכום
281	תרגילים

פרק 8: פקדי משתמש.....282

283.....	היכרות עם פקדי משתמש
283.....	יצירת פקדי משתמש
287.....	הוספת פקדי משתמש לדף תוכן או לדף ראשי
290.....	רישום פקדי משתמש ברחבי האתר
291.....	אזהרות לגבי פקדי משתמש
293.....	הוספת לוגיקה לפקדי משתמש
293.....	יצירת סוגי נתונים משלך עבור מאפיינים
299.....	יישום מאפייני ViewState
305.....	שיקולים לשימוש ב-ViewState
306.....	עצות מעשיות למשתמש בפקדי משתמש
306.....	סיכום
307.....	תרגילים

פרק 9: בדיקת אימות הקלט מהמשתמש.....308

309.....	איסוף נתונים מהמשתמש
310.....	אימות הקלט של המשתמש המוזן לדפי אינטרנט
310.....	פקדי האימות של ASP.NET
311.....	אזהרה בדבר בדיקת אימות בצד-לקוח
311.....	שימוש בפקדי אימות
314.....	פקדי האימות התקניים
315.....	ההבדל בין המאפיינים ErrorMessage ו-Text
315.....	אימות טווח – RangeValidator
316.....	אימות של ביטוי – RegularExpressionValidator
316.....	אימות השוואה – CompareValidator
320.....	הפקדים CustomValidator ו-ValidationSummary
325.....	עיבוד נתונים בשרת
326.....	שליחת דואר אלקטרוני מאתר האינטרנט שלך
326.....	קביעת תצורת האתר לצורך שליחת דואר אלקטרוני
327.....	יצירת הודעות דואר אלקטרוני
331.....	הבנת הצורך באימות של בקשה
332.....	קריאה מקבצי טקסט
337.....	עצות מעשיות לאימות נתונים
338.....	סיכום
339.....	תרגילים

פרק 10: ASP.NET AJAX.....340

341.....	מבוא ל-Ajax
341.....	שימוש ב-ASP.NET Ajax בפרויקטים שלך
342.....	יצירת דפים חופשיים מהבהוב
342.....	הפקד UpdatePanel
345.....	מבט מקרוב על הפקד UpdatePanel
346.....	הפקד ScriptManager
347.....	מתן משוב למשתמשים

347	הפקד UpdateProgress
352	שימוש בפקד Timer
357	שימוש בשירותי Web באתרי אינטרנט מבוססי Ajax
357	מהם שירותי Web?
359	יצירת שירותי Web
362	שימוש בשירותי Web באתר אינטרנט מבוסס Ajax
363	קביעת תצורת שירותי Web
363	קביעת תצורת ScriptManager
370	זו רק ההתחלה
370	עצות מעשיות להפעלת Ajax
371	סיכום
372	תרגילים

פרק 11: מבוא לבסיסי נתונים

374	מהו בסיס נתונים?
375	סוגים שונים של בסיסי נתונים טבלאיים
375	שימוש ב-SQL לעבודה עם נתוני בסיס נתונים
378	אחזור וביצוע פעולות על נתונים
378	קריאת נתונים
378	בחירת נתונים
379	סינון נתונים
381	סידור נתונים
385	צירוף נתונים
389	יצירת נתונים
389	עדכון נתונים
389	מחיקת נתונים
393	יצירת טבלאות משלך
393	סוגי נתונים במערכת SQL Server
394	הבנת מפתחות ראשיים וזהויות
398	יצירת קשרים בין טבלאות
402	עצות מעשיות לעבודה עם בסיסי נתונים
403	סיכום
404	תרגילים

פרק 12: הצגה ועדכון של נתונים

405	פקדי נתונים (Data Controls)
406	פקדים קשורי-נתונים (Data-Bound Controls)
406	פקדי רשימה (List Controls)
407	פקדי פריט יחיד (Single Item Controls)
408	פקדי דפדוף (Paging Controls)
408	פקדי מקור נתונים (Data Source Controls)
408	פעולה משותפת של פקדים קשורי-נתונים ופקדי מקור נתונים
409	הצגה ועריכה של נתונים על ידי GridView
415	הכנסת נתונים על ידי DetailsView

418		אחסון מחרוזות החיבור (Connection Strings) ב-web.config
420		סינון נתונים
427		התאמה אישית של מראה פקדי הנתונים
427		קביעת תצורת עמודות או שדות של פקדים קשורי-נתונים
433		עדכון והכנסת נתונים
433		שימוש ב-DetailsView להכנסת נתונים ועדכוןם
448		עצות מעשיות להצגת נתונים
448		סיכום
449		תרגילים

פרק 13: LINQ

451		מבוא ל-LINQ
452		הסוגים השונים של LINQ
452		LINQ to Objects
452		LINQ to XML
452		LINQ to SQL
452		הצגת LINQ to SQL
453		מיפוי מודל הנתונים למודל אובייקטים
458		הצגת תחביר השאילתה
458		אופרטורי שאילתה תקינים
459		Select
459		From
459		Order By
460		Where
460		Count, Average, Max, Min, Sum
461		SkipWhile, TakeWhile, Skip, Take
461		SingleOrDefault, Single
462		LastOrDefault, Last, FirstOrDefault, First
463		עיצוב נתונים עם סוגים אנונימיים (Anonymous Types)
468		שימוש בפקדי שרת עם שאילתות LINQ
469		פקדים חדשים שהוצגו ב-ASP.NET 3.5
469		הפקד LinqDataSource
475		הצגת הפקד ListView
492		הפקד DataPager
498		כמה הערות לגבי ביצועים
498		עצות מעשיות לעבודה ב-LINQ
499		סיכום
500		תרגילים

פרק 14: הצגת נתונים - נושאים מתקדמים

502		עיצוב הפקדים באמצעות סגנונות
503		מבוא לסגנונות
507		שילוב סגנונות, ערכות נושא ו-Skins
511		טיפול באירועים

512	ביקור חוזר במחזור החיים של דפי ופקדי ASP.NET
518	מחזור החיים של דף ASP.NET ואירועים בפקדי נתונים
525	טיפול בשגיאות שקורות בפקדי מקור נתונים
530	מיטמון (Caching)
531	מלכודות נפוצות במיטמון נתונים
532	הימנעות משימוש בנתונים שהתיישנו
532	אל תסמוך על כך שהנתונים יהיו שם
533	דרכים שונות להטמנת נתונים ביישומי אינטרנט בסביבת ASP.NET
534	מיטמון הפלט (Output Caching)
538	מיטמון עם פקדי מקור נתונים
539	מיטמון על ידי תכנות
543	עצות מעשיות לגבי נתונים
543	סיכום
544	תרגילים

פרק 15: אבטחת אתר אינטרנט בסביבת ASP.NET 3.5

546	היכרות עם נושאי האבטחה
546	זהות: מי אתה (identity)
546	אימות: איך תוכל להוכיח מי אתה?
546	הרשאה: מה אתה מורשה לעשות? (authorizations)
547	מבוא לשירותי היישום של ASP.NET
548	הצגת פקדי התחברות (Login)
554	פקדי Login
554	Login
556	LoginView
557	LoginStatus
557	LoginName
560	CreateUserWizard
563	PasswordRecovery
564	ChangePassword
567	קביעת התצורה של יישום האינטרנט
571	מנהל התפקיד (role manager)
572	קביעת תצורת מנהל התפקיד
572	ניהול משתמשים עם WSAT
577	עיצוב יישום אינטרנט כדי שיפעל עם תפקידים
582	בדיקת תפקידים על ידי תכנות
586	עצות מעשיות בנושא אבטחה
586	סיכום
587	תרגילים

פרק 16: התאמה אישית של אתרי אינטרנט

589	הבנת היישום Profile
590	עיצוב Profile
590	יצירת מאפייני Profile פשוטים

591	יצירת קבוצות פרופיל
591	שימוש בסוגי נתונים לא-תקינים
597	שימוש ב-Profile
615	דרכים אחרות להתמודדות עם Profile
616	זיהוי אנונימי
616	ניקוי פרופילים אנונימיים ישנים
618	הצצה לפרופילים של משתמשים אחרים
622	עצות מעשיות בנושא התאמה אישית
622	סיכום
623	תרגילים

פרק 17: טיפול בחריגים, ניפוי ומעקב

624 (Exception Handling, Debugging, and Tracing)

625	טיפול בחריגים
625	סוגים שונים של שגיאות
625	שגיאות תחביר
626	שגיאות לוגיות
627	שגיאות זמן ביצוע
627	תפיסת חריגים והטיפול בהם
635	טיפול גלובלי בשגיאות ודפי שגיאות מותאמים אישית
641	יסודות ניפוי השגיאות
645	תמיכת כלים בניפוי שגיאות
645	שוטטות בקוד במהלך הניפוי
646	חלונות ניפוי שגיאות
646	צפייה במשתנים
646	החלון Watch
647	החלון Autos
648	החלון Locals
648	חלונות אחרים
648	חלון Breakpoints
648	חלון Call Stack
648	החלון Immediate
652	ניפוי תסריטי צד-לקוח
656	מעקב (Tracing) אחר דפי האינטרנט של ASP.NET
657	שימוש ביכולות המעקב הרגילות
657	מעקב אחר דפים יחידים
658	מעקב אחר האתר כולו
660	הוספת מידע משלך ל-Trace
663	מעקב וביצועים
663	הערת אבטחה
663	עצות מעשיות בנושא ניפוי
664	סיכום
665	תרגילים

פרק 18: פריסת אתר אינטרנט (Deploying Your Web Site) 666

667	הכנת האתר לפריסה (deployment)
667	הימנעות מהגדרות מובנות בתוכנה
668	הקובץ web.config
668	תחביר הביטוי (expression)
669	המחלקה ConfigurationManager
674	העתקת האתר
675	יצירת עותק פשוט של האתר
678	הפצת האתר (publishing)
679	הפעלת האתר בשליטת IIS
679	התקנת שרת האינטרנט וקביעת התצורה שלו
680	כיצד לוודא ש-IIS מותקן
680	Windows XP ו- Windows Server 2003
681	Windows Vista
681	Windows Server 2008
683	התקנת ASP.NET וקביעת התצורה
688	האבטחה ב-IIS
689	הגדרות NTFS עבור Planet Wrox
694	איתור וטיפול בשגיאות שרת אינטרנט
695	העברת נתונים לשרת מרוחק
696	שימוש באשף הפצת בסיס הנתונים
698	יצירת בסיס הנתונים מחדש
699	רשימת תיוג לפריסה
701	מה הלאה?
701	סיכום
702	תרגילים

נספח א': תשובות לתרגילים 703

נספח ב': קביעת התצורה של SQL Server 2005/2008 724

724	קביעת התצורה של SQL Server 2005
725	מינוח ומושגים
725	אימות SQL Server (SQL Server Authentication)
725	אימות Windows
726	בחירה בין אימות Windows לבין אימות שרת
726	שימוש ב-SQL Server Management Studio
726	השגת והתקנת SQL Server Management Studio Express
727	הפעלת חיבורים מרחוק אל שרת SQL Server
728	צירוף בסיסי נתונים אל SQL Server
731	חיבור היישום אל SQL Server 2005
731	תרחיש 1 - שימוש באימות SQL Server
735	תרחיש 2 - שימוש באימות Windows כאשר IIS ובסיס הנתונים באותו מחשב

738.....	קביעת התצורה של שירותי היישום
739.....	קביעת התצורה של בסיס הנתונים עבור שירותי היישום
740.....	דריסת מחרוזת החיבור של LocalSqlServer
741.....	דריסת ההגדרות של שירותי היישום

744 נספח ג': תכונות חדשות (SP1)

744.....	ASP.NET 3.5 Service Pack 1
744.....	היסטוריית Ajax
745.....	jQuery
745.....	Entity Framework
746.....	נתונים דינמיים (Dynamic Data)
746.....	שירותי הנתונים של ADO.NET

747 נספח ד': Silverlight

747.....	מהו Silverlight?
752.....	ארכיטקטורת Silverlight
752.....	XAML
755.....	סגנונות ותבניות
756.....	קשירת נתונים
758.....	סיכום

759 אינדקס

הקדמה

כדי לבנות אתר אינטרנט מבוסס בסיס נתונים, שיהיה יעיל ומושך, יש צורך בשני דברים: סביבת עבודה יציבה ומהירה שבה ירוצו דפי האינטרנט, וסביבה עשירה ונרחבת ליצירה ותכנות של דפים אלה. שתי הדרישות מתמלאות עם ASP.NET 3.5 ו-Visual Web Developer 2008. ביחד הן מהוות את הפלטפורמה בה"א הידיעה ליצירת יישומי אינטרנט אינטראקטיביים ודינמיים.

ASP.NET 3.5 בנויה על גב קודמתה הפופולרית ASP.NET 2.0. סביבת העבודה NET 3.5 של מיקרוסופט בכלל, ו-ASP.NET 3.5 בפרט, הוסיפו מספר רב של תכונות נחוצות לתערוכת, בעודן שומרות על תאימות לאחור עם אתרים שנבנו תוך שימוש בגרסה הישנה יותר.

גרסה 2.0 של סביבת העבודה NET הכניסה את הרעיון של "פחות קוד", ASP.NET 3.5 ממשיכה רעיון זה ומאפשרת להשיג יותר עם עוד פחות קוד. תכונות חדשות כמו LINQ שהתווספו לסביבת העבודה NET, מאפשרות לגשת לבסיס נתונים ללא צורך בכתיבת קוד או עם כתיבה מעטה בלבד. שילוב ASP.NET Ajax של מיקרוסופט לתוך סביבת העבודה ASP.NET, ו-Visual Web Developer פירושו שביכולתך ליצור כעת ממשקי אינטרנט מהירי תגובה ואלגנטיים על ידי גרירת כמה פקדים אל הדף והגדרת כמה מאפיינים. הספר נותן מבט מעמיק בשתי טכנולוגיות אלו.

תמיכת Visual Web Developer ב-Cascading Style Sheets (CSS), שפת עיצוב דפי האינטרנט, עברה שיפוץ משמעותי. התמיכה בזמן עיצוב, שמציגה את הדף כפי שייראה בסופו של דבר בדפדפן, שופרה במידה ניכרת. בנוסף, Visual Web Developer מגיעה כעת עם מגוון כלים שהופכים את כתיבת CSS לנעימה וקלה.

עם זאת, התמיכה בגרור-ושחרר ובכלים חזותיים אינה הדבר היחיד שתלמד מספר זה. ASP.NET 3.5 ו-Visual Web Developer 2008 מגיעות עם קבוצת כלים מעולה ומקיפה שמטרתה לסייע בתכנות יישומי אינטרנט. טווח כלים זה נע מהתחביר החדש של LINQ שמאפשר לתשאל נתונים ובסיסי נתונים מיישומי אינטרנט, ועד ליכולות הניפוי המשופרות משמעותית שמאפשרות ניפוי יישום מ-JavaScript של צד-לקוח, לאורך כל הדרך עד לקוד צד-שרת. כל הכלים האלה מצויים בממשק המשתמש המוכר ובפקודות והפעולות המוכרות.

ASP.NET 3.5 משתמשת באותו runtime כמו גרסה 2.0. הדבר מבטיח תאימות מעולה לאחור עם הגרסה הישנה, כלומר, יישומי ASP.NET 2.0 ימשיכו להתבצע בסביבת העבודה החדשה. אבל אל תוטעה על ידי העובדה שה-runtime לא השתנה. למרות שהמבנה הטכני הבסיסי שנדרש לביצוע יישום אינטרנט לא השתנה, הרי שסביבת העבודה NET 3.5 ו-ASP.NET הוסיפו תכונות חדשות רבות, כפי שתגלה בספר זה.

קרוב לוודאי שהדבר הטוב ביותר בנוגע ל- Visual Web Developer 2008 הוא מחירה. היא זמינה בחינם. למרות שהגרסאות המסחריות של Visual Studio 2008 מגיעות עם Visual Web Developer, ביכולתן להוריד ולהתקין את Express Edition החינמית. קרוב לוודאי שהדבר הופך את Visual Web Developer 2008 ו- ASP.NET 3.5 לטכנולוגיות פיתוח האינטרנט המושכות ביותר והמשכנעות ביותר שזמינות כיום.

למי מיועד ספר זה

ספר זה נועד לכל מי שרוצה ללמוד כיצד לבנות אתרי אינטרנט עשירים ואינטראקטיביים שפועלים על בסיס פלטפורמת מיקרוסופט. הידע שתשיג מספר זה יהווה בסיס נהדר לבניית אתר אינטרנט מסוג כלשהו, החל מאתרים שקשורים לתחביב וכלה באתרים למטרות מסחריות.

גם מי שתכנות אינטרנט חדש עבורו, יוכל לעקוב אחר הספר, מכיוון שהספר אינו מניח רקע קודם בתכנות אתרי אינטרנט. הספר מתחיל ממש בצעדים הראשונים של תכנות אינטרנט על ידי כך שהוא מראה כיצד להשיג ולהתקין את Visual Web Developer. הפרקים העוקבים מציגים טכנולוגיות חדשות באופן הדרגתי, תוך הסתמכות על הידע שנצבר בפרקים הקודמים.

האם יש לך העדפה ברורה של Visual Basic על פני C#, או להפך? ואולי אתה סבור ששתי השפות "מגניבות"? או שמא לא גיבשת דעה וברצונך ללמוד את שתי השפות? כך או כך, הספר ימצא חן בעיניך מכיוון שכל דוגמאות הקוד מוצגות בשתי השפות!

גם אם הנך בן-בית בגרסאות קודמות של ASP.NET, בפרט גרסאות 1.x, תוכל ללמוד רבות מספר זה. למרות שרעיונות רבים הועברו מ- ASP.NET 2.0 אל ASP.NET 3.5, תגלה שאפשר למצוא בספר חומר חדש רב, כולל מבוא ל-LINQ, כלי הניפוי החדשים של CSS ו-JavaScript, פקדים חדשים של ASP.NET, תמיכה משולבת ב- ASP.NET Ajax, תכונות חדשות ב- SP1 ומבוא על Silverlight.

במה עוסק הספר

ספר זה ילמד אותך ליצור אתר אינטרנט אינטראקטיבי, מונע-נתונים ועשיר בתכונות. למרות שזו משימה כבדה למדי, תמצא שבועזרת Visual Web Developer 2008 היא אינה כה קשה. תעקוב אחר כל התהליך של בניית אתר אינטרנט, החל מהתקנת Visual Web Developer 2008 בפרק 1, וכלה בפריסת יישום האינטרנט בשרת production בפרק 18. הספר מחולק ל-18 פרקים שכל אחד מהם מוקדש לנושא מסוים ו-4 נספחים.

פרק 1 - ASP.NET 3.5 מתחילים

בפרק זה תלמד כיצד להשיג ולהתקין את Visual Web Developer 2008. תקבל הוראות להורדת והתקנת גרסת החינם של Visual Web Developer 2008 הקרויה Express Edition. תערוך גם היכרות עם HTML, השפה שמאחורי כל דף אינטרנט. הפרק מסתיים בסקירה של אפשרויות ההתאמה האישית שמציעה Visual Web Developer.

פרק 2 - בניית אתר אינטרנט בסביבת ASP.NET

פרק זה מראה כיצד ליצור אתר אינטרנט חדש, וכיצד להוסיף לו אלמנטים חדשים כמו דפים. בנוסף ללימוד נושא יצירת אתר בנוי היטב, תלמד גם כיצד להשתמש בכלים הרכיבים של Visual Web Developer ליצירת דפי HTML ו-ASP.NET.

פרק 3 - עיצוב דפי האינטרנט שלך

Visual Web Developer מגיעה עם כלים רבים שמאפשרים ליצור דפי אינטרנט מעוצבים היטב ומושכים. בפרק זה תלמד כיצד להשתמש בכלים אלה בצורה חכמה. בנוסף, תלמד על CSS, השפה שמשמשת לעיצוב דפי אינטרנט.

פרק 4 - עבודה עם פקדי ASP.NET

פקדי שרת של ASP.NET הם אחד המושגים החשובים ביותר ב-ASP.NET. הם מאפשרים ליצור אתרי אינטרנט מורכבים ועשירי תכונות, על ידי קוד מועט ביותר. בפרק תערוך היכרות עם פקדי שרת זמינים רבים, ותלמד למה הם משמשים וכיצד להשתמש בהם.

פרק 5 - תכנות דפי Web ב-ASP.NET

למרות שכלי CSS המובנים ופקדי השרת של ASP.NET יכולים לקדם אותך מאוד ביצירת דפי אינטרנט, סביר להניח שתשתמש בשפת תכנות כדי לשפר את הדפים. פרק זה משרת כמבוא לתכנות, עם דגש חזק על תכנות דפי אינטרנט. כל הדוגמאות שתראה בפרק זה (וביתרת הספר) הן בשתי השפות, Visual Basic ו-C#, ותוכל לבחור את המועדפת עליך.

פרק 6 - יצירת אתרי web בעלי מראה עקבי

עקביות היא גורם חשוב בהענקת מראה מקצועי ומושך לאתר אינטרנט. ASP.NET מסייעת ליצירת דפים בעלי מראה עקבי באמצעות השימוש בדפים ראשיים, אשר מאפשרים להגדיר את המראה והתחושה הגלובליים של דף. ערכות נושא ו-skins עוזרים למרכז המראה של פקדים ואלמנטים חזותיים אחרים באתר. תלמד גם ליצור דף בסיס שעוזר למרכז קוד שמשמשים בו בכל הדפים באתר.

פרק 7 - ניווט באתר

ASP.NET מגיעה עם כמה פקדי ניווט שתפקידם לעזור למבקרים למצוא את דרכם באתר. פקדים אלה משמשים לבניית מבנה הניווט של האתר. הם יכולים להיות מחוברים למפה המרכזית של האתר, שמגדירה את הדפים באתר האינטרנט. תלמד גם כיצד לשלוח משתמשים מדף אחד לאחר על ידי תכנות.

פרק 8 - פקדי משתמש

פקדי משתמש הם קטעי דפים שניתן להשתמש בהם שימוש חוזר בדפים רבים באתר האינטרנט. ככאלה הם מעולים לתכנים חוזרים כמו תפריטים, כרזות פרסום וכו'. בפרק זה תלמד כיצד ליצור ולהשתמש בפקדי משתמש, ולשפר אותם עם אינטליגנציה תכנותית.

⊙ פרק 9 - בדיקת אימות הקלט מהמשתמש

חלק ניכר מהאינטראקטיביות באתר נקבע על ידי הקלט מהמשתמשים. הפרק מראה כיצד לקבל קלט מהמשתמשים, לבדוק את תקפותו, ולעבד אותו תוך שימוש בפקדי שרת של ASP.NET. בנוסף, תלמד כיצד לשלוח דואר אלקטרוני מיישום אינטרנט מבוסס ASP.NET, וכיצד לקרוא מקבצי טקסט.

⊙ פרק 10 - ASP.NET Ajax

ASP.NET Ajax של מיקרוסופט מאפשר ליצור דפי אינטרנט בעלי מראה נאה וחופשיים מהבהוב, שסוגרים את הפער בין יישומי שולחן העבודה המסורתיים לבין יישומי אינטרנט. בפרק זה תלמד כיצד להשתמש בתכונות המובנות של Ajax כדי לשפר את נוכחות דפי האינטרנט שלך. תוצאת הדבר היא אינטראקציה חלקה יותר עם האתר.

⊙ פרק 11 - מבוא לבסיסי נתונים

הבנת השימוש בבסיסי נתונים חיונית לבניית אתרי אינטרנט מודרניים, מכיוון שרוב אתרי האינטרנט המודרניים משתמשים בבסיס נתונים. תלמד את יסודות SQL, שפת השאילתות שמאפשרת גישה ושינוי הנתונים בבסיס נתונים. בנוסף, תכיר את כלי בסיס הנתונים שנמצאים ב- Visual Web Developer ומסייעים ליצירת וניהול בסיסי נתונים מסוג SQL Server.

⊙ פרק 12 - הצגה ועדכון של נתונים

פרק זה מתבסס על הידע שרכשת בפרק הקודם. הפרק מראה כיצד להשתמש בפקדים קשורי-נתונים ובפקדי מקור נתונים של ASP.NET, כדי ליצור ממשק עשיר שיאפשר למשתמשים אינטראקציה עם הנתונים בבסיס הנתונים שפקדים אלה מכוונים אליהם.

⊙ פרק 13 - LINQ

LINQ הוא הפתרון החדש של מיקרוסופט לגישה לאובייקטים, בסיסי נתונים, XML ועוד. בפרק זה תלמד כיצד להשתמש ב- LINQ to SQL כדי לגשת לבסיסי נתונים מסוג SQL Server. במקום לכתוב קוד רב באופן ידני, תיצור קבוצת אובייקטים של LINQ שיעשו עבורך את "העבודה השחורה". פרק זה מסביר מהי LINQ, כיצד להשתמש במעצב החזותי של LINQ, המובנה ב- Visual Web Developer, וכיצד לכתוב שאילתות LINQ כדי לקבל נתונים מבסיס נתונים SQL Server או לכתוב אליו.

⊙ פרק 14 - הצגת נתונים - נושאים מתקדמים

בעוד שהפרקים הקודמים התמקדו בעיקר ביסודות הטכניים של פעילות עם נתונים, פרק זה מתבונן באותו נושא מנקודת המבט של ממשק המשתמש. תלמד כיצד לשנות את המראה החזותי של הנתונים באמצעות שימוש בסגנונות שליטה. תלמד גם ליצור אינטראקציה עם פקדים קשורי-נתונים, וכיצד להאיץ את היישום על ידי שמירת עותק מקומי של נתונים שניגשים אליהם בתדירות גבוהה.

פרק 15 - אבטחת אתר אינטרנט בסביבת ASP.NET 3.5

אבטחה זה נושא בעל חשיבות מהמעלה הראשונה, למרות היותו מוצג מאוחר למדי בספר. פרק זה מלמד כיצד להשתמש בתכונות המובנות של ASP.NET שקשורות לאבטחה. תלמד על כמה שירותי יישום שמסייעים לקידום האבטחה. תלמד גם כיצד לאפשר למשתמשים להירשם לקבלת חשבון באתר, כיצד להבדיל בין משתמשים אנונימיים למשתמשים מחוברים, וכיצד לנהל את המשתמשים במערכת.

פרק 16 - התאמה אישית של אתרי אינטרנט

פרק זה מראה כיצד ליצור דפי אינטרנט מותאמים אישית בעלי תוכן שמכוון למשתמשים אינדיבידואליים, זאת תוך שהוא מתבסס על תכונות האבטחה שהוצגו בפרק 15. תלמד כיצד להגדיר ולהשתמש ב-Profile של ASP.NET שמאפשר לאחסן נתונים מותאמים אישית על מבקרים ידועים ואנונימיים.

פרק 17 - טיפול בחריגים, ניפוי ומעקב

כדי להבין, לשפר ולתקן את הקוד שכתבת עבור דפי אינטרנט בסביבת ASP.NET, יש צורך בכלי ניפוי טובים. Visual Web Developer מגיעה עם תמיכה מעולה בניפוי, שמאפשרת לאבחן את מצב היישום בזמן ביצוע, ומסייעת במציאה ותיקון של בעיות לפני שהמשתמשים נתקלים בהן.

פרק 18 - פריסת אתר אינטרנט

כשתגיע לסיום הספר יהיה בידך אתר אינטרנט מוכן להיחשף לעולם. איך בדיוק תעשה זאת? איזה דברים עליך לדעת ולהבין לפני שתחשוף את האתר לציבור הרחב? פרק זה עונה על שאלות אלו, ומספק תובנות לגבי תצורת מערכות ייצור שונות שבהן עשוי אתר האינטרנט הסופי להתבצע.

נספח א' - תשובות לתרגילים

נספח ב' - קביעת התצורה של SQL Server 2005 (תקף גם לגרסה 2008).

נספח ג' - תכונות חדשות (SP1)

בנספח זה נציג בקצרה את התוספות שמיקרוסופט שחררה במסגרת העדכון NET 3.5 Service Pack 1. קבוצת תכונות זו מתפרסת על פני סביבת העבודה NET, ובכלל זה גם ASP.NET. נספח זה עוסק בעיקר בתכונות החדשות של ASP.NET.

נספח ד' - היכרות עם Silverlight

בנספח זה תערוך היכרות עם הטכנולוגיה החדשה של מיקרוסופט שמהווה תשתית cross-platform, cross-browser, המאפשרת למפתחים ומעצבים לייצר את הדור הבא של מערכות אינטרנט.

אינדקס

מבנה הספר

גישת הספר היא להסביר מושגים צעד אחר צעד, תוך שימוש בדוגמאות עבודה והסברים מפורטים. הסעיפים המפורסמים של Wrox, **נסה זאת וכיצד זה פועל**, מנחים אותך צעד אחר צעד בביצוע משימה, ומפרטים את הדברים החשובים תוך כדי התקדמות המשימה. אחרי כל משימת נסה זאת בא סעיף **כיצד זה פועל** שמסביר את הצעדים שבוצעו בתרגיל.

בסופו של כל פרק תמצא תרגילים שיעזרו לך לבדוק את הידע שהשגת בפרק זה. התשובות לכל השאלות נמצאות בנספח א' בסוף הספר. אל תדאג אם אינך יודע את התשובות לשאלות. פרקים מאוחרים יותר אינם מניחים שביצעת את משימות סעיף התרגילים בפרקים קודמים.

מה נדרש כדי להשתמש בספר

הספר מניח שהמערכת שלך עונה על הדרישות הבאות:

⊙ מסוגלת להפעיל את Visual Web Developer. לגבי הדרישות המדויקות של המערכת פנה לקובץ readme שמגיע עם התוכנה.

⊙ מערכת ההפעלה היא לפחות Windows XP Professional Edition, או Windows Vista (לפחות גרסת Home Premium), או אחת מגרסאות השרת של Windows.

את רוב התרגילים ניתן אמנם לבצע בסביבת Windows XP Home או Windows Vista Basic, אבל כמה מפרקי הספר דורשים שימוש ב-IIS, שרת האינטרנט של מיקרוסופט, שמגיע רק עם גרסאות Windows שמופיעות ברשימת הדרישות לעיל.

הפרק הראשון מראה כיצד להשיג ולהתקין את Visual Web Developer 2008, אשר בתורה מתקינה את NET Framework version 3.5 של מיקרוסופט ואת SQL Server 2005 Express Edition. כל מה שאתה זקוק לו זו מערכת הפעלה טובה ודחף לקרוא את הספר! קרא עוד על התוכנות הנדרשות בפרק 1.

מוסכמות

כדי לעזור לך להפיק את המרב מהטקסט ולעקוב אחר מה שקורה, נשתמש לאורך הספר בכמה מוסכמות.

מוסכמות נסה זאת

נסה זאת הוא תרגיל שעליך לבצע בעקבות הטקסט בספר.

1. התרגיל מורכב מקבוצת צעדים.
2. כל צעד ממוספר.
3. עקוב אחר הצעדים עם עותק הקוד שבידך.
4. אחר כך קרא את הסעיף **כיצד זה פועל** כדי להבין מה קורה.

כיצד זה פועל

לאחר כל נסה זאת יוסברו הפעולות שביצעת והקוד שהקלדת לפרטיהן.

הערה: עצות, רמזים והערות נמצאים בפונט שונה כמו זה.

סגנונות בטקסט:

- ⊙ מונחים חדשים ומילים חשובות מופיעים בגופן עבה כשהם מוצגים לראשונה.
- ⊙ לציון פריטי תפריט שדורשים לחיצה על תת-תפריט משמש הסימן ←. לדוגמה, File ← New ← Folder.
- ⊙ קוד או תוכן שאינם רלוונטיים לדיון מושמטים לחלוטין או מוחלפים בשלוש נקודות עוקבות, כך:

```
<tr>
  <td style="white-space: nowrap;">
    ... Menu items go here; not shown
  </td>
</tr>
```

אנו נשתמש בשלוש הנקודות מבלי להתחשב בשפת התכנות של הדוגמה, לפיכך תראה אותן עבור C#, Visual Basic, HTML, CSS ו-JavaScript. כשהן מופיעות בקוד שעליך להקליד בעורך הקוד, דלג על שלוש הנקודות ועל כל מה שבא בעקבותיהן באותה שורה.

- ⊙ קוד שמוצג בפעם הראשונה, או קוד רלוונטי אחר, יופיע בצורה הבאה:

```
Dim albumOwner As String
albumOwner = "Imar"
lblOwner.Text = albumOwner
```

בניגוד לכך, קוד פחות רלוונטי, או קוד שכבר ראית קודם, ייראה כך:

```
albumOwner = "Imar"
```

שים לב, רצוי בכל מקרה להשתמש בקוד שתוריד מאתר האינטרנט ולא להקליד.

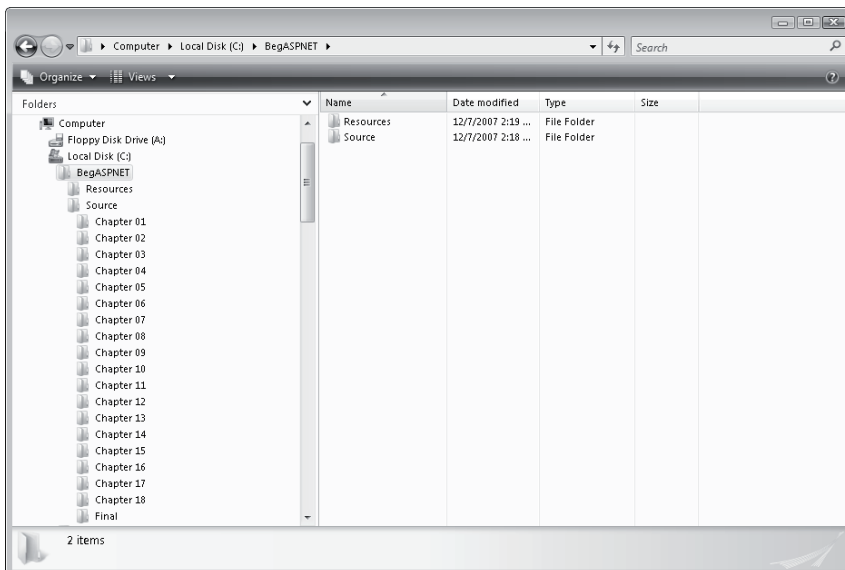
- ⊙ טקסט שמופיע על מסך ייראה לעתים קרובות כך: Each Word Start With A Capital Letter, גם אם טקסט המסך המקורי משתמש באופן שונה באותיות רישיות. הדבר נועד להבליט את טקסט המסך משאר הטקסט.

קוד מקור

כשתבצע את הדוגמאות בספר תוכל לבחור בין הקלדת כל הקוד באופן ידני (לא מומלץ) לבין שימוש בקבצי קוד המקור שנלווים לספר. כל קוד המקור שמשתמשים בו בספר זמין להורדה מ- www.wrox.com. כדי להשיג את כל קוד המקור עבור הספר, היכנס לאתר, אתר את כותרת הספר - Beginning ASP.NET 3.5 In C# and VB (על ידי שימוש בתיבה Search או באחת מרשימות הכותרות), ולחץ על הקישור Download Code בדף פרטי הספר. מכיוון שלספרים רבים יש כותרות דומות, ייתכן שהדרך הקלה ביותר היא לחפש לפי ה-ISBN. ה-ISBN של ספר זה הוא 978-0-470-18759-3.

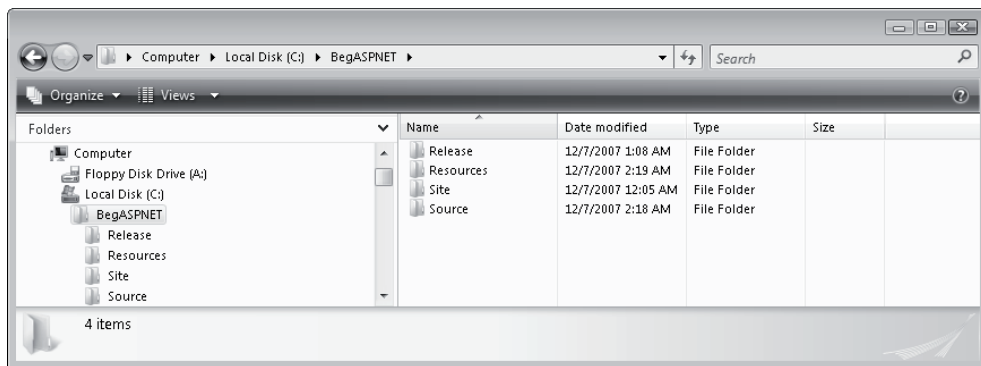
הערה: תוכל להוריד את הקבצים האלה גם מאתר **הוצאת הוד-עמי**. מצא את דף הספר ולחץ להורדה על "קוד מקור C#" או "קוד מקור VB". המשך לפי הפסקאות הבאות.

באפשרותך להוריד את כל המקור עבור ספר זה כקובץ יחיד (VB או C#) שאותו תוכל לפרוס בעזרת כלי הפריסה המועדף עליך. כשתשלוף את המקור עליך לוודא שיישמר המבנה המקורי של התיקיה שהינה חלק מהורדת הקוד. כלי הפריסה השונים משתמשים בשמות שונים לתכונה זו, אבל חפש תכונה כמו User Folder Names או Maintain Directory Structure. ברגע ששלפת את הקבצים מהקוד שהורד, אתה אמור לסיים עם תיקייה בשם Source ותיקיה בשם Resources. אחר כך צור תיקייה חדשה בשם BegASPNET בשורש כונן C שלך, והעבר את התיקיות Source ו-Resources אל התיקיה החדשה. אתה אמור לסיים עם תיקיות **C:\BegASPNET\Source** ו-**C:\BegASPNET\Resources**. תיקיית המקור מכילה את המקור עבור כל אחד מ-18 פרקי הספר, ואת הגרסה הסופית של יישום PlanetWrox שאיתה תפעל לאורך הספר. התיקיה Resources מכילה קבצים שתזדקק להם בכמה מתרגילי הספר. אם הכל ילך כשורה, אתה אמור לסיים עם המבנה שמוצג בתרשים ה-1.



תרשים ה-1

בפרקים מאוחרים יותר תיצור בתוך התיקייה C:\BegASPNET תיקיות בשם Site ו-Release. מבנה התיקיות יהיה אז דומה לזה שבתרשים ה-2.



תרשים ה-2

התיקייה Site מכילה את האתר כפי שייבנה במהלך הספר. התיקייה Release תכיל את הגרסה הסופית בסיום הספר. בכל פעם ש"תיתקע" באחת מדוגמאות הספר, תוכל להעיף מבט בתיקייה Source כדי לראות כיצד היו הדברים אמורים להיראות.

כשתרצה להפעיל את האתר עבור פרק מסוים כדי לראות כיצד הוא פועל, עליך לפתוח את תיקיית הפרק ב- Visual Web Developer כאתר אינטרנט. לפיכך תפתח, לדוגמה, את C:\BegASPNET\Source\Chapter13 ולא את תיקיית האב C:\BegASPNET\Source.

שים לב: יכול להיות שמראה האתר השתנה במקצת מזה המוצג בספר. דבר זה אינו צריך להפריע לך בעבודה ובהבנה.

היצמדות למבנה זה מבטיחה ביצוע חלק של תרגילי נסה זאת בספר. ערכוב או מיקום לא נכון של תיקיות אלו יקשה על ביצוע התרגילים, ועלול אף לגרום למצבים בלתי-צפויים ולשגיאות. בכל פעם שתיתקל בבעיה או שגיאה שאינן מוסברות בספר, ודא שמבנה האתר עדיין קרוב לזה שמוצג כאן.

טעויות דפוס או קוד

נעשה כל מאמץ כדי לוודא שהטקסט והקוד נקיים משגיאות, עם זאת, אף אחד אינו מושלם, ושגיאות אכן קורות. אם תמצא בספר שגיאה, כמו שגיאת איות או קטע קוד שגוי, שלח אלינו ונעדכן באתר. דיווח על טעויות דפוס יכול לחסוך לקורא אחר שעות של תסכול, ובו-בזמן יעזור לספק מידע איכותי עוד יותר. שלח מייל ל- info@hod-ami.co.il ובנושא כתוב 59447.

דף שגיאות Errata תוכל למצוא באתר של הספר באנגלית www.wrox.com. בתרגום לעברית בספר זה השתדלנו להכניס את כל התיקונים שהיו באתר בזמן התרגום.

הוצאת הוד-עמי תוכל לסייע לך בהורדת הקבצים מהאתר, או בכל תקלה שתמצא בספר. הוצאת הוד-עמי אינה נותנת שירותי סיוע בתכנות או בהכנת הכתוב. לשם כך פנה אל יועץ תכנות.

כדי להשתתף בדיונים עם המחבר ועמיתים, הצטרף לפורומים של P2P ב- p2p.wrox.com. הפורומים הם מערכת מבוססת אינטרנט שבה תוכל לפרסם הודעות בנושא ספרי Wrox וטכנולוגיות קשורות, וליצור קשר עם קוראים אחרים ומשתמשי הטכנולוגיות. ניתן להיות מנוי בפורומים ולקבל הודעות דואר אלקטרוני בתחומי העניין שלך בכל פעם שנוספים לפורומים פרסומים חדשים.

ב- p2p.wrox.com תמצא כמה פורומים שיעזרו לך לא רק במהלך קריאת הספר, אלא גם בפיתוח היישומים שלך. כדי להצטרף לפורומים עקוב אחר הצעדים הבאים:

1. היכנס ל- p2p.wrox.com ולחץ על הקישור Register Now.
2. קרא את תנאי השימוש ולחץ Agree.
3. השלם את המידע הנדרש כדי להצטרף, וכל מידע בחירה אחר שברצונך לספק, ולחץ Submit.
4. תקבל הודעת דואר אלקטרוני עם מידע שמתאר כיצד לאמת את החשבון שלך, והשלם את תהליך ההצטרפות.

אפשר לקרוא הודעות מהפורומים מבלי להצטרף ל-P2P, אבל כדי לפרסם הודעות משלך אתה חייב להצטרף.

לאחר ההצטרפות תוכל לפרסם הודעות חדשות ולהגיב על הודעות שפרסמו משתמשים אחרים. את הפורום שעוסק בספר זה (באנגלית) תמצא תחת הקטגוריה Books שזמינה מדף הבית על ידי לחיצת View All Forums בתפריט שמשמאל. תוכל לקרוא הודעות ברשת בכל שעה. אם ברצונך שהודעות חדשות שנוגעות לפורום מסוים תישלחנה אליך בדואר אלקטרוני, לחץ על הסמל Subscribe to this Forum שליד שם הפורום ברשימת הפורומים.

למידע נוסף על השימוש ב- Wrox P2P קרא את P2P FAQs, שמכיל תשובות לשאלות לגבי פעולת תוכנת הפורומים ולשאלות נפוצות רבות שייחודיות ל-P2P ולספרי Wrox. כדי לקרוא את ה-FAQs לחץ על הקישור FAQ בדף כלשהו של P2P.

SP1

נספח ג' נותן היכרות עם התכונות של Service Pack 1.

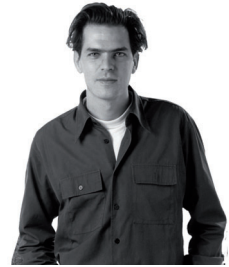
מידע נוסף תוכל למצוא בספרון: **New Feature in ASP.NET 3.5 Service Pack 1** של הוצאת הספרים WROX. תוכל למצוא אותו באתר של WROX בכתובת:

<http://www.wrox.com/WileyCDA/WroxTitle/New-Features-in-ASP-NET-3-5-Service-Pack-1.productCd-0470457341.html>

על המחבר

Imar Spaanjaars סיים את לימודיו האקדמיים בתחום ניהול הזמן הפנוי (leisure Management) בהולנד, אך במהרה החליף את תחום התעניינותו ועיסוקו לתחום האינטרנט.

הוא עבד בחברה גדולה וכיום הוא מועסק בחברה Design IT בהולנד, שבה הוא התמחה באינטרנט וביישומים הנלווים שנבנים בטכנולוגיית התוכנה של מיקרוסופט, כמו ASP.NET 3.5. בתפקידו כדירקטור טכני ומעצב תוכנה הוא אחראי להקמת מערכות בינוניות וגדולות.



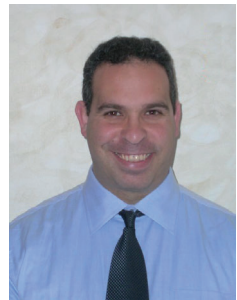
בתוקף מעמדו המקצועי הבכיר הוא מנחה את Dynamicweb Nederland שהינה סניף של החברה בשם זה בדנמרק.

Imar כתב ספר על ASP.NET 2.0 ועל Dreamweaver, שניהם יצאו לאור על ידי Wrox. הוא אחד התורמים הבכירים לפורום הידע p2p.wrox.com. ניתן להתקשר איתו דרך האתר האישי <http://imar.spaanjaars.com>.

על העורך המקצועי

היועץ המקצועי מאיר קרודו עובד בחברת Matrix, ומשמש כראש צוות פיתוח ומוביל טכנולוגי בבנק לאומי. מאיר עוסק בפיתוח והטמעה של יישומיים בנקאיים מתקדמים בסביבת מחשוב עם תשתיות הפיתוח של מיקרוסופט, ומשמש יועץ מקצועי לחברות בטכנולוגיות מיקרוסופט חדשניות, כגול ASP.NET, SQL Server ו-Silverlight.

מאיר משמש כיועץ מקצועי לספרים נוספים בהוד-עמי, וביניהם "Visual C# 3.0 סדנת לימוד".



על מחבר נספח ד'

אלכס גולש הינו מומחה בעל שם ב-Silverlight ו-WPF, בעל תואר MVP של מיקרוסופט. אלכס יועץ בחברות שונות בארץ ובעולם, מפתח ומטמיע RIA ו-Smart Client based solution, מדריך ומנחה מאות מומחי IT בארץ ובעולם. אלכס מרצה מבוקש בנושאים Silverlight, WPF ו-XNA. בבלוג שלו http://blogs.microsoft.co.il/blogs/alex_golesh/ הוא יועץ לכותבים בנושא פיתוח ב-Silverlight ונושאים אחרים.

פרק 1

ASP.NET 3.5 מתחילים

הגרסה הראשונה של .NET Framework 1.0. שוחררה לשימוש בשנת 2002, מאז השקיעה מיקרוסופט מאמצים רבים וזמן פיתוח רב ב-ASP.NET, החלק של Net Framework שמאפשר לבנות יישומי Web איכותיים בעלי ממשק משתמש עשיר. הגרסה הראשונה הייתה שינוי קיצוני לעומת הטכנולוגיה הישנה של מיקרוסופט לבניית אתרי Web שנקראה **ASP** (Active Server Pages), ונקראת כיום לעתים קרובות **ASP הקלאסית**. הכנסה לשימוש של **ASP.NET 1.0** בשילוב סביבת הפיתוח **Visual Studio.NET 2002**, נתנו למפתחים את התועלות הבאות לעומת ASP הקלאסית:

- ⊙ הפרדה ברורה בין תצוגה וקוד. ב-ASP הקלאסית לוגיקת הקוד הייתה 'מפוזרת' יחד עם תגיות ה-HTML של הדף והדבר הקשה מאוחר יותר על הכנסת שינויים לדף.
 - ⊙ מודל פיתוח שהיה קרוב הרבה יותר למודל בו השתמשו כדי לבנות יישומי שולחן העבודה. דבר זה הקל על מתכנתי Visual Basic לעבור ליישומי Web.
 - ⊙ סביבת עבודה עשירה (Visual Studio.NET) המאפשרת למפתחים ליצור ולקודד את יישומי Web שלהם באופן חזותי.
 - ⊙ אפשרות בחירה בין כמה שפות תכנות מוכוון-עצמים שהפופולריות ביניהן כיום הן Visual Basic ו-Visual C# (מבטאים C-Sharp).
 - ⊙ גישה לספריית .Net Framework. משמעות הדבר הייתה שבפעם הראשונה ניתנה למפתחי Web דרך גישה קלה ואחידה דרך שימוש במחלקות רבות ומגוונות המאפשרות עבודה עם בסיסי נתונים, קבצים, דואר אלקטרוני, רכיבי תקשורת ועוד.
- למרות היתרונות הרבים של ASP.NET על פני המודל הישן, הפעלת סביבת עבודה זו פירושו גם גידול במורכבות ובידע הדרושים לבניית יישומים באמצעותה, והדבר הקשה על מתכנתים חדשים רבים להתחיל לעבוד עם ASP.NET.

לאחר השחרור הראשוני ב-2002, שחררה מיקרוסופט לשוק גרסה אחרת של .Net Framework (קרויה .NET 1.1), וב-2003 את כלי הפיתוח Visual Studio.NET. רבים ראו זאת כחבילת שירות לגרסה הראשונית, למרות שהיא כללה שיפורים חדשים רבים, הן לסביבת העבודה והן לכלי הפיתוח.

בנובמבר 2005 שוחררו לשוק Visual Studio 2005 ו- ASP.NET 2.0. להפתעתם הנעימה של מפתחים רבים בעולם הצליחה מיקרוסופט פעם נוספת להרחיב את המוצר ולשפרו באופן משמעותי, תוך הוספת תכונות וכלים רבים שעזרו להקטין את המורכבות שהוכנסה עם ASP.NET 1.0. אשפים חדשים ופקדים חכמים אפשרו להפחית את הקוד הנדרש לבניית יישום, להקטין את עקומת הלמידה של מפתחים חדשים ולהגדיל את היעילות.

הגרסה הנוכחית, **ASP.NET 3.5**, בנויה על בסיס הגרסה המצליחה **ASP.NET 2.0**, תוך שהיא משאירה תכונות רבות, ומוסיפה תכונות וכלים חדשים בתחומים אחרים.

ב-18 הפרקים הבאים תלמד לבנות אתר Web שלם בסביבת ASP.NET, תוך שימוש ב- Visual Web Developer, כלי הפיתוח של מיקרוסופט ליישומי Web מבוססי ASP.NET. ספר זה ינחה אותך בתהליך היצירה של אתר Web בעל פונקציונליות מלאה המונע על ידי בסיסי נתונים. נתחיל בפרק זה עם אתר Web בסיסי, ונגיע בסוף הדרך, בפרק 18, לפריסת סביבת הייצור שלו.

בפרק זה נדון בנושאים הבאים:

⊙ סביבות הפיתוח Visual Web Developer 2008 Express Edition ו- Visual Studio 2008 – היכן להשיג וכיצד להתקין.

⊙ יצירת אתר Web הראשון שלך על ידי Visual Web Developer.

⊙ נבחן את הדרך בה דף של ASP.NET מעובד ונשלח לדפדפן.

⊙ השימוש בסביבת הפיתוח והתאמתה לצרכיך.

הפרק מסתיים בסקירה של דוגמת האתר Planet Wrox אשר ילווה אותנו לאורך הספר. בפרק זה תראה מה יש לאתר להציע, ואיך להשתמש בו; שאר הספר יראה את דרכי הפעולה הפנימית של האתר וכיצד הוא בנוי.

האתר לדוגמה, וכל הדוגמאות בספר זה, בנויים על ידי **VWD** (Visual Web Developer), לכן חשוב שהמוצר יהיה מותקן במחשב הפיתוח שלך ושתדע איך לגשת לתכונות הבסיסיות ביותר שלו. הסעיף הבא מראה איך להשיג ולהתקין VWD. ברגע שהמוצר יהיה מותקן ופעיל תלמד כיצד ליצור את אתר Web הראשון שלך, ולאחר מכן נסקור את התכונות הרבות של VWD.

כלי הפיתוח Visual Web Developer של מיקרוסופט

למרות שבאופן תיאורטי אתה יכול לכתוב יישומי Web בסביבת ASP.NET על ידי שימוש ב-Notepad או בעורך תמלילים אחר, הרי אין ספק שתמצא את התקין עותק של Visual Web Developer של מיקרוסופט. Visual Web Develop, או בשמה המקוצר: VWD, פותחה במיוחד למטרת בניית אתרי Web בסביבת ASP.NET וככזו היא מכילה כמות עצומה של כלים אשר יעזרו לך ליצור במהירות יישומי Web מורכבים בסביבה זו.

Visual Web Developer מגיעה בשני אופנים: כגרסת חינום עצמאית הקרויה Microsoft Visual Web Developer 2008 Express Edition, וכחלק מחבילת הפיתוח הגדולה יותר הקרויה Visual Studio 2008, שהיא עצמה זמינה במהדורות שונות, כל אחת בעלת תג מחיר משלה. למרות

ש- Express Edition של VWD היא גרסת חינוך, נכללים בה כל העוזרים והכלים הנדרשים ליצירת יישומי Web מורכבים. את כל הדוגמאות שתמצא בספר ניתן לבנות עם גרסת החינוך של Express Edition, כך שכדי לעקוב אחר הספר אין צורך להשקיע כסף ברכישת כלי הפיתוח המסחרי של Visual Studio 2008. קל להשיג את VWD. אתה יכול להוריד את קובץ ההתקנה מאתר מיקרוסופט כפי שיוסבר להלן.

הורדה של Visual Web Developer

את גרסת החינוך של VWD תוכל להשיג באתר מיקרוסופט בכתובת הבאה: <http://www.microsoft.com/express/vwd/Default.aspx>. מדף זה תוכל להוריד את Visual Web Developer 2008 Express Edition כ- Web Install. כלומר, אתה מוריד רק את תוכנת ההתקנה, בעוד ששאר הקבצים יורדים במשך ההתקנה. ודא שאתה בוחר Visual Web Developer בדף ולא אחד ממוצרי Express חינוך אחרים. האתר גם מאפשר להוריד בנוחיות את כל מוצרי Express כקובץ ISO שתוכל לצרוב על DVD. גודל הקובץ שתוריד הוא מעט פחות מ-3MB. זהו רק קובץ ההתקנה שבהפעלתו יורדים הקבצים הנדרשים דרך האינטרנט. סך כל הקבצים שיורדים הוא כ-1.3GB.

אם תרצה לנסות את הגרסה המלאה של Visual Studio 2008, שכוללת גם את VWD, באפשרותך לחתום על תקופת ניסיון בחינם בת 90 ימים, ולהורידה מאתר מיקרוסופט <http://msdn2.microsoft.com/vstudio>. תוכל לבחור להוריד קובץ ISO שניתן לצריבה על DVD. הערה: נכון לכתיבת שורות אלו, סביבת הפיתוח הקיימת באתר כוללת את חבילת השירות (SP 1).

התקנת Visual Web Developer Express Edition

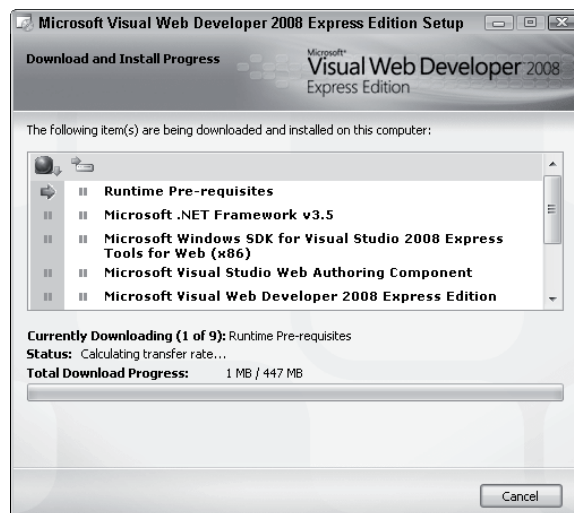
התקנת Visual Web Developer היא תהליך ישיר, אם כי ארוך במידת מה. התקנת VWD עשויה להימשך כמה שעות והדבר תלוי בשיטת ההתקנה שבחרת, במחשב שלך ובמהירות ההעברה ברשת האינטרנט.

נסה זאת התקנת Visual Web Developer 2008 Express Edition

תרגיל זה ינחה אותך בתהליך התקנת VWD Express Edition במחשב שלך. ההנחה היא שבחרת באפשרות התקנה מהאינטרנט, אם כי תהליך התקנת Express Edition מ-DVD כמעט זהה. הצעדים שעליך לבצע כדי להתקין את הגרסה המלאה של Visual Studio 2008 דומים גם הם, אם כי המסכים שתראה שונים במידת מה. ללא קשר לגרסת VWD שתתקין, חשוב שתתקין גם את SQL Server 2008 Express Edition - רכיב הכרחי, כאשר ברצונך לעקוב אחרי דוגמאות רבות בספר זה. כאשר אתה מתקין את הגרסה המלאה של Visual Studio 2008, האפשרות להתקין SQL Server כלולה ברשימת האפשרויות להתקנה שאתה רואה במהלך הגדרת ההתקנה. אם אתה מתקין את VWD Express Edition, תינתן לך האפשרות לבחור SQL Server בתיבת הדו-

שיח Installer Options. אם האפשרות SQL Server אינה מופיעה באותן תיבות דו-שיח אזי, ככל הנראה, SQL Server 2008 Express Edition כבר מותקן אצלך.

1. הפעל את קובץ ההתקנה שהורדת מאתר מיקרוסופט.
2. ברגע שההתקנה החלה, לחץ Next, קרא וקבל את תנאי הרישיון, לחץ Next פעם נוספת.
3. בדף Installer Options בחר Microsoft SQL Server 2008 Express Edition. אם אינך רואה את האפשרות SQL Server, המוצר כבר מותקן אצלך. לחץ Next שוב.
4. אם יש מספיק מקום בדיסק הראשי שלך, אזי בדף Destination Folder השאר את הגדרת ברירת המחדל של השדה Install in folder. אחרת, לחץ על הלחצן Browse ובחר במיקום אחר.
5. לחץ על הלחצן Install. אם אתה משתמש בתוכנת ההתקנה מהאינטרנט, אזי תחילה תטען תוכנית ההתקנה את הקבצים למחשב שלך דרך האינטרנט. במשך תהליך ההתקנה תראה מסך (דומה לתרשים 1-1) שציג את התקדמות ההורדה וההתקנה של VWD.
6. ברגע שתסתיים ההתקנה עשויה להופיע תיבת דו-שיח שתבקש ממך לאתחל מחדש את המחשב שלך. לחץ Restart. VWD מוכנה לשימוש ברגע שהמחשב עלה מחדש.



תרשים 1-1

כיצד זה פועל

תהליך ההתקנה הישיר הנחה אותך דרך ההגדרה של VWD Express Edition. בתיבת הדו-שיח Installer Options יכולת לבחור ב- Microsoft SQL Server 2008 Express Edition שהינה מנוע בסיסי הנתונים החינמי של מיקרוסופט.

החל מפרק 11 בספר נדון ונשתמש רבות ב- SQL Server 2005/2008. נספח ב' מראה כיצד להגדיר את תצורת האבטחה עבור גרסאות שונות של SQL Server 2008, תוך שימוש ב- SQL Server Management Studio Express Edition החינמי.

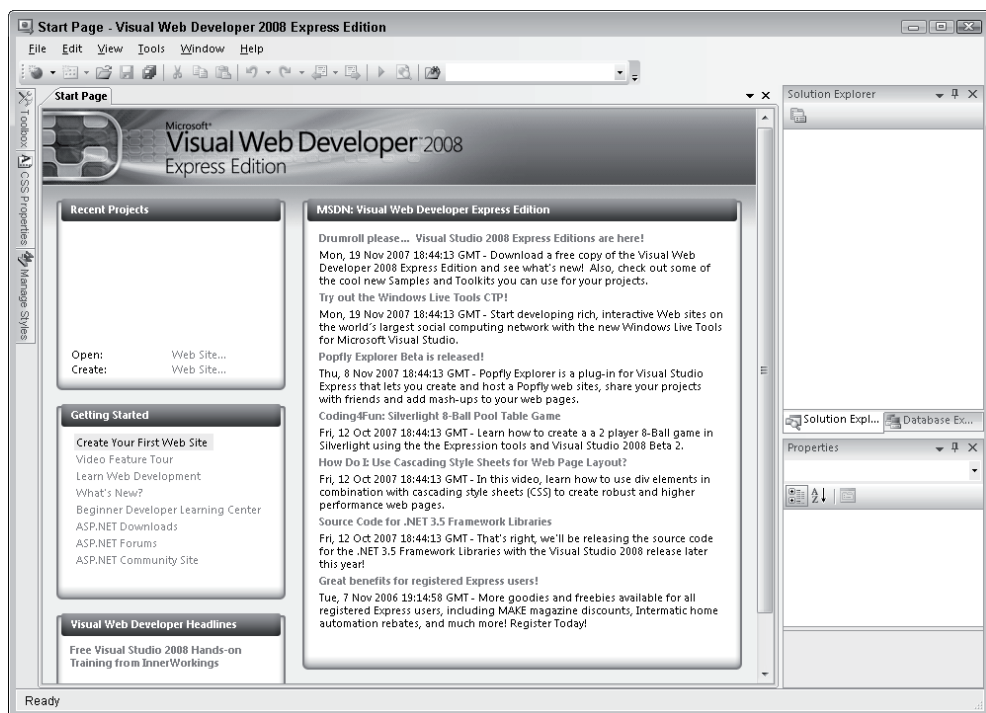
בסעיף הבא תלמד כיצד ליצור את האתר הראשון שלך בעזרת VWD. תראה איך ליצור אתר, איך להוסיף תוכן לדף Web ואיך לצפות בדף זה בדפדפן שלך.

יצירת אתר Web הראשון שלך בסביבת ASP.NET 3.5

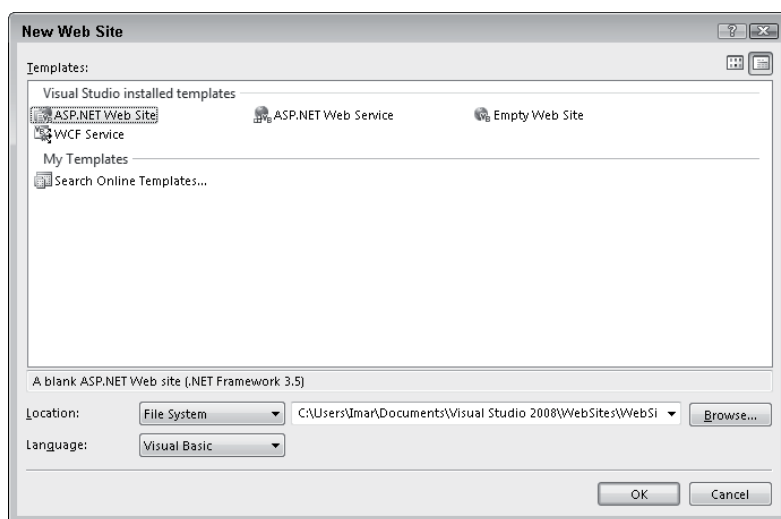
הגיע הזמן להתחיל בבניית אתר ה-Web הראשון שלך בסביבת ASP.NET, אז במקום לתת סקירה תיאורטית של אתרי אינטרנט ב-VWD, ניגש ישר לעניין, בתרגיל "נסה זאת" הבא נראה כיצד לבנות את פרויקט האינטרנט הראשון שלך. לאחר מכן, בסעיף ההסבר "כיצד זה פועל" ובסעיף שאחריו, תקבל תמונה טובה על מה שקורה מאחורי הקלעים כאשר אתה מתבונן בדף של ASP.NET בדפדפן שלך.

נסה זאת יצירת דף web ראשון בסביבת ASP.NET

1. הפעל את VWD מתפריט Windows Start. בהפעלה הראשונה של VWD קיימת השהיה ארוכה לפני שניתן להשתמש במוצר מפני ש-VWD עסוק בבניית התצורה שלו. ההפעלות הבאות של היישום תהיינה מהירות הרבה יותר.
 2. אם אתה משתמש בגרסה המסחרית של Visual Studio, בהפעלה הראשונה של המוצר תופיע תיבת דו-שיח שתאפשר לבחור מבין אוספים שונים של הגדרות. הבחירה שתעשה בתיבת דו-שיח זו משפיעה על עיצוב החלונות, ארגוני הכלים, התפריטים וקיצורי הדרך. בחר ב- Web Development Settings מכיוון שהגדרות אלו עוצבו במיוחד עבור מפתחים ב-ASP.NET. תוכל תמיד לבחור בפרופיל שונה על ידי קביעה מחדש של ההגדרות, כפי שיוסבר בהמשך הפרק.
 3. ברגע ש-VWD השלימה את בניית התצורה, יופיע המסך הראשי כמו בתרשים 1-2.
 - בסעיף הבא ניתן תיאור מלא של החלונות, סרגלי הכלים והתפריטים. כרגע נתמקד ביצירת אתר web חדש. לחץ על התפריט File בפינה השמאלית העליונה ובחר New Web Site. תיבת הדו-שיח New Web Site תופיע כפי שמוצג בתרשים 1-3.
 4. בחלק Templates של תיבת הדו-שיח ודא שבחרת ASP.NET Web Site. ודא גם שברשימה הנפתחת Location נבחרה האפשרות File System. אם תרצה, תוכל לשנות את המיקום בדיסק שבו יאוחסן אתר ה-Web, על ידי לחיצה על הלחצן Browse, ובחירה במיקום אחר בדיסק הקשיח במחשב שלך. בשלב זה, המיקום שנבחר כברירת מחדל בהחלט מספק, ותוכל להשאירו כפי שהוא.
 5. באפשרותך לבחור את שפת התכנות העיקרית שבה תשתמש באתר שלך. הבחירה מתבצעת ברשימה הנפתחת Language.
- כל הדוגמאות בספר זה מוצגות הן ב- Visual Basic והן ב- C#, בחר בשפה המועדפת עליך.

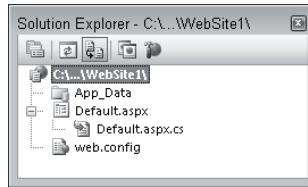


תרשים 1-2



תרשים 1-3

6. לחץ OK. VWD יוצרת עבורך אתר web חדש. האתר כולל: דף ASP.NET אחד בשם Default.aspx, קובץ הגדרות בשם web.config ותיקייה ריקה בשם App_Data, כפי שמוצג בתרשים 1-4. כמו כן, VWD פותחת את הקובץ Default.aspx כדי שתוכל לראות את קוד הדף.



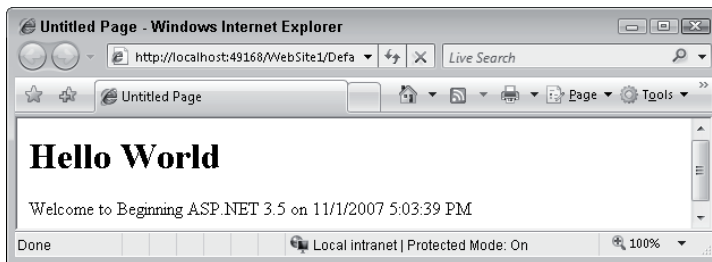
תרשים 1-4

7. בין תגיות הפתיחה והסיום <div> בדף, הקש את הטקסט והקוד הבאים:

```
<div>
  <h1>Hello World</h1>
  <p>Welcome to Beginning ASP.NET 3.5 on <%= DateTime.Now.ToString() %></p>
</div>
```

- ⊙ קוד בתבנית כזו יופיע פעמים רבות בספר. כאשר תקבל הוראה להקליד קוד זה, עליך להקליד את הקוד עם הרקע האפור בלבד. שאר הקוד כבר נמצא בקובץ.
- ⊙ כאשר תראה קוד כזה תוך כדי דיון - למשל בסעיף "כיצד זה פועל" - החלק שעליך להתמקד בו הוא הקוד עם הרקע האפור, הקוד ללא רקע הוא בעל חשיבות פחותה.
- ⊙ החלק עם הסוגריים הזוויתיים (<>) בקוד אינו צריך להטריד אותך; יותר מאוחר בספר תלמד כיצד הוא פועל. למרות שהקוד לעיל עשוי להיראות לך לא מוכר כרגע, קרוב לוודאי שאתה יכול לשער מה הוא עושה: הוא מדפיס את התאריך של היום ואת השעה.

8. כדי לפתוח את הדף בדפדפן ברירת המחדל שלך, כפי שמוצג בתרשים 1-5, עליך לבחור ב- Start Without Debugging (או הקש Ctrl+F5).

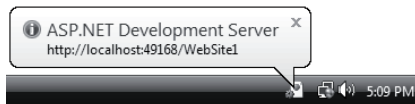


תרשים 1-5

אם התאריך והשעה אינם מופיעים בדף, או אם קיבלת שגיאה, הבט שנית בקוד של הודעת הפתיחה. הוא מתחיל בסוגר זוויתי (<) ולאחריו סימן אחוזים (%) וסימן שוויון (=). הקוד מסתיים בסוגר זוויתי אחר (>). ודא גם שהקלדת את הקוד בדיוק כפי שהוא מופיע כאן, כולל אותיות רישיות ("גדולות"). הדבר נכון במיוחד עבור C#, שהיא שפה רגישה לאותיות רישיות.

הערה: אם אתה מקבל סרגל Information שכולל אזהרה לגבי הגדרות Intranet ב- Internet Explorer, לחץ על הסרגל ובחר Enable Intranet Settings. אם ברצונך ללמוד קודם לכן על ההשלכות של הגדרות אלו, בחר What are Intranet Settings מתפריט הקיצור.

9. שים לב לסמל קטן עם המלל שהופיע בסרגל Windows (תרשים 6-1). סמל זה שייד ל- ASP.NET Development Server. שרת אינטרנט זה הופעל באופן אוטומטי על ידי VWD כדי לשרת את הבקשה לדף שלך. בשלבים מאוחרים יותר נלמד כיצד שרת זה מסוגל לעבד את הדף שלך.



תרשים 6-1

זהו. זה עתה יצרת את אתר האינטרנט הראשון שלך בסביבת ASP.NET 3.5.

כיצד זה פועל

אף על פי שדף האינטרנט שיצרת בתרגיל הקודם הוא פשוט למדי, התהליך שסופו היה הצגת הדף בדפדפן שלך אינו כה פשוט. לכשעצמו, דף ASP.NET שיצרת אינו יכול לעשות הרבה (יש שמתייחסים לדף כדף ASPX מכיוון שזו הסימון הפיזית של שם הקובץ). הדף צריך להיות מעובד על ידי שרת אינטרנט לפני שיועבר לדפדפן וזה מציג אותו. זו הסיבה שלצורך הטיפול בבקשה לדף הפעילה VWD באופן אוטומטי את ASP.NET Development Server המובנה. לאחר מכן הפעילה VWD את דפדפן ברירת המחדל והפנתה אותו לכתובת שרת האינטרנט: <http://localhost:49168/WebSite1>. שים לב שהמספר שכלול בכתובת עשוי להיות שונה בכל הפעלה של שרת האינטרנט, מכיוון שהוא נבחר אקראית על ידי VWD.

חשוב להבין שקובץ ASPX שיצרת באמצעות VWD אינו זהה לקובץ שבסופו של דבר מוצג על ידי הדפדפן. כאשר אתה יוצר דף ב-VWD אתה מוסיף לו סימן עריכה (markup). סימן העריכה בדף ASPX הוא צירוף כלשהו של טקסט רגיל, HTML, קוד עבור פקדי שרת של ASP.NET (שנלמד עליהם יותר בהמשך הפרק ובפרק 4), קוד שכתוב ב- Visual Basic.NET או C# ועוד.

כאשר אתה מבקש דף ASPX בדפדפן שלך, שרת האינטרנט מעבד את הדף, מבצע כל קוד שהוא מוצא בקובץ, והופך את סימני העריכה של ASP.NET ל-HTML רגיל שנשלח לדפדפן ומוצג על ידו. קוד HTML שנוצר בתרגיל נסה זאת הקודם גרם לדפדפן להציג את התאריך והזמן הנוכחיים. **HTML** (HyperText Markup Language) היא השפה שדפדפנים משתמשים בה כדי להציג דף web. יותר מאוחר בפרק תלמד איך HTML נראה ואיך להשתמש בו.

כדי לראות את השוני בין קוד HTML הסופי לבין דף ASPX המקורי, פתח את קוד המקור של הדף בדפדפן שלך. ברוב הדפדפנים תוכל להציג את חלון המקור על ידי לחיצה ימנית על הדף ובחירת View Source. פעולה זו מפעילה את עורך הטקסט שהוא ברירת המחדל שלך, שמציג את קוד HTML עבור הדף.

רוב קוד HTML שתראה בעורך הטקסט דומה לדף ASPX המקורי. אולם, אם תתבונן בשורה שמציגה את התאריך והשעה הנוכחיים תבחין בהבדל גדול. במקום הקוד בין הסוגריים הזוויתיים וסימני האחוזים, תראה כעת את התאריך והשעה הנוכחיים:

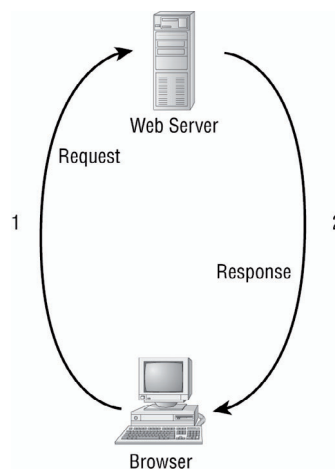
```
<h1>Hello World</h1>
<p>Welcome to Beginning ASP.NET 3.5 on 11/1/2007 5:03:39 PM</p>
</div>
```

לאחר ששרת האינטרנט עיבד את הדף, הוא בדק את התאריך והשעה הנוכחיים במחשב שלך והכניס אותם ל-HTML שנשלח לדפדפן.

בסעיף הבא תראה בפירוט רב יותר איך זה פועל.

היכרות עם ASP.NET 3.5

כאשר אתה מקליד כתובת אינטרנט דוגמת www.wrox.com ומקיש Enter, הדפדפן שולח בקשה לשרת האינטרנט באותה כתובת. הדבר מתבצע על ידי פרוטוקול HTTP (HyperText Transfer Protocol). פרוטוקול HTTP מאפשר לדפדפני ושרתי האינטרנט לתקשר ביניהם. כאשר אתה שולח את הכתובת, אתה שולח בקשה (request) לשרת. אם השרת פעיל והבקשה מאומתת, השרת מקבל את הבקשה, מעבד אותה ושולח את התשובה (response) אל דפדפן הלקוח. הקשר בין הבקשה לתשובה מוצג בתרשים 1-7.



תרשים 1-7

עבור קבצים סטטיים פשוטים, דוגמת קבצי HTML או תמונות, שרת האינטרנט קורא את הקובץ מהדיסק הקשיח המקומי שלו, ושולח אותו לדפדפן. אולם, עבור קבצים דינמיים, כמו דפי ASPX, הדבר אינו מספיק טוב. אם שרת האינטרנט היה שולח את קובץ ASPX ישירות לדפדפן כקובץ טקסט, לא היית רואה בדפדפן את התאריך והשעה הנוכחיים אלא את הקוד הנוכחי

(>% = DateTime.Now.ToString())<). לפיכך, במקום לשלוח את הקובץ ישירות, שרת האינטרנט מעביר את הבקשה לרכיב תוכנה אחר שמסוגל לעבד את הדף. הדבר מבוצע תוך שימוש במושג Application Mapping או Handler Mapping, כל רכיב תוכנה יודע לטפל בקבצים שונים לפי הסיומת של הקובץ. במקרה של דף aspx הבקשה מטופלת ומעובדת על ידי רכיב תוכנה בשם **ASP.NET runtime**. רכיב תוכנה זה הוא חלק מתשתית (.NET framework) של מיקרוסופט שתוכננה במיוחד כדי לטפל בבקשות web.

במהלך עיבוד הדף יש שלושה תחומים חשובים שמשפיעים על הדרך שבה בסופו של דבר יוצג הדף על ידי הדפדפן:

- ⊙ **טקסט סטטי.** כל טקסט סטטי שאתה מציב בדף, כמו HTML, CSS או קוד JavaScript, נשלח ישירות לדפדפן. תלמד יותר על HTML, CSS וקוד JavaScript בפרק זה ובפרקים הבאים, כולל פרק 3 אשר ייתן לך מבט מפורט על CSS.
- ⊙ **פקדי שרת של ASP.NET.** פקדים אלה נמצאים בדף ASPX שלך, וכאשר הם מעובדים, הם 'פולטים' HTML שהוכנס בדף. תלמד יותר על פקדי שרת (server controls) לאחר הדיון על HTML בפרק זה, ופרק 4 מוקדש כולו לפקדי שרת של ASP.NET.
- ⊙ **קוד תכנות.** באפשרותך לשבץ קוד, כמו Visual Basic.NET או C#, ישירות בדף, כפי שראית בתרגיל האחרון. בנוסף, תוכל להציב קוד בקובץ קוד נפרד שנקרא קובץ Code Behind. קוד זה יוכל להתבצע באופן אוטומטי על ידי רכיב הריצה (runtime), או כתוצאה מפעולה של המשתמש. כך או כך, ביצוע הקוד משפיע מאוד על הדרך שבה הדף מוצג בין אם נעשית גישה לבסיסי נתונים, ביצוע חישובים, הצגת או הסתרת פקדים מסוימים ועוד. תכנות דף האינטרנט שלך יידון בפירוט רב בפרק 5.

ברגע שהושלם עיבוד הדף וכל תגיות ה-HTML עבור הדף נאספו, הוא נשלח חזרה לדפדפן. הדפדפן קורא את קוד HTML, מבצע פעולה שנקראת parse, שבסימאה מוצג הדף בפניך. מכיוון ש-HTML הוא כה קריטי להצגת דפי web, בסעיף הבא תקבל סקירה כללית בנושא HTML.

הבנת HTML

השפה התקנית ליצירת דפי web היא HTML, ולכן היא מובנת על ידי כל הדפדפנים שקיימים כיום בשוק. מאז תחילת שנות ה-90 הייתה HTML הכוח המניע של ה-World Wide Web, אותו חלק של האינטרנט שעוסק בדפי web. מסמכי HTML הם קבצי טקסט פשוטים שמכילים סימני עריכה (markup), טקסט ונתונים נוספים שמשפיעים על הטקסט.

אלמנטים של HTML

HTML משתמש בסוגריים זוויתיים כדי לציין איך תוכן המסמך יוצג בדפדפן. הסוגריים הזוויתיים נקראים תגיות (tags); זוג תגיות שיש ביניהן תוכן כלשהו נקרא **אלמנט** (element). הבט במסמך ה-HTML שראית בתרגיל הקודם, היכן שפתחת את חלון המקור עבור הדף בדפדפן:

```
<h1>Hello World</h1>
```

```
<p>Welcome to Beginning ASP.NET 3.5 on 11/1/2007 5:03:39 PM</p>
```

השורה הראשונה בדוגמה זו מכילה אלמנט `<h1>`, עם תגית פתיחה (`<h1>`) ותגית סיום (`</h1>`). אלמנט זה משמש לציון כותרת ברמה אחת. שים לב שהאלמנט נסגר עם תגית דומה, אבל בתוספת לוכסן (/) בתוכה: `</h1>`. כל טקסט בין תגיות הפתיחה והסיום האלו נחשב לחלק מהאלמנט, ולכן הוא יוצג ככותרת. במרבית הדפדפנים פירוש הדבר הוא שהטקסט יוצג עם גופן גדול יותר. בדומה לתגית `<h1>` יש תגיות ליצירה של עד שש רמות של כותרת, כגון `<h2>`, `<h3>` וכו'.

מתחת לאלמנט הכותרת אתה רואה את האלמנט `<p>`, שמשמש לציון פסקה. כל טקסט בין צמד התגיות `<p>` נחשב לחלק מהפסקה. כברירת מחדל, דפדפן מציג פסקה עם תוספת מרווח מסוים בתחתיתה, אך באפשרותך לעקוף התנהגות זו.

תגיות רבות זמינות ב-HTML, רבות מדי מכדי שניתן יהיה להציג את כולן כאן. הטבלה הבאה מציגה כמה מהתגיות היותר חשובות ונפוצות, ומתארת כיצד להשתמש בהן.

לרשימה מלאה של האלמנטים של HTML פנה לאתר Web של הארגון שמתחזק את HTML: www.w3.org/TR/html401/index/elements.html.

הערה: מידע נוסף תוכל למצוא בספר "HTML & CSS" למפתחי אתרים באינטרנט, בהוצאת הוד-עמי.

תגית	תיאור	דוגמה
<code><html></code>	ציון ההתחלה והסוף של הדף כולו.	<code><html></code> כל תוכן הדף ישוּב כאן... <code></html></code>
<code><head></code> <code><title></code>	ציון קטע מיוחד של הדף שמכיל נתונים לגבי הדף, כולל הכותרת.	<code><head></code> <code><title>My Site</title></code> <code></head></code>
<code><body></code>	ציון ההתחלה והסוף של גוף הדף.	<code><body></code> כאן יופיע גוף הדף... <code></body></code>
<code><a></code>	קישור דף web אחד לאחר.	<code></code> Visit the Wrox site <code></code>
<code></code>	שיבוץ תמונות בדף.	<code></code>
<code></code> <code><i></code> <code><u></code>	עיצוב הטקסט עם גופן עבה, מוטה או עם קו מתוח תחתיו.	This is <code></code> bold text <code></code> while <code><i></code> this text is in italic <code></i></code>
<code><form></code> <code><textarea></code> <code><select></code> <code><input></code>	הגדרת טפסי קלט שמאפשרים למשתמשי אתר אינטרנט לשלוח מידע אל השרת.	<code><input type="text" value="Some Text" /></code>

תגית	תיאור	דוגמה				
<table><tr><td></td></tr></table>		תגיות ליצירת טבלה. התגית <table> מגדירה את הטבלה כולה, ואילו התגיות <tr> ו-<td> משמשות להגדרת השורות והתאים בהתאמה.</td></tr></table>	משמשות להגדרת השורות והתאים בהתאמה.	<table><tr><td>This is a Cell in Column 1</td><td>This is a Cell in Column 2</td></tr></table>	This is a Cell in Column 1	This is a Cell in Column 2
משמשות להגדרת השורות והתאים בהתאמה.						
This is a Cell in Column 1	This is a Cell in Column 2					
	שלוש תגיות ליצירת רשימות סדורות ממוספרות או עם תבליטים. רשימה עם תבליטים. רשימה ממוספרת. משמשת לייצוג פריטים ברשימה.	- First item with a bullet - Second item with a bullet - First item with a number - Second item with a number				
	התגית נועדה לעטוף ולהשפיע על אלמנטים אחרים בדף. היא כלולה בשורה וכך אינה גורמת שבירת שורה נוספת בדף.	This is some normal text while this text appears in red </p>				
<div>	מיכל לאלמנטים אחרים. התגית פועלת כ-block element אשר גורמת כברירת מחדל לשבירת השורה לאחר התגית <div>.	This is some text on 1 line This text is put directly under the previous text on a new line.				

מאפיינים של HTML

בנוסף לאלמנטים מציגה הטבלה לעיל גם את מאפייניהם. מאפיינים מכילים מידע נוסף שמשלים וגם משנה את הדרך שבה מתנהג או מוצג אלמנט מסוים. לדוגמה, התגית משמשת להצגת תמונה, אך המאפיין src מגדיר את מקור התמונה. באופן דומה, התגית מכילה מאפיין סגנון (style) שמשנה את צבע הטקסט לאדום. הערך של המאפיין סגנון (color: red;) הינו חלק מ-CSS (Cascading Style Sheet), שיידון בפירוט רב בפרק 3. רשימה ארוכה של המאפיינים הזמינים נמצאת באתר W3C: www.w3.org/TR/html401/index/attributes.html.

אין צורך ללמוד בעל-פה את כל האלמנטים והמאפיינים האלה. רוב הזמן הם נוצרים עבורך באופן אוטומטי על ידי VWD. במקרים אחרים, כאשר תצטרך להקישם ידנית, יספק לך VWD כלי נהדר למציאת התגית או המאפיין הנכונים. כלי זה, שנקרא **IntelliSense**, יידון בשלב מאוחר יותר בספר.

ההבדל בין HTML לבין XHTML

בנוסף ל-HTML אתה עשוי להיתקל במונח XHTML. למרות שלשניים יש שמות דומים מאוד, יש ביניהם כמה הבדלים מעניינים שעליך להיות מודע להם. XHTML הוא ניסוח מחדש של HTML ב-XML (eXtensible Markup Language). XML הוא תקן בינלאומי, אשר מבוסס על טקסט ותגיות ומשמש לתיאור נתונים וגם מהווה שפת בסיס לשפות רבות אחרות, כולל HTML.

XHTML היא למעשה במידה רבה HTML אשר מעוצבת מחדש בהתאם לכללי XML. כללים אלה פשוטים למדי, ובדרך כלל תעזור לך VWD לעשות זאת נכון, או שתציג לפניך רשימת שגיאות והצעות כיצד לתקן אותן.

תמיד סגור את האלמנטים

כל האלמנטים ב-XHTML חייבים להיות סגורים. כך, כאשר אתה מתחיל פסקה עם `<p>`, במקום מאוחר יותר בדף חובה עליך לסגור את הפסקה עם `</p>`. הדבר נכון גם לגבי תגיות כגון `<img>` או `<br>`, שאין להם תגיות סגירה משל עצמם. ב-XHTML נכתבות תגיות אלו כתגיות שסוגרות את עצמן, שבהם הלוכסן הסוגר מוטמע ישירות בתגית עצמה כמו ב- `<br />` או ``.

תמיד השתמש באותיות "קטנות" עבור התגיות ושמות המאפיינים

XML רגישה לרישיות (גודל האות). XHTML מיישמת כלל זה על ידי שהיא כופה עליך לכתוב את כל התגיות והמאפיינים באותיות קטנות. למרות שהתגיות והמאפיינים חייבים להיכתב באותיות רגילות, אין זה חובה לגבי הערך של המאפיין. כך, בדוגמה הקודמת שמציגה את תמונת הלוגו, קוד XHTML הינו תקף לחלוטין למרות ה-L בשם התמונה.

תמיד הקף את ערך המאפיין בגרשיים

בכל פעם שאתה מוסיף מאפיין בתוך תגית, ודא שערך המאפיין נמצא בין גרשיים (""). לדוגמה, כאשר אתה כותב את התגית `<img>` ואת המאפיין `src`, כתוב זאת כך:

```

```

ולא כך

```
<img src=Logo.gif />
```

שים לב שאפשר לתחום את ערך המאפיין גם על ידי גרש בודד (') מכל צד. כך לדוגמה:

```
<img src='Logo.gif' />
```

לפעמים נחוץ לקנן (to nest) גרשים יחידים וגרשיים. כאשר תחביר מיוחד של ASP.NET דורש שימוש בגרשיים, עליך להשתמש בגרשים יחידים כדי לתחום את ערך המאפיין:

```
<asp:Label ID="DescriptionLabel" runat="server" Text='<# Eval("Description") %>' />
```

תחביר זה יופיע פעמים רבות בפרקים אחרים בספר.

הערה: למען העקביות, ספר זה משתמש בדרך כלל בגרשיים בכל HTML שמגיע אל הדפדפן.

קנן את התגיות כהלכה

כאשר תכתוב תגיות מקוננות, ודא לסגור תחילה את התגית הפנימית שנפתחה אחרונה, ואז לסגור את התגית החיצונית יותר. התבונן בדוגמה הנכונה הבאה, אשר מעצבת קטע טקסט עם גופן מודגש ומוטה:

```
<b><i>This is some formatted text</i></b>
```

שים לב שהתגית `<i>` נסגרת לפני התגית `<b>`. החלפת סדר התגיות הסוגרות תוביל ליצירת קוד XHTML שאינו תקף:

```
<b><i>This is some formatted text</b></i>
```

תמיד הוסף הצהרת DOCTYPE לדף שלך

DOCTYPE מספק לדפדפן מידע על סוג ה-HTML שעליו לצפות לו. VWD מוסיפה DOCTYPE לדף שלך כברירת מחדל עבור XHTML 1.0:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

ל-DOCTYPE יש השפעה מרובה על האופן שבו דפדפנים דוגמת Internet Explorer מציגים את הדף. ברירת המחדל של VWD עבור XHTML 1.0 נותנת עירוב טוב בין סימני עריכה תקפים ודפים שמוצגים באופן זהה בכל הדפדפנים העיקריים.

הערה: אם ברצונך ללמוד יותר אודות XHTML, תמצא מידע מקיף באתר W3C:

<http://www.w3.org/TR/xhtml1>

דף web של ASP.NET יכול להכיל לצד HTML גם סימני עריכה אחרים. כדי להוסיף פונקציונליות לדף יכילו מרבית הדפים פקדי שרת של ASP.NET. בסעיף הבא נעסוק בקצרה בפקדי שרת אלה, פרק 4 יעסוק בהם לעומק.

מבט ראשון ב-ASP.NET Markup

יש דמיון מסוים בין שיטת הסימון של שרתי פקד ב-ASP.NET לבין זו של HTML. כמו HTML, כך גם שיטת הסימון של הפקדים משתמשת בתגיות ומאפיינים, בסוגריים זוויתיים ובתגיות סגירה. עם זאת, יש גם כמה הבדלים.

רוב התגיות של ASP.NET מתחילות עם התחילית "asp:". לדוגמה, הגדרה של כפתור ב-ASP.NET נראית כך:

```
<asp:Button ID="Button1" runat="server" Text="Button" />
```

שים לב שהתגית סוגרת את עצמה עם תו הלוכסן (/), על ידי כך אין צורך להקליד תגית סגירה נפרדת.

דבר נוסף שאולי הבחנת בו הוא שהתגית ושמות המאפיינים אינם בהכרח באותיות קטנות בלבד. הדבר מתאפשר מפני שפקדי שרת של ASP.NET נמצאים בשרת ולפיכך אינם חייבים לדבוק בכללי XHTML שהדפדפן משתמש בהם במחשב הלקוח. זכור כי הדף שיגיע לדפדפן יהיה שונה, ושורת ההגדרה של הכפתור יוחלף עם שורה בה יוגדר הכפתור כתגית XHTML. הקוד עבור הלחצן לעיל יראה כך:

```
<input type="submit" name="Button1" value="Button" id="Button1" />
```

שים לב שהתגית כולה ומאפייניה תואמים לתקן XHTML.

כעת, כשאתה מבין את היסודות של דף ASP.NET וה-HTML שהוא מייצר, הגיע הזמן לשוב ולהתבונן ב-VWD. הידיעה כיצד להשתמש ביישום, בחלונות ובכלים הרבים שלו, היא צעד חשוב בבניית אתרי אינטרנט פונקציונליים, מהנים, ובעלי ממשק משתמש עשיר.

היכרות עם סביבת הפיתוח (IDE)

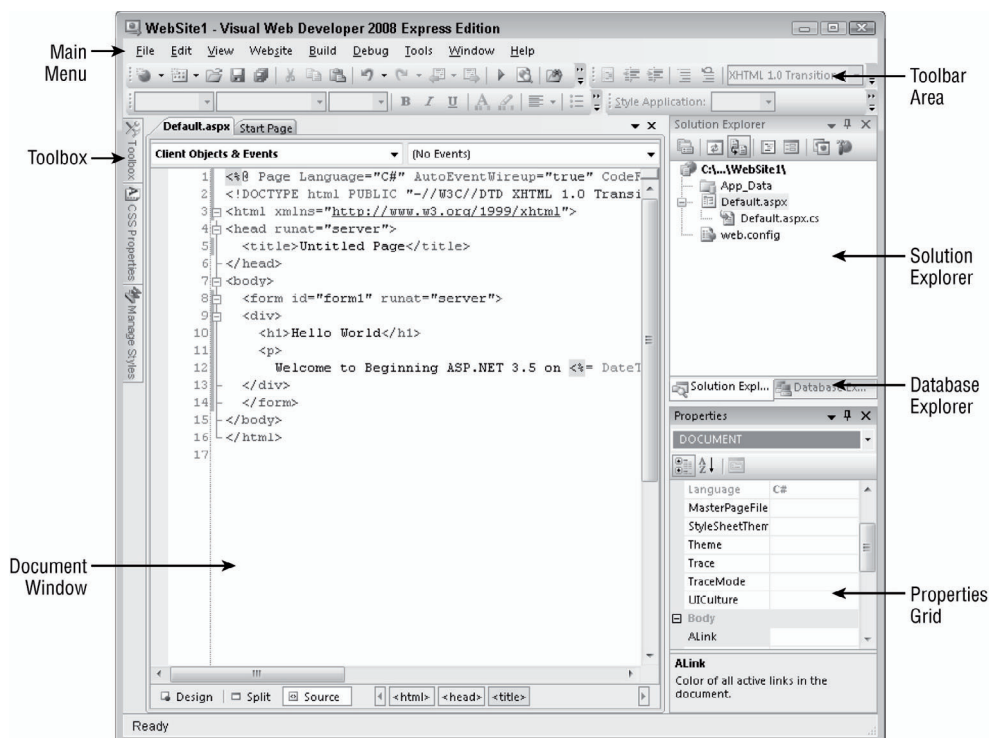
סביבת הפיתוח של VWD היא ללא ספק כלי הפיתוח המשובח הטוב ביותר לבניית דפי אינטרנט בסביבת ASP.NET. ב-VWD משולבים בסביבה אחת כל הכלים השונים שאתה זקוק להם לבניית יישומי אינטרנט מורכבים. במקום לכתוב קוד באמצעות עורך תמלילים, להדר את הקוד בשורת הפקודה, לכתוב HTML ו-CSS ביישום נפרד, ולנהל את בסיס הנתונים שלך ביישום אחר, מאפשרת VWD לבצע את כל המשימות האלה, ויותר, באותה סביבה. פרט ליכולות שנובעת מכך שאינך צריך להחליף כלים כל הזמן, הדבר גם מאפשר ללמוד תחומים חדשים של VWD, מכיוון שהכלים המובנים פועלים באותה הדרך.

שטח הפיתוח הראשי

כדי להכיר את הכלים הרבים שארוזים בממשק VWD, הבט בתרשים 8-1. מוצג בו המסך שקיבלת לאחר שיצרת את אתר האינטרנט הראשון שלך באמצעות VWD, אלא שכעת הוא מסמך כמה מהאלמנטים היותר חשובים של המסך. אם מותקנת אצלך גרסה קודמת של Visual Studio המסך עשוי להיראות שונה, מכיוון ש-Visual Studio 2008 מסוגלת לייבא הגדרות מגרסאות ישנות יותר.

התפריט הראשי (Main Menu)

בראש היישום, תחת סרגל הכותרת של Windows, אתה רואה את התפריט הראשי. סרגל תפריט זה מכיל פריטים מוכרים שאתה מוצא כמותם ביישומים רבים אחרים של Windows, כמו תפריטי File, Edit ו-Help וגם תפריטים ייחודיים ל-VWD כגון תפריטי Website ו-Debug. התפריט משתנה באופן דינמי בהתאם למשימה שאתה עוסק בה, כך שבמהלך בניית היישום תראה פריטי תפריט מופיעים ונעלמים.



תרשים 1-8

אזור סרגל הכלים (Toolbar Area)

מייד מתחת לתפריט אתה רואה את אזור סרגל הכלים, שמסוגל להציג סרגלי כלים שונים שמאפשרים גישה מהירה לפונקציות הנפוצות ביותר של VWD. בתרשים 1-8 ניתן מקום רק לארבעה סרגלי כלים, אבל VWD מכילה סרגלי כלים רבים אחרים שאתה יכול להשתמש בהם בהתאם למשימה מסוימת. כמה סרגלי כלים מופיעים באופן אוטומטי כאשר אתה עוסק במשימה שדורשת את נוכחותו של סרגל כלים מסוים, אבל ביכולתך להציג או להסתיר סרגלי כלים כאוות נפשך. כדי להציג או להסתיר סרגל כלים, לחץ לחיצה ימנית על סרגל כלים קיים או על סרגל התפריט, ובחר בסרגל כלים מהתפריט שמופיע.

תיבת הכלים (Toolbox)

בצד שמאל של המסך הראשי מופיעה כרטיסיית תיבת הכלים. אם תעביר את סמן העכבר על פני הכרטיסייה, תיחשף תיבת הכלים ותוכל לראות את תכולתה. אם תלחץ על סמל הסיכה הקטן בפינה הימנית העליונה של תיבת הכלים היא תישאר פתוחה (כך הדבר גם לגבי כל לוח אחר שבו מופיע סמל סיכה זה).

כמו עם סרגל התפריט וסרגל הכלים, תיבת הכלים מתעדכנת באופן אוטומטי כדי להציג תוכן רלוונטי למשימה שאתה עוסק בה. כאשר אתה עורך דף ASPX, מראה תיבת הכלים את הפקדים

הרבים הזמינים לדף שלך. אתה יכול לגרור פריט מתיבת הכלים ולשחררו במקום בדף שאתה רוצה להציג בו את הפריט. פקדים אלה נדונים בפירוט רב בפרק 4.

תיבת הכלים מכילה קטגוריות רבות עם כלים, וניתן להרחיב או להסיר אותן כפי שאתה רואה לנכון, כך שיהיה קל יותר למצוא את הכלי המתאים. אתה גם יכול לסדר מחדש את הפריטים ברשימה, להוסיף או להסיר פריטים מתיבת הכלים, ואפילו להוסיף לה כלים משלך. התאמה אישית של סביבת הפיתוח נדונה בהמשך הפרק.

אם תיבת הכלים אינה נראית על המסך, הקש `Ctrl+Alt+X` כדי לפתוח אותה, או בחר `Toolbox` מתפריט `View`.

מתחת לכרטיסיית תיבת הכלים יש שתי כרטיסיות נוספות: `CSS Properties` (מאפייני `CSS`) ו-`Manage Styles` (ניהול סגנונות). שתיהן נדונות בהרחבה בפרק 3.

סייר הפתרון (Solution Explorer)

בימין המסך אתה רואה את סייר הפתרון. זהו חלון חשוב, מכיוון שהוא נותן סקירה של הקבצים שנכללים באתר האינטרנט שלך. במקום להציב את כל הקבצים בתיקייה אחת גדולה, מאפשר לך חלון זה להציב את הקבצים בתיקיות נפרדות, וכך ליצור מבנה לוגי ומאורגן של האתר. באפשרותך להשתמש בחלון זה כדי להוסיף קבצים חדשים לאתר, להזיז קבצים (על ידי שימוש בתכונה `drag & drop`), למחוק קבצים ועוד. מרבית הפונקציונליות של סייר הפתרון נחבאת מאחורי תפריט הלחיצה הימנית שלו, אשר משתנה בהתאם לפריט שלחצת עליו לחיצה ימנית. בראש סייר הפתרון מופיע סרגל כלים קטן אשר נותן לך גישה מהירה לפונקציונליות שקשורה לאתר שלך, כולל פתיחת חלון המאפיינים עבור הפריט הנבחר, רענון החלון, אפשרות לקנן קבצים קשורים, ושני לחצנים שמאפשרים להעתיק ולבנות את התצורה של אתר האינטרנט שלך. כל הפונקציונליות הזו נדונה יותר מאוחר בספר.

באפשרותך לגשת לסייר הפתרון על ידי בחירה בתפריט הראשי `View` ולאחר מכן `Solution Explorer`, או להקיש `Ctrl+Alt+L`.

סייר בסיס הנתונים (Database Explorer)

חלון זה, שמוסתר בתרשים 8-1 מאחורי סייר הפתרון, מאפשר לך לעבוד מול בסיסי נתונים. הוא נותן לך את הכלים ליצירת בסיס נתונים חדש, לפתיחת בסיס נתונים קיים, הוספת טבלאות ושאלות חדשות לבסיס הנתונים, וגישה לכלים אחרים שמאפשרים לך לפעול על הנתונים בבסיס הנתונים שלך. בגרסה מסחרית של `Visual Studio`, כמו `Visual Studio 2008 Professional`, נקרא חלון זה סייר השרת (`Server Explorer`) ועשוי להימצא בצד השמאלי של המסך.

סייר בסיס הנתונים נדון בפירוט רב יותר בפרקים שעוסקים בבסיסי נתונים, החל מפרק 11.

טבלת המאפיינים (Properties Grid)

טבלת המאפיינים מאפשרת להתבונן ולערוך את המאפיינים של פריטים רבים בעזרת Visual Studio, כולל קבצים בסייר הפתרון, פקדים בדף אינטרנט, מאפייני הדף עצמו ועוד. החלון מעדכן עצמו כל הזמן כדי לשקף את הפריט שנבחר. אפשר לפתוח את טבלת המאפיינים במהירות על ידי הקשת F4. אפשר להשתמש בקיצור דרך זה כדי לכפות על טבלת המאפיינים להציג את פרטיו של פריט שנבחר.

חלון המסמך (Document Window)

חלון המסמך הוא השטח העיקרי באמצע מסך היישום. כאן מתרחשת מרבית הפעילות. תוכל להשתמש בחלון המסמך כדי לפעול עם תבניות מסמך רבות ושונות, כולל קבצי ASPX ו-HTML, קבצי CSS ו-JavaScript, קבצי קוד של VB ו-C#, קבצי טקסט ו-XML, ואפילו תמונות. בנוסף, תוכל להשתמש באותו חלון לניהול בסיסי נתונים, ליצירת עותקים של האתר שלך, לצפייה בדפים באתר שלך בעזרת המיני-דפדפן המובנה ועוד.

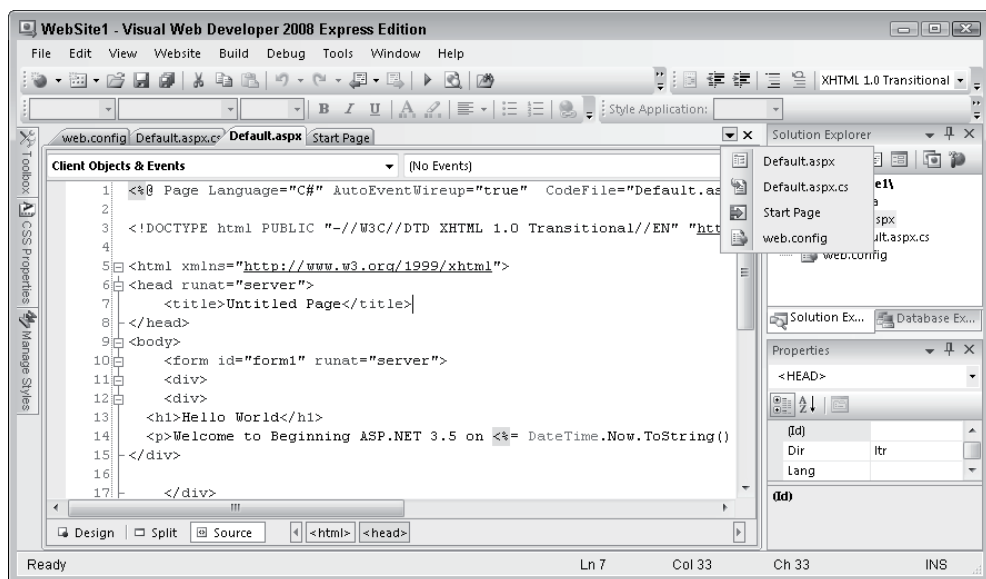
בתחתית חלון המסמך בתרשים 9-1 מופיעים שלושה לחצנים בעלי השמות Split, Design ו-Source. לחצנים אלה מופיעים באופן אוטומטי כאשר אתה עובד עם קבצים שמכילים סימני עריכה, כגון דפי ASPX ו-HTML. הלחצנים מאפשרים לפתוח את ה-Design View של דף (הדבר ייתן לך מושג כיצד ייראה הדף בדפדפן), את ה-Markup View של הדף (מאפשר לראות את ה-HTML וסימני עריכה אחרים), או את שניהם יחד. כיצד הדבר מתאפשר? הדבר יוסבר בפירוט רב יותר בפרק 2. כרגע חשוב להבין שאתה יכול לעבור בין Markup View לבין Design View על ידי לחיצה על הלחצן המתאים. ה-Markup View נקרא לעתים קרובות גם Source View או Code View. כדי למנוע בלבול, נשתמש בספר זה במונח Markup View בלבד.

כברירת מחדל, חלון המסמך הוא חלון כרטיסיות. פירושו של דבר שהוא יכול לארח מסמכים רבים שכל אחד מהם מובדל מהאחרים על ידי כרטיסייה עם שם הקובץ בחלק העליון של החלון. עבור כל כרטיסייה, התפריט שנפתח בלחיצה ימנית מכיל כמה קיצורים שימושיים לעבודה עם הקובץ, כולל שמירתו וסגירתו, ופתיחת תיקיית האם של הקובץ בסייר החלונות.

כדי לעבור בין המסמכים תוכל להקיש Ctrl+Tab, או ללחוץ על החץ שמצביע מטה שנמצא בפנינה הימנית העליונה של חלון המסמך, כפי שרואים בתרשים 9-1. לחיצה על החץ שמצביע מטה חושפת רשימה של מסמכים פתוחים, כך שניתן לבחור אחד מהם בקלות.

דף הפתיחה (Start Page)

בכל פעם שתפעיל את VWD ייטען דף הפתיחה לחלון המסמך. בעזרת דף הפתיחה תוכל ליצור במהירות אתרי web חדשים, או לפתוח אתרים קיימים. דף הפתיחה גם מקנה לך גישה לכמה נושאי עזרה נפוצים ומציג כותרות מאתר Web של מיקרוסופט. החלק העיקרי של דף הפתיחה משתמש ב-RSS כדי להציג מידע מצוות MSDN Visual Web Developer.



תרשים 9-1

כדי לקבל תחושה איך תוכל להשתמש בכל החלונות האלה, מראה לך תרגיל נסה זאת הבא כיצד לבנות דף web פשוט שמכיל כמה פקדי שרת של ASP.NET.

נסה זאת תכנון דף web הראשון שלך על ידי ASP.NET

תרגיל זה מנחה אותך ליצור אתר אינטרנט חדש בעל דף בודד שמכיל כמה פקדי שרת של ASP.NET. תלמד איך להשתמש בחלונות דוגמת דף הפתיחה וסייר הפתרון, ואיך להשתמש בתיבת הכלים ובטבלת המאפיינים להוספת פקדי שרת של ASP.NET לדף ולשינוי המראה שלהם.

1. הפעל את VWD. אם אינך רואה את דף הפתיחה, בחר מהתפריט הראשי: View ← Start Page ← Other Windows.

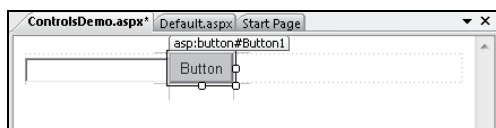
2. בדף הפתיחה, בשטח Recent Projects, לחץ על Web Site שליד התווית Create. פעולה זו מפעילה את תיבת הדו-שיח New Web Site. אם אינך רואה בדף הפתיחה את הקישור ליצירת אתר web חדש, בחר במקום זאת מהתפריט הראשי של VWD את File ← New Web Site או את File ← New Project ← Web Site. ודא שבחרת ASP.NET Web Site, וכי מהרשימה הנפתחת Location בחרת File System. לחץ OK ליצירת האתר החדש.

3. בסייר הפתרון לחץ לחיצה ימנית על אתר Web החדש. ודא שלחצת על האלמנט העליון שתוכנו דומה ל- C:\.\WebSite2\). זהו האלמנט המסומן בתרשים 1-4. מתפריט התוכן שמופיע בחר Add New Item.

4. בחלון החדש שמופיע לחץ Web Form והקלד את השם ControlsDemo. ברגע שתלחץ על הלחצן OK תתווסף הסיומת ASPX באופן אוטומטי. אתה יכול להשאיר את שאר ההגדרות בתיבת הדו-שיח לפי ברירות המחדל שלהן. הדף ייפתח ב- Markup View.

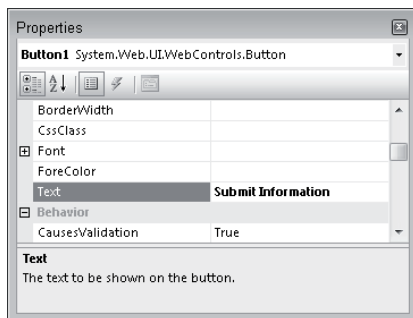
ותוכל לראות את ברירות המחדל של HTML, כמו האלמנטים <html>, <head>, <title> ו- <body> ש- Visual Web Developer מוסיפה שם באופן אוטומטי כאשר אתה יוצר דף חדש.

5. העבר את הדף ל- Design View על ידי לחיצה על הלחצן Design בתחתית חלון המסמך.
6. אם תיבת הכלים איננה פתוחה עדיין, הקש Ctrl+Alt+X לפתיחתה, או רחף עם העכבר על פני כרטיסיית תיבת הכלים כדי לגרום להצגתה, ואז לחץ על סמל הסיכה כדי לגרום לכך שתיבת הכלים תהיה גלויה כל הזמן. גרור TextBox ו- Button מתיבת הכלים אל האזור המקווקו ב- Design View של הדף. בסופו של דבר תסיים עם דף שייראה דומה לתרשים 1-10.



תרשים 1-10

7. לחץ לחיצה ימנית על הפקד Button ובחר Properties. בטבלת המאפיינים, אתר את המאפיין Text הנמצא תחת קטגוריית Appearance ושנה את הערך מ-'Button' ל- 'Submit Information'. ברגע שתקיש Tab או תלחץ במקום כלשהו מחוץ לטבלת המאפיינים, יעודכן ה- Design View ויראה את הטקסט החדש על כפתור.



תרשים 1-11

8. לפתיחת הדף בדפדפן שלך הקש Ctrl+F5. שים לב שאין צורך בשמירה מפורשת של השינויים בדף (אם כי זהו רעיון טוב לבצע זאת לעתים קרובות על ידי שימוש בקיצור הדרך Ctrl+S). ברגע שתקיש Ctrl+F5 תשמור באופן אוטומטי את כל השינויים שנעשו במסמכים פתוחים.
9. הקלד טקסט כלשהו בתיבת הטקסט ולחץ על הכפתור. שים לב שלאחר שהדף נטען מחדש, הטקסט עדיין מוצג בתיבת הטקסט. פרט לכך לא קרה הרבה, מפני שטרם כתבת קוד כלשהו עבור אירוע הלחיצה של הכפתור.

כיצד זה פועל

כאשר גררת Button ו-TextBox מתיבת הכלים על פני הדף ב- Design View, הוסיפה VWD באופן אוטומטי את הקוד המתאים ב- Markup View. באופן דומה, כאשר שינית בטבלת המאפיינים את המאפיין Text של הכפתור, עדכנה VWD באופן אוטומטי את סימני העריכה של הפקד ב- Markup View. במקום להשתמש בטבלת המאפיינים, באפשרותך להקליד את הטקסט ישירות בין סימני הגרשיים של המאפיין Text בחלון הקוד.

לאחר שינוי המאפיין Text ייראה כך הדף שלך ב- Markup View:

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:Button ID="Button1" runat="server" Text="Submit Information" />
```

כאשר אתה מקיש Ctrl+F5 כדי לצפות בדף שבדפדפן, שרת האינטרנט מקבל את הבקשה, הדף מעובד על ידי ASP.NET runtime, וה-HTML שהתקבל כתוצאה מכך נשלח לדפדפן.

הפקודה View Source של הדפדפן מאפשרת להתבונן ב-HTML שהתקבל כתוצאה מעיבוד הדף (אם כבר סגרת את הדף, הפעל אותו מחדש מ-VWD על ידי הקשת Ctrl+F5). תראה קוד דומה לזה:

```
<input name="TextBox1" type="text" value="Hello World" id="TextBox1" />
<input type="submit" name="Button1" value="Submit Information" id="Button1" />
```

בדיוק כמו בדוגמה הקודמת, ה-HTML שהתקבל שונה באופן משמעותי מסימני העריכה המקוריים של ASPX.

לאחר שתקליד מעט טקסט ותלחץ על הלחצן, התהליך חוזר על עצמו כמעט באופן דומה: שרת האינטרנט מקבל את הבקשה, הדף עובר עיבוד, והתוצאה נשלחת חזרה לדפדפן. כאשר תלחץ על הלחצן אתה גורם ל-**postback** (פנייה חזרה לשרת). כלומר, כל מידע שמוכל בדף - כמו הטקסט שהקלדת בתיבת הטקסט - נשלח חזרה אל השרת. תגובת ASP.NET ל-**postback** היא פעולת יצירה של הדף, אולם הפעם ASP.NET שותל בפקדים, כמו למשל ב-TextBox, את הערכים שנשלחו אל הדף. **postback** הוא מושג חשוב ב-ASP.NET, ונלמד אודותיו יותר בפרקים אחרים, וביניהם הפרקים 4 ו-9.

VWD מארחת מספר גדול חלונות ולוחות כלים יותר מאשר אלה שראית עד כה. הסעיף הבא מתייחס בקצרה לכמה מהחלונות שתשתמש בהם בתכיפות בעת בניית דפי אינטרנט בסביבת ASP.NET. כל החלונות המוזכרים נגישים מתפריט View הראשי של VWD.

חלונות שמכילים מידע

פרט לחלונות ש-VWD חושפת כברירת מחדל בעת הפעלתה, יש ב-VWD מספר רב הרבה יותר של חלונות זמינים. את מרביתם תראה בפעולה בהמשך הספר, אבל כמה מהם כדאי להדגיש עכשיו. החלונות הבאים הינם חלונות מידע.

רשימת השגיאות (Error List)

רשימת השגיאות נגישה מתפריט View, ומציגה רשימה של דברים שאינם תקינים כרגע באתר שלך, כולל סימני עריכה לא נכונים בקבצי ASPX או HTML, ושגיאות תכנות בקבצי הקוד (VB או C#). חלון זה יכול להראות לך גם שגיאות בקבצי XML ו-CSS. רשימת השגיאות מציגה את הודעותיה בשלוש קטגוריות שמבטאות את חומרת הבעיה: **שגיאות** (Errors), **אזהרות** (Warnings) ו**הודעות** (Messages). תרשים 1-12 מציג את רשימת השגיאות עבור דף שיש לו כמה בעיות ב-CSS וב-XHTML.

	Description	File	Line	Column	Project
1	Validation (CSS 2.0): 'balck' is not a valid value for the 'color' property.	Default.aspx	11	24	CA...WebSite2\
2	Validation (XHTML 1.0 Transitional): Element 'div' is missing its closing tag.	Default.aspx	11	6	CA...WebSite2\

תרשים 1-12

חלון הפלט (Output Window)

כאשר אתה מנסה לבנות אתר באמצעות התפריט Build, חלון הפלט אומר לך אם הבנייה הצליחה או לא. אם הבנייה נכשלה, תמצא בחלון את תיאור הסיבה לכישלון. בגרסאות המסחריות של Visual Studio משמש חלון הפלט גם למתן מידע נוסף, כולל הסטטוס של תוכניות תוסף (plug-in) חיצוניות. בניית אתרי אינטרנט תידון מאוחר יותר בספר, כולל פרק 18 אשר דן בפריסת (deployment) אתר האינטרנט שלך.

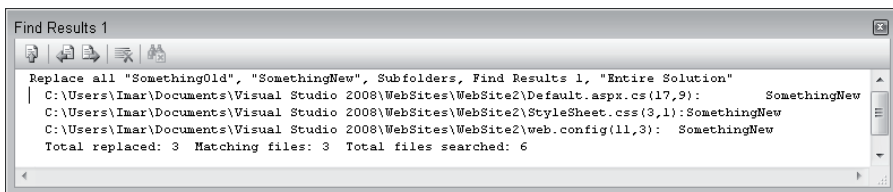
חלון הסימניה (Bookmark Window)

בחלון המסמך תוכל להוסיף סימניה לקבצי קוד רבים על ידי הקשה פעמיים על Ctrl+K. מקש קיצור דרך זה מאפשר לשתול סימן כלשהו בשולי שורת קוד כדי שמאוחר יותר תוכל לבחור בו תוך שימוש בחלון הסימניה. הדבר מאפשר תנועה זריזה בקוד, אשר שימושית במיוחד כאשר האתר מתחיל לגדול.

החלון Find Results

התכונות Find (חפש) ו-Replace (החלף) של VWD הם כלים חשובים לניהול תוכן האתר. לעתים קרובות תצטרך להחליף טקסט במסמך הנוכחי או אפילו באתר כולו. התוצאות של Find in Files (Ctrl+Shift+F) ושל Replace in Files (Ctrl+Shift+H) מופיעות בחלון Find Results (תוצאות חיפוש), כפי שמוצג בתרשים 1-13.

אם תפתח חלונות מידע רבים בו-זמנית, הדבר עלול לתפוס שטח מסך יקר, ולפיכך מומלץ לעגן אותם (הצמדה לקצה). בדרך זו, רק אחד מהם יהיה גלוי בכל רגע נתון, ועדיין תהיה לך גישה מהירה אל האחרים. בשלב הבא תלמד כיצד להתאים את סביבת הפיתוח לצרכיך, כולל עיגון חלונות.



תרשים 1-13

התאמה אישית של סביבת הפיתוח (IDE)

למרות שהגדרת ברירת המחדל של VWD וחלונות הכלים שלה שימושית ונוחה למדי, קרוב לוודאי שתמצאו להתאים את סביבת הפיתוח לטעמך האישי. אתה עשוי לרצות לשנות מיקום של כמה מהחלונות כדי שהגישה אליהם תהיה קלה יותר עבורך, או שתמצאו לפתוח חלונות נוספים שתדירות השימוש בהם גבוהה. VWD ניתנת להתאמה אישית מלאה ומאפשרת לך לקבוע בעדינות כל פרט בסביבת הפיתוח. בסעיפים הבאים תלמד לבצע את משימות ההתאמה האישית הנפוצות ביותר.

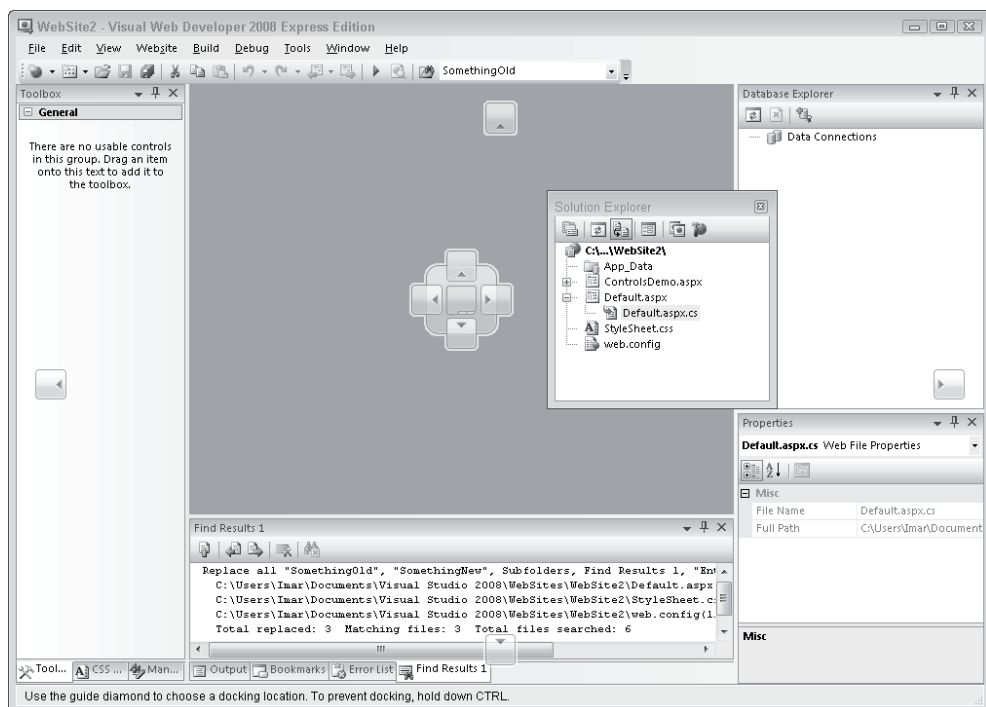
סידור מחדש של החלונות

כדי לתת לכל חלון את המקום הראוי לו, גרור ושחרר אותו במסך הראשי: תפוס את סרגל הכותרת של החלון, או את הכרטיסייה התחתונה שלו, וגרור אותם בכיוון המיקום החדש. ברגע שתתחיל לגרור תראה ש-VWD נותנת לך סימנים חזותיים באשר למקום שבו החלון יוצב בסופו של דבר (ראה תרשים 1-14).

בנוסף לעיגון חלונות, אפשר שיהיו גם חלונות צפים. כדי להפוך חלון מעוגן לחלון צף, גרור אותו ממקומו הנוכחי ושחרר אותו במקום כלשהו בסביבת הפיתוח, מבלי לפגוע באחד מהסימנים החזותיים שבמסך, או בחר מהתפריט הראשי: Window ← Floating. אם תרצה להחזיר לוח צף למיקום הקבוע הקודם שלו, בחר מהתפריט הראשי: Window ← Dockable, ואז לחץ לחיצה כפולה על סרגל הכותרת שלו.

התאמת תיבת הכלים

גם את תיבת הכלים ניתן לשנות. אפשר לסדר את הפריטים מחדש בסדר אלפביתי באופן שיקל עליך למצוא אותם ברשימה. כדי לבצע זאת, פתח את תיבת הכלים (הקש Ctrl+Alt+X), לחץ על אחד מהפריטים בקטגוריה (למשל TextBox בקטגוריה Standard), ובחר Sort Items Alphabetically. אפשר למחוק פריטים מהרשימה על ידי לחיצה ימנית עליהם ואז לבחור Delete בתפריט ההקשר. אל תדאג לגבי אובדן בלתי הפיך של פריטים, באפשרותך לאפס את תיבת הכלים על ידי בחירת Reset Toolbox באותו תפריט.



תרשים 1-14

אתה יכול גם להוסיף פריטים משלך לתיבת הכלים. השימוש הנפוץ ביותר לכך הוא זה של קטעי קוד. סמן טקסט או קטע קוד כלשהו בחלון המסמך וגרור אותו לתיבת הכלים. לאחר מכן באפשרותך להעניק לפריט שם בעל משמעות כך שתוכל לזהותו בקלות. דרך הביצוע: לחץ לחיצה ימנית על הפריט ואז בחר **Rename Item**.

כדי להימנע מבלבול בין תיבת הכלים לבין קטעי הקוד שלך, שקול יצירת קטגוריה נפרדת עבורם. דרך הביצוע: לחץ לחיצה ימנית על **Toolbox**, מהתפריט שיפתח בחר **Add Tab**, הקלד שם והקש **Enter**. כרטיסיית תיבת הכלים מוכנה לשימוש.

בתרגיל נסה זאת שלהלן ניתנת לך הזדמנות "לשחק" עם סביבת הפיתוח של VWD, כדי להתאים אותה לטעמך.

נסה זאת התאמה אישית של סביבת הפיתוח

בתרגיל זה תוכל להתאמן בפתיחה וסידור מחדש של החלונות בסביבת הפיתוח של Visual Web Developer. תלמד לסדר באופן אלפביתי את הפריטים בקטגוריה Standard של תיבת הכלים, כדי שיהיה קל יותר למצוא את הפקדים. אל תחשוש "לבלגן" את סביבת הפיתוח. יותר מאוחר בפרק מופיעות הוראות כיצד לאפס את סביבת הפיתוח למצבה ההתחלתית.

1. אם סגרת את אתר האינטרנט שלך מאז תרגיל נסה זאת הקודם, פתח אותו שוב או צור חדש, תוך שימוש ב- **Start Page** או בתפריט **File**.

2. מתפריט View בחר Error List כדי לפתוח את חלון רשימת השגיאות. שים לב כיצד הוא מעוגן כברירת מחדל, מתחת לחלון המסמך.
3. מתפריט View בחר Task List. כברירת מחדל רשימה זו תעוגן באותו מרחב של Error List, כאשר הכרטיסיות של שני החלונות סמוכות אחת לשנייה.
4. לחץ על כרטיסיית Task List, ובעודך מחזיק את לחצן העכבר לחוץ, גרור את הרשימה Task List ממקומה לכיוון חלון המסמך. ברגע שתשחרר את החלון, הוא יופיע כחלון צף. כדי לאפס את החלון, לחץ לחיצה כפולה על סרגל הכותרת שלו. שים לב כיצד הכרטיסייה חוזרת לאותה קבוצת כרטיסיות, אם כי אולי אל מיקום אחר. כדי לשנות את הסדר לפיו מופיעות הכרטיסיות בקבוצת כרטיסיות, גרור כרטיסייה על פני הכרטיסיות האחרות ושחרר אותה במקום הרצוי לך.
5. באפשרותך לחזור על הצעדים הקודמים עבור חלונות אחרים שגלויים כברירת מחדל, או עבור חלונות שתמצא תחת התפריט View. הקדש זמן מה כדי להרגיל את עצמך לכל החלונות השונים, ולדרך שבה אתה יכול לארגנם על המסך. מכיוון שתעבוד הרבה עם חלונות אלה בשאר הספר, רצוי להכיר את מיקומם.
6. מתוך סייר הפתרון פתח את הדף Default.aspx על ידי לחיצה כפולה עליו. כאשר הדף נפתח, תיבת הכלים תהפוך באופן אוטומטי לגלויה, ואם היא לא, הקש Ctrl+Alt+X כדי לפתוח אותה.
7. כברירת מחדל, הקטגוריה Standard צריכה להתרחב, ואם היא לא, לחץ על הסמל (+) בשוליים השמאליים של תיבת הכלים. לאחר מכן לחץ לחיצה ימנית במקום כלשהו בפקד שמופיע בקטגוריה Standard, ובחר Sort Items Alphabetically. הפעולה תסדר את הפקדים, ורק אותם, בקטגוריה Standard בסדר אלפביתי. אם רצונך בכך, תוכל לחזור על צעד זה עבור קטגוריות אחרות בתיבת הכלים.
8. לחץ שוב לחיצה כפולה על תיבת הכלים, ובחר Add Tab. הקלד את השם החדש, **HTML Fragments**, והקש Enter. הדבר יוסיף לתיבת הכלים קטגוריה חדשה אשר תתנהג בדיוק כמו כל האחרות.
9. כשחלון המסמך מראה את הדף Default.aspx ב- Markup View, הקלד <h1> בין תגיות הפתיחה והסגירה <div>. שים לב ש-WVD מכניסה עבורך באופן אוטומטי את תגית הסגירה של <h1>. אתה צריך לסיים בסופו של דבר עם חלון קוד שנראה כך:

```
<div>
<h1></h1>
</div>
```
10. סמן את תגיות הפתיחה והסגירה **<h1>**, ואז גרור את הבחירה מהחלון Markup View אל הכרטיסייה החדשה שיצרת בתיבת הכלים בצעד 8 לעיל. הבחירה נראית כך:

```
<h1></h1>
```
11. לחץ לחיצה ימנית על פריט תיבת הכלים שיצרת זה עתה, בחר Rename Item והקלד את השם **Heading 1**.

12. חזור על צעדים 11-9 כשאתה יוצר כותרות מרמה h2 ועד רמה h6.

מעטה והלאה, בכל פעם שתזדקק לכותרת במסמך שלך ב- Markup View, עליך להציב את הסמן בחלון המסמך במקום שרצוי לך עבור הכותרת, ואז ללחוץ לחיצה כפולה על הכותרת המתאימה בתיבת הכלים.

כיצד זה פועל

מרבית הצעדים בתרגיל האחרון מסבירים את עצמם. התחלת בפתיחה של כמה חלונות שאתה נזקק להם לעתים קרובות בעת בניית יישומי אינטרנט. לאחר מכן השתמשת בתכונה 'גרור ושחרר' של סביבת הפיתוח כדי לסדר מחדש את מערך החלונות על פי העדפותיך האישיות. גם סידרת מחדש את הפריטים בתיבת הכלים באופן שיקל עליך למצוא אותם.

סיימת את התרגיל בהוספת כמה קטעי HTML לכרטיסייה מותאמת בתיבת הכלים. כאשר אתה גורר איזושהו סימן עריכה אל תיבת הכלים, VWD יוצרת עבורו פריט מתאים בתיבת הכלים אשר מכיל את סימן העריכה הנבחר. בכל פעם שאתה זקוק לעותק של סימן עריכה זה בדף שלך, עליך ללחוץ לחיצה כפולה על הפריט או לגרור אותו מתיבת הכלים אל חלון Markup View. הדבר גורם לחיסכון רב בזמן עבור קטעי HTML שמשמשים בהם בתכיפות.

פרט לפריסת החלון ותיבת הכלים, מאפשרת לך VWD להתאים באופן אישי דברים רבים נוספים. הסעיף הבא מסביר איך להתאים באופן אישי שלוש תכונות חשובות נוספות של סביבת הפיתוח: חלון המסמך, סרגלי כלים וקיצורי דרך במקלדת.

התאמת חלון המסמך (Document Window)

התוכנה Visual Web Developer מאפשרת גמישות רבה בעת הצגת טקסט בחלון המסמך. אתה יכול לשנות דברים כמו גודל הגופן, צבע הגופן ואפילו את צבע הרקע של הטקסט. יש לך גישה להגדרות Font and Colors באופן זה: בחר Tools ← Options, בחר Show All Settings בתחתית תיבת הדו-שיח, בחר Environment ← Fonts and Colors.

אחד הדברים שאני אוהב להתאים אישית בחלון המסמך הוא מרחק הדילוג של מקש ה-Tab. כדי לשנות את מרחק הדילוג, בחר Tools ← Options, לאחר מכן, תחת Text Editor, בחר All Languages ← Tabs. אם אינך רואה אפשרות בחירה זו, בחר תחילה Show All Settings בתחתית המסך. בדרך כלל אני קובע הן את Tab Size והן את Indent Size ל-2, ומשאיר את שאר ההגדרות בלוח Tab ללא שינוי.

התאמת סרגלי כלים

סרגלי כלים יכולים להיות מותאמים בשלוש דרכים: אתה יכול להציג או להסתיר את סרגלי הכלים המובנים, באפשרותך להוסיף או להסיר לחצנים מסרגלי כלים קיימים, ואתה יכול ליצור

סרגלי כלים משלך עם לחצנים שהשימוש בהם תכוף. אתה מוסיף או מסתיר סרגלי כלים קיימים על ידי לחיצה ימנית על סרגל כלים קיים כלשהו, או על סרגל התפריט, ואז בוחר בפריט המתאים מהרשימה. ברגע שסרגל הכלים מוצג, אתה יכול להשתמש בסימן הגרירה שלו, שבשמאל סרגל הכלים, כדי לגרור אותו למקומו החדש. אתה יכול לגרור את סרגלי הכלים למקום כלשהו, כולל הצד הימני או השמאלי של המסך, למקום שבו תעגן אותם כסרגלים אנכיים. באפשרותך גם ליצור אותם כסרגלי כלים צפים ולמקם אותם בכל מקום על פני המסך.

עריכת סרגלי כלים קיימים

אם אתה סבור שבסרגל כלים קיים חסר לחצן חשוב, או שהסרגל מכיל לחצנים שהשימוש בהם נדיר, אתה יכול להתאים את הלחצנים בסרגל לפי דרישותיך. כדי לעשות זאת, לחץ לחיצה ימנית על סרגל כלים קיים כלשהו או על סרגל התפריט, ובחר Customize. עכשיו סמן את החזית של סרגל הכלים שברצונך להתאים. כך תבטיח שהוא יופיע. לאחר מכן עבור לכרטיסייה Commands, בחר קטגוריה מהרשימה שבשמאל, ואתר את הפקודה ברשימה Command שבצד ימין. כעת תוכל לגרור את הפקודה מהחלון Customize אל סרגל הכלים.

הסרת לחצן מסרגל כלים קלה אף יותר. בעוד החלון Customize פתוח, לחץ לחיצה ימנית על הלחצן הממוקם בסרגל ובחר Delete.

בעורך בתיבת הדו-שיח Customize ייתכן שתמצא לאפשר קיצורי מקשים על ידי סימון Show Shortcut Keys בהגדרה ScreenTips שבסרגל Toolbars. בדרך זו, סרגלי הכלים עבור הלחצן יציגו את מקש קיצור הדרך המקושר ללחצן בזמן שתעבור עם סמן העכבר מעל הלחצן, כך שסביר יותר שתשגן אותם ותשתמש בהם. מקשי קיצור דרך הם לעתים קרובות קלים יותר לשימוש מאשר לחצני סרגל הכלים או התפריט המקבילים להם.

יצירת סרגלי כלים משלך

יצירת סרגל כלים משלך היא צעד שימושי כאשר ברצונך לקבץ כמה תכונות שאתה משתמש בהן בתכיפות. כדי ליצור סרגל כלים חדש פתח את החלון Customize כפי שמוסבר בסעיף הקודם. לחץ על הלחצן New והקלד שם עבור סרגל הכלים החדש. כאשר תלחץ OK יוצג סרגל הכלים החדש על המסך. עכשיו אתה יכול לגרור אליו פקודות באותה הדרך שנקטת כאשר שינית סרגלי כלים קיימים.

התאמת קיצורי דרך למקלדת

מפתחים רבים אוהבים לשנות את הגדרות קיצורי הדרך במקלדת. קיצורי דרך במקלדת תורמים לחיסכון בזמן מכיוון שהם מאפשרים לבצע משימה על ידי פקודת מקלדת פשוטה במקום אחיזת העכבר ובחירת הפריט המתאים מהתפריט. כדי לשנות את הגדרות קיצורי הדרך במקלדת פתח שוב את תיבת הדו-שיח Customize על ידי לחיצה ימנית על סרגל כלים או בחירה מהתפריט Tools. לאחר מכן לחץ על הלחצן Keyboard. ברשימת הפקודות אתר את הפקודה שברצונך לשנות את קיצור הדרך שלה. מכיוון שהרשימה ארוכה, תוכל לסנן אותה על

ידי הקלדת כמה אותיות מהפקודה. לדוגמה, הקלדת **print** בפקודות Show שמכילות שדה, מציגה את רשימת כל הפקודות הקשורות להדפסה.

הקלד כעת קיצור דרך חדש בשדה VWD.Press shortcut keys מאפשרת להגדיר שני קיצורי דרך עבור אותה פקודה. לדוגמה, אתה יכול לקשר את הפקודה Close All Documents עם הקיצורים Ctrl+O, Ctrl+K. כעת, כדי לבצע פקודה זו עליך ללחוץ בזריזות על שני שילובי המקשים בזה אחר זה. למרות שהקשה על שני השילובים האלה עשויה להיראות לך כמאמץ יתר, הדבר מגדיל במידה ניכרת את מספר מקשי הקיצור הזמינים.

איפוס (reset) השינויים

אם אתה חש ש"בלגנת" את VWD על ידי ניסוי אפשרויות התאמה רבות, הסר דאגה מלבך. קיימות דרכים רבות להשיב את VWD למצבה הקודם.

איפוס פריסת החלונות

הגדרה זו נגישה מתפריט Window ← Reset Window Layout ומאפסת את כל החלונות ומשחזרת אותם למצב שהיו בו כאשר VWD הופעלה לראשונה. פקודה זו שימושית למקרה שמיקמת חלונות רבים מדי שלא במקומם ולפניך סביבת פיתוח עמוסה ולא נוחה לעבודה.

איפוס ארגו הכלים

אם הסרת בטעות לחצן כלשהו מארגו הכלים או אפילו מחקת כרטיסייה שלמה, באפשרותך לאפס את ארגו הכלים למצבו המקורי על ידי לחיצה ימנית על ארגו הכלים ובחירת Reset Toolbox. חשוב פעמיים לפני שתשתמש בפקודה זו, משום שהיא תמחק גם את כל קטעי הקוד המותאם אישית שלך.

איפוס כל ההגדרות

כדי להשיב לחלוטין את כל ההגדרות VWD למצב שהיו בו מייד לאחר ההתקנה, בחר Import and Export Settings מהתפריט Tools. לאחר מכן בחר באפשרות Reset All Settings ולחץ Next. אם ברצונך, אתה יכול ליצור גיבוי של ההגדרות הנוכחיות; אחרת בחר No, Just Reset Settings. לבסוף לחץ Finish. פעולה זו תגרום לאיפוס כל ההגדרות אל ברירות המחדל שלהן, כולל ארגון החלונות, סרגלי הכלים וההתאמה האישית של Toolbox, מקשי קיצור הדרך וכל דבר ששינית בתיבת הדו-שיח Options של VWD. לפיכך, עליך להשתמש בפקודה זו רק כאשר אתה בטוח לחלוטין שאתה רוצה בהגדרה חדשה של VWD.

אם עקבת אחר תרגילי נסה זאת הקודמים, ואחר כך התחלת להתנסות באפשרויות ההתאמה האישית, כלי הפיתוח שלך נמצא כעת, ככל הנראה, באחד משני מצבים: הוא נראה בדיוק כפי שרצית שייראה, או שהוא נראה כבלגן אחד גדול. במקרה הראשון אתה יכול לדלג על התרגיל הבא; במקרה השני – הישאר וראה עד כמה קל לנקות את הבלגן.

תרגיל זה מראה לך איך לאפס את סביבת הפיתוח למצבה המקורי כאשר הפעלת את VWD בפעם הראשונה. לפני ביצוע התרגיל ודא שאכן ברצונך לעשות זאת, מכיוון שהתרגיל יאפס את כל ההגדרות החשובות, כולל ההתאמות האישיות של חלונות וארגז הכלים שטרכת עליהן, וכל האפשרויות שהגדרת בתיבת הדו-שיח Options.

1. הפעל את Import and Export Settings Wizard על ידי בחירת Tools ← Import and Export Settings Wizard .
2. בחר באפשרות Reset All Settings שבתחתית המסך, ולחץ Next.
3. תן ל-VWD ליצור גיבוי להגדרות הקיימות שלך על ידי בחירת הפריט הראשון בכפתורי הבחירה. עם גיבוי זה תוכל תמיד לחזור להגדרות הנוכחיות על ידי הפעלה חוזרת של Import and Export Settings Wizard.
4. לחץ Finish. האשף מאפס את כל ההגדרות שלך ומציג הודעה על כך שההגדרות אופסו בהצלחה.
5. לחץ Close כדי לצאת מהאשף. תיווכח שסביבת הפיתוח שלך היא כעת בדיוק כפי שהייתה בפעם הראשונה שהפעלת אותה.

כיכר זה פועל

כל השינויים שביצעת בסביבת הפיתוח מאוחסנים בקובץ תצורה במבנה XML שבתקייה Visual Studio Settings שלך, אשר נמצאת, כברירת מחדל, בתקייה Visual Studio 2008\Settings תחת תיקיית Documents הראשית שלך ב-Windows.

כאשר אתה בוחר לאפס את ההגדרות שלך, VWD דורסת את קובץ ההגדרות שלך על ידי ברירות המחדל. אם אתה בוחר ליצור גיבוי, גם קובץ הגיבוי שמכיל בתוכו את התאריך הנוכחי, נשמר באותה תיקייה. בעזרת קובץ הגיבוי ניתן לשחזר את ההגדרות במועד מאוחר יותר.

כעת, משיש לך קצת ידע בסיסי על דפי ASP.NET ו-VWD, הגיע הזמן לפעולה רצינית. בפרק הבא תלמד בפירוט רב יותר כיצד ליצור אתרי אינטרנט ודפי אינטרנט בסביבת ASP.NET. תלמד כיצד לארגן את האתר שלך בדרך לוגית ומובנית, איך להוסיף לאתר סוגים רבים ושונים של קבצים ואיך להשתמש בהם, וכיצד לקשר את הדפים באתר שלך.

עדיין נותר נושא אחד חשוב שראוי ללמוד לפני שתוכל להמשיך לפרק הבא: היישום לדוגמה שכלול בספר זה.

היישום לדוגמה

ספר זה עוסק בבניית אתרי אינטרנט (web) ועל כן הגיוני שהוא יכלול אתר לדוגמה, מלא ופונקציונלי, שמשמש להצגת רבות מהיכולות של ASP.NET.

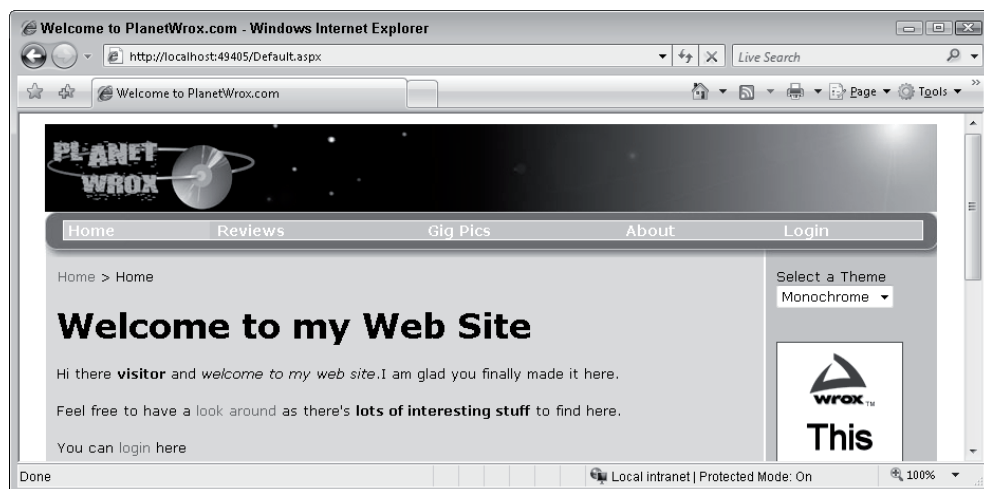
האתר לדוגמה שתבנה בספר זה נקרא **Planet Wrox**, והוא נועד לשמש כקהילה מקוונת עבור אנשים שמתעניינים במוזיקה. האתר מציג למבקרי את התכונות הבאות:

- ⊙ סקירות (ביקורות) על תקליטורים וקונצרטים אשר מתפרסמות באתר על ידי מנהל המערכת.
- ⊙ Gig Pics - אלבום תמונות מקוון שמאפשר למשתמשים לשתף תמונות שצולמו בקונצרטים.
- ⊙ יכולת מעבר בין הנושאים הגרפיים השונים שהאתר מציג, נותנת לך הזדמנות לשנות את המראה והתחושה של האתר מבלי לשנות את תוכנו.
- ⊙ העדפות מוזיקליות שמספיעות על המידע שאתה רואה באתר.
- ⊙ גישה לתוכן בונס למשתמשים שפותחים חשבון באתר.

כמנהל האתר יש באפשרותך:

- ⊙ להוסיף סקירות/ביקורות ולתחזק אותן.
- ⊙ לנהל את הסגנונות המוזיקליות השונים במערכת.
- ⊙ לקבוע לאיזה משתמשים תורשה גישה לתכנים מוגנים, כמו אלבומי תמונות מיוחדים.

תרשים 1-15 מציג את דף הבית של Planet Wrox.



תרשים 1-15

תרשים 1-16 מציג דף נוסף, שמיישם נושא אחר. הדף מאפשר למשתמשים להכניס מידע אישי ולציין את העדפותיהם לגבי הסגנונות המוזיקליים האהובים עליהם.



תרשים 1-16

באתר www.PlanetWrox.com תוכל למצוא דוגמה מקוונת שניתן להפעילה. תוכל להשתמש עם האתר מנקודת המבט של משתמש קצה.

באפשרותך גם להוריד את קוד המקור של היישום לדוגמה, ושל כל שאר הדוגמאות שבספר, מהאתר של הוצאת הספרים "הוד-עמי" בכתובת: www.hod-ami.co.il (היכנס לדף הספר ומצא את הלינק "קוד מקור") או מהאתר של Wrox שכתובתו <http://p2p.wrox.com/>. ראה פרטים נוספים בהקדמה.

שים לב: יכול להיות שמראה האתר השתנה במקצת מזה המוצג בספר. דבר זה אינו צריך להפריע לך בעבודה ובהבנה.

כאשר תסיים את הלימוד בספר, יהיו בידך הכלים הדרושים לבניית אתר זה. אל תדאג, גם אם זה נשמע כאוסף של דברים מורכבים, אני אנחה אותך כמיטב יכולתי, צעד אחר צעד, מתחילת היישום ולאורך כל הדרך עד להוספת התכונה האחרונה שתבחר. כל עוד אתה נהנה לעשות זאת, אני בטוח שתצליח לסיים את הדרך.

עצות מעשיות לגבי Visual Web Developer

רוב הפרקים בספר מסתיימים בקטע קצר של עצות מעשיות. ייתכן שעצות אלו לא השתלבו במקום כלשהו בטקסט שבפרק, או שמטרתן לעודד אותך להוסיף לחקור או לבחון דברים. לפעמים הן עלולות להיראות בלתי-רלוונטיות או קשות להבנה במבט ראשון, אבל תיווכח לדעת שככל שתתקדם בספר ותביט לאחור על עצות מפרקים קודמים, הדברים יתחילו להיות הגיוניים. אל תדאג אם אינך מבין דברים מסוימים במלואם בפעם הראשונה שאתה פוגש בהם. הקדש מעט מחשבה וחזור לנושא כמה ימים לאחר מכן. התקווה היא שעליך לאפשר לרעיונות 'לשקוע', ואז הדברים יתחילו להיות מובנים באופן אוטומטי. האמור לעיל נכון לא רק לקטע העצות המעשיות, אלא לגבי הספר כולו.

- ⊙ לפני שאתה עובר לפרק הבא, שחק עוד קצת עם VWD. הוסף כמה דפים לאתר שלך. גרור ושחרר כמה פקדים מ-Toolbox על הדפים שלך וצפה בהם בדפדפן שלך. בדרך זו, כאשר תתחיל את הפרק הבא תהיה לך הבנה טובה יותר של הכלים ושל הפקדים הרבים הזמינים.
- ⊙ ערוך היכרות עם האפשרויות הרבות להגדרת Visual Web Developer. כאשר תבנה אתרי אינטרנט תבלה את מרבית זמנך בסביבת הפיתוח, ולכן הגיוני להתאים אותה ככל האפשר לטעמך. אל תחשוש לבלבל את הסדר, כי תמיד תוכל לחזור להגדרות הקודמות.
- ⊙ הקדש זמן מה לדפדוף בין ההגדרות שתמצא בתיבת הדו-שיח Options של VWD (נגישה דרך התפריט Tools ← Options). רבות מההגדרות מסבירות את עצמן, ויכולות ממש לעזור בהתאמת סביבת הפיתוח לנוחיותך.

סיכום

פרק זה עסק בתחומים חשובים רבים במטרה לתת לך התחלה טובה עם ASP.NET 3.5 ועם VWD. הוא התחיל בהיסטוריה קצרה של סביבת העבודה NET של מיקרוסופט בכלל, ושל ASP.NET בפרט.

לאחר מכן למדת איך להשיג ולהתקין את Visual Web Developer 2008 Express Edition. VWD היא הכלי הזמין הנרחב והרב-תכליתי ביותר ליצירת דפי אינטרנט (web) בסביבת ASP.NET 3.5. כדי לאפשר לך להשתמש בה ביעילות, הראה לך פרק זה כיצד להשתמש ולהתאים אישית את התכונות העיקריות של סביבת הפיתוח. בפרקים הבאים תשתמש ותרחיב ידע זה כדי לעבוד עם הכלים הרבים שמספקת VWD.

חשוב להבין כיצד דף ב-VWD מוצא את דרכו אל הדפדפן שלך. מידע על הדרך בה שרת האינטרנט משרת את הבקשה, וכיצד הדף מעובד כדי לספק את ה-HTML הסופי שמגיע אל הדפדפן, הוא חיוני להבנת ASP.NET. פרק זה נתן לך מבוא קצר על הדרך שבה דף אינטרנט מבוקש ומוגש לדפדפן.

בפרק הבא תקבל הסבר מפורט הרבה יותר על יצירת אתרי אינטרנט.

תרגילים

1. הסבר את ההבדלים בין סימני העריכה של דף ב-VWD (בזמן פיתוח), לבין דף HTML הסופי בדפדפן.
2. הסבר את ההבדלים בין HTML לבין XHTML. מה הקשר בין שתי השפות?
3. נניח שיש בידך כמה קטעי HTML שאתה צופה שתשתמש בהם רבות במקומות שונים באתר. מהי הדרך הטובה ביותר לגרום לכך שקטעים אלה יהיו זמינים ב-VWD?
4. מנה שלוש מהדרכים שבהן תוכל לאפס חלק או כל ההגדרות המותאמות אישית בסביבת הפיתוח.
5. אם ברצונך לשנות מאפיין של פקד בדף, למשל הטקסט של פקד Button, מהן שתי האפשרויות שעומדות לרשותך לבצע את השינוי?

פרק 2

בניית אתר אינטרנט בסביבת ASP.NET

כדי ליצור אתרי אינטרנט פונקציונליים בעלי ממשק משתמש עשיר, עליך להבין כמה טכנולוגיות ושפות חשובות, כולל HTML, ASP.NET, CSS (Cascading Style Sheets) ו-JavaScript. פרק זה והבאים אחריו מספקים בסיס איתן לטכנולוגיות אלו, ועל כן, כאשר תסיים את הלימוד בספר תרגיש בנוח עם המושגים החשובים ביותר.

פרט לטכנולוגיות הנזכרות לעיל עליך גם להבין את סביבת הפיתוח Visual Web Developer אשר הוצגה בפרק הקודם. עליך לדעת כיצד ליצור אתרים, להוסיף דפים, ולנהל את כל סרגלי הכלים והחלונות שהסביבה מציעה לך. בנוסף, עליך לדעת איך לבנות ולעצב דפי אינטרנט ב-VWD באמצעות HTML ופקדי צד שרת.

פרק זה מראה לך בפירוט כיצד ליצור ולנהל אתרי אינטרנט. הוא גם מראה איך ליצור דפי אינטרנט בסביבת ASP.NET ואיך להוסיף להם סימני עריכה, דבר שיאפשר לך ליצור דפים שימושיים שיכולים להציג מידע למשתמש ולהגיב בהתאם לבקשותיו.

פרק זה בוחן במיוחד את הנושאים הבאים:

- ⊙ שני סוגים שונים של פרויקטים לבניית אתרי אינטרנט בסביבת ASP.NET.
- ⊙ התבניות השונות הזמינות לפרויקט כדי להתחיל איתם את פיתוח האתר, וכיצד להשתמש בהן.
- ⊙ סוגי הקבצים הרבים והשונים הזמינים ב-ASP.NET ולמה הם משמשים.
- ⊙ הדרכים ליצירת אתרי אינטרנט מובנים וקלים לניהול, עכשיו ובעתיד.
- ⊙ איך להשתמש בכלי העיצוב ליצירת דפי אינטרנט מעוצבים.

למרות שבפרק הקודם כבר יצרת את אתר האינטרנט הראשון שלך בסביבת ASP.NET, פרק זה מתחיל עם עוד מבט מעמיק על תהליך יצירת אתר אינטרנט חדש. כאשר מתחילים לבנות אתר אינטרנט חדש אנו עומדים בפני אפשרויות בחירה רבות, וכדי לבחור באפשרות המתאימה עבור התסריט שלך, חשוב להבין את כל אפשרויות הבחירה השונות.

יצירת אתרים באמצעות VWD 2008

בפרק הקודם ניתנה סקירה מהירה על יצירת אתר אינטרנט עם VWD. מתפריט File בחרת New Web Site, לאחר מכן בחרת בתבנית הרגילה ASP.NET Web Site, בחרת שפה ולחצת OK. אבל, תיבת הדו-שיח New Web Site מכילה יותר ממה שראית בפרק הקודם. ייתכן ששמת לב לכך שאתה יכול לבחור מבין כמה תבניות שונות, אשר מאפשרות לך ליצור סוגים שונים של אתרים. אבל לפני שתתבונן בתבניות השונות שעליהן תוכל לבסס את האתר החדש שלך, עליך לדעת מעט יותר על סוגי הפרויקטים השונים הזמינים ב-VWD.

סוגים שונים של פרויקטים

ייתכן שתהיה לך אפשרות בחירה בין שני סוגים שונים של פרויקטים, הדבר תלוי בגרסת VWD שאתה משתמש בה. אם אתה משתמש ב- Visual Web Developer Express Edition עם Service Pack 1 או משתמש באחת מהגרסאות המסחריות של Visual Studio 2008 יש לך שתי אפשרויות: **Web Site Projects** ו-**Web Application Project**. אלה שני סוגי הפרויקט שנדרון בהם כאן.

הערה: אם אתה עובד עם גרסת Visual Web Developer Express Edition ללא Service Pack 1 תוכל לעבוד רק עם פרויקט מסוג **Web Site Projects**.

Web Site Projects

Web Site Project מייצג ב-VWD פרויקט אתר אינטרנט. אתה יוצר Web Site Project חדש על ידי בחירת File ← New Web Site, או File ← New Project ← Web Site, מהתפריט הראשי של Visual Web Developer.

Web Site Project הוכנס לשימוש ב- Visual Studio 2005 ואפשר גמישות רבה ביצירת אתרי אינטרנט ובעבודה אתם. בניגוד לאתרים שנבנו עם גרסאות קודמות של Visual Studio.NET, אתר של Web Site Projects הוא למעשה תיקיית Windows עם קבוצה של קבצים ותת-תיקיות בתוכה. אין קובץ מרכזי (ידוע כקובץ הפרויקט ובעל סיומת .vbproj או .csproj). אשר עוקב אחר כל הקבצים הפרטיים באתר. כל שעליך לעשות הוא להפנות את VWD לתיקייה, ו-VWD מייד פותחת אותה כאתר אינטרנט. הדבר מקל מאוד על יצירת עותקים של האתר, הזזתם, או אפילו שיתופם עם אחרים, מכיוון שאין תלות ביניהם לבין קבצים אחרים שנמצאים במחשב המקומי שלך. מכיוון שאין קובץ מרכזי לפרויקט, מתייחסים בדרך כלל אל Web Site Projects כאל **Web Sites**, וזהו המונח שנשתמש בו מכאן ואילך.

מלבד תגובות אוהדות רבות למהלך זה, קיבלה מיקרוסופט גם תגובות שליליות רבות ממפתחים שהתלוננו כי Web Site Projects הגבילו מדי את סביבת הפיתוח שלהם. מכיוון שאין קובץ מרכזי שעוקב אחר כל האתר, קשה הרבה יותר להרחיק קבצים או תיקיות מהאתר ולעבוד עם מערכת לניהול תצורה - מערכת מרכזית שמאפשרת למפתחים לעבוד על פרויקט בשיתוף פעולה, ועוקבת באופן אוטומטי אחר שינויים. בנוסף, Web Site Projects השפיעו על הדרך

שבה אתרי אינטרנט מהודרים ומופצים, והדבר הקשה על מפתחים שהיו רגילים למודל הקודם, ליישם את הידע והכישורים שלהם לפרויקט מהסוג החדש.

בתגובה לביקורת שחזרה מיקרוסופט במאי 2006 את Web Application Projects כתוספת ל- Visual Studio 2005 standard edition ולגרסאות שבאו לאחר מכן. תחילה, Web Application Projects לא היו זמינים עבור Visual Web Developer Express Editions, כולל Visual Web Developer 2008 החדשה. אך עם שחרור SP1 יש ביכולתך להשתמש גם ב- Web Application Projects.

Web Application Projects

בגרסה החדשה של Visual Studio 2008 משולב Web Application Project לחלוטין בסביבה. למעשה, יוצרים פרויקט יישום אינטרנט, להבדיל מפרויקט אתר אינטרנט שהוצג קודם. הדבר מקל על מפתחים שעובדים בצוותים, או על אלה שזקוקים ליתר בקרה על תכני האתר ועל תהליכי ההידור וההפצה שלהם.

אתה יוצר Web Application Project חדש ב- Visual Studio 2008, מתיבת הדו-שיח שנפתחת בעקבות New Project ← File. בתיבת דו-שיח זו, ציין את שפת התכנות המועדפת עליך (Visual Basic או Visual C#) ואז לחץ על הקטגוריה Web, שם תמצא את התבנית: ASP.NET Web application.

בספר זה נשתמש באופן בלעדי במודל Web Site Projects, וכך במידה ולא התקנת את התוסף Service Pack 1 עדיין תוכל לעקוב אחר כל הדוגמאות.

עכשיו, כשאתה יודע אודות סוגי הפרויקט השונים, הדבר הבא שיש להביאו בחשבון הוא תבניות האתר השונות והאפשרויות שלהן.

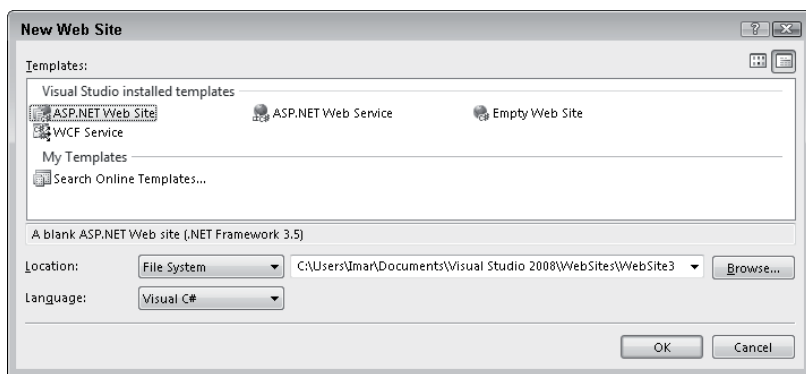
בחירת תבנית האתר הנכונה

תיבת הדו-שיח New Web Site ב-VWD מכילה תבניות שונות של אתרי אינטרנט, שכל אחת מהן משרתת מטרה ייחודית.

תרשים 1-2 מציג את תיבת הדו-שיח New Web Site ב-VWD (שים לב שהתרשים שלך כאן ובמשך הלימוד עלול להיראות שונה, על פי הגרסה של סביבת הפיתוח). ניתן לפתוח תיבת דו-שיח זו על ידי בחירת File ← New Web Site, או File ← New Project ← Web Site, תלוי בגרסת VWD שבשימושך.

החלק העליון של האזור Templates מראה את תבניות האתר שמותקנות כברירת מחדל בסביבת ASP.NET. כל אחת מהן נדונה בקטע הבא. החלק השני, עם התווית My Templates, מכיל קישור לחיפוש מקוון אחר תבניות. בנוסף, לאחר שיצרת תבניות משלך (דבר שתלמד לעשותו בפרק 6), או שיש לך תבניות מותקנות ממקורות אחרים, הן מופיעות באזור זה.

מרשימת התבניות המובנות הזמינות, נשתמש בספר בתבנית ASP.NET Web Site בלבד. השאר מתוארות בקצרה בסעיפים הבאים, וכך תדע כיצד להשתמש בהן.



תרשים 2-1

הערה: התרשים עלול להיראות שונה על פי הגרסה של סביבת הפיתוח.

ASP.NET Web Site

תבנית זו מאפשרת להגדיר אתר אינטרנט בסיסי בסביבת ASP.NET. היא מכילה קובץ תצורה של ASP.NET: web.config, דף אינטרנט בשם Default.aspx כולל קובץ הקוד, ותיקייה ריקה בשם App_Data. סוגי הקבצים השונים נדונים בהמשך הפרק, התיקייה App_Data תידון בפרק 11. תבנית זו היא נקודת התחלה טובה לכל אתרי האינטרנט שלך בסביבת ASP.NET.

ASP.NET Web Service

התבנית ASP.NET Web Service היא נקודת ההתחלה עבור אתרים חדשים שמכילים שירותי אינטרנט (web services). שירות אינטרנט מאפשר לך ליצור תוכנה בשרת האינטרנט אשר זמינה לשרת יישומים אחרים שממוקמים על אותה מכונה, או על מחשבים אחרים אי שם ברשת או באינטרנט. כאשר אתה יוצר אתר שמבוסס על תבנית זו אתה מקבל קובץ שירותי אינטרנט, קובץ קוד נוסף שנקרא על שם השירות, וקובץ הגדרות web.config לשימוש שירותי האינטרנט. בפרק 10 תלמד כיצד ליצור ולצרוך שירותי אינטרנט מהדפדפן.

WCF Service

התבנית WCF Service דומה במקצת לתבנית Web Service בכך שהיא מאפשרת ליצור אתר אינטרנט שכולל שירותים שניתן לפנות אליהם על פני רשת. עם זאת, Windows Communication Foundation Services מרחיקים לכת הרבה יותר מאשר שירותי אינטרנט פשוטים, ומציעים גמישות רבה יותר. WCF Services הם מחוץ לתחום ספר זה, אבל אם ברצונך ללמוד עליהם, תוכל למצוא ברחבי הרשת מאמרים רבים בנושא.

תבנית זו נותנת לך בדיוק את מה שמרמז שמה: כלום. כל מה שתקבל הוא אתר אינטרנט ריק כנקודת התחלה. התבנית 'אתר אינטרנט ריק' שימושית כאשר יש לך אוסף של קבצים קיימים שאתה רוצה להשתמש בהם ליצירת אתר אינטרנט חדש.

למרות שלכאורה יש לבחור מראש ובאופן ברור בתבנית הנכונה, זה ממש לא המקרה המתאים. מכיוון שאתר אינטרנט בסביבת ASP.NET ב-VWD הוא למעשה רק הפניה לתיקייה, קל להוסיף סוגי קבצים מתבנית אחת לאחרת. לדוגמה, ניתן להוסיף קובץ שירות אינטרנט לאתר אינטרנט בסביבת ASP.NET, כפי שתראה בפרק 10.

יצירה ופתיחה של אתר אינטרנט חדש

יש כמה דרכים ליצירת אתרי אינטרנט חדשים ולפתיחת אתרים קיימים. הבחירות שעומדות בפניך כאן מושפעות במידה רבה מהדרך שבה אתה ניגש לאתר (מהמחשב המקומי או ממחשב מרוחק), והאם אתה רוצה להשתמש בשרת האינטרנט המובנה שמגיע עם VWD או בשרת האינטרנט שמגיע עם Windows.

כל הדוגמאות בספר זה מניחות שאתה פותח אתרים מהמחשב המקומי שלך, ומשתמש בשרת האינטרנט המובנה, משום שנוח מאוד לפתח אתרים בעזרתו. עם זאת, בפרק 18 תלמד איך להגדיר תצורה ולהשתמש ב-IIS (Internet Information Services), שרת האינטרנט המתקדם שכלול כמעט בכל המהדורות של Windows. בדרך כלל משמש IIS באתרי ייצור, מכיוון שהוא בעל יכולת לשרת דפי אינטרנט בתרחישים של תנועה כבדה.

יצירת אתרים חדשים

תרגיל נסה זאת הבא מנחה אותך ליצור את האתר Planet Wrox, שהוא הפרויקט שעליו תעבוד בספר זה. כל התרגילים במהלך הספר מניחים שאתר זה פתוח ב-VWD, אלא אם נאמר אחרת. התרגיל מורה לך לאחסן את האתר בדיסק המקומי: C:\BegASPNET\Site בתיקייה בשם: C:\BegASPNET\Site. שים לב לשם התיקייה, מפני שנשתמש בשם זה במהלך הלימוד. אם תחליט להשתמש בתיקייה אחרת, אזי בכל פעם שתראה בספר את שם התיקייה, עליך לשנות בהתאם לפי המיקום שבחרת.

נסה זאת יצירת אתר אינטרנט חדש בסביבת ASP.NET 3.5

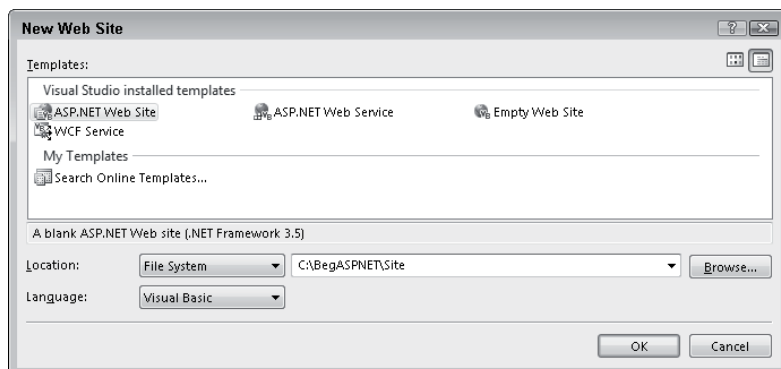
1. אם עקבת אחר ההוראות בהקדמה והורדת את קבצי המקור של הספר, התיקייה BegASPNET כבר קיימת אצלך וכוללת את תיקיות המשנה Source ו-Resources. אם לא, חזור להקדמה ועקוב אחר ההוראות.

בתוך התיקייה BegASPNET, צור תיקיית משנה בשם **Site**.

2. הפעל את Visual Web Developer 2008 ובחר File ← New Web Site, או File ← Web Site ← New Project, בהתאם לגרסת VWD שלך.

3. ודא שבחלק Templates אכן נבחר ASP.NET Web Site.

4. ודא שתחת Location נבחרה האפשרות File System.
5. לחץ על הלחצן Browse שסמוך לרשימה הנפתחת Location, דפדף לתיקייה C:\BegASPNET\Site, ולחץ Open.
6. ברשימה הנפתחת Language בחר בין Visual Basic לבין Visual C#. כל הדוגמאות בספר מופיעות בשתי השפות, וכך אתה יכול לבחור בזו החביבה עליך. המסך הסופי שלך צריך להיראות בדומה לזה שבתרשים 2-2, פרט אולי לרשימה הנפתחת Language במקרה שבחרת Visual C#.
7. לחץ OK ו-VWD תיצור עבורך אתר חדש.

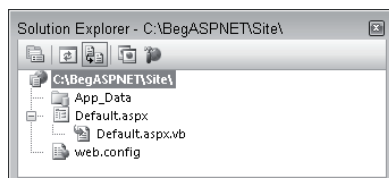


תרשים 2-2

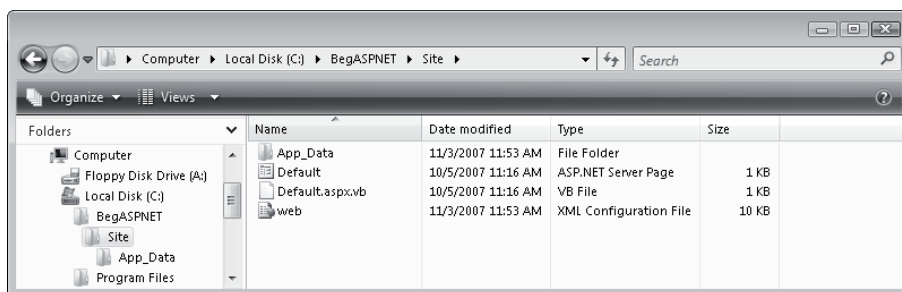
כיצד זה פועל

ברגע שלחצת OK, יצרה עבורך VWD אתר חדש. אתר אינטרנט חדש זה מכיל דף אינטרנט בשם Default.aspx כולל קובץ הקוד Default.aspx.vb או Default.aspx.cs (לפי השפה שבחרת), קובץ תצורה בשם web.config, ותיקייה ריקה App_Data. בסייר הפתרון האתר שלך מוצג כמו תרשים 2-3 (שבו הורחב Default.aspx כדי שתוכל לראות גם את קבצי הקוד).

מכיוון שאתר אינטרנט שמבוסס על התבנית Web Site Project הוא רק תיקיית Windows פשוטה ש-VWD מתבונן בה, התיקייה הממשית על הדיסק מכילה בדיוק את אותם קבצים. לא נעשה שימוש בקבצים נוספים לצורך יצירת האתר, כפי שרואים בתרשים 2-4, אשר מציג את הקבצים בתיקייה C:\BegASPNET\Site.



תרשים 2-3

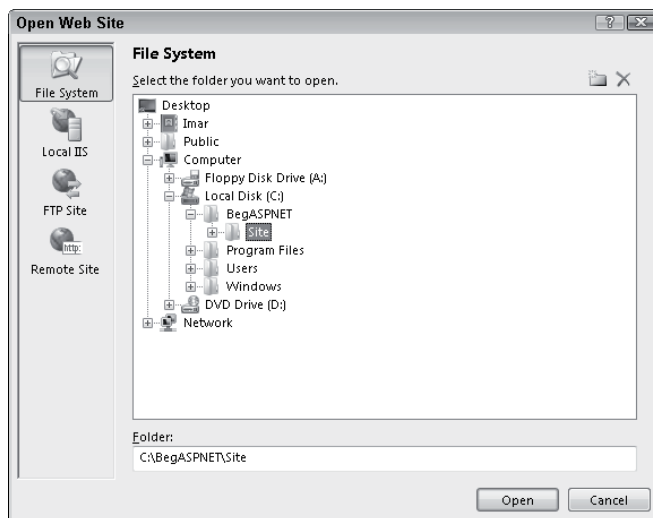


תרשים 2-4

פתיחה של אתרי אינטרנט קיימים המבוססים על התבנית Web Site Project דומה מאוד ליצירת אתרים חדשים. בסעיף הבא תקבל סקירה מהירה על פתיחת אתרים קיימים על ידי VWD.

פתיחת אתרי אינטרנט קיימים

בדיוק כמו עם יצירת אתרים חדשים, פתיחת אתר קיים על ידי VWD נותנת כמה אפשרויות בחירה למיקום המקור של אתר האינטרנט. אתה יכול לפתוח את האתר ממערכת הקבצים המקומית, משרת IIS מקומי, משרת מרוחק תוך שימוש ב-FTP, או מאתר מרוחק תוך שימוש ב-Microsoft FrontPage Server Extensions. תרשים 2-5 מראה את תיבת הדו-שיח Open Web Site ב-VWD.



תרשים 2-5

כל הדוגמאות בספר מניחות שאתה תמיד פותח את האתר Planet Wrox המקומית, על ידי שימוש בלחצן File System שהוא הלחצן הראשון בטור השמאלי של החלון. לאחר מכן מצא את האתר שלך בחלונת הימנית (C:\BegASPNET\Site בדוגמה זו), ולחץ על הלחצן Open.

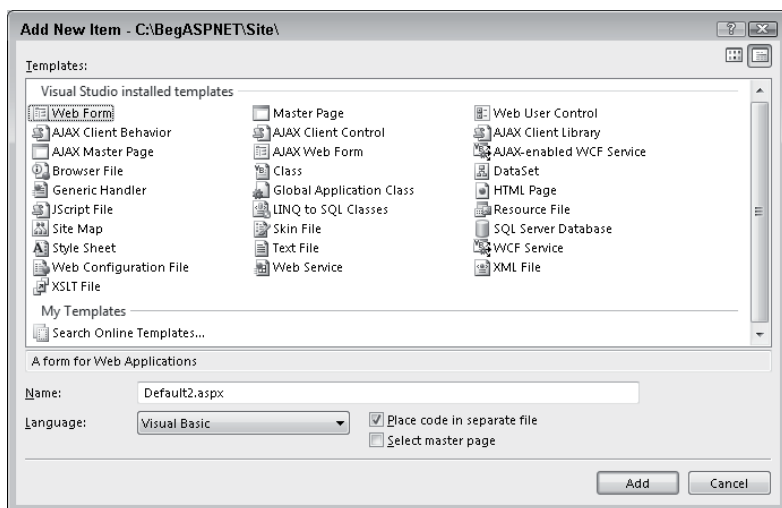
האתר שיצרת בתרגיל נסה זאת הקודם הוא אתר ללא "בשר", מכיוון שהוא מכיל דף אינטרנט אחד בלבד. כדי להפוך את האתר לשימושי יותר, תצטרך להוסיף לו קבצים. ישנם סוגי קבצים רבים שעומדים לרשותך, והדרך שבה מוסיפים אותם לאתר, הם הנושא הבא שנדון בו.

עבודה עם קבצים באתר האינטרנט שלך

אתר אינטרנט בסביבת ASP.NET 3.5 מורכב מדף אינטרנט (קובץ עם סיומת aspx) אחד לפחות, אבל בדרך כלל הוא מכיל מספר גדול יותר של קבצים. סוגים שונים רבים של קבצים זמינים ב-VWD, וכל אחד מהם מציע פונקציונליות ייחודית. בקטע הבא תכיר את סוגי הקבצים החשובים ביותר שבשימוש VWD. בנוסף, תלמד כמה דרכים שונות להוספת קבצים אלה לאתר שלך.

סוגי הקבצים הרבים של אתר אינטרנט בסביבת ASP.NET 3.5

כדי שתקבל מושג בכמה קבצים שונים אתה יכול להשתמש ב-ASP.NET, תוכל לראות בתרשים 2-6 את תיבת הדו-שיח שמאפשרת להוסיף קבצים חדשים לאתר (התיבה נגישה על ידי לחיצה ימנית באתר על סייר הפתרון ובחירת Add New Item, או על ידי בחירת Website ← Add New Item מהתפריט הראשי).



תרשים 2-6

הערה: הרשימה יכולה להיראות קצת שונה בהתאם לגרסת ה-VWD שברשותך.

קבצים אלה יכולים להיות מקובצים בכמה קטגוריות שונות. הקבצים החשובים ביותר – אלה שבהם תשתמש בכל הדוגמאות בספר – נדונים להלן.

קבצי אינטרנט

קבצי אינטרנט (Web Files) הינם ייחודיים ליישומי אינטרנט. הם יכולים להיות מבוקשים ישירות על ידי דפדפן, או שמשמשים בהם לבניית חלק מהדף המבוקש על ידי הדפדפן. בטבלה הבאה מוצגת רשימת קבצי האינטרנט השונים והסיומות שלהם, ותיאור השימוש בכל קובץ.

סוג הקובץ	סיומת (extension)	תיאור
Web Form AJAX Web Form	.aspx	קבצים אלה הם 'סוסי העבודה' של כל אתר אינטרנט בסביבת ASP.NET. טפסי אינטרנט מייצגים את הדפים שהמשתמשים שלך צופים בהם בדפדפן. טופס אינטרנט מסוג AJAX דומה לטופס אינטרנט רגיל, אבל הוא הוכן לעבודה עם פקדי AJAX. על פקדים אלה תלמד בפרק 10. טפסי אינטרנט נדונים בפירוט מלא בהמשך הפרק.
Master Page AJAX Master Page	.master	קבצים אלה מאפשרים לך להגדיר את המבנה הכללי של אתר האינטרנט, את המראה שלו ואת התחושה שייצור אצל המשתמש. בפרק 6 תלמד איך להשתמש בהם. כמו עם טפסי אינטרנט מסוג AJAX. גם הדף הראשי מסוג AJAX מכיל קוד שדרוש לעבודה עם פקדי AJAX.
Web User Control	.ascx	פקד זה כולל קטעי דף שניתן לעשות בהם שימוש חוזר בדפים רבים באתר. פרק 8 מוקדש כולו לפקד זה.
HTML Page	.htm / .html	אפשר להשתמש בדף HTML זה להצגת מידע סטטי באתר.
Style Sheet	.css	גיליון הסגנון מכיל קוד CSS שמאפשר לך לעצב ולסגנון את אתר האינטרנט שלך. בפרק הבא תלמד יותר על CSS.
Web Configuration File	.config	הקובץ מכיל מידע תצורה גלובלי שמשתמשים בו באתר כולו. מאוחר יותר בספר תלמד כיצד להשתמש ב- web.config.

סוג הקובץ	סיומת (extension)	תיאור
Site Map	.sitemap	מפת האתר, זהו קובץ XML הכולל תצוגה היררכית של קבצים באתר. מפת האתר משמשת לניווט. מידע נוסף ניתן בפרק 7.
JScript File	.js	הקובץ מכיל קוד JavaScript (שמייקרוסופט קוראת לו JScript) שיכול להתבצע בדפדפן הלקוח.
Skin File	.skin	הקובץ מכיל מידע עיצוב עבור פקדים באתר. קבצים מסוג זה נדונים בפרק 6.

תרגיל נסה זאת הבא מראה איך להוסיף דף מסוג Master Page חדש לאתר. דף זה משמש מכאן ואילך לכל אורך הספר.

נסה זאת הוספת קבצים לאתר שלך

1. פתח את האתר Planet Wrox שיצרת קודם (אם אינו פתוח עדיין) על ידי בחירת File → Open Web Site. ודא שאתה פותח את האתר מתפריט File System. מצא את התיקייה שמכילה את האתר שלך (C:\BegASPNET\Site) ולחץ על הלחצן Open.
2. בסייר הפתרון לחץ לחיצה ימנית על האתר שלך, ובחר Add New Item. ודא שאתה לוחץ על האתר עצמו ולא על אחד מהקבצים שנמצאים בו, אחרת לא תקבל את התפריט הנכון. לחילופין, באפשרותך לבחור File → New File מהתפריט הראשי של Visual Web Developer.
3. מתיבת הדו-שיח שמופיעה בחר Master Page. אתה יכול להשאיר את השם MasterPage.master. ודא שהרשימה הנפתחת Language מוגדרת לשפה שברצונך להשתמש בה באתר זה, וכי האופציה Place Code in Separate File מסומנת. לבסוף לחץ על הלחצן Add. הדף הראשי מתווסף לאתר, ונפתח באופן אוטומטי בחלון המסמך.

כיצד זה פועל

תרגיל פשוט זה הראה לך כיצד להוסיף פריט חדש לאתר האינטרנט שלך בשתי דרכים שונות. למרות שבשלב זה האתר אינו מלהיב במיוחד, הקובץ שהוספת מהווה את הבסיס לשלבי הלימוד הבאים בספר זה.

הסעיפים הבאים עוסקים בקצרה בשאר סוגי הקבצים. הוספת קבצים אלה לאתר זהה לדרך שבה הוספת קבצי אינטרנט. הטבלה הבאה מתארת את הסוגים השונים של קבצי קוד.

סוג הקובץ	סיומת (extension)	תיאור
Web Service	.asmx	הגישה לשירותי האינטרנט מתבצעת על ידי מערכות אחרות, כולל דפדפנים, השירות יכול להכיל קוד שניתן לבצעו בשרת שלך. שירותי האינטרנט מוסברים בפרק 10.
Class	.cs / .vb	הקובץ יכול להכיל קוד לתכנות אתר האינטרנט שלך. שים לב שגם קבצי קוד לדף האינטרנט (יידונו מאוחר יותר) הם בעלי אותה סיומת, מכיוון שהם למעשה קבצי מחלקה.
Global Application Class	.aspx	קבצים אלה יכולים להכיל קוד שנוצר בתגובה לדברים מעניינים שקורים באתר שלך, כמו הפעלת היישום או כאשר מתעוררת שגיאה במקום כלשהו באתר. יותר מאוחר תלמד איך להשתמש במחלקה זו.

פרט לקטגוריה 'קבצי קוד' יש עוד קבוצה אחת של קבצים שכדאי להתבונן בה: 'קבצי נתונים'.

קבצי נתונים

בקבצי נתונים משתמשים כדי לאחסן נתונים שיישומים אחרים באתר עשויים להשתמש בהם. הקבוצה מורכבת מקבצי XML ומקבצי בסיס נתונים.

סוג הקובץ	סיומת (extension)	תיאור
XML File	.xml	הקובץ משמש לאחסון נתונים בפורמט XML. בנוסף לקבצי XML רגילים, תומכת ASP.NET גם בכמה קבצים נוספים מבוססי XML. שניים מהם כבר ראית: web.config ומפת האתר.
SQL Server Database	.mdf	הקבצים בעלי הסיומת .mdf הם בסיסי נתונים ששרת SQL של מיקרוסופט משתמש בהם.
LINQ to SQL Classes	.dbml	קבצים אלו מאפשרים גישה אל בסיסי נתונים באופן הצהרתי, ללא צורך בכתיבת קוד. למעשה אלו לא קבצי נתונים רגילים, כי אינם מכילים את הנתונים הממשיים. עם זאת, מכיוון שיש להם קשר הדוק אל בסיס הנתונים, הגיוני לקבץ אותם תחת כותרת זו. תלמד יותר על LINQ to SQL בפרק 13.

כפי שראית בתרגיל נסה זאת האחרון, הוספת קובץ חדש מאחד מהסוגים לעיל היא פעולה קלה למדי. הדבר קל כמו להוסיף קובץ קיים לאתר.

הוספת קבצים קיימים

לא כל קובץ שאתה יוצר באתר האינטרנט שלך חייב להיות חדש לחלוטין. יש מקרים שבהם הגיוני לנצל קבצים מפרויקטים אחרים. לדוגמה, ייתכן שתמצא לעשות שימוש חוזר בלוגו או בקובץ CSS באתרים רבים שתקיים. בקלות ניתן להוסיף קבצים קיימים. עושים זאת בלחיצה ימנית על האתר בסייר הפתרון, ובחירת Add Existing Item. בתיבת הדו-שיח שמופיעה אתה יכול לדפדף כדי למצוא את הקובץ או הקבצים הרצויים, ולבחור מספר קבצים על ידי לחיצה עליהם ועל המקש Ctrl באותו זמן. לבסוף, כאשר תלחץ Add, יתווספו הקבצים לאתר האינטרנט.

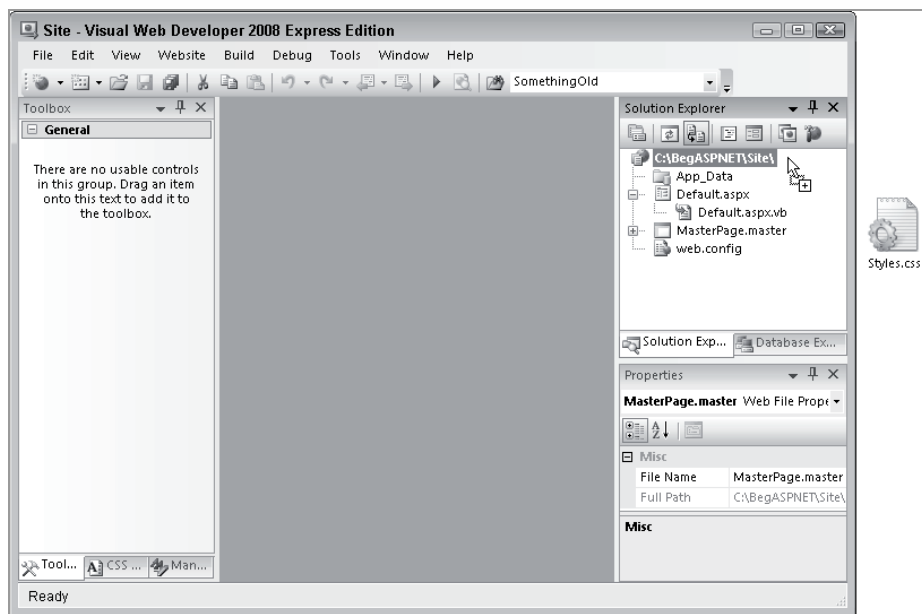
יש דרך קלה עוד יותר להוספת קבצים לאתר, שיכולה לתרום לחיסכון משמעותי בזמן כאשר ברצונך להוסיף קבצים קיימים רבים ותיקיות אל האתר: השתמש בתכונה 'גרור ושחרר'. תרגיל נסה זאת הבא מראה כיצד זה פועל.

נסה זאת הוספת קבצים קיימים לאתר שלך

1. לחץ לחיצה ימנית על שולחן העבודה של Windows ובחר New ← Text Document.
אם אינך רואה אפשרות זו, צור מסמך טקסט חדש על ידי Notepad.
2. קרא לקובץ הטקסט בשם **Styles.css**. ודא שהסיומת **txt** הוחלפה ב-**css**. עליך לראות את סמל הקובץ אשר משתנה מזה של קובץ טקסט לזה של קובץ CSS (זכור שבירת המחלף לקובץ זה היא סמל זהה לזה של קובץ טקסט, אבל עם סימן כלים מעליו, ייתכן שתוכנה כלשהי שהותקנה אצלך שינתה את הסמל של קבצי CSS). אם אינך רואה את שינוי הסיומת, פירושו שהגדרות של Windows גרמו להסתרת סיומות של סוגי קבצים מקובלים. אם זה המקרה, פתח את סייר Windows, ובחר Tools ← Folder Options (ב-Windows XP), או לחץ על הלחצן Organize (ב-Windows Vista) ואז בחר Folder and Search Options. בשני המקרים, עבור אל הכרטיסייה View ובטל את הבחירה באפשרות בעלת התווית Hide Extensions for Known File Types. ייתכן שתצטרך כעת לשנות את שם הקובץ מ-**Styles.css.txt** ל-**Styles.css**.
3. כאשר אתה משנה את שם הקובץ מ-**txt** ל-**css**, Windows עשויה להוציא אזהרה שהקובץ יהפוך לבלתי-שמיש אם תמשיך. במקרה כזה לחץ Yes.
4. סדר מחדש את VWD באופן שתוכל לראות חלק משולחן העבודה וגם את קובץ CSS. כדי להוציא את VWD ממצב מסך מלא, אתה יכול להשתמש בלחצן Restore Down שנמצא בסמוך ל-X-הסגירה הגדול והאדום שבחלק העליון של החלון.
4. לחץ על קובץ CSS שבשולחן העבודה, ובעודך לוחץ את לחצן העכבר, גרור את הקובץ אל סייר הפתרון. ודא שאתה אכן גורר את הקובץ אל סייר הפתרון, ולא אל חלקים

אחרים של VWD, כי אחרת הקובץ לא יתווסף. לדוגמה, אם תגרור את הקובץ אל חלון המסמך, VWD תפתח עבורך את הקובץ אבל לא תוסיף אותו לאתר.

5. כאשר תשחרר את העכבר בעורך על גבי אתר האינטרנט בסייר הפתרון (מוצג בתרשים 2-7), קובץ CSS יתווסף אל האתר שלך.



תרשים 2-7

כיצד זה פועל

מה שחשוב להבין מתרגיל זה הוא ש-VWD יוצרת עותק של הקובץ כשהיא מוסיפה אותו לאתר. לפיכך, הקובץ המקורי Styles.css שעל שולחן העבודה אינו מושפע משינויים שאתה עושה בעותק שב-VWD. בדרך זו קל לגרור קבצים מאתרי אינטרנט קיימים ולשחררם באתר החדש שלך מבלי להשפיע על המקור. הדבר תקף גם לגבי קבצים שאתה מוסיף תוך שימוש בתיבת הדו-שיח Add Existing Item של VWD.

אם הוספת לתיקיית האתר שלך קבצים שמקורם מחוץ ל-VWD, ייתכן שהם לא יופיעו מייד בסייר הפתרון. אתה יכול לרענן את הסביבה על ידי לחיצה על Refresh בסרגל הכלים של סייר הפתרון, הדבר יגרור לסנכרון בין ה-VWD לבין הספרייה.

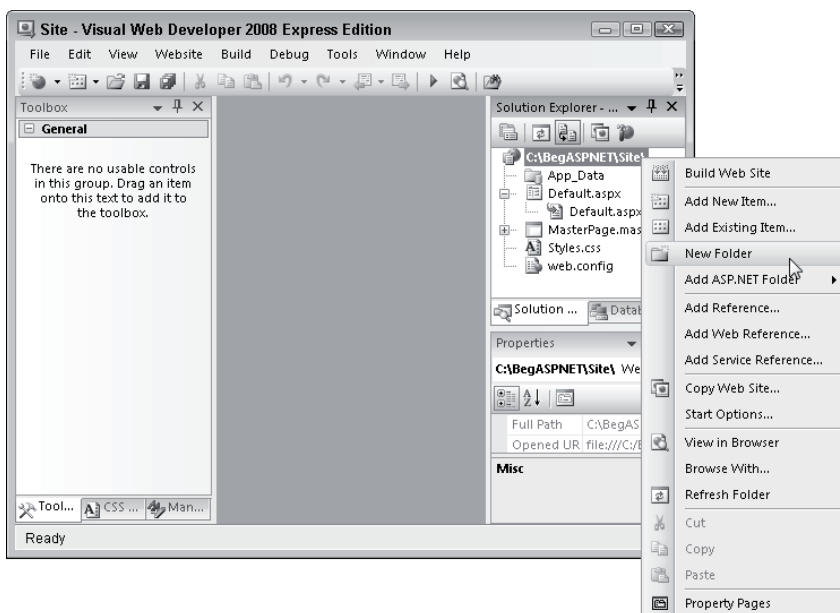
ארגון האתר שלך

בשל הקבצים הרבים שמרכיבים את האתר, מומלץ לקבץ אותם בתיקיות נפרדות בהתאם לפונקציה שהם ממלאים. לדוגמה, כל קבצי גיליון הסגנון (עם סיומת **.css**) יכולים להיכנס לתיקייה בשם Styles, קבצים בעלי סיומת js יכולים להיכנס לתיקייה בשם Scripts, פקדי

משתמש ייכנסו לתיקייה פקדים, ודפים ראשיים ניתן לאחסן בתיקייה MasterPages. זה כמובן עניין של העדפה אישית, אבל אתרים מובנים ומאורגנים קלים יותר לניהול, להבנה ולתחזוקה. תרגיל נסה זאת הבא מסביר כיצד להוסיף תיקיות חדשות לאתר, וכיצד להזיז קבצים ממקום אחד לאחר.

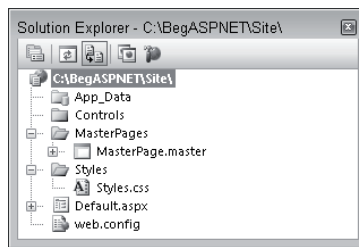
נסה זאת ארגון אתר האינטרנט שלך

1. לחץ לחיצה ימנית על האתר Planet Wrox בסייר הפתרון, ובחר New Folder, כפי שמוצג בתרשים 2-8.



תרשים 2-8

2. הקלד **MasterPages** עבור שם התיקייה החדשה, והקש Enter.
 3. צור שתי תיקיות נוספות, **Styles** ו-**Controls**, בהתאמה. תיקיות אלו יישמשו אותנו בהמשך הלימוד בספר.
 4. גרור את הקובץ MasterPage.master ושחרר אותו בתיקייה MasterPages שיצרת זה עתה. כך הקובץ מועבר מתיקיית הבסיס של האתר אל התיקייה המיועדת.
 5. גרור את הקובץ Styles.css משורש האתר ושחרר אותו בתיקייה Styles.
 6. אם הכל הלך כשורה, סייר הפתרון שלך צריך להיראות כמו תרשים 2-9.
- אם סייר הפתרון שלך נראה שונה מזה שמוצג בתרשים 2-9, עקוב אחר תרגיל זה שוב, עד שהאתר שלך ייראה זהה לחלוטין, עם אותו מבנה תיקיות וקבצים בתוכו. תרגילי נסה זאת הבאים בספר מניחים שהתיקיות והקבצים הנכונים נמצאים באתר.



תרשים 2-9

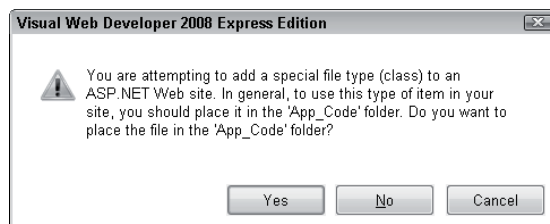
כיצד זה פועל

מבנה וארגון הם בעלי חשיבות רבה לשמירת האתר במצב בו ניתן לנהל ולתחזק אותו. למרות שאתה עשוי להתפתות להוסיף את כל הקבצים שלך לתיקיית הבסיס של הפרויקט, עדיף לא לעשות כך. באתר קטן מאוד, ייתכן שלא תרגיש בהבדל כלשהו, אבל ברגע שהאתר יתחיל לגדול תמצא שהרבה יותר קשה לנהל אותו כאשר הוא חסר מבנה. הצבת קבצים בעלי נושא משותף בתיקיות נפרדות היא הצעד הראשון לאתר מאורגן. אחסון קבצים מאותו סוג בתיקייה אחת היא רק דרך אחת לאופטימיזציה של האתר שלך. בפרקים מאוחרים יותר תלמד שתיקיות נפרדות משמשות גם לאיסוף קבצים בעלי פונקציונליות דומה. לדוגמה, כל הקבצים שנגישים אך ורק למנהלי האתר רצוי שיהיו מקובצים בתיקייה Management.

התכונה 'גרור ושחרר' של VWD מקלה מאוד על ארגון האתר. עליך 'לתפוס' קובץ או מספר קבצים ו'לשחרר' אותם במקומם החדש. אם תמשיך ליישם נהלים לארגון קבצים בעורך מרחיב את האתר תמצא, מחר או בעוד חצי שנה, שאין לך כל קושי לאתר את הקובץ הנכון כאשר אתה צריך אותו.

סוגים מיוחדים של קבצים

כמה מהקבצים שהוזכרו בסעיף הקודם חייבים להיות מוצבים בתיקייה מיוחדת ומתאימה להם, בשונה ממבנה התיקיות הארגוני שהוצע בסעיף הקודם. כאשר תנסה להוסיף קובץ לתיקייה שאינה מתאימה לו, תקבל אזהרה מסכיבת העבודה ויוצע לך ליצור תיקייה מתאימה ולשים בה את הקובץ. לדוגמה, כאשר תנסה להוסיף קובץ מסוג Class (עם סיומת vb או cs) לאתר, תופיע האזהרה שאתה רואה בתרשים 2-10.



תרשים 2-10

קבצים מסוג Class אמורים להימצא בתיקייה מיוחדת App_Code. כאשר תקבל תיבת דו-שיח כזו, תמיד לחץ Yes, אחרת הקובץ שלך לא יתפקד נכון (כאשר תלחץ Yes, סביבת העבודה תיצור עבורך את התיקייה App_Code אם היא לא קיימת). עבור סוגי קבצים אחרים, כולל קבצי skin ובסיסי נתונים, תקבל תיבות דו-שיח דומות.

עתה, כשיש לך הבנה טובה של סוגי הקבצים השונים אשר מרכיבים את האתר שלך, הגיע הזמן להתבונן באחד מהם בפירוט רב יותר: קבצי aspx, הידועים גם כטפסי אינטרנט.

לעבוד עם טפסי אינטרנט (Web Forms)

טפסי אינטרנט, שמיוצגים על ידי קבצים עם סיומת aspx, הם הליבה של כל יישום אינטרנט בסביבת ASP.NET 3.5. אלה הם הדפים הממשיים שהמשתמשים רואים בדפדפנים שלהם כאשר הם מבקרים באתר שלך.

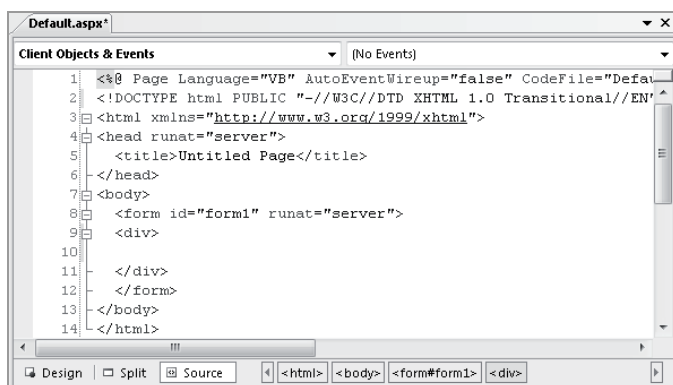
כפי שראית בפרק הקודם, טפסי אינטרנט עשויים להכיל תערובת של HTML, פקדי שרת של ASP.NET, JavaScript של צד לקוח, CSS וקוד תכנות. כדי להבין כיצד כל הקוד הזה מסיים את דרכו בדפדפן ומוצג בו, מציעה לך VWD כמה אפשרויות תצוגה של הדפים שלך.

אפשרויות תצוגה שונות בטפסי אינטרנט

VWD מאפשרת לך להתבונן בטפסי האינטרנט מכמה זוויות שונות. כאשר בחלון המסמך נמצא קובץ HTML או ASPX פתוח, אתה רואה שלושה לחצני צפייה בתחתית החלון. בעזרת לחצנים אלה, שנראים בתרשים 11-2, אתה יכול לעבור בין התצוגות השונות.

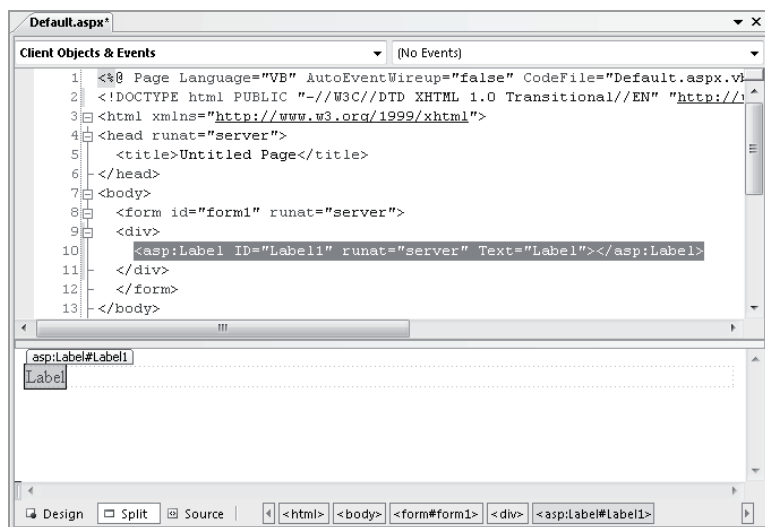
כאשר אתה פותח דף, תקבל את תצוגת ברירת המחדל – Source View. תצוגת Source View מציגה את ה-HTML הגולמי וסימני עריכה אחרים עבור הדף. תצוגה זו שימושית מאוד כאשר ברצונך להגדיר את תכני הדף, ואתה יודע היטב מה ברצונך לשנות והיכן. כפי שהוסבר בפרק הקודם, אני משתמש במונח Markup View ולא ב-Source View כדי להתייחס לסימני העריכה של דפי ASPX ו-HTML.

הלחצן Design מאפשר להעביר את חלון המסמך ל-Design View, שנותן מושג כיצד הדף ייראה בסופו של דבר. כאשר אתה ב-Design View, באפשרותך להשתמש בתת-התפריטים Visual Aids ו-Formatting Marks מהתפריט הראשי של View, כדי לשלוט בסימנים חזותיים כמו מעברי שורה, גבולות ומרווחים. שני תת-התפריטים מציעים תת-תפריט בשם Show שמאפשר להפעיל את כל העזרים החזותיים או לכבותם בבת אחת. כיכוי השניים שימושי כאשר ברצונך לקבל מושג איך הדף שלך יוצג בדפדפן. עם זאת, אתה צריך להשתמש ב-Design View רק כדי לקבל מושג איך הדף יוצג. למרות ש-VWD מציגה את הדף ב-Design View באופן טוב למדי, אתה חייב תמיד לבדוק את הדפים שלך גם בדפדפנים אחרים מפני שמה שאתה רואה ב-VWD הם סימני העריכה עבור הדף לפני שהוא מעובד. פקדי שרת בדף עשויים לפלוט HTML שמשנה את המראה של הדף בדפדפן. לכן מומלץ לצפות בדף בדפדפן לעתים קרובות ככל האפשר כדי לבדוק אם הדף ייראה כפי שאתה מצפה.



תרשים 2-11

הלחצן Split מאפשר לצפות ב- Design View וב- Markup View בו-זמנית, כפי שאתה רואה בתרשים 2-12.



תרשים 2-12

Split View נהדר כשאתה רוצה לראות את הקוד ש-WVD יוצרת לאחר הוספת פקדים אל Design View של הדף שלך. גם הדרך ההפוכה שימושית מאוד: כשאתה עושה שינויים בסימני העריכה של דף ב- Markup View, אתה יכול לראות איך הדף יוצג ב- Design View. יש זמנים שבהם ה- Design View מפסיק להיות מסונכרן עם Markup View. אם זה המקרה, תופיע הודעה בראש חלון Design View. לחיצה על ההודעה או שמירת הדף כולו מספיקים כדי לסנכרן בין השניים.

אם ברצונך שהדפים שלך ייפתחו בתצוגה שונה מאשר Markup View, בחר Tools ← Options. לאחר מכן בחר בקטגוריית HTML Designer, ובכרטיסייה General הגדר את התצוגה שאתה מעדיף. בנוסף ל-HTML ולסימני עריכה שאתה רואה בחלון Markup View, יכול טופס אינטרנט להכיל גם קוד בשפת C# או בשפת Visual Basic.NET. המיקום של קוד זה תלוי בסוג טופס האינטרנט שאתה יוצר. בסעיף שלהלן ניתן הסבר מפורט של שתי האפשרויות העומדות בפניך.

בחירה בין קוד ברקע לבין דפים עם קוד כלול

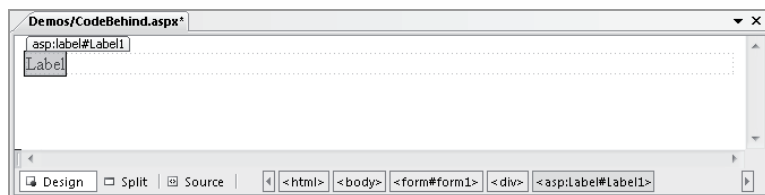
טפסי אינטרנט מופיעים באחת משתי צורות: כקובץ `aspx` עם קובץ קוד ברקע (קובץ בעל שם זהה לזה של טופס האינטרנט, אבל עם סיומת נוספת, `vb` או `cs`), או כקבצי `aspx` שהקוד שלהם מוטמע בתוכם (לעיתים נקראים קוד כלול **(inline code)**). למרות שלא תראה הרבה קוד עד לפרק 5, חשוב להבין את ההבדל בין שני סוגים אלה של טפסי אינטרנט. בתחילה נראים טפסי אינטרנט עם קוד כלול כקלים יותר להבנה. מכיוון שהקוד הדרוש לתכנות אתר האינטרנט שלך הוא חלק מאותו טופס עצמו, אתה יכול לראות בבירור את הקשר בין הקוד לבין הקובץ. עם זאת, כאשר הדף הולך וגדל ואתה מוסיף לו עוד פונקציונליות, הרי שלעיתים קרובות קל יותר לעבוד כאשר הקוד נמצא בקובץ נפרד. בדרך זו הוא מובדל לחלוטין מסימני העריכה, ואתה יכול להתמקד במשימה שלשמה נועד הקוד.

בתרגיל הבא תוסיף שני קבצים שמדגימים את ההבדל בין קוד ברקע לבין קוד כלול.

נסה זאת הוספת טפסי אינטרנט עם קוד לאתר שלך

היישום הסופי אינו זקוק לקבצים שאתה עומד להוסיף בתרגיל זה. כדי למנוע בלבול בפרויקט, עליך להציב אותם בתיקייה נפרדת בשם `Demos`.

1. בסייר הפתרון, לחץ לחיצה ימנית על האתר שלך ובחר `New Folder`. קבע לתיקייה את השם `Demos` והקש `Enter`.
 2. לחץ לחיצה ימנית על התיקייה `Demos` ובחר `Add New Item`. בתיבת הדו-שיח שמופיעה לחץ על התבנית `Web Form` וקרא לקובץ בשם `CodeBehind.aspx`. ודא שנבחרה תיבת הסימון `Place Code in Separate File`, ובחר בשפת התכנות שלך מהרשימה הנפתחת. לבסוף, לחץ על הלחצן `Add`. הדף צריך להיפתח ב- `Markup View` כדי שתוכל לראות את קוד `HTML` שלו.
 3. בחלק התחתון של חלון המסמך לחץ על הלחצן `Design`, כדי להעביר את הדף מתצוגת `Markup View` לתצוגת `Design View`. הדף שאתה רואה הוא בעל רקע לבן עם מלבן קטן מקווקו בחלקו העליון. המלבן המקווקו הוא התגית `<div>` שראית ב- `Markup View`.
 4. מתיבת הכלים, גרור פקד `Label` מהקטגוריה `Standard` ושחרר אותו בשטח המקווקו של הדף. אם תיבת הכלים אינה פתוחה עדיין פתח אותה בעזרת קיצור הדרך `Ctrl+Alt+X`.
- ב- `Design View` המסך שלך צריך להיראות כעת בדומה לתרשים 2-13.



תרשים 2-13

5. לחץ לחיצה כפולה במקום כלשהו בשטח הלבן שמתחת לקו המקווקו של התגית <div>. כעת VWD עוברת מ- Design View אל Code Behind (קוד ברקע) של הקובץ, ומוסיפה קוד שיוורן במהלך טעינת הדף לדפדפן:

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Me.Load  
  
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)  
{  
  
}
```

נעסוק בהבנת תחביר הקוד בשלב מאוחר יותר. ברוב המקרים VWD מוסיפה אותו באופן אוטומטי, כפי שראית זה עתה. בפרקים מאוחרים יותר תראה בדיוק איך קוד זה פועל, אך כעת חשוב להבין שהקוד שאתה עומד להציב בין השורה שמתחילה ב- Protected Sub לבין זו שמתחילה ב- End Sub (ב-VB), ובין הסוגריים המסולסלים (ב-C#), יופעל כאשר הדפדפן יבצע פנייה לבקשת קובץ זה מהשרת. אם אתה משתמש ב-VB, הקו התחתון (.) שנראה בקטע הקוד של VB.NET לא יופיע. הוספתי אותו כאן כדי לפצל את הקוד על פני שתי שורות. את הסיבה לכך תמצא בסעיף 'כיצד זה פועל' שלאחר תרגיל זה.

כל דוגמאות הקוד שתראה מכאן ואילך כוללות גרסה ב- Visual Basic (VB.NET), וגרסה ב-C#, בחר תמיד בזו שתואמת את שפת התכנות שלך.

6. הצב את הסמן בשורה הפתוחה בקוד שיצרה VWD, והוסף את שורת הקוד המוצגת, שמקצה את התאריך והשעה הנוכחיים לתווית אשר תופיע בסופו של דבר בדפדפן:

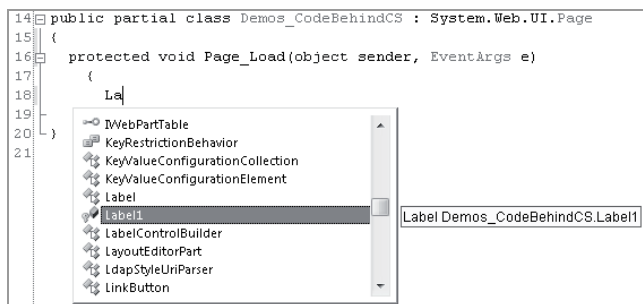
VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Me.Load  
  
    Label1.Text = "Hello World; the time is now " & DateTime.Now.ToString()  
  
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)  
{  
  
    Label1.Text = "Hello World; the time is now " + DateTime.Now.ToString();  
  
}
```

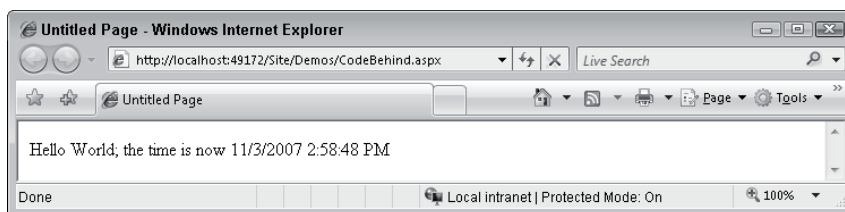
שים לב שלאחר הקלדת האות L עבור Label1 קיבלת רשימה עם אפשרויות בחירה. רשימה זו היא חלק מ- IntelliSense של VWD, כלי מצוין שעוזר לכתובת קוד בצורה מהירה. במקום להקליד את כל המלה Label1, אתה צריך להקליד את האות L בלבד, או את האותיות La, ואז לבחור את הפריט המתאים מהרשימה. ראה תרשים 14-2.



תרשים 2-14

להשלמת המילה שנבחרה, הקש Enter או Tab או אפילו נקודה (.). במקרה האחרון תקבל מייד את רשימת המאפיינים של הפקד, המאפשרת לבחור במילה Text על ידי הקלדת כמה מהאותיות הראשונות, ולהשלים אותה על ידי הקשת Enter או Tab. כלי יעיל מאוד ואפקטיבי, מפני שבעזרתו אתה יכול לכתוב קוד במינימום הקשות על המקשים. רכיב IntelliSense גם עבור סוגי קבצים רבים אחרים, כולל ASPX, HTML, CSS, JavaScript ו-XML. במקרים רבים תצוץ רשימת האפשרויות באופן אוטומטי ברגע שתתחיל להקליר. אם לא כך, הקש Ctrl+Spacebar כדי לגרום להופעת הרשימה. אם הרשימה מכסה חלק מהקוד שבחלון הקוד, לחץ והחזק את המקש Ctrl כדי להפוך את החלון לשקוף.

7. לחץ לחיצה ימנית על הדף (CodeBehind.aspx) בסייר הפתרון, ובחר מהתפריט View in Browser. אם מופיעה תיבת דו-שיח ששואלת אם ברצונך לשמור את השינויים לחץ Yes והדף יופיע בדפדפן, בדומה לחלון הדפדפן שאתה רואה בתרשים 2-15.



תרשים 2-15

אם אינך רואה את ההודעה עם התאריך והשעה, או שאתה מקבל שגיאה על הדף בדפדפן, ודא ששמרת את השינויים עבור כל הדפים הפתוחים. כדי לשמור את כל הדפים "במכה", הקש Ctrl+Shift+S או לחץ על הלחצן Save All בסרגל הכלים (זה שיש בו כמה סמלי תקליטונים סגולים). בנוסף, ודא שהקלדת את הקוד עבור השפה הנכונה. כאשר יצרת דף חדש זה, בחרת שפת תכנות שחלה על הדף כולו. אינך יכול לערבב שפות תכנות בדף אחד, ולכן, אם התחלת עם דף של Visual C# עליך לוודא שהכנסת את קטע הקוד של C# מהתרגיל.

8. הגדרת דף עם קוד כלול דומה מאוד. הוסף קובץ חדש לתיקייה Demos. קרא לו **InLine.aspx**, ודא שביטלת את הסימון של האפשרות Place Code in Separate File.

9. כפי שעשית בצעדים 3-5, העבר את הדף לתוך Design View, גרור תווית אל התגית <div>, ואז לחץ לחיצה כפולה על הדף במקום כלשהו מחוץ ל-<div> (שעכשיו מכילה את התווית). במקום לפתוח קובץ קוד ברקע, VWD מעבירה כעת את הדף שלך אל Markup View ומוסיפה את הקוד Page_Load ישירות בדף.
10. בשורה הריקה בבלוק הקוד שהכניסה VWD, הקלד את השורה המוצללת שאתה רואה בצעד 6 של תרגיל זה. ודא שאתה משתמש בשפת התכנות הנכונה. עליך לסיים עם הקוד הבא בחלק העליון של קובץ aspx שלך:

VB.NET

```
<script runat="server">
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        Label1.Text = "Hello World; the time is now " & DateTime.Now.ToString()
    End Sub
</script>
```

C#

```
<script runat="server">
    protected void Page_Load(object sender, EventArgs e)
    {
        Label1.Text = "Hello World; the time is now " + DateTime.Now.ToString();
    }
</script>
```

11. לחץ לחיצה ימנית על הדף (Inline.aspx) בסייר הפתרון ובחר View in Browser. לחילופין, הקש Ctrl+F5 כדי לפתוח את הדף בדפדפן. אתה אמור לראות את אותו המסך שקיבלת בצעד 7.

כיצד זה פועל

בזמן הביצוע, הדפים עם קוד כלול מתנהגים בדיוק כמו הדפים שמשתמשים בקוד ברקע. בשני המקרים רואה רכיב התשתית ASP.NET runtime את הקוד Page_Load ואז הוא מבצע כל קוד שכלול בו. בתרגיל נסה זאת לעיל משמעות הדבר שקבעת שהמאפיין Text של הפקד Label1 יכול את התאריך והשעה הנוכחיים.

שים לב שבדוגמה זו הקוד ב-C# נראה דומה מאוד לקוד ב-VB.NET. הקוד שמגדיר את הטקסט של הפקד Label כמעט זהה בשתי השפות. הבדל אחד הוא ש-VB.NET משתמש בתו & כדי להדביק שני קטעי טקסט זה לזה, ואילו C# משתמש בסימן החיבור (+). ההבדל השני הוא שב-C# חייבות כל שורות הקוד להסתיים בנקודה ופסיק (;) כדי לציין את הסיום של יחידת קוד, ואילו VB משתמש בשבירת שורה (שורה חדשה).

זו גם הסיבה לקו התחתני () הנוסף שראית בקטע קוד זה החל מצעד 6 של התרגיל:

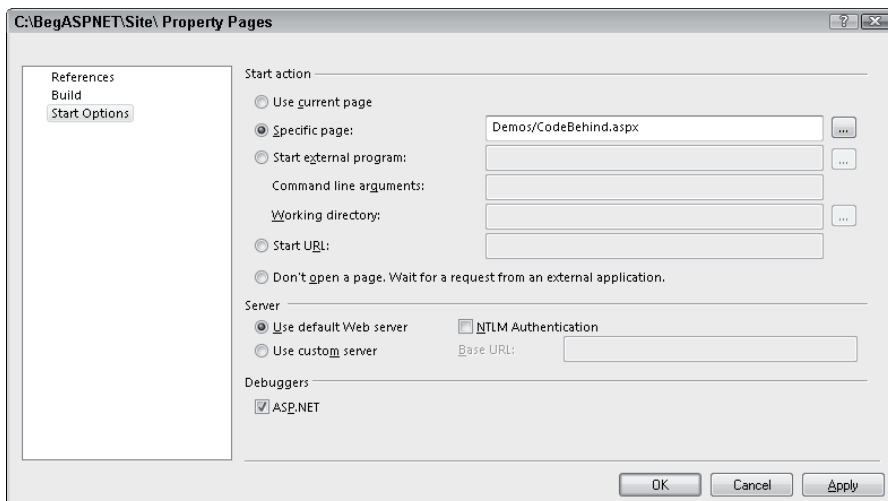
```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Me.Load
    Label1.Text = "Hello World; the time is now " & DateTime.Now.ToString()
End Sub
```

שים לב שבדף שלך לא תראה את הקו התחתני שבסוף השורה הראשונה של הדוגמה ב-VB. ב-VB אתה יכול להוסיף תו המשך שורה זה כדי לשבור שורות ארוכות ולכתוב אותן על פני מספר שורות פיזיות, מבלי לפגוע במשמעות המקורית של הקוד. התו התחתני נוסף כאן מפני שדפי הספר אינם רחבים מספיק להצגת המשפט השלם של הקוד בשורה אחת. בדוגמאות אחרות של Visual Basic שבהמשך הלימוד בספר תראה תווים תחתיים רבים. אם תחליט להקליד את התו התחתני באופן ידני כדי להפוך את הקוד שלך ליותר קריא, אל תשכח להקליד מרווח נוסף לפני התו התחתני, אחרת הקוד שלך לא יפעל כראוי.

ב-C# אין צורך בתו זה מפני שהשפה עצמה מרשה לך לשבור שורות ארוכות על ידי הקשת Enter בלבד. אפשר לעשות זאת מכיוון ש-C# משתמשת בנקודה ופסיק (;) לציון סוף שורה ולא ב- line break (מעבר שורה) כפי שעושה VB.

פתיחת הדף בדפדפן שלך התבצעה על ידי לחיצה ימנית על הדף ובחירה ב- View in Browser, או על ידי הקשת Ctrl+F5. כאשר תשתמש באפשרות View in Browser, תמיד ייפתח הדף שלחצת עליו לחיצה ימנית. כאשר אתה משתמש בקיצור הדרך Ctrl+F5, אתה פותח את הדף שהוא כרגע המסמך הפעיל בחלון המסמך, או את הדף שהוא כרגע הדף הנבחר בסייר הפתרון, או את הקובץ שנקבע כדף הפתיחה (start page) עבור אתר האינטרנט. בנוסף, כל הקבצים הפתוחים נשמרים באופן אוטומטי ונערכת בדיקת שגיאות באתר, לפני שהדף המבוקש נפתח בדפדפן.

אתה יכול לקבוע דף כלשהו כדף הפתיחה, על ידי לחיצה ימנית עליו בסייר הפתרון, ולחיצה על Set As Start Page. אם בשלב מאוחר יותר תרצה לשנות קביעה זו, לחץ לחיצה ימנית על האתר בסייר הפתרון ובחר Property Pages. בקטגוריה Start Options תוכל לציין שברצונך לפתוח את הדף הפעיל הנוכחי, או שתקבע דף מסוים שתבחר, כפי שמראה תרשים 2-16.



תרשים 2-16

בתרגיל הקודם למדת איך להוסיף דף שמכיל פקד Label פשוט. בנוסף, למדת לכתוב קוד שמעדכן את הפקד עם התאריך והשעה הנוכחיים. אתה יכול להתעלם מקוד זה בינתיים; הוא שירת רק להדגמת ההבדלים בין קוד ברקע לבין קוד כלול. בפרק 5 תלמד יותר על תכנות ב-VB וב-C#.

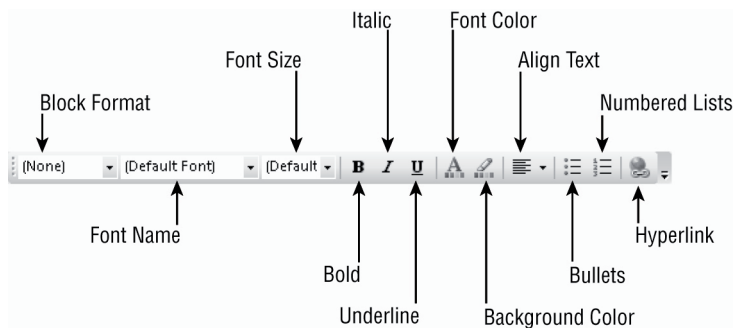
כדי ליצור דפים שישכנעו את המשתמשים בהם להמשיך ולצפות, אתה זקוק, ללא ספק, לתוכן רב הרבה יותר מאשר סתם פקד Label פשוט שמראה את התאריך והשעה הנוכחיים. הקטע הבא מראה לך איך להוסיף תוכן ו-HTML לדפים שלך ואיך לסגנן ולעצב אותם.

הוספת סימני עריכה לדף

יש כמה דרכים להוסיף HTML וסימני עריכה אחרים לדפים שלך. ראשית, אתה יכול להקליד זאת בחלון Markup View. עם זאת, זו אינה תמיד האפשרות הטובה ביותר, מפני שהיא כופה עליך להקליד קוד רב באופן ידני. כדי להקל על הוספת HTML חדש ולהחיל עליו עיצוב, מצייע החלון Design View כמה כלים מועילים, אשר כוללים, בין השאר, את סרגל הכלים Formatting ואת פריטי Format ו-Table. כדי שכלים אלה יהיו פעילים, המסמך צריך להיות ב-Design View. אם אתה עובד במצב Split View עליך לוודא שהחלק Design View נמצא במוקד, אחרת תמצא שרוב הכלים אינם זמינים.

הוספה ועיצוב טקסט

אתה יכול להקליד טקסט הן ב-Design View והן ב-Markup View. לשם כך, הצב את הסמן במקום הרצוי והחל להקליד. כאשר תעבור ל-Design View, יהפוך סרגל הכלים Formatting לזמין, עם האפשרויות שמוצגות בתרשים 2-17.



תרשים 2-17

הרשימה הנפתחת הקרויה Block Format מאפשרת להכניס תגיות HTML כמו <p> עבור פסקה, <h1> עד <h6> עבור כותרות, , ו- עבור רשימות. באפשרותך לבחור פריט רצוי מתוך הרשימה הנפתחת והוא יוכנס לדף, או שתבחר תחילה טקסט כלשהו ואז תבחר מהרשימה את האלמנט המתאים אשר יעטוף את הטקסט בתוך התגיות.

הרשימה הנפתחת הקרויה Font Name מאפשרת לשנות את סוג הגופן, ואילו הרשימה הנפתחת Font Size מאפשרת לשנות את גודל הגופן.

שאר הלחצנים שבסרגל הכלים מתפקדים בדיוק כפי שהם מתפקדים בסביבות עריכה אחרות. לדוגמה, הלחצן B מעצב את הטקסט עם גופן מודגש. בדומה לכך, הלחצנים I ו-U גורמים לטקסט להיות מוטה או עם קו הדגשה מתחתיו, בהתאמה.

בתרגיל נסה זאת הבא תלמד כיצד להשתמש בכלים אלה ליצירת דף הבית של האתר Planet Wrox.

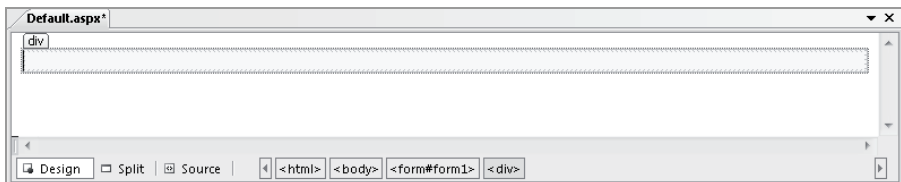
נסה זאת הוספת טקסט מעוצב

בתרגיל זה נשנה את הדף Default.aspx שהתווסף לאתר באופן אוטומטי כאשר יצרת אותו. בפרק 6 תשנה את הדף פעם נוספת, כאשר תתחיל להשתמש בדפים ראשיים.

1. פתח את הדף Default.aspx, ועבור ל- Design View על ידי שימוש בלחצן Design שבחלק התחתון של חלון המסמך.

2. ודא שסרגל הכלים Style Application גלוי. אם לא כך, לחץ לחיצה ימנית על סרגל כלים קיים, ואז בחר Style Application מהרשימה. כאשר סרגל הכלים גלוי, ודא שהרשימה הנפתחת Style Application הוגדרה כ-Manual ולא כ-Auto.

3. לחץ בתוך המלבן המקווקו עד שתראה את הסמל הקטן מעל המלבן מראה שהאלמנט <div> פעיל כרגע. באותו הזמן, נָט התגיות (tag navigator) שבתחתית חלון הקוד צריך להדגיש את הבלוק האחרון שמכיל בתוכו את הטקסט <div>, כפי שרואים בתרשים 2-18.



תרשים 2-18

4. הקלד "Hi there visitor and welcome to Planet Wrox" והדגש את הטקסט על ידי העכבר. מהרשימה הנפתחת Block Format (נראית בתרשים 2-17) בחר <h1> Heading 1. שים לב שסמל קטן עם הטקסט h1 מופיע מייד מעל הטקסט, כדי לציין ש-VWD יצרה עבורך כותרת. תרשים 2-19 מראה את ה- Design View עם האלמנט <h1>.



תרשים 2-19

5. הצב את הסמן בקצה הכותרת לאחר המילה Wrox והקש Enter. פסקה חדשה (מסומנת על ידי סמל קטן עם האות p בתוכו) מוכנסת עבורך, כך שאתה יכול להתחיל להקליד באופן ישיר.

6. הקלד את הטקסט שאתה רואה בתרשים 2-20 (או טקסט אחר רצוי) כדי לקדם בברכה את המבקר באתר Planet Wrox. בחר בטקסט "paying a visit", לחץ על הלחצן Foreground Color בסרגל הכלים Formatting, ומתיבת הדו-שיח שמופיעה בחר בצבע אחר. עכשיו בחר טקסט אחר, כגון "reviews and concert pictures", ולחץ על הלחצן Bold. כשתסיים צריך ה- Design View להראות משהו דומה לתרשים 2-20.



תרשים 2-20

הקוד עבור דף הבית צריך להיראות עכשיו דומה לקוד הבא (הקוד עוצב מעט כדי להתאים לרוחב דף הספר):

<div>

```
<h1>Hi there visitor and welcome to Planet Wrox</h1>
```

```
<p>
```

```
We're glad you're <span class="style1">paying a visit</span> to  
<a href="http://www.PlanetWrox.com">www.PlanetWrox.com</a>,  
the coolest music community site on the Internet.
```

```
</p>
```

```
<p>
```

```
Feel free to have a look around; as there are lots of interesting <b>reviews  
and concert pictures</b> to be found here.
```

```
</p>
```

</div>

7. פתח את הדף בדפדפן שלך על ידי הקשת Ctrl+F5, או על ידי לחיצה ימנית על הדף בסייר הפתרון, ואז בחר View in Browser.

כיצד זה פועל

כשאתה משתמש בלחצנים השונים של סרגל הכלים Formatting, כגון Foreground Color, תראה ש-VWD מוסיפה עבורך את תגיות HTML ועיצוב מתאים. לדוגמה, כאשר אתה לוחץ על הלחצן B תראה ש-VWD מוסיפה זוג תגיות סביב הטקסט הנבחר. כאשר אתה לוחץ על הלחצן I, תראה ש-VWD מוסיפה זוג תגיות <i> סביב הטקסט, כדי לגרום לו להיות מוטה. VWD מכניסה

גם את המאפיין class (מוצג בקטע הקוד הקודם) שמצביע אל מחלקה בשם style1. הקוד עבור סגנון זה התווסף לראש הקובץ שלך, ונראה דומה לזה:

```
<style type="text/css">
.style1
{
    color: #FF0000;
}
</style>
```

הקוד שלך עשוי להיראות שונה מעט אם בחרת בצבע אחר. הקוד שאתה רואה כאן יוסבר בפרק הבא. בשלב זה מספיק לזכור שהקוד קובע שצבע קטע הטקסט שהוא חל עליו יהיה אדום.

ה- Style Application שהגדרת בתחילת תרגיל זה קובע את המקום שאליו VWD תוסיף את הקוד שלך. במצב Manual אתה שולט במקום שאליו יתווסף הקוד על ידי בחירה שאתה בוחר מהרשימה הנפתחת Target Rule. במצב Auto הבחירה היא בידי VWD. מרבית הדוגמאות בספר מניחות שקבעת את המצב Manual.

שים לב ש-VWD החליפה את התו גרש (') ב-"we're" שבהודעת הפתיחה ב: "'". שימוש בקוד מסוג זה מאפשר להוסיף לךף תווים שלדפדפן עלול להיות קושי להציגם, או שהם תווים בעלי משמעות מיוחדת ב-HTML עצמו, כמו התו & שנכתב כ-"&". כשאתה מקליד טקסט ב- Design View, תראה ש-VWD יהפוך באופן אוטומטי את הקידוד של תווים רלוונטיים. אם תקליד ישירות ב- Markup View תצטרך לעשות זאת בעצמך.

אל תדאג אם הקוד שלך נראה שונה ממה שמוצג כאן. הגדרות רבות ב-VWD משפיעות על הקוד שנוצר עבורך.

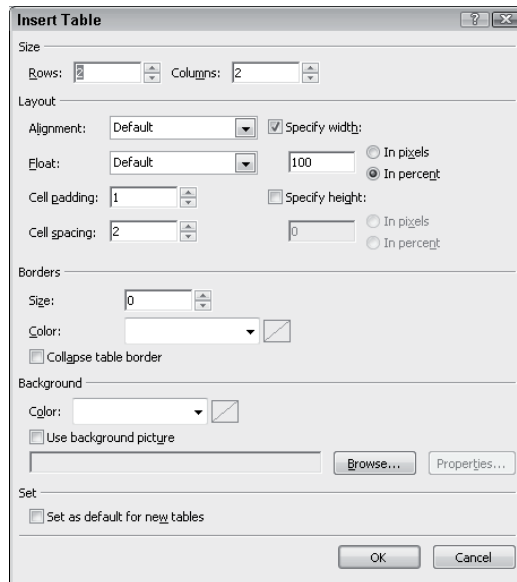
עד כה עסקו התרגילים בהוספת טקסט לךף ובסגנונו. עם זאת, VWD מאפשרת לך גם להוסיף תגיות HTML אחרים כמו טבלאות ותבליטים. הסעיף הבא מראה כיצד לבצע זאת.

הוספת טבלאות וסימני עריכה אחרים

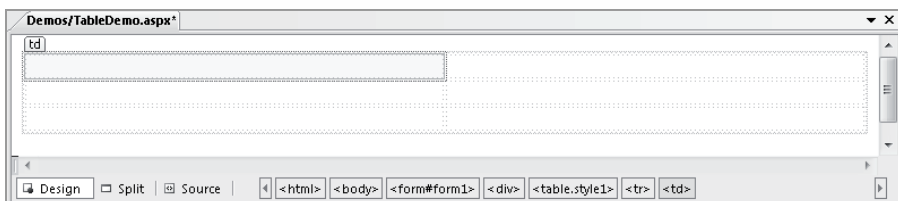
טבלאות HTML הן נהדרות אם אתה צריך להציג נתונים כמבנה או נתונים שחוזרים על עצמם, כמו רשימת מוצרים בסל קניות, תמונות באלבום תמונות, או פקדי קלט בטופס. יש ויכוח גדול באינטרנט לגבי השאלה אם להשתמש בטבלאות גם לפריסת הדף. לדוגמה, אם הדף שלך מכיל כותרת עם לוגו, אזור תוכן ראשי וכותרת תחתונה, אתה יכול להשתמש בטבלה בת שלוש שורות כדי להשיג זאת. השימוש בטבלאות למטרה זו נחשב בדרך כלל לנהול עבודה רע, מפני שטבלאות מוסיפות לךף המון סימני עריכה חיצוניים, ולעתים קרובות הן קשות לתחזוקה. בנוסף, לעתים ניתן להגיע לאותה תוצאה בעזרת CSS (תלמד זאת בפרק הבא). למרות החסרונות הכרוכים בשימוש בטבלאות, הן עדיין נכס רב-ערך בארגז הכלים של HTML כשהדברים נוגעים להצגת מידע טבלאי או מובנה בצורה אחרת.

בתרגיל זה תלמד איך להוסיף טבלאות לדף שלך תוך שימוש בתפריט Table, וכיצד להוסיף לטבלה שורות ועמודות. תלמד גם איך להוסיף אלמנטים מוכנים אחרים כמו רשימות עם תבליטים.

1. תחת התיקייה Demos צור טופס אינטרנט חדש בשם **TableDemo.aspx**. ודא שהוא משתמש בקוד ברקע על ידי בדיקת האפשרות Place code in separate file.
2. העבר את הדף ל- Design View, לחץ בתוך המלבן המקווקו שמייצג בדף את התגית `<div>`, ובחר Table ← Insert Table. תופיע תיבת הדו-שיח Insert Table, כפי שמוצג בתרשים 2-21.
3. שנה את כמות השורות ל-3 והשאר את כל שאר ההגדרות לבירורות המחדל שלהן ולחץ OK. הטבלה תוכנס לדף.
4. אם אתה רואה רק תא טבלה אחד, ולא את הטבלה השלמה עם שלוש שורות ושתי עמודות, עליך לאפשר את תמיכת Visual Aid בטבלאות. כדי להשיג זאת בחר מהתפריט הראשי View ← Visual Aids ← Visible Borders. כעת אתה אמור לראות את גבולות הטבלה. ה- Design View צריך להיראות כעת כמו תרשים 2-22.

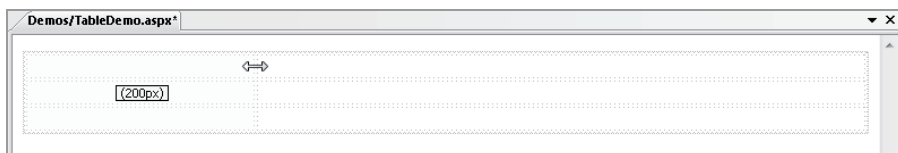


תרשים 2-21



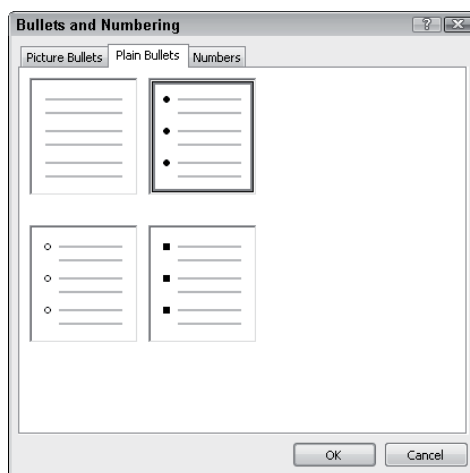
תרשים 2-22

5. גרור את הגבול הימני של התא הראשון בטבלה שמאלה. תראה מחוון חזותי שמציג את רוחב התא. המשך לגרור את התא שמאלה עד שיגיע לרוחב של 200 פיקסל, כמו בתרשים 2-23.



תרשים 2-23

6. כדי להוסיף שורות ועמודות לטבלה אתה יכול ללחוץ לחיצה ימנית על אחד התאים הקיימים. מתפריט הקיצור שמופיע בחר Insert, להוספת שורות ועמודות במקומים שונים. בצורה דומה אתה יכול להשתמש באפשרויות Delete, Select, ו-Modify כדי למחוק שורות או עמודות, למזג תאים, ולבצע בחירות. בתרגיל זה אין צורך להוסיף שורות או עמודות.
7. הצב את הסמן בתא הראשון (שמאלי עליון) והקלד **Bulleted List**.
8. הצב את הסמן מימין לתא הראשון ובחר Bullets and Numbering מתפריט Format.
9. עבור לכרטיסייה Plain Bullets, לחץ על התמונה עם הכדורים המלאים (ראה תרשים 2-24), ולחץ OK.

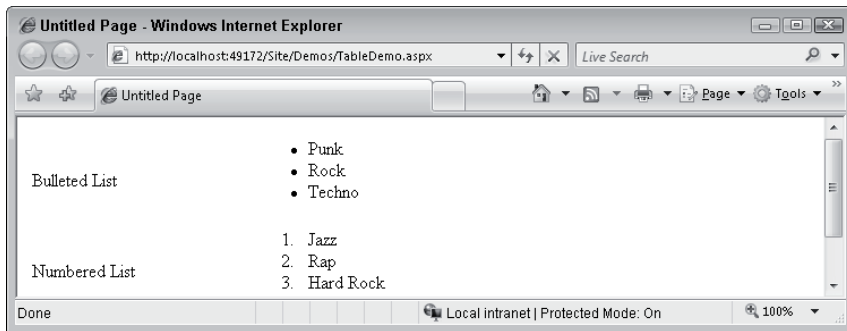


תרשים 2-24

10. הקלד טקסט כלשהו, למשל את הסגנון המוזיקלי האהוב עליך (רוק, ראפ וכו'), והקש Enter. כעת VWD תוסיף באופן אוטומטי תבליט חדש, כדי שתוכל להמשיך ולהוסיף פריטים לרשימה. הוסף עוד שני סגנונות לרשימה.
11. חזור על צעדים 7 עד 10, אבל הפעם צור רשימה ממוספרת. ראשית, בתא הראשון של השורה השנייה הקלד **Numbered List**, ואז הצב את הסמן בתא השני של אותה שורה ובחר Format ← Bullets and Numbering. עבור לכרטיסייה Numbers (נראית בתרשים

2-24 מאחורי הכרטיסייה (Plain Bullets) ולחץ על התמונה אשר מראה רשימה ממוספרת, ולחץ OK. הקלד כמה פריטים לרשימה והקש Enter אחרי כל פריט.

12. פתח את הדף בדפדפן על ידי הקשת Ctrl+F5. אתה אמור לראות מסך דומה לתרשים 2-25.



תרשים 2-25

כיצד זה פועל

כאשר אתה מכניס באופן חזותי אלמנטים חדשים אל הדף, כמו טבלאות ורשימות, באמצעות התפריטים הזמינים, VWD מוסיפה את סימני העריכה הדרושים ב-Markup View. כאשר אתה מכניס טבלה אל הדף, VWD מוסיפה את התגית `<table>` וכמה תגיות `<tr>` ו-`<td>` כדי להגדיר שורות ותאים, בהתאמה. VWD גם מוסיפה לטבלה את המאפיין `style` כדי לשלוט ברוחב הטבלה. כאשר גררת את רוחב העמודה עד ל-200 פיקסל יצרה VWD סגנון עבור האלמנטים `<td>`. באופן דומה, כאשר אתה מוסיף רשימה, מתווספת תגית `<ol>` עבור רשימות ממוספרות, ותגית `<ul>` עבור רשימות תבליטים. בתוך תגיות אלה משמשים האלמנטים `<li>` להגדרת כל פריט ברשימה.

פרט לתגיות HTML שראית עד כה יש תגית חשובה אחרת שעליך להכיר: זוהי התגית `<a>` אשר משמשת ליצירת קישור בין דפים.

קישור בין דפים

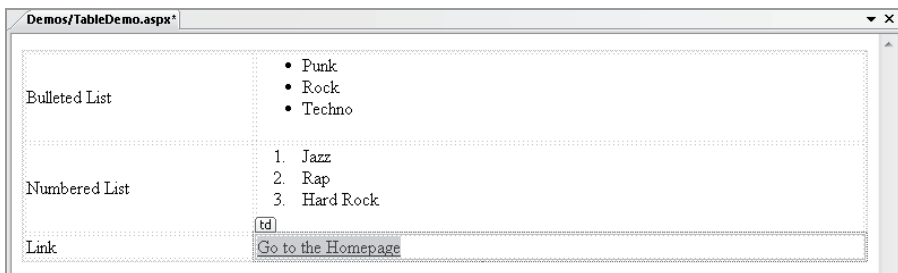
חלק חשוב בכל אתר אינטרנט הם הקישורים שמחברים את הדפים באתר כדי ליצור יחידה אחת. קישורים מאפשרים למבקרים באתר לעבור מדף אחד לאחר, באותו אתר, או לאתר שונה לחלוטין באינטרנט. יש כמה דרכים ליצור קישורים בין דפים, וביניהם:

- ⊙ על ידי האלמנט `<a>` של HTML, שמוסבר בפרק זה.
- ⊙ שימוש בפקד `<asp:HyperLink>`, כפי שמוסבר בפרק 7.
- ⊙ על ידי תכנות קוד, כפי שמוסבר מאוחר יותר בספר.

התרגיל הבא מראה כמה קל לקשר בין דף אחד לאחר.

בתרגיל זה תשנה את הדף TableDemo.aspx שיצרת קודם, על ידי הוספת טקסט שמקשר אותו אל דף אחר. ברגע שתריץ את הדף בדפדפן ותלחץ על קישור זה, הדף החדש יחליף את הדף הישן.

1. פתח את הדף TableDemo.aspx מהתיקיה Demos.
2. אם נחוץ, עבור ל- Design View.
3. בתא הראשון של השורה השלישית הקלד **Link**.
4. בתא השני של אותה שורה הקלד את הטקסט **Go to the Homepage** והדגש אותו על ידי העכבר.
5. בסרגל הכלים Formatting לחץ על הלחצן Convert to HyperLink. זהו הלחצן האחרון בסרגל הכלים, שיש מעליו גלובוס ירוק. אם אינך רואה את הלחצן משום שהוא מוסתר על ידי סרגלי כלים אחרים, תוכל לגרור את סרגל הכלים Formatting למקום אחר, או שתבחר Format ← Convert to Hyperlink כדי לגרום להופעת אותה תיבת דו-שיח.
6. בתיבת הדו-שיח שמופיעה לחץ על הלחצן Browse, ובחר את הדף Default.aspx ולחץ OK. לחץ OK פעם נוספת כדי לשחרר את תיבת הדו-שיח Hyperlink. ה- Design View של הדף שלך צריך להיראות דומה לזה שמוצג בתרשים 2-26.



תרשים 2-26

7. עבור ל- Markup View ושים לב לקוד HTML שהתווסף עבור הקישור:

```
<a href="../../Default.aspx">Go to the Homepage</a>
```

- שים לב שהמאפיין href מצביע אל הדף שאתה רוצה להתחבר אליו.
8. אם ברצונך לשנות מחלון הקוד את הדף שאתה עומד להיות מקושר אליו, לחץ במקום כלשהו בין הגרשיים הפותחים והסוגרים של המאפיין href והקש Ctrl+Spacebar. מופיעה תיבת דו-שיח שמאפשרת לבחור דף אחר. לחילופין, אתה יכול ללחוץ על האפשרות Pick URL ולדפדף אל הדף החדש שנמצא אי שם באתר שלך.
9. לחץ לחיצה ימנית על הדף TableDemo.aspx בסייר הפתרון, ובחר View in Browser. כאשר הדף יסיים להיטען, לחץ על הקישור Go to the Homepage. הבקשה נשלחת אל שרת האינטרנט, וכתגובה תקבל את דף הבית של האתר.

קישורים (links) בין דפים הם, ככל הנראה, אחד האלמנטים החשובים ביותר בדף אינטרנט, מפני שהם מאפשרים ליצור חיבור בין דף באתר שלך לבין דף אחר, בין אם הדף האחר נמצא באתר שלך ובין אם הוא נמצא בשרת שונה לחלוטין אי-שם באינטרנט. עבור קישורים פשוטים שצריכים להופיע במקום כלשהו בדף שלך, השימוש בתגית `<a>` של HTML יחד עם הגדרת המאפיין `href` הם הדרך הקלה ביותר. כאשר המשתמש לוחץ על קישור כזה, הדפדפן מבקש את הדף החדש מהשרת ומציג אותו. שים לב שאינך מוגבל לחיבור לדפים באתר שלך בלבד. אם ברצונך להגיע לדפים חיצוניים, עליך להחליף את ערך המאפיין `href` בכתובת המלאה של הדף, כפי שמוצג בדוגמה הבאה:

```
<a href="http://www.wrox.com ">Go to the Wrox Homepage</a>
```

חשוב לכלול את התחילית `http://`, אחרת הדפדפן יחפש באתר שלך קובץ או תיקייה בשם `www.wrox.com`.

עצות מעשיות לעבודה עם טפסי אינטרנט

להלן כמה עצות לעבודה עם טפסי אינטרנט:

- ⊙ תמיד נסה להעדיף טפסי אינטרנט עם קוד ברקע על פני אלה שיש בהם קוד כלול. בהתחלה אולי לא תבחין בהבדל גדול בעבודה עם טפסים אלה, אך ככל שהדפים והאתר שלך יגדלו, תמצא שקל יותר לעבוד עם דפים שבהם הקוד נפרד מסימני העריכה. בנוסף, כאשר העבודה על האתר היא עבודת צוות, קוד ברקע מאפשר ליותר מאדם אחד לעבוד על אותו קובץ: מפתח אחד יכול לעבוד על העיצוב בקובץ `aspx` בעוד שמפתח אחר יכול להוסיף קוד לקובץ `aspx.vb` או לקובץ `aspx.cs`.
- ⊙ הקדש זמן מה להכרת הפריטים השונים של התפריטים `Table-1 Format`. מרביתם יוצרים תגיות HTML שמוכנסות לדף שלך. הבט בתגיות HTML ובמאפיינים שנוצרו עבורך ונסה לשנותם ישירות בקוד, באמצעות התפריטים וארגזי הכלים. בדרך זו תקבל תחושה טובה לגבי התגיות השונות הזמינות וכיצד הן מתנהגות.
- ⊙ ערוך ניסויים עם קישורים לחיבור דפים באתר שלך. שים לב כיצד VWD יוצרת קישורים שונים בהתאם למיקום של הדף שאתה מתקשר אליו. פרק 7 עוסק בפירוט בנושא הקישורים ובדרכים השונות לפנות לדפים באתר שלך.

סיכום

בפרק זה הוצגו כמה נושאים חשובים שיעזרו לך לבנות יישומי אינטרנט מוכנים וקלים לתחזוקה בסביבת ASP.NET. הבנת ההבדלים בין סוגי התבניות והפרויקטים השונים מאפשרת לך להתחיל בפיתוח פרויקט אינטרנט רק עם הקבצים שאתה זקוק להם.

אותו דבר חל גם על סוגי הקבצים השונים שאתה יכול להוסיף לאתר שלך. מכיוון שכל סוג קובץ משרת מטרה מוגדרת, חשוב להבין מהי מטרה זו, וכיצד אתה יכול להשתמש בקובץ.

החלטה חשובה אחרת שאתה צריך להחליט בזמן בניית אתרי אינטרנט, היא האם אתה בונה דפי אינטרנט עם קוד ברקע או עם קוד כלול. למרבה המזל, החלטה זו אינה משפיעה על האתר כולו, אך תצטרך לחזור ולשקול אותה לגבי כל קובץ שתוסיף. כלומר, אם התחלת עם קוד כלול ומצאת שקשה לעבוד איתו כאשר הדפים והאתר שלך גדלים, תוכל להתחיל להוסיף דפים עם קוד ברקע החל מאותה נקודה.

פעילות שכיחה אחת שתבצע בזמן בניית דפי אינטרנט בסביבת ASP.NET היא הוספת סימני עריכה לדף. כפי שראית בפרק זה ובפרק הקודם, סימני עריכה הינם מסוגים שונים, כולל HTML פשוט ופקדי שרת של ASP.NET. הידיעה איך להוסיף סימני עריכה אלה לדף תוך שימוש באפשרויות הרבות שמציעים התפריטים וארגזי הכלים שמספקת VWD, היא קריטית לבניית דפי אינטרנט נעימים למראה.

כעת, כשיש לך הבנה מוצקה של יצירה ושינוי של טפסי אינטרנט, הגיע הזמן לראות כיצד אפשר, בעזרת כמה פקדים, להפוך את הדפים בעלי המראה המשעמם בשחור-לבן לדפי אינטרנט אטרקטיביים. בפרק הבא תלמד איך לעבוד עם הכלים הרבים של CSS שנמצאים ב-VWD, כדי ליצור את האפקט הרצוי.

תרגילים

1. מנה שלושה קבצים חשובים בקטגוריה Web Files שבאפשרותך להוסיף לאתר שלך. תאר את המטרה של כל אחד.
2. מה עליך לעשות כדי להפוך קטע טקסט בדף למודגש ונטוי? כיצד ייראה קוד HTML שיתקבל?
3. מנה שלוש דרכים שונות ב-VWD להוספת קבצים קיימים לאתר אינטרנט בסביבת ASP.NET.
4. מהן התצוגות השונות שמציעה לך VWD עבור דפי ASPX?

פרק 3

עיצוב דפי האינטרנט שלך

הדפים שיצרת בשני הפרקים הקודמים נראו משעממים ולא אטרקטיביים, משום שאינם מעוצבים. עיצוב הדפים נעשה באמצעות השפה **CSS** (Cascading Style Sheets). CSS היא השפה העיקרית הנהוגה לעריכה ולעיצוב של מידע באינטרנט, ובכלל זה דפי אינטרנט בסביבת ASP.NET. עם CSS באפשרותך לשנות במהירות את התצוגה של דפי האינטרנט שלך, ולהקנות להם את המראה הרצוי.

למרות שגרסאות קודמות של Visual Web Developer חסרו כלים טובים לעבודה עם CSS, הרי לסביבה של Visual Web Developer 2008 יש תמיכה מצוינת ב-CSS והיא מסוגלת לעצב דפים עוד בזמן פיתוח בצורה הקרובה ביותר לזו שבה הם יוצגו בדפדפן. הכלים החדשים מאפשרים לך ליצור בקלות עיצוב CSS מבלי שתצטרך לדעת, לזכור או לעסוק בכל פרט קטן ב-CSS.

בפרק זה תלמד בהרחבה את הנושאים הבאים:

- ⊙ מהי CSS ומדוע אתה זקוק לה.
- ⊙ איך נראית שפת CSS ואיך לכתוב בה.
- ⊙ הדרכים השונות להוספת קוד CSS לדפי ASP.NET ולקבצים חיצוניים.
- ⊙ הכלים הרבים שמציעה VWD לכתיבה מהירה של CSS.

כדי להבין את הרלוונטיות של CSS ואת הצורך בה בפרויקטים הפועלים בסביבת ASP.NET, עליך להכיר תחילה את המגרעות של HTML. הסעיף הבא דן בבעיות שכרוכות בשימוש ב-HTML פשוט, ואיך CSS מסוגלת להתגבר על בעיות אלו.

הערה: לימוד נוסף של CSS תמצא בספר "**HTML & CSS למפתחי אתרים באינטרנט**" בהוצאת הוצאת עמ"י.

מדוע אתה צריך CSS?

בראשית ימיו של האינטרנט, דפי web היו מורכבים בעיקר מטקסט ותמונות. הטקסט עוצב תוך שימוש ב-HTML פשוט ותגיות, כמו לדוגמה **** כדי להגדיר טקסט מודגש (עבה), ו-**** כדי לקבוע את סוג הגופן, גודלו וצבעו. המפתחים של יישומי אינטרנט הבינו עד מהרה שדרושים להם כלים חזקים יותר כדי לעצב את הדפים שלהם ואמנם, CSS נוצרה כדי לענות על מגרעות HTML בתחום עיצוב סגנון הדפים.

בעיות בעריכה עם HTML

אחת הבעיות בשימוש ב-HTML לעריכה היא שקבוצת האפשרויות שמציעה HTML לעיצוב סגנון הדפים מוגבלת למדי. אתה יכול להשתמש בתגיות כמו *<i>*, **** ו-`<font>` כדי לשנות את ההופעה של טקסט, ובמאפיינים כמו bgcolor כדי לשנות את צבע הרקע של אלמנטים של HTML. כמו כן, עומדים לרשותך כמה מאפיינים שמאפשרים לשנות את הדרך שבה מופיעים קישורים בדפים שלך.

ברור שקבוצת מאפיינים זו אינה עשירה מספיק לעיצוב דפי האינטרנט האטרקטיביים שהמשתמשים שלך מצפים להם ודורשים אותם.

בעיה אחרת של HTML המשפיעה בצורה רבה על בניית דפי האינטרנט שלך, היא הדרך שבה משולב המידע על העיצוב בדף. כתוצאה מעיצוב זה, כופה עליך HTML להטמיע את הנחיות העיצוב כחלק ממסמך HTML שלך, והדבר מקשה על שימוש חוזר או שינוי העיצוב מאוחר יותר. התבונן בדוגמה הבאה:

```
<p><font face="Arial" color="red" size="+1">  
This is red text, in an Arial type face and slightly larger than the default text  
</font></p>
```

הבעיה עם קטע קוד זה היא שאין הפרדה בין תגיות HTML לבין המידע אשר מורה על דרך הצגת התגיות. הנתונים ומידע העיצוב צריכים להיות נפרדים, כדי שיהיה קל יותר לשנות אחד מהם מבלי להשפיע על האחר.

דמיין לעצמך שהאתר שלך מכיל 40 דפים. ובכל דף נעשה שימוש בתגיות `<p>` ו-`<font>` כפסקה הראשונה. קוד שכזה קשה לתחזוקה. למשל, מה יקרה כשאתה מחליט לשנות את צבע הגופן מאדום לכחול כזה? ו/או קיבלת דרישה לשנות את הגופן ל-Verdana במקום Arial? במקרים כאלה היית צריך לעבור על כל דף ודף באתר שלך ולהכניס בו את השינויים הנדרשים.

פרט לעניין התחזוקה מתעוררת בעיה נוספת עם עיצוב HTML – אינך יכול לשנות בקלות את העיצוב בזמן שהדף מוצג בדפדפן. אם תשתמש ב-HTML שבקטע הקוד הקודם לא תהיה לך כל דרך לאפשר למבקר באתר לשנות דברים, כמו גודל הגופן וצבעו. אם ברצונך להציע למבקרים באתר גרסה אלטרנטיבית של הדף, עם גופן גדול יותר או בעל צבע אחר, אתה צריך ליצור עותק של הדף המקורי ולהכניס בו בעצמך את השינויים הנחוצים.

הבעיה האחרונה עם עיצוב HTML היא שסימני העריכה הנוספים בדף שלך מגדילים במידה ניכרת את נפח הדף. דבר זה גורם להאטה בהורדת הדף והצגתו (תעבורת הרשת), מכיוון שהמידע צריך להיטען עם כל דף באתר שלך. דבר זה גם מקשה על תחזוקת הדפים בזמן פיתוח מכיוון שעליך לגלול דרך קבצי HTML גדולים כדי למצוא את התוכן הדרוש לך.

לסיכום, עיצוב עם HTML סובל מהבעיות הבאות:

⊙ קבוצת המאפיינים של HTML מגבילה באופן חמור את אפשרויות העיצוב שנדרשות על ידי הדפים שלך.

- ⊙ נתונים ומידע תצוגה מעורבבים זה בזה באותו קובץ.
- ⊙ אין אפשרות קלה להחלפת העיצוב בדפדפן בזמן ריצה באופן דינמי.
- ⊙ התגיות ומאפייני העיצוב הנדרשים מגדילים את הדפים שלך, והדבר גורם להאטה בטעינתם ובהצגתם.

למרבה המזל מאפשרת לך CSS להתגבר על כל הבעיות האלו.

כיצד פותרת CSS את בעיות העיצוב

CSS תוכננה לעצב את דפי האינטרנט שלך כמעט בכל דרך אפשרית. היא מציעה מגוון עשיר של אפשרויות לעיצוב דפי האינטרנט שלך, כולל גופנים (גודל, צבע, משפחה וכו'), צבעים וצבעי רקע, גבולות סביב אלמנטים של HTML, הצבת אלמנטים בדף ועוד. CSS נתמכת על ידי כל הדפדפנים העיקריים כיום, ולכן היא השפה המקובלת להגדרת תצוגה של דפי אינטרנט, ופופולרית מאוד בין מפתחי אינטרנט.

CSS מתגברת על בעיית ערכוב הנתונים ומידע התצוגה על ידי הגדרת כל מידע העיצוב בקבצים חיצוניים. דפי ASPX או HTML שלך יכולים אז לפנות לקבצים אלה, והדפדפן הוא זה שמיישם את הסגנונות הנכונים. עם החלת הפרדה זו, מסמך HTML מכיל כעת את מה שאתה רוצה להציג, בעוד קובץ CSS מגדיר איך אתה רוצה להציג זאת. הפרדה זו מאפשרת לך לשנות, או להחליף, את אחד משני המסמכים בעוד שהאחר יישאר ללא שינוי. בסעיף הבא תראה כיצד זה נעשה. בנוסף, ניתן להציב קוד CSS באופן ישיר בדף HTML או ASPX, וכך באפשרותך להוסיף קטעים קטנים של קוד CSS בדיוק במקום שאתה צריך אותם. כשאתה מציב את קוד CSS ישירות בדף HTML או ASPX, עליך לגלות זהירות, משום שכתוצאה מכך אתה מאבד את הגמישות של שליטה על העיצוב ממקום מרכזי.

מכיוון שניתן להציב את כל קוד CSS בקובץ נפרד, קל להציע למשתמש אפשרויות בחירה בין סגנונות שונים, לדוגמה סגנון עם גופן גדול יותר. אתה יכול ליצור עותק של גיליון הסגנון החיצוני, לערוך את השינויים הנדרשים, ואז להציע למשתמש את גיליון הסגנון האלטרנטיבי. בפרק 6, כאשר יידונו נושאי ASP.NET, תראה כיצד זה פועל.

תועלת אחרת מקובץ גיליון סגנון נפרד היא הקטנת התעבורה ברשת. גיליונות סגנון אינם משתנים בכל בקשה, ולכן הדפדפן יכול לשמור עותק מקומי של גיליון הסגנון בפעם הראשונה שהוא טוען אותו. מאותו רגע ואילך, הדפדפן משתמש בעותק הנשמר במרחב הזיכרון שלו במקום לבקש אותו מהשרת בכל בקשה. לפעמים הדבר עלול לפעול נגדך, והדפדפן לא יטען קבצי CSS הכוללים את השינויים העדכניים ביותר שלך. אם תמצא שהדפדפן אינו בוחר בקובץ CSS עם השינויים שעשית, השתמש בקיצור הדרך Ctrl+F5 (לא ב-VWD), כדי לקבל עותק רענן מהשרת.

כעת, כשהבנת מדוע CSS כה חשובה, הגיע הזמן ללמוד איך היא נראית וכיצד להשתמש בה.

מבחינת התחביר שלה, CSS היא שפה קלה ללימוד. השפה מורכבת מכמה מושגים בלבד ולכן קל יחסית להתחיל להשתמש בה. הארגון העולמי שמתחזק את תקן HTML, ה-W3C (World Wide Web Consortium) כולל תקן CSS בשלוש גרסאות שונות: 1.0, 2.1 ו-3.0. מבין שלוש גרסאות אלו, גרסה 2.1 היא המקובלת ביותר כיום. היא מכילה כל מה שמכילה גרסה 1.0, ואפשרויות רבות מעבר לכך. בגרסה זו משתמשת VWD, ואותה היא יוצרת כברירת מחדל. גרסה 3.0 נמצאת כעת בפיתוח, וצפוי שיעבור זמן מה לפני שהדפדפנים העיקריים יתמכו בה באופן גורף.

לפני שנתבונן בתחביר CSS, רצוי להתבונן תחילה בדוגמה. בתרגיל הבא תתבקש לכתוב דף ASPX פשוט שמכיל מעט CSS כדי לעצב את תוכן הדף. הדבר יעזור להבנת שפת CSS שעליה נדון בפירוט מלא בסעיף הבא.

נסה זאת **כתיבת קוד CSS הראשון שלך**

בתרגיל זה תכתוב קוד CSS קצר שמשנה את עיצוב הכותרת, ושתי פסקאות. בשלב זה תקודד את הדף באופן ידני; בחלקו השני של פרק זה תלמד כיצד להשתמש בכלי CSS הזמינים ב-VWD, אשר הופכים את כתיבת CSS לפשוטה וקלה.

1. בפרויקט Planet Wrox צור קובץ חדש בתיקייה Demos, וקרא לו Csdemo.aspx. עבור תרגיל זה אין חשיבות לכך שתבחר קוד כלול או קוד ברקע, כמו כן גם לשפת התכנות של הדף אין חשיבות, מכיוון שתשתמש בה רק בחלק הדף שמכיל סימני עריכה.
2. ודא שהדף מוצג ב-Markup View, ואז אתר את התגית <head> בדף. לפני תגית זו הקלד את הקוד המוצלל שלהלן:

```
<title>Untitled Page</title>
<style type="text/css">
</style>
```

</head>

שים לב שברגע שהקלדת את סוגר הפתיחה הזוויתי (<) צצה רשימה שמאפשרת לך לבחור בתגית <style>. הדבר נכון גם לגבי המאפיין type. הקלד את האותיות ty והמאפיין type ייבחר עבורך מהרשימה. כל מה שעליך לעשות כדי להשלים את המילה הוא להקיש Tab או Enter. זכור שאותה עזרה זמינה גם עבור הערך text/css של המאפיין. לשם כך בחר בערך מהרשימה והקש Tab או Enter, והערך יוכנס עבורך באופן אוטומטי כשהוא מוקף בגרשיים.

3. בין תגיות הפתיחה והסגירה <style> הקלד את קוד CSS המוצלל הבא:

```
<style type="text/css">
h1
{
font-size: 20px;
color: Green;
}
```

```
p
{
    color: Blue;
    font-style: italic;
}

.RightAligned
{
    text-align: right;
}
```

</style>

הקלד את הקוד בצורה מדויקת, מכיוון ש-CSS מקפידה מאוד בענייני תחביר! הפריט הראשון ברשימה הוא תגית h1 שמיועדת לעיצוב סגנון כותרת ברמה הראשונה, כדי שהתווים שבה יהיו בגודל 20 פיקסל, והיא תוצג בגופן ירוק. שים לב לתו נקודתיים (: שבין font-size ו-20px, ולכך שהשורה נסגרת בנקודה-פסיק (;).

הפריט השני ברשימה מכיל את האות p ומגדיר את המראה והתחושה של כל האלמנטים <p> בדף.

הפריט האחרון מתחיל בנקודה (. ולאחריו הטקסט RightAligned. פריט זה משמש ליישור לימין של קטעי טקסט מסוימים בדף.

4. גלול מעט מטה בדף עד שתראה את תגית הפתיחה <div>. מייד לאחר תגית זו הקלד את הטקסט המוצלל הבא:

<div>

```
<h1>Welcome to this CSS Demo page</h1>
```

```
<p>CSS makes it super easy to style your pages.</p>
```

```
<p class="RightAligned">
```

```
    With very little code, you can quickly change the looks of a page.
```

```
</p>
```

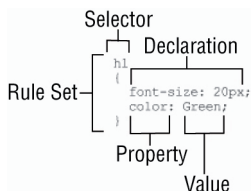
```
</div>
```

במקום להקליד קוד זה ישירות, אתה יכול להשתמש בסרגל הכלים Formatting, מתוך Design View, כדי ליצור אלמנטים כמו <h1> ו-<p>. בינתיים תצטרך להקליד class="RightAligned", אבל בתרגילים הבאים, תלמד איך לגרום ל-IDE לכתוב קוד זה עבורך.

5. אם תעבור ל- Design View תבחין כי הטקסט שלך מוצג בהתאם להגדרת העיצוב שקבעת באלמנט <style> של הדף. תרשים 1-3 מציג את הדף ב- Split View ואתה יכול לראות כעת את הקוד והעיצוב בו-זמנית.

הספר מודפס בשחור-לבן ולכן קשה להבחין בהבדלים בצבעי הגופנים. לעומת זאת, ניתן לראות בתרשים 1-3 בכירור של-<h1> יש גופן גדול יותר מאשר לטקסט. בתרשים גם רואים שכל הפסקאות (הן הרגילה והן המעוצבת עם class="RightAligned") מוצגות

{ } . הבלוק נקרא **rule set** או בפשטות **rule** (כלל). הכלל מגדיר את התצוגה עבור כל האלמנטים <h1> בדף שלך. ה- <h1> בראש בלוק הקוד נקרא **selector** (בורר) ותפקידו להצביע על האלמנטים שלגביהם יחול כלל העיצוב. במקרה שלנו, ה-selector ממופה ישירות לאלמנט <h1> של HTML, אבל זמינים גם selectors אחרים שתראה מאוחר יותר בסעיף זה. תרשים 3-2 מראה איך האלמנטים קשורים ביניהם.



תרשים 3-2

בין הסוגריים המסולסלים אתה רואה את מידע סגנון העיצוב שברצונך להחיל על הכותרת. כל שורה בין הסוגריים המסולסלים נקראת **הצהרה** (declaration). הצהרה מורכבת ממאפיין (property) שלאחריו נקודתיים (:), ולאחריהן ערך (value). התו נקודה-פסיק (;) בסוף ההצהרה מפריד אותה מההצהרה הבאה, והוא חייב להיות בכל ההצהרות, פרט להצהרה האחרונה. עם זאת, מטעמי עקביות, יהיה זה נוהג נכון לסיים את כל ההצהרות בנקודה ופסיק, וכך תראה בכל הדוגמאות הבאות שבספר.

כשהדפדפן טוען דף זה, הוא קורא את הסגנונות שהגדרת בין תגיות <style>. בכל פעם שהדפדפן פוגש באלמנט HTML שתואם את ה-selector, הוא מחיל עליו את כללי CSS הרלוונטיים לו. כך מוחלים הכללים שמופיעים בצעד 3 על האלמנטים <h1> ו- <p>, בהתאמה. הדבר יגרום לכותרת להופיע בגופן גדול בצבע ירוק, בעוד שהפסקה תוצג בצבע כחול וגופן נטוי.

אבל מדוע הופיעה הפסקה האחרונה בצבע כחול כשהיא מיושרת לימין? ב-CSS הכללים יכולים לבוא ממקורות שונים. תגית <p> האחרונה מקבלת את מידע הסגנון שלה מבורר p סטנדרטי שבהגדרת הסגנון, והוא זה שקבע לפסקה גופן כחול ונטוי. עם זאת, בנוסף הוצמד לתגית <p> דרך מאפיין Class סגנון נוסף בשם RightAligned, שגורם לטקסט להיות מיושר לימין החלון. בסופו של דבר, אלמנט <p> האחרון מקבל את הכללים שלו בו-זמנית משני מקורות.

בסעיף הבא נתעמק הרבה יותר בתחביר ובמבנה של CSS.

שפת CSS

כפי שראית בתרגיל נסה זאת הקודם, CSS היא למעשה אוסף של כללים. כלל הוא שילוב כלשהו של selector והצהרה אחת או יותר המכילה מאפיין וערך. קרוב לוודאי שאתה מסוחרר מעט מכל המונחים החדשים שהכרת בפסקאות האחרונות, לפיכך תפגוש את רובם שוב בסעיף הבא, בשילוב הסבר מפורט ודוגמאות קוד שיראו לך למה הן משמשות וכיצד הן פועלות.

גיליון סגנון (Style Sheet)

גיליון הסגנון מכיל את כל מידע הסגנון הרלוונטי שצריך ליישמו לאלמנטים של דף. בצורתו הפשוטה ביותר, גיליון סגנון נראה כך:

```
h1
{
  color: Green;
}
```

גיליון סגנון יכול לכלול יותר מכלל אחד, כפי שראית בתרגיל הקודם. בו בזמן, כל כלל יכול להכיל כמה הצהרות, ומאפשר לך לקבץ אותן תחת selector יחיד, כמו בדוגמה:

```
h1
{
  font-size: 20px;
  color: Green;
}
```

קוד שלעיל זהה מבחינה פונקציונלית לקוד זה:

```
h1
{
  font-size: 20px;
}
h1
{
  color: Green;
}
```

הצורה הדחוסה, שבה שתי ההצהרות מקובצות יחד תחת אותו selector, קלה הרבה יותר לקריאה ולהבנה. לכן מומלץ להשתמש בתחביר זה ככל האפשר.

כדי שדפדפן יוכל לסגנון אלמנט של דף, עליו לדעת שלושה דברים:

- ⊙ איזה אלמנט של הדף צריך לסגנון?
- ⊙ איזה חלק מהאלמנט צריך להיות מסוגל?
- ⊙ איך אתה רוצה שייראה חלק האלמנט שנבחר?

התשובות לשאלות אלו ניתנות על ידי selectors, מאפיינים וערכים.

Selectors (בוררים)

selector משמש כמצביע על סוג אלמנט מסוים מהדף שלך. כאשר נמצאים מספר selectors שונים הזמינים עבורך, הדבר מקנה לך שליטה טובה על בחירת האלמנט שברצונך לסגנון. ה-selector עונה על השאלה הראשונה: איזה אלמנט של הדף צריך לסגנון? להלן תיאור ארבעת הסוגים החשובים ביותר של selectors.

Universal selector

selector זה הינו אוניברסלי ומסומן בכוכבית (*), והוא חל על כל האלמנטים בדף. ניתן להשתמש בו לקביעת הגדרות גלובליות כמו משפחת גופנים. הכלל הבא משנה את הגופן של כל האלמנטים בדף שלך ל-Arial:

```
*
{
    font-family: Arial;
}
```

Type selector

selector זה מאפשר להצביע על אלמנט HTML מסוים, וגורם לכך שכל האלמנטים מאותו הסוג יסוגנו בהתאם.

```
h1
{
    color: Green;
}
```

בדוגמה זו העיצוב חל על כל התגיות מסוג <h1> בקוד שלך, וקובע עבורם צבע ירוק. selector מסוג Type אינם רגישים לרישיות, ועל כן תוכל להשתמש הן ב-h1 והן ב-H1 כדי להתייחס לאותה כותרת.

ID Selector

selector זה מאפשר לזהות ולהתייחס לאלמנט יחיד בדף. הגדרת ה-selector תבצע על ידי שימוש בתחילית (#). לכל אלמנט בדף HTML או ASPX אפשר לקבוע ID ייחודי על ידי שימוש במאפיין id. עם ID selector אתה יכול לשנות את ההתנהגות של אלמנט יחיד זה. למשל:

```
#IntroText
{
    font-style: italic;
}
```

המזהה (ID) חייב להיות ייחודי רק בתחומי דף נתון. מכיוון שייתכן שימוש חוזר באותו מזהה גם בדפים אחרים באתר שלך. אפשר להשתמש בכלל זה כדי לשנות במהירות את המראה של האלמנט שאתה משתמש בו מספר פעמים. קוד HTML הבא מדגים זאת:

```
<p id="IntroText">I am italic because I have the right ID.</p>
<p id="BodyText">I am NOT italic because I have a different ID.</p>
```

בדוגמה זו הבורר בשם #IntroText משנה את הגופן של הפסקה הראשונה - שהיא בעלת מאפיין id זהה - אבל משאיר את הפסקה השנייה ללא שינוי.

Class Selector

selector זה מאפשר לסגנן מספר אלמנטים של HTML על ידי המאפיין class. הדבר נוח כשברצונך לקבוע עיצוב דומה לכמה אלמנטים בלתי קשורים של HTML. הכלל הבא משנה את הגופן למודגש בכל האלמנטים של HTML שהמאפיין class שלהם הוגדר כ-Highlight:

```
.Highlight
{
  font-weight: bold;
  color: Red;
}
```

קטע הקוד הבא משתמש ב-Class selector בשם Highlight כדי לגרום לתוכן של האלמנטים ו-<a> להופיע עם גופן מודגש בצבע אדום.

```
This is normal text but <span class="Highlight">this is Red and Bold</span>
This is also normal text but
<a href="CssDemo.aspx" class="Highlight">this link is Red and Bold as well</a>
```

שים לב שבהגדרת הבורר מופיעה נקודה (.), אבל אתה אינך משתמש בנקודה זו כשאתה מתייחס לבורר במאפיין class. המאפיין class שימושי מאוד מכיוון שהוא מאפשר לחזור ולהשתמש באותו קטע קוד של CSS למטרות רבות ושונות, ללא קשר לאלמנט HTML שמשתמש ב-class.

CSS תומכת בסוגים נוספים של selectors שנותנים שליטה טובה עוד יותר על האלמנטים שאתה רוצה להתמקד בהם, אבל השימוש הנפוץ ביותר הוא בארבעת הסוגים שראית זה עתה.

הקבצה ושילוב של Selectors

CSS מאפשרת לקבץ כמה selectors. יכולת זו שימושית כשברצונך להחיל את אותם הסגנונות על אלמנטים שונים. הכלל הבא הופך את כל הכותרות בדף לאדומות:

```
h1, h2, h3, h4, h5, h6
{
  color: Red;
}
```

יתר על כן, עם CSS ניתן לשלב selectors. יכולת זו מאפשרת להצביע באופן היררכי על אלמנט מסוים בדף. מבצעים זאת על ידי הפרדה ברווח בין ה-selectors. הדוגמה הבאה מתמקדת בכל האלמנטים <p> שנמצאים בתחומו של אלמנט בעל מאפיין id בשם MainContent:

```
#MainContent p
{
  font-size: 18px;
}
```

הכלל משפיע רק על הפסקאות בתחום האלמנט MainContent, בעוד שכל שאר הפסקאות נשארות כפי שהן.

שים לב ששילוב (combining) שונה מאוד מהקבצה (grouping). הקבצה היא רק קיצור דרך שמטרתה לחסוך הקלדה חוזרת של אותן הצהרות שוב ושוב. שילוב מאפשר לכוון לאלמנטים מסוימים במסמך.

פעולת השילוב אינה מוגבלת לבוררים מסוג ID ו-Type. באפשרותך להשתמש בה גם עם בוררים אחרים, כפי שמדגימה הדוגמה הבאה:

```
#MainContent p.Highlight
{
    font-size: 18px;
    font-weight: bold;
}
```

כלל זה משנה את כל הפסקאות בעלות מאפיין Class שערכו Highlight ונמצאות בתוך אלמנט בעל מזהה (מאפיין id) בשם MainContent, כל שאר הפסקאות תשארנה ללא שינוי. קטע HTML הבא יבהיר את השפעת הכלל:

```
<div id="MainContent">
    <p class="Highlight">Because I have a class called Highlight, my text appears in bold</p>
    <p>This text is not bold, as it lacks the Highlight class</p>
</div>
<p class="Highlight">I am NOT bold because I don't fall within MainContent</p>
```

השאלה השנייה שיש לענות עליה כדי ליישם סגנון מסוים בדף היא: איזה חלק מהאלמנט חייב להיות מסוגנן? הדבר מתבצע עם מאפיינים.

מאפיינים (Properties)

מאפיינים הם אותו חלק של האלמנט שברצונך לשנות על ידי גיליון הסגנון שלך. מפרט CSS מגדיר רשימה ארוכה של מאפיינים (מגנון הרשימה IntelliSense של VWD מציג למעלה ממאה פריטים), אם כי רוב אתרי האינטרנט אינם משתמשים בכולם. בטבלה הבאה מופיעים כמה מהמאפיינים היותר נפוצים של CSS, תיאורם, וכיצד להשתמש בהם.

מאפיין	תיאור	דוגמה
background-color background-image	מציין את צבע הרקע או תמונת הרקע של אלמנט	background-color: White; background-image: url(Image.jpg);
border	מציין את גבולות האלמנט	border: 3px solid black;
color	משנה את צבע הגופן	color: Green;
display	משנה את הדרך שבה מוצגים אלמנטים. מאפשר להסתירם או להראותם.	display: none; בדוגמה זו האלמנט מוסתר, ואינו תופס שטח מסך.

מאפיין	תיאור	דוגמה
float	מאפשר "להציף" אלמנט בדף, הצפה ימנית או שמאלית. שאר התוכן נמצא בצד הנגדי.	<code>float: left;</code> הגדרה זו גורמת לתוכן שלאחר ה- <code>float</code> להיות מוצב בפינה הימנית העליונה של האלמנט. יותר מאוחר בפרק זה נראה כיצד מאפיין זה פועל.
font-family font-size font-style font-weight	משנה את המראה של גופנים שמשמשים בהם בדף.	<code>font-family: Arial;</code> <code>font-size: 18px;</code> <code>font-style: italic;</code> <code>font-weight: bold;</code>
height width	קובע את הגובה והרוחב של אלמנטים בדף.	<code>height: 100px;</code> <code>Width: 200px;</code>
margin padding	קובע את כמות השטח החופשי שבתוך (padding) ומחוץ (margin) האלמנט.	<code>margin: 20px;</code> <code>padding: 0;</code>
visibility	קובע אם אלמנט יוצג בדף או לא. אלמנטים שאינם נראים עדיין תופסים שטח מסך; אתה רק אינך רואה אותם.	<code>visibility: hidden;</code> פקודה זו גורמת להסתרת האלמנט. עם זאת, הוא עדיין תופס את השטח המקורי שלו בדף, כאילו שהאלמנט עדיין שם, אלא שהוא בלתי נראה.

למרבה המזל, VWD, בעזרת כלי CSS הרבים שלה, עוזרת למצוא את המאפיין הנכון, כדי שלא תצטרך לזכור את כולם. כדי שמאפיין יהיה שימושי עליך לתת לו ערך. פעולה זו עונה על השאלה השלישית: איך אתה רוצה שייראה אותו חלק של האלמנט הנבחר?

ערכים (Values)

כמו מאפיינים, כך גם לערכים יש כמה אפשרויות בחירה. הערכים שזמינים עבורך תלויים במאפיין. לדוגמה, המאפיין color מקבל ערכים שמייצגים צבעים. הדבר יכול להיות שם של צבע (לבן, למשל), או על ידי שימוש ב-RGB עם אפשרות לשימוש במספר הקסדצימלי שמייצג צבע (#FF0000), או הגדרת RGB (קביעת כמות צבעי היסוד: אדום, ירוק וכחול). בדוגמה הבאה השימוש בסגנון יניב תוצאה זהה לשלוש ההצהרות.

```
h1
{
  color: Red;
}
```

```
h1
{
  color: #FF0000;
}

h1
{
  color: rgb(100%, 0%, 0%);
}
```

ההצהרה הראשונה משתמשת בשם הצבע אדום, בעוד ששתי האחרות משתמשות בערך RGB כדי לציין צבע אדום. שימוש בשמות הצבעים יכול להגדיל את יכולת הקריאה וההבנה של קוד CSS שלך, אבל מכיוון שאתה מוגבל לרשימה קצרה יחסית של שמות צבעים, תזדקק לעתים קרובות לסימון ההקסדצימלי כדי לקבל את הצבע המדויק שאתה רוצה בו.

ישנם מגוון אפשרויות לקביעת ערכים אפשריים רבים אחרים, כולל יחידות גודל (px, em וכו'), משפחות גופנים, תמונות (אשר שם הקובץ הינו בתבנית URL), או רשימה כמו border-style, אשר מאפשרת להגדיר את סגנון הגבול כ-solid, dashed, double וכו'.

שימוש בהגדרה מקוצרת

רבים ממאפייני CSS מאפשרים לכתוב גרסה מקוצרת של הגדרותיהם, כמו גם גרסה מורחבת יותר. קח, לדוגמה, את המאפיין border. בצורתו הקצרה ביותר הוא יכול להיות מוגדר כך:

```
border: 1px solid Black;
```

מאפיין border זה מיישם גבול לכל אחד מארבעת צדדיו של אלמנט HTML. גודל הגבול יהיה 1px, הסגנון יהיה solid (כמה מהאפשרויות האחרות הן dotted, dashed ו-double), וצבע הגבול יהיה שחור.

זו דרך קלה להגדיר את כל ארבעת הגבולות של HTML עם אותם ערכים. עם זאת, אם ברצונך לשלוט טוב יותר על כל גבול בנפרד ועל המאפיינים של הגבולות, אתה יכול להשתמש בגרסה המורחבת, באופן זה:

```
border-top-width: 1px;
border-top-style: solid;
border-top-color: Black;
border-right-width: 1px;
border-right-style: solid;
border-right-color: Black;
border-bottom-width: 1px;
border-bottom-style: solid;
border-bottom-color: Black;
border-left-width: 1px;
border-left-style: solid;
border-left-color: Black;
```

גרסה ארוכה זו תגרום להצגה זהה כמו בדיוק כמו הגרסה המקוצרת: גבול שחור מלא בכל ארבעת הצדדים, בעובי של 1px. ברוב המקרים עליך להעדיף את הסימון המקוצר על פני מקבילו המורחב, מכיוון שהראשון קל הרבה יותר לקריאה ולתחזוקה. עם זאת, אם אתה צריך שליטה מוחלטת על הגבולות – לדוגמה, אם ברצונך גבול מקווקו בעובי 2px מצד שמאל ולמעלה, וגבול מלא וירוק מצד ימין ומצדו התחתון של אלמנט HTML – טוב לדעת שהדבר אפשרי וניתן להגדיר בנפרד כל מאפיין גבול לכל אחד מארבעת הכיוונים באופן פרטי.

מאפייני CSS אחרים שתומכים בגרסה מקוצרת כוללים את font, background, list-style, padding-1 margin. אם אינך בטוח שמאפיין מסוים תומך בגרסה מקוצרת, השתמש ברשימה של IntelliSense, שמוצגת לאחר הקשת Ctrl+Space בזמן שאתה מכניס ערך לקובץ CSS או לבלוק <style> ב-VWD.

בתרגיל הבא תשנה את דף הבית של האתר אשר יצרת בפרק הקודם. כעת תוסיף את המתווה הבסיסי של האתר, ולאחר מכן תסגנון אותו תוך שימוש בגיליון סגנון. בפרק 6 תשתמש בדף זה שוב כשתשדרג אותו לדף ראשי.

נסה זאת עיצוב דף הבית של Planet Wrox

בתרגיל זה תשנה שני קבצים. תחילה תוסיף את האלמנטים הבסיסיים של המתווה לדף Default.aspx, כדי ליצור מקום לכותרת, תפריט, אזור התוכן העיקרי, סרגל צד וכותרת תחתונה. לאחר מכן תשנה את הקובץ Styles.css שבתיקייה Styles באופן שהגודל והמיקום של האלמנטים הבסיסיים ישתנו. לבסוף תחבר את גיליון הסגנון לדף, כדי שמידע העיצוב ייושם בעת שהדף יוצג ב-designer או בדפדפן.

1. ב-Markup View פתח את הקובץ Default.aspx.
2. מייד לפני טקסט Welcome שיצרת בפרק הקודם, הוסף את קוד HTML המוצלל הבא. ודא שאתה מחליף את ה-HTML הקיים בתוך האלמנט <form>.

```
<form id="form1" runat="server">
  <div id="PageWrapper">
    <div id="Header">Header Goes Here</div>
    <div id="MenuWrapper">Menu Goes Here</div>
    <div id="MainContent">
    </div>
    <div id="Sidebar">Sidebar Goes Here</div>
    <div id="Footer">Footer Goes Here</div>
  </div>
  <h1>Hi there visitor and welcome to Planet Wrox</h1>
  ...
</form>
```

קוד זה משתמש באלמנטים <div> כדי לקבוע אזורים חשובים בדף, כמו הכותרת, התפריט ואזור התוכן הראשי.

3. בחר בכותרת ובפסקאות (מיוצגות בקטע הקוד על ידי ...) שהוספת לך זה קודם לכן, גזור אותם (על ידי סימונם ואז לחץ לחיצה ימנית ובחר ב-Cut או הקש על שילוב המקשים Ctrl+X) והדבק אותם בין תגיות הפתיחה והסגירה של תגית <div> בעלת מזהה MainContent (לחץ לחיצה ימנית במיקום ובחר Paste או הקש על שילוב המקשים Ctrl+V). מהלך זה יעביר את הקוד הקיים שלך אל בלוק התוכן הראשי שהוגדר בתגית <div> בעלת מזהה MainContent. אתה אמור לסיים עם משהו שדומה לזה:

```
<form id="form1" runat="server">
  <div id="PageWrapper">
    <div id="Header">Header Goes Here</div>
    <div id="MenuWrapper">Menu Goes Here</div>
    <div id="MainContent">
      <h1>Hi there visitor and welcome to Planet Wrox</h1>
      ...
    </div>
    <div id="Sidebar">Sidebar Goes Here</div>
    <div id="Footer">Footer Goes Here</div>
  </div>
</form>
```

ודא שהקוד ברף דומה לקוד מצעד 2. חשוב להקפיד שתגיות <div> השונות וכן המזהים שלהם יוצגו כהלכה, טקסט welcome עצמו הוא בעל חשיבות משנית.

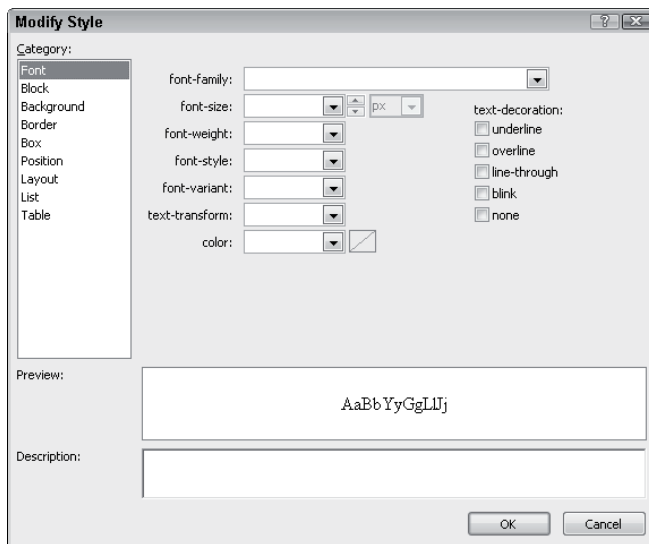
4. פתח את הקובץ Styles.css מהתיקייה Styles. אם הוספת לקובץ זה קוד בתרגילי נסה זאת קודמים - מחק את תוכנו.

5. בראש הדף הקלד את הקוד הבא, אשר משתמש ב-ID selector כדי לבחור בתגית <div> בעלת מזהה Header:

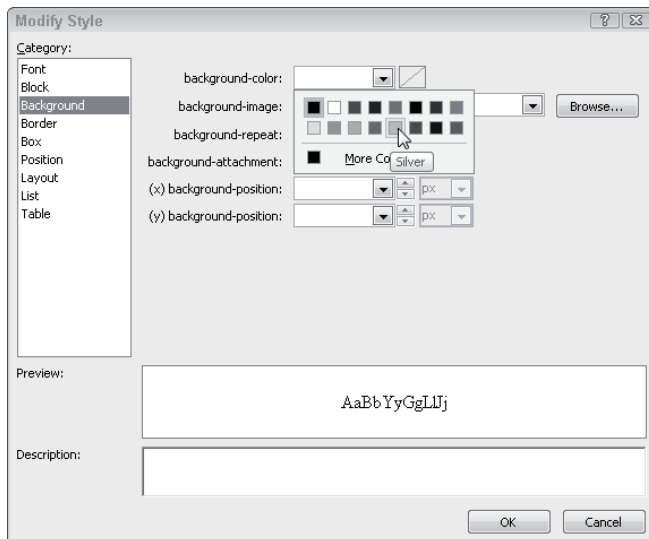
```
#Header
{
}
}
```

6. הצב את הסמן בין הסוגריים המסולסלים ובחר Styles ← Build Style מהתפריט הראשי. לחילופין, אתה יכול לבחור באותו פריט על ידי לחיצה ימנית על #Header, או על ידי לחיצה על הלחצן Build Style שבסרגל הכלים Styles. כעת תופיע תיבת הדרו-שיח Modify Style שמוצגת בתרשים 3-3.

7. ברשימה Category שמשמאל לחץ על Background. פתח את הרשימה הנפתחת עבור צבע הרקע. מאוסף הצבעים שמופיע לחץ על הצבע Silver, כפי שמוצג בתרשים 3-4. לחילופין, אתה יכול להקליד את הקוד ההקסדצימלי עבור הצבע Silver (#C0C0C0) ישירות בתיבת הטקסט background-color.



תרשים 3-3



תרשים 3-4

8. עבור לקטגוריית Position. הלוח שמופיע מאפשר להגדיר מידע תלוי-מיקום, כולל הגובה והרוחב. עבור הרוחב הכנס 844, וודא שברשימה הנפתחת שמימין נבחר px. עבור הגובה הכנס 83. לחץ OK כדי לשחרר את תיבת הדו-שיח, כעת הקוד נראה כך:

```
#Header
{
  background-color: #C0C0C0;
  width: 844px;
  height: 83px;
}
```

9. חזור על צעדים 5-8, אך הפעם צור את הכללים הבאים:

```
*
{
    font-family: Arial;
}

h1
{
    font-size: 20px;
}

#PageWrapper
{
    width: 844px;
}

#MenuWrapper
{
    width: 844px;
}

#MainContent
{
    width: 644px;
    float: left;
}

#Sidebar
{
    background-color: Gray;
    width: 200px;
    float: left;
}

#Footer
{
    background-color: #C0C0C0;
    width: 844px;
    clear: both;
}
```

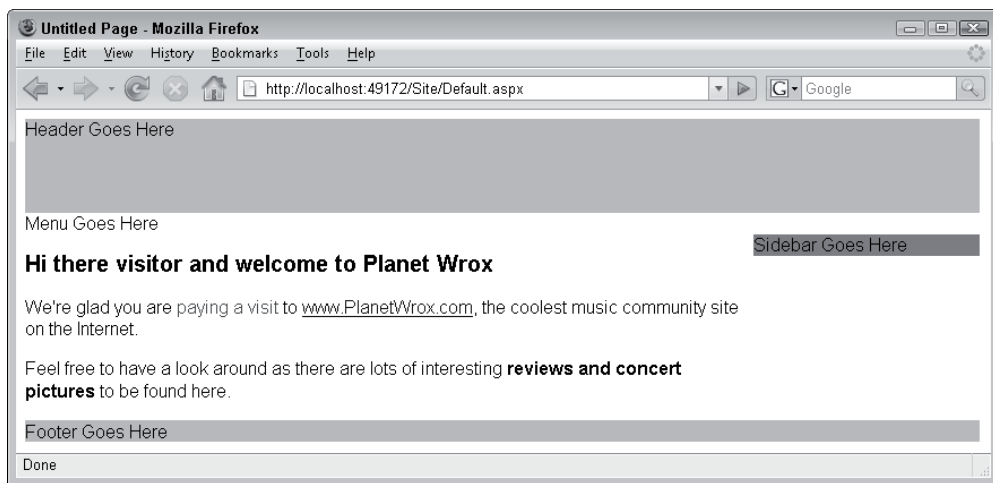
את המאפיינים float ו-clear תמצא בקטגוריה Layout של תיבת הדו-שיח Modify Style. 10. כשתסיים ליצור את הכללים, שמור וסגור את הקובץ Styles.css. בינתיים לא תזדקק לו עוד.

11. פתח שוב את הקובץ Default.aspx ועבור ל- Design View. מתוך סייר הפתרון גרור את הקובץ Styles.css מהתיקייה Styles אל הדף. VWD מוסיפה קוד לקטע <head> של הדף ב- Markup View. קוד זה מחבר את גיליון הסגנון למסמך.

```
<head runat="server">
  <title>Untitled Page</title>
  <style type="text/css">
    .style1
    {
      color: #FF0000;
    }
  </style>
  <link href="Styles/Styles.css" rel="stylesheet" type="text/css" />
</head>
```

אתה גם יכול לגרור גיליון סגנון קיים מתוך סייר הפתרון ישירות לקטע <head> של דף ב- Markup View. כשתעשה זאת, VWD תוסיף את אותו אלמנט <link>.

12. לבסוף, שמור את השינויים לכל המסמכים הפתוחים (הקש Ctrl+Shift+S), ואז הצג את דף Default.aspx בדפדפן שלך על ידי הקשה על Ctrl+F5. המסך שלך צריך להיראות דומה לתרשים 3-5, אשר מציג את הדף בדפדפן Mozilla Firefox.



תרשים 3-5

כיצד זה פועל

הכלי Style Builder מקל על בחירת מאפייני CSS ועל שינוי ערכיהם. אינך צריך לזכור כל פרט קטן אודות CSS, ובמקום זאת אתה יכול ליצור את קוד CSS שלך באופן חזותי. למרות שהכלי Style Builder יכול לעשות עבורך את רוב העבודה, היכולת לקרוא ולהבין קוד CSS עדיין שימושית. לעתים, כשאתה צריך לעשות התאמות קטנות בקוד שלך, מהיר יותר לעשות זאת בחלון המסמך מאשר לפתוח את ה- Style Builder.

שנים לב של-Header, PageWrapper, MenuWrapper, ול-Footer יש רוחב מדויק של 844 פיקסל. בדרך זו האתר מתאים היטב למסכים בגודל 1024X768 פיקסל, שהוא גודל מסך נפוץ. מערכות בעלות מסכים גדולים יותר יגדילו את הרקע הלבן בימין הדף.

שנים לב שהאזורים MainContent ו-Sidebar נמצאים זה לצד זה. הדבר מתבצע בעזרת המאפיין float של CSS:

```
#MainContent
```

```
{  
    width: 644px;  
    float: left;  
}
```

```
#Sidebar
```

```
{  
    background-color: Gray;  
    width: 200px;  
    float: left;  
}
```

כללים אלה מורים ל-MainContent "לצוף" בצד השמאלי של Sidebar, שלמעשה מציב את Sidebar לימין התוכן. הרוחב המאוחד של שני האלמנטים מסתכם ב-844 פיקסל, שהוא בדיוק הרוחב של אלמנט-האב שלהם, ה-PageWrapper.

כדי לסיים את תוקף הגדרת המיקום ולהורות לאלמנט Footer להתמקם ישירות מתחת לאלמנטים MainContent ו-Sidebar, משתמשים במאפיין clear אשר מנקה הגדרת מיקום כלשהי (ימנית או שמאלית) שעשויה להיות בתוקף.

```
#Footer
```

```
{  
    background-color: #C0C0C0;  
    width: 844px;  
    clear: both;  
}
```

הרקעים האפורים נוספים לקוד רק באופן זמני, כדי שיהיה קל יותר לראות איזו תגית <div> מסתיימת, והיכן. בתרגילים שתעשה מאוחר יותר, תתבקש לשנות שוב את קובץ CSS, כדי שיתאים לסכימה של האתר Planet Wrox.

קישור הדף לקובץ ה-CSS מתבצע באמצעות הוראה זו בראש הדף:

```
<link href="Styles/Styles.css" rel="stylesheet" type="text/css" />
```

הוראה זו אומרת לדפדפן לגשת לתיקייה Styles, לחפש קובץ בשם Styles.css וליישם את כל הכללים שבקובץ זה על הדף הנוכחי. לאחר שהדפדפן טוען את קובץ CSS, הוא מיישם את כל הסגנונות שהוא מוצא בו אל האלמנטים של HTML שלך, פעולה שתוצאתה היא הפריסה שמוצגת בתרשים 3-5.

בתרגיל זה ראית כיצד לקשר גיליון סגנון לדף תוך שימוש בתגית <link>. עם זאת, יש לזכור שיש דרכים רבות כדי לכלול גיליונות סגנון בדפי אינטרנט.

הוספת CSS לדפים שלך

הדרך היעילה ביותר להוספת סגנונות CSS לדף אינטרנט היא באמצעות התגית <link> שמצביעה על קובץ CSS חיצוני, כפי שראית בדוגמה הקודמת. הבט בתגית <link> הבאה כדי לראות אילו אפשרויות קיימות לקישור גיליון סגנון לדף שלך.

```
<link href="StyleSheet.css" rel="Stylesheet" type="text/css" media="screen" />
```

המאפיין href מצביע על קובץ באתר שלך, בדיוק כפי שראית בפרק הקודם כשיצרת קישור בין שני דפים. התכונות rel ו-type אומרות לדפדפן שהקובץ המקושר הוא למעשה גיליון סגנון מסוג CSS. התכונה media מעניינת למדי: היא מאפשרת לייעד את הדף אל התקנים שונים, וביניהם המסך, מדפסת, מכשירי כף יד, ואפילו כלי תמיכה למכשירים המיועדים לאנשים בעלי מוגבלויות. ברירת המחדל של התכונה media היא המסך, ולכן ניתן להשמיט מאפיין זה כאשר אתה מייעד את הדף אל דפדפנים תקינים.

דרך אחרת לכלול גיליונות סגנון הוצגה בקצרה בתחילת הפרק: שימוש באלמנטים <style> כחלק בלתי נפרד מהדף. התגית <style> חייבת להימצא בראש דף ASPX או HTML שלך, בין תגיות <head>. בתוך תגיות <style> אתה יכול לכתוב את אותו קוד CSS שראית קודם. לדוגמה, כדי לשנות את המראה של אלמנט <h1> בדף הנוכחי בלבד, אתה יכול להוסיף את הקוד הבא ל- <head> של הדף שלך:

```
<head runat="server">
  <title>Untitled Page</title>
  <style type="text/css">
    h1
    {
      color: Blue;
    }
  </style>
</head>
```

הדרך האחרונה ליישום CSS על אלמנטים של HTML היא לכתוב את ההצהרה באופן ישיר בתוך הגדרת התגיות השונות תחת המאפיין style שראית בפרק הקודם:

```
<span style="background-color: Black; color: White;">
  This is white text on a black background
</span>
```

בחירת האפשרות לקישור גיליונות סגנון

מכיוון שיש אפשרויות כה רבות להוסיף סגנון לאתר שלך, נשאלת השאלה: מהי השיטה הטובה ביותר לעשות זאת? באופן כללי, עליך להעדיף גיליונות סגנון חיצוניים על פני גיליונות המוטמעים בדף, אשר עדיפים על סגנונות הכלולים באלמנטים. גיליונות סגנון חיצוניים

מאפשרים לשנות את המראה של האתר כולו באמצעות קובץ יחיד. עשה שינוי אחד בקובץ גיליון הסגנון החיצוני שלך, וכל הדפים שמשתמשים בסגנון זה ישתנו באופן אוטומטי.

כאשר ברצונך לשנות את המראה של דף יחיד, מבלי להשפיע על דפים אחרים באתר, אזי שימוש בגיליון סגנון המוטמע בדף זו הבחירה הטובה ביותר.

כאשר ברצונך לשנות את ההתנהגות של אלמנט אחד בדף אחד, ואתה בטוח למדי שלא תזדקק לאותה הצהרה עבור אלמנטים אחרים של HTML, השתמש בסגנון הכלול בהצהרת האלמנטים.

עניין חשוב שיש לתת עליו את הדעת הוא סדר הקדימויות כאשר ישנם גיליונות סגנון חופפים. אם יש לך selectors זהים רבים, עם ערכים שונים למאפיינים, ה-selector שמוגדר אחרון מקבל קדימות. לדוגמה, כלל שמוגדר בגיליון סגנון חיצוני בשם Styles.css, ואשר קובע כי כל תגיות <h1> יהיו בעלי צבע ירוק:

```
h1
{
    color: Green;
}
```

כעת דמיין שאתה מספח גיליון סגנון זה לדף שיש לו הגדרת כלל כחלק מהדף עבור אותו h1, אלא שכלל זה קובע צבע אחר:

```
<link href="Styles/Styles.css" rel="stylesheet" type="text/css" />
<style type="text/css">
h1
{
    color: Blue;
}
</style>
```

עם קוד זה, צבע התגית <h1> בפועל שמוצגת בדף יהיה כחול, משום הדפדפן קורא את הנחיות העיצוב מלמעלה למטה, ומכיוון שההגדרה האחרונה (התחתונה) דרסה את הקביעה של הקובץ החיצוני. הדבר מדגים את רעיון הדירוג של גיליונות סגנון מדורגים, כשסגנון אחד מדורג מתחת לאחר. אם התפוך את סדר הסגנונות כך:

```
<style type="text/css">
h1
{
    color: Blue;
}
</style>
<link href="Styles/Styles.css" rel="stylesheet" type="text/css" />
```

הכותרת תהיה ירוקה מכיוון שההגדרה בגיליון הסגנון החיצוני מופיעה לאחר ההגדרה המקומית. אותו עקרון חל גם על גיליונות סגנון הכלולים באלמנטים של HTML, ההגדרות שלהם מקבלות קדימות.

הערה: יש ב-CSS הרבה יותר ממה שהראינו כאן. כדי ללמוד יותר על CSS, עיין בספר "HTML & CSS למפתחי אתרים באינטרנט" מאת זהר עמיהוד, הוצאת הוד-עמי.

ככלל, מומלץ להגדיר ראשית קישור לקבצים חיצוניים בראש הקטע <head>, ולאחריהם את הגדרת הסגנון ברמת הדף. בדרך זו יגדירו הקבצים החיצוניים את המראה הגלובלי של אלמנטים, ותוכל להשתמש בסגנונות המוטמעים בדף כדי לדרוס את ההגדרות החיצוניות.

VWD מקלה על העברת גיליונות סגנון המוטמעים בדף לקובץ CSS חיצוני, כפי שמוסבר בסעיף הבא, אשר דן בשאר הכלים של CSS הנתמכים ב-VWD.

עבודה עם CSS ב- Visual Web Developer

ל-VWD יש כמה כלים שימושיים לעבודה עם CSS. רובם הופיעו לראשונה בגרסת 2008 של VWD, מכיוון שתמיכה טובה ב-CSS הייתה אחת ממטרות התכנון של גרסה זו. הכלים הבאים עומדים לרשותך:

- ⊙ **סרגל הכלים Style Sheet** – מאפשר גישה מהירה ליצירת כללים וסגנונות חדשים.
- ⊙ **טבלת המאפיינים של CSS** – מאפשרת לשנות ערכים של מאפיינים.
- ⊙ **החלון Manage Styles** – מאפשר לארגן סגנונות באתר, לשנות אותם מגיליונות סגנון מוטמעים ברמת הדף לקבצים חיצוניים ולהיפך, לשנות את סדרם, לקשור גיליונות סגנון קיימים למסמך, וליצור גיליונות סגנון חדשים במגוון האפשרויות.
- ⊙ **החלון Apply Style** – ניתן להשתמש בו כדי לבחור סגנון מתוך כל הסגנונות באתר שלך, וליישום מהיר של בחירתך לאלמנטים שונים בדף.
- ⊙ **Style Builder** – יכול לשמש ליצירת הצהרות באופן חזותי.
- ⊙ **החלון Add Style Rule** – מסייע בבניית selectors (בוררים) מורכבים יותר.

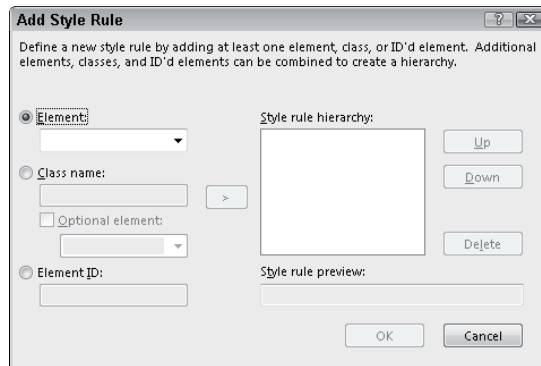
למרות שכבר ראית כמה מהכלים האלה בפרק זה, הסעיף הבא מציג מבט מפורט על הפונקציונליות שהם מציעים. תלמד איך לבצע משימות שונות כמו יצירת סגנונות חדשים, שינוי סגנונות קיימים, והחלת סגנונות על אלמנטים קיימים. במהלך ההסבר תקבל הבנה טובה של הכלים שהוזכרו כאן.

יצירת סגנונות חדשים בגיליונות סגנון חיצוני

בתרגיל נסה זאת קודם הוספת selectors לקובץ CSS באופן ידני, ואז השתמשת ב- Style Builder כדי לכתוב כללים. באפשרותך להשתמש בכלי VWD כדי לכתוב את הכללים. בתרגיל נסה זאת הבא תלמד איך להשתמש בחלון Add Style Rule ליצירת סגנון חדש בקובץ חיצוני. לאחר מכן תשתמש ב- Style Builder כדי לשנות את הכלל.

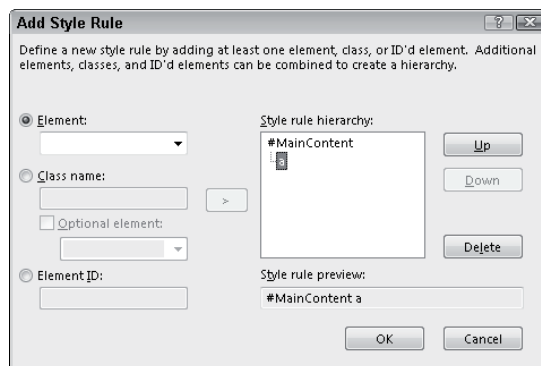
בתרגיל זה תיצור סגנון חדש שישפיע על כל הקישורים באזור MainContent. על ידי שימוש בבוררים משולבים, תוכל לשנות את העיצוב של הקישורים באזור התוכן בלבד, בעוד האחרים יישארו ללא שינוי.

1. פתח את הקובץ Styles.css מהתיקייה Styles.
2. גלול מטה בקובץ, והצב את הסמן בסופו, מייד לאחר הכלל #Footer.
3. בחר Styles ← Add Style Rule מהתפריט הראשי. כעת תופיע תיבת הדו-שיח שמוצגת בתרשים 3-6.



תרשים 3-6

- באמצעות תיבת דו-שיח זו אתה יכול ליצור בורר משולב באופן חזותי.
- בחר באפשרות האחרונה, הקרויה Element ID, ובתיבת הטקסט שלה הקלד MainContent. להוספת הבורר להיררכיה Style rule, לחץ על הלחצן עם החץ ימינה שבמרכז המסך.
4. סמן את האפשרות Element שבראש תיבת הדו-שיח ואז, מהרשימה הנפתחת שלה, בחר a (עבור קישורים) ולחץ על לחצן החץ פעם נוספת. המסך שלך צריך כעת להיראות כמו בתרשים 3-7.



תרשים 3-7

אם אינך רואה את התו # משמאל ל-MainContent בתיבת ההיררכיה של Style rule, ודא שבצעד 3 אכן בחרת ב- Element ID ולא ב-Element.

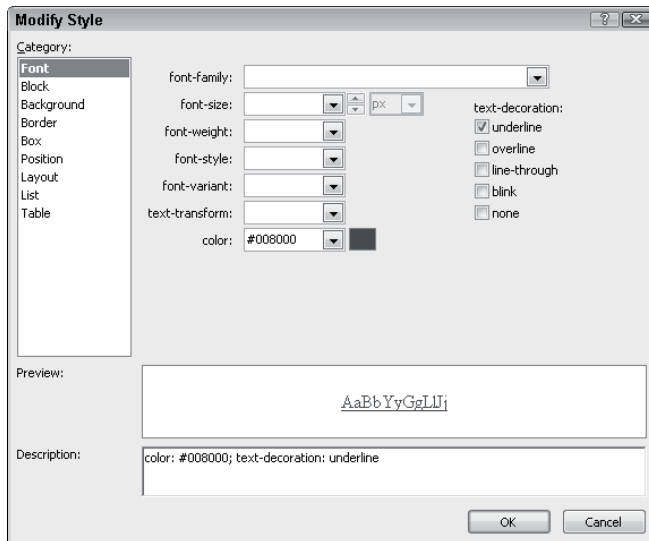
5. לחץ OK להוספת הבורר לקובץ גיליון הסגנון שלך. אתה אמור לסיים עם הכלל הריק הבא:

```
#MainContent a
{
}
}
```

6. לחץ לחיצה ימנית בין הסוגריים המסולסלים של הכלל שזה עתה הוספת, ובחר Build Style.

7. בקטגוריה Font שנה את המאפיין color ל- #008000 על ידי לחיצה על החץ של תיבת הרשימה הנפתחת, ואז לחץ על הריבוע הירוק בשורה העליונה.

8. בקטע text-decoration שבצידה הימני של תיבת הדו-שיח סמן את underline בסימן ✓. תיבת הדו-שיח Style צריכה להיראות דומה לזו שמוצגת בתרשים 3-8.



תרשים 3-8

9. לחץ OK לשחרור תיבת הדו-שיח. בחזרה לקובץ CSS, בחר את הכלל שזה עתה יצרת, העתק אותו, והדבק אותו פעמיים לאחר הכלל המקורי.

10. שנה את שם הבורר הראשון שזה עתה הדבקת מ- #MainContent a לשם #MainContent a:visited. סגנון זה ישמש עבור קישורים שהשתמש כבר ביקר בהם.

11. לחץ לחיצה ימנית על הכלל החדש שזה עתה יצרת, ובחר Build Style. בקטגוריה Font שנה את הצבע מירוק לאדום על ידי הקלדת #FF0000 בתיבת הטקסט color, ואז לחץ OK.

12. שנה את שם הבורר השלישי בקובץ מ- #MainContent a לשם #MainContent a:hover. סגנון זה ייושם לקישורים כאשר המשתמש עובר מעליהם עם העכבר.

13. פעם נוספת, לחץ לחיצה ימנית על הכלל החדש שזה עתה יצרת ובחר Build Style. בקטגוריה Font שנה את הצבע מירוק לכתום על ידי הקלדת #FFA500 בתיבת הטקסט color. לחץ OK לסגירת תיבת הדו-שיח Modify Style. אתה אמור לסיים עם שלושת הכללים הבאים בקובץ CSS שלך תחת הסגנונות שכבר היו נוכחים בו:

```
#MainContent a
{
    color: #008000;
    text-decoration: underline;
}

#MainContent a:visited
{
    color: #FF0000;
    text-decoration: underline;
}

#MainContent a:hover
{
    color: #FFA500;
    text-decoration: underline;
}
```

14. סגור ושמור את הקובץ Styles.css מכיוון שסיימת לעסוק בו בינתיים.

כיצד זה פועל

התחלת עם יצירת כלל חדש תוך שימוש בתיבת הדו-שיח Add Style Rule. לעתים קרובות תמצא שיותר קל ומהיר להקליד את הכלל ישירות בעורך הקוד. עם זאת, כשאתה יוצר כללים משולבים מורכבים, תיבת הדו-שיח Add Style Rule יכולה לעזור לך להבין וליצור את ההיררכיה של הכלל.

תיבת הדו-שיח Modify Style היא כלי מצוין ליצירת הצהרות CSS חדשות. במקום לזכור את כל המאפיינים השונים של CSS ואת ערכיהם, אתה יכול רק לבחור אותם ולשנותם בקלות בתיבת דו-שיח מאורגנת. המאפיינים השונים של CSS מקובצים תחת קטגוריות לוגיות, והדבר מקל על מציאתם ועל הכנסת שינויים בהם.

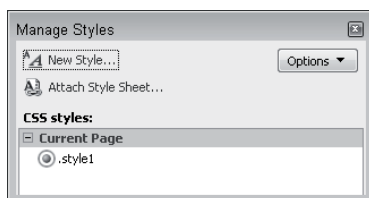
החלקים hover ו-visited: בבורר הם חדשים עבורך ככל הנראה. בוררים אלה נקראים **pseudo class selectors**. הבורר a:visited מיושם רק עבור קישורים שבהם ביקרת כבר באתר שלך. הבורר a:hover מיושם רק לגבי התגית <a> כאשר המשתמש חולף עם העכבר על פני הקישור. בתרגיל נסה זאת הבא תראה את ההשפעה של שני בוררים אלה בדפדפן.

הדבר הבא שעליך לעשות, לאחר שנוצר גיליון הסגנון, הוא לחבר גיליון סגנון זה לדף שלך. יש כמה דרכים לבצע זאת כולל הקלדה ידנית של הקוד או גרירה ושחרור הקובץ ב- Markup View או ב- Design View. תרגיל נסה זאת הבא מראה לך אפשרות נוספת: שימוש בתיבת הדו-שיח Manage Styles.

נסה זאת צירוף גיליון הסגנון החדש למסמך שלך

בתרגיל זה תסיר ותצרף מחדש את גיליון הסגנון שנקרא Styles.css אל דף Default.aspx, כך שתראה חלופה נוספת לצירוף קובץ גיליון סגנון לדף ASPX. לאחר מכן תוסיף מעט טקסט וקישורים לדף זה, כדי שתוכל לראות את ההתנהגות של מערכות הכללים שיצרת קודם.

1. העבר את הדף Default.aspx ל- Markup View, והסר את האלמנט `</link>` מהקטע `<head>` שהוספת מוקדם יותר בפרק. אחר כך עבור ל- Design View, ודא שהחלון Manage Styles פתוח. אם החלון אינו פתוח לחץ במקום כלשהו בחלון Design View ובחר View ← Manage Styles מהתפריט הראשי. יופיע החלון המוצג בתרשים 9-3.



תרשים 9-3

החלון Manage Styles כולל סקירה כללית של כל גיליונות הסגנון החיצוניים והפנימיים שחלים על המסמך הנוכחי. שים לב ש-VWD רואה שהמסמך הנוכחי כבר מכיל את הסגנון style1 שיצרת בפרק הקודם.

2. לחץ על הקישור Attach Style Sheet בחלון Manage Styles, דפדף לתיקייה Styles בשורש האתר, ובחר בקובץ Styles.css. לחץ OK וגיליון הסגנון יוכנס שנית לדף שלך:

```
</style>
```

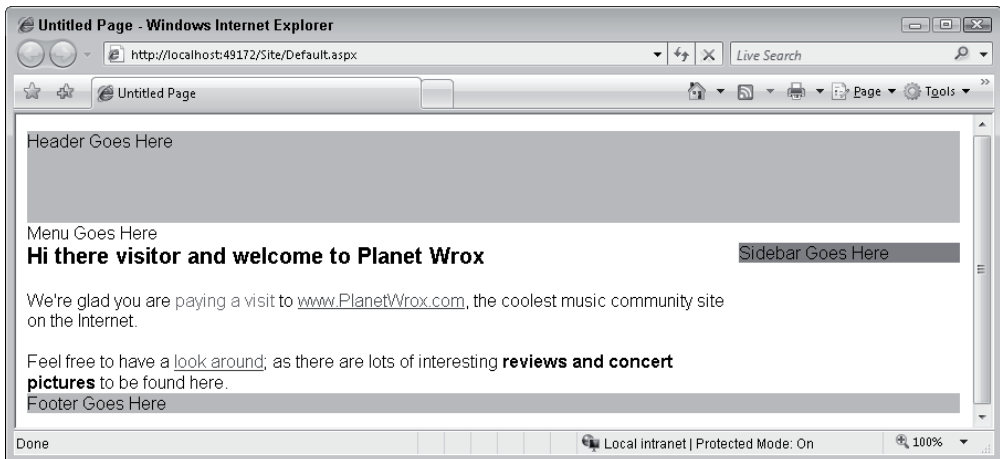
```
<link href="Styles/Styles.css" rel="stylesheet" type="text/css" />
```

```
</head>
```

3. כשאתה מספח את גיליון הסגנון, החלון Manage Styles (מוצג בתרשים 10-3) מתעדכן ומראה כעת את גיליון הסגנון המצורף החדש.
4. כשהדף Default.aspx עדיין פתוח ב- Design View, סמן את הטקסט "look around". אם הקלדת משהו אחר בתרגיל נסה זאת הקודם, סמן את הטקסט שהקלדת. לצורך הדוגמה חשוב שתסמן טקסט כלשהו שתוכל להפוך לקישור.
5. בסרגל הכלים Formatting, לחץ על הלחצן Convert to Hyperlink (זה עם הגלובוס וסמל קישור עליו). בתיבת הדו-שיח שמופיעה לחץ על הלחצן Browse, ובחר Default.aspx משורש האתר. בדרך זו הקישור מצביע אל הדף שבו הוא מוגדר (זו אינה בעיה מבחינת תרגיל זה). לחץ OK פעמיים לשחרור תיבות הדו-שיח.



תרשים 3-10



תרשים 3-11

6. שמור את השינויים של כל המסמכים הפתוחים (בחר File ← Save All מהתפריט הראשי, או הקש Ctrl+Shift+S), ואז הקש Ctrl+F5 כדי לצפות בדף Default.aspx בדפדפן. אתה אמור לראות שהדף מופיע ובו הקישור "look around" מודגש בקו תחת, כפי שמוצג בתרשים 3-11.

7. רחף עם העכבר על פני הקישור "look around"; שים לב שהוא הופך לכתום.

8. לחץ על הקישור "look around" כדי שהדף ייטען מחדש. הקישור הפך עכשיו לאדום. אם הקישור כבר היה אדום בפעם הראשונה שביקרת בו, אל דאגה. ככל הנראה כבר פתחת את הדף בדפדפן, והדבר גרם לדפדפן לסמן את הדף כאחד שכבר ביקרו בו. הדפדפן מנהל רישום של הדפים שאתה מבקר בהם, ואז הוא מחיל את הסגנון הנכון על קישורים חדשים ועל קישורים המפנים לאתרים שבהם ביקרת. ניתן לראות איך מתנהג הדפדפן למקרה שמוצג קישור לדף שטרם ביקרת בו על ידי מחיקת רשימת

הדפים. בדפדפן פתח את Internet Options על ידי בחירה ב-Tools ← Internet Options. בכרטיסייה General לחץ על Delete. בתיבת הדו-שיח שצצה לחץ על Delete History. הפעולה מנקה את כל היסטוריית הגלישה שלך. כך, כאשר תטען את הדף מחדש על ידי הקשת Ctrl+F5 בדפדפן, הקישור יהפוך לירוק.

כיצד זה פועל

החלון Manage Styles נותן סקירה מהירה של גיליונות הסגנון הפעילים עבור הדף הנוכחי, או כגיליונות סגנון חיצוניים ומצורפים, או כגיליונות סגנון מוטמעים בחלק <head> של הדף. זהו חלון שימושי מאוד לצירוף סגנונות חדשים למסמך הנוכחי, ולהזזת סגנונות ממקום אחד לאחר, דבר שתלמד לעשותו בסעיף הבא. כשאתה פותח את הדף בדפדפן, נטען גיליון הסגנון המעודכן, והדפדפן מיישם אז את הבורר #MainContent a:visited לכל הקישורים לדפים שכבר ביקרת. כשאתה חולף עם העכבר על פני קישור, מיושם הבורר #MainContent a:hover, דבר שגורם לקישור להפוך לכתום.

חרף היעילות הגבוהה של גיליונות סגנון חיצוניים, יש מצבים שבהם תרצה להשתמש בסגנונות ברמת הדף או האלמנט. יצירה וניהול של סגנונות אלה קלה באותה מידה, כפי שתראה בסעיף הבא.

יצירת גיליונות סגנון

כשאתה עובד עם דף ב- Design View, אתה צריך לעתים לבצע התאמות קטנות לחלק מהדף, כמו עיצוב קטע טקסט, יישור תמונה, או הגדרת גבול לאלמנט. בנקודה זו אתה צריך להחליט איזה סוג גיליון סגנון ליצור (חיצוני, פנימי, ברמת הדף/אלמנט). כפי שראית קודם, אתה צריך להעדיף גיליונות סגנון חיצוניים אם אתה צופה שתעשה בסגנון שימוש חוזר. ל-VWD לא אכפת איזה סגנון תיצור, היא מאפשרת ליצור סגנונות בכל שלוש הרמות. טוב מכך, היא מאפשרת לשדרג בקלות סגנון מוטמע בדף לקובץ חיצוני, או להעתיק מידע עיצוב מאלמנט למקום אחר. יכולות אלה נותנות לך גמישות רבה ואפשרות לשנות את דעתך מאוחר יותר.

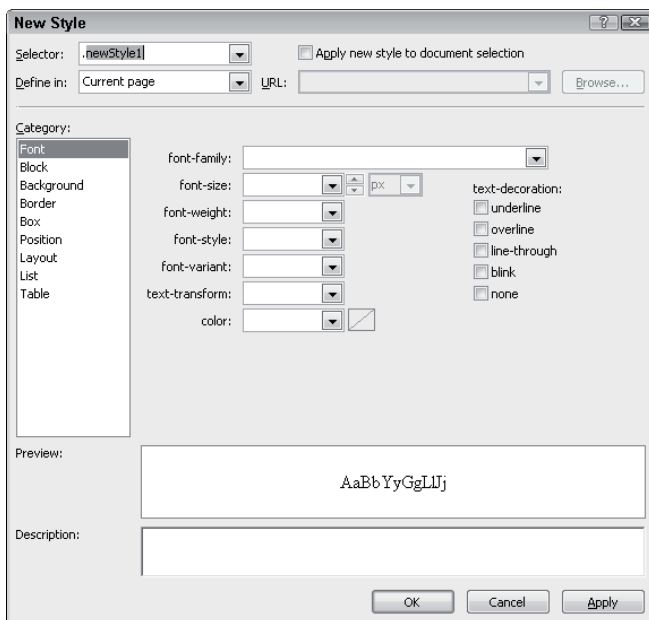
בתרגיל הבא תלמד כיצד ליצור גיליונות ברמות הדף והאלמנט. יותר מאוחר תראה איך להעביר סגנונות אלה לגיליון סגנון חיצוני, פעולה שתאפשר גם לדפים אחרים להשתמש באותם סגנונות.

נסה זאת יצירת סגנונות מוטמעים בדף וכלולים באלמנט

בתרגיל זה תוסיף כלל סגנון לאלמנט <h1> של הדף. מטרת הכלל היא לסלק את השוליים המוצגים בדפדפן סביב הכותרת כברירת מחדל. בנוסף, תעצב את הפסקה הראשונה על ידי שימוש בסגנון מחלקה כדי לתת לפסקה מראה שונה שיבליט אותה לעומת שאר הפסקאות בדף.

1. חזור ל-VWD וודא שהדף Default.aspx פתוח ב- Design View.

2. לחץ פעם אחת על האלמנט h1 כדי לבחור בו, ואז בחר Format ← New Style. תופיע תיבת הדו-שיח New Style (נראית בתרשים 3-12) שדומה למדי לתיבת הדו-שיח Modify Style שראית קודם.



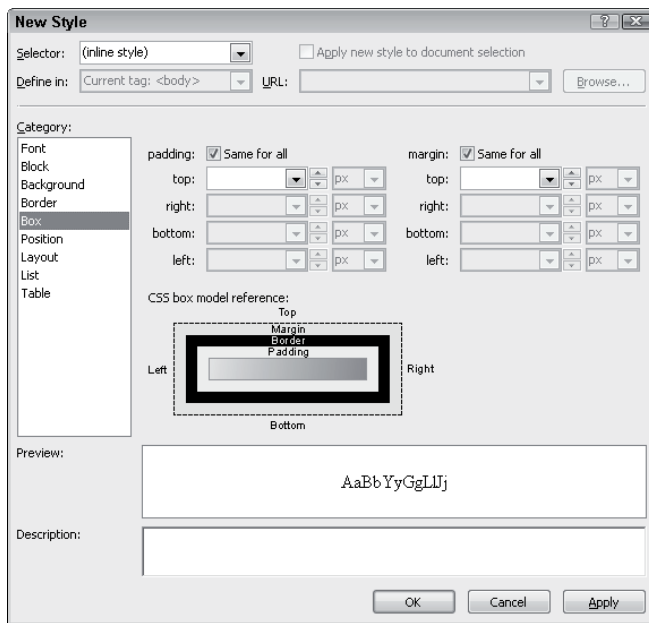
תרשים 3-12

3. בראש המסך פתח את הרשימה הנפתחת selector ובחר (inline style). זהו הפריט הראשון ברשימה. הדבר מבטיח שהסגנון החדש ייושם לאלמנט `<h1>`.

4. עבור לקטגוריה Box שמוצגת בתרשים 3-13.

תיבת דו-שיח זו כוללת תרשים שימושי שמראה לך היכן מוצגים מאפייני CSS כמו padding (מילוי), border (גבול) ו-margin (שוליים). באמצע אתה רואה מלבן כחול שמייצג את האלמנט שלך כמו `<h1>` או `<body>`. סביב מלבן זה נמצא מלבן אחר שמייצג את ה-padding. סביב מלבן שני זה מיושם ה-border. לבסוף, בחלק החיצוני של התרשים, אתה רואה את ה-margin שמיושם מחוץ לגבול של אלמנט.

כברירת מחדל משרטטים דפדפנים אזור לכן כלשהו מעל, או מתחת, לאלמנט `<h1>`. כדי לתת לכל דפדפן את אותן ההגדרות, אתה יכול להגדיר את ה-padding ל-0, ואז ליישם margin קטן בחלק התחתון של הכותרת, דבר שייצור מרחק מסוים בין הכותרת לאלמנטים שבאים אחריה. כדי להשיג זאת, הגדר את margin כ-0 בתיבה העליונה. השאר את האפשרות Same for all מסומנת ב-✓, ו-VWD תיצור עבורך הצהרה מקוצרת. לאחר מכן הסר את הסימן ✓ מ-Same for all עבור הקטע margin, הכנס 0 בתיבות top, right ו-left, ו-10 בתיבה bottom.

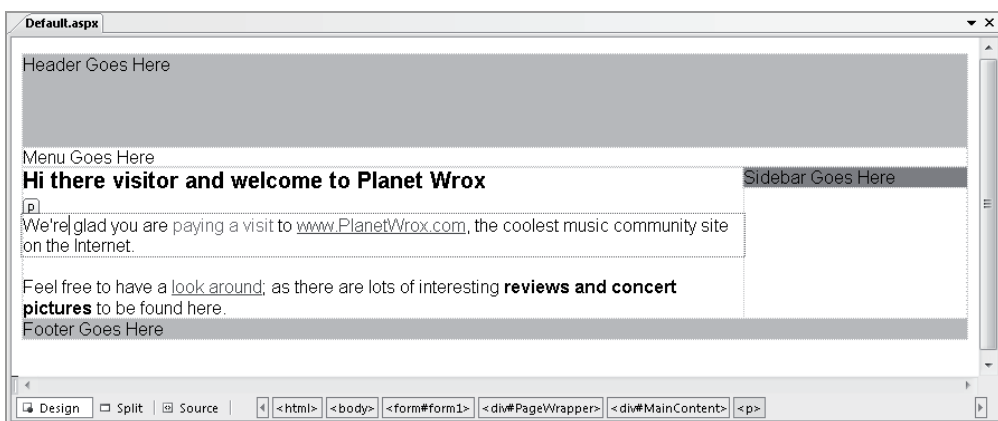


תרשים 3-13

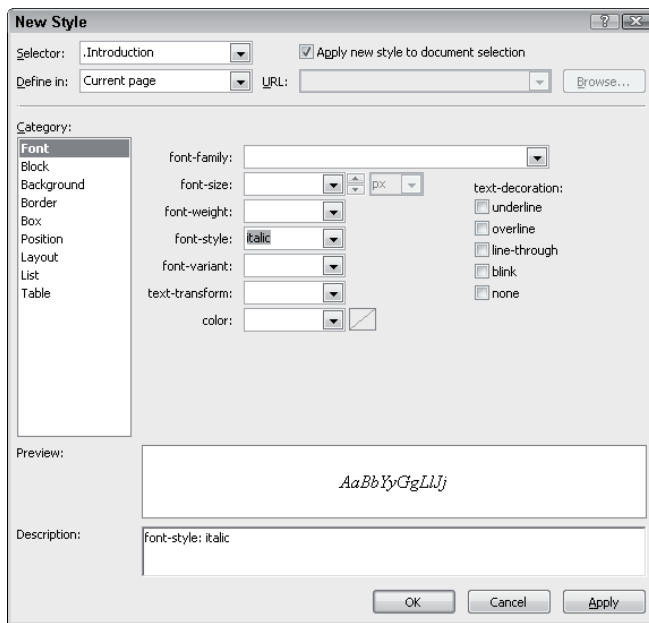
השאר את כל הרשימות הנפתחות מוגדרות ל-px ולחץ OK. ניתן לראות שתסיים ב- Markup View עם אלמנט <h1> הבא שמכיל סגנון כלול:

```
<h1 style="padding: 0px; margin: 0px 0px 10px 0px">
  Hi there visitor and welcome to Planet Wrox
</h1>
```

5. עכשיו, ב- Design View, לחץ על הפסקה הראשונה. מופיע ריבוע קטן שמראה שבחרת באלמנט <p>, כפי שמוצג בתרשים 3-14. ודא ש- Tag selector שבתחתית חלון המסמך מדגיש את האלמנט <p> ולא משהו אחר. אם אינך רואה את הריבוע הקטן, עליך לבחור Block Selection מהתפריט View ← Visual Aids של VWD.

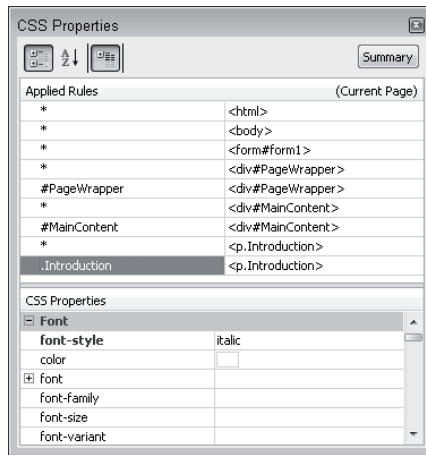


תרשים 3-14



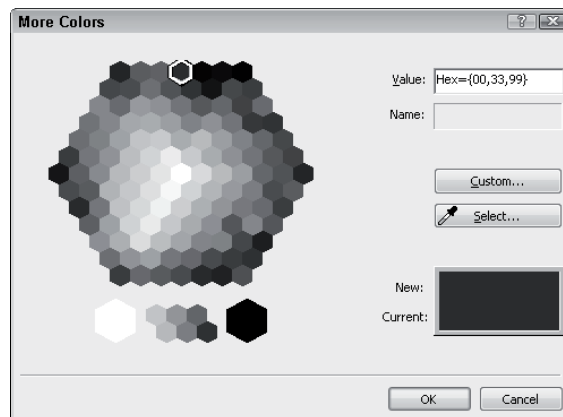
תרשים 3-15

6. כשהפסקה עדיין נבחרת, בחר Format ← New Style. הקלד הפעם את הטקסט **Introduction**. בתיבה selector שמוצגת בתרשים 3-15. אל תשכח את הנקודה (.). לפני שם הבורר.
 7. בראש המסך בחר בתיבת הסימון Apply new style to document selection. עם הגדרות אלה בתוקף, המחלקה החדשה שאתה עומד ליצור תחול על התגית <p>.
 8. מהרשימה הנפתחת font-style בחר Italic. תיבת הדו-שיח New Style שלך צריכה כעת להיראות דומה לתרשים 3-15.
 9. לבסוף, לחץ OK. שים לב שכל הפסקה מוצגת כעת בגופן נטוי.
 10. כשהתגית <p> עדיין נבחרת, פתח את תיבת הדו-שיח CSS Properties (ראה תרשים 3-16) על ידי בחירת View ← CSS Properties. תיבת הדו-שיח זו מציגה סקירה של כל מאפייני CSS, ומראה איזה מהם פעילים כרגע עבור הדף שלך.
- בחלק העליון של תיבת הדו-שיח זו מוצגת רשימת הכללים המיושמים. החלק התחתון של תיבת הדו-שיח משמש להצגת מאפייני CSS עבור כללים אלה. בתרשים 3-16 אתה רואה את הכללים ששימיים ל- Introduction selector. מאפיינים שמופיעים בגופן כחול ומודגש הם אלה שעבורם הוגדרו ערכים; המאפיינים האחרים מופיעים בגופן רגיל.



תרשים 3-16

11. ברשימה CSS Properties בחצי התחתון של המסך, אתר את המאפיין color והגדר אותו לצבע כחול עמוק, דוגמת #003399. כדי להשיג זאת, פתח את הרשימה הנפתחת עבור ערכי המאפיין, ובחר צבע מאוסף הצבעים. אם הצבע שאתה מחפש אינו זמין, לחץ על הלחצן More Colors כדי להעלות את אוסף הצבעים המורחב, המוצג בתרשים 3-17.

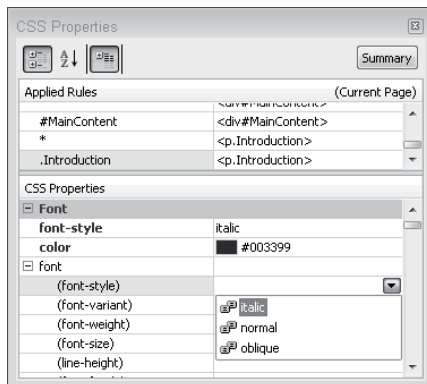


תרשים 3-17

במקום להשתמש באוסף הצבעים אתה יכול להקליד את הערך ישירות. זו הדרך שבה פועלים כל המאפיינים בטבלת המאפיינים CSS Properties: הם מאפשרים להזין באופן ישיר את הערך הרצוי, או לשנות את הערך באופן חזותי תוך שימוש בחץ או בלחצן עם אליפסות בקצה תיבת הערכים של המאפיין. תרשים 3-18 מראה את האפשרויות השונות שעומדות בפניך עבור המאפיין font-style ברשימה נפתחת נוחה.

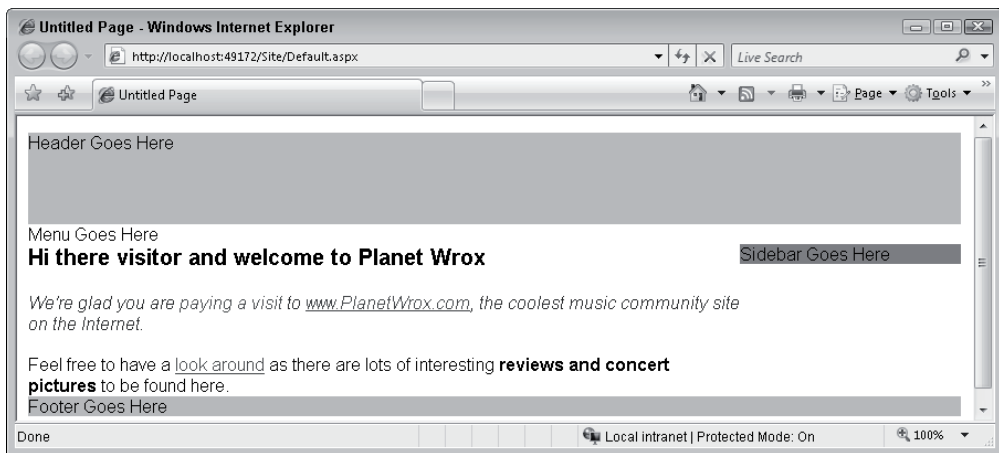
שים לב במיוחד לשלושת הלחצנים שבראש החלון שמספקים פונקציונליות שימושית. שני הלחצנים הראשונים מאפשרים לעבור מצורת הצגה לפי קטגוריות להצגה אלפביתית, כדי להקל על מציאת המאפיין הנכון. הלחצן השלישי מאפשר להציג את

המאפיינים הנבחרים בראש הרשימה (זהו המקרה בתרשים 3-18) או במקומות ברירת המחדל שלהם ברשימה.



תרשים 3-18

12. לבסוף, שמור את כל השינויים ופתח את Default.aspx בדפדפן שלך (ראה תרשים 3-19). תראה שהפסקה הראשונה מוצגת כעת בגופן כחול נטוי, פרט לקישור בטקסט אשר צריך להיות אדום.



תרשים 3-19

13. עבור חזרה ל-VWD והתבונן בדף שלך ב-Markup View. בקטע <head> של הדף אתה אמור לראות את גיליון הסגנון הבא:

```
.Introduction
{
    font-style: italic;
    color: #003399;
}

</style>
<link href="Styles/Styles.css" rel="stylesheet" type="text/css" />
</head>
```

כיצד זה פועל

הכלים הרבים שמציעה CSS הופכים את כתיבת CSS עבור האתר שלך למלאכה קלה. אינך צריך לקודד דבר, או לזכור את המאפיינים השונים שתקן CSS תומך בהם. במקום זאת, אתה יכול פשוט לבחור אותם מרשימות שונות בטבלת CSS Properties. טבלה זו מאפשרת להכניס ערכים באופן ידני, אבל מציעה גם כלים שימושיים לבחירת צבעים, קבצים ופריטים מרשימות נפתחות.

כל השינויים שאתה עושה ב- Properties Grid חלים על גיליונות הסגנון הרלוונטיים ללא תלות ברמת גיליון הסגנון: רמת אלמנט, דף פנימי או חיצוני. ה- Design View מתעדכן באותו הזמן, וכך הוא משקף את האפשרויות החדשות של CSS שהגדרת.

כשתבונן באלמנט <h1> תוכל לראות ש-VWD יצרה סגנון עם padding מוגדר כ-0px כדי להשפיע על כל ארבעת הצדדים בבת אחת, ו-margin מוגדר כ-0px 0px 10px 0px כדי לשלוט בכל אחד מהצדדים בנפרד.

ברגע שיצרת קבוצה של סגנונות שימושיים שניתנים לשימוש חוזר, אתה זקוק לדרך להחיל אותם על דפים אחרים או אלמנטים של HTML. בסעיף הבא תראה כיצד זה פועל.

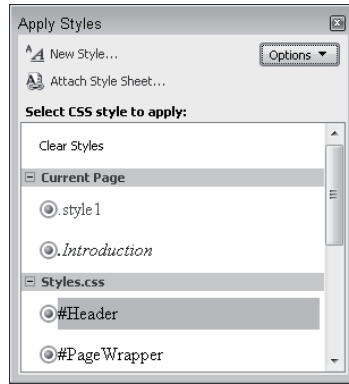
החלת סגנונות

אם אתה בעל ניסיון ב-Word של מיקרוסופט, ייתכן שאתה מורגל לתיבת הדו-שיח Styles (סגנונות), אשר מפרטת את כל הסגנונות הזמינים ומאפשרת לך להחיל אותם על חלקים נבחרים של טקסט. בדרך זו אתה יכול לקבוע בזריזות עריכה זהה לבלוקים של טקסט. הדבר פועל באופן דומה ב-VWD. עם הלווח Apply Styles - נגיש מהתפריט הראשי על ידי בחירת View ← Apply Styles - אתה יכול בקלות להחיל כללי סגנון לאלמנטים בדף.

נסה זאת יישום סגנונות

בתרגיל זה תעשה שימוש חוזר בבורר Introduction. ותחיל אותו גם על הפסקה השנייה בדף. בדרך זו שתי הפסקאות יראו אותו הדבר.

1. כשאתה ב-Default.aspx, ודא שאתה נמצא במצב Design View ובחר בפסקה השנייה בדף על ידי לחיצה עליה. ודא שה- Tag selector שבחלק התחתון של חלון המסמך מראה שנבחרה התגית <p>, ולא תגית אחרת כלשהי דוגמת שעשויה להוות חלק מהאלמנט <p>. אם יש לך רק פסקה אחת של טקסט, צור תחילה פסקה נוספת (ב- Design View הקש Enter אחרי הפסקה הראשונה), הקלד טקסט ובחר בפסקה החדשה.
2. פתח את תיבת הדו-שיח Apply Styles על ידי בחירת View ← Apply Styles. חלון זה מראה את כל ה- selectors שנמצאו בדף הנוכחי ובאיזושהו גיליון סגנון שמסופח אליו. אם אינך רואה את כל הסגנונות שמוצגים בתרשים 20-3, לחץ על הלחצן Options ובחר Show All Styles.



תרשים 20-3

VWD משתמשת ברמזים חזותיים כדי לעזור לך למצוא את הסגנון הנכון. ראשית, תיבת הדו-שיח Apply Styles משתמשת בנקודות אדומות, ירוקות וצהובות כדי לייצג בוררים מסוגי ID, Class וסגנונות הכלולים באלמנטים, בהתאמה. תרשים 20-3 מראה נקודות אדומות וירוקות בלבד (עליך להאמין לי), מפני שאין שום סגנונות פרטיים שיישמתי על האלמנט `<p>`. עם זאת, אם תבחר באלמנט `<h1>`, יופיע הסגנון Introduction. אם אכן תנסה זאת, ודא אחר כך שתבחר שוב באלמנט `<p>`. יתר על כן, סגנונות שנמצאים כעת בשימוש בדף מוקפים בעיגול נוסף, כפי שקורה עם כל הבוררים בתרשים 20-3.

3. לחץ על המחלקה Introduction בתיבת הדו-שיח כדי ש-VWD תוסיף מאפיין class לתגית `<p>`:

```
<p class="Introduction">
    Feel free to have a <a href="Default.aspx">look around</a>; as there are lots of
    interesting <b>reviews and concert pictures</b> to be found here.
</p>
```

אם ברצונך להחיל מחלקות רבות, לחץ על המקש Ctrl בעורך לוחץ על אחת מהמחלקות האחרות ברשימה. הדבר גורם להחלת רשימה של מחלקות מופרדות ברווח על המאפיין class של האלמנט. אתה יכול לעקוב אחר אותם צעדים כדי להחיל את הסגנונות הנבחרים ב-Markup View.

4. על ידי שימוש בלחצן Clear Styles אתה יכול להסיר מחלקות קיימות וסגנונות כלולים מתוך התגית. היזכר בקטע HTML שראית בפרק הקודם כשהשתמשת בסרגל הכלים Formatting כדי לערוך טקסט בדף. אם השתמשת בלחצן Foreground Color, סיימת בוודאי עם קוד דומה לזה:

```
We're glad you are <span class="style1">paying a visit</span>
```

כדי לסלק את המאפיין class, לחץ על התגית `<span>` ב-Markup View ואז לחץ Clear Styles בתיבת הדו-שיח Apply Styles (נראית בראש תרשים 20-3). כעת תסיים עם קוד HTML הבא:

```
We're glad you are paying a visit
```


VWD תסלק את התגית מכיוון שלתגית ריקה סביב הטקסט אין שימוש.
באותו אופן תוכל להסיר גם מאפייני סגנון מאלמנטים של HTML.

כיצד זה פועל

גם במקרה זה, VWD מסוגלת לעדכן את כל החלונות הרלוונטיים בו-זמנית: Design View, Markup View וכלי העיצוב השונים של CSS. כשאתה מחיל מחלקה מהחלון Apply Styles, מוסיפה VWD את המחלקה המבוקשת לאלמנט HTML הנבחר ב-Markup View. היא גם מעדכנת אז את חלון Design View. באופן דומה, כשאתה מסיר selector או הצהרה מסגנון ב-Design View, מתעדכנים חלונות Markup View ו-CSS Tools.

הפונקציונליות האחרונה של CSS שאתה צריך להתבונן בה בפרק זה נמצאת בחלונות Manage Styles ו-Apply Styles. מלבד עזרה בצירוף קבצי CSS למסמכים שלך, חלונות אלה מאפשרים לך גם ניהול קל של הסגנונות שלך.

ניהול סגנונות

מכיוון שקל מאוד להוסיף סגנונות חדשים, הדף שלך עלול להיות "מבולגן" תוך זמן קצר. כדי להשיג אפשרות לשימוש חוזר, עליך להעביר רבים מהסגנונות הפנימיים שלך לגיליון סגנון חיצוני. זו בדיוק הפעולה שהחלונות Manage Styles ו-Apply Styles מאפשרים לך לעשות.

נסה זאת ניהול סגנונות

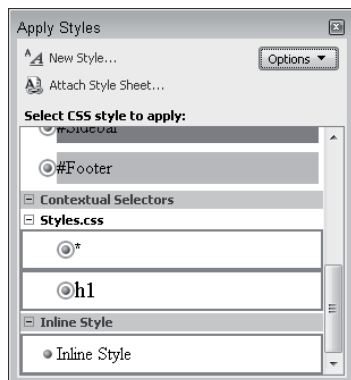
מוקדם יותר בפרק זה שיניית את האלמנט `<h1>` והחלת על הכותרת padding ו-margins. עם זאת, Default.aspx אינו הדף היחיד שיכול ליהנות מסגנון זה, ולכן הגיוני להעביר את הסגנון לקובץ Styles.css. באופן דומה, גם המחלקה Introduction נראית שימושית במידה כזו שראוי לכלול אותה ב-Styles.css כדי שגם דפים אחרים יוכלו לגשת אליה. תרגיל זה מראה לך איך להעביר סגנונות באתר.

1. ודא ש-Default.aspx עדיין פתוח, עבור ל-Markup View.
2. אתר את התגית `<h1>` ולחץ עליה פעם אחת. VWD הופכת את התגית למודגשת כדי לציין שהיא התגית האקטיבית, כפי שמוצג בתרשים 3-21.

```
17 <div id="PageWrapper">
18   <div id="Header">Header Goes Here</div>
19   <div id="MenuWrapper">Menu Goes Here</div>
20   <div id="MainContent">
21     <h1 style="padding: 0px; margin: 0px 0px 10px 0px">Hi there visitor and we
22     <p class="Introduction">We're glad you are paying a visit to <a href="
23     <p class="Introduction">Feel free to have a <a href="Default.aspx">look ar
24   </div>
25   <div id="Sidebar">Sidebar Goes Here</div>
26   <div id="Footer">Footer Goes Here</div>
```

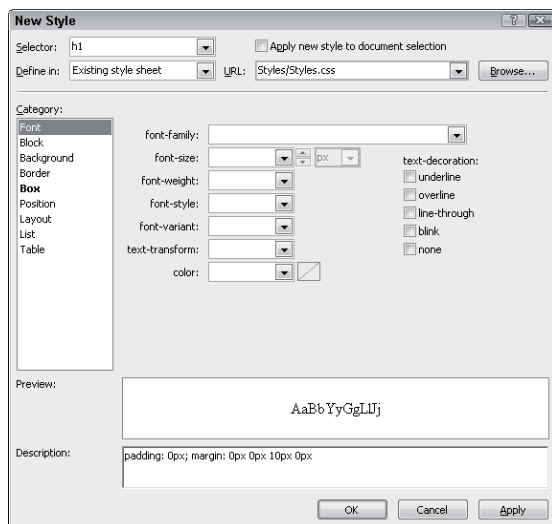
תרשים 3-21

3. פתח את החלון Apply Styles על ידי בחירת View ← Apply Styles מהתפריט הראשי. לחילופין, אם החלון הוא חלק מקבוצת חלונות, בחר בכרטיסייה שלו כדי להפוך אותו לאקטיבי. בחלק התחתון של החלון Apply Styles יופיע inline style (ראה תרשים 3-22).



תרשים 3-22

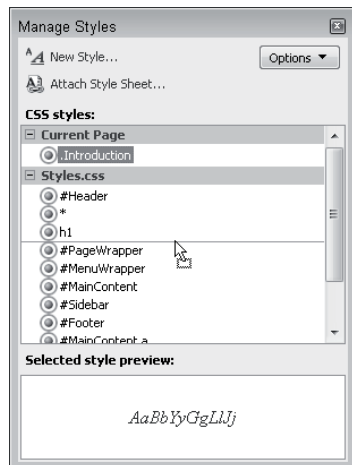
4. לחץ לחיצה ימנית על 'Inline Style' ובחר 'New Style Copy'. כעת מוצגת תיבת הדו-שיח 'New Style', שמאפשרת ליצור סגנון חדש שמבוסס על המבחר הנוכחי. בראש החלון בחר 'h1' מהרשימה הנפתחת 'selector'. מהרשימה הנפתחת 'Define' בחר 'Existing style sheet'. מהרשימה הנפתחת 'URL' בחר 'Styles/Styles.css'. אם פריט זה אינו זמין, לחץ על הלחצן 'Browse' כדי לאתר ולבחור בו. תיבת הדו-שיח שלך צריכה להיראות כמו תרשים 3-23.



תרשים 3-23

5. לחץ OK לשחרור תיבת הדו-שיח. VWD יוצרת עותק של הסגנון 'h1' ומציבה אותו בקובץ 'Styles.css'. שים לב שעבור 'h1' בקובץ 'Styles.css' נוצר בורר חדש, במקום להוסיף את המידע הנוגע ל-'padding' ול-'margin' אל הבורר הקיים. אם רצונך בכך, תוכל לשלב את שני הבוררים לאחד באופן ידני.

6. בתיבת הדו-שיח Apply Styles לחץ שוב לחיצה ימנית על Inline Style, והפעם בחר מתיבת ההקשר ב- Remove Inline Style. פעולה זו מסלקת את המאפיין style מהאלמנט h1.
7. סגור את תיבת הדו-שיח Apply Styles ופתח את החלון Manage Styles. תחת הפריט Current Page אתר את הבורר Introduction. הנראה בתרשים 3-24.



תרשים 3-24

8. לחץ על הבורר **Introduction**. פעם אחת, ואז גרור אותו לאזור שתחת Styles.css ושחרר אותו, למשל, אחרי הבורר h1. תרשים 3-24 מראה כיצד Introduction selector נגרר מ- Current Page לתוך Styles.css, ומשוחרר בין הבוררים h1 ו- #PageWrapper.
9. ברגע ששחררת את הבורר באזור Styles.css שבחלון Manage Styles, מוסר הסגנון המשוחרר מהדף הנוכחי שלך, ומוכנס ל- Styles.css. מכיוון שקובץ CSS זה נכלל בדף הנוכחי שלך באמצעות האלמנט <link>, לא תבחין בהבדל ב- Design View. אתה יכול לסלק את התגית <style> הריקה מ- Default.aspx, מכיוון שלא תזדקק לה יותר.
10. שמור את כל השינויים התלויים ועומדים שעשויים להיות לך, ואז פתח את הדף Default.aspx בדפדפן על ידי הקשת Ctrl+F5. שים לב שהפסקאות לא השתנו ועדיין משתמשות באותו גופן כחול ונטוי.

כיצד זה פועל

לרוע המזל, VWD אינה מאפשרת להעביר סגנונות פנימיים אל קבצי גיליונות סגנון חיצוניים. עם זאת, על ידי יצירת עותק של הסגנון הנוכחי ומחיקת הסגנון הפנימי המקורי, אתה יכול להשיג את אותה התוצאה. העברת גיליונות סגנון בין קבצים, קלה הרבה יותר. אתה גורר סגנון מקובץ אחד אל הקובץ האחר, ו- VWD מעבירה את הקוד באופן אוטומטי. זה הופך את פעולת ארגון ה- CSS שלך לקלה ביותר. במקום להשאיר בדף את כל קוד CSS המוטמע שלך מפני שאתה חושש לגעת בו, אתה יכול עכשיו לגרור ולשחרר אותו לתוך קובץ חיצוני. הדבר מאפשר שימוש חוזר קל באותם סגנונות גם בדפים אחרים, מקטין את גודל הדף ונפחו, ועושה

את האתר שלך קל לניהול. מובן מאליו וגם חשוב שהקובץ שאתה מעביר אליו את קוד CSS שלך יהיה מצורף לדפים שאתה עובד אתם.

עצות מעשיות לעבודה עם CSS

כדי להפיק את המרב מ-CSS, עליך לעקוב אחר העצות הבאות:

- ⊙ הקדש את הזמן הדרוש להכרת המאפיינים הרבים שנתמכים על ידי CSS. הדרך הטובה ביותר לעשות זאת היא ליצור דף חדש לחלוטין בתיקייה Demos, ליצור כמה אלמנטים של HTML, כמו התגיות `<div>` ו-`<p>`, ואז לערוך ניסויים עם כל המאפיינים השונים. התנסות עם מאפיינים רבים מהטבלה CSS Properties תיתן לך תחושה של האפשרויות הזמינות. הדבר יקל עליך מאוחר יותר כשתרצה להחיל עיצוב ועריכה מסוימים על טקסט.
- ⊙ כשתיצור מחלקות CSS מותאמות אישית נסה לקבוע להן שמות שיתארו את התנהגות הכלל, ולא את המראה והתחושה. לדוגמה, מחלקה בשם Introduction. שתפקידה לסגן את הפסקה הראשונה בדף היא תיאור טוב. הדבר מאפשר לשנות את הערכים שאינם נראים מבלי להשפיע על המשמעות הממשית של השם. לעומת זאת, מובטח לך שמחלקות בעלות שם כמו BlueAndItalic. תגרומנה לך בעיות מאוחר יותר. מה יקרה אם תחליט לשנות את הכחול לשחור? אחת מהשתיים: או שתסיים עם שם מחלקה מוזר ביותר שאינו מתאר את התנהגותה, או שתצטרך לשנות את שם המחלקה ואז לעדכן את כל האתר, שמשמעותו - שינוי שם המחלקה מ-BlueAndItalic לשם החדש BlackAndItalic.
- ⊙ נסה ליצור מערכות כללים קטנות וניתנות לשימוש חוזר שתוכל לשלב כפי הצורך, ולא ליצור כללים גדולים וכלליים שאפשר להשתמש בהם רק באלמנט UI יחיד. לדוגמה, במקום ליצור סגנון כמו זה:

```
. ImportantHeading
{
    font-size: 20px;
    font-weight: bold;
    color: red;
}
```

מוטב שתיצור כמה סגנונות פשוטים, שקל לעשות בהם שימוש חוזר:

```
h1
{
    font-size: 20px;
}
```

```

.Attention
{
    font-weight: bold;
    color: red;
}

```

כשתיישם את המחלקה Attention, עבור כותרת: `<h1 class="Attention">`, תקבל בדיוק את אותה התנהגות שקיבלת כשיישמת את המחלקה ImportantHeading. עם זאת, עם המחלקה הנפרדת Attention יצרת כלל שניתן לשימוש חוזר, שתוכל להחיל על אלמנטים אחרים שדורשים את תשומת-לב המשתמש, כמו התגיות `<p>` או `<span>`.

סיכום

פרק זה נתן לך מבט טוב על CSS, השפה החשובה ביותר כדי לסגן דפי HTML ו-ASPX באינטרנט.

CSS מאפשרת להתגבר על מגבלות HTML ביחס לקביעת סגנון דפי האינטרנט שלך, מכיוון שהיא מתוכננת:

- ⊙ להקטין את נפח הדף ככל שניתן.
- ⊙ לאפשר לך שליטה טובה יותר על המראה של הדפים.
- ⊙ לסייע לך ליצור אתרי אינטרנט שייטענו מהר יותר ויהיו קלים יותר לתחזוקה.

CSS קלה ללימוד באופן יחסי, הודות למערכת כללי הדקדוק הפשוטה שלה. לפני שתתחיל להשתמש במלוא הפוטנציאל של CSS עליך להכיר את המונחים החשובים של השפה. ראשית עליך להבין מהו כלל. כלל (rule) זו צירוף ושילוב של בורר והצהרה אחת או יותר, שמגדירות את המראה של אלמנטים בדף. בורר (selector) מצביע על אלמנט אחד או יותר בדף שלך, והצהרה קובעת איך ייראה האלמנט. הצהרה (declaration) ניתן לפצל לשני חלקים: מאפיין וערך. מאפיין (property) קובע איזה חלק מהאלמנט שעליו הצביע הבורר צריך לסגן, ערך (value) קובע איך האלמנט המסוגן יוצג.

כשתהיה לך הבנה טובה של מונחי CSS תמצא שקל יותר לעבוד עם הכלים הרבים של CSS שמציעה לך VWD. כלים כמו החלונות Manage Styles ו-Apply Styles, ה-Style Builder וה-IntelliSense המתוחכמים הופכים את כתיבת CSS לקלה במיוחד. במקום לזכור כל פרט קטן אודות CSS, אתה יכול לסמוך על הכלים המובנים שיכתבו עבורך קוד מתאים.

בסוף הפרק ראית כיצד להשתמש בחלון Manage Styles כדי להעביר כללים מדף ASPX אל גיליון סגנון חיצוני. הדבר שימושי ביותר כאשר ברצונך לשמור את הדפים שלך נקיים ומסודרים. אתה יכול להתחיל להוסיף גיליונות סגנון מוטמעים באופן ישיר למסמך. כשתסיים לטפל במסמך תוכל להעביר את כל הכללים, או חלקם, אל קבצי סגנון חיצוניים, כך שכל הדפים באתר שלך יוכלו להשתמש בהם. תיבת דו-שיח זו שימושית גם אם ברצונך לסדר מחדש את הבוררים השונים בקובץ CSS יחיד: גרור ושחרר אותם במקום החדש.

פרק זה הניח תשתית טובה לעבודה עם CSS ב- VWD 2008. בשארית הספר תשתמש בידע זה כדי ליצור דפי אינטרנט שנראים טוב ומאורגנים היטב.

אפשר ליישם את CSS לא רק עבור HTML, כפי שראית בפרק זה, אלא גם עבור פקדי שרת של ASP.NET. קוד CSS שאתה מיישם לפקדים אלה יופיע לבסוף בדפדפן כ-HTML פשוט שלגביו ישימים אותם עקרונות שראית בפרק זה. הפרק הבא נותן לך מבט מפורט על מגוון פקדי השרת של ASP.NET.

תרגילים

1. מהי התועלת העיקרית משימוש בסגנון חיצוני לעומת שימוש בגיליונות סגנון מוטמעים בדף?
 2. כתוב כלל CSS שמשנה את המראה של כל הכותרות מרמה אחד (h1) שנמצאות בדף שלך. לפי הדרישות הבאות:
 - ⊙ הכותרת משתמשת בגופן Arial.
 - ⊙ צבע הכותרת צריך להיות כחול.
 - ⊙ גודל הגופן של הכותרת חייב להיות 18 פיקסל.
 - ⊙ מצדה העליון ומצדה השמאלי של הכותרת יהיה גבול (border) דק בצבע כחול.
- עבור הדרישה האחרונה: בדוק את הרשימה IntelliSense של VWD בקובץ CSS כדי לגלות גרסת קיצור אחרת למאפיין border.
3. איזה משני הכללים הבאים קל יותר לשימוש חוזר בדפים באתר שלך. האם תוכל להסביר מדוע?

```
#MainContent
{
    border: 1px solid blue;
}

.BoxWithBorders
{
    border: 1px solid blue;
}
```

4. VWD מאפשרת לצרף גיליון סגנון חיצוני אל הדף בכמה דרכים שונות. נקוב בשלוש האפשרויות השונות לבצע זאת?

פרק 4

עבודה עם פקדי ASP.NET

פקדי שרת (Server Controls) הם סוסי העבודה של ASP.NET. כמעט כל הדפים שתבנה ב-VWD יכילו פקד שרת אחד או יותר. ישנם פקדים פשוטים כמו Button ו-Label, וישנם פקדים מורכבים שמסוגלים להציג נתונים מבסיס נתונים, כמו TreeView ו-GridView, אשר מוצגים בפרקים 7 ו-12.

הארכיטקטורה של פקדי שרת בסביבת ASP.NET משולבת מאוד בזו של ASP.NET, ונותנת לפקדים קבוצת תכונות ייחודית בטכנולוגיות העכשוויות לבניית אתרי אינטרנט.

פרק זה מסביר מהם פקדי שרת, כיצד הם פועלים, ואיזה מהם זמינים באופן מיידי כחלק מ-Visual Web Developer.

בפרק זה תלמד בהרחבה על הנושאים הבאים:

- ⊙ מהם פקדי שרת בסביבת ASP.NET.
- ⊙ הסוגים השונים של פקדי שרת שעומדים לשימושך.
- ⊙ אופן ההתנהגות המשותף למרבית הפקדים.
- ⊙ איך מעבד רכיב ASP.NET runtime את פקדי השרת שבדרך שלך.
- ⊙ איך יכולים פקדי שרת לשמר את מצבם בין העברת בקשה לשרת וחזרה מהשרת לדפדפן (postbacks).

הפרק מתחיל בדיון כללי על פקדי שרת. תלמד איך להגדיר אותם בקוד על ידי גרירתם מארגז הכלים אל Design View או אל Markup View.

הסעיף הבא מציג מבט מעמיק על הפקדים הרבים הזמינים בארגז הכלים של VWD.

מבוא לפקדי שרת

חשוב להבין כיצד פועלים פקדי שרת, ואת השוני הגדול בינם לבין הגדרת הפקדים בשפות אחרות כמו ASP או PHP, שהינה שפת תכנות פופולרית אחרת ליצירת אתרי אינטרנט דינמיים.

לדוגמה, בשפות אלו, כדי להשפיע על הטקסט בתיבת טקסט, עליך להשתמש ב-HTML פשוט, ולערבב אותו עם קוד צד-השרת. הדבר פועל באופן דומה לדוגמה בפרק 2, שבה הוצגו בדף

התאריך והשעה הנוכחיים. ב-ASP הקלאסית, למשל, עליך להשתמש בקוד הבא כדי ליצור תיבת טקסט עם הודעה שכוללת בתוכה את הזמן הנוכחי:

```
<input type="text" value="Hello World, the time is now <%=Time()%>" />
```

כפי שניתן לראות, קוד זה מכיל HTML פשוט, מעורבב עם בלוק של צד-השרת, אשר תחום על ידי $\%< - \%>$, שמפיק את הזמן הנוכחי תוך שימוש בסימן השוויון (=). המגרעת העיקרית בסוג זה של קוד היא בכך שהערבוב של HTML וקוד צד-השרת מקשה על כתיבה וניהול של הדפים שלך. למרות שדוגמה זו היא שטחית, מכיוון שעדיין קל להבין את הקוד, הרי שסוג זה של תכנות יכול להידרדר במהירות לדפים מאוד מורכבים בעלי קטעי קוד שונים אשר משולבים ביניהם.

פקדי שרת פועלים באופן שונה. ב-ASP.NET הפקדים "חיים" בשרת בתוך דף ASPX. כשהדף מבוקש על ידי הדפדפן, פקדי צד השרת מעובדים על ידי רכיב ASP.NET runtime - המנוע שאחראי לקבלה ועיבוד של בקשות לדפי ASPX. כתוצאה מהעיבוד פולטים הפקדים קוד HTML אשר מצורף לפלט הסופי של הדף. בסופו של דבר, קוד HTML זה מסיים את דרכו בדפדפן, שבו הוא משמש לבניית הדף.

כך, במקום להגדיר פקדי HTML ישירות בדפים שלך, אתה מגדיר פקד שרת של ASP.NET עם התחביר הבא, כשהקטעים בגופן נטוי שונים בכל פקד.

```
<asp:TypeOfControl ID="ControlName" Runat="Server" />
```

ControlName = סוג הפקד. *TypeOfControl* = שם מזהה הפקד.

לדוגמה, כדי ליצור תיבת טקסט שתכיל את אותה הודעת קבלת פנים ואת הזמן הנוכחי, תוכל להשתמש בתחביר הבא:

```
<asp:TextBox ID="txtMessage" Runat="Server" />
```

שים לב שלפקד יש שני מאפיינים: ID ו-Runat. המאפיין ID משמש לזיהוי ייחודי של פקד בדף, כדי שתוכל להתייחס אליו בעת התכנות בצד השרת. יש חשיבות לכך שכל פקד בדף יהיה בעל ID ייחודי, שאם לא כן, ASP.NET runtime לא יבין לאיזה פקד אתה מתכוון. אם במקרה תקליד פעמיים את אותו ID, תתקבל הודעת שגיאה מ-VWD. המאפיין ההכרחי Runat משמש לציון העובדה שמדובר בפקד שנמצא בשרת. ללא ציון מאפיין זה הפקדים לא יעובדו, אלא יוצגו במקור בתבנית HTML שלהם. אם אי-פעם תרגיש שחסר לך פקד בפלט הסופי ב-HTML של הדפדפן, ודא שהוא כולל את המאפיין הנדרש הזה. שים לב שעבור אלמנטים שאינם של שרת, דוגמת אלמנטים של HTML פשוט, המאפיין Runat הוא רשות. אם מאפיין זה יופיע בפקדים שאינם פקדי שרת, הקוד שלך (בשרת) יוכל להגיע אליהם. בהמשך הספר תלמד יותר על נושא זה.

באפשרותך לתכנת כנגד תיבת טקסט זו מתוך קוד שכלול בדף, או מקוד שנמצא בקובץ קוד ברקע נפרד, כפי שראית בפרק 2. כדי להכין את הודעת קבלת הפנים והזמן תוכל להשתמש בקוד הבא:

VB.NET

```
txtMessage.Text = "Hello World, the time is now " & DateTime.Now.TimeOfDay.ToString()
```

C#

```
txtMessage.Text = "Hello World, the time is now " + DateTime.Now.TimeOfDay.ToString();
```

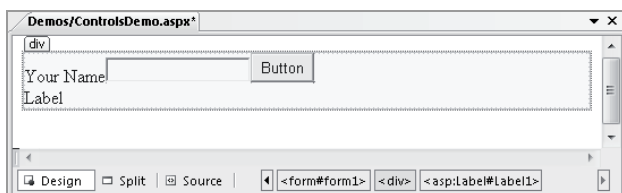
כפי שניתן לראות, הגדרת הפקד על גבי הדף מופרדת כעת מהקוד שמגדיר את טקסט הפקד. עובדה זו מקלה על ההגדרה והתכנות של תיבת הטקסט (או כל פקד אחר) מכיוון שהיא מאפשרת להתמקד בכל פעם במשימה אחת בלבד: או בהצגת הפקד והופעתו החזותית, או בתכנות התנהגותו (מבlook של קוד).

בתרגיל הבא תראה כיצד פקדי שרת שולחים את ייצוג ה-HTML שלהם אל הלקוח.

נסה זאת עבודה עם פקדי שרת

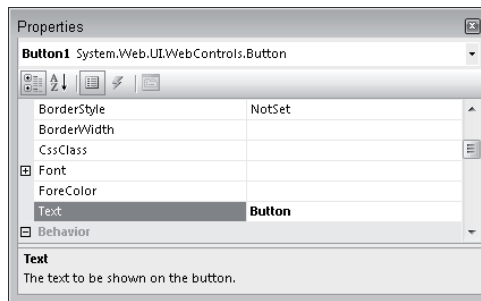
בתרגיל זה תוסיף לדף את הפקדים האלה: תיבת טקסט (**TextBox**), תווית (**Label**) ולחצן (**Button**). כשתבצע בקשה של הדף מהשרת על ידי הדפדפן, תשתנה צורת פקדי השרת ל-HTML אשר יישלח ללקוח (לדפדפן). כשתבונן בדפדפן בקוד HTML הסופי עבור הדף, תראה שקוד זה שונה לחלוטין מסימני העריכה הראשוניים ב-ASP.NET.

1. פתח את הפרויקט Planet Wrox ב- Visual Web Developer.
2. בתיקיה Demos שבסייר הפתרון צור דף חדש בשם ControlsDemo.aspx. ודא שהדף משתמש באפשרות קוד ברקע (Code Behind), ושהרשימה הנפתחת של השפה מכילה את שפת התכנות שלך.
3. עבור ל- Design View. גרור TextBox, Button ו-Label מארגז הכלים אל משטח העיצוב שבתחום הקווים המקווקווים של התגית <div> שנוספה עבורך כשיצרת את הדף. הקלד את הטקסט **Your Name** לפני ה-TextBox והוסף סיום שורה בין Button לבין Label על ידי הצבת הסמן בין שני הפקדים ב- Design View והקשת Enter. אם אתה נתקל בקושי להציב את הסמן בין הפקדים, מקם אותו לאחר הפקד Label ואז לחץ פעמיים על מקש החץ השמאלי. כתוצאה מהלחיצה הראשונה ייבחר Label; לאחר הלחיצה השנייה ימוקם הסמן בין שני הפקדים, ויאפשר לך להקיש Enter. תצוגת Design View צריכה להיראות עכשיו דומה לתרשים 4-1.



תרשים 4-1

4. לחץ לחיצה ימנית על Button ובחר Properties כדי לפתוח את טבלת המאפיינים עבור הפקד. הקשת F4 לאחר בחירת Button תביא לאותה תוצאה. החלון שמופיע, מוצג בתרשים 4-2, מאפשר לשנות את המאפיינים של הפקד, אשר בתורם משפיעים על דרך ההתנהגות של הפקד בזמן הריצה.
5. הקלד: Submit Information עבור המאפיין Text של הפקד ושנה את ה-ID שלו ל-btnSubmit (את ה-ID תמצא הרחק בתחתית הרשימה).
6. שנה את ה-ID של TextBox ל-txtName תוך שימוש בטבלת המאפיינים.
7. נקה את המאפיין Text של Label תוך שימוש בטבלת המאפיינים. אתה יכול להשאיר את ה-ID שלו מוגדר כ-Label1.



תרשים 4-2

8. כשאתה ב- Design View, לחץ לחיצה כפולה על הלחצן כדי לגרום ל-VWD להוסיף קוד אשר יופעל כשילחצו על הלחצן בדפדפן. הוסף את שורת הקוד המוצגת לבלוק הקוד ש-VWD הכניסה עבורך:

VB.NET

```
Protected Sub btnSubmit_Click(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles btnSubmit.Click
    Label1.Text = "Your name is " & txtName.Text
End Sub
```

C#

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
    Label1.Text = "Your name is " + txtName.Text;
}
```

הערה: שים לב שדוגמת קוד VB.NET משתמשת בקו תחתון (.) כדי לפצל את הקוד לשתי שורות. הדבר נעשה אך ורק כדי להציג את הקוד במגבלות תחומי אורך השורה בספר. בדף web שלך תראה את כל הקוד בשורה אחת.

9. שמור את השינויים שביצעת בדף, ואז פתח אותו בדפדפן על ידי הקשת Ctrl+F5. כשהדף יופיע בדפדפן אל תלחץ על הלחצן, אלא פתח את המקור של הדף על ידי לחיצה ימנית על הדף בדפדפן ובחירת View Source או View Page Source. אתה אמור לראות את קוד HTML הבא:

```
...
<div>
  Your Name<input name="txtName" type="text" id="txtName" />
  <input type="submit" name="btnSubmit" value="Submit Information" id="btnSubmit" />
  <br />
  <span id="Label1"></span>
</div>
```

10. עבור חזרה לדפדפן, הקלד את שמך בתיבת הטקסט, ואז לחץ על הלחצן. פתח שוב את מקור הדף בדפדפן תוך שימוש בתפריט לחיצה ימנית של הדפדפן. הקוד אמור להיראות עכשיו כך:

```
<div>
  Your Name<input name="txtName" type="text" value="Krudo" id="txtName" />
  <input type="submit" name="btnSubmit" value="Submit Information" id="btnSubmit" />
  <br />
  <span id="Label1">Your name is Krudo</span>
</div>
```

שים לב ששתי השורות המוצללות השתנו, ומראות כעת את השם שהכנסת בתיבת הטקסט.

כיצד זה פועל

כפי ששמו מעיד עליו, פקד שרת של ASP.NET נמצא בשרת בדף ASPX שלך, שבו הוא מעובד על ידי ASP.NET runtime. כשאתה מבקש דף בדפדפן, ה-ASP.NET runtime יוצר בזיכרון ייצוג של קובץ ASPX יחד עם הפקדים שיצרת. כאשר רכיב runtime עומד לשלוח את קוד HTML לדפדפן, הוא מבקש מכל אחד מהפקדים בדף את קוד HTML שלו ומכניס אותו לתוך התוצר הסופי. לדוגמה, כשהפקד Label מתבקש לתת את קוד HTML שלו בפעם הראשונה שהוא נטען, הוא יחזיר:

```
<span id="Label1"></span>
```

כפי שניתן לראות, למרות שהגדרת את הפקד Label עם התחביר <asp:Label>, הוא מופיע בדפדפן כתגית פשוטה. מכיוון שהמאפיין Text של הפקד Label ריק, אינך רואה טקסט כלשהו בין שתי התגיות . אותו הדבר חל על פקדים אחרים: <asp:TextBox> מופיע כ- <input type="text">, בעוד ש- <asp:Button> מופיע כ- <input type="submit">.

כשאתה לוחץ על הלחצן, הפקד גורם ל-**postback**, פעולת שליחה של המידע הנוגע לפקדים שבדף אל השרת, שבו הדף נטען שוב. בנוסף, מבוצע הקוד שכתבת לטיפול באירוע הלחיצה של הלחצן. קוד זה לוקח את השם שהקלדת בתיבת הטקסט ומקצה אותו לפקד Label:

```
Label1.Text = "Your name is " + txtName.Text;
```

אל תדאג בעניין הבנת תחביר הקוד שמטפל באירוע הלחיצה של הלחצן. בפרק 5 תראה כיצד הוא פועל ומדוע אתה זקוק לו.

בשלב זה הפקד Label מכיל את הטקסט שהקלדת בתיבת הטקסט. כך, כשהפקד מתבקש לשלוח את קוד HTML שלו, הוא מחזיר:

```
<span id="Label1">Your name is Krudo</span>
```

בהמשך הלימוד בפרק זה, כאשר נדון במנוע המצב (state engine) של ASP.NET, תבין טוב יותר את פעולת ה-postbacks.

מבט מקרוב על שרתי פקד של ASP.NET

במרבית הזמן שתקדיש לבניית דפי האינטרנט שלך תעסוק בפקדי שרת, ולפיכך עליך ללמוד כיצד הם פועלים וכיצד להשתמש בהם. בסעיף הבא תלמד איך להוסיף פקדים לדפים, ואיך לשנות את דרך התנהגותם בדפדפן. בסעיף שלאחריו תקבל סקירה על ההתנהגות המשותפת לכל פקדי השרת. כשתבין התנהגות משותפת זו, תמצא שקל ליישם אותה גם לפקדים אחרים, חדשים, ולהגיע במהירות לתפועל יעיל שלהם.

הגדרת פקדים בדפים שלך

כפי שהודגם בתרגיל נסה זאת קודם, אתה יכול לגרור פקדים מארגז הכלים אל משטח העיצוב (design surface) של הדף. על כן, הפעולה ההתחלתית של הוספת קבוצת פקדים לדף הופכת לקלה מאוד. עם זאת, בגלל הדרך שבה פועל משטח העיצוב, קשה לעתים להוסיף את הפקדים בדיוק במקום שבו אתה רוצה אותם. למרבה המזל, באותה מידה של קלות ביכולתך לגרור פקד מארגז הכלים ב-Markup View. בנוסף, ביכולתך להקליד את סימני העריכה של הפקד ישירות ב-Markup View, כשאתה מאפשר ל-IntelliSense לעזור לך עם המאפיינים והתגיות השונים. תמצא גם שטבלת המאפיינים פועלת ב-Markup View. פשוט לחץ על האלמנט הרלוונטי, וטבלת המאפיינים תעודכן כדי לשקף את התגית שעליה לחצת. זה מקל על שינוי המאפיינים של הפקד, בעודך רואה בדיוק איזה סימני עריכה נוצרים עבורך.

אם תסתכל על כמה פקדים בטבלת המאפיינים, תשים לב שרבים מהפקדים הם בעלי מאפיינים דומים. בסעיף הבא תראה בדיוק מי הם פקדים אלה ולמה הם משמשים.

מאפיינים משותפים לכל הפקדים

לרוב שרתי הפקד שנמצאים בארגז הכלים של VWD יש התנהגות משותפת מסוימת. לדוגמה, לכל פקד יש: ID שנותן לו זהות ייחודית בדף, מאפיין Runat שמוגדר תמיד כ-Server כדי לציין שהפקד חייב להיות מעובד בשרת.

פרט למאפיינים אלה יש לרבים מפקדי השרת מאפיינים משותפים נוספים. הטבלה הבאה מציגה את המאפיינים הנפוצים ביותר ומתארת את שימושם.

מאפיין	תיאור
AccessKey	מאפשר להגדיר תו מקלדת שבאמצעותו ניתן לגשת לפקד בצד הלקוח על ידי החזקת Alt ולחיצה על האות המקושרת.
BackColor ForeColor	מאפשר לשנות את צבע הרקע (BackColor) וצבע הטקסט (ForeColor) של הפקד שבדפדפן.
BorderColor BorderStyle BorderWidth	משנה את גבולות הפקד בדפדפן. הדמיון למאפייני הגבול ב-CSS (שראית בפרק הקודם) אינו מקרי. כל אחד משלושת מאפייני ASP.NET אלה ממופה ישירות למקבילו ב-CSS.
CssClass	מאפשר להגדיר את המאפיין class של HTML עבור הפקד שבדפדפן.
Enabled	קובע אם למשתמש יכולה להיות אינטראקציה עם הפקד בדפדפן. לרוגמה, עם ההגדרה Enabled="false" אינך יכול לבצע שום פעילות עם הפקד.
Font	מאפשר הגדרות עבור גופן, כגון: Font-Size, Font-Names, Font-Bold ו-Font.
Height Width	קובע את גובה ורוחב הפקד בדפדפן.
TabIndex	מגדיר את המאפיין tabindex של HTML שקובע את הסדר שבו יוכלו משתמשים לנוע בין הפקדים בדף על ידי הקשת Tab.
ToolTip	מאפשר להגדיר תיאור עבור הפקד בדפדפן. מוצג כמאפיין title ב-HTML. מוצג כאשר המשתמש מרחף עם העכבר על פני אלמנט HTML הרלוונטי.
Visible	קובע אם הפקד יישלח לדפדפן. פקד עם הגדרת Visible = false לעולם לא יישלח לדפדפן. הדבר שונה מאוד ממאפייני התצוגה והמראה של CSS (שראית בפרק הקודם) שמסתירים את האלמנט בצד הלקוח.

כדי לראות איך כל המאפיינים האלו מוצגים בדפדפן, שקול את השימוש בסימני העריכה הבאים עבור פקד השרת TextBox:

```
<asp:TextBox AccessKey="a" BackColor="Black" ForeColor="White"
  BorderColor="Blue" BorderStyle="Dashed" Font-Size="30px"
  BorderWidth="4" CssClass="TextBox" Enabled="true" Height="40"
  Width="200" TabIndex="1" ToolTip="Hover text here"
  Visible="true" ID="txtName" runat="server" Text="Hello World">
</asp:TextBox>
```

כאשר יוצג הדף בדפדפן עם הפקד הזה, תקבל בסיום את קוד HTML הבא:

```
<input name="TextBox1" type="text" value="Hello World" id="TextBox1" accesskey="a"
  tabindex="1" title="Hover text here" class="TextBox" style="color:White;
  background-color:Black;border-color:Blue;border-width:4px;
  border-style:Dashed;font-size:30px;height:40px;width:200px;"
/>
```

התוצאה של הקוד היא הצגה בדפדפן של תיבת הטקסט הנראית בתרשים 4-3. שים לב שרוחב מאפייני הפקד הועברו בקוד ה-HTML לסגנונות כלולים ברמת האלמנט של CSS תחת המאפיין .style.



תרשים 4-3

כשבונים אתרי אינטרנט, נדיר למדי להגדיר TextBox בצורה כזו. כפי שלמדנו בפרק הקודם, עליך להימנע ככל האפשר מסגנונות כלולים ברמת התגית ולהעדיף במקומם גיליונות סגנון חיצוניים. תוכל להשיג התנהגות זוהי לחלוטין על ידי שימוש בפקד צד-שרת זה:

```
<asp:TextBox ID="TextBox1" AccessKey="a" CssClass="TextBox" Enabled="true"
  TabIndex="1" ToolTip="Hover text here" Visible="true" runat="server"
  Text="Hello World">
</asp:TextBox>
```

ומחלקת CSS הבאה:

```
.TextBox
{
  color: White;
  background-color: Black;
  border-color: Blue;
  border-width: 4px;
  border-style: Dashed;
  font-size: 30px;
  height: 40px;
  width: 200px;
}
```

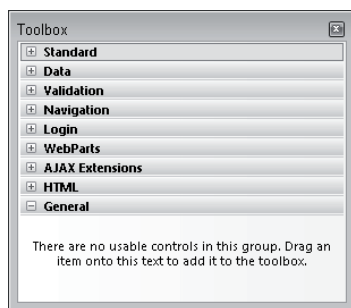
ברור שהדוגמה השנייה קלה הרבה יותר לקריאה, לתחזוקה ולשימוש חוזר. אם תרצה תיבת טקסט נוספת בעלת אותו מראה בדיוק, עליך רק להגדיר: `CssClass = "TextBox"` לאותו פקד.

למרות שמומלץ להשתמש במחלקות CSS במקום הסגנונות הכלולים האלה, טוב לדעת אודות מאפייני פקד צד-שרת למקרה שתזדקק לשליטה עדינה בהם. אם אתה משנה את מאפייני הפקד באמצעות קוד (דבר שתלמד לעשות מאוחר יותר), הם בסופו של תהליך יוגדרו כסגנונות כלולים, ולכן הם דורסים הגדרות בגיליונות סגנון חיצוניים וברמת הדף.

עכשיו, כשראית את ההתנהגות הכללית המשותפת לכל פקדי צד-שרת, הגיע הזמן להתבונן במספר הרב של פקדים שכלולים בתוכנה ASP.NET 3.5.

סוגי פקדים

ASP.NET 3.5 כוללת מספר רב של פקדי שרת שתומכים במרבית הצרכים של פיתוח אתרים. כדי להקל עליך למצוא את הפקד המתאים הוצבו הפקדים בקטגוריות נפרדות בארגז הכלים של סביבת הפיתוח (נגיש על ידי הקשה על **Ctrl+Alt+X**). תרשים 4-4 מראה את כל הקטגוריות הזמינות בארגז הכלים.



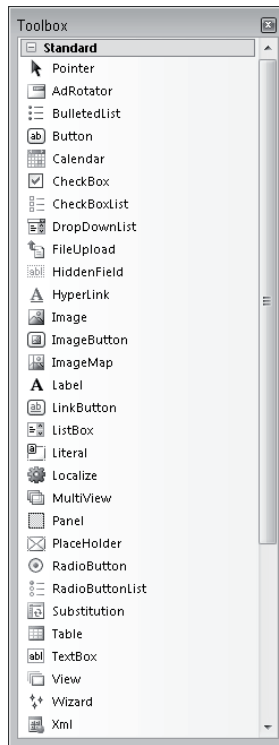
תרשים 4-4

בסעיפים הבאים תכיר את הפקדים השייכים לכל קטגוריה, המשימות שנועדו להם, וכיצד תוכל להשתמש בהם בדפי ASP.NET שלך.

בדיון על הפקדים השונים תראה אזכור של **מאפייני פקד**. לדוגמה, ל-`TextBox` יש מאפיין `Text` (בין רבים אחרים), בעוד של-`ListBox` יש מאפיין `SelectedItem`. כמה מאפיינים ניתנים להגדרה על ידי כתיבת קוד בלבד, ולא באמצעות טבלת המאפיינים (Properties Grid). בפרק הבא נדון בפירוט בקריאה ושינוי של מאפייני פקד על ידי שימוש בקוד.

פקדים בקטגוריה Standard

הקטגוריה **Standard** מכילה רבים מהפקדים הבסיסיים שכמעט כל דף אינטרנט זקוק להם. כמה מהם, כמו הפקדים `TextBox`, `Button` ו-`Label`, כבר ראית בתחילת פרק זה. תרשים 4-5 מציג את כל הפקדים בקטגוריה זו.



תרשים 4-5

שמות רבים מהפקדים מדברים בעד עצמם, כך שבמקום לתת תיאור מפורט של כל הפקדים האלה, יעסקו הסעיפים הבאים בכמה פקדים חשובים.

פקדים פשוטים

ארגז הכלים מכיל כמה פקדים בסיסיים, כמו Button, Label, HyperLink, RadioButton ו-CheckBox. הסמלים שלהם בארגז הכלים נותנים לך רמז טוב כיצד הם מופיעים בדפדפן. בשארית הספר נעשה שימוש רב בפקדים אלה.

פקדי רשימה

ארגז הכלים מכיל כמה פקדים שמוצגים בדפדפן כרשימות. פקדים אלה כוללים את ListBox, RadioButtonList, CheckBoxList, DropDownList ו-BulletedList. כדי להוסיף פריטים לרשימה, הגדר אלמנטים מסוג <asp:ListItem> בין תגיות הפתיחה והסגירה של הפקד, כפי שניתן לראות בדוגמה הבאה:

```
<asp:DropDownList ID="lstFavoriteLanguage" runat="server">
  <asp:ListItem Value="C#">C#</asp:ListItem>
  <asp:ListItem Value="Visual Basic">Visual Basic</asp:ListItem>
  <asp:ListItem Value="CSS">CSS</asp:ListItem>
</asp:DropDownList>
```


כדי לזהות את הפריט הנבחר ברשימה על ידי שימוש בקוד, עליך להסתכל במאפיינים SelectedValue, SelectedItem או SelectedIndex שלו. המאפיין SelectedValue מחזיר מחרוזת שמכילה את הערך עבור הפריט שנבחר, כמו C# או Visual Basic בדוגמה לעיל. SelectedIndex מחזיר אינדקס, שמתחיל מ-0, של הפריט ברשימה. בהתייחס לדוגמה לעיל, אם המשתמש בחר ב-C#-0, אזי SelectedIndex יהיה 0. באופן דומה, כשהמשתמש בחר ב-CSS, האינדקס יהיה 2 (CSS הוא הפריט השלישי ברשימה). הפקד BulletedList אינו מאפשר למשתמש לבחור, ולפיכך אינו תומך במאפיינים אלה.

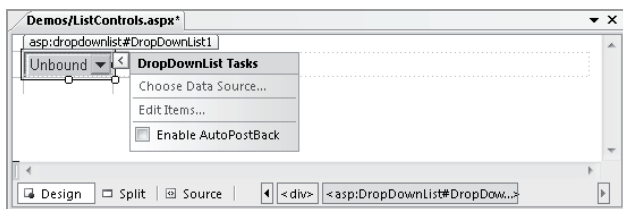
עבור פקדים שמאפשרים מספר בחירות (כמו CheckBoxLayout ו-ListBox), ניתן לבצע בדיקה על כל הפריטים שבאוסף Items ולראות אילו פריטים נבחרו. במקרה זה מחזיר SelectedItem את הפריט הראשון שנבחר ברשימה, ולא את רשימת כל אלה שנבחרו. בתרגיל הבא תראה איך לגשת לכל הפריטים שנבחרו.

בתרגיל הבא תלמד איך להוסיף פריטי רשימה לפקד רשימה, ואיך לקרוא את הערכים הנבחרים. תעשה זאת על ידי יצירת טופס web פשוט עם שני פקדי רשימה, ששואלים את המשתמש מהי שפת התכנות המועדפת עליו.

נסה זאת עבודה עם פקדי רשימה

בתרגיל זה תוסיף שני פקדי רשימה לדף. בנוסף, תוסיף לחצן, שכאשר לוחצים עליו, הוא מציג את רשימת הפריטים הנבחרים כטקסט בפקד Label.

1. בתיקיה Demos צור טופס אינטרנט חדש בשם ListControls.aspx. ודא שיצרת קובץ 'קוד ברקע' על ידי סימון האפשרות Place Code in Separate.
2. עבור ל- Design View וגרור DropDownList מארגז הכלים אל משטח העיצוב שבתחום הקווים המקווקווים של התגית <div> שכבר נמצאת בדף שלך.
3. שים לב שברגע שאתה משחרר את הפקד DropDownList בדף, מופיע תפריט קיצור שכותרתו DropDownList Tasks, כפי שמוצג בתרשים 4-6.

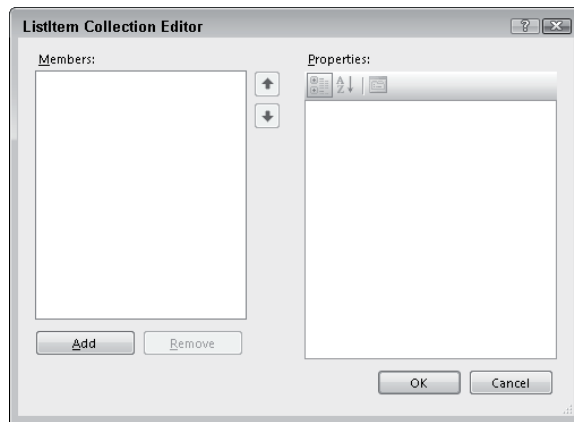


תרשים 4-6

⊙ תפריט קיצור זה נקרא Smart Tasks. כשהוא מופיע, הוא מאפשר גישה למשימות הנפוצות ביותר של הפקד שהוא משתייך אליו. במקרה של הפקד DropDownList תקבל שלוש אפשרויות. הראשונה מאפשרת לחבר את הפקד עם מקור נתונים, כפי שתלמד בפרק 12. השנייה מאפשרת להוסיף פריטים לרשימה באופן ידני. האפשרות האחרונה מגדירה את המאפיין AutoPostBack של הפקד. כשאפשרות

זו מסומנת, אזי ברגע שהשתמש בוחר פריט חדש מהרשימה, הדף בו נמצא יישלח חזרה אל השרת.

- ⊙ תפריט Smart Tasks מופיע רק עבור הפקדים היותר מורכבים, בעלי מספר רב של מאפיינים. לא תראה אותו עבור פקדים פשוטים כמו Button או Label. כדי לפתוח מחדש את לוח Smart Tasks, לחץ לחיצה ימנית על הפקד ב-designer ובחר Show Smart Tag. לחילופין, לחץ על החץ הקטן בפינה הימנית העליונה של הפקד (ראה תרשים 4-6).
- ⊙ בתפריט Smart Tasks של DropDownList, לחץ על הקישור Edit Items ויפתח חלון ListItem Collection Editor (ראה תרשים 4-7).
- ⊙ תיבת דו-שיח זו מאפשרת להוסיף פריטים חדשים לפקד הרשימה. הפריטים שתוסיף באמצעות חלון זה יתווספו כאלמנטים מסוג <asp:ListItem> בין התגיות של הפקד.



תרשים 4-7

4. כדי להכניס פריט חדש לרשימה, לחץ על Add שבצד השמאלי של המסך. בטבלת המאפיינים שבצד ימין הכנס את הערך C# עבור המאפיין Text, ולחץ Tab. ברגע שתדלג הלאה מהמאפיין Text, יועתק הערך גם למאפיין Value (בהחלט אפשרי, וגם מקובל, להקצות למאפיין Value ערך אחר).
5. חזור על הצעד הקודם פעמיים, כשאתה יוצר רשימת פריטים עבור Visual Basic ו-CSS. תוכל להשתמש בחיצים מעלה/מטה שבאמצע תיבת הדו-שיח כדי לשנות את סדר הופעת הפריטים ברשימה. לבסוף, לחץ OK כדי להכניס את הפריטים לדף. בסיום תקבל את הקוד הבא:

```
<asp:DropDownList ID="DropDownList1" runat="server">
  <asp:ListItem>C#</asp:ListItem>
  <asp:ListItem>Visual Basic</asp:ListItem>
  <asp:ListItem>CSS</asp:ListItem>
</asp:DropDownList>
```

6. אם יש צורך, עבור ל- Markup View וגרור פקד CheckBoxList מארגז הכלים ישירות אל חלון הקוד, מייד לאחר DropDownList.

7. העתק את שלושת האלמנטים מסוג <asp:ListItem> מה-DropDownList שיצרת בצעדים 4 ו-5, והדבק אותם בין תגיות הפתיחה והסגירה של CheckBoxList. בסיום תקבל את הקוד הבא:

```
<asp:ListItem>CSS</asp:ListItem>
</asp:DropDownList>
<asp:CheckBoxList ID="CheckBoxList1" runat="server">
  <asp:ListItem>C#</asp:ListItem>
  <asp:ListItem>Visual Basic</asp:ListItem>
  <asp:ListItem>CSS</asp:ListItem>
</asp:CheckBoxList>
```

8. עבור ל- Design View וגרור את Button מארגז הכלים ב- Design View אל מימין לפקד CheckBoxList. הפקד Button ימוקם תחת הפקד CheckBoxList. לאחר מכן גרור פקד Label ושחרר אותו ליד Button. צור רווח מסוים בין Button לבין Label על ידי הצבת הסמן בין הפקדים והקשה פעמיים על Enter. לחץ לחיצה כפולה על Button כדי לפתוח את קובץ הקוד של הדף.

9. בבלוק הקוד שהוסיפה עבורך VWD, עליך להוסיף את הקוד המוצלל הבא, אשר יבוצע כאשר המשתמש ילחץ על הלחצן.

VB.NET

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Button1.Click
    Label1.Text = "In the DDL you selected " & DropDownList1.SelectedValue & "<br />"

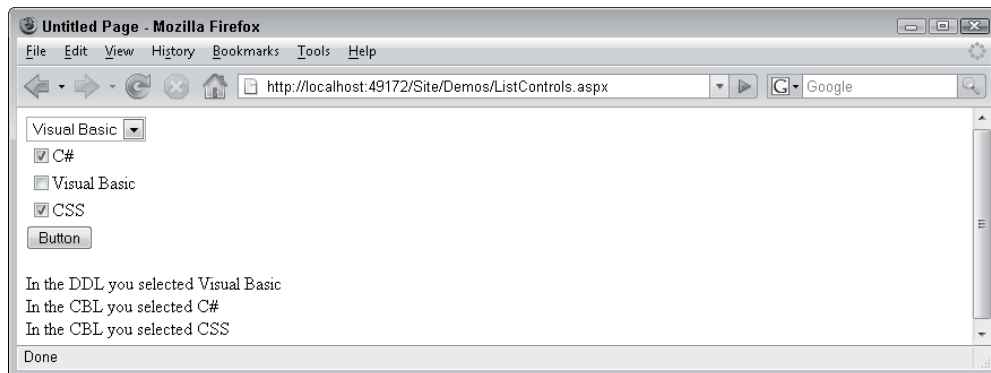
    For Each item As ListItem In CheckBoxList1.Items
        If item.Selected Then
            Label1.Text &= "In the CBL you selected " & item.Value & "<br />"
        End If
    Next
End Sub
```

C#

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = "In the DDL you selected " + DropDownList1.SelectedValue + "<br />";

    foreach (ListItem item in CheckBoxList1.Items)
    {
        if (item.Selected == true)
        {
            Label1.Text += "In the CBL you selected " + item.Value + "<br />";
        }
    }
}
```

10. שמור את השינויים שבוצעו בדף, ואז הרץ את הדף (Ctrl + F5). בחר פריט מתוך DropDownList, סמן פריט אחד או יותר ב-CheckBoxList ולחץ על הלחצן. כעת תראה משהו דומה לתרשים 4-8.



תרשים 4-8

כיצד זה פועל

כל פקדי הרשימה השונים משתמשים באלמנטים מסוג `<asp:ListItem>`. הדבר מקל על שימוש חוזר בפקדים על ידי העתקתם מפקד אחד לאחר. מכיוון שפקד DropDownList תומך רק בפריט נבחר אחד בכל רגע נתון, קל למדי לקבל את הערך הנבחר. כל מה שנדרש זו שורת קוד אחת (אשר מוצגת לפניך ב-C#):

```
Label1.Text = "In the DDL you selected " + DropDownList1.SelectedValue + "<br />";
```

הפקד CheckBoxList מאפשר למשתמש לבחור בכמה פריטים. לכן, אתה זקוק למעט יותר קוד כדי לבדוק את המאפיין Selected של כל פריט באוסף הפריטים (גם כאן, הקוד מוצג ב-C#):

```
foreach (ListItem item in CheckBoxList1.Items)
{
    if (item.Selected == true)
    {
        Label1.Text += "In the CBL you selected " + item.Value + "<br />";
    }
}
```

הפקד CheckBoxList ושאר פקדי הרשימה הם בעלי אוסף Items שמכיל את כל הפריטים שהגדרת בקוד. כך, על פי הקוד מתרגיל נסה זאת זה, הרשימה CheckBoxList מכילה שלושה פריטים: עבור C#, Visual Basic ו-CSS. בהתאמה. כל ListItem בתורו מכיל מאפיין Selected שקובע אם המשתמש סימן, או לא סימן, את הפריט ברשימה.

באמצעות הלולאה foreach (ראה For Each ב-VB.NET) אתה יכול לעבור על אוסף האלמנטים ListItem ולבדוק את המאפיין Selected עבור כל אחד מהפריטים. אם הפריט נבחר ברשימה, המאפיין Selected הוא true, וה-Value שלו מתווסף לטקסט של ה-Label. שים לב לשימוש של

`+=` בדוגמת הקוד האחרונה (`&=` ב-VB.NET) כדי להעביר את Value של פריט הרשימה יחד עם הטקסט "In the CBL you selected" אל המאפיין Text של הפקד Label. התחביר `+=` ו-`&=` הוא קיצור של:

```
Label1.Text = Label1.Text + "In the CBL you selected " + item.Value + "<br />";
```

הקוד לוקח את הטקסט הנוכחי מהפקד Label, מוסיף לו את המחרוזת:

```
"In the CBL you selected " + item.Value + "<br />"
```

ומקצה מחדש את המחרוזת כולה למאפיין Text של הפקד Label. השימוש בתחביר `+=` הוא לעתים קרובות קל יותר לכתיבה והבנה.

הן VB.NET והן C# תומכות בלולאה For Each, אם כי שתי השפות משתמשות בתחביר שונה במעט. בפרק הבא תלמד הרבה יותר על לולאות ומבנים אחרים של השפה.

שים לב גם לדרך שבה נבנית רשימת הפריטים ListItems. בדוגמה הראשונה, לפני התרגיל נסה זאת, ראית אלמנטים של ListItem עם ערך וטקסט:

```
<asp:ListItem Value="#">C#</asp:ListItem>
<asp:ListItem Value="Visual Basic">Visual Basic</asp:ListItem>
<asp:ListItem Value="CSS">CSS</asp:ListItem>
```

כשאתה מוסיף פריט לרשימה בעצמך, בעזרת ListItem Collection Editor, אינך מקבל את התכונות Value:

```
<asp:ListItem>C#</asp:ListItem>
<asp:ListItem>Visual Basic</asp:ListItem>
<asp:ListItem>CSS</asp:ListItem>
```

לא קיבלת את המאפיין Value, מכיוון שלא סיפקת ערך מפורש לכל פריט ב- ListItem Collection Editor. כשאתה משמיט את Value, הטקסט שבין תגיות הפתיחה והסגירה של ListItem משמש באופן עקיף כערך, מצב שהוא תקין במקרים רבים. עם זאת, הקיום של מאפייני Value ו-Text שונים ברשימה נפוץ למדי. לדוגמה, כאשר יש רשימה של ארצות, תוכל להשתמש בשם המלא של הארץ כ-Text (למשל, Israel) ולהשתמש בקוד הרשמי של הארץ (il) כ-Value עבור הרשימה הנפתחת. תראה את פקדי הרשימה בשימוש בפרקים אחרים בספר.

פקדי מכולה

לעתים קרובות למדי רצוי שתהיה יכולת לקבץ פקדים ותוכן שיש ביניהם קשר. ניתן להשיג הקבצה זו על ידי כך ששמים את הפקדים (וסימני עריכה אחרים) תחת פקד מכולה כמו Panel, Placeholder, MultiView או Wizard. לדוגמה, אתה יכול להשתמש בפקד Placeholder, או בפקד Panel, כדי להסתיר או להראות כמה פקדים בו-בזמן. במקום להסתיר כל פקד בנפרד, אתה מסתיר את המכולה שכוללת את כל הפקדים הפרטיים ואת סימני העריכה. לכל אחד משני הפקדים יתרונות וחסרונות. היתרון של הפקד Placeholder בכך שהוא אינו מפיק HTML משלו

לדף, ולכן אתה יכול להשתמש בו כמכולה מבלי שתהיינה לו תופעות לוואי בדף הסופי. עם זאת אין לו תמיכה בזמן העיצוב, והדבר מקשה על ניהול הפקדים שבתוך Placeholder בזמן העיצוב ב-VWD. בניגוד לכך, הפקד Panel מאפשר גישה קלה לכל הפקדים ולתכנים אחרים שנמצאים בו, אבל הוא בסופו של דבר יוצג בדפדפן כתגית <div>. במקרים רבים אין בכך בעיה, ולכן עדיף בדרך-כלל להשתמש בפקד Panel בגלל תמיכתו בזמן העיצוב.

הפקד MultiView (אשר יכול להכיל פקד <asp:View> אחד או יותר) והפקד Wizard דומים, בכך שהם מאפשרים לפצל דף ארוך למספר שטחים או תחומים. הדבר מקל על השליטה והעבודה בדף, כמו לדוגמה, בעת מילוי טופס ארוך. לפקד Wizard יש תמיכה מובנית בתנועה מדף לדף תוך שימוש בלחצנים: "הקודם" (Previous), "הבא" (Next) ו"סיום" (Finish), בעוד שאת פקד MultiView ניתן להגדיר באמצעות תכנות בקוד.

מבט מקרוב על הפקד Panel

בתרגיל הבא תשתמש בפקד Panel כדי ליצור מכולה עבור פקדים וסימני עריכה אחרים. בשלב זה תוסיף רק מעט טקסט, אבל בתרגיל נסה זאת שיבוא לאחר מכן תוסיף פקדי ASP.NET אל תוך פקד Panel.

נסה זאת שימוש בפקד Panel

בתרגיל זה תראה כיצד להשתמש בפקד Panel כמכולה עבור טקסט פשוט. בנוסף, תשתמש בפקד CheckBox פשוט כדי לשלוט בהצגה של פקד ה-Panel מצד השרת.

1. בתיקיה Demos, צור טופס אינטרנט חדש עם 'קוד ברקע', בשם Containers.aspx.
2. העבר את הדף ל- Design View וגרור פקד CheckBox ופקד Panel מארגז הכלים אל משטח העיצוב, לתוך אלמנט <div> המקווקו.
3. הוסף לפקד CheckBox תיאור משמעותי על ידי הגדרת המאפיין Text שלו כ- " Show Panel", והגדר את המאפיין AutoPostBack שלו כ-True, תוך שימוש בטבלת המאפיינים.
4. הגדר את המאפיין Visible של הפקד Panel כ-False תוך שימוש בטבלת המאפיינים. דבר זה יגרום להסתרת הפקד Panel כשהדף ייטען לראשונה.
5. הקלד טקסט כלשהו בתוך הפקד Panel (למשל, **I am visible now**). שים לב שה-Panel מתנהג כמו שאר חלקי משטח העיצוב של VWD. אתה יכול להוסיף לו בקלות טקסט, לבחור בו ולעצב אותו, ואפילו להוסיף לו פקדים על ידי גרירתם מארגז הכלים. הקוד עבור ה-Panel אמור להיראות בערך כך:

```
<asp:Panel ID="Panel1" runat="server" Height="50px" Visible="False" Width="125px">
  I am visible now
</asp:Panel>
```

6. לחץ לחיצה כפולה על הפקד CheckBox ב- Design View, ובתוך הקוד ש-VWD הוסיפה עבורך, הכנס את שורת הקוד המוצללת הבאה:

VB.NET

```
Protected Sub CheckBox1_CheckedChanged(ByVal sender As Object, _  
    ByVal e As System.EventArgs) Handles CheckBox1.CheckedChanged  
    Panel1.Visible = CheckBox1.Checked  
End Sub
```

C#

```
protected void CheckBox1_CheckedChanged(object sender, EventArgs e)  
{  
    Panel1.Visible = CheckBox1.Checked;  
}
```

7. שמור את כל השינויים, הקש Ctrl+F5 כדי לפתוח את הדף בדפדפן.

8. כשהדף ייטען לראשונה, תוכל לראות את פקד ה-CheckBox והטקסט שלצדו. כשתלחץ על הפקד CheckBox כדי לסמנו, ייטען הדף מחדש ויראה כעת את הטקסט שהקלדת בצעד 5. אם שום דבר לא קורה, חזור למקור של הדף ב-VWD וודא שמאפיין בשם AutoPostBack מוגדר כ-True בפקד CheckBox.

אם תתבונן ב-HTML בדפדפן (לחץ לחיצה ימנית על הדף ובחר View Source או View Page Source) תראה שהטקסט שהקלדת בצעד 5 מוקף בתגית <div> עם id שמוגדרת כ-Panel1.

```
<div id="Panel1" style="height:50px;width:125px;">  
    I am visible now  
</div>
```

כיצד זה פועל

בצעד 4 של התרגיל הגדרת את המאפיין Visible של הפקד Panel כ-False. על כן, כאשר הדף נטען, הפקד אינו נראה בצד השרת וה-HTML שלו לעולם לא יגיע לדפדפן. כשסימנת את פקד CheckBox התרחש postback, אשר שלח את המידע הכלול בטופס אל השרת. בכל פעם שתיבת הסימון משנה את מצבה ממסומנת ללא-מסומנת, או להיפך, מתבצע בשרת קוד מסוים. בתוך בלוק קוד זה מתבצעת השורה הבאה (מוצגת ב-C#):

```
Panel1.Visible = CheckBox1.Checked;
```

פירוש הדבר שהפקד Panel נראה רק כאשר תיבת הסימון מסומנת. כשהיא אינה מסומנת, הפקד Panel מוסתר.

בפרק הבא תלמד הרבה יותר אודות הקוד שגורם לכל זה לקרות, ומדוע אתה זקוק לשורה זו.

כפי שאתה רואה, קל להוסיף טקסט וסימני עריכה אחרים לפקד Panel. כרגע הוספת טקסט פשוט בלבד, אבל בסעיף הבא תראה איך להוסיף את הפקד Wizard ואיך להשתמש בו.

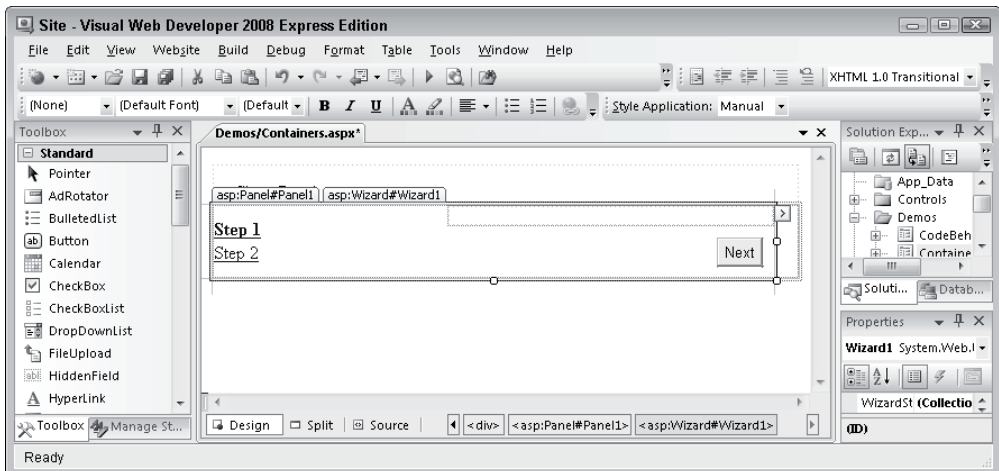
הפקד Wizard

הפקד Wizard הוא כלי מעולה לשבירת טפסי web גדולים לחלקים אשר "ניתנים לעיכול", ומקלים על הצגת המידע שבהם למשתמש. במקום לבלבל את המשתמש עם דף אחד שבו פקדים רבים וטקסט, ביכולתך 'לשבור' את הדף לחלקים, ולהציג כל חלק בדף נפרד של ה-Wizard. פקד זה מטפל בכל נושאי הניווט על ידי יצירה אוטומטית של הלחצנים "הבא" (Next), "הקודם" (Previous) ו"סיום" (Finish). בתרגיל הבא תשתמש ב-wizard כדי לשאול משתמש לשמו, ומהי שפת התכנות המועדפת עליו. למרות שהדוגמה עצמה שטחית למדי, ובקלות היית יכול למקם את שתי השאלות באותו הדף, היא מראה כיצד ה-wizard פועל ומדוע הוא שימושי. תוכל ליישם בקלות טכניקות אלו על הטפסים שלך, שייתכן שהם גדולים יותר.

נסה זאת שימוש ב- Wizard ליצירת טפסים קלים לשימוש

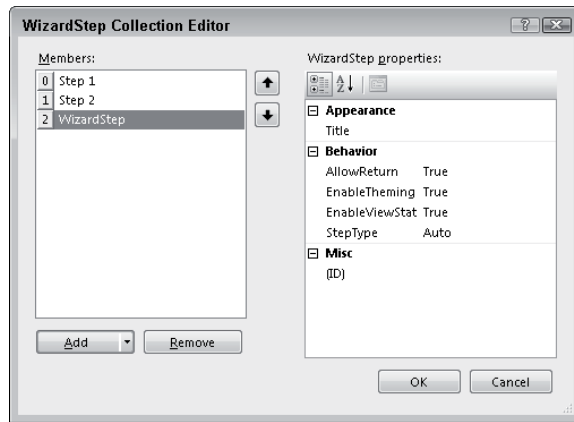
בתוך ה-Panel שיצרת בתרגיל הקודם תמקם פקד Wizard שמאפשר למשתמש למלא טופס שמשתרע על פני כמה דפים. ל-wizard יהיו שני צעדים בהם יוכל משתמש להכניס פרטים, ודף תוצאות שיציג את הנתונים שסיפק המשתמש.

1. ודא שהדף Containers.aspx עדיין פתוח ב- Design View. הסר את הטקסט "I am visible now" שהקלדת בתרגיל **נסה זאת** הקודם, ואז גרור פקד Wizard מארגז הכלים לתוך ה-Panel. גרור את שפתו הימנית רחוק יותר ימינה, כדי להגדיל את רוחב הפקד ל-500 פיקסל. עבור ל- Markup View והסר את המאפיינים Width ו-Height והערכים שלהן מהפקד Panel (השאר את Width של Wizard מוגדר ל-500 פיקסל). כאשר תעבור חזרה ל- Design View, הדף שלך יהיה דומה לתרשים 4-9.



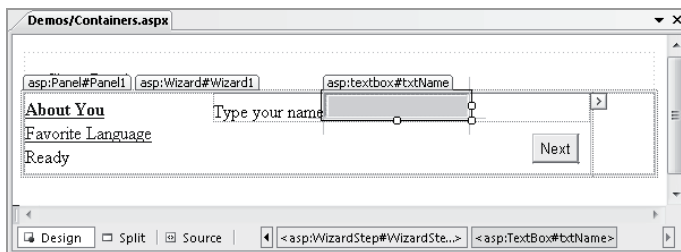
תרשים 4-9

2. ב- Design View לחץ לחיצה ימנית על Wizard ובחר Add/Remove WizardSteps. בתיבת הדו-שיח שמופיעה, לחץ על Add כדי להוסיף צעד שלישי לאשף, כפי שמוצג בתרשים 4-10.



תרשים 4-10

3. ברשימה Members שבצד שמאל לחץ על WizardStep הראשון שנושא את התווית Step 1, ושנה את ה-Title שלו מ- Step 1 ל- About You. הגדר את ה-Title של שני הצעדים האחרים ל- Ready ו- Favorite Language בהתאמה.
4. שנה את StepType של הצעד השני (שכעת נושא את התווית Favorite Language) ל- Finish, ואת זה של הצעד האחרון שנה ל- Complete. אתה יכול להשאיר את StepType של הצעד הראשון מוגדר כ- Auto. לחץ OK לאישור.
5. ב- Design View לחץ על About You ברשימה שבצד שמאל, כדי לגרום לו להיות הצעד הפעיל, ואז גרור Label ו- TextBox לצד הימני של ה-Wizard. אתה צריך לגרור אותם אל תוך המלבן הכחול שבפינה הימנית העליונה של ה-Wizard, אחרת הפקדים לא יוכנסו אל ה-Wizard. הגדר את המאפיין Text של Label ל- Type your name, ואז שנה את ה-ID של TextBox ל- txtName. כשתסיים האשף שלך יהיה כמו זה שבתרשים 4-11.



תרשים 4-11

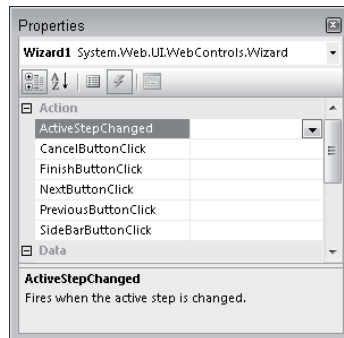
6. לחץ על הפריט Favorite Language ברשימה שמשמאל, ואז הוסף פקד DropDownList למלבן עם הגבול הכחול של צעד האשף. הגדר את ID הפקד ל- lstFavoriteLanguage. פתח את לוח Smart Tasks של הפקד DropDownList ובחר Edit Items. הוסף את שלושת הפריטים שהוספת בתרגיל נסה זאת הקודם: עבור C#, Visual Basic ו- CSS בהתאמה. אם רצונך בכך, תוכל להעתיק את שלושת הפריטים מהדף ListControls.aspx ולהדביק אותם בין תגיות <asp:DropDownList> שבצעד השני. כעת תקבל את הקוד הבא עבור הצעד השני:

```

</asp:WizardStep>
<asp:WizardStep ID="WizardStep2" runat="server" Title="Favorite Language"
    StepType="Finish">
    <asp:DropDownList ID="lstFavoriteLanguage" runat="server">
        <asp:ListItem>C#</asp:ListItem>
        <asp:ListItem>Visual Basic</asp:ListItem>
        <asp:ListItem>CSS</asp:ListItem>
    </asp:DropDownList>
</asp:WizardStep>
<asp:WizardStep runat="server" Title="Ready">

```

7. עבור ל- Markup View ואז, בתוך WizardStep האחרון שנושא את התווית Ready, גרור פקד Label מארגז הכלים והגדר את ה-ID שלו כ- lblResult. אם תנסה לעבור לצעד האחרון ב- Design View, אתה עשוי לשים לב לכך שהאשף נעלם. אם זה קרה, חזור אל Markup View והגדר שוב את מאפיין ה-wizard - ActiveStepIndex כ- 0.
8. לחץ לחיצה כפולה על ה-wizard ב- Design View והוסף את הקוד המוצלל הבא. קוד זה יתבצע כשהמשתמש ילחץ על הלחצן Finish בצעד האחרון. אם יש לך קשיים לגרום ל-WVD ליצור עבורך את הקוד הנכון (ראה קטע הקוד הבא), בחר ב-Wizard, פתח את טבלת המאפיינים של הפקד על ידי הקשה על F4, ואז לחץ על הלחצן עם סמל הברק (הלחצן הרביעי משמאל בסרגל הכלים של רשת המאפיינים) שמוצג בתרשים 4-12.



תרשים 4-12

בקטגוריית Action לחץ לחיצה כפולה על FinishButtonClick. בשתי השיטות אתה אמור לסיים עם הקוד כפי שניתן לראות להלן (בפרק הבא תלמד מה עושה הלחצן עם סמל הברק ולמה הוא משמש):

VB.NET

```

Protected Sub Wizard1_FinishButtonClick(ByVal sender As Object, ByVal e As _
    System.Web.UI.WebControls.WizardNavigationEventArgs) _
    Handles Wizard1.FinishButtonClick
    lblResult.Text = "Your name is " & txtName.Text
    lblResult.Text &= "<br />Your favorite language is " & _
        lstFavoriteLanguage.SelectedValue
End Sub

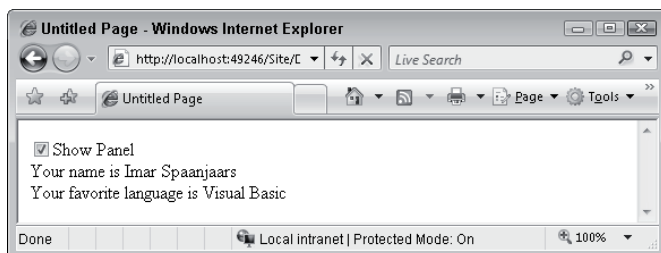
```

C#

```
protected void Wizard1_FinishButtonClick(object sender, WizardNavigationEventArgs e)
{
    lblResult.Text = "Your name is " + txtName.Text;
    lblResult.Text += "<br />Your favorite language is " +
        lstFavoriteLanguage.SelectedValue;
}
```

9. עבור חזרה ל- Design View, פתח את טבלת המאפיינים של ה-Wizard וודא ש-ActiveStepIndex מוגדר כ-0. ה-designer זוכר את הצעד האחרון שעיצבת ומאחסן את הערך ב-ActiveStepIndex של ה-Wizard ב- Markup View. כדי להבטיח שהאשף יתחיל תמיד בדף הראשון, עליך להחזיר תמיד את הגדרת ActiveStepIndex ל-0 (או ללחוץ על הצעד הראשון בפקד Wizard ב- Design View) לפני שאתה שומר את השינויים ומריץ את הדף.

10. הקש Ctrl+F5 כדי לפתוח את הדף בדפדפן. סמן את פקד CheckBox כדי לגרום לפקד Panel להיראות, ואז הקלד את שמך בדף הראשון של ה-wizard. לחץ Next ובחר בשפת התכנות המועדפת עליך. שים לב שכעת זמין עבורך לחצן Previous, שמאפשר לחזור אל הצעד הראשון של האשף אם ברצונך לשנות את שמך. במקום ללחוץ על הלחצנים Next ו-Previous, אתה יכול ללחוץ על הקישורים שנמצאים בדפדפן משמאל ל-wizard. כשתלחץ על הלחצן Finish, תראה את סיכום המידע שהקלדת ב-wizard (ראה תרשים 4-13).



תרשים 4-13

כיצד זה פועל

הפקד Wizard מבצע עבורך את מרבית העבודה הידנית. הוא מטפל בניווט, קובע מתי להראות את הלחצנים הנכונים (כמו Next, Previous ו-Finish). הוא מבטיח שהערכים עבור הפקדים שהוספת בצעדים עדיין יהיו זמינים בדף שמתקבל, וכך תוכל להציגם תחת פקד Label. wizard עושה שימוש במנגנון מצב תצוגה (view state) שעליו תלמד בפירוט לקראת סוף הפרק.

כל שעליך לעשות כעת הוא להגדיר את הצעדים ולקבוע את ה-StepType שלהם. ה-StepType של הצעד הראשון הוגדר כ-Auto. על-סמך הגדרה זו מחליט האשף איזה לחצנים להציג. מכיוון שזהו הצעד הראשון ב-Wizard, הוא אינו מציג את הלחצן Previous כי אין צעד קודם שאפשר ללכת אליו. את ה-StepType של הצעד השני הגדרת כ-Finish, והדבר אומר

לאשף לשרטט לחצן Previous ולחצן Next. כאשר אתה לוחץ על הלחצן Finish, האשף מגיע לצעד האחרון שה-StepType שלו מוגדר כ-Complete. בצעד זה מוסתרים לחצני הניווט, וכל מה שתראה זה סיכום בחירותיך מוצג בפקד ה-Label (בעזרת הקוד בצד השרת). בפרק 5 תלמד על המושגים של הקוד ב'קוד ברקע' שמאפשרים זאת.

פרט לפקדים שראית בפרק הקודם, יש כמה פקדים נוספים שכדאי לבחון אותם. לא כולם נדונים כאן בפירוט, מכיוון שלא נשתמש בהם יותר בספר. תוכל למצוא תיאור מפורט של פקדים אלה בתיעוד של MSDN שתקבל עם VWD.

פקדים נוספים בקטגוריה Standard

סעיף זה דן בקצרה בשאר הפקדים בקטגוריה Standard תחת ארגז הכלים. תיווכח שרבים מהם משמשים ביישום לדוגמה שנעסוק בו בהמשך הלימוד.

ImageButton ו-LinkButton

הפקדים ImageButton ו-LinkButton פועלים באופן דומה לפקד Button רגיל. שניהם גורמים ל-postback לשרת כאשר לוחצים עליהם. ה-LinkButton מציג עצמו כאלמנט <a> פשוט אבל שולח נתונים חזרה לשרת (postback) במקום לבקש דף חדש. ה-ImageButton עושה את אותה פעולה, אבל מציג תמונה שהמשתמש יכול ללחוץ עליה כדי לעורר את פעולת ה-postback.

Image ו-ImageMap

פקדים אלה דומים למדי בכך שהם מציגים תמונה בדפדפן. ImageMap מאפשר להגדיר "נקודות חמות" בתמונה, שכאשר לוחצים עליהן נגרם postback לשרת, או שהן מנווטות אותך לדף אחר.

Calendar

הפקד Calendar מאפשר למשתמש לבחור תאריך מחלון בעל ממשק עשיר. תלמד על כך יותר לקראת סוף הפרק, כשנדון במנוע המצב (state engine) של ASP.NET.

FileUpload

הפקד FileUpload מאפשר למשתמש לטעון קבצים שיכולים להיות מאוחסנים בשרת. תפגוש בפקד זה בהמשך הספר.

Literal, Localize ו-Substitute

שלושה פקדים אלה נראים כדומים מעט לפקד Label, מכיוון שכולם יכולים להציג טקסט סטטי או HTML. היתרון הגדול ביותר של Literal הוא בכך שאינו הופך בדפדפן לתגית כלשהי, אלא רק מציג את מה שהוקצה למאפיין Text שלו. על כן הוא שימושי מאוד להצגת HTML או JavaScript שמגיע מצד השרת, או מבסיס נתונים.

AdRotator

פקד זה מאפשר להציג פרסומות אקראיות באתר האינטרנט. הפרסומות מגיעות מקובץ XML שאתה יוצר בשרת שלך. הפקד אינו בשימוש רב באתרי אינטרנט עכשוויים משום שהוא חסר תכונות מתקדמות, כמו מעקב ורישום של הקשות וכניסות של משתמשים, כפי שנדרש ברוב המקרים, פרט לפשוטים ביותר.

HiddenField

פקד זה מאפשר לאחסן נתונים בדף שנשלח עם כל בקשה. הדבר שימושי אם ברצונך שהדף יזכור נתונים מסוימים מבלי שהמשתמש יראה אותם בדף. לעולם אל תאחסן נתונים רגישים בפקד זה, משום שהשדה מוצג בקוד המקור HTML של הדף, ולפיכך נגיש למשתמש הקצה.

XML

הפקד XML מאפשר להמיר נתונים מפורמט XML לפורמט אחר (דוגמת XHTML) לצורך הצגתם בדף.

Table

הפקד `<asp:Table>` זהה במובנים רבים למקבילו `<table>` ב-HTML. עם זאת, מכיוון שהפקד פועל בשרת, באפשרותך לתכנת באמצעותו, ליצור באופן דינמי עמודות ושורות חדשות ולהוסיף להן נתונים דינמיים.

בזאת סיימנו את הדיון בפקדים בקטגוריה Standard של ארגז הכלים. במרבית דפי האינטרנט תשתמש לפחות בכמה מהם. בהמשך סעיף זה נדון בקטגוריות האחרות של ארגז הכלים. מכיוון שמכאן ואילך נשתמש ברובם בצורה כלשהי, ניתן להלן רק תיאור קצר שלהם, כדי שתדע מה תפקידם ולמה הם משמשים. כדי למצוא מידע נוסף אודותם שים לב להפניות אל פרקים אחרים.

פקדי HTML

הקטגוריה HTML מכילה כמה פקדי HTML שנראים דומים לאלה שישנם בקטגוריה Standard. לדוגמה, אתה מוצא את `Input (Button)` שנראה כמו `<asp:Button>`. באופן דומה, המקבילים לפקד `Select` הם `<asp:DropDownList>` ו-`<asp:ListBox>`.

פקדי HTML, בניגוד לפקדי שרת ASP.NET, הם פקדי צד-לקוח, ומסיימים דרכם ישירות ב-HTML הסופי בדפדפן. תוכל לחשוף אותם לקוד צד-שרת על ידי הוספת המאפיין `Runat="Server"`. מהלך כזה מאפשר גישה אליהם באמצעות תכנות בצד השרת, כדי להשפיע, למשל, על המראה שלהם. לפקדי HTML יש פונקציונליות פחותה בהרבה מזו של פקדי הקטגוריה Standard. לפקד `Select`, לדוגמה, אין תמיכה בזמן עיצוב להוספת פריטים חדשים לרשימה באמצעות `Listitem Collection`. Editor. דבר זה מכריח אותך לכתוב את הפריטים באופן ידני ב- Markup View של VWD.

מכיוון שהפקדים בקטגוריות Standard ו-HTML דומים למדי זה לזה, ידון הסעיף הבא בהבדלים ביניהם, ויצג כמה רעיונות מתי להעדיף קטגוריה אחת על השנייה.

איך לבחור בין פקדי Standard לבין פקדי HTML

כאמור, יש חפיפה מסוימת בין פקדי הקטגוריות Standard ו-HTML של ארגז הכלים. אם כך, במי תבחר, ומתי? ככלל, פקדי השרת האמיתיים שבקטגוריה Standard מציעים פונקציונליות רבה יותר, הן במונחי תמיכה בזמן עיצוב ב-VWD, והן במה שהם יכולים לעשות בזמן ביצוע. אבל לפונקציונליות זו יש מחיר. בגלל מורכבותם הגדולה יותר, זמן העיבוד שלהם ארוך יותר. עם זאת, ברוב אתרי האינטרנט לא תרגיש בהבדל. רק באתר עם עומס תנועה גבוה ופקדים רבים בדף, יתנו פקדי HTML ביצועים טובים יותר ויצרכו פחות זיכרון בשרת בהשוואה לפקדי שרת. ראוי לציין שהפרשי הביצועים הינם קטנים.

ברוב התרחישים העדף את פקדי השרת על פני פקדי HTML המקבילים להם. כתוצאה מהפונקציונליות הרבה, מציעים פקדי השרת גמישות גדולה יותר ביצירה והפעלה של הדפים, ומספקים חוויית עבודה עשירה יותר עבור המשתמש. גם התמיכה הטובה יותר בזמן עיצוב עושה את פקדי השרת לבחירה ראויה.

בחר בפקדי HTML אם אתה בטוח לחלוטין שאינך זקוק לפונקציונליות שמציעים לך פקדי השרת או במידה ואין לך צורך לגשת לפקדים אלו בצד השרת. בהמשך סעיף זה תקבל סקירה מהירה של הקטגוריות האחרות שבארגז הכלים.

פקדי נתונים (Data)

פקדי נתונים נכנסו לשימוש ב-ASP.NET 2.0. הם מאפשרים דרך קלה מאוד לגישה למקורות שונים של נתונים, כמו בסיסי נתונים, קבצי XML ואובייקטים. במקום לכתוב קוד רב לגישה למקורות הנתונים, כפי שצריך היה לעשות בגרסאות קודמות של ASP.NET, אתה צריך רק לכוון את פקד הנתונים למקור נתונים מתאים, והרכיב ASP.NET runtime דואג לרוב הנושאים הקשים. תלמד הרבה יותר אודות פקדים אלה בפרקים שעוסקים בנתונים, החל מפרק 12.

פקדי בדיקת תקפות (Validation)

פקדי בדיקת תקפות מאפשרים ליצור במהירות טפסי אינטרנט עם בדיקת תקפות שמונעת משתמשים מהכנסת נתונים שאינם תקפים. לדוגמה, אתה יכול לכפות על משתמשים להכניס ערכים לשדות בהם חובה להזין נתונים, ולבדוק אם הנתונים שהוכנסו תואמים לתבנית נדרשת כמו תאריך תקף או מספר בין 1 ל-10. הם גם מאפשרים לכתוב קוד מותאם ליצירת פונקציות שירות תקפות שאינן מכוסות על ידי הפקדים התקנים. היופי של פקדי בדיקת תקפות בכך שהם יכולים להתבצע הן בצד הלקוח והן בצד השרת. פרק 9 עוסק בפקדים אלה.

פקדי ניווט (Navigation)

הפקדים בקטגוריה Navigation של ארגז הכלים מאפשרים למשתמש למצוא את דרכו באתר. הפקד TreeView מציג נתונים בתצוגה היררכית, ויכול לשמש להצגת מבנה האתר שלך, תוך

מתן גישה קלה לכל הדפים שנמצאים בו. הפקד Menu עושה פעולה דומה, ומספק אפשרויות לקבל תפריטים נפתחים בכיוון אופקי או אנכי.

הפקד SiteMapPath יוצר טכניקת ניווט בדפי web שלך, שמאפשרת למשתמשים למצוא בקלות את דרכם בהיררכיית הדפים.

בפרק 7, שעוסק באופן בלעדי בניווט באתרי אינטרנט, תפגוש שוב את כל הפקדים האלה.

פקדי הזדהות - כניסה למערכת (Login)

כמו פקדי הנתונים כך גם פקדי הכניסה למערכת הוכנסו לשימוש ב- ASP.NET 2.0, ועשו את דרכם ל- ASP.NET 3.5. פקדים אלה מאפשרים ליצור, במאמץ קטן מאוד, אתרי אינטרנט מאובטחים שהמשתמשים נדרשים להירשם אליהם כדי להיכנס למערכת, לפני שהם מורשים לגשת לחלקים מסוימים של האתר, או אפילו לאתר בכללותו. בנוסף, פקדים אלה מספקים למשתמשים כלים לשינוי הסיסמה שלהם ובמקרה ששכחו את הסיסמה הישנה יכולים לבקש סיסמה חדשה, וגם להציג נתונים שונים בהתאם לתפקיד של המשתמש. פרק 15 מספק פרטים נוספים אודות תכונות האבטחה ופקדי ההזדהות של ASP.NET.

הרחבות Ajax

למעלה משנה לאחר השחרור הרשמי של ASP.NET 2.0, בנובמבר 2005, שחררה מיקרוסופט לשוק את ASP.NET 2.0 Ajax Extensions 1.0 כתוספת ל- ASP.NET 2.0. הרחבות אלו מאפשרות ליצור יישומי אינטרנט אשר נותנים תחושה כמו בעבודה מול מערכות מקומיות שמסוגלים לאחזר נתונים מהשרת באמצעות JavaScript שבצד-לקוח, ללא postback מלא. מאז ש-Ajax קיבעה את מעמדה כטכנולוגיה מקובלת ב-2005, התאמצה מיקרוסופט להיות בראש מיישמי Ajax. הגרסה האחרונה של VWD אינה חריגה מבחינה זו: Ajax שולבה באופן מלא בסביבת .NET, תוך מתן גישה קלה לקבוצה העשירה של מאפייניה. פרק 10 עוסק ב-Ajax.

עתה, לאחר שראית את הקבוצה הגדולה של פקדים שכלולה ב- ASP.NET 3.5, נעלה נושא חשוב אחר: מנוע המצב של ASP.NET. למרות שהתנהגות כמה מהפקדים שראית עשויה להיראות כדבר פשוט ושקוף, הרי שמאחורי הפונקציונליות שלהם עומד למעשה מנגנון טכני מובנה ולוגי. בסעיף הבא נעסוק בפירוט בעולם של מנוע המצב, נלמד מהו, מה זה מצב תצוגה, כיצד פועל המנוע ואיך ניתן להשתמש בו.

מנוע המצב (State Engine) של ASP.NET

בפרק הקודם יצרת דף על ידי הפקדים TextBox ו-Button. בתרגיל נסה זאת לאחר הצגת הדף בדפדפן, הקלדת מעט טקסט ולחצת על הלחצן. הלחצן גרם ל-postback אל השרת, וכאשר הדף נטען מחדש, הטקסט נשאר בתיבת הטקסט. בפרק זה עשית דבר דומה למדי עם הפקד Wizard, כאשר הערכים מתיבת הטקסט ומהרשימה הנפתחת נשמרו גם הם. אם אתה מכיר

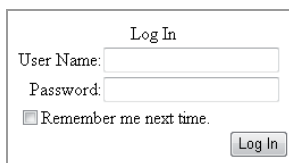
טכנולוגיות אינטרנט אחרות, דוגמת ASP או PHP, סביר שהדבר הפתיע אותך. בשפות אלו אתה נדרש לעתים קרובות לכתוב קוד רב כדי לגרום לכך לקרות. אם כך, מדוע ואיך הדבר מתבצע באופן אוטומטי ב-ASP.NET?

הטקסט בתיבת הטקסט נשמר על ידי מנוע המצב של ASP.NET. תכונה זו ב-ASP.NET runtime מאפשרת לפקדים לשמור מצב על פני postbacks באופן שהערכים וההגדרות שלהם נשארים זמינים אחרי כל postback של הדף.

מה זה מצב (State) ומדוע הוא חשוב?

כדי להבין מהו מצב, חשוב להבין ש-HTTP – הפרוטוקול שמשמש להעברת הבקשה מדפדפן לשרת ובחזרה – הינו, מעצם עיצובו, חסר מצב (stateless). פירוש הדבר שהשרת אינו עוקב אחר הבקשות שנעשו מדפדפן מסוים. ככל שהדבר נוגע לשרת האינטרנט, כל בקשה שאתה מפנה אליו על ידי דפדוף אל דף כלשהו ולחיצה על קישורים לדפים אחרים, עומדת בפני עצמה. השרת אינו 'זוכר' את הדפים שביקשת קודם.

דרך עבודה זו מעוררת כמה בעיות מעניינות. קח לדוגמה דף פשוט של כניסה למערכת, שמאפשר להיכנס לאתר אינטרנט כמו, למשל, תוכנית הדואר האלקטרוני המועדפת עליך. דוגמה לתיבת כניסה למערכת מופיעה בתרשים 4-14.



תרשים 4-14

כעת דמיין לעצמך שאתה מנסה להיכנס למערכת עם שם משתמש נכון, אבל עם סיסמה שגויה. הדף יודיע לך שניסיון הכניסה למערכת נכשל. באופן אידיאלי היית רוצה ששם המשתמש שלך ימולא באופן אוטומטי, ושהבחירה שעשית בתיבת הסימון Remember me next time תישמר אף היא. בדרך זו קל למשתמש להקליד את הסיסמה הנכונה וללחוץ על הלחצן Log In פעם נוספת. זו דוגמה פשוטה למדי, אבל קל למצוא תרחישים רבים נוספים שלגביהם היה שימושי אילו יכלו פקדים לשמור את מצבם לעת הצורך.

מכל מקום, כברירת מחדל, דף או פקד אינם יכולים לעשות זאת עבור עצמם. מכיוון שכל בקשה עומדת לעצמה, השרת לא ימלא שוב את תיבות הטקסט לאחר postback, אלא יעביר את הדף לדפדפן בדרך שעשה זאת כשטען את הדף לראשונה. בטכנולוגיות אינטרנט אחרות, כמו ASP הקלאסית או PHP, יכולת לעקוף זאת על ידי כתיבה ידנית של קטע קוד אשר בכל בקשה מזין מראש נתונים בפקדים במצב של postback. למרבה המזל, ASP.NET מקלה בנושא זה מאוד על ידי שילוב יכולת זו כחלק מקבוצת המאפיינים של ASP.NET.

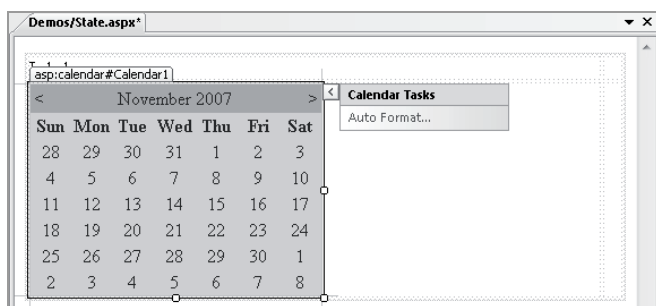
כיצד פועל מנוע המצב

מנוע המצב ב-ASP.NET מסוגל לאחסן מצב עבור פקדים רבים. הוא יכול לאחסן מצב לא רק עבור פקדי קלט של משתמש דוגמת TextBox ו-CheckBox, אלא גם עבור פקדים כמו Label ואפילו Calendar. הדבר יובהר בצורה הטובה ביותר על ידי הדגמה. התרגיל הבא מראה כיצד ליצור דף עם פקדים שמסוגלים לשמור על המצב שלהם. בסעיפים הבאים תמצא הסבר איך ASP.NET עושה זאת.

נסה זאת בחינת מנוע המצב של ASP.NET

בתרגיל זה תוסיף לדף את הפקדים Label, Button ו-Calendar. פקדים אלה משמשים להדגמת חלק מהפעילות הפנימית של ASP.NET, כולל postbacks והדרך שבה ASP.NET שומרת מצב.

1. צור דף חדש תחת התיקייה Demos וקרא לו State.aspx. ודא שהוא משתמש בקוד ברקע, ואל תשכח לבחור מהרשימה הנפתחת את שפת התכנות המועדפת עליך.
2. העבר את הדף ל- Design View, לחץ בתוך <div> המקווקו כדי להתמקד בו, ואז הוסף טבלה בת שתי שורות ושתי עמודות על ידי בחירת Table ← Insert Table מהתפריט הראשי.
3. גרור פקד Label מארגז הכלים אל התא השמאלי העליון. גרור פקד Calendar אל התא השמאלי התחתון. אם סידרת את ארגז הכלים בסדר אלפביתי בתרגיל נסה זאת קודם, Calendar הוא הפקד הרביעי בקטגוריה Standard של ארגז הכלים. אחרת תמצא את Calendar בהמשך הרשימה של הקטגוריה Standard.
4. שים לב שברגע ששחררת את Calendar בתא, הופיע לוח Smart Tasks שלו, כפי שמוצג בתרשים 4-15.

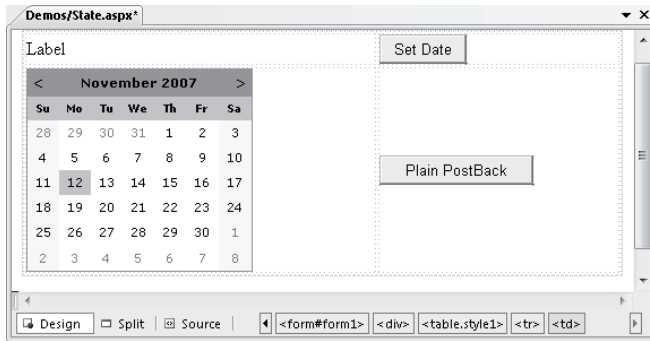


תרשים 4-15

במקרה של Calendar, לוח זה כולל אפשרות אחת בלבד, AutoFormat, אשר מאפשרת לשנות את מראה לוח השנה. לחץ על הקישור, בחר צבע מתוך סכמות הצבעים המוגדרות מראש, למשל Simple, ולחץ OK.

5. גרור פקד Button לכל אחד משני התאים של העמודה הימנית של הטבלה.

6. לחץ על Button בשורה הראשונה, ואז הקש F4 לפתיחת טבלת המאפיינים. הגדר את ID של Button ל-btnSetDate ואת המאפיין Text שלו ל-Set Date. את המאפיין ID תמצא בקצה הרשימה של המאפיינים או בתחילתה, אם הרשימה אוחסנה עם סדר אלפביתי.
7. חזור על הצעד הקודם עבור הלחצן האחר. הגדר את ID ל-btnPlainPostBack והגדר את המאפיין Text שלו ל-Plain PostBack. כשתסיים, הדף אמור להיראות ב- Design View כמו תרשים 4-16.



תרשים 4-16

8. ב- Design View, לחץ לחיצה כפולה על הלחצן Set Date, והוסף את הקוד הבא בשורה הריקה שבין שורות הקוד ש-VWD הכניסה עבורך:

VB.NET

```
Protected Sub btnSetDate_Click(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles btnSetDate.Click
    Label1.Text = DateTime.Now.ToString()
End Sub
```

C#

```
protected void btnSetDate_Click(object sender, EventArgs e)
{
    Label1.Text = DateTime.Now.ToString();
}
```

9. פתח את הדף בדפדפן על ידי הקשת Ctrl+F5. בחר תאריך בלוח השנה על ידי לחיצה על אחד מהמימים. שים לב שברגע שלחצת על התאריך, הדף נראה כאילו הוא נטען מחדש. פעולה זו נגרמת על ידי postback. תלמד יותר אודות זה בסעיף כיצד זה פועל שלאחר תרגיל זה.
10. לחץ על הלחצן Set Date כמה פעמים. גם עכשיו הדף נשלח חזרה לשרת והפקד Label מתעדכן עם התאריך והשעה הנוכחיים בכל פעם שתלחץ על הלחצן. המתן כמה שניות, ואז לחץ על הלחצן Plain PostBack. שוב מבוצעת הפעולה postback ואחריה הדף נטען מחדש.

כעת תן מבט בטקסט של Label. הוא עדיין מכיל את התאריך והשעה שהוצגו כשלחצת על הלחצן Set Date בפעם האחרונה. לחץ על הלחצן Plain PostBack כמה פעמים נוספות, ושים לב שתווית אינו משתנה.

11. חזור ל-VWD ופתח את טבלת המאפיינים של הפקד Label ב- Design View. אתר את המאפיין EnableViewState והגדר אותו כ- false על ידי בחירת הערך מהרשימה הנפתחת.
12. חזור על צעדים 9 ו-10 על ידי פתיחת הדף מחדש בדפדפן, ולחץ על לוח השנה והלחצנים. כשתלחץ הפעם על הלחצן Plain PostBack, תראה שהפקד Label מקבל את הטקסט הראשוני שלו, Label, כברירת מחדל.

כיצד זה פועל

כדי להבין כיצד כל זה פועל, יש להתבונן בכמה אלמנטים חשובים. תחילה, פתח שוב את הדף בדפדפן והתבונן במקור HTML שלו. תוכל לעשות זאת על ידי לחיצה ימנית על הדף בדפדפן, ואז בחר בפריט התפריט View Source או View Page Source. סמוך לראש החלון תראה את האלמנט `<form>` הבא:

```
<form name="form1" method="post" action="State.aspx" id="form1">  
...  
</form>
```

משתמשים באלמנט `<form>` של HTML כדי לאפשר למשתמש לשלוח מידע מהדפדפן אל השרת. משתמש יכול להכניס מידע תוך שימוש בפקדים כמו תיבות טקסט, רשימות נפתחות, תיבות סימון וכו'. טופס יכול להישלח באחת משתי דרכים: עם POST (כפי שמוצג באלמנט `<form>` לעיל), או עם GET. במקרה הראשון, כל הנתונים מהטופס מתווספים לגוף הבקשה ונשלחים אל השרת. במקרה של GET, כל הנתונים מתווספים לכתובת הממשית של הבקשה. המורכבות של ההבדלים אינה כה חשובה כרגע; אך חשוב להבין למה משמש האלמנט `<form>`: הוא עוטף פקדי טופס שערכיהם נשלחים יחד חזרה אל השרת.

כשלוחצים על פקד כמו Button, הוא יוצר postback לשרת. במשך postback זה נשלח חזרה לשרת כל המידע הרלוונטי בטופס, ובשרת הוא יכול לשמש לבנייה מחדש של הדף.

ברירת המחדל של כל טפסי web היא להשתמש תמיד ב-POST לשליחת נתונים אל השרת. בנוסף, כברירת מחדל, דף ASP.NET מכיל תמיד בדיוק טופס אחד. מכיוון שהדבר כה נפוץ, ונדרש על ידי ASP.NET כדי לפעול באופן תקין, דף חדש שנוצר על ידי VWD כבר מכיל את האלמנט `<form>` ולכן אינך צריך להוסיפו בעצמך. לבסוף, חשוב להבין שכברירת מחדל, טופס web של ASP.NET תמיד שולח חזרה אל עצמו. בסביבות אינטרנט אחרות, כמו ASP הקלאסית או PHP, אין זה נדיר להגדיר את התכונה action של הדף לדרך אחר שתפקידו לעבד את הנתונים שהמשתמש שולח. אולם, עם דפי ASP.NET תמצא שאפילו אם הגדרת את התכונה action במפורש בפקד, רכיב ASP.NET runtime יחזיר את המצב לקדמותו, כלומר לדרך הנוכחי.

הדבר הבא להתבונן בו הוא השדה ViewState שאתה רואה מוצלל בקטע קוד HTML הבא:

```
<form name="form1" method="post" action="State.aspx" id="form1">
```

```
...
```

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwULLTE5Njc4MzkzNDdkZI+1OWZMZpVv0hc7i/HFGMd008oc" />
```

```
</div>
```

למרות שבמבט ראשון הטקסט נראה כמכיל תווים אקראיים בלבד, הרי שלמעשה הוא מכיל מידע שימושי. כדי להגן על המידע שמאוחסן בשדה זה ולהקטין את גודלו, ASP.NET ממירה את מצב הדף במחרוזת שאתה רואה לעיל. אם היית מסוגל להסתכל לתוך ערכי השדות, היית מוצא ערך לפקד Label1 עם התאריך והשעה הנוכחיים.

כאשר הדף נטען, ASP.NET runtime ממלא את השדה המוסתר הזה במידע אודות הדף. לדוגמה, הוא הוסיף את הערך עבור המאפיין Text של הפקד Label1 כשגרמת ל-postback על ידי לחיצה על הלחצן Set Date, באופן דומה, השדה מכיל את היום הנבחר עבור Calendar. כשהדף מוחזר על ידי postback, הערך בשדה מוסתר זה נשלח עם הבקשה. ואז, כאשר ASP.NET יוצרת את הדף החדש בשרת, נקרא המידע מהשדה ViewState ומיושם לפקדים. בדרך זו יכול פקד כמו Label לשמור על הטקסט שלו גם לאחר שהדף מוחזר לשרת.

כדי לסכם את שנאמר, להלן סקירה מהירה של התהליך שקרה בתרגיל נסה זאת הקודם:

1. ביקשת את הדף בדפדפן על ידי פתיחתו מתוך VWD.
2. הדפדפן ביקש וקיבל את הדף מהשרת.
3. ASP.NET runtime קרא את קובץ הדף מהדיסק, עיבד אותו, ושלח את קוד HTML שנוצר אל הדפדפן. בשלב זה, כל הפקדים הוגדרו לברירות המחדל של הערכים שלהם, כפי שנקבע בסימני העריכה של הדף. לדוגמה, המאפיין Text של הפקד Label הוגדר כ-Label.
4. כשהדף הוצג בדפדפן, לחצת על הלחצן Set Date. הדבר גרם ל-postback לשרת.
5. הדף נבנה מחדש בשרת, בדומה לפעם הראשונה שבה נטען, ואשר הוצג בצעד 3. בשלב זה כל הפקדים מכילים את ערכי ברירות המחדל שלהם, כך, שוב המאפיין Text של הפקד Label1 מוגדר כ-Label. זמן קצר לאחר שנקבעו ערכי ברירות המחדל, רכיב runtime מגדיר את ערכי הפקדים שהוא מוצא ב-ViewState. עם זאת, מכיוון שזהו postback הראשון, והמאפיין Text של הפקד Label עדיין לא השתנה, הערך שלו אינו נוכח ב-ViewState ולכן המאפיין Text מכיל את ערך ברירת המחדל שלו - המילה Label.
6. בשעה שהעיבוד של הבקשה נמשך, ASP.NET runtime הפעיל את הקוד בלחצן btnSetDate_Click. כפי שראית, קוד זה שינה את המאפיין Text של הפקד Label לתאריך ולשעה הנוכחיים. ASP.NET runtime רואה את השינוי ומאחסן גם את הערך החדש הזה ב-ViewState, וכך הוא נשאר זמין ל-postbacks הבאים.
7. אחר כך לחצת על הלחצן Plain PostBack. כמו עם הלחצן האחר, גם פעולה זו גרמה ל-postback. הדף נבנה שוב, וכל ברירות המחדל נקבעו מחדש. פירוש הדבר שהמאפיין Text של הפקד Label1 מכיל כעת את המילה Label. אבל, זמן קצר לאחר

מכן, במהלך אותו מחזור עיבוד, ASP.NET runtime מעבד את מצב התצוגה, כשהוא משחזר את הפקדים שמצא בשדה המוסתר ViewState. בדוגמה שלפנינו הוא מצא את המאפיין Text עם התאריך והשעה הנוכחיים והקצה אותה שוב לפקד Label. מכיוון שהלחצן Plain PostBack אינו משנה עוד את הערך Text של Label, המאפיין Text אינו משתנה: הוא מכיל את התאריך והשעה מה-postback הקודם. בסופו של דבר, הדף כולו נשלח לדפדפן, שבו התווית מציגה כהלכה את הערך הקודם שלה.

8. לבסוף, הפסקת את פעולת ViewState עבור הפקד Label על ידי הגדרת EnableViewState כ-`false`. לאחר כיבוי הגדרה זו, רכיב ASP.NET runtime מפסיק לעקוב אחר הפקד Label. כך, כאשר תלחץ על הלחצן Plain PostBack ה-`runtime` אינו מוצא מידע כלשהו עבור התווית ב-`ViewState`, וכתוצאה, התווית מציגה את טקסט ברירת המחדל שלה - המילה `Label`.

לא כל הפקדים סומכים על ViewState

עליך לדעת שלא כל הפקדים סומכים על מצב התצוגה כל הזמן. יש כמה פקדים שמסוגלים לשמור את המצב שלהם. פקדים אלה כוללים את `TextBox`, `CheckBox`, `RadioButton` ואת `DropDownList`. הם מסוגלים לשמור את הערך שלהם, משום שהם מתורגמים בדפדפן כפקדי טופס תקינים (מסוג `Standard`) של `HTML`. לדוגמה, פקד שרת מסוג `TextBox` נראה בדפדפן בדומה לזה:

```
<input name="TextBox1" type="text" value="Initial Text" id="TextBox1" />
```

כשדף שפקד `TextBox` כזה כלול בו מוחזר, הדפדפן שולח לשרת גם את הערך שלו. ASP.NET runtime יכול כעת לבחון ערך זה כדי לאכלס את תיבת הטקסט שוב, במקום לקחת את הערך מ-`ViewState`. ברור שזה יעיל יותר מאשר אחסון הערך גם ב-`ViewState`. אם זה היה המקרה, הערך היה נשלח לשרת פעמיים: פעם אחת בתיבת הטקסט, ופעם שנייה ב-`ViewState`. עם ערכים גדולים, מהלך כזה היה מעלה את גודל הדף אשר מתבטא בסופו של דבר בזמן טעינתו.

הערה לגבי ViewState וביצועים

מכיוון שמנוע ViewState מוסיף כמות נכבדה של מידע לדף, הרי שלעתים כשאינך זקוק לו, מומלץ להפסיק את פעולתו. בדרך זו תוכל להוריד מאוד את גודל השדה המוסתר ViewState, שמקטין את גודל הדף ולפיכך משפר את זמן טעינתו לדפדפן. הפסקת ViewState היא קלה, ויכולה להתבצע כך:

⊙ **ברמת האתר.** אתה יכול לעשות זאת בקובץ `web.config` על ידי שינוי האלמנט `<pages>`, והגדרת התכונה `enableViewState` כ-`false`:

```
<pages enableViewState="false">
...
</pages>
```

אפשרות זו אינה מומלצת בדרך כלל, מכיוון שהיא עלולה להשפיע בצורה חמורה על פונקציונליות האתר - שבירת דפים שמסתמכים על ViewState.

- ⊙ **ברמת הדף.** בראש כל דף נמצאת הנחיית הדף (page directive), סדרת הוראות ל- ASP.NET runtime כיצד הדף אמור להתנהג. בהנחיית הדף אפשר להגדיר את EnableViewState כ-true, הנה כך:

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="State.aspx.vb" Inherits="Demos_State" EnableViewState="false" %>
```

אפשרות זו שימושית עבור דפים שלגביהם אתה בטוח שלא תצטרך את ViewState כלל.

- ⊙ **ברמת הפקד.** כל פקד שרת של ASP.NET מאפשר להגדיר את EnableViewState באופן פרטני. דבר זה מאפשר להפסיק את פעולת ViewState עבור פקדים מסוימים ולהשאיר פעילה עבור אחרים. שים לב שאם הפסקת את ViewState ברמת הדף או ברמת האתר, הרי שגם אם הגדרת את EnableViewState כ-true עבור פקדים מסוימים, הוא יישאר מופסק.

בהתחשב בגמישות שמקנה האפשרות האחרונה, מומלץ להשתמש בה רוב הזמן. כדי ליצור דפי אינטרנט בעלי ביצועים טובים, הרי שכברירת מחדל עליך להפסיק את פעולת ViewState בכל פקד בכל עת שניתן, ולעוררה רק כשאתה באמת זקוק לה. מאוחר יותר נחזור ונעסוק שוב בנושא ViewState של פקדים.

עצות מעשיות לגבי עבודה עם פקדים

בסעיף זה תמצא כמה עצות מעשיות לעבודה עם פקדים.

- ⊙ הקדש זמן מה לניסויים עם הסוגים השונים של פקדים. למרות שאנו משתמשים ברבים מהם בספר זה ודינים בהם, כדאי לדעת יותר איך להשתמש בהם וכיצד הם פועלים. עריכת ניסויים בכמה דפים לדוגמה, תאפשר לך יתרון לקראת השימוש בפקדים בפרקים הבאים. בגלל המורכבות, התעלם מהפקדים שאינם בקטגוריה Standard, ואשר עבורם דרוש רקע רחב יותר שיינתן לך בהמשך הלימוד בספר.
- ⊙ שקול ברצינות את הפסקת הפעולה של ViewState עבור כמה מהפקדים שלך. במקרים רבים לא תבחין בהבדל. עבור פקדים מונעי-נתונים, שבהם נדון בפרקים 12 ואילך, ביטול ה-ViewState יכול להקטין באופן משמעות את גודל דפי האינטרנט שלך, כדי לקצר את זמני הטעינה של הדפים ולשפר את חווית המשתמשים.
- ⊙ לפני שאתה מתכנן טופס web מורכב, עם מספר פקדים לקבלת קלט משתמשים, קח צעד אחורה מהמחשב, קח פיסת נייר ועט ושרטט את הפונקציונליות הנדרשת. קל יותר ליצור ממשק משתמש עקבי ומתוכנן היטב, כאשר חושבים על העיצוב (הטכני) של היישום לפני שמתחילים לגרור ולשחרר בו פקדים או לקודד אותם. אם יתברר לך

שתכנון הדף שלך היה שגוי ותצטרך להכניס בו שינויים משמעותיים, הדבר תמיד יגזול יותר זמן מאשר עשייה נכונה בפעם הראשונה.

נסה את המנגנון של ViewState כדי להבין טוב יותר כיצד הוא פועל. צור כמה דפים דומים לדף שיצרת בפרק האחרון. הפסק את פעולת ViewState ברמת הדף או הפקד, וראה איך הדף מתנהג. שים לב לפקדים כמו TextBox שמסוגלים לשמור את הערך שלהם גם כאשר ViewState אינו פעיל.

סיכום

בפרק זה הכרת מקורב קבוצה גדולה של פקדי שרת ב-ASP.NET. מכיוון שפקדים אלה הם כה חשובים, ומשמשים בכל יישום של ASP.NET, יש חשיבות רבה להכרתם והבנת דרך פעולתם. חשוב לדעת איזה פקדים זמינים עבורך בארגו הכלים ולמה הם משמשים.

לפקדי שרת רבים יש קבוצה גדולה של מאפיינים משותפים. ברגע שתבין את קבוצת הכללים וההתנהגויות הבסיסית, הדבר יקל עליך להבין את דרך הפעולה של פקדים חדשים. בדרך זו גם תוכל להגיע להיכרות מהירה עם פקדי צד שלישי אשר תוכל לטעון מהאינטרנט כרכיבים מוכנים לשימוש חוזר.

מכיוון שפקדי שרת נמצאים בשרת, עליך לציין אותם בדפי ASP.NET שלך עם המאפיין Runat=Server. בדרך זו יודע ASP.NET runtime איך להתמודד עם הפקד. במהלך העיבוד של הדף נשאל כל פקד על ידי רכיב runtime לגבי ה-HTML שלו. קוד HTML המאוחד עבור כל הפקדים בדף נשלח בסופו של דבר אל הדפדפן, שבו הוא משמש לבניית ממשק המשתמש שהמבקר באתר רואה לנגד עיניו.

עקב המספר הגדול של שרתי הפקד, הם מקובצים יחד באופן הגיוני לקטגוריות שונות בארגו הכלים של VWD. הקטגוריה Standard כוללת את הפקדים שצפוי שיהיה בהם שימוש רב יותר. קטגוריה זו כוללת פקדים נפוצים כמו Button, DropDownList, Label ו-HyperLink. עם פקדים אלה אתה יכול ליצור במהירות טפסי web בעלי פונקציונליות מלאה, שמאפשרים למשתמש להכניס נתונים בדרכים שונות ורבות, ויכולים להישלח לשרת כדי לעבור עיבוד נוסף.

אחת ההמצאות הגדולות ב-ASP.NET הוא מנוע המצב שמאפשר לפקדים לשמור את המצב שלהם על פני postbacks. מנוע המצב מביא לחיסכון ממשי בזמן, ומשחרר אותך מכתובת כמות גדולה של קוד משעמם וחדגוני בכל דף כדי לשחזר את התנהגותו. הודות למנוע המצב, פקדים כמו <asp:Wizard> קלים מאוד לשימוש. כל שעליך לעשות הוא לגרור אותם אל הדף, להגדיר מספר צעדים של האשף ולהוסיף להם פקדים, בכל השאר מטפל ASP.NET runtime. מובן מאליו שהאמור ישים לכל הפקדים, ולא רק ל-Wizard.

לפונקציונליות זו יש מחיר. מכיוון שכל הפקדים שומרים את מצבם בשדה המוסתר ViewState, הדף גדל במהירות. אם לא תיזהר, אתה יכול בקלות ליצור דפים שטעינתם תהיה איטית, והצורך שלהם לכוון עיבוד של השרת יהיה גדול מכפי שתרצה. למרבה המזל, קל לשלוט בדרך שבה משתמשים ב-ViewState. מבין שלוש האפשרויות להפסקת ViewState (ברמת היישום,

הדף או הפקד), אפשרות רמת הפקד היא ברוב המקרים גם הטובה ביותר. על ידי הפסקת ViewState ברמת הפקד אתה יכול לקבוע בדיוק איזה פקדים יתייחסו ל-ViewState ואיזה לא. הדבר מאפשר להפסיק את פעולת ViewState עבור פקדים שאינם צריכים אותה באמת.

בפרק זה גם למדת להכיר מעט קוד שנמצא בצד-שרת ב- Visual Basic .NET ו- C#. מכיוון שהדוגמאות היו שטחיות למדי, יכולת מן הסתם, להבין די טוב את הקוד שהוצג. בפרק הבא תלמד להכיר טוב יותר את התכנות של דפי ASP.NET. תלמד איך נראית שפת תכנות, איזה אלמנטים היא מכילה, ואיך אתה יכול לכתוב בעצמך קוד לשימוש בדפי ASP.NET. וטוב מכל, הדוגמאות מוצגות גם ב- Visual Basic .NET וגם ב- C#, ולכן תוכל להשתמש בשפת התכנות הרצויה לך.

תרגילים

1. תאר את המנגנון שמאפשר לפקדי שרת לשמור את מצבם.
2. כיצד מסוגל ASP.NET runtime לעקוב אחר מצב פקד בין postbacks?
3. ציין הבדל בין <asp:DropDownList> לבין <asp:ListBox>.
4. איזה מאפיין עליך לשנות כדי לגרום ל-postback אל השרת כאשר אתה משנה את מצב הסימון ב-CheckBox שבדפדפן?
5. לשרתי פקד רבים יש קבוצת מאפיינים משותפת שמשפיעה על הופעתם ב-runtime. קרא בשם של שלושה מאפיינים שמשנים אלמנטים של סגנון כמו צבע, גבולות וגודל.
6. במקום להגדיר מאפייני פקד פרטניים, כמו BackColor ו-ForeColor, מוטב להגדיר מאפיין יחיד אשר תלוי ב-CSS. מהו השם של מאפיין זה ואיזו תועלת הוא מביא לך?

פרק 5

תכנות דפי Web ב-ASP.NET

בפרקים הקודמים יצרת כמה טפסי Web שהכילו בעיקר פקדי שרת של ASP.NET ושל קוד HTML פשוט. רק דוגמאות מעטות הכילו קוד ממשי שנכתב ב- Visual Basic.NET (VB.NET) או ב- C#, ורוב הקוד הזה היה פשוט למדי. עם זאת, קידוד הוא מרכיב חשוב בכל אתר אינטרנט. אמנם, בהשוואה למה שנדרש בגרסאות הישנות ממשפחת 1.x של סביבת .NET, או בטכנולוגיות אינטרנט אחרות כמו ASP הקלאסית או PHP, הרי שהמספר הרב של פקדי השרת המתוחכמים שעומדים לרשותך מקטין את כמות הקוד שאתה צריך לכתוב, ועדיין, היכולת לקרוא, להבין ולכתוב קוד היא נכס חיוני בערכת הכלים של מפתח אינטרנט.

בפרק זה תכיר את יסודות התכנות עבור יישומי אינטרנט. תלמד בהרחבה על הנושאים הבאים:

- ⊙ איך לעבוד עם אובייקטים ואוספים (collections) בסביבת תכנות.
- ⊙ דרכים שונות לקבלת החלטות בקוד שלך.
- ⊙ האפשרויות ליצירת בלוקים של פונקציונליות לשימוש רב-פעמי.
- ⊙ דרכים שונות לכתוב קוד מסודר ומתועד.
- ⊙ מה זה תכנות מכוון עצמים וכיצד להשתמש בו לבניית היישומים שלך.

כמו כל שאר הדוגמאות בספר, גם הדוגמאות בפרק זה מובאות בשתי השפות: VB.NET ו- C#. לכל מושג או תיאוריה שמוצגים בפרק זה תראה, בו-זמנית, דוגמה ב- VB.NET ודוגמה ב- C#. בחירת השפה המועדפת עליך תלויה בכך בלבד.

כדי להשיג את המרב מהלימוד של פרק זה, מומלץ לנסות את כל, או לפחות את רוב, דוגמאות הקוד שמוצגות בפרק. קריאה והבנת קוד הן דבר אחד; לראות את הקוד פועל למעשה זה משהו שונה לחלוטין. את רוב הדוגמאות ניתן לבחון עם דף ASPX פשוט. גרור פקד Label ופקד Button אל הדף שלך, לחץ לחיצה כפולה על הלחצן בתצוגת עיצוב (Design View), ואז הקלד את הקוד הרלוונטי בשורה הפתוחה של בלוק הקוד שנוסף על ידי סביבת הפיתוח.

שים לב שכמה מהדוגמאות קוראות לקוד פיקטיבי שלא יפעל כהלכה. הן משמשות להמחשת הנושאים הנדונים בלבד.

מבוא לתכנות

כדי להתחיל בתכנות חשוב להבין תחילה את המינוח המקובל, שמשותף לכל המתכנתים בכל שפות התכנות והיישומים. בהמשך הלימוד בפרק זה תכיר מספר גדול יחסית של מונחים ומושגים. רוב המונחים מלווים בדוגמאות קוד, כדי שתוכל לראות איך משתמשים בהם בקוד.

חשוב להבין שהפרק אינו מהווה מבוא שלם לתכנות. איננו עוסקים בכל פרט של שפת תכנות זו או אחרת. במקום זאת, נתמקד במושגי המפתח שעליך להבין כדי לבנות בהצלחה אתרי אינטרנט לשימוש יום-יומי. לאחר שתבין את הנקודות העיקריות, תמצא שקל לך יותר להעמיק את הידע שלך בתכנות על ידי לימוד התכונות היותר מתקדמות של שפת התכנות המועדפת עליך.

הערה: לימוד מקיף על תכנות ב-C# בעברית, תמצא בספר **Visual C# 3.0 דונת לימוד** בהוצאת הוד-עמי.

הקוד שאתה עומד לכתוב, בפרק זה ובפרקים הבאים, ייתכן שיתווסף לקוד ברקע של דף web, או שיוצב כחלק מקובץ מסוג מחלקה שימוקם בתיקייה App_Code שנדון בה בהמשך הפרק. כאשר רכיב ASP.NET runtime מעבד בקשה לדף שמכיל קוד, הוא מהדר תחילה כל קוד שהוא מוצא בדף, בין אם זה קוד ברקע או קובץ מחלקה. קוד שעבר הידור משנה את צורתו משפת תכנות קריאה על ידי בני אדם (כמו C# או VB.NET) לקוד שהמחשב מבין ומסוגל לבצע. תוצאת תהליך הידור של אתר web בסביבת ASP.NET הוא קובץ אחד, או יותר, עם סיומת **DLL**, שמאוחסן בתיקייה זמנית במערכת שלך. תהליך הידור זה קורה רק בפעם הראשונה שהדף מבוקש לאחר שהוכנסו בו שינויים. הבקשות הבאות לאותו דף מתמלאות על ידי שימוש חוזר באותו DLL. למרבה המזל, באתרי אינטרנט בסביבת ASP.NET, ההידור מתבצע ברקע, באופן שאינך צריך לטפל בו.

בתחילה נתבונן בסוגי הנתונים ובמשתנים השונים, מכיוון שהם אבני הבניין של כל שפת תכנות. כל יישום .NET. משתמש בסוגי נתונים שונים, שלעתים קרובות מאוחסנים במשתנים. בסעיף הבא נלמד עליהם.

סוגי נתונים ומשתנים

כשאתה חושב על נתונים בפעם הראשונה, ייתכן שלא תבין שלכל יחידת נתונים יש סיווג כלשהו. אתה עשוי לחשוב שמחשב יאחסן את המילים "Hello World" בדיוק באותה צורה כמו שיאחסן את התאריך הנוכחי או את הספרה "6". אולם, כדי לעבוד עם נתונים בעילול, שפות תכנות מגדירות **סוגי נתונים** (data types) שונים, כשכל סוג נתונים מוגבל לסוג מסוים של מידע. בסביבת ASP.NET יש רשימה ארוכה של סוגי נתונים שמאפשרים לעבוד עם מספרים (כמו Integer, Short ו-Double), מחרוזות טקסט (String ו-Char), תאריכים (DateTime), ערך בוליאני true/false (Boolean) ועוד. רשימה של סוגי הנתונים הנפוצים נמצאת בהמשך הסעיף.

לכל סוג עיקרי של נתונים יש סוג נתונים מיוחד. כדי לעבוד עם סוג זה, אתה יכול לאחסן אותו במשתנה (variable) שעליך להכריז (declare) עליו כשאתה משתמש בסוג הנתונים בפעם

הראשונה. ב-VB.NET תשתמש במבנה הכרזה `Dim myVariable As DataType` (כאשר `myVariable` הינו שם המשתנה ו-`DataType` הוא סוג המשתנה), בשעה שב-C# תשתמש בהכרזה `DataType myVariable` (גם כאן, `DataType` הוא סוג המשתנה ו-`myVariable` הינו שם המשתנה). הדוגמה הבאה מראה איך להכריז על שני משתנים: אחד לאחסון מספר והשני לאחסון מחרוזת או קטע טקסט.

VB.NET

```
' הכרזה על סוג משתנה מספרי שלם בטווח מספרים בינוני בשם: distanceInMiles
Dim distanceInMiles As Integer
```

```
' הכרזה על סוג משתנה מחרוזת בשם: firstName
Dim firstName As String
```

C#

```
// הכרזה על סוג משתנה מספרי שלם בטווח מספרים בינוני בשם: distanceInMiles
int distanceInMiles;
```

```
// הכרזה על סוג משתנה מחרוזת בשם: firstName
string firstName;
```

לאחר ההכרזה על משתנה, תוכל להקצות לו ערך. סוגים כמו מספרים עשרוניים ובוליאניים תוכל להקצות ישירות למשתנה. כדי להקצות מחרוזת (String) למשתנה עליך להקיף אותה בגרשיים ("").

VB.NET

```
Dim distanceInMiles As Integer
distanceInMiles = 437
```

```
Dim firstName As String
firstName = "Sara"
```

C#

```
int distanceInMiles;
distanceInMiles = 437;
```

```
string firstName;
firstName = "Sara";
```

במקום הכרזה והקצאת ערך בשתי פעולות נפרדות כמו בדוגמאות הקודמות, באפשרותך להכריז ומייד להקצות ערך בהוראה אחת בלבד. ראה דוגמאות אלו:

VB.NET

```
Dim distanceInMiles As Integer = 437
```

C#

```
int distanceInMiles = 437;
```

למרות ששם משתנה יכול להיות כל שם שעולה בדעתך, מומלץ לקבוע לכל משתנה שם בעל משמעות שמתאר את מטרותו. לדוגמה, מחרוזת שאמורה להחזיק שם פרטי תיקרא `firstName`, ומשתנה שמחזיק את הגיל ייקרא `age`. כדי לעזור לך מאוחר יותר למצוא בקוד את סוג המשתנה, הרי ש-VWD וכל הכלים האחרים ב- Visual Studio, מציגים תיאור של סוג המשתנה כשאתה עובר על פני משתנה בעורך הקוד. דבר זה הופך את זיהוי סוג המשתנה לקל ביותר. תרשים 5-1 מראה שהמשתנה `distanceInMiles` בדוגמה של C# הוא מסוג `int`.

```

19 // Declare a variable of type int to hold medium sized whole numbers.
20 int distanceInMiles;
21
22 // Declare a variable to hold some text like a first name.
23 string firstName;
24
25 // Assign distanceInMiles a value
26 distanceInMiles = 437;
27 (local variable) int distanceInMiles
28

```

תרשים 5-1

מומלץ לא להוסיף לשם המשתנה קידומת של כמה אותיות, כדי לציין את סוגו. לדוגמה, קרא למשתנה מחרוזת מסוג `string` המחזיק שם: `firstName` ולא `sFirstName`, ההמלצה היא כדי שהקוד שלך יהיה קל יותר לקריאה (`age` יותר קריא מאשר `iAge`) וקל יותר לתחזוקה.

סביבת .NET של מיקרוסופט (ולכן גם סביבת ASP.NET) תומכת במספר גדול של שפות תכנות שונות, ובכללן VB.NET, C# ואחרות. כל השפות האלו מסוגלות לתקשר זו עם זו. לדוגמה, אתה יכול לכתוב קוד ב-C#, להשתמש ב- Visual C# Express Edition כדי להדר אותו לקובץ `dll`, ואז להשתמש בקובץ זה ביישום `web` שהשפה העיקרית שלו היא VB.NET. יכולת זו קיימת הודות למנגנון המאפשר לכל שפות התכנות בסביבת .NET. להבין זו את זו. מנגנון זה קרוי **CTS** (Common Type System) ותפקידו להגדיר את סוגי הנתונים שנגישים לכל השפות שתואמות לה. כל שפה חופשית להגדיר סוגים שונים שהם, ביסודו של דבר, קיצורי דרך או שמות נרדפים לסוגי התיאור הקיימים בסביבת .NET. כך, לדוגמה אם CTS מגדירה סוג בשם `System.Int32`, שפה כמו C# תעדיף לקרוא לסוג זה `int` כדי להקל על מפתח התוכנה להשתמש בו.

הטבלה הבאה מציגה את סוגי המשתנים הנפוצים ביותר בסביבת .NET. המוגדרים על ידי CTS, ואת השמות הנרדפים שבהם משתמשות C# ו-VB.NET. הטבלה מפרטת גם את טווח המשתנים ומסבירה למה הם משמשים.

תיאור סוג המשתנה	VB.NET	C#	.NET
אחסון של כמות מספרים קטנה, בעלי ערכים חיוביים מ-0 עד 255. כאשר לא מוקצה כל ערך במפורש, ברירת המחדל היא 0.	Byte	byte	System.Byte
אחסון של מספרים שלמים שערכיהם בין -32,768 ל-32,767. ברירת המחדל היא 0.	Short	short	System.Int16

.NET	C#	VB.NET	תיאור סוג המשתנה
System.Int32	int	Integer	אחסון של מספרים שלמים שערכיהם בין -2,147,483,648 לבין 2,147,483,647. ברירת המחדל היא 0.
System.Int64	long	Long	אחסון של מספרים שלמים שערכיהם בין -9,223,372,036,854,775,808 לבין 9,223,372,036,854,775,807. ברירת המחדל היא 0.
System.Single	float	Single	אחסון של מספרים שלמים שערכיהם בין -3.4028235E+38 לבין 3.4028235E+38. ברירת המחדל היא 0.
System.Double	double	Double	אחסון של מספרים לא שלמים. הערכים בעלי השברים אינם מדויקים כמו הערכים השלמים, אולם כאשר אין צורך בדיוק מרבי קיצוני, עדיף להשתמש בסוג Double על פני Decimal, מכיוון שהסוג Double מהיר יותר בחישובים. ברירת המחדל היא 0.0.
System.Decimal	decimal	Decimal	אחסון של מספרים לא שלמים פשוטים במידת דיוק גבוהה. ברירת המחדל היא 0.
System.Boolean	bool	Boolean	אחסון של ערכים בוליאניים: True (ב-VB), true (ב-C#) או False (ב-VB), false (ב-C#).
System.DateTime	n/a	Date	ב- VB.NET יש שם נרדף (alias) עבור System.DateTime לאחסון נתוני תאריך וזמן. C# אינה מגדירה שם נרדף עבור סוג נתונים זה, ולכן השתמש בשם System.DateTime בלבד. ברירת המחדל היא: 1/1/0001: 12:00 am
System.Char	char	Char	אחסון תו יחיד. ברירת המחדל ב-VB היא Nothing, או null ב-C#.
System.String	string	String	אחסון מחרוזת טקסט באורך עד 2 מיליון תווים. ברירת המחדל ב-VB היא Nothing, או null ב-C#.
System.SByte	sbyte	SByte	אחסון מספרים בערכים נמוכים, בטווח -128 עד 127. ברירת המחדל היא 0.

תיאור סוג המשתנה	VB.NET	C#	.NET
דומה לסוג הנתונים System.Int16, אולם סוג זה יכול לאחסן ערכים ללא סימן מינוס (-), בין 0 לבין 65,535. ברירת המחדל היא 0. סוגי הנתונים האחרים בעלי התחילית U גם הם ערכים ללא סימן.	UShort	ushort	System.UInt16
יכולת אחסון של מספרים שלמים שערכיהם בין 0 לבין 4,294,967,295. ברירת המחדל היא 0.	UInteger	uint	System.UInt32
יכולת אחסון של מספרים שלמים שערכיהם בין 0 לבין 18,446,744,073,709,551,615. ברירת המחדל היא 0.	ULong	ulong	System.UInt64
זהו סוג הראשי, האב של כל הסוגים המקובלים ב-.NET. כולל את סוגי CTS שאתה עשוי להגדיר בעצמך. כל סוג נתונים הוא גם אובייקט, כפי שתלמד מאוחר יותר בספר זה. ברירת המחדל ב-VB היא Nothing, או null ב-C#.	Object	object	System.Object

לעתים צריך להמיר נתונים מסוג אחד לאחר. לדוגמה, ייתכן שתקבל ערך Integer ממקור מסוים, שאתה צריך לפנות אליו כ-Double. תוכל לעשות זאת בכמה דרכים.

המרת סוגי נתונים

הדרך הנפוצה ביותר להמיר סוג נתונים אחד לאחר היא להמיר אותו ל-String. יישומי אינטרנט משתמשים בסוגי מחרוזות בכל מקום. לדוגמה, נתון Text שמוחזר מ-TextBox הוא String, וכך גם SelectedValue של DropDownList הוא מחרוזת. כדי להמיר Object ל-String אתה יכול לקרוא לשיטה ToString() שלו. כל אובייקט בעולם של .NET תומך בשיטה זו, אם כי ההתנהגות המדויקת יכולה להיות שונה בכל אובייקט. בשלב זה חשוב להבין ש-ToString() (שיטה (method) - או פעולה - על אובייקט כמו String או Double, ואפילו על אובייקט האב Object עצמו. בהמשך הפרק, כאשר נדון בנושא תכנות מכוון אובייקטים (object-oriented), תלמד יותר על שיטות ואובייקטים.

השימוש ב-ToString() קל ופשוט, כפי שמדגימה הדוגמה הבאה שכותבת את התאריך והשעה הנוכחיים למשתנה Label.

הערה: זכור שכדי לנסות את דוגמאות הקוד עליך ליצור דף חדש בתיקייה Demos, להוסיף Label ו-Button, ללחוץ לחיצה כפולה על הלחצן ב-Design View, להוסיף את הקוד הבא, ואז להקיש Ctrl+F5.

לאחר שתסתיים טעינת הדף, לחץ על הלחצן והקוד יתבצע. אתה צריך את הפקד Label מכיוון שכמה מדוגמאות הקוד כותבות מעט טקסט לדף.

VB.NET

```
Label1.Text = System.DateTime.Now.ToString()
```

C#

```
Label1.Text = System.DateTime.Now.ToString();
```

דרך אחרת להמיר סוגי נתונים היא שימוש במחלקה Convert. מחלקה זו מכילה פונקציונליות מתאימה להמרת סוגי נתונים רבים לסוגים אחרים. להלן דוגמה פשוטה של המרת String שמכיל ערך שנראה כמו Boolean לסוג Boolean אמיתי:

VB.NET

```
Dim myBoolean1 As Boolean = Convert.ToBoolean("True")           ' מחזיר ערך True  
Dim myBoolean2 As Boolean = Convert.ToBoolean("False")          ' מחזיר ערך False
```

C#

```
bool myBoolean1 = Convert.ToBoolean("True");                     // מחזיר ערך true  
bool myBoolean2 = Convert.ToBoolean("False");                     // מחזיר ערך false
```

דרך נוספת להמיר סוג אחד לאחר היא על ידי שימוש ב-casting. עם casting אתה למעשה כופה סוג מסוים להיות לאחר; זאת בשונה מהמרה, שבה הערך היסודי של סוג נתונים עובר שינוי לערך חדש.

casting פועלת רק על סוגים תואמים. לדוגמה, אינך יכול לבצע casting של DateTime לתוך Integer. אתה כן יכול לבצע casting בין סוגים דומים, כמו Double לתוך Integer או String לתוך Object. ההיפוך של הדוגמה האחרונה אינו תמיד נכון. מוקדם יותר אמרתי שכל סוג נתונים בסביבת NET מבוסס על סוג הנתונים Object. פירוש הדבר הוא, לדוגמה, ש-String הוא Object. עם זאת, לא כל Object הוא גם String. זכור זאת כשאתה מנסה לבצע casting של סוג אחד לאחר ומקבל שגיאת הידור. בפרקים מאוחרים יותר בספר תלמד כיצד לבצע casting של סוגים תואמים זה לזה.

VB.NET מעמידה לרשותך כמה דרכים לביצוע casting של סוג אחד לאחר. ראשית, אתה יכול להשתמש בשיטות CType ו- DirectCast. שימוש ב-CType תאפשר לך גמישות רבה יותר בכך שהיא מאפשרת לך לבצע casting בין שני אובייקטים שנראים דומים. לעומת זאת, DirectCast מאפשרת להמיר רק בין סוגים תואמים, אבל ביצועיה טובים יותר במידת מה. הדוגמה הבאה, ב-VB.NET, מראה כיצד זה נעשה:

```
Dim o1 As Object = 1  
Dim i1 As Integer = DirectCast(o1, Integer)           ' יעבוד, כיוון שמשתנה o1 הינו מסוג Integer  
Dim i2 As Integer = CType(o1, Integer)                 ' יעבוד, כיוון שמשתנה o1 הינו מסוג Integer  
  
Dim o2 As Double = 1  
Dim i3 As Integer = DirectCast(o2, Integer)           ' יכשל, כיוון שמשתנה o2 הינו מסוג Double  
Dim i4 As Integer = CType(o2, Integer)                 ' יעבוד, כיוון שמשתנה o2 מחזיק ערך שנראה כמו ערך מסוג Integer
```

בחלק הראשון של הדוגמה, מוכרו אובייקט בשם o1 ומוקצה לו הסוג Integer וערך 1. למרות ש-o1 נראה כאובייקט, הערך היסודי שלו הוא Integer. כשקוראים ל-DirectCast, פעולת casting מצליחה, מכיוון ש-o1, אחרי הכל, הוא למעשה Integer.

בחלק השני של הדוגמה, מוכרו אובייקט בשם o2 כ-Double, זהו סוג נומרי שנראה דומה מעט ל-Integer אבל אינו ממש כזה. במקרה זה נכשלת הקריאה ל-DirectCast מכיוון שלא ניתן לבצע casting של Double ל-Integer. לעומת זאת, CType תפעל היטב מכיוון שהערכים היסודיים של o2 נראים כ-Integer ולכן ניתן לבצע casting שלהם ל-Integer.

האפשרות השלישית לביצוע casting ב-VB.NET היא להשתמש במילת המפתח TryCast, שדומה במקצת לשתי האפשרויות הקודמות. כאשר לא ניתן לבצע casting של אובייקט בצורה נכונה, TryCast מחזירה Nothing, בעוד DirectCast ו-CType גורמות לכישלון של הקוד.

C# מציעה שתי דרכים לביצוע casting של אובייקטים. הדרך הנפוצה ביותר היא לשים את סוג הנתונים בסוגריים בחזית הביטוי שברצונך לעצב:

```
object o1 = 1;
int i1 = (int)o1; // עובד

double o2 = 1;
int i2 = (int)o2; // עובד
```

לחילופין, אתה יכול להשתמש במילת המפתח as, שפעולתה דומה לזו של TryCast ב-VB.NET, במובן זה שהקוד אינו נכשל אם ה-casting אינו מצליח. דוגמת הקוד הבאה מראה שאינך יכול להמיר Integer ל-ArrayList (שאותו תפגוש מאוחר יותר בפרק). במקום להיכשל, המשתנה myList מכיל את הערך null.

```
object o1 = 1;
ArrayList myList = o1 as ArrayList; // הנתון אינו מוחלף, אך גם לא תתקבל שגיאה
```

בפרקים הבאים בספר נמשיך ונפגוש בהמרה והחלפה.

שימוש במערכים (Arrays) ואוספים (Collections)

סוגי הנתונים שפגשת עד כה היו ברורים וחד משמעיים באופן יחסי, ומורכבים מאובייקט אחד. לדוגמה, אחסנת ערך של True או False בסוג Boolean, ואחסנת מספר כמו 123 בסוג Integer. אבל מה יקרה אם יתעורר הצורך לאחסן מספרים שלמים רבים, כקבוצה? צורך כזה מתעורר, למשל, כאשר רוצים לאחסן את ציון הקודקודים של צורה גרפית מורכבת כגון מצולע. ומה יקרה כשתרצה לאחסן במשתנה אחד את התפקידים ושמות המנהלים בארגון שלך, כדי שתוכל להציגם בדף אינטרנט? זה המקום שבו מערכים ואוספים באים לעזרתך.

הגדרה ועבודה עם מערכים (Arrays)

אתה יכול לחשוב על מערך כעל שק גדול, תיבה, או רשימה שמכילים עצמים או דברים מאותו הסוג. סוג הנתונים של כל פריט במערך מוגדר כאשר אתה מכריז על המערך. כל פריט במערך

מזוהה על ידי האינדקס שלו, שמתחיל מ-0, ולכן אומרים שהמערכים הם מבוססי-אפס. ב-VB.NET, כשאתה מכריז על מערך וניגש אליו אתה משתמש בסוגריים רגילים. ב-C# אתה משתמש בסוגריים מרובעים. לאחר הגדרת המערך ומילוי ערכים בפריטים, אלמנטים או איברים (כך נכנה זאת בהמשך) שנמצאים בו, אתה יכול לגשת אל איברים אלה באמצעות האינדקס של כל איבר (0, 1, 2, וכו').

בקטע הקוד הבא מוגדר מערך בשם roles שיכול להחזיק בו-זמנית עד שני איברים, עבור שני תפקידים:

VB.NET

```
Dim roles(1) As String
```

C#

```
string[] roles = new string[2];
```

האם אתה מבחין בהבדל בין הדוגמאות ב-VB.NET וב-C#? אין כאן שגיאת דפוס. בקוד VB.NET אתה מגדיר את גודל המערך על ידי ציון חסם עליון (upper bound). החסם העליון הוא האיבר האחרון במערך שיש לך גישה אליו. מכיוון שמערכים הם מבוססי-אפס (כלומר, אתה פונה לאיבר הראשון במערך עם אינדקס 0), הרי שאם יש צורך במקום לשני איברים, החסם העליון הוא 1, ולרשותך יש איברים 0 ו-1.

מצד שני, ב-C# אינך מגדיר חסם עליון, אלא מגדיר את גודל המערך, או מספר האיברים שלו. לפיכך, ב-C# אתה מציין 2 כדי לקבל מערך עם שני איברים.

בנוסף, C# מחייבת אותך להשתמש במילת המפתח New, אשר יוצרת עבורך מערך חדש. VB.NET עושה זאת באופן אוטומטי, ואם תוסיף בה את מילת המפתח New, כמו בדוגמת C#, תקבל שגיאה. מאוחר יותר בפרק תפגוש שוב את מילת המפתח new (ב-VB).

כדי להכניס למערך את שמות התפקידים, השתמש בתחביר הבא:

VB.NET

```
roles(0) = "Administrators"  
roles(1) = "ContentManagers"
```

C#

```
roles[0] = "Administrators";  
roles[1] = "ContentManagers";
```

כדי לפנות אל האיברים במערך אתה משתמש בסוגריים, כמו בהכרזת המערך. סוגריים רגילים ב-VB.NET וסוגריים מרובעים ב-C#. שים לב ש- roles[0] מתייחס לאיבר הראשון במערך ו- roles[1] מתייחס לשני.

מעצם תכנונם, יש למערכים גודל קבוע. לפיכך, לפי הדוגמה הקודמת הוגדר מערך שיש בו שני איברים, ולכן כתיבת הקוד הבא תגרום לשגיאה:

VB.NET

```
roles(2) = "Members" ' שגיאה
```

C#

```
roles[2] = "Members"; // שגיאה
```

קוד זה מנסה להכניס למערך תפקיד שלישי, אל האיבר השלישי שלא קיים, כי הוגדרו רק שניים. מובן מאליו שהפעולה לא תצליח ותקבל הודעת שגיאה שאומרת "Index was outside the bounds of the array". מתעוררת שאלה: מה יקרה אם תצטרך יותר איברים במערך בשלב מאוחר יותר, בזמן ביצוע? ב-VB.NET הדבר קל למדי, אתה יכול להשתמש במשפט ReDim:

```
ReDim Preserve roles(2)
roles(2) = "Members" ' יעבוד ללא בעיה כעת
```

שורת קוד זו משנה את ממדי המערך אל הגודל המבוקש, ובמקרה זה, חסם עליון 2, ועל ידי כך יש מקום לשלושה איברים. מילת המפתח Preserve נחוצה כדי להשאיר את הפריטים הנוכחיים במערך ללא שינוי. בלעדיה המערך המוגדל לא ישמור על ערכיו והמערך יהיה ריק.

ב-C# נדרשת מעט יותר עבודה. בשפה זו עליך ליצור מערך חדש בעל הגודל המבוקש, להעתיק את האיברים הקיימים מהמערך הישן לתוך החדש, ורק אז תוכל להעביר את האיבר השלישי החדש אל המערך:

```
string[] tempRoles = new string[3]; // יצירת מערך זמני חדש עם הגודל המבוקש

Array.Copy(roles, tempRoles, roles.Length); // העתקת נתונים למערך הזמני
roles = tempRoles; // השמת המערך הזמני למשתנה מערך הקיים
roles[2] = "Members"; // הכנסת ערך למערך באיבר השלישי
```

במקום ליצור מערך חדש ולהעתיק אליו את האיברים, ביכולתך להשתמש ב-Resize עם השימוש ב-**generics**, שעליה תלמד יותר בסעיפים הבאים.

לאמיצים, הנה קטע קוד שמשתמש ב-Resize ומשיג אותה תוצאה כמו קטע הקוד הקודם:

```
Array.Resize<string>(ref roles, 3); // שינוי גודל המערך לשלושה איברים

roles[2] = "Members"; // הכנסת ערך למערך באיבר השלישי
```

אל תדאג עכשיו לגבי תחביר זה. כנראה שלא תזדקק לו לעתים קרובות, מכיוון שסביבת .NET מציעה חלופות למערכים בעלי גודל קבוע.

כשתתחיל לעבוד עם מערכים תמצא שהשימוש בהם בזמן ביצוע מהיר, אבל הפונקציונליות שלהם לוקה בחסר. לדוגמה, לא קל להוסיף איברים חדשים או להסיר איברים קיימים מהמערך. למרבה המזל, סביבת .NET מציעה מגוון אוספים שימושיים שנותנים את הפונקציונליות שאתה זקוק לה.

הגדרה ועבודה עם אוספים (Collections)

אוספים דומים למערכים, בכך שהם מאפשרים לאחסן יותר מאובייקט אחד במשתנה יחיד. האוספים מאפשרים לפעול בדרך שונה עם הנתונים שמאוחסנים בהם. במקום להכניס כל נתון לפי האינדקס שלו באופן ישיר, רוב האוספים משתמשים בשיטה Add שמאפשרת להוסיף פריט לאוסף. באופן דומה האוספים משתמשים בשיטות Clear ו-Remove כדי להסיר פריט, או כמה פריטים, מהאוסף. אוספים, בדיוק כמו מערכים, מאפשרים לך לבצע לולאה על פניהם כדי לקבל מתוכם את הפריטים הנחוצים.

כשאוספים הוצגו לראשונה בסביבת NET 1.0, האובייקטים ArrayList ו-Hashtable היו פופולריים עקב קלות השימוש בהם. האוסף ArrayList מאפשר להוסיף אובייקטים באופן שרירותי לאחסנה לפי סדר הוספתם, בשעה שהאוסף Hashtable מאפשר לאחסן אובייקטים שההתייחסות אליהם היא באמצעות מפתח מותאם. התועלת העיקרית של אוספים אלה לעומת המערכים היא בכך שהם יכולים לגדול בהתאם לביקוש. בשונה מהדוגמה הקודמת, שבה היית צריך לשנות את גודל המערך כדי ליצור מקום לתפקיד השלישי, האוסף ArrayList גדל באופן דינמי כשהדבר נדרש. הדוגמה הבאה מראה כיצד זה פועל:

VB.NET

Dim roles As New ArrayList()	' הגדרת אוסף (ללא ציון גודל)
roles.Add("Administrators")	' הוספת התפקיד הראשון
roles.Add("ContentManagers")	' הוספת תפקיד נוסף
roles.Add("Members")	' האוסף ימשיך לגדול לפי הצורך

C#

ArrayList roles = new ArrayList();	// הגדרת אוסף (ללא ציון גודל)
roles.Add("Administrators");	// הוספת התפקיד הראשון
roles.Add("ContentManagers");	// הוספת תפקיד נוסף
roles.Add("Members");	// האוסף ימשיך לגדול לפי הצורך

מכיוון שקוד זה קורא לשיטה (Add) במקום להקצות פריט לאינדקס מוגדר מראש במערך, יש להשתמש בסוגריים, הן ב-VB.NET והן ב-C#. בהמשך הפרק נדון בשימוש בשיטות.

אוספים פתרו אמנם כמה מהבעיות שהיו למערכים, אבל הם הציגו כמה בעיות משל עצמם. החיסרון הגדול ביותר של ArrayList הוא בכך שאינו קשיח מבחינת סוגי האובייקטים שנכנסים אליו. כלומר, על ידי השיטה Add אתה יכול להכניס לרשימה כל אובייקט. פירוש הדבר ש-ArrayList יכול להחזיק בו-זמנית אובייקטים מסוגים שונים. בהתחלה זה עשוי להיראות עניין חסר חשיבות, אבל כרגע שתתחיל לעבוד עם ArrayList שכולל אובייקטים מסוגים שונים, תיווכח לדעת שהדבר גורר בעיות רבות. הבה נתבונן שוב בדוגמת התפקידים בארגון. הן בגרסת המערך והן בגרסת ה-ArrayList הקוד הוסיף כמה מחרוזות שהכילו את שמות התפקידים. אתה יכול להשתמש בשלוש מחרוזות אלו, נאמר, לשם בניית רשימה נפתחת בטופס web כדי לאפשר למשתמש לבחור תפקיד רצוי. עד כאן אין בעיה. אבל מה יקרה אם

אחד הפריטים ברשימה אינו מחרוזת? מה יקרה אם מתכנת אחר יכתוב בטעות קוד שמוסיף פקד DropDownList ל-ArrayList? מכיוון שה-ArrayList מקבל את כל סוגי האובייקטים, הוא לא יפיק הודעת שגיאה או הודעת התראה. עם זאת, הקוד שלך יקרוס אם הוא מצפה ל-String ומקבל במקום זאת פקד DropDownList.

generics הוצגו על ידי מיקרוסופט ב- .NET 2.0, ותבניות הקוד generics הן עדיין בעלות נוכחות חזקה בגרסה .NET 3.5. הן אפשרו להתגבר על הבעיות שאוספים הציגו באי יכולתם להקפיד על סוגי הנתונים שמאוחסנים בהם, כמו לדוגמה האוסף ArrayList.

מבוא לתבניות הקוד Generics

מאז הצגתם ב- .NET 2.0 צצו תבניות קוד מסוג generics במקומות שונים בסביבת .NET. למרות שלעיתים קרובות משתמשים בהם כפי שמשתמשים באוספים, התבניות הגנריות אינן מוגבלות לאוספים, וניתן להשתמש בהן גם עם אובייקטים רגילים.

אפשר לדמות שימוש ב-generics בקוד לתפקידן של תבניות עריכה (templates) של Word בעיבוד תמלילים. הן מאפשרות לכתוב תבנית קוד שיכולה לשמש בתסריטים שונים, עם סוגים שונים של אובייקטים. עם generics ניתן להגדיר תבנית קוד שאינה מציינת את סוג הנתון במפורש. אתה מגדיר את הסוג רק בזמן השימוש בקוד זה. התועלת העיקרית בכך היא שאתה יכול לחזור ולהשתמש באותה תבנית פעם ועוד פעם, עבור סוגים שונים של נתונים, מבלי שתצטרך לחזור ולהקליד ולתחזק מספר גרסאות של הקוד. פרט לשימוש בתבניות קוד אלו בהגדרות הקוד שלך, תמצא בסביבת .NET אובייקטים ואוספים רבים שהם תבניות קוד generics ומוכנים לשימוש על ידי הקוד שלך.

כדי להבין כיצד תוכל להפיק יתרון מהתבניות הגנריות, הבט בדוגמה הבאה: זהו למעשה אותו הקוד שראית קודם בעת השימוש ב-ArrayList, אבל הפעם הסוג של הרשימה מוגבל לקבלת מחרוזות בלבד.

VB.NET

```
Dim roles As New List(Of String)

roles.Add("Administrators")
roles.Add("ContentManagers")
roles.Add("Members")
```

C#

```
List<string> roles = new List<string>();

roles.Add("Administrators");
roles.Add("ContentManagers");
roles.Add("Members");
```

כתוצאה מהשימוש בהגדרות List (Of String) ב-VB.NET, או List<string> ב-C#, הרשימה החדשה מוגדרת כעת באופן שמאפשר להוסיף לה מחרוזות בלבד, כתוצאה מהשימוש בשיטה Add. שורת הקוד הבאה נכונה מבחינת המהדר:

```
roles.Add("Administrators");
```

מאידך, הקוד הבא ייכשל עוד בזמן הידור התוכנית, משום שהערך 33 אינו מחרוזת:

```
roles.Add(33);
```

על ידי שימוש ב-generics תוכל אף ליצור רשימות שיכילו סוגים אחרים. לדוגמה:

VB.NET

Dim intList As New List(Of Integer)	' רשימה המכילה מספרים בלבד
Dim boolList As New List(Of Boolean)	' רשימה המכילה ערכים בוליאניים בלבד
Dim buttonList As New List (Of Button)	' רשימה המכילה פקדי לחצן בלבד

C#

List<int> intList = new List<int>();	// רשימה המכילה מספרים בלבד
List<bool> boolList = new List<bool>();	// רשימה המכילה ערכים בוליאניים בלבד
List<Button> buttonList = new List<Button>();	// רשימה המכילה פקדי לחצן בלבד

הערה: הנושא generics רחב בהרבה ממה שהראינו כאן, וראוי לספר משל עצמו. חומר רב תוכל למצוא ברחבי הרשת, וכן בספר **Visual C# 3.0** דגנית לימוד שם מוקדש פרק שלם לנושא זה.

דוגמאות השימוש ב-generics שראית לעיל רק חשפו מעט מהאפשרויות הטמונות ב-generics. בכל אופן, כשבונים אתרי אינטרנט בסביבת ASP.NET, בדרך כלל אין צורך בכל החומר המתקדם שמציעה generics. האוסף List הוא שימושי מאוד ולכן יש לדון בו כאן. ללא ספק תשתמש באוסף זה בקוד שלך בצורה כלשהי.

משפטים (Statements)

כדי לגרום לתוכנית או אתר web לעשות משהו שימושי, אתה צריך לספק להם **משפטים** (Statements) לביצוע. משפטים עוסקים במגוון רחב של פעולות, כמו: הצגת לחצן, שליחת דואר אלקטרוני, ביצוע קוד מסוים בעקבות פעילות של המשתמש וכו'. עם זאת, ביצוע פשוט של פעולות אינו מספיק, כי לעתים קרובות צריך להתנות אותן **בתנאים** (conditions) מסוימים שמתקיימים או שאינם מתקיימים. לדוגמה, אם המבקר באתר למסחר אלקטרוני קונה סחורות בשווי של יותר מ-\$100 בביקור אחד, הוא עשוי לקבל הנחה של 10%, אחרת הוא ישלם את מלוא המחיר. תנאים או החלטות הם על כן משפטים חשובים מאוד בשפת תכנות. קבוצה אחרת של משפטים חשובים היא **לולאות** (loops) מאפשרות לחזור על ביצוע קטע קוד מסוים מספר פעמים. לדוגמה, יכולה להיות לולאה שתתבצע 10 פעמים, ובכל מעבר (איטרציה) היא תבצע פעולה מסוימת. יכולה להיות לולאה שמאפשרת לאסוף מוצרים לסל קניות, ובסוף תסכם את מחיר כל הפריטים שבסל.

הקבוצה האחרונה של משפטים חשובים היא האופרטורים. **אופרטורים** (operators) מאפשרים לעשות משהו עם הערכים או, ליתר דיוק, לפעול על הערכים. לדוגמה, תשתמש באופרטורים כדי לחסר ערכים, לכפול, לשרשר מחרוזות, או להשוות ביניהם.

אופרטורים (Operators)

את האופרטורים החשובים ביותר ניתן לקבץ באופן לוגי לחמישה סוגים שונים. מבין חמש קבוצות אלו, הקלים ביותר להבנה ושימוש הם אופרטורי הצבה.

אופרטורי הצבה

אופרטורי הצבה (assignment operators) משמשים להקצאת ערך למשתנה. ערך זה יכול להתקבל ממקורות רבים: ערך קבוע, כמו המספר 6, הערך של משתנה אחר, או התוצאה של ביטוי או פונקציה, שבהם נדון מאוחר יותר. בצורתו הפשוטה ביותר נראה אופרטור הצבה כך:

VB.NET

```
Dim age As Integer = 36
```

C#

```
int age = 36;
```

מה יקרה אם האדם שהמשתנה age מתייחס אליו בדיוק חגג את יום הולדתו? תצטרך להוסיף 1 לערך של age. זה המקום שבו דרושים אופרטורים אריתמטיים.

אופרטורים אריתמטיים

אופרטורים אריתמטיים (arithmetic operators) מאפשרים לבצע את מרבית החישובים המוכרים (כמו חיבור, חיסור וחילוק), על משתנים וערכים. בטבלה הבאה תמצא את רשימת האופרטורים האריתמטיים הנפוצים עבור VB.NET ו-C#.

VB.NET	C#	תיאור
+	+	מחבר בין ערכים. ערכים אלה יכולים להיות מסוגים נומריים כמו Int32, במקרה של מחרוזות (String) הפעולה תשרשר את הערכים.
-	-	מחסיר ערך אחד מאחר.
*	*	מכפיל שני ערכים.
/	/	מחלק שני ערכים.
\	n/a	מחלק שני ערכים, אבל תמיד מחזיר מספר שלם מעוגל.
^	n/a	מעלה ערך אחד בחזקת האחר.
Mod	%	אופרטור השארית - מחלק שני מספרים שלמים ומחזיר את השארית.

חמשת האופרטורים הראשונים נראים מוכרים, והשימוש בהם ברור למדי. קטע הקוד הבא מראה את הפעולות הבסיסיות שניתן לבצע עם אופרטורים אלה:

VB.NET

```
Dim firstNumber As Integer = 100
Dim secondNumber As Single = 23.5
Dim result As Double = 0

result = firstNumber + secondNumber      ' Results in 123.5
result = firstNumber - secondNumber      ' Results in 76.5
result = firstNumber * secondNumber      ' Results in 2350
result = firstNumber / secondNumber      ' Results in 4.25531914893617
result = firstNumber \ secondNumber      ' Results in 4
```

C#

```
int firstNumber = 100;
float secondNumber = 23.5F;
double result = 0;

result = firstNumber + secondNumber;    // Results in 123.5
result = firstNumber - secondNumber;    // Results in 76.5
result = firstNumber * secondNumber;    // Results in 2350
result = firstNumber / secondNumber;    // Results in 4.25531914893617
```

VB.NET תומכת באופרטור "י" (לוכסן הפוך). שים לב שהלוכסן "/" מיועד לביצוע פעולת חילוק רגילה, אבל הלוכסן "י" מאפשר לקבל כתוצאה ערך שלם מעוגל (ללא שארית או שבר עשרוני). פירוש ה"עיגול" במקרה זה הוא למעשה ביטול השארית והצגת המספר השלם בלבד. ל-C# אין אופרטור ייעודי למטרה זו. עם זאת, כשתנסה לחלק שני מספרים שלמים, התוצאה תהיה תמיד מספר שלם. פירוש הדבר שהתוצאה של 7 (מאוחסן כ-int) מחולק ב-2 (מאוחסן כ-int) היא 3. חשוב להבין שעיגול זה של התוצאה מתרחש, כדי שלא תסיים את החישוב עם תוצאות בלתי צפויות.

שים לב שבדוגמה של C# עליך להוסיף את האות F לערך 23.5 כדי לומר למהדר שאתה באמת רוצה שהערך יהיה float ולא double.

שני האופרטורים האחרונים דורשים מעט יותר הסבר. הבה נתבונן באופרטור ^ להעלאת מספר אחד בחזקת השני. אופרטור זה זמין בשפה VB.NET בלבד:

VB.NET

```
Dim result As Double

result = 2 ^ 3      ' Results in 8
result = 3 ^ 2      ' Results in 9
```

אופרטור זה אינו נתמך ב-C#, אבל תוכל לשכפל את התנהגותו בקלות על ידי שימוש בפונקציה מובנית בשם Math.Pow. שמסופקת על ידי סביבת .NET. קטע הקוד הבא שקול פונקציונלית לקודם:

C#

```
result = Math.Pow(2, 3);           // Results in 8
result = Math.Pow(3, 2);           // Results in 9
```

הפונקציה Math.Pow זמינה גם ב-VB.NET ולכן, אם אתה משתמש בשפה זו, יש לך שתי אפשרויות לבחור מתוכן.

האופרטור האחרון ברשימה הוא אופרטור השארית אשר מחזיר את השארית מפעולת החילוק של שני מספרים. הנה כך:

VB.NET

```
Dim firstNumber As Integer = 17
Dim secondNumber As Integer = 3
Dim result As Integer

result = firstNumber Mod secondNumber ' Results in 2.
```

C#

```
int firstNumber = 17;
int secondNumber = 3;
int result;

result = firstNumber % secondNumber; // Results in 2.
```

במילים פשוטות, אופרטור זה מנסה להפחית את המספר השני מהראשון מספר פעמים רב ככל האפשר, ואז מחזיר את השארית. בדוגמה לעיל הדבר יצליח חמש פעמים, והסכום הכולל שיופחת מהמספר הראשון הוא 15, וכך תתקבל השארית 2 שתוחזר ותאוחסן בתוצאה.

קבוצה נפוצה אחרת של אופרטורים היא אופרטורי השוואה, שמאפשרים להשוות ערכים זה לזה.

אופרטורי השוואה

כמו במקרה של האופרטורים האריתמטיים כך גם כאן, לכל אחת מהשפות VB.NET ו-C# יש קבוצת אופרטורי השוואה (comparison operators) נפרדת לשם השוואה בין ערכים. אופרטור השוואה תמיד משווה שני ערכים או ביטויים ומחזיר ערך בוליאני כתוצאה. הטבלה הבאה מציגה את אופרטורי ההשוואה היותר נפוצים.

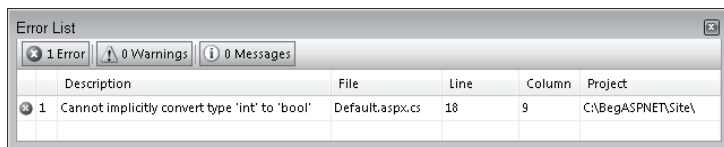
VB.NET	C#	תיאור
=	==	בודק אם שני ערכים שווים זה לזה.
<>	!=	בודק אם שני ערכים אינם שווים.
<	<	בודק אם הערך הראשון קטן מהערך השני.

VB.NET	C#	תיאור
>	>	בודק אם הערך הראשון גדול מהערך השני.
<=	<=	בודק אם הערך הראשון קטן או שווה מהערך השני.
>=	>=	בודק אם הערך הראשון גדול או שווה לערך השני.
Is	is	ב-VB.NET: משווה שני אובייקטים. ב-C#: בודק אם משתנה הוא מסוג מסוים.

הדבר הראשון שתשים אליו לב הוא ש-C# משתמשת בסימן שוויון (=) כפול עבור אופרטורי ההשוואה התקניים. הדבר מבליט בכירור שהוא שונה מאופרטור ההצבה. השימוש בסימן שוויון אחד לשם השוואת שני ערכים הוא שגיאה שכיחה ב-C#. שקול את הדוגמה הבאה:

```
if (result = 4)
{
    // Do something here with result
}
```

הכוונה בדוגמה זו היא לראות אם result שווה ל-4. עם זאת, מכיוון שמשתמשים באופרטור ההצבה (=) ולא באופרטור השוואה (==), תתקבל שגיאת ההידור המוצגת בתרשים 5-2.



תרשים 5-2

תחילה עשויה הודעת השגיאה להיראות מוזרה מעט. אבל אם תתבונן בקוד, תבין את ההיגיון שבדבר. תוצאת משפט ההצבה result=4 היא שהמשתנה result מקבל את הערך השלם 4. מצד שני, משפט "אם" (if) צריך ערך בוליאני כדי לקבוע אם יש לבצע את הקוד המותנה אם לאו. מכיוון שלא ניתן להמיר כך ערך שלם לערך בוליאני, אתה מקבל שגיאת הידור. התיקון הוא קל. עליך להשתמש באופרטור השוואה נכון.

```
if (result == 4)
{
    // Do something here with result
}
```

בדומה לאופרטור ההשוואה הפשוט, ניתן להשתמש באופרטורים האחרים להשוואת ערכים:

VB.NET

4 > 5	' False אינו גדול מ-5 – מתקבל ערך בוליאני
4 <> 5	' True אינו שווה ל-5 – מתקבל ערך בוליאני
5 >= 4	' True גדול או שווה ל-4 – מתקבל ערך בוליאני

C#

```
4 > 5           // false 4 אינו גדול מ-5 - מתקבל ערך בוליאני
4 != 5          // true  4 אינו שווה ל-5 - מתקבל ערך בוליאני
5 >= 4          // true  5 גדול או שווה ל-4 - מתקבל ערך בוליאני
```

מילות המפתח Is ב-VB.NET ו-is ב-C# מבצעות פעולות שונות לחלוטין. ב-VB.NET, Is משווה בין שני מופעים של אובייקטים, כפי שתלמד אודותיו יותר בחלק השני של הפרק. ב-C# משתמשים ב-is כדי לברר אם משתנה מסוים תואם לסוג מסוים. אתה יכול להשיג זאת ב-VB.NET על ידי שימוש באופרטור TypeOf. שתי הדוגמאות הבאות שקולות מבחינה פונקציונלית:

VB.NET

```
Dim myTextBox As TextBox = New TextBox()

If TypeOf myTextBox Is TextBox Then
    ' Run some code when myTextBox is a TextBox
End If
```

C#

```
TextBox myTextBox = new TextBox();

if (myTextBox is TextBox)
{
    // Run some code when myTextBox is a TextBox
}
```

אחד האופרטורים האריתמטיים מאפשר לחבר שני ערכים זה לזה. אבל מה יקרה אם ברצונך לאחד שני ערכים במקום לחבר אותם? זהו המקום שבו משתמשים באופרטורי שרשור.

אופרטורי שרשור

כדי לצרף, או לאחד, מחרוזות זו לזו, דרושים **אופרטורי שרשור** (concatenation operators). ב-C# עליך להשתמש לצורך זה בסימן החיבור (+), וב-VB.NET תשתמש בתו השרשור &. בנוסף, אתה יכול להשתמש בצירוף התווים += (עבור C#), או &= (עבור VB.NET), כדי לשלב את אופרטור השרשור ואת אופרטור ההצבה. עיין בדוגמה הבאה:

VB.NET

```
Dim firstString As String = "Hello "
Dim secondString As String = "World"
Dim result As String

' "Hello World" - כל שלושת קטעי הקוד הבאים מניבים אותה תוצאה

result = firstString & secondString

result = firstString
result = result & secondString
```

```
result = firstString  
result &= secondString
```

C#

```
string firstString = "Hello ";  
string secondString = "World";  
string result;  
  
// "Hello World" - כל שלושת קטעי הקוד הבאים מניבים אותה תוצאה  
  
result = firstString + secondString;  
  
result = firstString;  
result = result + secondString;  
  
result = firstString;  
result += secondString;
```

ב-VB.NET, בנוסף לאופרטורים & ו- &= לכיצוע השרשור, אתה יכול להשתמש גם בתו + וגם בתווים +=. עם זאת, כתלות בסוג הנתונים של הביטויים שאתה מנסה לשרשר, אתה עשוי שלא לקבל את התוצאה המצופה. התבונן בקוד הבא:

```
Dim firstNumber As String = "4"  
Dim secondNumber As Integer = 5  
Dim result As String = firstNumber + secondNumber
```

מכיוון ש-firstNumber הוא מסוג String, אתה עשוי לצפות שהתוצאה הסופית תהיה 45, שזה שרשור של 4 ו-5. אבל, כברירת מחדל, המהדר של VB.NET ימיר את String="4" למספר 4, ואחר כך יתרחש חיבור של הספרות ולא שרשור שלהן, כך ש-result יקבל את הערך 9.

כדי למנוע את אי-הבהירות הזו, השתמש תמיד באופרטורים & ו- &= כדי לשרשר ערכים. בנוסף, על ידי הוספת שורת הקוד הבאה בראש קבצי הקוד שלך, אתה יכול להורות ל-VB.NET להפסיק להמיר ערכים אלה באופן אוטומטי:

```
Option Strict On
```

כך אתה כופה על המהדר להודיע על שגיאה כשעומדת להתרחש המרה לא מפורשת, כמו בדוגמה האחרונה.

הקבוצה האחרונה של אופרטורים שראוי להתבונן בה היא האופרטורים הלוגיים, שבהם נדון בסעיף הבא.

אופרטורים לוגיים

אופרטורים לוגיים (logical operators) משמשים לשילוב התוצאות של מספר ביטויים נפרדים, וכדי לוודא לדוגמה, שמספר תנאים הם אמת (true) או שקר (false). הטבלה הבאה מציגה את האופרטורים הלוגיים הנפוצים ביותר.

VB.NET	C#	תיאור
And	&&	מחזיר את הערך True כאשר תוצאת שני הביטויים היא הערך True.
Or		מחזיר True כאשר התוצאה של לפחות אחד משני הביטויים היא הערך True.
Not	!	הופך את תוצאת הביטוי (אמת לשקר, או true ל- false, ולהפך).
AndAlso	&&	מאפשר לקצר בבדיקת התנאים הלוגיים.
OrElse		מאפשר לקצר בבדיקת התנאים הלוגיים.

השימוש באופרטורים And, Or ו- Not (&&, || ו- ! ב- C#) פשוט למדי, ומודגם בקטעי הקוד הבאים:

VB.NET

```
Dim num1 As Integer = 3
Dim num2 As Integer = 7

If num1 = 3 And num2 = 7 Then           ' True And True = True

If num1 = 2 And num2 = 7 Then           ' False And True = False

If num1 = 3 Or num2 = 11 Then            ' True Or False = True

If Not num1 = 5 Then                     ' Not False = True
```

C#

```
int num1 = 3;
int num2 = 7;

if (num1 == 3 && num2 == 7)              // true And true = true

if (num1 == 2 && num2 == 7)              // false And true = false

if (num1 == 3 || num2 == 11)             // true Or false = true

if (!(num1 == 5))                        // ! false = true
```

האופרטורים AndAlso ו- OrElse של VB.NET פועלים באופן דומה ל- And ו- Or המקבילים להם ב- C#. ההבדל הוא שעם האופרטורים AndAlso ו- OrElse, לעולם לא מתבצעת הערכה של הביטוי השני, מכיוון שהביטוי הראשון כבר קובע את התוצאה עבור ערך הביטוי כולו. כך, עם אופרטור And פשוט:

```
If num1 = 2 And num2 = 7 Then
```

מוערכים שני הביטויים. כלומר, הן num1 והן num2 נשאלים אודות הערך שלהם כדי לראות אם הם שווים 2 ו-7 בהתאמה. אבל, מכיוון ש-num1 אינו שווה ל-2, הרי שאין שום טעם לשאול את num2 לגבי הערך שלו, מכיוון שהתשובה לעולם לא תשנה את התוצאה הסופית של הביטוי המאוחד. זה המקום שבו האופרטור AndAlso מאפשר לגרום קיצור בלוגיקת הבדיקה:

```
If num1 = 2 AndAlso num2 = 7 Then
```

עם קוד זה, הביטוי num2=7 לא יוערך לעולם, מכיוון שכבר num1 אינו עונה על הקריטריון הנדרש.

עם ביטויים פשוטים אלה ייתכן שהדבר אינו נראה כעניין גדול, אבל כשאחד הביטויים הוא פעולה איטית שזמן ביצועה ארוך, הדבר יכול לגרום שיפור משמעותי בביצועים. הבט בקוד הפיקטיבי הבא:

```
If userName = "Administrator" And GetNumberOfRecordsFromDatabase() > 0 Then
```

הקוד זה מתבצע רק כאשר המשתמש הנוכחי נקרא Administrator והפנייה הפיקטיבית לבסיס הנתונים מחזירה רשומה אחת לפחות. עכשיו דמיינ ש- GetNumberOfRecordsFromDatabase() היא פעולה שזמן ביצועה ארוך. יהיה זה בזבוז זמן לבצע אותה אם המשתמש הנוכחי אינו Administrator. גם במקרה זה האופרטור AndAlso פותר את הבעיה:

```
If userName = "Administrator" AndAlso GetNumberOfRecordsFromDatabase() > 0 Then
```

כעת GetNumberOfRecordsFromDatabase() יתבצע רק כשהמשתמש הנוכחי הוא Administrator. עבור כל שאר המשתמשים תהיה התעלמות מהקוד, והדבר יאין את הביצועים עבורם.

ב-C# אין אופרטורים אשר שקולים למילות המפתח (האופרטורים) AndAlso ו-OrElse. אין לה צורך בכך, מכיוון שהלוגיקה הבוליאנית כבר מאפשרת, כברירת מחדל, את הקיצור הלוגי בשפה זו. פירוש הדבר שהביטוי השני לעולם אינו מחושב אם התוצאה של הביטוי הראשון הופכת את חישוב הביטוי השני לבלתי נחוץ.

רוב הדוגמאות הקודמות השתמשו במשפטי If כדי להדגים את האופרטורים הלוגיים. גם משפט If עצמו הוא מבנה לשוני רב חשיבות. משפט If ודרכים אחרות לקבלת החלטות, נידונים להלן.

קבלת החלטות

קבלת החלטות ביישום היא אחד הדברים השכיחים ביותר שאתה עושה כמפתח תוכנה. לדוגמה, כשמשתמש אינו מנהלן, עליך להסתיר מפניו לחצן בטופס ה-Web. דוגמה אחרת, אתה צריך להציג את השורות הזוגיות בטבלה עם רקע אפור בהיר, בעוד שהשורות האי-זוגיות תוצגנה עם רקע לבן. כל ההחלטות האלו יכולות להתבצע על ידי כמה מבנים לוגיים: If, If Else, ElseIf ומשפטי Switch או Select.

המבנים If Else, If Else ו-ElseIf

משפט If הוא הפשוט ביותר מבין משפטי קבלת ההחלטות. משפט If מכיל שני חלקים רלוונטיים: התנאי שנבדק והקוד שמתבצע כשהתנאי מוערך כ-True. לדוגמה,

VB.NET

```
If User.IsInRole("Administrators") Then  
    btnDeleteArticle.Visible = True  
End If
```

C#

```
if (User.IsInRole("Administrators"))  
{  
    btnDeleteArticle.Visible = true;  
}
```

שים לב, שכדי לציין את כלוק הקוד שיתבצע, VB.NET משתמשת במילות המפתח If ו-End If, בעוד C# משתמשת ב-If יחד עם זוג סוגריים מסולסלים. בנוסף, ב-C# נדרשים סוגריים סביב התנאי שנבדק, בעוד ש-VB.NET דורשת שימוש במילת המפתח Then לאחר התנאי.

לעתים קרובות תרצה לבצע פעולה אחרת אם התנאי אינו True. על ידי שימוש באופרטור השלילה Not או "!", תוכל לכתוב משפט אחר:

VB.NET

```
If User.IsInRole("Administrators") Then  
    btnDeleteArticle.Visible = True  
End If  
If Not User.IsInRole("Administrators") Then  
    btnDeleteArticle.Visible = False  
End If
```

C#

```
if (User.IsInRole("Administrators"))  
{  
    btnDeleteArticle.Visible = true;  
}  
if (!User.IsInRole("Administrators"))  
{  
    btnDeleteArticle.Visible = false;  
}
```

כל זה מוביל בבירור לקוד לא מסודר, מכיוון שאתה צריך לחזור על הערכת כל הביטוי פעמיים: פעם למקרה של True ופעם נוספת למקרה של False. למרבה המזל, יש פתרון קל יותר: הבלוק Else (ב-C#).

VB.NET

```
If User.IsInRole("Administrators") Then
    btnDeleteArticle.Visible = True
Else
    btnDeleteArticle.Visible = False
End If
```

C#

```
if (User.IsInRole("Administrators"))
{
    btnDeleteArticle.Visible = true;
}
else
{
    btnDeleteArticle.Visible = false;
}
```

מבנה זה של If Else פועל היטב עבור תנאים פשוטים. אבל שקול תסריט שבו יש יותר משתי אפשרויות. בתסריטים אלה באפשרותך להשתמש ב-ElseIf (VB.NET) או else-if (C#).

תוכל כעת להניח שהאתר שלך משרת שלושה בעלי תפקידים שונים: מנהלנים, מנהלי תוכן וחברים רגילים. מנהלנים יכולים ליצור ולמחוק תוכן; מנהלי תוכן יכולים ליצור תוכן חדש בלבד, ואילו חברים רגילים אינם יכולים לבצע אף לא אחת מפעולות אלו. כדי להציג או להסתיר את הלחצנים הרלוונטיים לכל בעל תפקיד, תוכל להשתמש בקוד הבא:

VB.NET

```
If User.IsInRole("Administrators") Then
    btnCreateNewArticle.Visible = True
    btnDeleteArticle.Visible = True
ElseIf User.IsInRole("ContentManagers") Then
    btnCreateNewArticle.Visible = True
    btnDeleteArticle.Visible = False
ElseIf User.IsInRole("Members") Then
    btnCreateNewArticle.Visible = False
    btnDeleteArticle.Visible = False
End If
```

C#

```
if (User.IsInRole("Administrators"))
{
    btnCreateNewArticle.Visible = true;
    btnDeleteArticle.Visible = true;
}
else if (User.IsInRole("ContentManagers"))
{

```

```

    btnCreateNewArticle.Visible = true;
    btnDeleteArticle.Visible = false;
}
else if (User.IsInRole("Members"))
{
    btnCreateNewArticle.Visible = false;
    btnDeleteArticle.Visible = false;
}

```

ניתן לראות בבירור כי השימוש ב-ElseIf או else if, עוזר לקבלת קוד קריא יותר. עם זאת, כשצריך לבדוק ביטויים רבים, אתה עלול לסיים עם קוד מורכב. אם זהו המקרה, ביכולתך להשתמש במשפטים, או המבנים, (VB.NET) Select Case או (C#) switch.

המבנים / Switches / Select Case

תוכל להניח שאתה בונה אתר אינטרנט עבור אולם קונצרטים שיש לו מופעים בשבת. מבקרים יכולים לקנות במשך השבוע כרטיסים למופע של שבת. כדי להניע מבקרים לרכוש כרטיסים מוקדם ככל האפשר, אתה מחליט לתת הנחה למקדימים. ככל שהכרטיסים נקנים מוקדם יותר בשבוע, מחירם זול יותר. הקוד לחישוב שיעור ההנחה יכול להיראות כך, תוך שימוש במשפטים (VB) Select Case או (C#) switch:

VB.NET

```

Dim today As DateTime = DateTime.Now
Dim discountRate As Double = 0

Select Case today.DayOfWeek
    Case DayOfWeek.Monday
        discountRate = 0.4
    Case DayOfWeek.Tuesday
        discountRate = 0.3
    Case DayOfWeek.Wednesday
        discountRate = 0.2
    Case DayOfWeek.Thursday
        discountRate = 0.1
    Case Else
        discountRate = 0
End Select

```

C#

```

DateTime today = DateTime.Now;
double discountRate = 0;

switch (today.DayOfWeek)
{

```



```

case DayOfWeek.Monday:
    discountRate = 0.4;
    break;
case DayOfWeek.Tuesday:
    discountRate = 0.3;
    break;
case DayOfWeek.Wednesday:
    discountRate = 0.2;
    break;
case DayOfWeek.Thursday:
    discountRate = 0.1;
    break;
default:
    discountRate = 0;
    break;
}

```

לכל יום מימות השבוע שלגביו חלה הנחה (שני עד חמישי) יש בלוק Case נפרד. כמעט ואין הבדל בתחביר בין VB.NET לבין C#: C# משתמשת באותיות רגילות עבור המילה case, ודורשת נקודתיים (:). לאחר כל תווית case. בנוסף, עליך לצאת מכל בלוק case על ידי משפט break. בזמן ביצוע, מוערך התנאי (today.DayOfWeek) ומבוצע הבלוק הנכון. חשוב להבין שרק הבלוק המתאים מתבצע. כשלא נמצא בלוק העונה על התנאי, מתבצע הקוד ב- Case Else (VB) או ב- default (C#) עבור הימים שבין שישי לראשון. אינך חייב לכתוב את הבלוק ב- Case Else או ב- default אם כי מומלץ לעשות זאת, מכיוון שכך תקבל קוד יותר מפורש וקל לקריאה. הדוגמאות לעיל יכלו שלא לכלול אותו, כי בראש בלוק הקוד רואים ש-discountRate מקבל מלכתחילה את ערך ברירת המחדל - 0.

כדי לקבל תחושה של השימוש במשפטים שראית עד כה, הנה תרגיל נסה זאת שמציג איך להשתמש בהם ביישום הדגמה קטן.

נסה זאת יצירת מחשבון אינטרנט פשוט

בתרגיל זה תיצור מחשבון פשוט שמסוגל לחבר, לחסר, לכפול ולחלק ערכים. התרגיל מראה כיצד להשתמש בכמה מהאופרטורים הלוגיים ובאופרטור ההצבה, ומדגים את השימוש במבנים If, ו-Select Case / switch.

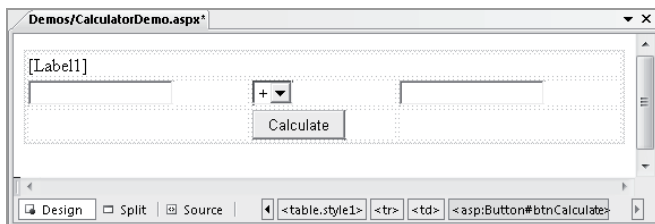
1. צור קובץ חדש בשם CalculatorDemo.aspx בתיקייה Demos. ודא שאינך קורא לדף בשם Calculator, כיוון שאחרת תיקלע לבעיות מאוחר יותר בפרק כשתיצור מחלקה בשם זה. ודא שוב שאתה משתמש בקוד ברקע (Code Behind), ומהרשימה הנפתחת של שפות בחר בשפה הרצויה.
2. העבר את הדף למצב עיצוב (Design View), לחץ במלבן המקווקו כדי להתמקד בו, והוסף טבלה בת שלוש שורות ושלוש עמודות על ידי פקודות התפריט Table ← Insert Table.

מזג את שלושת התאים של השורה הראשונה על ידי בחירתם, לחץ לחיצה ימנית על התא שנוצר ומהתפריט שנפתח בחר Merge Cells ←.

3. הוסף את הפקדים הבאים לדף, הגדר את ה-ID והמאפיינים האחרים שלהם בהתאם לטבלה הבאה, וסדר אותם כפי שמוצג בתרשים 5-3.

סוג הפקד	ID של הפקד	הגדרות המאפיין
Label	lblResult	נקה את המאפיין Text שלו. כדי לבצע זאת, לחץ לחיצה ימנית על המאפיין Text בטבלת המאפיינים ובחר Reset.
TextBox	txtValue1	
DropDownList	lstOperator	הוסף ארבעה ListItems עבור ארבעת האופרטורים הלוגיים הבאים: + - * /
TextBox	txtValue2	
Button	btnCalculate	הגדר את המאפיין Text של הלחצן כ-Calculate.

כשתסיים, הדף שלך יוצג ב- Design View כמו שנראה בתרשים 5-3.



תרשים 5-3

4. לחץ לחיצה כפולה על Calculate והוסף את הקוד המוצג להלן:

VB.NET

```
Protected Sub btnCalculate_Click(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles btnCalculate.Click
```

```
    If txtValue1.Text.Length > 0 AndAlso txtValue2.Text.Length > 0 Then
```

```
        Dim result As Double = 0
```

```
        Dim value1 As Double = Convert.ToDouble(txtValue1.Text)
```

```
        Dim value2 As Double = Convert.ToDouble(txtValue2.Text)
```

```
        Select Case lstOperator.SelectedValue
```

```
            Case "+"
```

```

        result = value1 + value2
    Case "-"
        result = value1 - value2
    Case "*"
        result = value1 * value2
    Case "/"
        result = value1 / value2
End Select
lblResult.Text = result.ToString()
Else
    lblResult.Text = String.Empty
End If

```

End Sub

C#

protected void btnCalculate_Click(object sender, EventArgs e)

```

{
    if (txtValue1.Text.Length > 0 && txtValue2.Text.Length > 0)
    {
        double result = 0;
        double value1 = Convert.ToDouble(txtValue1.Text);
        double value2 = Convert.ToDouble(txtValue2.Text);

        switch (lstOperator.SelectedValue)
        {
            case "+":
                result = value1 + value2;
                break;
            case "-":
                result = value1 - value2;
                break;
            case "*":
                result = value1 * value2;
                break;
            case "/":
                result = value1 / value2;
                break;
        }
        lblResult.Text = result.ToString();
    }
    else
    {
        lblResult.Text = string.Empty;
    }
}

```

5. שמור את כל השינויים והקש Ctrl+F5 כדי לפתוח את הדף בדפדפן. אם תקבל שגיאה במקום לראות את הדף, ודא שהקלדת את הקוד בדיוק כפי שהוצג כאן, ושקראת לכל הפקדים בהתאם לטבלה לעיל.
6. הכנס מספר בתיבות הטקסט הראשונה והשנייה, בחר באופרטור מהרשימה הנפתחת, ולחץ על Calculate. הקוד שנמצא בקובץ קוד ברקע יתבצע, ואז - בהתבסס על הפריט שבחרת מהרשימה הנפתחת - יתבצע החישוב הנכון והתויות תעודכן עם התוצאה.
7. נסה כמה מספרים ואופרטורים אחרים. תיווכח לדעת שהמחשבון מבצע את הפעולה הנכונה בכל פעם שאתה לוחץ על Calculate.

כיצד זה פועל

כשאתה מכניס שני ערכים ולוחץ על Calculate, מתבצע הקוד הבא שנמצא בקוד ברקע:

VB.NET

```
If txtValue1.Text.Length > 0 AndAlso txtValue2.Text.Length > 0 Then
```

C#

```
if (txtValue1.Text.Length > 0 && txtValue2.Text.Length > 0)
```

קוד זה נחוץ כדי להבטיח ששתי תיבות הטקסט מכילות ערך. (בפרק 9 תראה דרך הרבה יותר נקייה לבצע בדיקת תקפות זו.) כדי להבטיח ששני השדות מכילים ערך, הקוד משתמש במשפט If פשוט. הקוד משתמש גם באופרטור AndAlso של VB.NET כדי להימנע מבדיקת המאפיין Text של הפקד TextBox השני כאשר הראשון ריק.

כדי שיוכל להחזיק את תוצאת החישוב, הקוד מכריז על משתנים מסוג Double, ומקבל את הערכים משני פקדי תיבות הטקסט. כעת הקוד ממיר את הערכים ל-Double על ידי שימוש בשיטת ToDouble של המחלקה Convert, ומגדיר בלוק Select Case (ב-C# switch) כדי לטפל בסוג האופרטור שבחרת ברשימה הנפתחת:

VB.NET

```
Select Case lstOperator.SelectedValue  
Case "+"  
    result = value1 + value2
```

C#

```
switch (lstOperator.SelectedValue)  
{  
    case "+":  
        result = value1 + value2;  
        break;
```

לכל פריט ברשימה הנפתחת יש משפט case. כשבחרת מהרשימה באופרטור +, יתבצע הקוד שמופיע בבלוק case הראשון, והמשתנה result מקבל ערך שהוא סכום המספרים שהכנסת בשתי תיבות הטקסט. באופן דומה, כשתבחר באופרטור החיסור יופחתו שני הערכים זה מזה. בסוף התהליך מומרת התוצאה ל-String ומוצגת בתווית lblResult.

משפטי Select Case / switch מסיימים את הדיון בנושא קבלת החלטות בקוד. נשאר לנו לדון בקבוצה אחת נוספת של משפטים: לולאות. קטעי קוד הקרויים לולאה מאפשרים לבצע פעולה מחזורית (לולאה) על פני קוד או על פני אובייקטים שנמצאים באוסף.

לולאות

הלולאות (loops) שימושיות ואף נחוצות ביישומים רבים, מכיוון שהן מאפשרות לבצע קוד בצורה מחזורית, שוב ושוב, מבלי שצריך לכתוב קוד זה יותר מפעם אחת. לדוגמה, נניח שיש אתר אינטרנט שצריך לשלוח עלון חדשות אל 20,000 המנויים שלו באמצעות דואר אלקטרוני. את הקוד לשליחת עלון החדשות תכתוב פעם אחת, ותשתמש בלולאה לשליחתו לכל אחד מהמנויים שקוד התוכנית ימצא בבסיס נתונים.

יש כמה סוגים של לולאות, שלכל אחת מהן יש שימוש מיוחד וגם יתרונות.

הלולאה For

הלולאה **For** חוזרת על הקוד שלה מספר קבוע מראש של פעמים. את המספר המדויק של החזרות (איטרציות) קובעים בעת הגדרת הלולאה. הלולאה For היא בעלת המבנה הבא:

VB.NET

```
For counter [ As datatype ] = start To end [ Step step ]  
    ' Code that must be executed for each iteration  
Next [ counter ]
```

C#

```
for (startCondition; endCondition; step definition)  
{  
    // Code that must be executed for each iteration  
}
```

זה נראה מעט מוזר, אבל הדוגמה הבאה מציגה זאת באופן פשוט:

VB.NET

```
For loopCount As Integer = 1 To 10  
    Label1.Text &= loopCount.ToString() & "<br />"  
Next
```

C#

```
for (int loopCount = 1; loopCount <= 10; loopCount++)  
{  
    Label1.Text += loopCount.ToString() + "<br />";  
}
```

למרות שהתחביר שמשמשים בו בשתי השפות שונה למדי, שתי דוגמאות הקוד מבצעות את אותה הפעולה: הן כותבות מספרים מ-1 עד 10 בפקד Label. כלומר, הלולאה מתחילה בהצבת המספר 1 במשתנה loopCount. הערך מומר ל-String ומוצב בפקד Label. כעת מגדילים את loopCount ב-1 והלולאה ממשיכה. כך זה נמשך עד שהערך של loopCount הוא 10 והלולאה מסתיימת. בדוגמה זו השתמשנו במספרים שהוכנסו לקוד באופן ישיר. עם זאת, ניתן להחליף את תנאי ההתחלה והסיום בערכים דינמיים שמקורם במשתנים או באובייקטים אחרים. לדוגמה, אם נחזור לדוגמה של מערך התפקידים שראינו לעיל, אפשר לכתוב כל תפקיד במערך בדרך זו:

VB.NET

```
For loopCount As Integer = 0 To roles.Length - 1  
    Label1.Text &= roles(loopCount) & "<br />"  
Next
```

C#

```
for (int loopCount = 0; loopCount < roles.Length; loopCount++)  
{  
    Label1.Text += roles[loopCount] + "<br />";  
}
```

זכור כי מערכים הם מבוססי-אפס, ופירוש הדבר שהסימון של הפריט הראשון הוא roles(0) (VB.NET), או roles[0] (C#) בהתאמה. כלומר, הלולאה צריכה להתחיל מ-0 (ולא מ-1). המאפיין Length של המערך מחזיר את המספר הכולל של פריטים שהמערך מכיל. כאשר במערך יש שלושה תפקידים, Length יחזיר 3. לכן, הקוד מפחית 1 מ-Length ומשתמש בערך זה כתנאי לסיום הלולאה. במקרה שלפנינו, הלולאה תתבצע מ-0 עד 2, ולכן תיגש אל כל שלושת האיברים.

הדוגמה ב-C# אינה מפחיתה 1 מ-Length. במקום זאת היא משתמשת בביטוי הבא:

```
loopCount < roles.Length;
```

הלולאה תימשך כל זמן ש-loopCount הוא פחות מגודל המערך. גם כאן הדבר גורם ללולאה לגשת לכל שלושת הפריטים, מ-0 עד 2.

אם אתה מבצע לולאה על פני מערך או על פני אוסף נתונים, עומדת לרשותך גם לולאה אחרת, שקל יותר לקרוא ולהשתמש בה: הלולאה For Each או foreach.

הלולאות For Each / foreach

הלולאות **For Each** (VB.NET) ו-**foreach** (C#) סורקות את כל הפריטים באוסף. אם ניקח את מערך התפקידים כדוגמה, באפשרותך לבצע את הקוד הבא כדי להדפיס כל שם תפקיד בפקד :Label

VB.NET

```
For Each role As String In roles
    Label1.Text &= role & "<br />"
Next
```

C#

```
foreach (string role in roles)
{
    Label1.Text += role + "<br />";
}
```

מכיוון שמשתנה התפקידים הוא מערך של מחרוזות, עליך להגדיר את הלולאה גם עם String. באופן דומה, אם האוסף שעליו מתבצעת הלולאה מכיל את סוגי הנתונים Integer או Boolean, תצטרך להגדיר את הלולאה עם Integer או Boolean, בהתאמה.

פרט ללולאות For Each ו- For Each יש עוד לולאה אחת שצריך לברוק: הלולאה While.

הלולאות While ו-Until

כפי שמשמע משמה, לולאת **While** מתבצעת כל עוד מתקיים תנאי מסוים. בניגוד לשני סוגי הלולאות האחרים שמסתיימים מעצמם, הלולאה While יכולה לפעול לנצח אם לא תהיה זהיר. הדוגמה הבאה מראה כיצד להשתמש בלולאה While:

VB.NET

```
Dim success As Boolean = False

While Not success
    success = SendEmailMessage()
End While
```

C#

```
bool success = false;

while (!success)
{
    success = SendEmailMessage();
}
```

הקוד מנסה לשלוח הודעת דואר אלקטרוני, והוא יתמיד בכך עד שיצליח - כלומר, כל עוד המשתנה success מכיל את הערך False (false ב-C#). שים לב שהאופרטורים Not ו- ! הופכים את הערך של success. השיטה SendEmailMessage אמורה להחזיר True כאשר תצליח ו-False

כאשר תיכשל. אם הכל פועל כמתוכנן, הקוד נכנס ללולאה וקורא ל-SendEmailMessage. אם היא מחזירה True, תנאי הלולאה אינו מתקיים עוד, והלולאה מסתיימת. אבל אם SendEmailMessage מחזירה False, למשל, מפני ששרת הדואר קרס, הלולאה תמשיך ותקרא ל-SendEmailMessage שוב ושוב עד אינסוף.

כדי למנוע לולאה אינסופית עם הלולאה While, כדאי לעתים להוסיף תנאי שמסיים את הלולאה לאחר מספר מסוים של ניסיונות. לדוגמה, הקוד הבא עוזר למנוע לולאה אינסופית במקרה ששרת הדואר אינו זמין:

VB.NET

```
Dim success As Boolean = False
Dim loopCount As Integer = 0

While Not success And loopCount < 3
    success = SendEmailMessage()
    loopCount = loopCount + 1
End While
```

C#

```
bool success = false;
int loopCount = 0;

while (!success && loopCount < 3)
{
    success = SendEmailMessage();
    loopCount = loopCount + 1;
}
```

בקוד זה המשתנה loopCount אחראי ליציאה מהלולאה לאחר שלושה ניסיונות לקרוא ל-SendEmailMessage. במקום להשתמש בתבנית loopCount = loopCount + 1, אתה יכול להשתמש בשילוב של אופרטור השרשור ואופרטור ההצבה בדרך זו:

VB.NET

```
loopCount += 1
```

C#

```
loopCount += 1;

// Alternatively C# allows you to do this:
loopCount++;
```

לכל הדוגמאות תוצאה אחת: הערך של loopCount מוגדל ב-1, והסכום החדש מוצג שוב ב-loopCount. קיצור הדרך ב-C# שנכתב כך loopCount++, הוא דרך נפוצה מאוד להגדלת ערך של משתנה ב-1. באופן דומה משמשים loopCount -- או loopCount -= 1 להקטנת הערך ב-1.

פרט ללולאה While יש כמה חלופות נוספות, כמו הלולאה **Do While** אשר מבטיחה שהקוד לביצוע יתבצע תמיד לפחות פעם אחת, והלולאה **Do Until** שממשיכה עד שתנאי מסוים מתקיים, כמו במקרה של הלולאה While.

עד כה הקוד שראית מורכב מדוגמאות קצרות ופשוטות שניתן להציבן באופן ישיר בקוד ברקע של דף Web; כמו לדוגמה ב-Page_Load שראית קודם. עם זאת, באתרי אינטרנט בעולם הממשי, קרוב לוודאי שתמצא לבנות ולארגן את הקוד שלך הרבה יותר. בסעיף הבא תראה דרכים שונות להשיג זאת.

ארגון הקוד

כשתתחיל להוסיף דפים רבים לאתר שלך, קרוב לוודאי שתמצא שיש קוד שאפשר לחזור ולהשתמש בו בדפים שונים. לדוגמה, ייתכן שיהיה לך קוד שקורא הגדרות מקובץ web.config, שתצטרך לו במספר דפים; או שתמצא לשלוח מדפים שונים דואר אלקטרוני עם פרטי משתמש. מסיבה זו עליך למצוא דרך לרכז את הקוד שלך במקום שיהיה לך נוח להשתמש בו ולתחזק אותו. כדי להשיג זאת, באתר אינטרנט בסביבת ASP.NET 3.5 תוכל להשתמש בפונקציות ובשגרות שיידונו להלן. כדי שפונקציות ושגרות אלו תהיינה זמינות לכל הדפים באתר, עליך להציב אותן במיקום מיוחד שבו נדון אחר כך.

שיטות: פונקציות ושגרות

פונקציות ושגרות דומות מאוד זו לזו; שתיהן מאפשרות ליצור בלוק קוד לשימוש חוזר שתוכל להשתמש בו ממקומות אחרים באתר שלך. ההבדל בין פונקציה לבין שגרה הוא בכך שפונקציה יכולה להחזיר נתונים ושגרה אינה יכולה. פונקציות ושגרות נקראות בשם **שיטות** (methods). תפגוש במונח זה שוב בחלק האחרון של הפרק שעוסק בהכוונת אובייקטים.

כדי שפונקציות ושגרות תהיינה שימושיות יותר, ניתן לבטא אותן במונחים של פרמטרים. כלומר, אתה יכול להעביר להן מידע נוסף שבו תוכלנה להשתמש. בדרך כלל פונקציות ושגרות הן בעלות התבנית הבאה (שורות הרווח אינן הכרחיות, אלא לצורך הפרדה):

VB.NET

```
' הגדרת פונקציה
Public Function FunctionName ([parameterList]) As DataType

End Function

' הגדרת שגרה
Public Sub SubName ([parameterList])

End Sub
```

C#

```
// הגדרת פונקציה
public datatype FunctionName([parameterList])
{

}

// הגדרת שגרה
public void SubName([parameterList])
{

}
```

אל השורה המלאה הראשונה, שמתחילה ב-Public, מתייחסים כאל "חתימת השיטה". מכיוון שהיא מגדירה את המראה של הפונקציה, כולל שמה והפרמטרים שלה. מילת המפתח Public (public ב-C#) נקראת **מגביל גישה** (access modifier) וקובעת באיזו מידה יכולים דפים אחרים או קבצי קוד לראות את השיטה הזו. נדון בנושא בפירוט בשלב מאוחר יותר בפרק זה. בשלב זה עליך להבין ש-Public גורמת לכך שהשיטה זמינה לאחרים ואף זמינה לקריאה מקוד חיצוני כלשהו.

לאחר שם הפונקציה יש סוגריים, שבתורם יכולים להכיל רשימת פרמטרים (רשות). החלק הנטוי בדוגמאות קוד אלו יוחלף בקוד האמיתי שלך בערכים ממשיים. החלקים שבין הסוגריים המרובעים () הם רשות. כדי שהדברים יהיו יותר מעשיים, או תכליתיים, הנה כמה דוגמאות של פונקציות ושגרות:

VB.NET

```
Public Function Add(ByVal a As Integer, ByVal b As Integer) As Integer
    Return a + b
End Function

Public Sub SendEmail(ByVal emailAddress As String)
    ' Code to send an e-mail goes here
End Sub
```

C#

```
public int Add(int a, int b)
{
    return a + b;
}

public void SendEmail(string emailAddress)
{
    // Code to send an e-mail goes here
}
```

בדוגמאות קוד אלו זה ברור שהפונקציה מחזירה ערך, ושגרות אינן מחזירות. כך, השיטה Add משתמשת במילת המפתח Return (return באותיות קטנות ב-C#) כדי להחזיר את הערך הכולל של a ו-b. כאשר מגדירים שגרה באמצעות Sub ב-VB.NET ו-void ב-C#, אין חובה להשתמש במילת המפתח Return (שוב, מפני ששגרה אינה מחזירה ערך), אם כי אתה יכול להשתמש בה כדי לצאת מהשיטה בטרם עת.

לבסוף, הן לפונקציה והן לשגרה יש רשימת פרמטרים. במקרה של השיטה Add יש שני פרמטרים: אחד עבור האיבר השמאלי של פעולת החיבור והאחר – עבור האיבר הימני. השיטה SendEmail היא בעלת פרמטר יחיד String, שמחזיק את כתובת הדואר האלקטרוני של המשתמש.

בדוגמת VB.NET מופיעה מילת המפתח ByVal בחזית כל פרמטר שברשימת הפרמטרים. זו ברירת המחדל של הסוג עבור כל הפרמטרים, ו-IDE יוסיף אותה באופן אוטומטי במקרה שתשכח. הניגוד של ByVal היא ByRef. מילות מפתח אלו קובעות את הדרך שבה נשלח ערך לפונקציה או לשגרה. כשאתה מציין ByVal נוצר עותק של המשתנה. כל השינויים שייעשו בעותק זה בתוך השיטה יאבדו ברגע שהשיטה תסיים לפעול. בניגוד לכך, כשתציין ByRef נשלח אל השיטה המשתנה המקורי עצמו. כל השינויים שייעשו במשתנה הנכנס יישמרו גם לאחר היציאה מהשיטה. הדוגמה הקצרה הבאה מציגה כיצד זה פועל:

```
Public Sub ByValDemo(ByVal someValue As Integer)
    someValue = someValue + 20
End Sub

Public Sub ByRefDemo(ByRef someValue As Integer)
    someValue = someValue + 20
End Sub

Dim x As Integer = 0
ByValDemo(x)

Label1.Text = x.ToString()      ' המשתנה המקורי לא השתנה וערכו 0

Dim y As Integer = 0
ByRefDemo(y)

Label2.Text = y.ToString()      ' המשתנה הועבר לשגרה וערכו שונה ל-20
```

ל-C# יש מבנה דומה שמשתמש במילת המפתח ref. ההבדל הגדול ביותר לעומת VB.NET הוא בכך שאינך צריך לציין דבר אם אין ברצונך להשתמש בפרמטרים של הפנייה, אבל עליך לציין את מילת המפתח ref בקריאה לשיטה:

```
public void ByRefDemo(ref int x)
{
    x = x + 20;
}

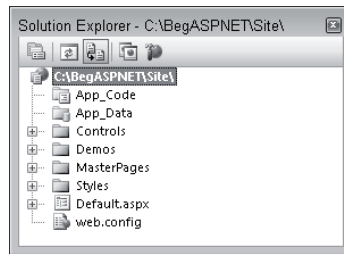
int y = 0;
ByRefDemo(ref y);              // המשתנה הועבר לשגרה וערכו שונה ל-20
```

היזהר כשאתה משתמש בפרמטרי הפנייה בצורה זו, כי השיטה עלולה לשנות פרמטרים חשובים בקוד הקורא עוד לפני שתשים לב לכך. הדבר עלול להוביל לתקלות שקשה לעלות על עקבותיהם.

כדי שהשיטות שמיועדות לשימוש בכל האתר תהיינה נגישות לדפים שבאתר, עליך להציב אותן במקום מרכזי. התיקייה המיוחדת App_Code היא מקום מתאים עבור הקוד שלך.

התיקייה App_Code

בדיוק כמו התיקייה App_Data שראית בפרקים 1 ו-2, כך גם התיקייה App_Code היא תיקייה מיוחדת של ASP.NET 3.5. היא עוצבה במיוחד להחזיק קבצי קוד, כמו מחלקות שבהן תשתמש בדפים שונים באתר. קוד שחל על דף אחד בלבד (כמו טיפול בלחיצה של פקד Button) צריך להישאר בקוד ברקע של הדף, כפי שראית עד כה. כדי להוסיף את התיקייה App_Code לאתר שלך, לחץ לחיצה ימנית על שם האתר בסייר הפתרון, ובחר Add ASP.NET Folder ← App_Code. התיקייה מתווספת לאתר ומקבלת סמל מיוחד: תיקייה עם מסמך קוד קטן בראשה, כפי שרואים בתרשים 5-4.



תרשים 5-4

כשהתיקייה App_Code במקומה, אתה יכול להתחיל להוסיף לה קבצי מחלקה. לקבצי מחלקה יש סיומת שתואמת את שפת התכנות שבחרת עבור האתר: cs עבור קבצי C# ו-vb עבור קבצים שמכילים קוד בשפה VB.NET. בתוך קבצי מחלקה אלה באפשרותך ליצור מחלקות שבתוכן יכילו שיטות (פונקציות ושגרות) שיכולות לבצע מטלות נפוצות. בחלק האחרון של הפרק נדון במחלקות ביתר פירוט. בשלב זה נתמקד בשיטות שנמצאות בקובץ הקוד וכיצד קוראים להן, ולא בשאלה מדוע צריך להוסיף תחילה את הקוד למחלקה.

התרגיל הבא מראה כיצד להשתמש בתיקייה App_Code כדי ליעל את המחשבון שיצרת בתרגיל נסה זאת קודם.

נסה זאת אופטימיזציה של המחשבון

בתרגיל זה תיצור מחלקה בשם Calculator שחושפת ארבע שיטות: Add, Subtract, Multiply, Divide. כשהמחלקה מוגדרת ומסוגלת לבצע את פעולות החישוב הנדרשות, תשנה את הקובץ CalculatorDemo.aspx באופן שישתמש במחלקה החדשה Calculator שיצרת. דוגמה זו הינה פשוטה למדי, בהתחשב בכמות הקוד שאתה צריך לכתוב ולגמישות הנוספת שאתה משיג על

ידי העברת הקוד מדף ASPX למחלקה Calculator שבתיקייה App_Code, כדי שגם יישומים אחרים יוכלו להשתמש בו. עם זאת, הדוגמה מאפשרת להבין את הרעיון הכללי של השימוש במחלקה. הדוגמה גם מקיפה במידה מספקת כדי להציג את הרעיון, ועדיין קצרה מספיק כדי לאפשר להבין את הקוד בצורה קלה.

1. אם עדיין לא עשית זאת, התחל על ידי הוספת התיקייה App_Code לאתר שלך על ידי לחיצה ימנית על האתר ובחירת Add ASP.NET Folder ← App_Code.
2. לחץ לחיצה ימנית על התיקייה החדשה App_Code ובחר Add New Item.
3. בתיבת הדו-שיח שמופיעה לחץ Class, ומהרשימה הנפתחת Language בחר בשפה הרצויה.
4. הקלד Calculator עבור שם הקובץ ולחץ Add. הפעולה יוצרת קובץ מחלקה שמכיל מחלקה בשם Calculator. שים לב, זהו נוהג נפוץ לקבוע למחלקות שמות תוך שימוש במה שנקרא Pascal Casing, שלפיה כל מילה בשם מתחילה באות רישית.
5. מייד לאחר שורת הקוד שמגדירה את המחלקה class הוסף את ארבע השיטות שלהלן:

VB.NET

Public Class Calculator

```
Public Function Add(ByVal a As Double, ByVal b As Double) As Double
    Return a + b
End Function

Public Function Subtract(ByVal a As Double, ByVal b As Double) As Double
    Return a - b
End Function

Public Function Multiply(ByVal a As Double, ByVal b As Double) As Double
    Return a * b
End Function

Public Function Divide(ByVal a As Double, ByVal b As Double) As Double
    Return a / b
End Function
```

End Class

C#

public class Calculator

{

```
public double Add(double a, double b)
{
    return a + b;
}
```

```

public double Subtract(double a, double b)
{
    return a - b;
}

public double Multiply(double a, double b)
{
    return a * b;
}

public double Divide(double a, double b)
{
    return a / b;
}

```

```

public Calculator()
{
    //
    // TODO: Add constructor logic here
    //
}

```

6. שנה את הקוד ברקע של הדף CalculatorDemo.aspx באופן שישתמש במחלקה שזה עתה יצרת. תצטרך לערוך שני שינויים: ראשית, להוסיף שורת קוד שיוצרת מופע של המחלקה Calculator, ושנית, לשנות כל בלוק Case כך שישתמש בשיטות החישוב הנדרשות לכל סוג פעולה:

VB.NET

```

Dim myCalculator As New Calculator() ' הגדרת משתנה מסוג המחלקה
Select Case lstOperator.SelectedValue
    Case "+"
        result = myCalculator.Add(value1, value2)
    Case "-"
        result = myCalculator.Subtract(value1, value2)
    Case "*"
        result = myCalculator.Multiply(value1, value2)
    Case "/"
        result = myCalculator.Divide(value1, value2)
End Select

```

C#

```

Calculator myCalculator = new Calculator(); // הגדרת משתנה מסוג המחלקה
switch (lstOperator.SelectedValue)
{
    case "+":
        result = myCalculator.Add(value1, value2);
        break;
    case "-":

```

```

    result = myCalculator.Subtract(value1, value2);
    break;
case "*":
    result = myCalculator.Multiply(value1, value2);
    break;
case "/":
    result = myCalculator.Divide(value1, value2);
    break;
}

```

7. שמור את כל השינויים ופתח את הדף בדפדפן. המחשבון עדיין פועל כמו קודם; אבל הפעם החישובים אינם מבוצעים בקובץ קוד ברקע של הדף, אלא על ידי המחלקה Calculator שבתיקייה App_Code.

כיצד זה פועל

הקובץ שיצרת בתיקייה App_Code מכיל מחלקה בשם Calculator. תלמד יותר אודות מחלקות בסעיף האחרון של פרק זה. בשלב זה חשוב לדעת, שמחלקה היא כמו הגדרה עבור אובייקט שיכול לחשוף שיטות שאפשר לקרוא להן בזמן ביצוע. במקרה שלפנינו, ההגדרה עבור המחלקה Calculator מכילה ארבע שיטות לביצוע פעולות אריתמטיות. שיטות אלו מקבלות פרמטרים עבור הצד השמאלי ועבור הצד הימני של נוסחת החישוב. כל שיטה מבצעת את החישוב שהיא נדרשת לבצע (Add, Subtract וכו') ומחזירה את התוצאה אל הקוד שפנה אליה, או "קרא" לה.

הקוד שנמצא בקוד ברקע של הדף CalculatorDemo.aspx יוצר מופע (instance) של המחלקה Calculator. כלומר, הוא יוצר בזיכרון המחשב אובייקט שמבוסס על הגדרת המחלקה. כדי לעשות זאת הקוד ברקע משתמש במילת המפתח New (ב-C#) ליצירת מופע של Calculator, אשר מאוחסן אחר כך במשתנה myCalculator. כאשר נדרון באובייקטים בהמשך הפרק, תלמד יותר על מילת המפתח New. שים לב שסוג הנתונים של משתנה זה הוא Calculator כשם המחלקה.

VB.NET

```
Dim myCalculator As New Calculator()
```

C#

```
Calculator myCalculator = new Calculator();
```

ברגע שנוצר המופע Calculator, אתה יכול לקרוא לשיטות שלו. בדיוק כפי שראית קודם לגבי שיטות אחרות, השיטות של המחלקה Calculator מקבלות פרמטרים שמועברים אליהן על ידי הקוד הקורא:

VB.NET

```
Case "+"
```

```
    result = myCalculator.Add(value1, value2)
```

C#

```
case "+":
```

```
    result = myCalculator.Add(value1, value2);
```

```
    break;
```

השיטה Add מחברת אחר כך את שני הערכים ומחזירה את התוצאה כ-double, שמאוחסן במשתנה result. כמו בגרסה הראשונה של המחשבון, בסופו של דבר מוצגת התוצאה בדף על ידי הפקד Label.

פונקציות ושגרות הן דרך מצוינת לארגן את יישומי האינטרנט שלך. הן מאפשרות ליצור בלוקים של קוד לשימוש חוזר שבקלות ניתן לקרוא להם מדפים שונים של תוכנת האתר. מכיוון שקוד שתצטרך לו יותר מפעם אחת מוגדר פעם אחת בלבד, הרבה יותר קל לתחזק ולהרחיב אותו ללא חשש מכפילויות שאינן תואמות. אם תמצא באג בפונקציה, עליך לתקן אותו פעם אחת במחלקה תחת התיקייה App_Code, וכל הדפים שמשתמשים באותה פונקציה ייהנו מהשינוי באופן אוטומטי. פרט לתחזוקה משופרת, פונקציות ושגרות עושות את הקוד שלך קריא יותר: במקום לעבור על רשימות קוד ארוכות בדף, כל שעליך לעשות הוא לקרוא לפונקציה יחידה ולפעול על הערך המוחזר שלה (אם קיים). הדבר יעסיק אותך פחות, ויקטין את הסיכוי לתקלות ביישום.

פונקציות ושגרות אינן הדרך היחידה לארגון הקוד בפרויקטים שלך בסביבת .NET. דרך נפוצה אחרת לארגן דברים היא להשתמש במרחבי שמות (namespaces).

ארגון קוד בעזרת מרחבי שמות (namespaces)

נראה כי מרחבי שמות גורמים לכלבול רב בקרב מפתחים חדשים. הם סבורים שמרחבי השמות מפחידים, ושיש הרבה יותר מדי מהם, או שאינם רואים צורך להשתמש בהם. אף אחד מאלה אינו נכון, ועם הסבר קצר של הנושא תשנה דעתך עליהם.

מרחבי שמות נועדו לפתור שתי בעיות מרכזיות: לארגן את הכמות העצומה של פונקציונליות בסביבת .NET. ובקוד שלך עצמך, ולמנוע התנגשויות שמות כאשר שני סוגי נתונים הם בעלי אותו שם.

כדי לראות כיצד נראים 'מרחבי שמות', פתח את אחד מקבצי הקוד ברקע של דפי ASPX שיצרת עד כה. תראה משהו דומה לזה:

VB.NET

```
Partial Class Demos_CalculatorDemo
    Inherits System.Web.UI.Page
```

C#

```
public partial class Demos_CalculatorDemo : System.Web.UI.Page
{
```

שים לב שלאחר הגדרת שם המחלקה מופיעה מילת המפתח Inherits (VB) או נקודתיים בלבד (C#), ואחריה כתוב System.Web.UI.Page. מאוחר יותר תראה למה משמשת מילת המפתח Inherits. בקוד זה, Page הוא שם של מחלקה (סוג נתונים), שמוגדרת במרחב השמות

System.Web.UI.Page. על ידי הצבת המחלקה Page במרחב השמות System.Web.UI יכולים מפתחי התוכנה (וגם מהדרים) לראות שמחלקה זו היא בהקשר של דף web. לעומת זאת, אפשר להניח שיש לנו את שם המחלקה (הברוויה) הבאה:

```
Microsoft.Word.Document.Page
```

גם קוד זה מתייחס למחלקה Page. אבל, מכיוון שהיא מוצבת במרחב השמות (הברווי) Microsoft.Word.Document, קל לראות שהיא מתייחסת לדף של מסמך Word ולא לדף web. דבר זה עוזר למהדר להבין לאיזו מחלקה אתה פונה.

תועלת אחרת של מרחבי שמות היא שהם עוזרים למצוא את סוג הנתונים הנכון. במקום להציג אלפים רבים של פריטים ברשימה IntelliSense, אתה מקבל כמה מרחבי שמות מהרמה העליונה. כשאתה בוחר בפריט מרשימה זו ולוחץ על המקש נקודה (.) תקבל רשימה אחרת, קצרה יחסית, שכוללת סוגים ומרחבי שמות אחרים שנמצאים בתוך מרחב השמות שנבחר.

מרחבי שמות אינם אלא מכולות פשוטות שאתה יכול להתייחס אליהן לפי שם תוך שימוש בסימון נקודה. הם משמשים להוספת תחילית לכל סוג נתונים שזמין ליישום שלך. לדוגמה, סוג הנתונים Double נמצא במרחב השמות System ולפיכך שמו המלא הוא System.Double. באופן דומה, הפקד Button שהוספת לדפי web שלך נמצא במרחב השמות System.Web.UI.WebControls ולכן שמו המלא יהיה System.Web.UI.WebControls.Button.

קל ליצור מרחבי שמות משלך. כל עוד שמם אינו מתנגש עם שם קיים, אתה יכול ליצור מרחבי שמות ככל שתמצא לנכון. לדוגמה, אתה יכול לארוז את המחלקה Calculator במרחב השמות הבא (Calculator.vb או Calculator.cs שנמצאים במסגרת App_Code):

VB.NET

```
Namespace Wrox.Samples
```

```
Public Class Calculator
    ...
End Class
```

```
End Namespace
```

C#

```
namespace Wrox.Samples
```

```
{
    public class Calculator
    {
        ...
    }
}
```

כשהמחלקה Calculator ארוזה במרחב שמות זה, אתה יכול ליצור מופע חדש של המחלקה באופן זה:

VB.NET

```
Dim myCalculator As New Wrox.Samples.Calculator()
```

C#

```
Calculator myCalculator = new Wrox.Samples.Calculator();
```

הקלדת שמות ארוכים אלה הופכת לאחר זמן קצר למשעממת וגם מכבידה. למרבה המזל יש תיקון גם לזה.

יבוא מרחבי שמות

לאחר שאתה יוצר מרחבי שמות משלך, או אם ברצונך להשתמש במרחבי שמות קיימים, עליך לדאוג לזמינותם עבור הקוד שלך. עשה זאת עם מילת המפתח Import (VB.NET) או עם using (C#). לדוגמה, כדי לגרום למחלקה Calculator להיות זמינה לדף CalculatorDemo, אתה יכול להוסיף את מרחב השמות הבא לקוד שלך:

VB.NET

```
Imports Wrox.Samples
```

```
Partial Class Demos_Calculator  
    Inherits System.Web.UI.Page
```

C#

```
using Wrox.Samples;
```

```
public partial class Demos_Calculator : System.Web.UI.Page  
{
```

כאשר אתה מצהיר על מרחבי השמות בראש הקובץ, אתה יכול להגדיר מופע של המחלקה מבלי לציין את מרחב השמות.

אם אתה משתמש ב-C#, תראה בקוד ברקע של דף ASPX כמה משפטי using כברירת מחדל עבור מרחבי שמות כמו System.Web.UI.WebControls ו-System.Web.UI. אם אתה משתמש ב-VB.NET לא תראה התייחסויות אלו. באתר אינטרנט מבוסס VB.NET, מרחבי השמות שהם ברירת המחדל כלולים בקובץ web.config במסגרת האיבר <namespaces>.

ברגע שתתחיל לכתוב כמות מרובה של קוד, אתה עלול לשכוח במהירות איפה הצהרת מה, או למה משמשים משתנה או שיטה. על כן מומלץ בחום שתכתוב בקוד שלך הערות (c.pmmments).

כתיבת הערות

גם אם אתה כותב קוד נקי, סביר שיגיע רגע שבו תפגוש בקטע קוד שיגרום לך להרים גבה ולשאול את עצמך, "מה לעזאזל אמור קוד זה לעשות?" במשך השנים תשתנה הדרך שבה אתה מתכנת, תלמד חומר חדש, תשפר את רמת התכנות שלך ותמצא דרכים לקודד ביתר יעילות. ייתכן גם שהקוד שכתבת יועבר לתחזוקה על ידי מישהו אחר. כדי להקל עליך ועל אחרים לזהות ולהבין את הקוד שנכתב לראשונה או עודכן במהלך השנים, מומלץ לכתוב בו הערות. יש שתי דרכים עיקריות להוספת הערות לקבצי קוד: לכלול אותן בקוד או להוסיפן כהערות XML.

הערות כלולות בקוד

הערות כלולות (inline) נכתבות ישירות בין משפטי הקוד שלך. באפשרותך לכתוב בהן על משתנים קיימים, לולאות מורכבות, הסברים למשפטי התניה ועוד. ב-VB.NET אתה יכול לכתוב הערה בשורה אחת בכל פעם תוך שימוש בתו גרש (') שמוצב בראשית הטקסט כדי לציין שהכוונה להערה. כדי לכתוב הערת שורה יחידה ב-C# תשתמש בשני לוכסנים (//). בנוסף, ב-C# אתה יכול להשתמש ב-/* וב-*/ כדי להפוך בלוק של קוד להערה. הדוגמאות הבאות מראות כמה דרכים שונות לכתיבת הערות:

VB.NET

```
' Usage: explains the purpose of variables, statements and so on.
```

```
' Used to store the number of miles the user has traveled last year.
```

```
Dim distanceInMiles As Integer
```

```
' Usage: comment out code that's not used (anymore).
```

```
' In this example, SomeUnfinishedFunction is commented out.
```

```
' to prevent it from being executed.
```

```
' SomeUnfinishedFunction()
```

```
' Usage: End of line comments.
```

```
If User.IsInRole("Administrators") Then ' Only allow admins in this area
```

```
End If
```

C#

```
// Usage: explains the purpose of variables, statements and so on.
```

```
// Used to store the number of miles the user has traveled last year.
```

```
int distanceInMiles;
```

```
// Usage: comment out code that's not used (anymore).
```

```
// In this example, SomeUnfinishedFunction is commented out.
```

```
// to prevent it from being executed.
```

```
// SomeUnfinishedFunction();
```

```
// Usage: End of line comments.
```

```
if (User.IsInRole("Administrators")) // Only allow admins in this area
```

```
{ }
```

```
/*
```

```
 * This is a block of comments that is often used to add additional
```

```
 * information to your code for example to explain a difficult loop
```

```
*/
```

כדי לשלב הערות בקוד עליך להקליד את התו (// או ') במקום שבו תרצה שההערה תתחיל. כדי להפוך בלוק קוד להערה, סמן אותו בעורך הטקסט ואז הקש Ctrl+K ו-Ctrl+C. באופן דומה, כדי להפוך בלוק קוד מהערה לקוד ביצועי, סמן אותו בעורך הטקסט ואז הקש Ctrl+K ו-Ctrl+U.

לחילופין, באפשרותך לבחור בתפריט הראשי Edit ← Advanced ← Comment Selection, או
תוכל לבחור Edit ← Advanced ← Uncomment Selection. תוכל גם ללחוץ על הלחצנים
המתאימים בסרגל הכלים של Text Editor, כפי שמוצג בתרשים 5-5.



תרשים 5-5

הערות כלולות משמשות בדרך כלל לתיעוד פרטים קטנים הקשורים לקוד. יחד עם זאת מומלץ
לספק סקירה כוללת על תכלית הקוד. לדוגמה, עבור שיטה שנקראת sendMail טוב שיהיה
תיאור קצר שיסביר מה השיטה עושה ומה התפקיד של הפרמטרים השונים. לצורך זה נשתמש
בהערות XML.

כתיבת הערות XML

את הערות XML מוסיפים לקוד כאיברים של XML (תוך שימוש בסוגריים זוויתיים <>). תפקידם
לתאר את מטרת הקוד, פרמטרים, הערך המוחזר וכו'. ה-IDE של VWD עוזר לך בכתיבת הערות
אלו. כל שעליך לעשות הוא להציב את הסמן על השורה שקודמת למחלקה או לשיטה ולהקליד
''' (שלושה גרשים יחידים) ב-VB, או /// (שלושה לוכסנים) ב-C#. ברגע שתעשה זאת, IDE
יכניס תגיות XML עבור התיאור של השיטה, ולפי בחירה גם עבור הפרמטרים והערך המוחזר.
השיטה SendEmail מקבלת שני פרמטרים מסוג String: אחד עבור כתובת הדואר האלקטרוני
שאליה תישלח ההודעה, והשני עבור גוף ההודעה. אם ניישם הודעות XML לשיטה, היא עשויה
להיראות כך:

VB.NET

```
''' <summary>
''' Sends out an e-mail to the address specified by emailAddress.
''' </summary>
''' <param name="emailAddress">The e-mail address of the addressee.</param>
''' <param name="mailBody">The body of the mail message.</param>
''' <returns>This method returns True when the message was sent successfully;
''' and False otherwise.</returns>
''' <remarks>Attention: this method assumes a valid mail server is available.</remarks>
Public Function SendEmail(ByVal emailAddress As String, ByVal mailBody As String) _
    As Boolean
    ' Implementation goes here
End Function
```

C#

```
/// <summary>
/// Sends out an e-mail to the address specified by emailAddress.
/// </summary>
```

```

/// <param name="emailAddress">The e-mail address of the addressee.</param>
/// <param name="mailBody">The body of the mail message.</param>
/// <returns>This method returns true when the message was sent successfully;
/// and false otherwise.</returns>
/// <remarks>Attention: this method assumes a valid mail server is available.</remarks>
bool SendEmail(string emailAddress, string mailBody)
{
    // Implementation goes here
}

```

מה שיפה בדרך זו של כתיבת הערות הוא שההערות שאתה מקליד כאן יופיעו ב-IntelliSense בעורך הקוד כשתנסה לקרוא לשיטה (ראה תרשים 5-6).

18	
19	bool result = SendEmail(
20	bool Demos_Test.SendEmail (string emailAddress, string mailBody)
21	emailAddress:
22	The e-mail address of the addressee.
23	

תרשים 5-6

יסודות תכנות מוכוון אובייקטים

פרק בנושא כתיבת קוד ב-ASP.NET לא יהיה שלם ללא סעיף בנושא תכנות מוכוון אובייקטים (Object Oriented), סגנון תכנות פופולרי מאוד, שבו התוכנה מעוצבת כקבוצת אובייקטים שיש ביניהם קשרים ויכולת עבודה משותפת. הכוונת אובייקטים היא בלבה של סביבת ASP.NET. פשוטו כמשמעו, כל דבר בתוך הסביבה הוא אובייקט, החל מדברים פשוטים כמו מספרים טבעיים ועד דברים מורכבים כמו פקד DropDownList, קישור לבסיס נתונים, או פקד מונע-נתונים.

מכיוון שהכוונת אובייקטים היא היבט כה חשוב של .NET, חשוב להכיר את המושגים הכלליים של תכנות מוכוון אובייקטים. בו-בזמן, אינך חייב להיות מומחה בנושא כדי שתוכל לבנות אתרי web בסביבת ASP.NET. סעיף זה נותן מבט-על על המונחים והמושגים החשובים ביותר. הדבר יעזור לך לעסוק ולהפעיל סוג סגנון תכנות זה בזמן קצר ביותר, וכך תוכל להתחיל לבנות יישומים שימושיים בפרק הבא, במקום לתקוע את האף בספרים במשך שלושת השבועות הבאים לשם לימוד הנושא.

מונחי OO חשובים

ב-OO הכל סובב סביב מושג האובייקט. למעשה, ב-OO כל דבר הוא אובייקט. אבל מהו בדיוק אובייקט? ומה צריכות מחלקות לעשות עם אובייקטים?

אובייקטים

אובייקטים (objects) הם אבני הבניין היסודיות של שפות תכנות מוכוון אובייקטים. כמו בעולם הממשי, אובייקט בעולם OO הוא דבר. הוא יכול להיות משתנה Integer שמחזיק את הגיל של

מישהו, או חיבור פתוח לבסיס נתונים בשרת SQL שנמצא בצד השני של כדור הארץ, אבל הוא יכול להיות גם משהו שהוא מושג, כמו דף web. ביישומים אתה יוצר אובייקט חדש בעזרת מילת המפתח New (ב-C#), כפי שראית בדוגמת המחשבון. הדבר חל על סוגים מורכבים או מותאמים אישית כמו Calculator, אבל גם על סוגים פשוטים כמו Integers ו-Strings:

VB.NET

```
Dim myCalculator As New Calculator()  
  
Dim age As Integer = New Integer()
```

C#

```
Calculator my Calculator = new Calculator()  
  
int age = new int();
```

מכיוון שכה נפוץ ליצור משתנים מסוגים פשוטים כמו Integer (ב-C#) ו-String (ב-C#), המהדר מאפשר להשמיט את מילת המפתח new. לכן, הקוד הבא שקול מבחינה פונקציונלית להכרזה הקודמת של age:

VB.NET

```
Dim age As Integer
```

C#

```
int age;
```

את כל סוגי הנתונים שמופיעים בתחילת פרק זה, פרט ל-System.Object, אפשר ליצור ללא שימוש במילת המפתח New. ברגע שיצרת אובייקט (מופע של מחלקה), כמו האובייקט myCalculator, הוא מוכן לשימוש. לדוגמה, כדי לעשות משהו שימושי באובייקט, אתה יכול לגשת לשיטות ולמאפיינים שלו. אבל לפני שתסתכל בשיטות ומאפיינים, עליך להבין תחילה מהן מחלקות.

מחלקות

מחלקה (class) היא כעין תוכנית/שרטוט של האובייקט. כפי שאתה יכול לבנות קבוצת בתים על-פי תוכנית אחת, אתה יכול להשתמש במחלקה אחת כדי ליצור כמה מופעים שלה. כלומר, המחלקה פועלת כהגדרה של האובייקטים שבהם אתה משתמש ביישום שלך. בצורתה הבסיסית ביותר מחלקה נראית כך:

VB.NET

```
Public Class ClassName  
  
End Class
```

C#

```
public class ClassName  
{  
  
}
```

מכיוון שקוד זה מגדיר מחלקה ריקה בלבד, הוא אינו יכול לעשות משהו שימושי. כדי לתת לו איזושהי אפשרות פעולה, אתה יכול לקבוע לו מאפיינים, שיטות ובנאים (constructors). בנוסף, ביכולתך לאפשר למחלקה לרשת ממחלקה קיימת כדי לתת לה את הבסיס ההתחלתי במונחי פונקציונליות והתנהגות. מונחים אלה יהיו מובנים יותר במהלך ההסברים להלן.

מאפיינים

מאפיינים (properties) של אובייקט הם אוסף התכונות המזהות שיש לאובייקט. לדוגמה האובייקט Person. איזה סוג של מאפיינים יש ל-Person? קל לתת תכונות מזהות רבות ושונות, אבל הנפוצות והברורות ביותר הן:

- ⊙ שם פרטי
- ⊙ שם משפחה
- ⊙ תאריך לידה

מאפיין במחלקה מוגדר עם מילת המפתח Property (VB.NET) או עם שם המאפיין, בדומה לשיטה, ב-C#. בשתי השפות תשתמש ב-Set (C#-ב) וב-Get (C#-ב) כדי לגשת למאפיין האובייקט יש להגדיר את מה שקרוי getters ו-setters של מאפיין בהתאמה. ניגשים אל ה-getter כאשר האובייקט נשאל לגבי הערך של מאפיין מסוים, בעוד ה-setter משמש להקצאת ערך למאפיין. מאפיינים מספקים גישה רק לנתונים בסיסיים שמאוחסנים באובייקט; הם אינם מכילים את הנתונים הממשיים. כדי לאחסן את הנתונים דרוש **משתנה גיבוי** (backing variable). זהו משתנה פשוט שמוגדר במחלקה, שיכול לאחסן את הערך עבור המאפיין החיצוני. המשתנה _firstName בדוגמה הבאה הוא משתנה הגיבוי עבור המאפיין firstName:

VB.NET

```
Public Class Person
    Private _firstName As String
    Public Property FirstName() As String
        Get
            Return _firstName
        End Get
        Set(ByVal value As String)
            _firstName = value
        End Set
    End Property
End Class
```

C#

```
public class Person
{
    private string _firstName;
    public string FirstName
    {
        get { return _firstName; }
        set { _firstName = value; }
    }
}
```

בדרך כלל תחביר משתנה הגיבוי הפרטי מוגדר בקו תחתון () ולאחריו המילה הראשונה כולה באותיות קטנות, כשלאחריה מילים נוספות שמתחילות באות רישית. כך, למאפיין `FirstName` יש משתנה גיבוי בשם `_firstName`, למאפיין `LastName` יש משתנה גיבוי בשם `_lastName`, וכך הלאה.

הסיבה העיקרית למאפיין במחלקה היא **כימוס** (encapsulation) של נתונים. הרעיון הוא שהמאפיין מאפשר לך לשלוט בנתונים שהוקצו לו. בדרך זו תוכל לבדוק תקפות (אימות) של הנתונים או לבצע עליהם פעולות כלשהן לפני שיאוחסנו במשתנה הגיבוי הבסיסי. תוכל להניח שאחד הכללים העסקיים של היישום שלך אומר שכל השמות הפרטיים חייבים להיכתב עם אות רישית. בשפה שאינה מוכוונת אובייקטים המפתח שמגדיר את השם יצטרך 'לחשוב' על הכלל בכל פעם שמוכנס שם פרטי למשתנה. בגישת OO אתה יכול לגרום לכך שהמאפיין `FirstName` יהיה אחראי לכלל זה, כך שאחרים לא יצטרכו עוד לדאוג לגביו. ביכולתך לבצע סוג זה של מניפולציה על נתונים ב-setter של המאפיין:

VB.NET

```
Set(ByVal value As String)
```

```
    If Not String.IsNullOrEmpty(value) Then  
        _firstName = value.Substring(0, 1).ToUpper() & value.Substring(1)  
    End If
```

```
End Set
```

C#

```
set
```

```
{
```

```
    if (!string.IsNullOrEmpty(value))  
    {  
        _firstName = value.Substring(0, 1).ToUpper() + value.Substring(1);  
    }  
}
```

קוד זה מדגים שהפרמטר `value` נגיש, הן ב-VB.NET והן ב-C#, בדיוק כמו פרמטר לשיטה. ב-VB.NET מוגדר הפרמטר `value` במפורש ב-setter של המאפיין. ב-C# הוא אינו מוגדר במפורש, אבל הוא נגיש בכל זאת.

תוך שימוש בשיטה הנוחה `String.IsNullOrEmpty` הקוד בודק תחילה אם `value` שמועבר אינו `Nothing` (null ב-C#), ושאינו מכיל מחרוזת ריקה.

כדי לקבל את האות הראשונה של `value`, משתמש הקוד בבלוק `If` בשיטה `SubString` של המחלקה `String`, שאליה הוא מעביר את הערכים 0 ו-1. 0 מציין את תחילת ה-substring בעוד '1' מציין את אורך המחרוזת שיש להחזיר. האינדקס של מחרוזות גם הוא מבוסס אפס, כך שהתחלה מ-0 ואורך של '1' מחזירים למעשה את התו הראשון של הפרמטר `value`. תו זה מוחלף כעת באות רישית באמצעות השיטה `ToUpper()`. לבסוף, הקוד לוקח את שארית הפרמטר `value`, כשהוא משתמש שוב ב-`SubString`, ומקצה את השם המאוחד חזרה למשתנה הגיבוי.

אתה יכול עכשיו להשתמש בקוד שמגדיר שם מבלי להתחשב באות רישית. אבל, כשתנסה לגשת לשם שוב, יופיע השם הראשון תמיד עם אות ראשונה רישית.

VB.NET

```
Dim myPerson As New Person()      ' Create a new instance of Person
myPerson.FirstName = "krudo"       ' Accessing the setter that changes the value

Label1.Text = myPerson.FirstName   ' Accessing the getter that now returns Krudo
```

C#

```
Person myPerson = new Person();    // Create a new instance of Person
myPerson.FirstName = "krudo";       // Accessing the setter that changes the value

Label1.Text = myPerson.FirstName;   // Accessing the getter that now returns Krudo
```

עבור מאפיינים פשוטים, שאינם צריכים לבצע בדיקת תקפות של נתונים או פעולות שונות (מניפולציות) על נתונים, אתה יכול להשתמש (רק ב-C#) במה שנקרא מאפיינים אוטומטיים. עם מאפיינים אלה אתה יכול להשתמש בתחביר דחוס הרבה יותר מבלי להזדקק למשתנה גיבוי פרטי. כשהקוד מהודר, המהדר יצור עבורך את משתנה הגיבוי הנדרש, אבל לא תוכל לגשת אליו מהקוד שלך. הנה המאפיין DateOfBirth של Person כתוב כמאפיין אוטומטי (ב-C#):

```
public DateTime DateOfBirth { get; set; }
```

אם בשלב מאוחר יותר תחליט שברצונך לכתוב קוד ב-getter או ב-setter של המאפיין, קל להרחיב את בלוקי הקוד הרלוונטיים בלי לשבור את היישום הקיים. עד אז יש בידך הגדרות מאפיין פשוטות ונקיות במחלקה.

יצירת מאפיינים לקריאה-בלבד ולכתיבה-בלבד

יש מצבים שבהם מאפייני קריאה-בלבד (read-only) או כתיבה-בלבד (write-only) הגיוניים לחלוטין. לדוגמה, ID של אובייקט יכול להיות לקריאה-בלבד אם הוא מוקצה על ידי בסיס הנתונים באופן אוטומטי. כאשר האובייקט נבנה מבסיס הנתונים, מוקצה ID למשתנה הגיבוי הפרטי. מאפיין ID הציבורי הופך אז לקריאה-בלבד, כדי למנוע מקור קורא לשנותו בטעות. באופן דומה, ייתכן שמסיבות אבטחה יהיה מאפיין כתיבה-בלבד. לדוגמה, אפשר שיהיה מאפיין Password לאובייקט Person, שאם אתה יודע אודותיו תוכל רק להקצות לו, אבל לא תוכל אחר כך לקרוא אותו. הקוד שבתוך המחלקה יוכל עדיין לגשת למשתנה הגיבוי כדי להשתמש בערך של הסיסמה.

קל ליצור מאפיינים לקריאה-בלבד או לכתיבה-בלבד ב-C#: לשם כך צריך לוותר על הגדרת ה-setter (עבור מאפיין קריאה-בלבד) או על ה-getter (עבור מאפיין כתיבה-בלבד). VB.NET היא מעט יותר מילולית ודורשת שתציין את מילות המפתח ReadOnly או WriteOnly במפורש. פיסת הקוד הבאה מראה מאפיין ID לקריאה-בלבד, בשתי השפות:

VB.NET

```
Private _id As Integer

Public ReadOnly Property Id() As Integer
    Get
        Return _id
    End Get
End Property
```

C#

```
private int _id;

public int Id
{
    get { return _id; }
}
```

אם תנסה להקצות ערך למאפיין שמיועד לקריאה-בלבד, VWD תוציא הודעת שגיאה. כפי שיש לאובייקטים מאפיינים, כך יש להם גם שיטות.

שיטות

אם מאפיינים הם דברים שיש למחלקה (התכונות המזהות שלה), אזי שיטות (methods) הן הפעולות שמחלקה יכולה לבצע. למכונת יש תכונות מזהות כמו Model, Brand ו-Color. השיטות שלה יכולות להיות Drive(), Brake() ו-OpenDoors(). שיטות נותנות לאובייקטים את ההתנהגות שמאפשרת להם לעשות משהו.

כבר ראית שיטות בפעולה כשדנו לעיל בפרק זה בכמה דרכים לכתיבת קוד מאורגן. כל הכללים שחלים על פונקציות ועל שגרות חלים גם כאן. אתה מוסיף שיטות למחלקה על ידי כתיבת פונקציה או שגרה בין איבר הפתיחה ואיבר הסיום של המחלקה. לדוגמה, תוכל להניח שלמחלקה Person יש שיטה Save שמאפשרת לאובייקט לשמור את נתוני המחלקה בבסיס הנתונים. חתימת השיטה עשויה להיראות כך:

VB.NET

```
Public Class Person
    Public Sub Save()
        ' Implementation goes here
    End Sub
End Class
```

C#

```
public class Person
{
    public void Save()
    {
        // Implementation goes here
    }
}
```

אם ברצונך לקרוא לשיטה Save, צור מופע של האובייקט Person, הגדר את המאפיינים הרלוונטיים כמו FirstName ואז קרא ל-Save:

VB.NET

```
Dim myPerson As New Person()  
  
myPerson.FirstName = "Joop"  
myPerson.Save()
```

C#

```
Person myPerson = new Person();  
  
myPerson.FirstName = "Joop";  
myPerson.Save();
```

השיטה Save תדע כיצד לשמור את Person בבסיס הנתונים.

שים לב שמופע חדש של המחלקה Person נוצר בעזרת מילת המפתח New (ב-C#) ולאחריה שם המחלקה. כשקור זה יתבצע, הוא יקרא לפונקציית הבנאי של האובייקט, אשר משמשת לבניית מופעים של אובייקטים.

פונקציות בנייה (בנאים)

פונקציות בנייה (constructors) הן שיטות מיוחדות במחלקה שעוזרות ליצור מופע של אובייקט. הן מתבצעות באופן אוטומטי ברגע שאתה מנסה ליצור מופע של מחלקה, וכך הן מהוות מקום מצוין לאתחל את האובייקטים שלך למצב ברירת המחדל שלהם. קודם לכן למדת ליצור מופע חדש של אובייקט על ידי שימוש במילת המפתח New (ב-C#):

VB.NET

```
Dim myCalculator As New Calculator()
```

C#

```
Calculator myCalculator = new Calculator();
```

פונקציית הבנייה של האובייקט מופיעה לאחר המילה New: שם המחלקה. כשאתה יוצר מחלקה חדשה ב- Visual Web Developer, אתה מקבל כברירת מחדל פונקציית בנייה עבור C# אבל לא עבור VB.NET. זו אינה בעיה רצינית, מכיוון שהמדר ייצור עבורך פונקציית בנייה של ברירת מחדל אם תשאיר זאת לטיפולו. לפונקציית הבנייה של ברירת המחדל אין ארגומנטים. ב-C# היא נקראת על שם המחלקה, וב-VB.NET משתמשים במילת המפתח השמורה New:

VB.NET

```
Public Class Person  
    Public Sub New()  
  
    End Sub  
End Class
```

C#

```
public class Person
{
    public Person()
    {

    }
}
```

למרות שבנאי ברירת המחדל הוא נחמד ליצירת מופעים סטנדרטיים של המחלקות שלך, יש פעמים שבהם ממש שימושי לשלוח מידע אל המחלקה באופן ישיר, כדי שיהיה זמין מייד עם סיום בניית המחלקה. לדוגמה, לגבי המחלקה Person, די שימושי להעביר לבנאי את השם הפרטי, שם המשפחה ותאריך הלידה. כך הנתונים יהיו זמינים מייד לאחר מכן. כדי ליישם אפשרות זו, ניתן להשתמש בהעמסת **בנאי** (overloaded). העמסת בנאי, היא למעשה עותק של שיטה קיימת באותו שם בדיוק, אבל עם חתימת שיטה אחרת. כדי לגרום לבנאי לקבל את השמות ותאריך הלידה יש לכתוב את הקוד הבא:

VB.NET

```
Public Sub New(ByVal firstName As String, ByVal lastName As String, _
                ByVal dateOfBirth As DateTime)
    _firstName = firstName
    _lastName = lastName
    _dateOfBirth = dateOfBirth
End Sub
```

C#

```
public Person(string firstName, string lastName, DateTime dateOfBirth)
{
    _firstName = firstName;
    _lastName = lastName;
    _dateOfBirth = dateOfBirth;
}
```

עם קוד זה אתה יכול ליצור אובייקט Person חדש:

```
Person myPerson = new Person("Meir", "Krudo", new DateTime(1975, 3, 2));
```

האובייקט myPerson מאותחל במלואו מייד לאחר שורת קוד זו, ולכל מאפייניו יש כעת ערכים.

בנוסף לבנאי מועמס מציעה NET 3.5. דרך מהירה נוספת ליצירת אובייקט ולאתחול כמה מאפיינים: **מאתחלי אובייקט** (object initializers). עם מאתחל אובייקט אתה מספק את הערכים ההתחלתיים עבור כמה מהמאפיינים באותו הזמן שאתה מכריז על מופע של האובייקט. הקוד הבא יוצר אובייקט Person ומקצה לו ערכי ברירת מחדל עבור המאפיינים FirstName ו-LastName:

VB.NET

```
Dim myPerson As New Person() With {.FirstName = "Meir", .LastName = "Krudo"}
```

C#

```
Person myPerson = new Person() { FirstName = "Meir", LastName = "Krudo" };
```

ב-VB.NET אתה צריך את מילת המפתח With בראשית רשימת המאפיינים. בנוסף, עליך להקדים כל שם של מאפיין עם נקודה (.). פרט לכך התחביר עבור שתי השפות דומה למדי. מאתחלי אובייקט הם מצוינים אם צריך להגדיר במהירות קבוצה של מאפיינים באובייקט, מבלי שתצטרך לכתוב גרסאות מיוחדות של בנאים מועמסים.

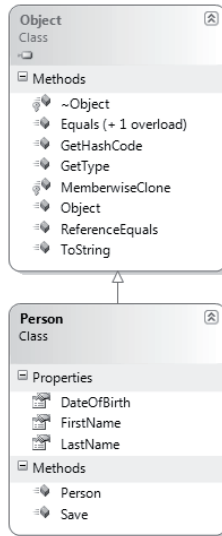
נוכחות המחלקה Person ביישום שלך היא שימושית, אבל יש פעמים שבהן תצטרך גרסאות מיוחדות שלה. לדוגמה, היישום שלך עשוי להזדקק למחלקות כמו Employee ו-Student. מה תעשה במקרה זה? האם תיצור שני עותקים של המחלקה Person ותקרא להם Employee ו-Student, בהתאמה?

למרות שגישה זו בוודאי תפעל, יש לה כמה מגרעות רציניות. הבעיה הגדולה ביותר היא הקוד הכפול. אם תחליט להוסיף מאפיין שיחזיק את מספר ת"ז, תצטרך עכשיו להוסיפו בכמה מקומות: במחלקה הכללית Person, ובמחלקות Employee ו-Student. **ירושת אובייקט** (object inheritance), עמוד תמיכה מרכזי של OO, תוכננה לפתור בעיות מסוג זה.

הורשה

קודם למדת שהאובייקט System.Object הוא האב של כל סוגי הנתונים האחרים ב-.NET, ובכלל זה כל הסוגים המובנים וסוגים שאתה מגדיר בעצמך. כלומר, כל סוג ב-.NET (פרט ל-Object עצמו) יורש (inherits) מ-Object. אחת התועלות של **הורשה** (Inheritance) היא שאתה יכול להגדיר התנהגות ברמה גבוהה (למשל במחלקה Object) והיא זמינה באופן אוטומטי למחלקות שירשות ממנה, מבלי שיהיה צורך לשכפל את הקוד. בסביבת .NET. המחלקה Object מגדירה כמה שיטות אשר מחלקות יורשות עושות בהן שימוש, כמו לדוגמה השיטה ToString().

למחלקה Person יש כמה מאפיינים כמו FirstName, LastName והשיטה Save. אבל אם היא יורשת מ-Object, האם יש לה גם את השיטה ToString()? התשובה חיובית. תרשים 5-7 מראה את היחסים בין המחלקה Object והמחלקה Person שירשת מ-Object.



תרשים 5-7

תרשים 5-7 מראה ש-Person יורשת מ-Object (הקו עם ראש חץ מעליו מצביע בכיוון המחלקה שממנה יורשים). משמעות הדבר היא שמופע של Person יוכל לעשות כל מה ש-Object יכול. כך, לדוגמה, אתה יכול לקרוא לשיטה ToString() מהאובייקט Person:

```
Label1.Text = myPerson.ToString() ' Writes out Person
```

התנהגות ברירת המחדל של השיטה ToString() שמוגדרת ב-Object היא להציג את השם שלה. בדוגמה לעיל, פירוש הדבר הוא שהמחלקה Person יורשת התנהגות זו ולפיכך אומרת את המילה Person כשמה. בדרך כלל התנהגות ברירת מחדל זו אינה מספיקה, ויהיה זה שימושי הרבה יותר אם Person תוכל, למשל, להחזיר את השם המלא של האדם שהיא מייצגת. אפשר להשיג זאת בקלות על ידי דריסת (overriding) השיטה ToString(). דריסת שיטה או מאפיין מגדירה למעשה מחדש את ההתנהגות שהמחלקה יורשת ממחלקת האב שלה. כדי לדרוס שיטה תשתמש במילת המפתח Overrides (VB.NET), או override ב-C#. הקוד הבא מגדיר מחדש את התנהגות ToString במחלקה Person:

VB.NET

```
Public Overrides Function ToString() As String
    Return _firstName & " " & _lastName
End Function
```

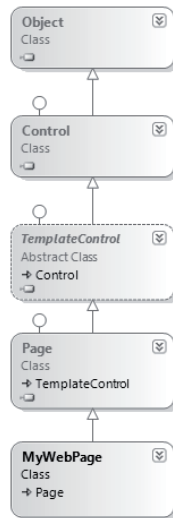
C#

```
public override string ToString()
{
    return _firstName + " " + _lastName;
}
```

עם הגדרה זו של התנהגות ToString במחלקה Person, המחלקה אינה מחזירה עוד את המילה Person, אלא את השם המלא של האדם שהיא מייצגת:

```
Label1.Text = myPerson.ToString() ' Writes out Meir Krudo
```

מנגנון הורשת אובייקטים ב-NET מאפשר לך ליצור היררכיה של אובייקטים שמשפרת אובייקטים אחרים, או מוסיפה להם פונקציונליות. הדבר מאפשר להתחיל עם מחלקה בסיסית גנרית (בשם Object). מחלקות אחרות יוכלו לרשת ממחלקה זו כשהן מוסיפות התנהגות מיוחדת לעצמן. אם אתה צריך מחלקות מיוחדות עוד יותר, אתה יכול לרשת שוב מהמחלקה שירשה מ-Object. על ידי כך תיצור היררכיה של מחלקות שהולכות ונעשות ייחודיות יותר. עיקרון זה פועל עבור מחלקות רבות בסביבת NET, כולל המחלקה Page. ייתכן שלא הבנת זאת, אבל כל דף ASPX שיצרת ב-VWD הוא למעשה מחלקה שיורשת מהמחלקה System.Web.UI.Page. מחלקה Page יורשת מהמחלקה TemplateControl, שיורשת מ-Control שיורשת מ-Object. ההיררכיה כולה מוצגת בתרשים 5-8. בתחתית התרשים נמצאת המחלקה MyWebPage שיכולה להיות דף web רגיל באתר שלך, בשם MyWebPage.aspx.



תרשים 5-8

בתרשים 5-8 ניתן לראות ש-TemplateControl היא מחלקה מופשטת (abstract class) - מחלקה שלא ניתן ליצור מופע שלה באמצעות מילת המפתח new (ב-C#). היא משמשת אך ורק כבסיס משותף למחלקות אחרות (כמו Page) שיכולות לרשת ממנה. המחלקות שבין Page ו-Object אינן ממש רלוונטיות בשלב זה. מה שחשוב הוא שהדף שלך יורש את כל ההתנהגות שיש למחלקה Page. העובדה שכל דפי ASPX שלך יורשים מ-Page הינה שימושית יותר מכפי שאתה עשוי לחשוב ברגע הראשון. מכיוון שהדפים יורשים מ-Page, אתה מקבל בחינם שיטות ומאפיינים רבים שמוגדרים במחלקה זו. לדוגמה, המחלקה Page חושפת מאפיין Title שכאשר הוא מוגדר הוא מופיע כאיבר <title> בדף. עליך רק להגדיר מאפיין זה, ומחלקת האב Page תטפל בכל השאר.

VB.NET

```
Title = "Beginning ASP.NET 3.5 by Wrox"
```

C#

```
Title = "Beginning ASP.NET 3.5 by Wrox";
```

כדי ליצור מחלקה שיורשת ממחלקה אחרת תשתמש במילת המפתח Inherits ב-VB.NET, ובנקודתיים (:)-ב-C#. התבונן פעם נוספת בהגדרת המחלקה שראית קודם כאשר דנו במרחבי שמות:

VB.NET

```
Partial Class Demos_CalculatorDemo  
Inherits System.Web.UI.Page
```

C#

```
public partial class Demos_CalculatorDemo : System.Web.UI.Page
```

קוד זה מגדיר מחלקה בשם Demos_CalculatorDemo שיורשת מהמחלקה System.Web.UI.Page. ניתן לומר ש-Demos_CalculatorDemo היא גרסה מיוחדת של Page, מכיוון שהיא יכולה לעשות כל דבר שעושה Page, ולהוסיף לכך התנהגות משלה.

בפרק הבא תשתמש בהורשה כשתיצור את המחלקה BasePage שמשמשת כמחלקת אב לכל מחלקות קוד ברקע (Code Behind) שלך.

בדוגמאות קודמות, ראית את מילת המפתח Public. בנוסף, כשיצרנו משתני גיבוי, ראית את מילת המפתח Private. מילות מפתח אלו נקראות **מגבילי גישה** (access modifiers) וקובעות את מידת החשיפה והגישה (נראות) של הקוד.

מגבילי גישה (Access Modifiers)

מוקדם יותר בפרק נאמר שאחד ממושגי היסוד של תכנות מוכוון אובייקטים הוא **כימוס**, או **כמיסה** (encapsulation). על ידי יצירת פונקציות ומאפיינים אתה גורם לאובייקט להיות אחראי לביצוע. אובייקטים אחרים, שפועלים בשיתוף עם אובייקט זה, מתייחסים לשיטות ומאפיינים אלה כאל "קופסאות שחורות". כלומר, הם מעבירים נתונים פנימה ומצפים לקבל חזרה איזו שהיא תוצאה (לא בהכרח תמיד). לאובייקטים אלה אין עניין בדרך שבה מבצעת השיטה את פעולתה; די להם בכך שהיא פועלת. מכאן, שכדי לאפשר לאובייקט להגן על כמה מפעולותיו הפנימיות עליך לשלוט בגישה אל הקוד שלו. הדבר מתבצע על ידי ציון של רמת גישה לפני שם המחלקה, המאפיין או השיטה. הטבלה הבאה מציגה את רמות הגישה הזמינות ב-C# ו-VB.NET, ומסבירה את שימושיהן.

C#	VB.NET	תיאור
public	Public	המחלקה, השיטה, המשתנה נגישים מכל מקום, כולל מקוד שמחוץ ליישום הנוכחי.

C#	VB.NET	תיאור
protected	Protected	קוד עם רמת גישה זו זמין רק מתוך הסוג שמגדיר אותו או מתוך הסוגים שיוורשים ממנו. לדוגמה, שיטה שמוגדרת Protected במחלקה Page, נגישה לדף ASPX שלך מכיוון שהדף יורש מ-Page.
internal	Friend	הנגישות לקוד מוגבלת רק לקוד אחר שנמצא באותו מאספ (assembly). מאספ זו קבוצה של אחד, או יותר, קבצי קוד מהודרים (קובץ exe או קובץ dll) שמכילים קוד NET. שאפשר להשתמש בו שימוש חוזר.
private	Private	חלק קוד שנגיש רק מתוך הסוג שמגדיר אותו. לדוגמה, המשתנה firstName_ נגיש רק מתוך המחלקה Person. לקוד אחר, כמו דף ASPX, אין נגישות כלשהי לשדה זה, ועליו לגשת למאפיין הציבורי firstName כדי לקבל או להגדיר את השם הפרטי של אדם כלשהו.

מבין ארבעת מגבילי הגישה האלה ניתן לאחד רק את protected ואת internal (Protected ו-Friend ב-VB). בכל האחרים חייבים להשתמש בנפרד. על ידי איחוד של internal-protected ו-internal אפשר ליצור קוד הנגיש על ידי המחלקה הנוכחית וכל מחלקה שיוורשת ממנה, בתנאי שהיא נמצאת במאספ הנוכחי.

כמו עם כמה מושגים אחרים של OO, לא תבזבז ימי עבודה כדי לציין מגבילי גישה בקוד שלך. עם זאת, טוב לדעת שהם קיימים ומה יכולותיהם. בדרך זו יהיה לך רמז מדוע לפעמים המחלקות שלך אינן מופיעות ברשימה של IntelliSense. במקרה כזה סביר להניח ששכחת לציין במחלקה את מגביל הגישה Public. ברירת המחדל היא internal (Friend ב-VB.NET), שגורם לכך שהמחלקה תיראה על ידי מחלקות אחרות באותו מאספ, אבל מסתיר אותה מקוד שמחוץ למאספ. בעיה זו תיפתר על ידי הוספת מילת המפתח Public בחזית הגדרת המחלקה.

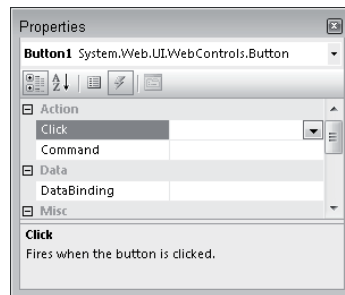
אירועים

הנושא החשוב האחרון שיש לדון בו בפרק זה הוא אירועים (events). ASP.NET היא סביבה מונעת אירועים (event driven). כלומר, הקוד של האובייקט יכול להתבצע רק בתגובה לאירועים מסוימים שקורים בקוד שלך. אובייקטים מסוימים ביישום מעוררים אירועים, ואלה מטופלים על ידי אובייקטים אחרים. בסביבת ASP.NET קיימים אובייקטים רבים שמסוגלים לעורר אירוע, ואתה יכול אפילו להוסיף אירועים משלך למחלקות שאתה כותב.

כדי להיות מסוגל לטפל באירוע שהועלה על ידי אובייקט, אתה צריך לכתוב מטפל אירוע (event handler), שהוא למעשה שיטה רגילה עם חתימה מיוחדת. אתה יכול לקשר בין מטפל האירוע לבין האירוע תוך שימוש בתחביר לקישור האירוע, אם כי בדרך כלל VWD דואגת לטפל בזה. כשאובייקט, למשל פקד בדף web, מעורר אירוע, ייתכן שהוא יצטרך להעביר מידע נוסף למטפל האירוע כדי ליידע אותו בדבר נתונים רלוונטיים שגרמו או השפיעו על האירוע.

אפשר לשלוח מידע זה תוך שימוש במחלקת הארגומנטים של אירוע, שהיא המחלקה System.EventArgs, או כל מחלקה שיורשת ממנה.

כדי לראות כיצד כל המונחים האלה משתבצים יחד, שקול מה קורה כשאתה לוחץ על לחצן בדף web. כשאתה לוחץ עליו, לחצן צד הלקוח בדפדפן גורם ל-postback. הפקד Button בשרת רואה שלחצו עליו בדפדפן ומעורר את האירוע Click שלו. כאילו שהלחצן אמר: "הו, ראו כולכם, זה עתה לחצו עלי. למקרה שמישהו מעוניין, הנה כמה פרטים." בדרך כלל, הקוד שמעוניין באירוע Click של הלחצן הוא הדף שלך, שצריך לכלול מטפל אירוע שיודע לקלוט את פעולת הלחיצה. באפשרותך ליצור מטפל אירוע עבור Button על ידי לחיצה כפולה עליו במעצב. לחילופין, תוכל ללחוץ לחיצה כפולה על האירוע הרלוונטי ברשימת הקטגוריה Events שבטבלת המאפיינים של הפקד (ראה תרשים 5-9). תוכל לפתוח את רשימת הקטגוריה על ידי לחיצת הלחצן עם חץ הברק בסרגל הכלים.



תרשים 5-9

אם תלחץ לחיצה כפולה על הפקד ב- Design View, או על שם האירוע ברשת המאפיינים, תראה ש-VWD תכתוב עבורך את קוד מטפל האירוע. קטע הקוד הבא מראה את המטפל ב-VB.NET וב-C#:

VB.NET

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Button1.Click  
End Sub
```

C#

```
protected void Button1_Click(object sender, EventArgs e)  
{  
}  
}
```

בדוגמת VB.NET אתה רואה שיטה רגילה עם כמה ארגומנטים, ואחריה Handles Button1.Click. זהו קוד של קישור האירוע שמלכד את האירוע Click של הפקד Button עם השיטה Button1_Click. כעת, בכל פעם שהלחצן יילחץ, יבוצע הקוד שבתוך Button1_Click.

בגרסת C# לא תראה את מילת המפתח Handles. במקום זאת תמצא שם ש-VWD הוסיפה קוד (מוצג בהדגשה) לפקד Button באזור סימני העריכה:

```
<asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click" />
```

עם קטע זה של סימני עריכה, NET runtime. ייצור את הקוד הנחוץ לקישור השיטה Button1_Click לאירוע Click של הלחצן. ב-runtime תראה אותה התנהגות בדיוק: כשלוחצים על הלחצן מבוצע הקוד של השיטה Button1_Click.

אתה יכול גם לראות שלמטפל האירוע Button1_Click יש שני פרמטרים: Object בשם sender ומחלקה EventArgs שנקראת e. זוהי סכמת מתן שמות תקנית ב-.NET, והיא מיושמת על ידי כל האובייקטים שיוצרים אירועים. הפרמטר sender מכיל התייחסות לאובייקט שעורר את האירוע, שנקרא Button1 במקרה שלנו. עובדה זו מאפשרת למצוא מי עורר אירוע, במקרה שאותו מטפל אירוע קשור למספר אירועים.

הפרמטר השני הוא מופע של המחלקה EventArgs, והוא מספק לאירוע ארגומנטים נוספים. באירוע של לחיצת לחצן אין נתונים רלוונטיים נוספים למשלוח, כך שמשמשים במחלקה EventArgs הפשוטה והריקה. אבל בפרקים מאוחרים יותר (למשל בפרק 9, שעוסק בטפסי web מונעי-נתונים) תראה כמה דוגמאות של מחלקות שמעוררות אירועים עם מידע שימושי יותר.

עם הסבר המושגים הקשורים באירועים סיימנו את הסעיף שדן בהכוונת אובייקטים. מטרת סעיף זה הייתה להכיר את המונחים החשובים ביותר שמשמשים בתכנות מוכוון אובייקטים. בהמשך הלימוד בספר תפגוש דוגמאות מעשיות שיתנו לך מבט נוסף והבנה טובה יותר של מושגים אלה.

עצות מעשיות לתכנות נכון

הרשימה הבאה מציגה כמה עצות מעשיות לתכנות:

- ⊙ קבע למשתנים שלך שמות בעלי משמעות. עבור מונים של לולאות פשוטות אתה יכול להשתמש ב-i אם כי קרוב לוודאי ש-loopCount הוא תיאור טוב בהרבה של מטרת המשתנה. שקול שמות שימושיים עבור שדות פרטיים, כמו _firstName_1 ו-categoryID_1 ולא strFName או catI, ושמות כמו Person_1 ו-FirstName עבור מאפיינים ציבוריים ומחלקות, בהתאמה. החריג היחיד מכללים אלה הוא הפקדים בדף שלך - את אלה מקדימים לעתים קרובות עם הסוג שלהם כמו, למשל, lstOperator לציון רשימה.
- ⊙ נסה ושוב נסה. הדבר חשוב בתכנות אפילו יותר מאשר בעת עבודה עם פקדים ודפי ASPX. הדרך הטובה ביותר ללמוד תכנות היא על ידי ביצוע למעשה. פשוט הקלד קטע קוד והקש Ctrl+F5 כדי לראות איך הוא מתנהג. המהדר יתריע כשמשוהו יהיה שגוי, ויספק לך רמזים שימושיים כיצד לתקן את הטעון תיקון. אל תחשוש לשבש משהו. המשך לנסות עוד וריאציות עד שהקוד יבצע את מה שאתה רוצה שיבצע.
- ⊙ ככל שניתן, נסה לתכנת יחד עם מישהו אחר. אין דבר יעיל יותר מתכנות בזוגות, כשאדם אחד מתכנת ומסביר את מעשיו, בעוד האחר מעיר ושואל שאלות. החליפו מקומות באופן סדיר, כך שתשים עצמך במקום חברך ואז אתה תעיר ותאתגר אותו.

⊙ בכתיבת פונקציות ושגרות, נסה למזער את מספר שורות הקוד. בדרך כלל, שיטות בעלות יותר מ-40 או 50 שורות קוד הן סימן לתכנון גרוע. כשתראה קוד כזה, שקול להעביר חלקים מסוימים לשגרה נפרדת. זה יהפוך את הקוד לקל יותר להבנה, וגם יוביל לקוד טוב יותר ועם פחות באגים. אפילו אם משתמשים בשיטה פעם אחת בלבד, הרי ששמירת נתח מהקוד בשיטה נפרדת יכולה לשפר באופן משמעותי את הבהירות, הפשטות והארגון של הקוד שלך.

⊙ כשתכתוב הערות בקוד, נסה לתאר את המטרה הכללית של הקוד במקום להסביר משפטים ברורים. לדוגמה, ההערה הבאה (שכמוה רואים רבות בקוד ממשי) היא חסרת-ערך לחלוטין:

```
Dim loopCount As Integer = 0 ' Declare variable loopCounter and initialize it to zero
```

כל אדם עם ניסיון בסיסי בקידוד יכול לראות מה עושה קוד זה. במקום זאת, תאר באיזו לולאה משתמשים במשתנה loopCount.

סיכום

פרק זה הכניס אותך לעולם התכנות בסביבת .NET. תכנות הוא נושא ענק שדנים בו אלפי ספרים. למרות שאפשר להגיע בתכנות למורכבות של ממש, הרי שהיסודות החשובים שאותם אתה צריך להבין הם קלים יחסית לתפיסה. הדבר המשעשע לגבי תכנות הוא שאינך צריך להיות מומחה כדי לכתוב תוכניות שימושיות. אתה יכול להתחיל עם דוגמת Hello World פשוטה, ולפלס דרכך משם, כך שכל פעם תרחיב במעט את מבטך על קוד.

כל שפה מגדירה סוגי נתונים כדי שאפשר יהיה לכתוב בה יישומים. סוגים אלה הם אבני הבניין של כל יישום ומשמשים לעבודה עם נתונים שהיישום שלך פועל עליהם. אפשר לאחסן נתונים במשתנים מסוגים נומריים פשוטים, מחרוזות כמו System.String או System.Boolean, או באובייקטים ואוספים אחרים, מורכבים יותר, שמוגדרים בסביבת .NET. או כאלו שאתה מגדיר לעצמך. מערכים ואוספים מאפשרים לקבץ יותר מאובייקט אחד בתוך משתנה יחיד, תוך מתן אפשרות גישה קלה לאובייקטים.

פרט לסוגי נתונים, משפטים הם חלק נוסף בדקדוק של שפת תכנות. בעזרת משפטים אתה אומר לתוכנית מה ברצונך שתבצע. קבוצת המשפטים הראשונה הם האופרטורים, שמאפשרים לפעול עם נתונים. לדוגמה, ביכולתך לחבר שני ערכים נומריים, לאחד שתי מחרוזות, או להשוות את התוצאה של ביטוי עם ערך מוגדר מראש.

קבלת החלטות בקוד היא פעולה נפוצה, כך ששתי השפות, VB.NET ו-C#, מציעות מספר דרכים שונות לעשות זאת באמצעות משפטי If Else ו-If Else, והבולקים Select Case או switch. כדי לחזור על פעולת בלוק קוד, או לעבור שוב ושוב על אוסף, עומד לרשותך מגוון מבני לולאות, כולל הלולאה For, הלולאות For Each / foreach ונוסחים שונים של הלולאות While ו-Do.

כדי לכתוב קוד קל לקריאה ולתחזוקה, עליך לנסות לכתוב בצורה מובנית ומובנת ככל האפשר. שימוש בפונקציות ושגרות מאפשר להעביר קוד חוזר לקבצי קוד נפרדים. מרחבי שמות מאפשרים לקבץ את הקוד קיבוץ לוגי, דבר שיקל על מציאת קטעי קוד ועל הניווט ביניהם. בזמן שאתה מקודד אל תשכח לכתוב הערות כלולות והערות XML לתיאור הקוד. אם תאמץ הרגל זה בשלב מוקדם, תודה לעצמך כעבור זמן כשתצטרך לתחזק קטע קוד ישן.

הסעיף האחרון בפרק זה עסק בתכנות מוכוון אובייקטים. בעוד OO לכשעצמו הוא נושא רחב מאוד, קל ללמוד את יסודותיו. בפרק זה למדת אודות האיברים הבסיסיים של תכנות OO: מחלקות, שיטות, מאפיינים ובנאים. למדת מעט גם על נושא ההורשה ועל הירושה של תכונות והתנהגות בין אובייקטים, הכוח המניע שמאחורי תכנון OO. בעזרת הורשה אתה יכול להגדיר אובייקטים שיוורשים מידע רב ממחלקות האב שלהם. יכולת זו מאפשרת ליצור מחלקות מיוחדות שחושפות התנהגות נוספת, מבלי שתצטרך לקודד מחדש את הפונקציונליות שהמחלקות קיבלו ממחלקת האב. הדבר מוביל לקוד ברור יותר, ולאובייקטים קלים לשימוש על ידי כלל המתכנתים.

בפרק הבא, שעוסק ביצירת דפי web בעלי מראה עקבי, תפגוש שוב בהורשה כשתיצור את המחלקה BasePage שמשרתת כמחלקת אב עבור כל המחלקות של קוד ברקע בפרויקט Planet Wrox.

תרגילים

1. בהתחשב בעובדה שהאדם הזקן ביותר בעולם חי עד גיל 122, מהו סוג הנתונים הנומרי הטוב ביותר לאחסון הגיל של אדם? תקבל נקודות בונים אם תציע חלופה טובה עוד יותר לאחסון הגיל.
2. מה עושה הקוד הבא?

VB.NET

```
DeleteButton.Visible = ShoppingCart.Items.Count > 0
```

C#

```
DeleteButton.Visible = ShoppingCart.Items.Count > 0;
```

3. לפניך המחלקה Person הבאה, איך ייראה הקוד עבור מחלקה חדשה בשם PersonWithPhoneNumber? השתמש במנגנון ההורשה כדי ליצור מחלקה חדשה זו.

VB.NET

```
Public Class Person
    Private _name As String

    Public Property Name() As String
        Get
            Return _name
        End Get
    End Get
End Class
```

```
        Set(ByVal value As String)
            _name = value
        End Set
    End Property
End Class
```

C#

```
public class Person
{
    public string Name { get; set; }
}
```

פרק 6

יצירת אתרי Web בעלי מראה עקבי

כשמתכננים ובונים אתר אינטרנט יש לשאוף תמיד שהעיצוב, העריכה וההתנהגות בו יהיו עקביים ככל האפשר. עקביות מקנה לאתר שלך מראה מקצועי, ומסייעת למבקרים בו למצוא את דרכם בקלות. למרבה המזל, ASP.NET 3.5 מציעה כמה תכונות מצוינות ליישום עיצוב עקבי. בנוסף, סביבת העבודה כוללת קבוצה של כלים מעולים ותצוגת עיצוב (Design View) עשירה שמסייעים ביצירת דפים שנראים נהדר בזמן קצר ביותר.

בפרקים קודמים למדת איך לעבוד עם HTML, CSS, VWD ופקדי שרת כדי ליצור דפי אינטרנט באופן חזותי. בפרק 5 ערכת היכרות עם תכונות בסביבת .NET. פרק זה הוא הראשון שמשלב את הרעיונות הללו. תראה כיצד לכתוב קוד שחל על כל הדפים באתר, תוך יצירת מקום מרכזי עבור התנהגות משותפת.

בפרק זה תלמד בהרחבה על הנושאים הבאים שמסייעים ביצירת דפי אינטרנט מעוצבים היטב, עקביים וקלים לתחזוקה:

- ⊙ שימוש בדפים ראשיים ודפי תוכן שמאפשרים להגדיר את המראה הכללי של דף אינטרנט.
- ⊙ עבודה עם דף בסיס מרכזי שמאפשר להגדיר התנהגות משותפת לכל הדפים באתר.
- ⊙ יצירת ערכות נושא (themes) של ASP.NET 3.5, להגדרת המראה והתחושה של האתר, תוך מתן אפשרות למשתמש לבחור בזמן הביצוע את ערכת הנושא המועדפת עליו.
- ⊙ יצירת skins כדי לבצע שינויים מהירים ברחבי האתר לשליטה בעיצוב ובעריכה.

הסעיף הבא מראה כיצד ליצור דף ראשי שמגדיר את המראה והתחושה הכללית של דף. דפי ASPX באתר יכולים אז להשתמש בדף ראשי זה מבלי שיהיה צורך לכתוב מחדש את קוד העיצוב והעריכה. שאר הסעיפים בפרק זה בנויים על פני הדף הראשי.

עקביות עריכה ועיצוב של דף בעזרת דפים ראשיים

ברוב האתרים, רק חלק מהדף משתנה במעבר מדף לדף. החלקים שאינם משתנים כוללים בדרך כלל אזורים משותפים כמו כותרת עליונה, תפריט וכותרת תחתונה. כדי ליצור דפי אינטרנט בעלי עריכה ועיצוב עקביים עליך לדעת להגדיר את האזורים הקבועים, שאינם משתנים כולם או חלקם, בקובץ תבנית (template) יחיד. פתרון התבנית לא היה קיים בגרסאות ASP.NET

שקדמו ל- ASP.NET 2.0, ועל כן, מפתח התוכנה היה נאלץ לשכפל את עיצוב ואת עריכת הדף עבור כל דף באתר, או לפנות לטריקים פרושים בתכנות. מצב זה השתנה תודות לדפים ראשיים (master pages). התועלת הגדולה ביותר של דפים ראשיים היא בכך שהם מאפשרים להגדיר במקום אחד את המראה והתחושה של כל הדפים באתר. פירוש הדבר, שאם רוצים לשנות את העריכה והעיצוב של האתר - כשרוצים למשל להעביר את התפריט מצד שמאל לצד ימין - יש צורך לשנות את הדף הראשי בלבד, והדפים שמבוססים על דף ראשי זה יקבלו את השינויים באופן אוטומטי.

לאחר שדפים ראשיים הוכנסו לשימוש בגרסה ASP.NET 2.0, הם אומצו במהירות על ידי קהילת המפתחים כפתרון מושלם לנושא התבניות בדפי ASP.NET. הסיבה לכך היא קלות השימוש בהם. יתר על כן, ב-VWD יש תמיכה מצוינת בזמן עיצוב, שמאפשרת ליצור ולהתבונן בדפים במהלך הפיתוח בזמן עיצובם, ולא רק להתבונן בתוצאת הדפדפן בזמן ביצוע.

דף ראשי דומה במידה מסוימת לדף ASPX רגיל. הוא מכיל HTML סטטי כמו התגיות `<html>`, `<head>` ו-`<body>`, והוא יכול להכיל גם HTML אחר ופקדי שרת של ASP.NET. בתוך הדף הראשי אתה מגדיר את סימני העריכה שברצונך שיחזרו בכל דף, כמו העיצוב והעריכה הכלליים של הדף והתפריט.

עם זאת, דף ראשי אינו דף ASPX אמיתי, ולא ניתן לבקש אותו ישירות בדפדפן; הוא משרת רק כתבנית, שדפי אינטרנט ממשיים - שנקראים **דפי תוכן** (content pages) - מבוססים עליה.

במקום ההנחיה @ Page שראינו בפרקים קודמים, הדף הראשי משתמש בהנחיה @ Master שמזהה את הקובץ כדף ראשי:

VB.NET

```
<%@ Master Language="VB" %>
```

C#

```
<%@ Master Language="C#" %>
```

כמו לדף ASPX רגיל, כך גם לדף ראשי יכול להיות קובץ קוד ברקע, שמזהה על ידי המאפיינים CodeFile ו-Inherits:

VB.NET

```
<%@ Master Language="VB" CodeFile="MasterPage.master.vb"
    Inherits="Masterpages_MasterPage" %>
```

C#

```
<%@ Master Language="C#" CodeFile="MasterPage.master.cs"
    Inherits="Masterpages_MasterPage" %>
```

כדי ליצור אזורי שדפי תוכן יוכלו למלא, צריך להגדיר בדף פקדים מסוג ContentPlaceholder, כך:

```
<asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
</asp:ContentPlaceHolder>
```


אתה יכול ליצור כמה שומרי מקום (placeholders) שרק תרצה, אם כי מספרם מוגבל בדרך כלל לארבעה-חמישה כדי שהדף יישאר בר-ניהול.

קבצי תוכן הם ביסודם קבצי ASPX רגילים ללא הקוד הרגיל שמוצאים בהם, כמו התגיות `<html>`, `<head>`, `<body>` ו-`<form>`. הם מחוברים לדף ראשי תוך שימוש במאפיין `MasterPageFile` של ההנחיה `Page`.

VB.NET

```
<%@ Page Language="VB"
    MasterPageFile="~/Masterpages/MasterPage.master"
    AutoEventWireup="false" CodeFile="Default.aspx.vb"
    Inherits="_Default">
```

C#

```
<%@ Page Language="C#"
    MasterPageFile="~/Masterpages/MasterPage.master"
    AutoEventWireup="true" CodeFile="Default.aspx.cs"
    Inherits="_Default">
```

התוכן שכלול בדף מוכנס לפקד `<asp:Content>` שמצביע על ה-`ContentPlaceHolder` המתאים לו בדף הראשי:

```
<asp:Content ID="Content1"
    ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
</asp:Content>
```

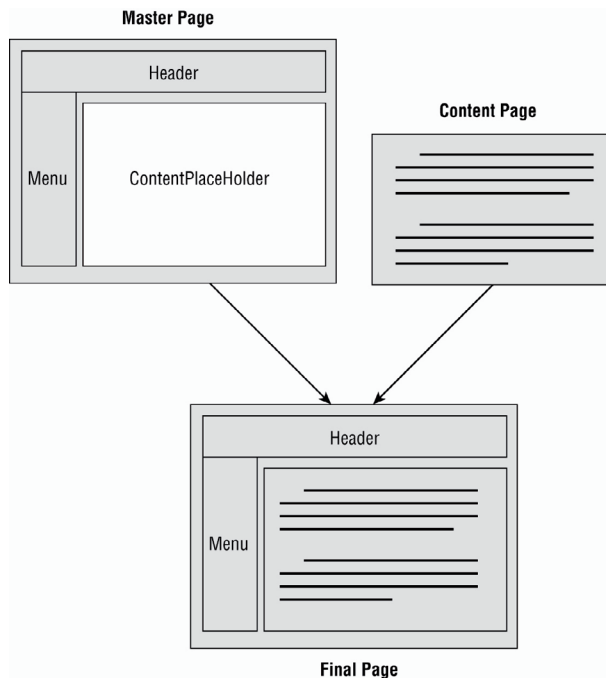
שים לב שהמאפיין `ContentPlaceHolderID` של הפקד `Content` מצביע על `ContentPlaceHolder` שמוגדר בדף הראשי. כרגע הוא מצביע על שם ברירת המחדל `ContentPlaceHolder1`, אבל בתרגיל מאוחר יותר תראה כיצד לשנות זאת.

כשהדף מבוקש בזמן ביצוע, משולבים סימני העריכה של הדף הראשי ושל דף התוכן, מעובדים לדף אחד ונשלחים לדפדפן. תרשים 1-6 מראה דף ראשי ודף תוכן שתוצאתם הוא הדף הסופי שנשלח לדפדפן.

כדי שתראה תהליך זה בפעולה, ינחו אותך הסעיפים הבאים ביצירת דפים ראשיים ודפי תוכן.

יצירת דפים ראשיים

תיבת הדו-שיח `Add New Item` משמשת להוספת דפים ראשיים לאתר. הדפים יכולים להיות מוצגים בכל מקום באתר, גם בתיקיית השורש, אבל מנקודת מבט ארגונית קל יותר לאחסן אותם בתיקייה נפרדת. כמו דפי ASPX רגילים, הם תומכים במודל קוד כלול וגם במודל קוד ברקע. הדפים הראשיים שבהם משתמשים בפרויקט הדוגמה `Planet Wrox` משתמשים אך ורק במודל קוד ברקע. בתרגיל הבא תיצור דף ראשי פשוט ותוסיף לו מעט קוד HTML כדי להגדיר את המבנה הכללי של הדפים באתר.



תרשים 6-1

נסה זאת יצירת דף ראשי

1. פתח את פרויקט Planet Wrox (אם טרם פתחת אותו).
2. בפרק 2 יצרת תיקייה בשם MasterPages כדי שתכיל את הדפים הראשיים שלך, והוספת לה דף ראשי אחד. אם לא ביצעת תרגיל זה, הוסף עכשיו את הדף הראשי. כדי לעשות זאת, צור את התיקייה MasterPages, לחץ על התיקייה החדשה לחיצה ימנית, בחר Add New Item ובחר Master Page. ודא שהדף הראשי משתמש בקוד ברקע, ושנבחרה עבודה שפת התכנות המועדפת עליך.
3. הוסף את הקוד המוצג להבא בין תגיות <form> של הדף הראשי. צעד זה יחליף את תגיות <div> ואת ContentPlaceholder אשר VWD הוסיפה עבורך בזמן שיצרת את הדף הראשי. שים לב שהקוד כמעט זהה לקוד שהוספת ל-Default.aspx בפרק 3, פרט לאלמנטים <asp:ContentPlaceholder> ו- <a> שבתוך ה- <div> עם מזהה ה-Header. האלמנט <a> מפנה את המשתמשים חזרה לדף הבית, והסגנון שלו ייקבע מאוחר יותר.

```

<form id="form1" runat="server">
  <div id="PageWrapper">
    <div id="Header">
      <a class="HeaderLink" href="~/ " runat="server"></a>
    </div>
    <div id="MenuWrapper">Menu Goes Here</div>
    <div id="MainContent">
      <asp:ContentPlaceholder ID="cpMainContent" runat="server">
  
```

```

        </asp:ContentPlaceholder>
    </div>
    <div id="Sidebar">Sidebar Goes Here</div>
    <div id="Footer">Footer Goes Here</div>
</div>
</form>

```

ודא ש-ContentPlaceholder נמצא בין תגיות <div> MainContent. תוכל לגרור אותו מארגז הכלים אל הדף, או להכניס את הקוד ישירות, כשאתה נעזר בכלי ה-IntelliSense בכל מקרה המזהה (ID) של הפקד יהיה cpMainContent.

4. עבור למצב Design View בדף הראשי, וגרור את הקובץ Styles.css מהתיקייה Styles בסיר הפתרון אל הדף הראשי. ברגע שתשחרר את הקובץ, VWD תעדכן את תצוגת העיצוב (Design View) ותציג את העיצוב והעריכה עבור האתר שיצרת בפרק 3. אם העיצוב אינו משתנה, עבור ל- Markup View כדי לוודא שהתגית <link> נמצאת בראש הדף, וגם מצביעה על קובץ CSS שלך. אם התגית אינה נמצאת שם, הוסף אותה ידנית:

```

<asp:ContentPlaceholder ID="head" runat="server">
</asp:ContentPlaceholder>
<link href=" ../Styles/Styles.css" rel="stylesheet" type="text/css" />
</head>

```

הדף שמוצג ב- Design View אמור להיראות עכשיו דומה לתרשים 6-2. שים לב לשטח התחום על ידי גבול בצבע ארגמן, בין התפריט לבין אזור הכותרת התחתונה. זהו הפקד ContentPlaceholder שבשימוש דפי התוכן. לאחר ההסבר על דף ראשי, תראה כיצד זה פועל.



תרשים 6-2

5. שמור וסגור את הדף, מכיוון שסיימנו לעסוק בדף ראשי בשלב זה.

כיצד זה פועל

דפים ראשיים מתנהגים בתוך Visual Web Developer כדפים רגילים. אפשר להוסיף להם קוד HTML ופקדי שרת, וניתן לנהל דף ראשי הן ב- Markup View והן ב- Design View. ההבדל הגדול הוא, כמובן, בכך שדף ראשי אינו דף אמיתי; הוא רק תבנית (template) עבור דפים אחרים באתר. בסעיפים הבאים נבהיר כיצד זה פועל.

יצירת דפי תוכן

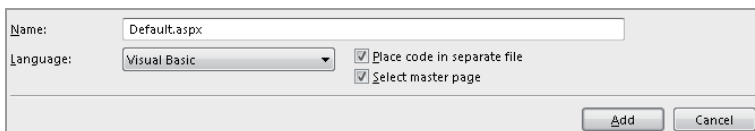
דף ראשי הוא חסר ערך ללא דף תוכן שמשמש בו. בדרך כלל יש דפים ראשיים בודדים ודפי תוכן רבים. כדי לבסס דף תוכן על דף ראשי עקוב אחר הצעדים הבאים: לחץ לחיצה ימנית על האתר בסייר הפתרון, ובחר Add New Item. בצד הימני התחתון של תיבת הדו-שיח שתיפתח, סמן את האפשרות Select Master Page. לחילופין, ביכולתך לקבוע את דף הבסיס ב- Markup View של דף התוכן על ידי הגדרת המאפיין MasterPageFile באופן ישיר בשורת ההנחיה @ Page, אותה ראית קודם לכן, כשלמדנו על דפים ראשיים.

דפי תוכן יכולים להכיל רק את הפקדים מסוג `<asp:Content>`, שכל אחד מהם ממופה לפקד `<asp:ContentPlaceHolder>` בדף הראשי. פקדי תוכן אלה, יכולים להכיל סימני עריכה תקינים כמו HTML והכרזות פקדי שרת. מכיוון שכל סימני העריכה בדף תוכן צריכים להיות "עטופים" על ידי תגיות `<asp:Content>`, לא קל להפוך דף ASPX קיים לדף תוכן. בדרך כלל, הדבר שקל ביותר לעשותו הוא להעתיק את התוכן שרוצים לשמור אל לוח העריכה (clipboard), למחוק את הדף הישן, ולהוסיף לאתר דף חדש שמבוסס על הדף הראשי. ברגע שהדף התווסף, אפשר להדביק לו את סימני העריכה בין תגיות `<asp:Content>` שלו. בתרגיל הבא נראה כיצד זה פועל.

נסה זאת הוספת דף תוכן

בתרגיל זה תראה איך להוסיף לאתר דף תוכן שמבוסס על הדף הראשי שיצרת קודם. כאשר הדף מתווסף, אפשר להוסיף תוכן לאזורי `<asp:Content>` שהוגדרו בו מראש.

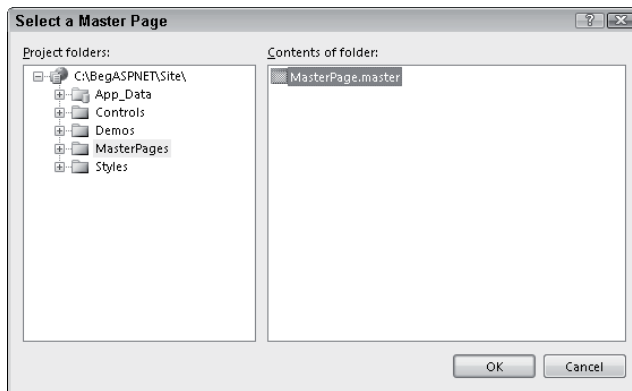
- בתרגילים קודמים הוספת דפי ASPX רגילים לפרויקט. כעת ברצוננו "לשדרג" את הדפים לשימוש בדף הראשי החדש. מכיוון של-VWD אין תמיכה מובנית בשינוי דף רגיל לדף תוכן, צריך להעתיק את התוכן באופן ידני מדף ASPX הישן אל הדף החדש. אם ברצונך לשמור את טקסט הפתיחה שהוספת קודם ל-Default.aspx, עליך להעתיק את כל קוד HTML שנמצא בין התגיות `<div> MainContent` אל לוח העריכה (כלומר, את האלמנט `<h1>` ושני האלמנטים `<p>` שיצרת קודם), ואז למחוק את הדף Default.aspx מסייר הפתרון. עכשיו, לחץ לחיצה ימנית על האתר ובחר Add New Item. בחר Web Form, קרא לדף בשם Default.aspx, בחר בשפת התכנות הנכונה, ובתחתית תיבת הדו-שיח בחר בתיבות הסימון עבור Place Code In Separate File ועבור Select Master Page, כפי שמוצג בתרשים 6-3.



תרשים 6-3

לבסוף, לחץ על הלחצן Add.

2. בתיבת הדו-שיח Select a Master Page (ראה תרשים 4-6), לחץ על התיקייה MasterPages שבחלונית השמאלית, ואז לחץ על MasterPage.master שבחלונית הימנית.



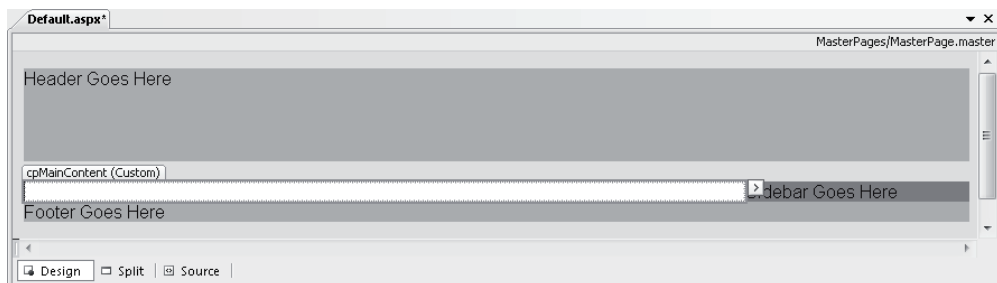
תרשים 4-6

לחץ OK לאישור ההוספה של הדף לאתר. במקום לקבל דף מלא ב-HTML, כפי שקיבלת במקרה של דפי ASPX רגילים, אתה מקבל כעת רק שני שומרי מקום עם תגיות `<asp:Content>`.

```
<%@ Page Language="VB" MasterPageFile="~/MasterPages/MasterPage.master"
    AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="_Default"
    title="Untitled Page" %>
```

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" Runat="Server">
</asp:Content>
```

3. עבור לתצוגה Design View ושים לב שפרט לאזור `<asp:Content>` שעבור `cpMainContent`, הכל מסומן באפור ומיועד לקריאה בלבד. תרשים 5-6 מראה כיצד הדף צריך להיראות כעת.



תרשים 5-6

4. אם בלוח העריכה נמצאים עדיין סימני עריכה ישנים מ-Default.aspx, לחץ פעם אחת בתוך שומר המקום `cpMainContent` והקש `Ctrl+V`. פעולה זו מוסיפה לדף את סימני העריכה, בדיוק בין תגיות `<asp:Content>`, כנדרש.

5. שמור את כל השינויים שביצעת על ידי הקשת Ctrl+Shift+S (או, מהתפריט הראשי, בחר File ← Save All), ופתח את הדף בדפדפן על ידי הקשת Ctrl+F5. הדפדפן אמור להציג את הדף בדומה למה שראית ב- Design View (ראה תרשים 5-6).
6. כעת הבט ב-HTML של הדף בדפדפן. עשה זאת על ידי לחיצה ימנית על הדף ובחירת View Source. שים לב שהמקור של הדף הסופי בדפדפן הוא שילוב כלשהו הכולל את המקור של הדף הראשי ואת דף התוכן.

```
<div id="PageWrapper">
  <div id="Header"><a href="/" class="HeaderLink"></a></div>
  <div id="MenuWrapper">Menu Goes Here</div>
  <div id="MainContent">
    <h1>Hi there visitor and welcome to Planet Wrox</h1>
    <p class="Introduction">
      We're glad you are paying a visit to <a href="http://www.PlanetWrox.com">
        www.PlanetWrox.com</a>, the coolest music community site on the Internet.
    </p>
    ...
  </div>
</div>
```

המקור של ארבע השורות הראשונות מגיע מהדף הראשי, בשעה שהשורות המוצללות של HTML מגיעות מדף התוכן.

7. חזור ל-VWD כדי ליצור דף חדש בשם Login.aspx. ודא שוב שהוא מבוסס על הדף הראשי ומשתמש בקוד ברקע. בשלב זה אין צורך להוסיף לדף פקדים כלשהם, אבל הוא ישמש כבסיס לפונקציונליות של ההתחברות (login) שתיוצר בפרק 15.
8. חזור ל-Default.aspx ועבור למצב Design View. מתחת לטקסט הפתיחה עם הכותרת ושני האלמנטים <p>, צור פסקה חדשה והקלד מעט טקסט (לדוגמה, (You can log in here. הדגש את הטקסט שהזנת, ומהתפריט הראשי בחר Format ← Convert to Hyperlink. בתיבת הדו-שיח שנפתחת לחץ על הלחצן Browse ובחר בדף Login.aspx שזה עתה יצרת. לחץ OK פעמיים לאישור.
9. שמור את כל השינויים והקש שוב Ctrl+F5 כדי לראות את הדף Default.aspx בדפדפן. שים לב שנשמרו בו העיצוב והעריכה הכלליים, כמו הכותרת וסרגל הצד. הדבר היחיד שמשתנה במעבר מדף לדף הוא התוכן באזור התוכן המרכזי.

כיצד זה פועל

כאשר דף תוכן (שמבוסס על דף ראשי) נדרש על ידי הדפדפן, השרת קורא הן את דף התוכן והן את הדף הראשי, ממזג את שניהם, מעבד אותם, ושולח את התוצאה הסופית לדפדפן. בצעד 6 בתרגיל הקודם ראית שקוד HTML של הדף המבוקש על ידי הדפדפן מכיל את סימני העריכה משני הקבצים.

כשהדברים נוגעים לעדכון או לשינוי קיצוני במראה האתר, הרי שהשימוש בדפים ראשיים חוסך עבודה רבה. מכיוון שכל העיצוב והעריכה של האתר מוגדרים בדף הראשי, תוכל לגעת בקובץ יחיד זה בלבד כשתרצה לערוך שינוי כלשהו. השינוי שתעשה יתבטא בכל דפי התוכן באופן אוטומטי.

מבט מקרוב בדפים ראשיים

עד כה ראית דף ראשי שיש בו פקד ContentPlaceholder השומר מקום עבור התוכן העיקרי. אבל אם תתבונן בדף הראשי ב- Markup View תמצא ContentPlaceholder נוסף בקטע של ראש הדף:

```
<head runat="server">
  <title>Untitled Page</title>
  <asp:ContentPlaceholder id="head" runat="server">
  </asp:ContentPlaceholder>
  ...
</head>
```

שומר מקום זה מתווסף באופן אוטומטי בכל פעם שאתה מוסיף לאתר דף ראשי חדש. אפשר להשתמש בו בדפי תוכן כדי להוסיף לדף שלך תוכן ייחודי שמקומו בין תגיות <head> של הדף, כמו CSS (הן גיליונות סגנון חיצוניים והן מוטמעים) ו-JavaScript. בהמשך הספר תלמד עוד אודות JavaScript. תצטרך להוסיף תוכן לשומר מקום זה ב- Markup View מפני שהוא אינו נראה ב- Design View.

שומר המקום ContentPlaceholder שנקרא בדף הראשי בשם cpMainContent, אינו מכיל כרגע סימני עריכה כלשהם. אין זה חייב להיות כך. אתה יכול להוסיף בו תוכן משלך, שישרת כברירת המחדל עבור דפי התוכן שלך כל עוד לא יידרס על ידי דף התוכן. לדוגמה, בדף הראשי אתה יכול לשנות את הפקד <asp:ContentPlaceholder>, כך שייראה כמו בדוגמה זו:

```
<asp:ContentPlaceholder ID="cpMainContent" runat="server">
  This is default text that shows up in content pages that don't
  explicitly override it.
</asp:ContentPlaceholder>
```

כשתבסס דף חדש על דף ראשי זה, לא תראה בתחילה את ברירת המחדל הזו ב- Markup View. עם זאת, באפשרותך לפתוח את לוח Smart Tasks של הפקד Content, שמוצג בתרשים 6-6, ולבחור Default to Master's Content.



תרשים 6-6

כשתישאל אם ברצונך שברירת המחדל תהיה הדף הראשי, ותלחץ Yes, תסיר VWD את הבלוק <asp:Content> כולו מ- Markup View של הדף. אבל, כשתבקש את הדף בדפדפן, עדיין תראה את

תוכן ברירת המחדל מהדף הראשי. התוכן אמנם נראה ב- Design View, אבל הוא מוצג במשטח העיצוב לקריאה-בלבד. דף ראשי עם תוכן ברירת המחדל יכול להיות שימושי כשתרצה בשלב מאוחר יותר להוסיף ContentPlaceHolder חדש. דפים קיימים יכולים להציג את תוכן ברירת המחדל מבלי שתצטרך לגעת בהם. דפים חדשים יכולים להגדיר בעצמם את התוכן שלהם.

ברגע שתוכן הדף הראשי הופך לברירת המחדל, תוכל ליצור שוב תוכן מותאם אישית על ידי פתיחת לוח Smart Tasks ובחירת Create Custom Content. הפעולה תגרום להוספת פקד `<asp:Content>` לדף, אשר תוכל למלא אותו בתוכן משלך.

קיון של דפים ראשיים

אפשר לקנן (nest) דפים ראשיים. דף ראשי מקונן הוא דף ראשי שמבוסס על דף ראשי אחר. אפשר לבסס דפי תוכן על דף ראשי מקונן. הדבר שימושי כשהאתר מכון לתחומים שונים ועם זאת רוצים לשוות לו מראה ותחושה משותפים. ניקח לדוגמה אתר של תאגיד, שמופרד לפי מחלקות. הדף הראשי החיצוני מגדיר את המראה והתחושה הכלליים של האתר, כולל הלוגו של התאגיד ואלמנטים אחרים העוסקים במיתוג. מחלקות שונות בתאגיד יכולות עכשיו להיות בעלות דפים ראשיים מקוננים שונים משל עצמן המתבססים על הדף הראשי של התאגיד ועם זאת יוצרים לעצמם תבנית ייחודית. לדוגמה, דפי התוכן של מחלקת המכירות יכולים להתבסס על דף ראשי שלהם השונה מזה של מחלקת השיווק. כך יכולה כל מחלקה להוסיף את חותמה האישי לחלק שלה באתר. התוכנה Visual Web Developer 2008 היא בעלת תמיכה טובה בדפים ראשיים מקוננים שיוצרים בזמן עיצוב ב- Design View. הדבר מאפשר מבט טוב על המראה הסופי של הדף.

יצירת דף ראשי מקונן קלה למדי: כשאתה מוסיף דף ראשי, סמן את תיבת הסימון Select Master Page, בדיוק כפי שאתה עושה כשאתה מוסיף דף תוכן רגיל לאתר. אחר כך הוסף פקדי `<asp:ContentPlaceHolder>` לפקדי `<asp:Content>` במקומות שברצונך לדרוס בדפי התוכן שלך.

אזהרות לגבי דפים ראשיים

למרות שדפים ראשיים הם נפלאים ויכולים לחסוך עבודה רבה, ראוי לשים לב לכמה אזהרות הקשורות בהם.

ראשית, ASP.NET runtime משנה את זיהוי הלקוח (client ID) של הפקדים בדף. זהו המאפיין id (זיהוי) שמשמש בתסריטי צד לקוח כדי לגשת לפקדים מתוך JavaScript שבדפדפן. בדרך כלל, עם דפי ASPX רגילים מוכנס ID צד-השרת של פקד, כפי שהוא, אל HTML הסופי. לדוגמה, בדף ASPX רגיל, פקד Button שמוגדר עם הקוד הבא:

```
<asp:Button ID="Button1" runat="server" Text="Button" />
```

נראה לבסוף כך ב-HTML:

```
<input type="submit" name="Button1" value="Button" id="Button1" />
```


ID צד-השרת שמוגדר באלמנט Button, מועתק ישירות למאפיין id של האלמנט <input> בדפדפן. עם זאת, אותו לחצן ייראה כך בתוך פקד <asp:Content>:

```
<input type="submit" name="ctl00$cpMainContent$Button1"
value="Button" id="ctl00_cpMainContent_Button1" />
```

המאפיינים name ו-id מקבלים כעת כקידומת את ה-ID שנוצר אוטומטית על ידי הדף הראשי (ct100), ואת ה-ID של הפקד ContentPlaceHolder (cpMainContent).

פירוש הדבר, שכל קוד בצד-לקוח שמתייחס ל-Button1 צריך כעת להתייחס אל ct100_cpMainContent_Button1.

שים לב שזו אינה בעיה של דף ראשי בלבד; תיתקל באותה התנהגות גם במצבים אחרים. לדוגמה, כשעובדים עם פקדי משתמש (יידונו בפרק 9) ופקדים מקושרי-נתונים (יידונו מפרק 12 והלאה).

האזהרה השנייה קשורה לאזהרה הראשונה. מכיוון שה-name וה-id של האלמנטים של HTML משתנים, הם מוסיפים באופן משמעותי לגודל הדף. במקרה של פקד יחיד זו איננה אולי בעיה, אבל כשיש דפים עם פקדים רבים, הם עלולים להשפיע על ביצועי האתר. הבעיה מחמירה עם דפים ראשיים מקוננים, כששני פקדי התוכן מצורפים ל-ID. בתוך דף ראשי מקונן יכול אותו לחצן להיראות כך:

```
<input type="submit"
name="ctl00$ctl00$ContentPlaceHolder1$ContentPlaceHolder1$Button1"
value="Button" id="ctl00_ctl00_ContentPlaceHolder1_ContentPlaceHolder1_Button1"
/>
```

כדי להקל על הבעיה, עליך לשמור את קודי הזיהוי (ID) של הפקד ContentPlaceHolder ושל Content קצרים ככל האפשר. הספר משתמש בשמות ארוכים, כמו cpMainContent, כדי שהקוד יהיה קריא יותר, אבל באתרים שלך אתה יכול לצמצם זאת ל-cpMC או cp1 במטרה לשפר את נפח התעבורה ברשת בכל בקשה.

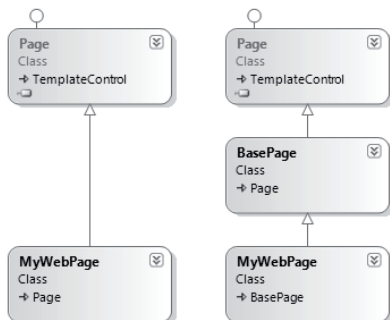
דפים ראשיים מאפשרים להגדיר את המראה והתחושה הכלליים של האתר במקום אחד. הדבר משפר את העקביות ואת יכולת התחזוקה של האתר. עם זאת, קיימת דרך נוספת לשיפור העקביות: ריכוז התנהגות הדפים באתר. תוכל לעשות זאת עם דף בסיס מרכזי, שבו נדון בסעיף הבא.

שימוש בדף בסיס מרכזי

בפרק 5 נאמר שכברירת מחדל, כל דפי ASPX יורשים ממחלקה בשם System.Web.UI.Page. פירוש הדבר שלכל הדפים יש, לפחות, את ההתנהגות המוגדרת במחלקה זו.

אבל יש נסיבות שבהן התנהגות זו אינה מספקת עבור הדפים שברצונך ליצור, ותרצה להוסיף התנהגות נוספת שתחול על כל הדפים באתר. במקום להוסיף התנהגות זו לכל דף בנפרד, אפשר ליצור דף בסיס (base page) משותף אשר יורש מהמחלקה System.Web.UI.Page ואז כל הדפים באתר יכולים לרשת מדף בסיס זה, במקום המחלקה Page הרגילה (בירושה מדף הבסיס אתה

מקבל גם התנהגות של המחלקה Page וגם התנהגות המוגדרת בדף הבסיס). בחלק השמאלי של תרשים 6-7 תוכל לראות כיצד דף ASPX רגיל בשם MyWebPage יורש ישירות מהמחלקה Page. החלק הימני מציג מצב שבו דף ASPX יורש ממחלקה בשם BasePage, שבתורה יורשת מ-Page.



תרשים 6-7

כדי שהדפים שלך יוכלו לרשת מדף בסיס זה, עליך לעשות שני דברים:

- ⊙ בתיקייה App_Code, צור מחלקה (דף הבסיס) שיורשת מ-System.Web.UI.Page.
- ⊙ שנה את הדפים באתר שלך כך שיירשו מדף בסיס זה, במקום שיירשו מהמחלקה Page הרגילה.

בתרגיל הבא תיצור מחלקת דף בסיס חדשה בתיקייה App_Code. בשלב זה, המטרה היחידה של מחלקה זו היא לבדוק את הכותרת (Title) של הדף בזמן ביצוע כדי למנוע מדפים עם שם ברירת המחדל - Untitled Page, שהוקצה באופן אוטומטי על ידי VWD - מלהגיע לדפדפן. פונקציונליות זו קלה להשגה, יחסית, וכך תוכל להתמקד ברעיון ההורשה ולא לעסוק בקוד הממשי. בהמשך הפרק, בסעיף שידון בערכות נושא (themes), תשנה את דף הבסיס פעם נוספת, כדי לקבוע את ערכת הנושא המועדפת על המשתמש.

לפני שתוכל ליישם את מחלקת דף הבסיס, עליך לדעת יותר על מחזור החיים של דף ב-ASP.NET, נושא חשוב זה מתאר את התהליך שעובר דף אינטרנט כשהוא מבוקש על ידי הדפדפן.

מבוא למחזור החיים של דף ASP.NET

כשחושבים כיצד דף כלשהו מוגש על ידי שרת אינטרנט לדפדפן, וחושבים על תהליך זה כמחזור חיים של דף, אפשר לגלות כמה רגעים חשובים בחייו של דף. לדוגמה, הבקשה הראשונית על ידי הדפדפן היא נקודת התחלה של "חיי" הדף. באופן דומה, לאחר שהדף שולח את כל קוד HTML שלו לדפדפן, נראה שחייו תמו. אבל יש עוד אירועים מעניינים במחזור החיים של דף. הטבלה שלהלן מתארת שמונה שלבים מלאים שהדף עובר. בכל שלב מתעורר לפחות אירוע אחד שמאפשר למפתח הדף להתחבר אל תהליך מחזור החיים שלו, ולבצע פעולות ברגע הנכון. בתרגיל הבא תראה דוגמה לכך. הבה נתבונן תחילה בשלבי מחזור החיים של הדף.

שלב במחזור החיים	תיאור
Page Request בקשת הדף	בקשה ל־dף ASPX מתחילה את מחזור החיים של אותו דף. במקרה שבו השרת מסוגל ומורשה להחזיר עותק מוטמן של הדף, כל מחזור החיים אינו מבוצע. עם בקשת הדף מחזור החיים עובר לשלב ההתחלה.
Start התחלת חיי הדף	בשלב ההתחלה של חיי הדף, הוא מקבל גישה למאפיינים כמו Request ו-Response, שמשמשים לאינטראקציה עם סביבתו. בנוסף, בשלב זה מתעורר האירוע <i>PreInit</i> כדי לאותת שהדף עומד להיכנס לשלב האתחול. מאוחר יותר תשתמש באירוע זה כדי להגדיר את ערכת הנושא של הדף.
Page Initialization אתחול הדף	במהלך שלב זה הפקדים השונים נעשים זמינים, הן אלה שהוגדרו בדף והן אלה שהתווספו לו על ידי תכנות. במהלך השלב, המחלקה Page מעוררת שלושה אירועים: <i>Init</i> , <i>InitComplete</i> ו- <i>PreLoad</i> . בשלב זה גם נטענים מאפייני הפקד שוב מ-ViewState במהלך פעולת ה- <i>postback</i> . כך, כאשר שינית פריט ב- <i>DropDownList</i> וגרמת ל- <i>postback</i> , זהו הרגע שבו הפריט הנכון נבחר שוב בפקד.
Load טעינת הדף	במהלך שלב זה הדף מעורר את האירוע Load.
Validation בדיקת אימות	במהלך שלב זה מעובדים פקדי האימות (Validation controls), שמשמשים לבדיקת האימות של קלט המשתמש. תלמד עליהם יותר בפרק 9.
Postback event handling טיפול בפעולת ה- <i>postback</i>	במהלך שלב זה עשויים הפקדים בדף לעורר אירועים משלהם. לדוגמה, במקרה שהמשתמש בוחר אפשרות אחרת ברשימה עשוי האובייקט <i>DropDownList</i> לעורר את האירוע <i>SelectedIndexChanged</i> . באופן דומה, במקרה שהמשתמש משנה את הטקסט לפני ה- <i>postback</i> לשרת, עשוי <i>TextBox</i> לעורר את האירוע <i>TextChanged</i> . כשמסתיים עיבוד כל האירועים, הדף מעורר את האירוע <i>LoadComplete</i> . במהלך שלב זה מתעורר האירוע <i>PreRender</i> כדי לאותת שהדף עומד לעבור רינדור (render) לדפדפן. זמן קצר אחר כך מתעורר <i>SaveStateComplete</i> כדי לציין שהדף סיים לאחסן את כל המידע הרלוונטי עבור הפקדים ב- <i>ViewState</i> .
Rendering הפקת הדף	רינדור הוא השלב שבו הפקדים (והדף עצמו) מפיקים את קוד HTML שלהם לדפדפן.

שלב במחזור החיים	תיאור
Unload שלב הסיום	שלב זה הוא שלב הניקיון. זהו הרגע שבו הדף והפקדים יכולים לשחרר משאבים כמו חיבור לבסיס נתונים. במהלך שלב זה מתעורר האירוע Unload כדי שיוכל לטפל בכל ניקוי שאתה צריך לעשותו.

אתה ודאי שואל את עצמך לשם מה עליך לדעת כל זאת. הסיבה העיקרית לצורך בהבנה מסוימת של מחזור החיים של דף, היא בכך שפעולות מסוימות ניתנות לביצוע בשלבים מסוימים של מחזור החיים שלו בלבד. לדוגמה, שינוי דינמי של ערכת נושא חייב להיעשות מוקדם במחזור החיים של הדף, כפי שתראה להלן. כדי להבין באמת את מחזור החיים של דף ב-ASP.NET, עליך לדעת מעט יותר על פקדים, מצב (state), אירועים וכו'. לפיכך נעסוק במחזור החיים של דף פעם נוספת, בפרק 14 שבו תלמד ביתר פירוט על כל האירועים השונים שמתרחשים, וסדר התרחשותם.

אחראי על פיתוח הדף יכול לקבוע את כותרת הדף על ידי תכנות במהלך אירועים רבים. לפיכך, בדיקת נכונות הכותרת חייבת להיעשות מאוחר ככל האפשר במחזור החיים של הדף. התרגיל הבא מראה כיצד זה נעשה.

יישום של דף הבסיס

קל למדי ליישם דף בסיס. כל שעליך לעשות הוא להוסיף קובץ מחלקה לתיקייה App_Code, לקבוע כי המחלקה יורשת מהמחלקה Page ולהוסיף לקובץ מעט קוד, ובכך סיימת. מעט יותר קשה לוודא שכל דף באתר אכן יורש מדף הבסיס החדש, ולא מהמחלקה הרגילה System.Web.UI.Page. לרוע המזל, כשמשתמשים בקוד ברקע אין דרך להגדיר ליישום תצורה שתגרום לביצוע הדבר באופן אוטומטי, ולפיכך יש צורך לשנות כל דף באופן ידני. Visual Web Developer מקלה עליך מעט בכך שהיא מאפשרת לייצא תבנית דף שכבר מכילה קוד זה. לאחר שתציור את דף הבסיס בתרגיל הבא, תלמד כיצד לייצא דף לתבנית כדי שתוכל להוסיף כהרף עין קבצים שמשתמשים בדף הבסיס.

נסה זאת יצירת דף בסיס

1. לחץ לחיצה ימנית על התיקייה App_Code בסייר הפתרון, ובחר Add New Item. ברשימה Templates בחר Class וקרא לקובץ בשם BasePage. באפשרותך לבחור באיזה שם שתצצה, אבל BasePage מתאר בבחיור את מטרת המחלקה, והדבר מקל על הבנת פעולתה. כמו עם דפי ASPX, עליך לבחור מהרשימה הנפתחת את שפת התכנות המועדפת עליך.
2. נקה את תוכן הקובץ, והוסף את הקוד הבא לקובץ המחלקה הזה:

VB.NET

```
Public Class BasePage
    Inherits System.Web.UI.Page

    Private Sub Page_PreRender(ByVal sender As Object, ByVal e As System.EventArgs) _
        Handles Me.PreRender
```

```

If Me.Title = "Untitled Page" Or String.IsNullOrEmpty (Me.Title) Then
    Throw New Exception("Page title cannot be ""Untitled Page"" or an empty string.")
End If
End Sub

```

```
End Class
```

C#

```

using System;
using System.Web;

public class BasePage : System.Web.UI.Page
{
    private void Page_PreRender(object sender, EventArgs e)
    {
        if (this.Title == "Untitled Page" || string.IsNullOrEmpty(this.Title))
        {
            throw new Exception("Page title cannot be \"Untitled Page\" or an empty string.");
        }
    }

    public BasePage()
    {
        this.PreRender += new EventHandler(Page_PreRender);
    }
}

```

3. שמור את הקובץ וסגור אותו. פתח את הדף Login.aspx שיצרת קודם. פתח את קובץ הקוד ברקע שלו ושנה את קוד Inherits שלו (נקודתיים [: ב-C#) כדי שדף Login יירש מ-BasePage שיצרת קודם.

VB.NET

```

Partial Class Login
    Inherits BasePage

```

```

...
End Class

```

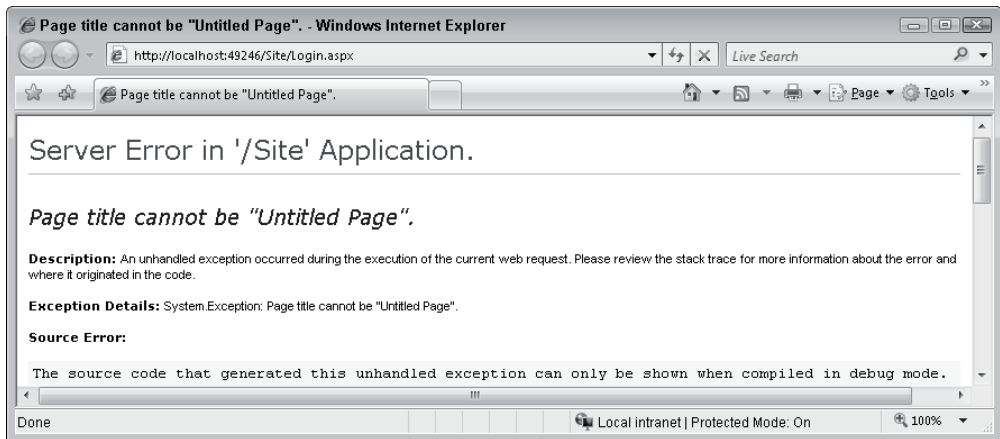
C#

```

public partial class Login : BasePage
{
    ...
}

```

4. שמור את הדף ואז בקש אותו בדפדפן על ידי הקשת Ctrl+F5. אם לא שינית קודם את כותרת הדף, אתה אמור לקבל בדפדפן את השגיאה המוצגת בתרשים 6-8 (יכול להיות שהמסך שלך יראה קצת שונה).



תרשים 6-8

5. חזור ל-VWD ופתח את דף ה-Login בתצוגת Markup View. אתר את המאפיין Title בקצה ההנחיה Page @, ושנה אותה ל- "Log in to Planet Wrox". קטע הקוד הבא מציג את ההנחיה Page @ ב-VB. גרסת C# כמעט זהה, פרט לתכונה Language. **שים לב:** אחד השינויים של SP1 הוא שכרירת המחדל ל-Title הוא מחרוזת ריקה.

```
<%@ Page Language="VB" MasterPageFile="~/MasterPages/MasterPage.master"
    AutoEventWireup="false" CodeFile="Login.aspx.vb" Inherits="Login"
    title="Log in to Planet Wrox" %>
```

6. חזור על צעדים 3 ו-5 עבור כל הדפים באתר. כדי לזרז זאת מעט, השתמש ב- Find and Replace כדי להחליף במהירות את כל המופעים של System.Web.UI.Page ב-BasePage. ודא שלא החלפת בטעות את המופעים גם בקובץ BasePage עצמו, בתיקייה App_Code כדי למנוע מקרה כזה, ודא שהנך מחפש בקבצי קוד ברקע בלבד. כך תעשה זאת:

⊙ פתח את תיבת הדו-שיח Replace in Files (הקש Ctrl+Shift+H או בחר Edit ← (Replace in Files ← Find and Replace).

⊙ בתיבה Find What הכנס System.Web.UI.Page. בתיבת הטקסט Replace With הכנס BasePage.

⊙ הרחב את הקטע Find Options, ובתיבת הטקסט Look at These File Types הכנס *.aspx.cs או *.aspx.vb, בהתאם לשפה שבשימוש. פעולה זו גורמת להתעלמות משינויים בקובץ BasePage, מכיוון שיש לו סיומת אחת בלבד של vb או cs.

⊙ לחץ Replace All ואז לחץ Yes כדי לאשר את הפעולה Replace.

7. שמור את השינויים שעשית בכל דף פתוח, ואז דפדף שוב ל-Login.aspx. אם הכל פעל כמתוכנן, השגיאה אמורה להיעלם ותראה כעת את דף Login שאתה מצפה לו.

למרות זאת, זכור שכל שאר הדפים באתר יוציאו הודעת שגיאה כשתנסה לגשת אליהם. התיקון קל: קבע לכולם Title חדש. במקרה שאינך אוהב התנהגות זו, גש

למחלקה BasePage והפוך את השורה שמתחילה ב- Throw New Exception להערה.
אל תשכח לקבוע לדפים כותרת בעלת משמעות, מכיוון שלא תקבל התראות נוספות.

כיצד זה פועל

כברירת מחדל, כל הדפים באתר יורשים מהמחלקה Page שמוגדרת במרחב השמות System.Web.UI. הדבר מקנה לדפים אלה את ההתנהגות הנדרשת כדי לגרום להם לפעול כדפי אינטרנט שיכולים להיות מבוקשים על ידי הדפדפן ומעובדים על ידי השרת. מכיוון שמודל ההורשה של .NET מאפשר ליצור שרשרת מחלקות שיורשות זו מזו, תוכל בקלות להכניס את מחלקת דף הבסיס שלך בין דף web לבין מחלקה Page רגילה. עשה זאת על ידי שינוי משפט Inherits (נקודתיים [:] ב-C#) ל-BasePage החדשה שלך.

VB.NET

```
Partial Class Login
    Inherits BasePage
```

C#

```
public partial class Login : BasePage
```

בתוך המחלקה BasePage החדשה עליך להוסיף מטפל אירוע שיפועל כאשר המחלקה תעורר את האירוע PreRender. כפי שלמדת קודם, אירוע זה מתעורר כמעט בסוף מחזור החיים של הדף, כשהדף כולו מוגדר ומוכן להיות מרונדר עבור הלקוח.

VB.NET

```
Private Sub Page_PreRender(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Me.PreRender
```

```
    ' Implementation here
```

```
End Sub
```

C#

```
private void Page_PreRender(object sender, EventArgs e)
{
    // Implementation here
}
```

שים לב ש-VB משתמשת במילת המפתח Handles כדי לומר למהדר שהשיטה Page_PreRender תשמש לטיפול באירוע. ב-C# אתה צריך לקשר את המטפל (handler) הזה באופן ידני. פונקציית הבנייה של המחלקה היא מקום טוב לעשות זאת:

```
public BasePage()
{
    this.PreRender += new EventHandler(Page_PreRender);
}
```

שורת הקוד המוצללת משרתת אותה מטרה כמו מילת המפתח Handles ב-VB.NET: היא אומרת למהדר איזו שיטה להפעיל כשהדף מעורר את האירוע PreRender.

הקוד של מטפל האירוע בודק את כותרת הדף הנוכחי. אם כותרת הדף היא עדיין "Untitled Page" או מחרוזת ריקה (ברירת המחדל לכל דף חדש שאתה מוסיף לפרויקט), המטפל זורק חריג (שגיאה) המוצג במסך.

VB.NET

```
If Me.Title = "Untitled Page" Then
    Throw New Exception("Page title cannot be ""Untitled Page"" or an empty string.")
End If
```

C#

```
if (this.Title == "Untitled Page")
{
    throw new Exception("Page title cannot be \"Untitled Page\" or an empty string.");
}
```

שים לב לאופן השימוש במילות המפתח Me (ב-VB.NET) ו-this (ב-C#). מילות מפתח אלו מתייחסות תמיד בהקשר למחלקה שבה משתמשים. בדוגמה לעיל, Me (ו-this) מתייחסות למופע הנוכחי של המחלקה Page. מופע זה של Page הוא בעל מאפיין Title שמאפשר לבדוק אם הוא כולל ערכים לא רצויים. אם הוא מכיל עדיין את כותרת ברירת המחדל, הקוד יעורר (או יזרוק, throw) מצב חריג. דבר זה מפסיק מיידית את ביצוע הדף. לך, כמפתח הדף, יש אפשרות לתקן את הבעיה על ידי קביעת כותרת תקפה לפני שהדף יוצג. בפרק 17 תלמד יותר אודות חריגים, כיצד למנוע אותם וכיצד לטפל בהם.

שתי השפות משתמשות בפורמט שונה כדי להציג גרשיים (") בהודעת השגיאה. ב-VB צריך להכפיל את הגרשיים. ב-C# צריך להקדים את הגרשיים בלוכסן אחורי (i). בשני המקרים הודעת השגיאה מוצגת בדפדפן עם זוג אחד של גרשיים בכל צד שלה:

```
Page title cannot be "Untitled Page" or an empty string.
```

קטע קוד קצר זה ב-BasePage מונע את האפשרות "לעלות לשידור" עם האתר שמכיל דפים שכותרתם היא עדיין לפי ברירת המחדל חסרת המשמעות.

כל דף באתר אמור כעת לרשת מדף הבסיס החדש, לפיכך עליך ליצור תבנית דף שכבר מכילה את הקוד (ירושה מדף הבסיס במקום מ-Page) הנכון בקוד ברקע שלה. הדבר יוכל להקל על הוספת הדף הנכון לאתר כבר מן ההתחלה.

יצירת תבניות דף לשימוש חוזר

התוכנה Visual Web Developer מגיעה עם כלי מעולה לייצא תבניות עבור כמה סוגי קבצים, כולל דפי ASPX, קבצי מחלקה, ואפילו קבצי CSS. על ידי יצירת תבנית מותאמת, תוכל להגדיר פעם אחת את הקוד או את סימני העריכה שאתה זקוק להם בכל קובץ, ואז - ליצור קבצים חדשים שמבוססים על תבנית זו. מהלך כזה יבטיח יצירה מהירה של דפים באתר שלך (הקוד כבר קיים - יקטין את כמות הקוד שצריך להקליד) וכן יבטיח כי לא תשכח לרשת מדף הבסיס. התרגיל הבא מראה כיצד ליצור תבניות משלך.

נסה זאת יצירת תבנית דף לשימוש חוזר

בתרגיל זה תלמד כיצד ליצור קובץ תבנית עבור כל דפי ASPX החדשים שברצונך להוסיף לאתר. כדי למנוע התנגשויות עם דפים קיימים באתר הנוכחי שלך, עליך ליצור דף חדש זמני כדי להשתמש בו ליצירת התבנית. לאחר מכן תוכל למחוק את הקובץ הזמני.

1. הוסף דף אינטרנט חדש לאתר וקרא לו `Template.aspx`. ודא שהוא משתמש בקוד ברקע ובשפת התכנות הרצויה, ושהוא מבוסס על הדף הראשי בתיקייה `MasterPages`.
2. פתח את הקוד ברקע של הדף החדש, ושנה בו את השורה `Inherits` (נקודתיים ב-`C#`) כדי שהדף יירש מ-`BasePage` במקום מ-`System.Web.UI.Page`. כמו כן, שנה את שם המחלקה מ-`Template` ל-`$safeitemname$`.

VB.NET

```
Partial Class $safeitemname$  
    Inherits BasePage
```

```
...
```

```
End Class
```

C#

```
public partial class $safeitemname$ : BasePage  
{
```

```
...
```

```
}
```

בגרסת `C#`, ודא שאינך מסיר בטעות קוד קיים, כמו משפטי `using` או השיטות `Page_Load`. אל תדאג לשגיאות מהדר שאתה עלול לקבל בגלל תווים בלתי צפויים כמו `$`. ברגע שתתחיל להוסיף דפים שמבוססים על תבנית זו, השם `$safeitemname$` יוחלף בשם הדף שאתה מוסיף.

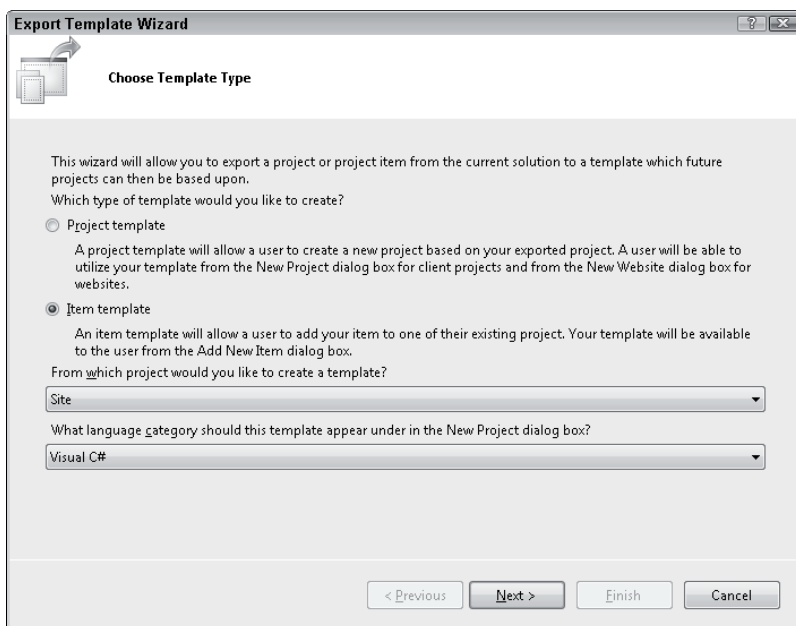
3. עבור לתצוגת Markup View של הדף `Template.aspx`, ושנה את המאפיין `Inherits` מ-`Template` ל-`$safeitemname$`.

```
<%@ Page Language="C#" MasterPageFile="~/MasterPages/MasterPage.master"  
    AutoEventWireup="true" CodeFile="Template.aspx.cs" Inherits="$safeitemname$"  
    Title="Untitled Page" %>
```

התעלם מהמאפיין CodeFile, VWD תשנה את ערכו באופן אוטומטי לשם קובץ קוד ברקע הנכון, בכל פעם שתוסיף דף חדש לאתר.

4. לפי בחירתך, הוסף לקבצים קוד אחר שברצונך שיופיע בדפים כברירת מחדל, כמו למשל, בלוק הערה עם הודעת זכויות יוצרים.

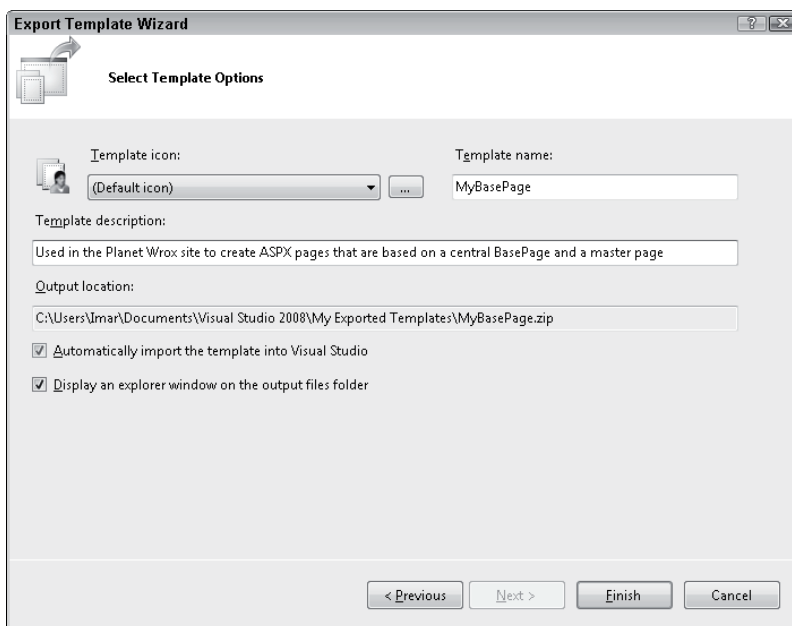
5. שמור את השינויים שבוצעו בדף, ובחר File ← Export Template. בתיבת הדו-שיח שמופיעה בחר Item Template, ומהרשימה הנפתחת שבתחתית המסך בחר בשפת התכנות שלך. ראה תרשים 6-9.



תרשים 6-9

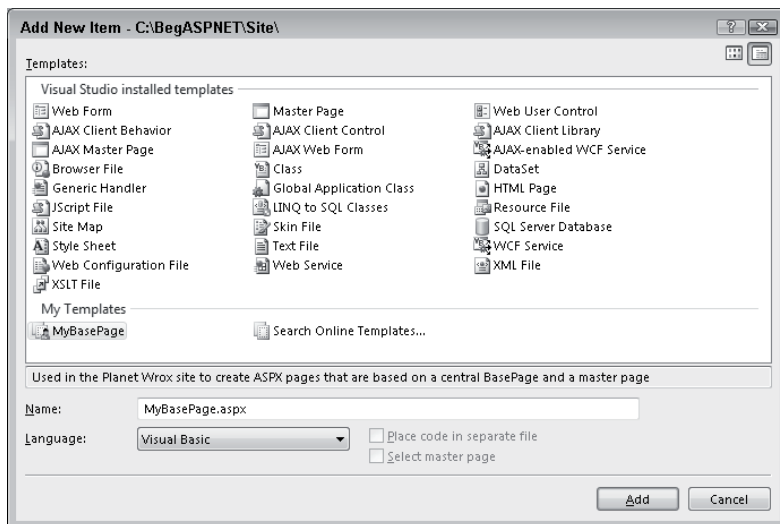
6. לחץ Next וסמן את האפשרות עם שם בקובץ: Template.aspx שתמצא בסמוך לתחתית הרשימה. לחץ Next שוב כדי לעבור לתיבת הדו-שיח Select Item References.

7. אינך צריך להגדיר מאומה בתיבת הדו-שיח Select Item References. אם אתר האינטרנט שלך מתייחס למאספים (קבצי dll) מסוימים, ביכולתך לסמן אותם כאן, כדי שבפעם הבאה שתוסיף קובץ שמבוסס על תבנית זו, VWD תוסיף את ההתייחסויות למאספים באופן אוטומטי. במקרה זה, לחץ שוב NEXT כדי לעבור למסך Select Template Options. הקלד את שם הדף החדש **MyBasePage** תחת Template Name, ולפי בחירתך הקלד הערה קצרה שמתארת את מטרת התבנית. תרשים 6-10 מציג את תיבת הדו-שיח הסופית.



תרשים 6-10

8. לחץ Finish כדי לסיים את יצירת התבנית. VWD פותחת סייר חלונות שמציג את התבנית החדשה כקובץ ZIP. אפשר לסגור את החלון מכיוון שאין לנו צורך בו. אם ברצונך לבצע תרגיל זה הן ב-VB.NET והן ב-C#, שנה את שם קובץ ה-zip לפני שאתה מייצא קובץ זהה עבור השפה האחרת, כדי שלא תדרוס את הכתוב בו. כדי לשנות את שם הקובץ: פתח את סייר החלונות, לך ל-My Documents (Documents ב-Vista) ודפדף אל Visual Studio 2008\Templates\ItemTemplates. כעת תמצא קובץ בשם MyBasePage.zip, שאתה יכול לשנות את שמו למשהו כמו MyBasePageCS.zip. שים לב שמיקום הקובץ שונה מזה שראית בתרשים 6-10; מקום הפלט מכיל רק עותק של התבנית שיוצאה, ואפשר להשתמש בקובץ זה כגיבוי.
9. חזור ל-VWD ומחק את הקובץ Template.aspx שיצרת. לחץ לחיצה ימנית על הפרויקט בסייר הפתרון, ובחר Add New Item. שים לב שהתבנית המותאמת שלך מופיעה עכשיו באזור של My Templates, בתחתית תיבת הדו-שיח שמוצגת בתרשים 6-11. אם תלחץ על התבנית, היא תציג את התיאור שהכנסת לתוכה קודם לכן.
10. הקלד שם חדש לדף, כמו למשל TestPage.aspx, ולחץ Add כדי להוסיפו לאתר. התבונן בסימני העריכה ובקוד ברקע של הקובץ. שים לב שהשם \$safeitemname\$ הוחלף ל-TestPage, כדי לשקף את השם החדש של הדף. אם הכל נראה תקין, אפשר למחוק את TestPage.aspx מכיוון שהוא אינו בשימוש באתר Planet Wrox.



תרשים 6-11

כיצד זה פועל

כשאתה מייצא את התבנית, התוכנה VWD יוצרת קובץ ZIP עם הקבצים הנחוצים, בתרגיל זה - קובץ ASPX וקובץ קוד ברקע. התבנית מאוחסנת בתת-תיקייה ItemTemplates של התיקייה Visual Studio 2008, תחת התיקייה Documents. כמה מהקבצים בקובץ ZIP מכילים את שומר המקום \$safeitemname\$. כשאתה מוסיף לאתר קובץ חדש שמבוסס על התבנית, תוך שימוש בתיבת הדו-שיח Add New Item, מחליפה VWD את \$safeitemname\$ בשם הממשי של הדף. בתרגיל שלנו הקלדת TestPage.aspx כשם החדש של הדף, ואכן VWD יצרה מחלקה בשם TestPage בקוד ברקע, אשר יורשת מדרך הבסיס BasePage. בנוסף ל-\$safeitemname\$ קיימים כמה שומרי מקום אחרים שתוכל להשתמש בהם. סרוק את אתר MSDN בכתובת: <http://msdn2.microsoft.com> כדי לאתר את המונח \$safeitemname\$ ולמצוא את הפרמטרים האחרים של התבנית.

אם צריך להכניס שינוי בתבנית המיוצאת, באפשרותך לחזור על כל תהליך הייצוא, או שתערוך את הקבצים אשר נמצאים בקובץ ה-zip באופן ידני.

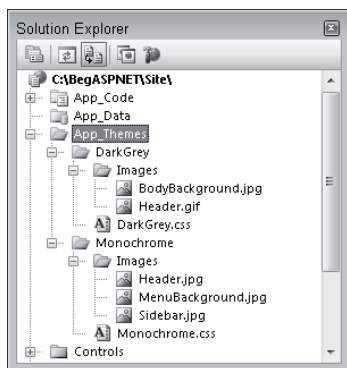
עם התבנית המיוצאת יש בידך כעת דרך מהירה מאוד להוסיף לאתר דפים שיורשים מהמחלקה BasePage. אינך צריך עוד לשנות באופן ידני את הקוד ברקע של קובץ המחלקה, או את סימני העריכה של הדף.

פרט לדפים ראשיים ולמחלקה BasePage המרכזית, יש דרכים נוספות ליצירת אתרים בעלי מראה עקבי. אחת מהן היא באמצעות ערכות נושא.

ערכות נושא

עד כה ראית כיצד ליצור דף ראשי כדי להגדיר את המראה והתחושה הכלליים של הדפים באתר. למדת גם כיצד לרכז התנהגות משותפת לדפים על ידי שימוש בדף בסיס מרכזי. יש דרכים נוספות להשפיע על המראה והתחושה של האתר: **ערכות נושא** (themes) ו**מעטפות** (skins). נדון תחילה בערכות נושא, ובמעטפות נעסוק בהמשך הפרק מכיוון שהם חלק מערכות הנושא.

ערכת נושא (theme) היא אוסף קבצים שמגדיר את המראה של הדף. האוסף יכול לכלול קבצי skin, קבצי CSS ותמונות. ערכות הנושא מוגדרות בתיקייה מיוחדת, App_Themes, שנמצאת בשורש האתר. בתוך תיקייה זו אתה צריך ליצור תת-תיקיות, אחת או יותר, שמגדירות את ערכות הנושא הממשיות. כל תת-תיקייה יכולה לכלול כמה קבצים שמהווים את ערכת הנושא. תרשים 6-12 מציג את סייר הפתרון של אתר אינטרנט שמגדיר שתי ערכות נושא: DarkGrey ו-1 Monochrome.



תרשים 6-12

בכל פעם שערכת הנושא מופעלת, מתווסף בקטע <head> של הדף שלך קישור אל כל קובץ CSS שנמצא בתיקיית הערכה. הוספת הקישור מתרחשת באופן אוטומטי. בהמשך תראה כיצד זה פועל. ההתייחסות לתמונות שבתיקיית ערכות הנושא היא באמצעות קבצי CSS. אפשר להשתמש בתמונות לשינוי אלמנטים משותפים של אתר האינטרנט, כמו תמונות רקע, או תמונות שמשמשים בהן ברשימות מובלטות או רשימות ניווט.

כדי ליצור ערכת נושא, בצע את הפעולות הבאות:

- ⊙ צור את התיקייה המיוחדת App_Themes, אם היא לא נמצאת עדיין באתר.
- ⊙ לכל ערכת נושא שברצונך ליצור, צור תת-תיקייה עם שם הערכה, כמו Monochrome או DarkGrey שבתרשים 6-12.
- ⊙ לבחירתך, צור קבצי CSS, אחד או יותר, שיהיו חלק מהערכה. אין דרישה ששמות קבצי ה-CSS יהיו כשמות ערכות הנושא, אבל הדבר יכול לסייע בזיהוי הקבצים הנכונים. כל קובץ CSS שתוסיף לתיקיית הערכה יתווסף לדף באופן אוטומטי בזמן ביצוע.

- ⊙ לבחירתך, הוסף תמונה אחת או יותר, לתיקיית הערכה. קבצי CSS צריכים להתייחס לתמונות אלו על ידי רישום מיקומם היחסי. תרשים 6-12 מראה שלערכת הנושא DarkGrey יש שתי תמונות, שמשתמשים בהן בכותרת וברקע של האתר.
- ⊙ לבחירתך, הוסף קבצי skin אחד או יותר לתיקיית הערכה. קבצי skin מאפשרים להגדיר מאפיינים ייחודיים (כמו ForeColor או BackColor) לפקד מסוים, אשר חלים עליו בזמן ביצוע.

לאחר ביצוע צעדים אלה, ביכולתך לקבוע את תצורת האתר, או את תצורת דף האינטרנט בו, באופן שישתמשו בערכת נושא זו. כדי שתהיה מסוגל להגדיר את ערכת הנושא הנכונה, עליך להיות מודע לכך שקיימים שני סוגים של ערכות נושא.

סוגים שונים של ערכות נושא

לדף ASP.NET יש שני מאפיינים שונים שמאפשרים להגדיר ערכת נושא: המאפיין Theme והמאפיין StyleSheetTheme. שני המאפיינים משתמשים בערכות הנושא שהגדרת בתיקייה App_Themes. למרות שתחילה הם נראים דומים מאוד, הרי התנהגותם בזמן ביצוע שונה. המאפיין StyleSheetTheme מוחל מוקדם מאוד במחזור החיים של הדף, זמן קצר לאחר יצירת המופע שלו. משמעות הדבר היא שדף מסוים יכול לדרוש את הגדרות ערכת הנושא על ידי החלה של מאפיינים הכלולים בפקדים. כך, לדוגמה, ערכת נושא עם קובץ skin שמגדירה BackColor של לחצן כירוק, יכולה להידרס על ידי הכרזת הפקד הבאה המכילה מאפיין BackColor וערכו אדום:

```
<asp:Button ID="Button1" runat="server" Text="Button" BackColor="red" />
```

מצד שני, המאפיין Theme מוחל בשלב מאוחר במחזור החיים של הדף, ולמעשה דורס כל התאמה אישית שעשית לפקדים מסוימים כלשהם.

בחירה בין Theme לבין StyleSheetTheme

מכיוון שהמאפיינים מסוג StyleSheetTheme יכולים להידרס על ידי הדף, בעוד המאפיין Theme יכול לשוב ולדרוש מאפיינים אלה, ברור ששניהם משרתים מטרות שונות. עליך להגדיר את StyleSheetTheme אם ברצונך לספק הגדרות ברירות מחדל לפקדים. כלומר, StyleSheetTheme יכול לספק לפקדים ברירות מחדל אשר יכולות להידרס ברמת הדף. במאפיין Theme תשתמש כשברצונך לאכוף את המראה והתחושה של הפקדים שלך. מכיוון שהגדרות Theme אינן ניתנות לדריסה (עקב החלתם בשלב מאוחר יחסית), ולמעשה משמידות כל התאמה אישית שביצעת, אתה יודע בוודאות שהפקדים שלך ייראו כפי שהגדרת אותם בערכת הנושא.

עם ערכת נושא מובטח לך שכל השינויים שתעשה בערכה יופצו לפקדים בדפי האינטרנט שלך, אלא אם כן הפקד מבטל את הערכה על ידי הגדרת EnableTheming ל-False. על הגדרת מאפיין זה והשפעתו תלמד לקראת סוף הפרק.

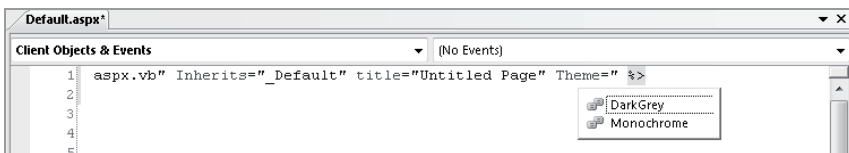
החלת ערכות נושא

יש שלוש אפשרויות להחיל ערכת נושא על אתר: ברמת הדף בהנחיה Page, ברמת האתר על ידי שינוי הקובץ web.config, ועל ידי תכנות.

⊙ הגדרת ערכת הנושא ברמת הדף. קל להגדיר Theme או StyleSheetTheme ברמת הדף: הגדר בהנחיה Page של הדף את המאפיין הרצוי.

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="_Default" Theme="DarkGrey" %>
```

כדי להחיל ערכת נושא שהגדרותיה יכולות להידרס על ידי הדף, החלף את המאפיין Theme ב-StyleSheetTheme. ברגע שתקליד Theme="", תציג VWD את רשימת כל ערכות הנושא שהיא מוצאת בתיקייה App_Theme. ראה תרשים 6-13.



תרשים 6-13

⊙ הגדרת ערכת הנושא ברמת האתר. כדי לאכוף ערכת נושא באתר כולו, אפשר להגדיר את הערכה בקובץ web.config. כדי לעשות זאת: פתח את הקובץ web.config, אתר את האלמנט <pages>, והוסיף לו את התכונה theme:

```
<pages theme="DarkGrey">
...
</pages>
```

מכיוון ש-XML בקובץ web.config רגישה לרישיות, ודא שאתה מקליד theme באותיות רגילות בלבד.

⊙ הגדרת ערכות נושא בתכנות. הדרך השלישית להגדרת ערכת נושא היא באמצעות קוד, כלומר תכנות. בגלל הדרך שבה פועלת ערכת נושא, עליך לעשות זאת מוקדם במחזור החיים של הדף. בתרגיל מאוחר יותר תראה כיצד זה פועל.

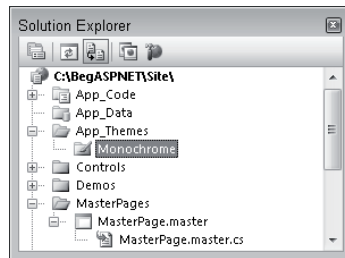
תרגיל נסה זאת הבא מראה כיצד פועלות ערכות נושא. תלמד כיצד ליצור ערכה, להוסיף את הוראות CSS הדרושות, ולהגדיר את תצורת היישום כדי שישתמש בערכת הנושא החדשה.

נסה זאת יצירת ערכת נושא חדשה עבור אתר אינטרנט

בתרגיל זה תיצור שתי ערכות נושא: Monochrome ו-DarkGrey. לכל ערכה תוסיף את העיצוב והעריכה של CSS, שיחולו על כל האתר באופן אוטומטי. תגדיר גם את האתר כך שישתמש באחת משתי הערכות, ואז תעבור לערכה השנייה כדי לראות את ההבדלים.

1. הוסף לאתר את התיקייה המיוחדת App_Themes. כדי לעשות זאת: לחץ לחיצה ימנית על הפרויקט בסייר הפתרון, ובחר Add ASP.NET Folder ← Theme. פעולה זו תיצור את התיקייה App_Themes, וגם תגרום לכך שמייד תיווצר גם תת-תיקייה בשם Theme1 (ברירת המחדל) לערכת הנושא. הקלד **Monochrome** כשם חדש במקום שם ברירת המחדל. סייר הפתרון אמור להיראות עכשיו כמו בתרשים 6-14.

2. העבר את הקובץ Styles.css מהתיקייה Styles לתיקייה החדשה Monochrome. אתה יכול לגרור את הקובץ ישירות אל התיקייה החדשה, או להשתמש ב-Ctrl+X כדי לגזור את הקובץ, ללחוץ על התיקייה Monochrome, ולהקיש Ctrl+V כדי להדביקו בתוכה. עזוב את התיקייה הריקה Styles, נשתמש בה שוב מאוחר יותר.



תרשים 6-14

3. שנה את שם הקובץ מ-Styles.css ל-Monochrome.css, כדי שמאוחר יותר תוכל לראות בבחירות מאין מגיע קוד CSS.

4. מכיוון שהעיצוב והעריכה העיקריים יישלטו מכאן והלאה על ידי ערכת הנושא, אין עוד צורך שהאלמנט <link> בקטע <head> של הדף הראשי יציב על קובץ CSS הישן, ואפשר להסיר אותו. לשם כך: פתח את הדף הראשי, עבור ל-Markup View והסר מהקוד את השורה המוצללת הבאה.

```
<head runat="server">
  <title>Welcome to Planet Wrox</title>
  <asp:ContentPlaceHolder id="head" runat="server">
  </asp:ContentPlaceHolder>
  <link href="../../Styles/Styles.css" rel="stylesheet" type="text/css" />
</head>
```

5. הצעד הבא הוא להחיל את ערכת הנושא על האתר כולו. כדי לעשות זאת: פתח את הקובץ web.config בשורש האתר, אתר את האלמנט <pages> והוסף את התכונה theme.

```
<pages theme="Monochrome">
```

6. כדי לבדוק את ערכת הנושא, שמור את כל השינויים ובקש את הדף Default.aspx בדפדפן. עיצוב האתר חייב להיות זהה למה שהיה. אם מתקבלת הודעת שגיאה על כותרת דף לא חוקית, חזור ל- Visual Web Developer ושנה את התכונה Title של ההנחיה Page של Default.aspx למשהו כמו למשל: "Welcome to Planet Wrox".

⊙ במקום התקשרות לקובץ CSS מהדף הראשי, קוד CSS כלול עכשיו במקור הדף באמצעות ערכת הנושא שהוגדרה בקובץ web.config. כדי לראות כיצד זה פועל, פתח בדפדפן את קוד HTML המקורי של הדף. בראש תראה את הקוד הבא (העיצוב והעריכה שונו מעט כדי לשפר את הקריאות).

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Untitled Page</title>
  <link href="App_Themes/Monochrome/Monochrome.css"
        type="text/css" rel="stylesheet" />
</head>
<body>
```

⊙ שים לב שלקטע <head> של הדף הוכנס קישור מתיקית ערכת הנושא Monochrome אל גיליון הסגנון. ASP.NET runtime עושה זאת לכל קובץ css שהוא מוצא בתיקייה של ערכת הנושא הפעילה. כדי למנוע מהדפדפן לכלול ולטעון קבצים לא נחוצים, עליך לוודא שתיקית ערכת הנושא נקיה. כמו כן תן את הדעת לכך שה- <link> התווסף בדיוק לפני תגית הסגירה </head>. הדבר מבטיח שקובץ הערכה יתווסף לאחר כל שאר הקבצים שאולי הוספת בעצמך (למשל באמצעות הדף הראשי). הדבר נוגד את האופן שבו פועלת התכונה StyleSheetTheme. מכיוון שסוג זה של ערכת נושא מאפשר להגדרות שלו להידרס, הוא מיובא בראש הקובץ. על ידי כך ניתן מרחב פעולה לקבצי CSS אחרים שבאים בעקבותיו לשנות את המראה והתחושה של הדף.

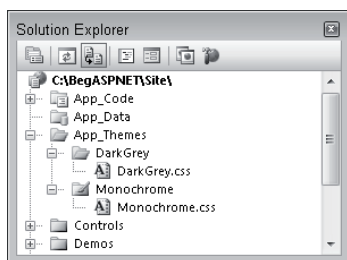
7. חזור אל Visual Web Developer ופתח את קובץ הדף הראשי ב- Design View. שים לב שכל העיצוב נעלם ו-VWD מציגה עכשיו שוב את העיצוב והעריכה הבסיסיים של הדף. לרוע המזל, VWD אינה מציגה את ערכת הנושא שהגדרת תוך שימוש בתכונה theme. אפשר להתגבר על מגבלה זו על ידי הגדרת StyleSheetTheme במקום theme. כדי לעשות זאת: פתח שוב את הקובץ web.config, אתר את האלמנט <pages> והוסף את התכונה הבאה:

```
<pages theme="Monochrome" styleSheetTheme="Monochrome">
```

8. שמור את כל השינויים שבוצעו ב-web.config, סגור את הדף הראשי ופתח אותו שוב, ועבור ל- Design View. תראה שכעת VWD מחילה על הדפים את מידע העיצוב הנכון.

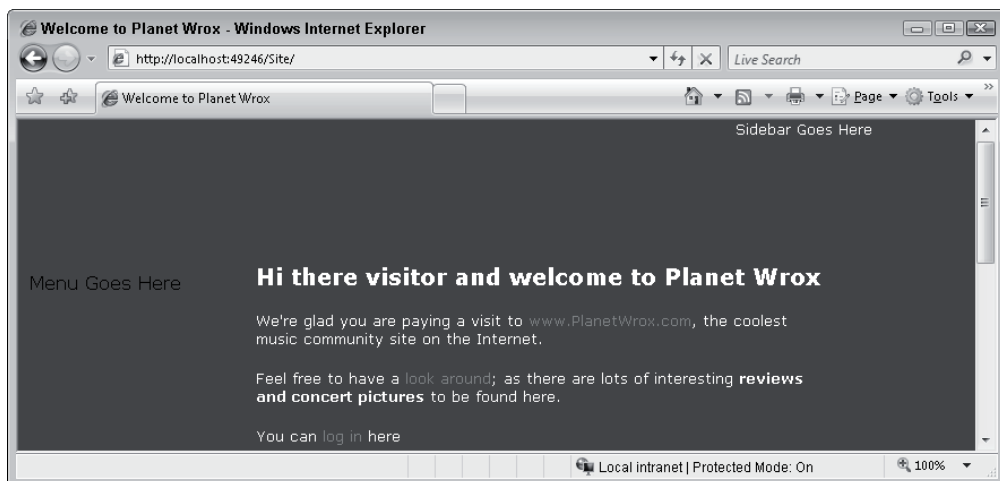
9. כדי להוסיף ערכת נושא חדשה לאתר, צור תיקייה חדשה תחת App_Themes וקרא לה DarkGrey. פתח את התיקייה שממנה שלפת את הקוד הנטען שמגיע עם הספר. אם עקבת אחר ההוראות בפרק ההקדמה, תיקייה זו צריכה להיות ב-C:\BegASPNET\Resources. פתח את התיקייה Chapter 06 ואחר כך את התיקייה DarkGrey. גרור את הקובץ DarkGrey.css מסייר הפתרון אל תיקיית ערכת הנושא DarkGrey ב-VWD. סייר הפתרון צריך כעת להיות דומה לתרשים 6-15.

בתרגיל מאוחר יותר תוסיף את התמונות שקובץ CSS פונה אליהן.



תרשים 6-15

10. פתח את הקובץ web.config פעם נוספת, ושנה את שני המופעים של Monochrome באלמנט <pages> ל-DarkGrey. שמור את השינויים שוב והקש Ctrl+F5. במקום הערכה Monochrome הירקרה, תראה עכשיו את האתר כשהערכה DarkGrey מוחלת עליו, כפי שמוצג בתרשים 6-16.



תרשים 6-16

כיצד זה פועל

בתרגיל זה החלתי את ערכת הנושא Monochrome על ידי שינוי האלמנט <pages> בקובץ web.config. כשרכיב runtime רואה שערכת נושא פעילה, הוא סורק את תיקיית הערכה ומחפש קבצי css, וכולל בקטע <head> של הדף קישור לכל הקבצים מסוג זה שמצא. במקרה של ערכת הנושא Monochrome הוא מוצא את הקובץ Monochrome.css ומוסיף אותו באופן אוטומטי לקטע <head>. תהליך זהה קרה כששיניתי את הערכה ל-DarkGrey.

כדי לאפשר תמיכה בזמן עיצוב ב- Visual Web Developer, עליך לשנות את styleSheetTheme גם בקובץ web.config. החיסרון היחיד של צעד זה הוא שקובץ CSS הרלוונטי ייכלל עכשיו פעמיים: פעם אחת עבור Theme ופעם שנייה עבור styleSheetTheme. מכיוון שאותו קובץ בדיוק נכלל פעמיים, הדבר אינו משפיע על העיצוב והעריכה של האתר. כל הבוררים (selectors) בקובץ השני מבטלים את אלה של הקובץ הראשון. עם זאת, אם אתה רואה בכפילות

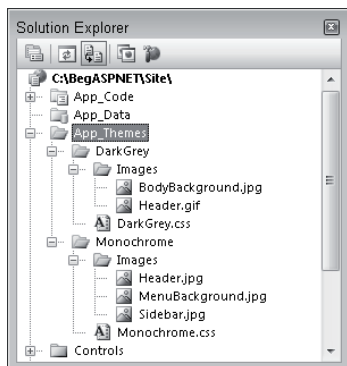
זו בזכו של משאב המחשב, מחק את התכונה styleSheetTheme מהקובץ web.config לפני שהאתר עולה לאוויר.

העיצוב והעריכה של הדף השתנו באופן קיצוני כתוצאה מקוד CSS שבקובץ DarkGrey.css. אם ברצונך לדעת איזה קוד CSS מכיל הקובץ, ואיזה אלמנטים בדף שונו על ידו, פתח את הקובץ ב-VWD. יש בו הערות רבות שמתארות כל בורר לפרטיו.

הרחבת ערכות נושא

פרט לקבצי CSS ו-skins (שבהם נדון לקראת סוף הפרק), ערכת נושא יכולה להכיל גם תמונות. השימוש הנפוץ ביותר של תמונות ערכת נושא הוא בהתייחסות אליהן מ-CSS. כדי שהשימוש בהן יהיה יעיל חשוב להבין כיצד CSS מתייחס לתמונות.

בהתאם לעיצוב, החיפוש אחר תמונה שבורר CSS מתייחס אליה הינו יחסי למיקום של קובץ CSS, אלא אם כן אתה קובע נתיב חיפוש שמתחיל בלוחסן נטוי קדימה (/) שמציין את שורש האתר. התבונן למשל, בתיקייה App_Themes שמוצגת בתרשים 6-17.



תרשים 6-17

כדי להתייחס לקובץ MenuBackground.jpg שבתיקייה Images של ערכת הנושא Monochrome, תוכל להוסיף את קוד CSS הבא לקובץ Monochrome.css:

```
#MenuWrapper
{
    background-image: url(Images/MenuBackground.jpg);
}
```

אם תרצה להתייחס לתמונה בתיקייה Images שבשורש האתר, תשתמש בקוד CSS הבא:

```
background-image: url(/Images/MenuBackground.jpg);
```

שים לב ללוחסן הנטוי קדימה בחזית נתיב התמונה, לציון שורש האתר. התחביר האחרון שימושי כשברצונך לשתף תמונות בין ערכות נושא שונות. שים את התמונות בתיקייה שמחוץ

לערכה מסוימת, כמו למשל התיקייה Images שבשורש, והשתמש בתחביר מבוסס-השורש לעיל כדי לפנות אליה. הפרק הבא מעמיק הרבה יותר בצורות השונות שיכולה ללבוש URL (כתובת אינטרנט) כדי שאפשר יהיה לפנות למשאב כמו תמונות.

נסה זאת הוספת תמונות לערכת נושא

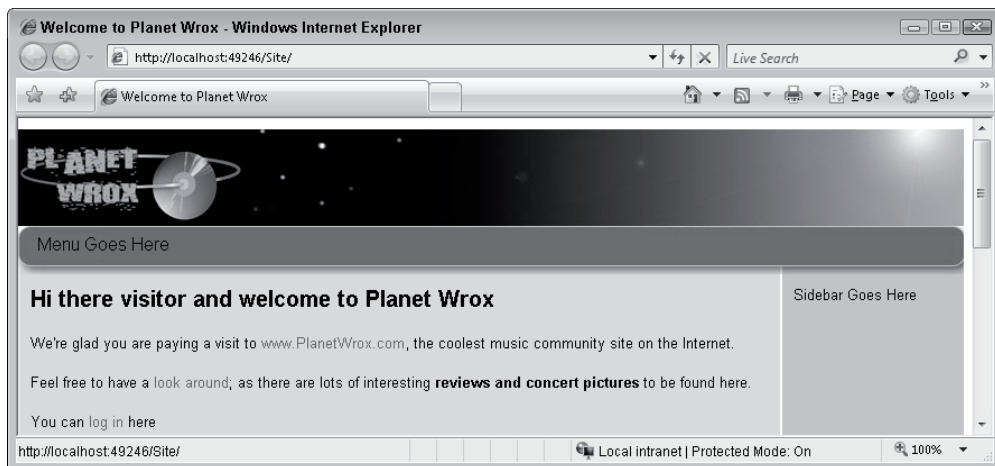
בתרגיל זה תוסיף תמונות וקבצי CSS לאתר, כדי להשלים את שתי ערכות הנושא. התמונות וקבצי CSS זמינים בקובץ zip שמכיל את הקוד לטעינה עבור פרק זה, ונמצא באתר www.wrox.com. מכיוון שאת ערכת הנושא Monochrome תכתוב בקובץ Monochrome.css, הרי שאם עשית התאמות כלשהן לקובץ זה, צור תחילה גיבוי שלו.

1. פתח את סייר הפתרון ונווט לקבצים שהורדת (C:\BegASPNET\Resources -ב). אם עדיין לא הורדת את הקבצים, בדוק בהקדמה ועקוב אחר ההוראות. פתח את התיקייה Chapter 06 ואחר כך את התיקייה Monochrome. בחר בתיקייה Images ובקובץ Monochrome.css.
2. ארגן את Visual Web Developer ואת סייר החלונות כך שתוכל להתבונן בשניהם בו-זמנית, ואז גרור את התיקייה והקבצים שנבחרו מחלון הסייר אל התיקייה Monochrome ב-VWD. כשתישאל לגבי דריסה של הקובץ Monochrome.css - לחץ Yes.
3. חזור על שני הצעדים הקודמים, אבל הפעם גרור את הקובץ DarkGrey.css ואת התיקייה Images מהתיקייה DarkGrey שבסייר החלונות אל תיקיית ערכת הנושא DarkGrey ב-VWD. סייר הפתרון נראה כעת כמו תרשים 6-17.
4. פתח את הדף Default.aspx בדפדפן על ידי שתלחץ עליו לחיצה ימנית ותבחר View in Browser. תראה את הדף עם תמונות מערכת הנושא DarkGrey, כפי שמוצג בתרשים 6-18.



תרשים 6-18

5. חזור ל-VWD, פתח את הקובץ web.config והחלף בין שני מאפייני ערכת הנושא פעם נוספת: החלף את DarkGrey ל-Monochrome. רענן את הדף בדפדפן (או פתח אותו שוב) ותראה את הדף עם ערכת הנושא החדשה ותמונות, כפי שמוצג בתרשים 6-19.



תרשים 6-19

כיצד זה פועל

מנקודת המבט של ערכת נושא, לא השתנה הרבה בתרגיל האחרון. כפי שראית קודם, ערכת הנושא מתווספת לראש הדף. עם זאת, הפעם הצביע גיליון הסגנון בערכה לתמונות שנמצאות בתיקייה Themes. הדפדפן קורא את קובץ CSS, עוקב אחר הקישור לתמונות, טוען אותן ומציג אותן במיקום הנכון, כפי שמוכתב על ידי בוררר CSS השונים בקובץ הקוד.

קבצי CSS שהוספת לשתי ערכות הנושא מכילים הערות רבות. אם תרצה לדעת מה עושים הקבצים, גש לתיקיות הערכות ובדוק אותם.

ערכות נושא הינן כלי עבודה שימושי מאוד, מכיוון שהן מאפשרות למפתח הדף לשנות במהירות את המראה, ואפילו את העיצוב והעריכה, של האתר. הן עשויות להיות שימושיות עוד יותר אם תאפשר למשתמשים שלך (אולי לא לכולם...) להחליפן בזמן ביצוע. בדרך זו יכולים המשתמשים להתאים את האתר לטעמם. הסעיף הבא מראה איך עושים זאת.

החלפת ערכות נושא באופן דינמי

להחלפת ערכות נושא בזמן ביצוע יש כמה תועלות. לדוגמה, המשתמש יהיה מאוד מרוצה אם הוא יוכל לבחור ערכת נושא עם צבעים, עיצוב ועריכה בהתאם לטעמו בזמן ביצוע. אתה יכול גם לפרוס ערכות נושא שיעזרו למשתמשים בעלי לקות ראייה. למשל, יצירת ערכת נושא בעלת סכמה צבעונית עם ניגודיות גבוהה וגופן גדול, תקל מאוד על הצפייה באתר. ערכות הנושא באתר

האינטרנט Planet Wrox משנות רק אלמנטים של מסך, כמו צבעים, עיצוב ועריכה, אבל קל ליצור עותק של אחת מערכות נושא אלו ולשנות בו את גודל הגופן וסכמת הצבעים.

בגלל הדרך שבה מוחלות ערכות נושא על דף בזמן ביצוע, עליך להגדיר את הערכה מוקדם במחזור החיים של הדף, ליתר דיוק, באירוע PreInit. המקום האידיאלי לעשות זאת הוא דף הבסיס של האתר, מכיוון שכל דף באתר יורש ממחלקה זו.

כדי לאפשר למשתמשים לשנות את ערכת הנושא, תוכל להוסיף תפריט נפתח, וכשמשתמש משנה ובוחר מרשימת ערכות הנושא את הערכה הרצויה, התפריט מבצע פעולת Postback לשרת עם האפשרות הנבחרת. השרת מקבל את ערכת הנושא הנבחרת מהרשימה, מחיל אותה על הדף, ומאחסן את הבחירה בעוגייה, כדי שניתן יהיה לאחזר אותה בביקורים חוזרים באתר.

עוגייה (cookie) היא קטע טקסט קטן שאפשר לאחסן אותו במחשב המשתמש. הנתונים שאתה מאחסן בעוגייה נשלחים רק לשרת שהגדיר אותה, ולכן אתרים אחרים אינם יכולים לקרוא את העוגייה. עם זאת, מכיוון שעוגיות מאוחסנות במחשב המשתמש כטקסט רגיל, לעולם אל תאחסן בעוגייה נתונים רגישים כגון סיסמה. לעומת זאת, אחסון נתונים לא מזיקים, כמו ערכת הנושא המועדפת, הוא דוגמה לשימוש מצוין בעוגיות.

השאלה אם שימוש בעוגיות עלול לפגוע בפרטיות עוררה ויכוח סוער. ככלל, עוגיות הן בטוחות, מכיוון שהן מאחסנות נתונים שהשרת שהגדיר אותם מחזיק בהם ממילא. העוגיות אינן יכולות לשמש לגניבת מידע רגיש מהמחשב שלך אם לא מסרת בעצמך את המידע הזה לשרת. במרבית התסריטים העוגיות משפרות את חווית הדפדוף של המשתמש, בכך שהן זוכרות כמיות קטנות של נתונים וחוסכות את הצורך לתחקר את המשתמש בכל פעם שהוא מבקר בדף. לרוע המזל, יש תאגידים גדולים, כמו סוכנויות פרסום, שמשתמשים בעוגייה ייחודית כדי לעקוב אחר פעולותיך באינטרנט, במטרה לקבל מידע על האתרים שאתה מבקר בהם ולפנות אליך אחר כך במבצעי פרסום. כדי להבטיח שמבקרים באתר שלך מבינים איזה מידע אתה שומר אודותם, מומלץ להוסיף לאתר הצהרת פרטיות שמתארת את הכוונה והשימוש בעוגיות, ואת הנתונים האישיים שאתה עשוי לשמור.

בשני התרגילים הבאים תראה כיצד ליישם את הפונקציונליות להחלפה דינמית של ערכות נושא. התרגיל הראשון ינחה אותך דרך שינוי בדף הראשי שיאפשר למשתמש לבחור בערכת נושא. בתרגיל זה תאחזר הערכה שהמשתמש בחר ותאחסן בעוגייה. בתרגיל השני תלמד איך להחיל את ערכת נושא בזמן ביצוע על כל דף שיוורש מ-BasePage.

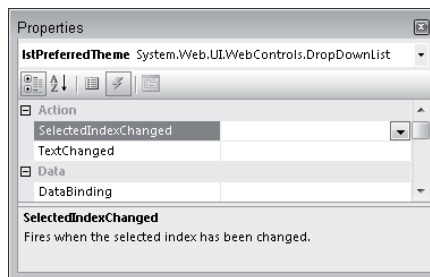
נסה זאת תן למשתמש לבחור ערכת נושא

בתרגיל זה תוסיף פקד DropDownList לדף הראשי. פקד זה מכיל את ערכות הנושא הזמינות לבחירת המשתמש. בחירת המשתמש מאוחסנת בעוגייה, ולכן היא זמינה בכל עת. צעד הסיום הוא לבחור מראש את הערכה הנכונה ברשימה הנפתחת, כשהמשתמש חוזר ומבקר בדף.

1. פתח את הדף הראשי ב- Markup View ואתר בו <div> בשם Sidebar. הסר את הטקסט הסטטי Sidebar Goes Here והחלף אותו בפקד DropDownList על ידי גרירה שלו

מארגז הכלים אל בין שתי תגיות div. שנה את ID של הפקד מ-DropDownList1 ל-lstPreferredTheme. הקלד מעט טקסט (למשל, Select a Theme) בחזית הרשימה הנפתחת כדי להסביר מה תפקידה.

2. עבור ל- Design View, פתח את הלוח Smart Task של הפקד ובחר Enable AutoPostBack.
3. באותו לוח Smart Task, לחץ על הקישור Edit Items והכנס שני פריטים: אחד עם הטקסט Monochrome והשני עם הטקסט DarkGrey.
4. לחץ לחיצה כפולה על הרשימה הנפתחת כדי לקבוע מטפל אירוע עבור האירוע SelectedIndexChanged בצד השרת. לחילופין, הקש F4, הדבר יביא אותך לטבלת המאפיינים של הפקד DropDownList. לחץ כעת על הכפתור עם סמל הברק כדי לעבור לרשימה Events, ואז לחץ לחיצה כפולה על SelectedIndexChanged. תרשים 6-20 מראה את טבלת המאפיינים במצב Events.



תרשים 6-20

הקוד במטפל SelectedIndexChanged מתבצע ברגע שהמשתמש בוחר בחירה חדשה מהרשימה הנפתחת. הוסף את הקוד הבא לקוד המטפל. תפקידו לאחזר את ערכת הנושא שנבחרה מהרשימה, ולאחסן אותה בעוגייה:

VB.NET

```
Protected Sub DropDownList1_SelectedIndexChanged(ByVal sender As Object, _  
    ByVal e As System.EventArgs) Handles lstPreferredTheme.SelectedIndexChanged  
    Dim preferredTheme As HttpCookie = New HttpCookie("PreferredTheme")  
  
    preferredTheme.Expires = DateTime.Now.AddMonths(3)  
    preferredTheme.Value = lstPreferredTheme.SelectedValue  
    Response.Cookies.Add(preferredTheme)  
    Response.Redirect(Request.Url.ToString())  
End Sub
```

C#

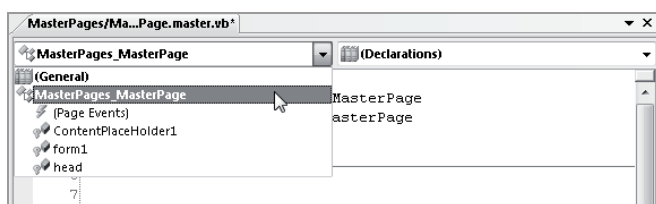
```
protected void lstPreferredTheme_SelectedIndexChanged(object sender, EventArgs e)  
{  
    HttpCookie preferredTheme = new HttpCookie("PreferredTheme");  
    preferredTheme.Expires = DateTime.Now.AddMonths(3);  
}
```

```

preferredTheme.Value = lstPreferredTheme.SelectedValue;
Response.Cookies.Add(preferredTheme);
Response.Redirect(Request.Url.ToString());
}

```

5. כשאתה נמצא עדיין בקוד ברקע של הדף הראשי, תצטרך להוסיף קוד שבוחר מראש את הפריט הנכון מהרשימה כשהדף נטען. המקום הטוב ביותר לשתילת קוד זה הוא באירוע Load של המחלקה Page. אם אתה משתמש ב-C#, המטפל Page_Load צריך כבר להיות שם. ב-VB אפשר להוסיף את המטפל באחת משתי דרכים: (1) ב-Design View, לחץ לחיצה כפולה בנקודה כלשהי בדף (זה פועל גם ב-C#); (2) מהרשימה הנפתחת השמאלית, ממש מעל חלון המסמך בקוד ברקע, בחר (Page Events), ראה תרשים 6-21, ואז בחר Load מהרשימה הנפתחת השנייה. זו דרך נאה להוסיף מטפלים גם לפקדים אחרים כמו DropDownList ו-Button.



תרשים 6-21

בתוך בלוק המטפל שהוסיפה עבורך VWD, הוסף את הקוד הבא:

VB.NET

```

Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Me.Load

```

```

    If Not Page.IsPostBack Then
        Dim selectedTheme As String = Page.Theme
        Dim preferredTheme As HttpCookie = Request.Cookies.Get("PreferredTheme")
        If preferredTheme IsNot Nothing Then
            selectedTheme = preferredTheme.Value
        End If
        If lstPreferredTheme.Items.FindByValue(selectedTheme) IsNot Nothing Then
            lstPreferredTheme.Items.FindByValue(selectedTheme).Selected = True
        End If
    End If
End Sub

```

C#

```

protected void Page_Load(object sender, EventArgs e)

```

```

{
    if (!Page.IsPostBack)
    {
        string selectedTheme = Page.Theme;

```



```

HttpCookie preferredTheme = Request.Cookies.Get("PreferredTheme");
if (preferredTheme != null)
{
    selectedTheme = preferredTheme.Value;
}
if (lstPreferredTheme.Items.FindByValue(selectedTheme) != null)
{
    lstPreferredTheme.Items.FindByValue(selectedTheme).Selected = true;
}
}
}

```

6. שמור את כל השינויים ואז בקש את Default.aspx בדפדפן. פתח את הרשימה הנפתחת בסרגל הצד ובחר באפשרות השנייה מהרשימה והדף ייטען מחדש. שים לב שאתה ממשיך לראות את אותה ערכת נושא למרות שבחרת מהרשימה את הערכה השנייה, זה קורה מפני שלא כתבת עדיין קוד כלשהו שמחיל את הערכה שנבחרה. תלמד איך לעשות זאת בתרגיל מאוחר יותר.

אם תקבל שגיאה, ודא שאין שגיאות הקלדה בקוד. אם אינך מבחין בהבהוב ונראה ששום דבר לא קורה (לדוגמה, הדף אינו שולח את עצמו חזרה לשרת), בדוק אם הגדרת את המאפיין AutoPostBack בפקד DropDownList כ-True.

כיצד זה פועל

עד כה ביצעת שלושה שינויים חשובים בדף הראשי. תחילה הוספת את הרשימה הנפתחת והגדרת את התכונה AutoPostBack כ-True. פעולה זו גרמה לדף לשלוח את עצמו חזרה לשרת לאחר בחירת פריט חדש מהרשימה. ברגע שזה קורה נכנס לפעולה הקוד במטפל SelectedIndexChanged. קוד זה יוצר עוגייה שיכולה להיות מאוחסנת בדפדפן המשתמש. כדי שהעוגייה תשרוד בין ה-sessions של הדפדפן, יש צורך לקבוע את המאפיין Expires. בדוגמת הקוד נקבע שתוקפה של העוגייה יפוג בעוד שלושה חודשים. כלומר, הדפדפן ימחק את העוגייה באופן אוטומטי בתום התקופה. עם זאת, בכל פעם שהמשתמש בוחר בערכת נושא חדשה, התקופה מוארכת ונקבעת לשלושה חודשים נוספים.

VB.NET

```

Dim preferredTheme As HttpCookie = New HttpCookie("PreferredTheme")
preferredTheme.Expires = DateTime.Now.AddMonths(3)

```

C#

```

HttpCookie preferredTheme = new HttpCookie("PreferredTheme");
preferredTheme.Expires = DateTime.Now.AddMonths(3);

```

לאחר יצירת העוגייה, מעבירים לעוגייה את שם ערכת הנושא, המאפיין SelectedValue של הרשימה (מכיל את שם ערכת הנושא) מועבר למאפיין Value של עוגייה. לאחר מכן העוגייה מתווספת אל האוסף Cookies תוך שימוש ב- Response.Cookies.Add:

VB.NET

```
preferredTheme.Value = 1stPreferredTheme.SelectedValue  
Response.Cookies.Add(preferredTheme)
```

C#

```
preferredTheme.Value = 1stPreferredTheme.SelectedValue;  
Response.Cookies.Add(preferredTheme);
```

שים לב שהעוגייה התווספה לאוסף Cookies של האובייקט *Response* שקשור בתגובה למשתמש. מאוחר יותר תראה כיצד לקרוא עוגייה זו שוב מהאוסף Cookies של האובייקט *Request*, שקשור לבקשה לקבלת הדף על ידי המשתמש.

צעד הסיום הוא הפנייה אל אותו דף שוב.

VB.NET

```
Response.Redirect(Request.Url.ToString())
```

C#

```
Response.Redirect(Request.Url.ToString());
```

צעד זה נחוץ למטרת רענון הדף, כי אחרת הערכה החדשה לא תחול, או תיושם, באופן מיידי על הדף. מכיוון שיש להגדיר את ערכת הנושא מוקדם במחזור החיים של דף, אין עוד אפשרות להגדירה עבור הבקשה הנוכחית. על ידי פנייה מחודשת אל הדף, נוצרת בקשה חדשה אשר יכולה להחיל בהצלחה את ערכת הנושא שנבחרה. התרגיל הבא מראה כיצד לעשות זאת.

השינוי האחרון שעשיית בדף הראשי היה שינוי המטפל *Page_Load*. בתוך שיטה זו מוכרז משתנה *String*, שמחזיק את שם הערכה הפעילה הנוכחית על ידי התבוננות במאפיין *Theme* של המחלקה *Page*, זו תהיה ערכת ברירת המחדל אם למשתמש אין עוגייה שמכילה את ערכת הנושא המועדפת עליו. בצעד הבא הקוד בודק אם קיימת עוגייה בשם *PreferredTheme*. אם היא קיימת, המחרוזת *selectedTheme* תקבל את הערך שלה. בסופו של דבר, משתנה *string* זה משמש למציאת הפריט ברשימה הנפתחת, ולבחירתו.

בדרך זו, הרשימה הנפתחת מציגה תמיד את ערכת הנושא של האתר המוגדרת כרגע, או את הפריט שנבחר על ידי המשתמש באופן ידני, אפילו אם המשתמש יחזור לאתר בשבוע הבא. שים לב לשימוש בשיטה *FindByValue* על האוסף *Items* בפקד *DropDownList*. שיטה זו מחזירה את הפריט, אם הוא נמצא, או *Nothing* (או *null* ב-C#) כשהפריט איננו שם. הדבר מבטיח שבמצב שבו העוגייה מכילה ערכת נושא שאינה קיימת עוד, הקוד לא ינסה לבחור פריט שאינו קיים ברשימה.

לאחר שלמדנו על היכולת לאפשר למשתמש לבחור ערכת נושא, הצעד המתבקש הוא להחיל את הערכה שנבחרה.

כפי שלמדת קודם, יש לקבוע את ערכת הנושא באירוע PreInit שמתרחש מוקדם במחזור החיים של דף. בתוך אירוע זה אפשר לראות אם קיימת העוגייה עם הערכה שנבחרה. אם כן, אפשר להשתמש בערך שלה כדי להגדיר את ערכת הנושא הנכונה.

נסה זאת החלת ערכת נושא שנבחר על ידי המשתמש

בתרגיל זה תשנה את דף הבסיס, ותוסיף קוד לאירוע PreInit כדי לקבוע את ערכת הנושא שהמשתמש בחר.

1. פתח את הקובץ BasePage מהתיקייה App_Code והוסף את הקוד הבא שקובע את הערכה שנבחרה במהלך האירוע PreInit. אפשר להוסיף קוד זה לפני, או אחרי, השיטה שבודקת את כותרת הדף.

VB.NET

```
Private Sub Page_PreInit(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Me.PreInit
    Dim preferredTheme As HttpCookie = Request.Cookies.Get("PreferredTheme")
    If preferredTheme IsNot Nothing Then
        Page.Theme = preferredTheme.Value
    End If
End Sub
```

C#

```
private void Page_PreInit(object sender, EventArgs e)
{
    HttpCookie preferredTheme = Request.Cookies.Get("PreferredTheme");
    if (preferredTheme != null)
    {
        Page.Theme = preferredTheme.Value;
    }
}
```

2. אם אתה מקודד ב-C#, עליך להגדיר מטפל אירוע בפונקציית הבנייה של המחלקה עבור האירוע PreInit. הדבר מתבצע על ידי הוספת שורת קוד בפונקציית הבנייה של המחלקה, בדיוק כפי שעשית בתרגיל קודם עבור מטפל האירוע PreRender. שורה זו אומרת ל- ASP.NET runtime איזו שיטה תטפל באירוע PreInit:

```
public BasePage()
{
    this.PreRender += new EventHandler(Page_PreRender);
    this.PreInit += new EventHandler(Page_PreInit);
}
```

3. שמור את השינויים לכל המסמכים הפתוחים, ואז בקש את Default.aspx בדפדפן. הדף ייטען עם ערכת הנושא שבחרת בפעם הקודמת ומופיעה בראש הרשימה הנפתחת.
4. בחר מהרשימה פריט חדש. הדף ייטען שוב, ויציג את ערכת הנושא החדשה.

כיצד זה פועל

לאחר שבוצעה הפעולה הכרוכה בקבלת ערכת הנושא החביבה על המשתמש ואחסונה בעוגייה, החלת הערכה קלה ופשוטה מאוד. בתחילה קוד האירוע PreInit מאמת את קיום העוגייה בשם PreferredTheme. הוא עושה זאת על ידי השוואת הערך המוחזר של השיטה Get ל-Nothing (C#-ב null).

VB.NET

```
Dim preferredTheme As HttpCookie = Request.Cookies.Get("PreferredTheme")
If preferredTheme IsNot Nothing Then
```

C#

```
HttpCookie preferredTheme = Request.Cookies.Get("PreferredTheme");
if (preferredTheme != null)
```

Request.Cookies משמשת כאן לקריאה מאוסף העוגיות ששלח הדפדפן ביחד עם הבקשה. אם העוגייה קיימת, משתמשים במאפיין Value כדי לקבל את שם ערכת הנושא:

VB.NET

```
Page.Theme = preferredTheme.Value
```

C#

```
Page.Theme = preferredTheme.Value;
```

הגדרה זו תקנה לך את המראה והתחושה המוגדרים בערכה.

היכולת להגדיר את ערכת הנושא בתכנות, מציעה למשתמשים דרך מהירה וקלה לשנות את הדף בהתאם לטעמם. ערכת הנושא משפיעה על הצבעים והפריסה של כל דף באתר האינטרנט כולו. בשילוב עם דפים ראשיים הדבר נותן דרך גמישה להשפיע על המראה והתחושה של דף שלם. מצד שני, גם היכולת לשנות פקדים מסוימים בדף עשויה להיות שימושית. לדוגמה, ייתכן שיהיה צורך לקבוע לכל הלחצנים באתר מראה אחיד. זה המקום שבו ASP.NET skins נכנסים למשחק.

Skins

Skins הם קבצי טקסט פשוטים שמכילים סימני עריכה עבור פקדים. אפשר לדמות **skin** לקליפה או למעטפת, שמעניקות מראה חיצוני לעצם כלשהו. הם מוצבים בתיקית ערכת נושא תחת App_Themes, ומהווים חלק אינטגרלי ממאפייני ערכות הנושא של ASP.NET. קובץ skin (בעל סיומת skin) מכיל את צד השרת של האלמנטים של התצוגה של פקד. הגדרות אלו מוחלות על כל הפקדים שה-skin חל עליהם. כדי לראות כיצד זה פועל, חשוב על הדוגמה הבאה שמגדירה את המעטפת, או המראה החיצוני, של פקד Button.

```
<asp:Button BackColor="#cccccc" ForeColor="#308462" runat="server" />
```

עם הגדרה זו של skin, כל הלחצנים באתר יהיו בעלי BackColor שערכו #cccccc, ו-ForeColor שערכו #308462. כל מה שנדרש ממך לעשות זה ליצור קובץ skin תחת תיקיית ערכות הנושא, ולהוסיף לו את סימני העריכה האלה. מאותו רגע והלאה, כל הלחצנים יותאמו לקביעה זו באופן אוטומטי. בדיוק כמו במקרה הקודם של הגדרה ישירה של מאפיינים בפקדים, עוברים גם מאפיינים כמו BackColor ו-ForeColor המרה לקוד HTML ולקוד CSS של צד-לקוח.

סימני עריכה אלה אמנם דומים לסימני העריכה של לחצן, אבל יש כמה הבדלים. בראש ובראשונה, פקד בקובץ skin אינו יכול להיות בעל התכונה ID. הקוד ID משמש לזיהוי ייחודי של פקד בדף, ומכיוון שה-skin מוחל על כל הפקדים, אין טעם לקבוע עבורו ID כלשהו. הבדל נוסף הוא במספר התכונות שאפשר להגדיר בסימני העריכה. לא כל המאפיינים של פקד ניתנים להגדרה על ידי skin. לדוגמה, אינך יכול להגדיר את המאפיין Enabled של הפקד Button באמצעות skin. בתיעוד MSDN של מיקרוסופט תמצא תחת כל מאפיין אם הוא יכול להיות 'skinned' או לא. דרך אחרת כדי לברר אם מאפיין מסוים ניתן להגדרה על ידי skin היא לנסות אותו. הגדר את המאפיין ב-skin, ואם אינך רשאי לעשות זאת תקבל הודעת שגיאה בזמן ביצוע.

ככלל, מאפיינים שמשפיעים על המראה (BackColor, ForeColor, BorderColor וכו') יכולים להיות מוגדרים על ידי skin, ואילו מאפיינים שמשפיעים על ההתנהגות (Enabled, EnableViewState ועוד) לא ניתנים להגדרה ב-skin.

כשיוצרים קובץ skin חדש, תוך שימוש בתיבת הדו-שיח Add New Item, מקבלים צרור של טקסט עטוף בבלוק הערות של צד-שרת. אפשר להסיר הערות אלו בבטחה, מכיוון שהן רק נותנות דוגמה קצרה כיצד פועל skin. אפשר להגדיר פקדים רבים בקובץ skin יחיד. עם זאת, מנקודת המבט של תחזוקה, ההמלצה היא ליצור לכל סוג פקד קובץ skin משלו. כמו לדוגמה Button.skin עבור לחצנים, Label.skin עבור תוויות וכד'.

לעתים קרובות עדיף להשתמש במאפיין CssClass כדי להצביע על מחלקת CSS באחד מקבצי CSS שלך, במקום להחיל אלמנטים של עיצוב ישיר של מאפייני הפקד ב-skin, ודרכם עיצוב של סימני העריכה הסופיים בדף. בדרך זו אפילו קל יותר לבצע שינויים באתר כולו ולהימנע מלנפח את קוד HTML הסופי. על פי הדוגמה הקודמת, קובץ עם הגדרת skin שלהלן, ומחלקה בקובץ CSS של ערכת הנושא, יתנו את אותה התוצאה:

```
<asp:Button CssClass="MyButton" runat="server" />

.MyButton
{
    color: #308462;
    background-color: #cccccc;
}
```

יצירת קובץ Skin

קבצי skin חייבים להיווצר באופן ישיר בתיקיית ערכת הנושא. אינך יכול לאחסן אותם בתת-תיקייה כפי שעשית עם תמונות הערכות. בתרגיל הבא תראה איך ליצור קובץ skin פשוט כדי לשנות את המראה והתחושה של כל פקדי הלחצן באתר האינטרנט שלך. פרקים מאוחרים יותר בספר יתבססו על הידע שתלמד בתרגיל זה, בכך שתגדיר skins מורכבים יותר עבור פקדים כמו GridView.

כשתתחיל להקליד בקובץ skin, תוכל לשים לב ש-IntelliSense המוכר אינו מתחיל לפעול. הדבר מקשה על הגדרת הפקדים ותכונותיהם. עם זאת, יש דרך פשוטה לעקוף את הבעיה:

1. פתח את תיבת הדו-שיח Options ב-VWD על ידי בחירת Tools ← Options.
2. ודא שנבחרה Show All Settings שבתחתית המסך.
3. הרחב את הקטגוריה Text Editor ולחץ File Extension.
4. בתיבה Extension הקלד skin, ומהרשימה הנפתחת Editor בחר User Control Editor.
5. לחץ על הלחצן Add ואז לחץ OK כדי לשחרר את תיבת הדו-שיח Options.

מעתה ואילך תקבל את IntelliSense גם בקבצי skin, כפי שתראה בתרגיל הבא. כשהגדרה לעיל בתוקף, אתה עשוי לקבל ב-Error List הודעת שגיאה אודות ספקי הבנייה (build providers). אפשר להתעלם משגיאה זו בבטחה, מכיוון ש-skins יפעלו היטב בזמן ביצוע עם ההגדרה שנעשתה.

נסה זאת יצירת Skin עבור הפקד Button

כדי להשתמש ב-skins ביעילות, עליך להשתמש בכל האפשרות בתכונה CssClass במקום להחיל תכונות כלולות שכולן מופיעות בקוד HTML הסופי של הדף, ובכך מגדילות אותו ואת זמן טעינתו. עם זאת, תרגיל זה מראה כיצד ליישם את שתי הדרכים, כדי לענות על מקרה שבו מתעורר דרישה מיוחדת להוסיף מאפייני תצוגה כלולים.

1. הוסף קובץ skin חדש לתיקיית ערכת הנושא Monochrome, וקרא לו Button.skin. את הקובץ מוסיפים על ידי לחיצה ימנית על התיקייה Monochrome ובחירת Add New Item. בתיבת הדו-שיח שנפתחת בחר Skin File, והקלד Button עבור שם הקובץ.
2. מחק את כל תוכן הקובץ והוסף את הקוד הבא:

```
<asp:Button CssClass="MyButton" BackColor="#cccccc" runat="server" />
```

שים לב שסימני עריכה אלה משתמשים בשילוב של מאפיין כלול (BackColor) ומאפיין CssClass להצבעה על בורר בקובץ CSS.

3. פתח את הקובץ Monochrome.css מתיקיית ערכת הנושא, והוסף את בורר CSS הבא לסוף הקובץ:

```
.MyButton
{
    color: #308462;
}
```

4. צור טופס אינטרנט חדש בתיקייה Demos וקרא לו SkinsDemo.aspx. ודא שביססת אותו על התבנית המיוצאת שיצרת קודם לכן. הגדר את Title של הדף כ- Skins Demo והוסף פקד Button על ידי גרירתו מארגז הכלים אל השטח cpMainContent שבדף. עליך לסיים פעולה זו עם הקוד הבא:

```
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" Runat="Server">
  <asp:Button ID="Button1" runat="server" Text="Button" />
</asp:Content>
```

5. שמור את כל השינויים ובקש את SkinsDemo.aspx בדפדפן. אם נחוץ, עבור לערכת הנושא Monochrome. הלחצן שהוספת בצעד האחרון יהיה עכשיו בעל רקע אפור עם טקסט ירוק בהיר עליו.

כיצד זה פועל

כדי להבין כיצד זה פועל, כדאי להתבונן בקוד HTML של הדף כפי שהוא מוצג בדפדפן. הפקד <asp:Button> עבר המרה לקוד HTML הבא.

```
<input type="submit" name="ctl00$cpMainContent$Button1"
  value="Button" id="ctl00_cpMainContent_Button1" class="MyButton"
  style="background-color:#CCCCCC;" />
```

שניים ממאפייני ה-skin – BackColor ו-CssClass – נוספו ל-HTML. הראשון סיים כמאפיין class של הלחצן, והשני עבר המרה למאפיין style. המחלקה MyButton בקובץ CSS קובעת ללחצן טקסט ירוק, בעוד שהסגנון הכלול קובע את צבע הרקע של הלחצן.

כפי שניתן לראות, skins קלים ביותר לשימוש ומאפשרים לשנות באופן קיצוני את המראה של פקדים מסוימים באתר. אבל מה אם אינך רוצה שינוי בו-זמני של כל הלחצנים לירוק ואפור? מה אם יש צורך בלחצן אחד מיוחד בעל רקע אדום? אפשר לעשות זאת עם named skins.

Named Skins

named skins זהים ל-skins רגילים עם הבדל אחד: מוגדר עבורם SkinID שמאפשר להתייחס אליהם באופן מפורש. פקדים בדפי ASPX יכולים להשתמש ב-SkinID זה כדי להחיל את ה-skin הרצוי על הפקד. התרגיל הבא מראה כיצד זה פועל.

נסה זאת יצירת Named Skin עבור הפקד Button

הדרך הקלה ביותר ליצור named skin היא על ידי העתקת הקוד של skin קיים והוספת המאפיין SkinID. שים לב שאם העתקת והדבקת הגדרה של skin, התוכנה Visual Web Developer תוסיף באופן אוטומטי את המאפיין ID (בהנחה שקישרת קבצי skin אל User Control Editor כפי שתואר קודם לכן). הגדרת מאפיין ID זה אסורה, ועליך להסיר אותה.

1. פתח שוב את Button.skin, העתק את כל הקוד, והדבק אותו מתחת לסימני העריכה הקיימים.

2. אם VWD הוסיפה את המאפיין ID, הסר אותו ואת ערכו.

3. הסר את המאפיין CssClass ואת ערכו, שנה את המאפיין BackColor של הלחצן לצבע Red, ואז הגדר את המאפיין ForeColor ל-Black.

4. קבע את הערך RedButton למאפיין SkinID. אתה אמור לקבל כעת את הקוד הבא:

```
<asp:Button CssClass="MyButton" BackColor="#cccccc" runat="server" />
```

```
<asp:Button SkinID="RedButton" BackColor="Red" ForeColor="Black" runat="server" />
```

5. שמור וסגור את קובץ skin.

6. פתח את SkinsDemo.aspx והוסף לחצן שני. הגדר את SkinID של לחצן זה כ-RedButton. הקוד עבור שני הלחצנים אמור להיראות כעת כך:

```
<asp:Button ID="Button1" runat="server" Text="Button" />
```

```
<asp:Button ID="Button2" runat="server" Text="Button" SkinID="RedButton" />
```

שים לב ש-IntelliSense פועל, ומציג את הערכים האפשריים עבור ה-SkinID הנכון.

7. פתח את SkinsDemo.aspx בדפדפן. אתה אמור לראות כעת שני לחצנים עם צבעים שונים.

כיצד זה פועל

named skins פועלים בדומה מאוד ל-skins הרגילים. אולם, עם named skin פקד יכול להצביע על skin מסוים באחד מקבצי ה-skin. הלחצן הראשון בדף Default.aspx מקבל את הגדרותיו מ-skin ברירת המחדל, חסר השם, והלחצן השני מקבל כעת את הגדרותיו מה-skin בעל מאפיין SkinID שהוגדר כ-RedButton.

named skins מעמידים לרשותך פתרון גמיש מאוד. עם skins רגילים אפשר לשנות במהירות את המראה של כל הפקדים באתר. אפשר להשתמש ב-named skin כדי לדרוס התנהגות זו עבור כמה פקדים כאשר דרוש שיוצגו באופן שונה.

הערה אחרונה על Skins

אם מסיבה כלשהי אין ברצונך להחיל skin על פקד מסוים, ביכולתך לבטל את ה-skin ברמת הפקד על ידי הגדרת המאפיין EnableTheming של הפקד בדרך זו:

```
<asp:Button ID="Button1" runat="server" EnableTheming="False" Text="Button" />
```

כש-EnableTheming מוגדר כ-False, ה-skin אינו חל על הפקד. למרות זאת, הגדרות CSS מקובץ CSS של ערכת הנושא עדיין מוחלות עליו.

עצות מעשיות ליצירת דפים עקביים

הרשימה הבאה מספקת כמה עצות מעשיות ליצירת דפים עקביים:

- ⊙ כשאתה יוצר אתר אינטרנט חדש, התחל תמיד בהוספת דף ראשי שעליו תבסס את כל הדפים האחרים. דף ראשי יעזור לך להבטיח מראה עקבי בכל דפי האתר גם אם אתה חושב שבאתר יהיו דפים ספורים בלבד. הוספת דף ראשי לאתר בשלב מאוחר יותר עלולה לגרור שינויים רבים בדפים הקיימים באופן ידני. מכאן שהוספת דף ראשי מייד בהתחלה חוסכת עבודה רבה בשלבים מאוחרים יותר.
- ⊙ ברגע שתוסיף מידע עיצוב לפקדים מורכבים כמו TreeView ו-Menu (נכון בהם בפרק הבא), או לפקדי נתונים כמו GridView (נכון בהם בפרק 12), שקול יצירת skin עבורם. העובדה שניתן לשלוט בעיצוב ובעריכה של כל הפקדים הדומים ממקום אחד מקלה מאוד על עדכון האתר. אם ברצונך לדרוס את העיצוב והעריכה של כמה פקדים, תוכל תמיד להשתמש ב- named skins עם SkinID, או לבטל את ה-skin לחלוטין על ידי הגדרת EnableTheming כ-False.
- ⊙ כשתיצור skins או תגדיר מאפייני סגנון באופן ישיר בפקד, שקול שימוש ב-CssClass כתחליף, ואז העבר את כל המאפיינים הנוגעים לסגנון אל קוד CSS של האתר או לערכת נושא. הדבר מקטין את גודל הדף ומקל על הכנסת שינויים בעיצוב ובעריכה לאחר מכן.
- ⊙ התכונה Export Template של Visual Web Developer תורמת לחיסכון משמעותי בזמן. תוכל להשתמש בה לא רק ליצירת תבנית עבור דף ASPX והקוד ברקע שלו, אלא גם עבור קבצים אחרים כמו מחלקות וקבצי CSS. התכונה מאפשרת יצירה מהירה של קבצים חדשים, וחוסכת הקלדה חוזרת ונשנית של אותו קוד עצמו.

סיכום

ASP.NET 3.5 מציעה כמה כלים מצוינים שתפקידם לסייע ביצירת אתר אינטרנט בעל מראה עקבי. מראה ותחושה עקביים של כל הדפים באתר חשובים לאטרקטיביות שלו בעיני הגולשים בו ולמראה המקצועי שלו. דבר זה מסייע למבקרים באתר למצוא את המידע הרצוי להם, ומגדיל את הסיכוי לביקור חוזר שלהם באתר.

הדרך הטובה ביותר להתחיל ליצור אתרים עקביים היא על ידי שימוש בדף ראשי. עם דף ראשי אפשר להגדיר את המראה הכללי והתחושה של דפים, להוסיף ישירות פריטים שהם סטטיים בדרך כלל, כמו כותרת עליונה ותחתונה, וליצור בעזרת הפקדים שומרי המקום מסוג ContentPlaceHolder, את הקצאת המקום שדפי התוכן יוכלו למלא.

דף תוכן מחובר לדף הראשי על ידי המאפיין MasterPageFile בהנחיה @ Page. הדבר היחיד שדף תוכן צריך להגדיר הוא אזור <asp:Content> עבור כל <asp:ContentPlaceHolder> בדף הראשי שברצונו לדרוס. אין צורך לכלול בדף התוכן תוכן שחוזר על עצמו, כמו כותרות

עליונות, כותרות תחתונות, או תפריטים. הדבר משחרר מהצורך לבצע שינויים חוזרים בכל דפי התוכן באתר כשרוצים לשנות את העיצוב והעריכה של האתר.

דפים ראשיים ודפי תוכן משמשים להגדרת העיצוב והעריכה הכלליים של דף במונחים של אזורי התוכן השונים שיש בדפים. ערכות נושא, לעומת זאת, משמשות לשינוי המראה של אלמנטים אלה. מכיוון שערכות נושא יכולות להכיל קבצי CSS, תמונות ו-skins, ביכולתן לשנות צבעים, גופנים, מיקום ותמונות על ידי החלת ערכת נושא. מספר ערכות הנושא אינו מוגבל, ואפשר לעבור ביניהן על ידי הגדרת יחידה בקובץ web.config או בדף ASPX מסוים. בנוסף, אפשר לעבור בין ערכות הנושא על ידי תכנות, יכולת שנותנת למשתמשים אפשרות לבחור ערכת נושא בהתאם להעדפותיהם. ערכות נושא מאפשרות גם את שיפור השימושיות באתר על ידי היכולת של משתמשים בעלי לקות ראייה לבחור לעצמם ערכה עם ניגודיות גבוהה וגופנים גדולים.

חלק חשוב נוסף בערכות נושא ב-ASP.NET 3.5 הוא ה-skins. עם skin ניתן לשלוט במראה של כל פקד ממקום אחד – קובץ skin שנמצא בתיקיית הערכה. skins מגיעים בשני סוגים: skin ברירת המחדל שחל על כל הפקדים שהוא מכון אליהם, ו-named skin המאפשר לפנות אליו בשמו. הדבר מאפשר לשנות את כל הפקדים הדומים "במכה אחת", תוך שמירת האפשרות לשנות פקד יחיד בכל דף רצוי.

אם ניקח את העקביות צעד נוסף קדימה, באפשרותך ליצור דף בסיס שממנו יירשו כל דפי ASPX שנמצאים באתר. עם דף בסיס זה ניתן לרכז את התנהגות הדפים. דף הבסיס הוא המקום האידיאלי לבדיקת דברים כמו בדיקת תקפות כותרת הדף, או להגדרת ערכת נושא שנבחרה על ידי המשתמש. כשמשמשים בתכונה Export Template של VWD, הרי שיצירת דפים שיוורשים מדף הבסיס קלה כמו הוספת דף ASPX רגיל.

אתר האינטרנט Planet Wrox נמצא כעת בשלבי צמיחה. פירוש הדבר שהמבקרים בו מתקשים יותר למצוא את הדפים הנכונים. הפרק הבא מציג כמה דרכים שבהן יכולים המשתמשים לנווט באתר באופן שלא יעמוד בפניהם כל קושי למצוא את הדף שהם מחפשים.

תרגילים

1. מה ההבדל בין `<asp:Content>` לבין פקד `<asp:ContentPlaceHolder>`? באיזה סוג דף תשתמש בכל אחד מהם?
2. כיצד תחבר פקד `<asp:Content>` בדף תוכן אל `<asp:ContentPlaceHolder>` שבדף הראשי?
3. הנח שעם הגדרת ה-skin הבאה יצרת skin שיחול על כל הלחצנים באתר:

```
<asp:Button runat="server" CssClass="MyButton" />
```

מחלקת CSS הדמיונית MyButton מגדירה את צבע הרקע של הלחצן ללבן, ואת צבע הכיתוב לשחור. כדי למשוך תשומת לב ללחצן ספציפי בדף אתה מחליט לקבוע לו צבע רקע אדום. מהן האפשרויות שיש לך לשליטה במראה של לחצן יחיד זה?
4. הסבר את ההבדל בין הגדרת המאפיין Theme לבין המאפיין StyleSheetTheme עבור דף.
5. נקוב בשם שלוש דרכים להגדרת המאפיין Theme עבור דף, והסבר את ההבדל בין האפשרויות.
6. מהי הסיבה העיקרית ליישום דף בסיס באתר אינטרנט?

פרק 7

ניווט באתר

כשהאתר שלך מכיל יותר מקומץ דפים, חשוב שיהיה לו מבנה יציב וברור לניווט בו, שיאפשר למשתמשים למצוא את דרכם באתר. יישום מערכת ניווט (navigation system) טובה יגרום לכך שכל הדפים בפרויקט, גם אלה שיש ביניהם קשר רופף, יהיו יחד אתר אינטרנט שלם ועקבי.

כשתחשוב על חלקים חשובים במערכת ניווט, הדבר הראשון שעשוי לעלות בדעתך הוא תפריט. תפריטים מגיעים בכל הסוגים והגדלים, החל מקישורי HTML פשוטים וסטטיים וכלה בתפריטים מורכבים, כמו תפריטים נפרשים (fold-out menus), שמופעלים על ידי CSS או JavaScript. אבל יש בניווט יותר מתפריטים בלבד. ASP.NET כוללת כמה פקדי ניווט שימושיים, שמאפשרים למשתמש להקים מערכת ניווט במהירות ובקלות. פקדים אלה כוללים את Menu, SiteMapPath ו-TreeView, שתלמד אודותם בפרק זה.

פרט לפקדים חזותיים כמו Menu, הניווט עוסק גם במבנה (structure) של האתר. למשתמשים קל לנווט באתר שהינו מאורגן היטב. מפת האתר, שנמצאת בשימוש פקדי הניווט, מסייעת להגדיר את המבנה הלוגי של האתר.

חלק חשוב אחר של הניווט מתרחש בצד השרת. תרחיש נפוץ מאוד הוא הפניית המשתמש מדף אחד לאחר בקוד ברקע, בהתבסס על תנאי כלשהו. לדוגמה, מנהלן מכניס תקליטור חדש או סקירת קונצרט, בחלק Management של אתר האינטרנט. כשהסקירה מסתיימת ייתכן שתרצה להראות את פירוט הסקירה למנהלן בדף חדש.

בפרק זה תלמד כיצד להשתמש באפשרויות הניווט השונות שעומדות לרשותך. תקבל הסבר מקיף בנושאים הבאים:

- ⊙ איך למצוא את דרכיך ברחבי האתר באמצעות פקדי שרת וקוד HTML פשוט.
- ⊙ כיצד לפנות לדפים ולמשאבים אחרים באתר (תמונות, למשל).
- ⊙ איך להשתמש בפקדי הניווט של ASP.NET :Menu, SiteMapPath ו-TreeView.
- ⊙ איך להפנות את המשתמש מדף אחד לאחר באמצעות קוד.

לפני שתתבונן בפקדי הניווט המובנים, עליך להבין את האפשרויות השונות שעומדות לרשותך כדי לפנות למשאבי האתר, כגון דפי ASPX ותמונות.

התמצאות ברחבי האתר

הדרך הנפוצה ביותר לאפשר למשתמש לנוע מדף אחד לאחר היא על ידי שימוש באלמנט `<a>`. אלמנט זה הוא בעל מאפיין `href` שמאפשר להגדיר את כתובת הדף או משאב אחר שרוצים להתקשר אליו. בין התגיות מציינים את תוכן הקישור, כמו למשל טקסט, תמונה או HTML אחר. קטע הקוד הבא מראה דוגמה פשוטה של האלמנט `<a>`:

```
<a href="Login.aspx">You can log in here</a>
```

כשקוד זה כלול בדף, ומשתמשים לוחצים על הטקסט "You can log in here", הם מועברים לדף `Login.aspx` שחייב להיות באותה תיקייה שבה נמצא הדף שמכיל את הקישור.

לאלמנט `<a>` יש מקביל בצד השרת שנקרא `HyperLink`, שאפשר ליצור אותו בסימני העריכה על ידי שימוש ב-`<asp:HyperLink>`. בסופו של דבר, התוצר של פקד `HyperLink` בדף הוא אלמנט `<a>`. המאפיין `NavigateUrl` של פקד זה ממופה ישירות למאפיין `href` של האלמנט `<a>`. לדוגמה, פקד צד שרת `HyperLink` כמו זה

```
<asp:HyperLink runat="server" id="lnkLogin" navigateUrl="Login.aspx">
  You can log in here</asp:HyperLink>
```

מייצר בדפדפן את קוד HTML הבא:

```
<a id="ctl00_cpMainContent_lnkLogin" href="Login.aspx">You can log in here</a>
```

פרט ל-ID הארוך שמוקצה על ידי `ASP.NET runtime`, קוד זה זהה לדוגמה הקודמת. בשני המקרים מצביע המאפיין `href` לדף `Login.aspx` כשהוא משתמש ב-URL יחסי. הנושא הבא מתאר את ההבדלים בין URL יחסי לבין URL מוחלט.

מהם URL מוחלט ו-URL יחסי

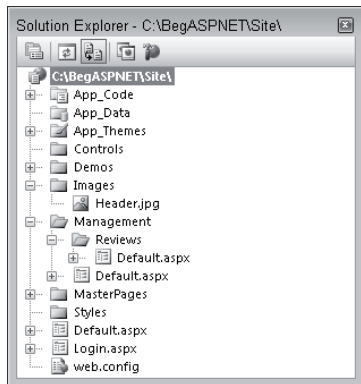
המפתח לעבודה עם קישורים באתר שלך היא הבנה טובה של הצורות השונות שיכול ללבוש URL (Uniform Resource Locator, בדרך כלל - כתובת אינטרנט) עבור משאב בתוך או מחוץ לאתר. URL משמש לזיהוי ייחודי של משאב באתר שלך או באתרי אינטרנט אחרים. משתמשים ב-URL במקומות שונים, וביניהם גם במאפיין `href` של `hyperlink` או של האלמנט `<link>` כדי להצביע על קובץ CSS, במאפיין `src` שמצביע על תמונה או על קובץ מקור של JavaScript, ובערך `url()` של מאפיין CSS. אפשר לבטא סוגי URL אלה כ-URL יחסי או כ-URL מוחלט. לכל אחד מהם יתרונות וחסרונות שעליך לדעת.

URL יחסי

בדוגמאות הקודמות ראית **URL יחסי** (relative URL) שמצביע על משאב אחר, באופן יחסי למקום שבו מוגדר ה-URL. פירושו של דבר, שהדף שמכיל את האלמנט `<a>` והדף `Login.aspx` שניהם חייבים להיות באותה תיקייה שבדיסק. כדי לפנות למשאבים בתיקיות אחרות תוכל להשתמש ב-URL אחרים כפי שתראה להלן. כל הדוגמאות מבוססות על אתר בעל המבנה שמוצג בתרשים 7-1.

כדי לפנות מדף Login.aspx הממוקם בשורש האתר אל Default.aspx שבתיקייה Management תוכל להשתמש ב-URL הבא:

```
<a href="Management/Default.aspx">Management</a>
```



תרשים 7-1

הוראה זו מצביעה על Default.aspx שבתיקייה Management שמוצבת בשורש, בסמוך לדף Login.aspx. כדי לפנות מהדף Default.aspx שבתיקייה Management לתמונה Header.jpg תוכל להשתמש ב-URL זה:

```

```

שתי הנקודות המובילות (..) "מנווטות" אל תיקייה אחת מעלה בכיוון תיקיית השורש, ואז כניסה לתיקייה Images כדי להציב על Header.jpg.

עבור היררכיית תיקיות עמוקה יותר אפשר להשתמש בכמה זוגות של נקודות מובילות, זוג לכל תיקייה מעלה בהיררכיית האתר. לדוגמה, האלמנט הבא פונה לאותה תמונה, אבל מדפים בתיקייה Reviews, שממוקמת תחת התיקייה Management:

```

```

תועלת אחת של URL יחסי היא העובדה שהוא מאפשר להעביר קבוצת קבצים לתיקייה אחרת באותה רמה, מבלי לשבור את הקישורים הפנימיים שלהם. עם זאת, יש קושי בהעברת קבצים לרמה אחרת בהיררכיית האתר. לדוגמה, אם תעביר את הדף Login.aspx לתיקייה נפרדת כמו Members, יישבר הקישור לתיקייה Management. התיקייה Management אינה תת-תיקייה של התיקייה החדשה Members, ולכן הקישור Management/Default.aspx אינו תקף עוד.

כדי להתגבר על בעיה זו אפשר להשתמש ב-URL יחסי מבוסס-שורש (root-based relative URL).

URL יחסי מבוסס-שורש

URL יחסי מבוסס-שורש מתחיל תמיד בלוחסן מוביל נטוי לימין (/), כדי לציין את שורש האתר. ניקח שוב את הקישור לתיקייה Management. הגרסה מבוססת-השורש שלו תיראה כך:

```
<a href="/Management/Default.aspx">Management</a>
```

שים לב ללוחסן המוביל הנטוי לימין שלפני שם התיקייה Management. קישור זה אינו משתמע לשתי פנים. הוא תמיד מצביע לקובץ Default.aspx שבתיקייה Management שבשורש. קישור זה לא יישבר עם העברת הדף Login.aspx לתת-תיקייה; הוא תמיד יצביע על אותו קובץ עצמו.

URL יחסי בפקד צד שרת

פקדי צד-שרת של ASP.NET מציעים אפשרות נוספת לפנות למשאבים באתר: תוכל להשתמש בתו ~ (טילדה) כדי להצביע על השורש הנוכחי של האתר. כדי להבין איזה בעיות נפתרות בדרך זו עליך להיות מודע לדרך שבה Visual Web Developer יוצרת אתרי אינטרנט חדשים ומחברת אותם לשרת web המובנה. כשאתה יוצר אתר אינטרנט חדש, VWD יוצרת כברירת מחדל, אתר בתיקיית יישום (application folder) נפרדת בשרת ה-web המובנה, שבו נוצרים כל הדפים שלך.

כך, לדוגמה, כשאתה יוצר אתר חדש ומקיש Ctrl+F5 כדי לפתוח את דף ברירת המחדל בדפדפן, אתה מקבל כתגובה כתובת דומה לזו: `http://localhost:23143/MyApplicationName`. בדרך כלל, כשהאתר "עולה לאוויר" בשרת מרוחק, אינך זקוק עוד לתיקיית יישום זו. במקום זאת מדפדפים המשתמשים אל URL כמו `http://www.PlanetWrox.com` ומצפים לראות את האתר. למרבה המזל קיימת דרך קלה לעצור את VWD מליצור את תיקיית היישום הנפרדת: אפשר להגדיר את המאפיין Virtual Path של הפרויקט ללוחסן נטוי קדימה (/), תוך שימוש בטבלת המאפיינים של האתר. תרגיל נסה זאת הבא מראה כיצד ליצור אתר אינטרנט פשוט ולהוסיף לו כמה תמונות שמשמשות בסכמות שונות של שמות URL. בשלב זה תגדיר את המאפיין Virtual Path ותראה כיצד משפיעה הגדרה זו על דרך הפנייה לקבצים.

נסה זאת חקירת התנהגות המאפיין Virtual Path

בתרגיל זה תיצור אתר אינטרנט חדש לחלוטין שמשמש לצורך התרגיל בלבד. אחר כך, אם תרצה, תוכל למחוק את האתר. לצורך התרגיל אתה זקוק לקובץ תמונה, למשל Header.jpg מהתרגיל הקודם; כל תמונה אחרת תתאים גם היא.

1. כדי ליצור אתר אינטרנט חדש לחלוטין, בחר File ← New Web Site.
2. הגדר את Location לאפשרות File System ושים לב ש-VWD מציעה נתיב שמסתיים ב-Website1. אם יצרת קודם אתרי אינטרנט מבלי שנתת להם שם מפורש, ייתכן שהנתיב יסתיים ב-Website2, Website3 וכו'. את ההגדרה ברשימה הנפתחת Language אפשר להשאיר כפי שהיא, מכיוון שהשפה אינה משנה בתרגיל זה. לחץ OK ליצירת האתר ב-VWD.
3. הוסף תמונה לשורש האתר וקרא לה Header.jpg. אתה יכול לשחרר בשורש האתר אחת מהתמונות של הפרק הקודם, או שתשתמש בתמונה קיימת ותוסיף אותה לאתר. אם לא תשנה את שם התמונה ל-Header.jpg, עליך להתאים את הקוד בצעד הבא.

4. כשאתה במצב Default.aspx, הוסף את הקוד הבא אל Markup View. הקוד מכניס שלושה פקדי Image של ASP.NET תוך שימוש בדרכים שונות לפנייה לתמונות. התמונות מופרדות זו מזו על ידי מעברי שורה:

```
<asp:Image ID="Image1" runat="server" ImageUrl="Header.jpg" /><br />
<asp:Image ID="Image2" runat="server" ImageUrl="/Header.jpg" /><br />
<asp:Image ID="Image3" runat="server" ImageUrl="~/Header.jpg" /><br />
```

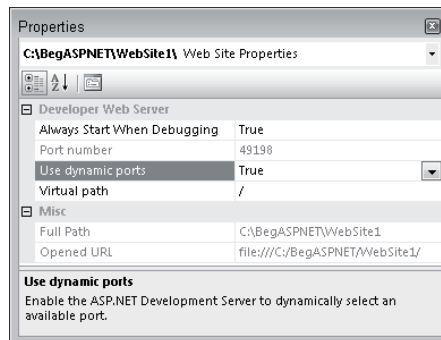
5. הקש Ctrl+F5 לפתיחת הדף בדפדפן. שים לב שסרגל הכתובת בדפדפן מראה משהו דומה ל- http://localhost:49198/WebSite1/Default.aspx. מספר היציאה (port) ושם היישום שלך עשויים להיות שונים במקצת, אבל מה שחשוב לשים אליו לב הוא שהאתר נמצא בתיקייה WebSite1 שבשרת אינטרנט בשם localhost. תמצא גם שהתמונה השנייה חסרה. הסיבה לכך היא שהלוכסן המוביל מפנה לשורש של שרת האינטרנט ולכן החיפוש אחר התמונה מתרחש בכתובת http://localhost:49198/Header.jpg שאינה קיימת, כי התמונה נמצאת בתת-תיקייה WebSite1.

פתח בדפדפן את קוד המקור של הדף והבט בשלושת האלמנטים :

```
<br />
<br />
<br />
```

שני ה-URL הראשונים זהים למה שהוספת לדף ASPX. השלישי פונה למיקום התיקייה בה נמצאת התמונה.

6. לחץ על שורש הפרויקט בסיר הפתרון, ואז הקש F4 כדי לפתוח את טבלת המאפיינים של האתר. שנה את הגדרת Virtual Path מ-WebSite1/ ל- /, כפי שמוצג בתרשים 7-2.



תרשים 7-2

7. הקש שוב Ctrl+F5 כדי לפתוח מחדש את Default.aspx בדפדפן. סרגל הכתובת מראה כעת משהו דומה לכתובת http://localhost:49198/Default.aspx. כפי שאפשר לראות, הדף Default.aspx נמצא עכשיו בשורש השרת. לכן כל שלוש התמונות מוצגות כהלכה.

8. חזור ל-VWD וצור תיקייה בשם Test. גרור את הקובץ Default.aspx משורש האתר אל התיקייה החדשה ואז בקש את הדף בדפדפן. התמונה הראשונה הפעם חסרה. אם תתבונן במקור של קוד HTML תוכל לראות זאת:


```
<br />
<br />
<br />
```

האלמנט `<img>` הראשון מנסה למצוא תמונה במיקום יחסי למסמך הנוכחי. מכיוון שהמסמך הנוכחי נמצא בתיקייה Test ואילו התמונה נמצאת בשורש האתר, התוצאה היא תמונה חסרה. שתי תכונות src האחרות מצביעות על התמונה הנכונה בשורש האתר.

9. סגור את פרויקט הדוגמה ב-VWD, ומחק אותו מהדיסק מפני שלא נצטרך אותו יותר.

כיצד זה פועל

הדוגמה מציגה כיצד תחביר ~ לציון שורש האתר הוא לעתים קרובות הדרך האמינה ביותר לפנייה למשאב כמו תמונה. בשלושת המקרים השונים שראית בתרגיל זה, רק התמונה השלישית הופיעה תמיד כהלכה. כשפקד כמו `<asp:Image>` צריך לנדר נתיב, הוא מוצא מהו שורש היישום הנוכחי ומתאים את הנתיב בהתאם. הדבר יכול להביא לחיסכון ממשי בזמן, במקרה שתחליט להעביר תיקיות וקבצים, או לפתח את האתר תוך שימוש בהגדרת ברירת המחדל של VWD שיוצרת תיקיית יישום נוספת, ואז מנסה לשים את האתר שלך בשרת ייצור שמאפשר גישה לקבצים מהשורש הממשי של השרת.

גם פקדים בקוד HTML פשוט יכולים להפיק תועלת מתחביר זה אם יהפכו אותם לפקדי שרת. תוכל לעשות זאת אם תוסיף לאלמנט את המאפיין `runat="server"`. לדוגמה, האלמנט `<img>` הבא מציג אותה התנהגות כמו `<asp:Image>` השלישי מהדוגמה הקודמת.

```
<br />
```

בהמשך הלימוד בספר נניח שהגדרת את המאפיין Virtual Path ללוחסן נטוי קדימה, לימין (/), דבר שמקל על פנייה לקבצים שלך תוך שימוש ב-URL ללא תחביר ~. אם תמצא שכמה משאבים אינם מופיעים בצורה נכונה, בדוק אם ההגדרה של המאפיין Virtual Path נעשתה כראוי.

URL מוחלט

בניגוד ל-URL יחסי שפונה למשאב מנקודת המבט של מסמך או של שורש האתר, הרי ש-URL מוחלט (absolute URL) פונה למשאב על ידי ציון הנתיב המלא אליו. כך, במקום לפנות באופן ישיר לתמונה ולציין תיקייה לפי בחירה, עליך לפרט את השם המלא של התחום (domain) ומידע על הפרוטוקול (התחילית http://). להלן דוגמה שמתייחסת ללוגו של Wrox באתר <http://p2p.wrox.com>:

```

```

URL מוחלט דרוש כשברצונך לפנות למשאב שנמצא מחוץ לאתר שלך. עם URL כזה, התחילית http:// חשובה. אם תשמיט אותה, הדפדפן יחפש תיקייה בשם p2p.wrox.com בתוך האתר שלך ולא יידע שהתכוונת לפנייה לאתר חיצוני.

URL מוחלט הוא חד-משמעי. הוא פונה תמיד אל מקום קבוע, וכך מובטח לך שהפנייה היא תמיד אל אותו משאב, ללא תלות במקום שנמצא מסמך המקור הפונה אל המשאב. הדבר עשוי להביא למחשבה ש-URL זה הינו המועדף לשימוש בכל מקום, גם בעת פנייה למשאבים בתוך האתר שלך. אבל הדבר אינו נכון! המידע הנוסף לגבי התחום והפרוטוקול מוסיף לגודל הדף בדפדפן, ולטעינה איטית יותר של הדף שלא לצורך. נגרמים גם קשיים במקרה של שינוי שם התחום, או כשברצונך להשתמש בפונקציונליות מסוימת באתר אינטרנט אחר. לדוגמה, אם האתר שלך רץ תחת www.mydomain.com אבל אתה מעביר אותו אל www.someotherdomain.com, תצטרך כעת לעדכן כל URL מוחלט בכל רחבי האתר.

גם במהלך הפיתוח עלולים להתעורר קשיים בעבודה עם URL מוחלט. לעתים קרובות האתר נבדק על ידי URL כמו <http://localhost>. אם תגדיר פנייה לתמונות עם שימוש ב-URL זה, ברגע שתעביר את האתר לסביבת הייצור בכתובת כמו www.PlanetWrox.com התמונות לא יופיעו.

בקיצור, השתמש ב-URL מוחלט בזהירות. תמיד תצטרך URL אלה כדי לפנות למשאבים שמחוץ לאתר שלך, אבל בתוך האתר שלך עליך להעדיף שימוש ב-URL יחסי היכן שרק אפשר.

הבנת דפי ברירת המחדל

בהקשר של URL לסוגיו, עליך לדעת גם על דפי ברירת המחדל (default documents). כשאתה מדפדף לאתר כמו <http://www.domainname.com> אתה רואה שבאורח פלא מופיע דף. כיצד זה קורה? ובכן, הדבר פשוט למדי: לכל שרת יש דפי ברירת מחדל, שאינם אלא רשימה של שמות דפים שהדפדפן יכול לבקש כשלא מספקים לו שם דף מפורש. כשאתה מדפדף אל <http://www.domainname.com>, שרת האינטרנט סורק את התיקייה המבוקשת ומעבד את הקובץ הראשון שהוא מוצא ברשימת דפי ברירת המחדל. מכאן, שכאשר אתה מדפדף אל <http://www.domainname.com> בשרת אינטרנט של ASP.NET, תקבל למעשה את הדף <http://www.domainname.com/Default.aspx>.

ככלל, בקישורים שאתה יוצר עליך להתעלם מתוספת שם הדף [Default.aspx](http://www.domainname.com/Default.aspx), כשאינן בו צורך. הדבר מקטין את גודל הדף, וחשוב יותר, מקל על המשתמשים להקליד את הכתובת.

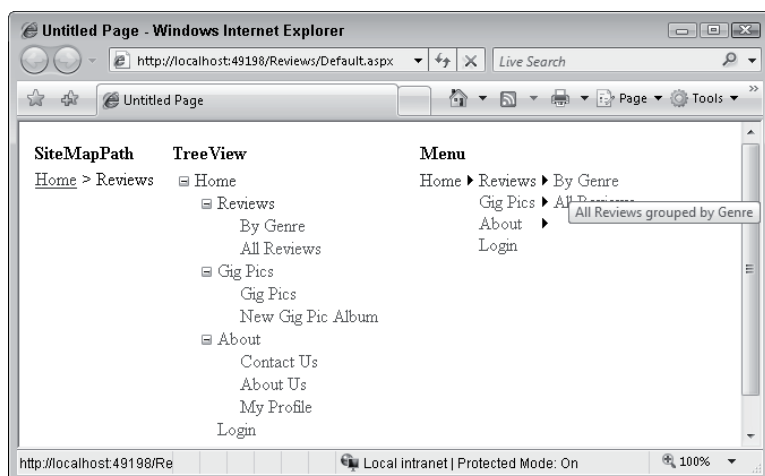
יש שרתים שבהם מוגדר [Index.aspx](http://www.domainname.com/Index.aspx) כדף ברירת המחדל. אם תמצא שהדפים שלך אינם נטענים בצורה נכונה באתרי web שבסביבת ISP, נסה לשנות את שם הדף מ-[Default.aspx](http://www.domainname.com/Default.aspx) ל-[Index.aspx](http://www.domainname.com/Index.aspx).

כעת, משראית כיצד להשתמש ב-URL כדי להצביע על דפים וקבצים אחרים, הגיע הזמן להתבונן בפקדי הניווט של ASP.NET שמשתמשים ב-URL.

שימוש בפקדי הניווט

ASP.NET 3.5 מציעה שלושה כלי ניווט שימושיים: SiteMapPath, Menu ו-TreeView. תרשים 7-3 מציג דוגמאות בסיסיות של שלושת פקדי ניווט (Navigation controls), מבלי שהוחל עליהם עיצוב כלשהו.

הפקד **SiteMapPath** בצד שמאל מראה למשתמש את הנתבי של הדף הנוכחי. הדבר מסייע כשמשמש רוצה לעלות רמה אחת, או יותר, בהיררכיה של האתר. הדבר גם עוזר למשתמשים להבין היכן הם נמצאים. הפקד **TreeView** במרכז יכול להציג את מבנה האתר ולאפשר הרחבה או כיווץ של הצמתים השונים. בתרשים 7-3 הורחב כל ענף. הפקד **Menu**, בצד ימין, מציג Home רק כשהדף נטען לראשונה. עם זאת, ברגע שתניע את העכבר על פניו, יופיע תת-תפריט שיציג את צמתי-הבן (child nodes) של אלמנט האב Home. בתרשים 7-3, אחד מהאלמנטים מסוג בן הוא התפריט Reviews, שגם לו עצמו יש אלמנטים מסוג בן.



תרשים 7-3

ארכיטקטורת פקדי הניווט

כדי להקל על הצגת דפים רלוונטיים באתר תוך שימוש ב-Menu, TreeView או SiteMapPath, משתמשת ASP.NET בקובץ מבוסס XML שמתאר את המבנה הלוגי של האתר. שם ברירת המחדל של קובץ זה הוא Web.sitemap (מפת האתר). פקדי הניווט באתר משתמשים בקובץ זה כדי להציג קישורים רלוונטיים בצורה מאורגנת. על ידי חיבור אחד מפקדי הניווט לקובץ Web.sitemap, באפשרותך ליצור אלמנטים מורכבים של ממשק המשתמש כמו תפריטים נפרשים (fold-out menus) או TreeView.

בחינת הקובץ Web.sitemap

כברירת מחדל, עליך לקרוא לקובץ מפת האתר בשם Web.sitemap. הדבר מאפשר לפקדים למצוא את הקובץ הנכון באופן אוטומטי. עבור תרחישים מתקדמים יותר תוכל להחזיק במספר קבצי מפת אתר בעלי שמות שונים, עם הגדרות תצורה ב-web.config שחושפות קבצים אלה למערכת. ברוב המקרים יספיק קובץ מפת אתר יחיד. גרסה בסיסית של קובץ מפת האתר עשויה להיראות כך:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="/" title="Home" description="Go to the homepage" >
    <siteMapNode url="/Reviews" title="Reviews"
      description="Reviews published on this site" />
    <siteMapNode url="/About" title="About"
      description="About this site" />
  </siteMapNode>
</siteMap>
```

קובץ מפת האתר מכיל אלמנטים siteMapNode, אשר ביחד יוצרים את המבנה הלוגי של האתר. בדוגמה לעיל קיים צומת שורש יחיד בשם Home, שבתורו מכיל שני אלמנטים בנים, את Reviews ואת About this site.

היבטים חשובים של הקובץ Web.sitemap

לכל siteMapNode יכולים להיות כמה צמתי-בן, עובדה זו מאפשרת ליצור מבנה אתר היררכי שיכול להיות בו-זמנית גם רחב וגם צמוק. בדוגמה שלנו הוגדרו שלושה מאפיינים באלמנטים siteMapNode: url, title ו-description. המאפיין url חייב להצביע על דף תקף באתר. ביכולתך להשתמש בתחביר ~ שראית בסעיף הקודם כדי לפנות לכל URL שמבוסס על שורש היישום. אמנם ASP.NET runtime אינו מאפשר לציין את אותו URL יותר מפעם אחת, אבל ניתן לעקוף את הבעיה על ידי הפיכת ה-URL לייחודי בדרך של הוספת מחרוזת שאילתה. לדוגמה, הכיתוב ~/Login.aspx והכיתוב ~/Login.aspx?type=Admin נראים כשני דפים נפרדים. דיון נוסף בנושא מחרוזות שאילתה יהיה בהמשך הפרק.

פקדי הניווט השונים משתמשים במאפיין **title** כדי להציג את שם הדף. מאוחר יותר, כשתעבוד עם הפקדים Menu ו-SiteMapPath, תוכל ללמוד עוד בעניין זה. המאפיין **description** משמש כתיאור כלי עבור האלמנטים שמופיעים בדפדפן.

פקדי הניווט פועלים יחד עם מנגנון האבטחה של ASP.NET. כלומר, ניתן באופן אוטומטי להסתיר אלמנטים של פקדים כמו Menu, כדי שלמשתמשים מסוימים לא תהיה גישה אליהם. דיון מקיף בנושא אבטחה מוצג בפרק 15.

כדי שתוכל לפעול עם קובץ Web.sitemap, משתמשת ASP.NET בפקד SiteMapDataSource, שכלול בקטגוריה Data של ארגז הכלים. כשאתה משתמש בפקד SiteMapPath כדי להציג "פירורי לחם" (breadcrumb) (כן, ממש כמו בסיפור של עמי ותמי - "פירורי הלחם" מאפשרים לנו למצוא

את הדרך חזרה למקום ממנו באנו), תיווכה ש-ASP.NET תמצא את קובץ Web.sitemap עצמו. עבור שני פקדי הניווט האחרים יש צורך לציין במפורש את SiteMapDataSource כשכבת ביניים לקובץ Web.sitemap.

כדי ליצור קובץ Web.sitemap שימושי, עליך להוסיף קובץ כזה לאתר, ואז להוסיף לו, באופן דינמי, את האלמנטים siteMapNode הנחוצים. Visual Web Developer אינה מציעה דרך אוטומטית ליצירת קובץ מפת אתר שמבוסס על המבנה הנוכחי של האתר. קיימים פתרונות צד שלישי שיכולים לסייע בכך.

נסה זאת **יצירת קובץ Web.sitemap**

בתרגיל זה תוסיף לאתר קובץ Web.sitemap חדש, ואחר כך תוסיף לו כמה אלמנטים siteMapNode. מפת אתר זו משמשת כבסיס עבור פקדי הניווט האחרים באתר.

1. פתח שוב את הפרויקט Planet Wrox ב-VWD, ובסייר הפתרון לחץ על הפרויקט כדי לבחור בו. הקש F4 כדי לפתוח את טבלת המאפיינים, ואז הגדר את המאפיין Virtual Path ללוחסן נטוי ימינה (/), כפי שמוצג בתרשים 2-7. מעתה והלאה ההנחה היא שאתה תמיד מפעיל את אתר האינטרנט שלך עם URL מבוסס-שורש זה.
2. לחץ לחיצה ימנית על האתר בסייר הפתרון, בחר Add New Item כדי להוסיף את הקובץ Web.sitemap לאתר, ואז לחץ Site Map. השאר את שם ברירת המחדל מוגדר ל-Web.sitemap ולחץ OK. בסיום יהיה לך אלמנט שורש אחד שמכיל שני צמתי-בן.
3. שנה את Web.sitemap באופן שיכיל את הקוד הבא:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
  <siteMapNode url="/" title="Home" description="Home">
    <siteMapNode url="/Default.aspx" title="Home" description="Go to the homepage" />
    <siteMapNode url="/Reviews/Default.aspx" title="Reviews"
      description="Reviews published on this site">
      <siteMapNode url="/Reviews/AllByGenre.aspx" title="By Genre"
        description="All Reviews Grouped by Genre" />
      <siteMapNode url="/Reviews/All.aspx" title="All Reviews"
        description="All Reviews" />
    </siteMapNode>
    <siteMapNode url="/About/Default.aspx" title="About" description="About this Site">
      <siteMapNode url="/About/Contact.aspx" title="Contact Us"
        description="Contact Us" />
      <siteMapNode url="/About/AboutUs.aspx" title="About Us"
        description="About Us" />
    </siteMapNode>
    <siteMapNode url="/Login.aspx" title="Login" description="Log in to this web site" />
  </siteMapNode>
</siteMap>
```

הערה: זכור שאינך חייב להקליד את כל הקוד הזה בעצמך. תוכל למצוא עותק של הקובץ בקובץ הקוד של פרק זה באתר של הוד-עמי, או באתר הספר המקורי.

4. שמור את הקובץ. בשלב זה סיימת את השימוש בו.

כיצד זה פועל

למרות שהקוד שהוספת לקובץ Web.sitemap נראה די פשוט, יש בו כמה דברים שכדאי לדון בהם. תחילה שים לב, שמפת האתר מכילה רק צומת שורש יחיד בשם Home. הדבר נכפה על ידי הקובץ Web.sitemap, שאינו מאפשר יותר מאשר אלמנט שורש אחד. החיסרון שבדבר הוא שאלמנט שורש יחיד זה יהיה גם פריט השורש של הפקדים Menu ו-TreeView. תרשים 3-7 מראה כיצד כל תת-התפריטים של TreeView מופיעים תחת הצומת Home. עם זאת, ברוב אתרי האינטרנט מקובל שהפריט Home הוא באותה רמה כמו האחרים. לכן, הוספת עוד צומת Home באופן ישיר תחת צומת ההורה כדי להעמידו בשורה אחת עם הפריטים About ו-Reviews. בתרגיל מאוחר יותר תלמד כיצד להסתיר אלמנט שורש מהפקדים, כדי להציג רק את "הבנים הראשונים" של צומת השורש ואת בניהם. כדי להתגבר על המגבלה שכל ה-URL באלמנטים siteMapNode צריכים להיות ייחודיים, הגדרת את הצומת הראשון ל- ~/ ואת השני ל- ~/Default.aspx. בגלל הדרך שבה מטפלים שרתי אינטרנט בדפי ברירת המחדל, הרי שבסופו של דבר שתי ההגדרות מצביעות לאותו קובץ.

שים לב שכמה מהצמתים מציינים דף בשם Default.aspx. מוקדם יותר למדת שעליך לנסות להשאיר את שם דפי ברירת המחדל כפי שהם, כדי שכל ה-URL יהיו קצרים וקלים יותר להקלדה. עם זאת, ללא התייחסות מפורשת לדף כמו Default.aspx, מפת האתר ופקדי הניווט יאבדו את עקבות הדף שאתה מנסה לפנות אליו. על כן, תמיד מוטב להוסיפו במפורש. זה אינו כרוך בקושי מיוחד משום שהמשתמשים אינם צריכים להקליד קישורים אלה בעצמם; הם יכולים ללחוץ עליהם בפקד הניווט הרלוונטי. המשתמשים עדיין יכולים לגשת לאותו דף באופן ידני, תוך שימוש בסימון הקצר יותר שלו.

קובץ Web.sitemap לכשעצמו אינו שימושי במיוחד. יש צורך להוסיף פקדי ניווט לאתר כדי להשתמש במפת האתר. בסעיף הבא תראה כיצד להשתמש בפקד Menu, ובפקדים האחרים.

שימוש בפקד Menu

הפקד <asp:Menu> קל מאוד לשימוש ולהתאמה. כל מה שצריך לעשות כדי ליצור תפריט בסיסי הוא להוסיף את הפקד לדף ולחבר אותו לפקד SiteMapDataSource, וזהו. יחד עם זאת, הפקד גמיש למדי ויש לו 80 מאפיינים ציבוריים (כולל אלה שמשותפים לכל הפקדים) שמאפשרים לשנות ולהתאים כל היבט חזותי שלו. הטבלה הבאה מציגה את המאפיינים הנפוצים ביותר שמשתמשים בהם עבור הפקד.

מאפיין	תיאור
CssClass	מאפשר להגדיר מאפיין מחלקת CSS שיטפל בעיצוב הפקד כולו.
StaticEnableDefaultPopOutImage	ערך בוליאני הקובע אם תמונות משמשות לציון תת-תפריטים של הפריטים ברמה העליונה של התפריט.
DynamicEnableDefaultPopOutImage	ערך בוליאני הקובע אם תמונות משמשות לציון תת-תפריטים של פריטי תת-תפריט.
DisappearAfter	קובע את הזמן באלפית שנייה עבור המשך התצוגה של פריטי תפריט לאחר שתזיז מהם את העכבר.
MaximumDynamicDisplayLevels	קובע כמה רמות של פריטי תת-תפריט יכול הפקד להציג. שימושי עם מפות אתר גדולות מאוד להגבלת מספר הפריטים שנשלחים לדפדפן.
DataSourceID	ה-ID (מזהה) של פקד מסוג SiteMapDataSource שמספק את הנתונים עבור התפריט מהקובץ Web.sitemap.
Orientation	קובע אם להשתמש בתפריט אופקי עם תת-תפריטים נפתחים (drop out), או בתפריט אנכי עם תת-תפריטים נפרשים (fold out).

הפקד Menu מכיל כמה מאפיינים שמתחילים ב-Static או Dynamic. המאפיינים Static משמשים לשליטה בפריטי התפריט הראשי שמופיעים כשהדף נטען. הפריטים נחשבים לסטטיים מכיוון שאינם משתנים ואינם מוסתרים כשמרחפים מעליהם. תת-התפריטים הם דינמיים מכיוון שהם מופיעים רק כשמופעלים הפריטים הרלוונטיים מהתפריט הראשי.

בנוסף למאפיינים הנזכרים לעיל יש לפקד גם כמה מאפייני עיצוב שמאפשרים לשנות את המראה והתחושה של חלקים שונים של התפריט. באחד התרגילים הבאים תראה איך להשתמש באלה.

דברים שיש לשקול כשמשתמשים ב-Menu

הפקד Menu אמנם קל מאוד להגדרה ולשימוש בדפדפן, אבל יש לו גם כמה חסרונות שעליך להיות מודע להם. לדוגמה, הפקד יוצר כמות רבה של סימני עריכה תוך שימוש בטבלאות HTML אשר מוסיפות נפח לגודל הדף.

בעיה אחרת עם הקוד הנוצר היא שדפדפנים שאינם סטנדרטיים (דפדפני שולחן העבודה) מתקשים לקרוא אותו. לדוגמה, עוורים שמשמשים בקורא מסך יתקשו למצוא את הדפים הנכונים באתר.

כדי להתגבר על מגבלות אלו שחררה לאחרונה מיקרוסופט את כלי התכנות "ASP.NET CSS Friendly Control Adapters" שבעזרתו סימני העריכה הסופיים של פקד התפריט עוברים סינון ואת ה-HTML הנקי ניתן לסגנן באמצעות CSS. תוכל למצוא חומר נוסף בנושא מתאמי פקד באתר הרשמי של ASP.NET: www.asp.net/CSSAdapters/Default.aspx.

יצירת גרסה בסיסית של הפקד Menu

הדרך הטובה ביותר ללמוד כיצד פועל הפקד Menu היא ליצור תחילה גרסה מאוד בסיסית שלו. ברגע שתבין כיצד הוא עובד, וכיצד פועל המנגנון הפנימי שלו, תוכל לעצב את התפריט בהתאם לטעמך באופן שיתמזג עם שאר פעולות עיצוב האתר שלך.

נסה זאת הוספת תפריט לאתר

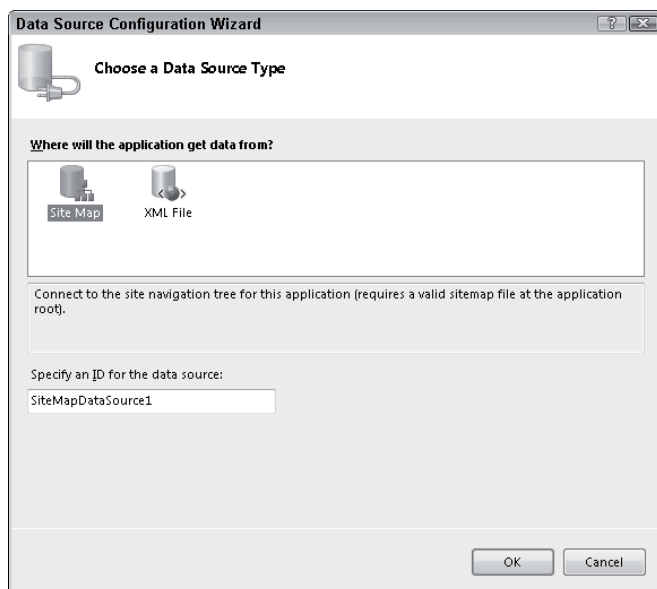
בתרגיל הבא תראה כיצד להוסיף בדף הראשי פקד Menu פשוט, שמשמש בקובץ Web.sitemap לבניית התפריט. את הפקד נוסיף לאזור MenuWrapper בדף הראשי, והוא יציג את פריטי התפריט בצורה אופקית. בגלל הכיוון שנקבע, מתאים Menu זה לערכת הנושא Monochrome בלבד. מאוחר יותר תוסיף פקד TreeView כדי לייצג את מבנה הדפים באתר עבור ערכת הנושא DarkGrey, כמו כן תכתוב קוד שיידע מתי להציג כל פקד לפי הערכת הנבחרת.

1. פתח את הדף הראשי ב- Markup View ואתר את `<div>` שנקרא MenuWrapper. הסר את שומר המקום Menu Goes Here. במקרה שבתרגילים קודמים הוספת טקסט ברירת מחדל בין התגיות ContentPlaceHolder, עכשיו הזמן להסיר אותו.
2. גרור Menu מהקטגוריה Navigation של ארגז הכלים, ושחרר אותו בין התגיות MenuWrapper. קבע את CssClass של הפקד Menu ל-MainMenu:

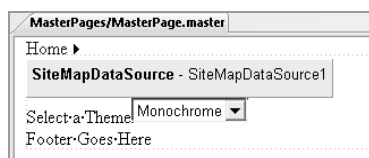
```
<div id="MenuWrapper">  
<asp:Menu ID="Menu1" runat="server" CssClass="MainMenu"></asp:Menu>  
</div>
```

3. עבור ל- Design View. ייתכן שתשים לב לכך ש- Design View אינו דומה עוד לדף הסופי. הסיבה היא שיתכן והסרת את המאפיין styleSheetTheme מהאלמנט `<pages>` ב- web.config. בינתיים אתה יכול להשאיר זאת כפי שזה. הדבר אינו כה חשוב, מכיוון שחלק ניכר מהעיצוב כבר נעשה. עם זאת, עדיין ניתן לראות איך ייראה בדפדפן התוכן שנמצא בשומר המקום cpMainContent.
4. לחץ על החץ האפור של הפקד Menu כדי לפתוח את לוח Smart Tasks.
5. מהרשימה הנפתחת Choose Data Source בחר `<New data source>`. בתיבת הדו-שיח שמופיעה לחץ על הסמל Site Map. תרשים 4-7 מראה כיצד המסך שלך ייראה כעת.
6. לחץ OK לאישור ולסגירת תיבת הדו-שיח.

7. כשתחזור לדף יראה הפקד Menu את אלמנט הרמה העליונה Home. ראה תרשים 7-5.



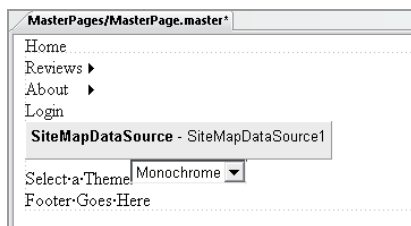
תרשים 7-4



תרשים 7-5

אם Design View שלך אינו נראה כך, אלא נראה דומה הרבה יותר לדף הסופי, פתח את הקובץ web.config והסר את המאפיין styleSheetTheme מהאלמנט <page>.

8. לחץ פעם אחת על SiteMapDataSource ואז הקש F4 לפתיחה או להפעלה של טבלת המאפיינים. שנה את המאפיין ShowStartingNode מ-True ל-False. שים לב שברגע שעשית זאת עודכן הפקד Menu במעצב והוא מראה את כל תפריטי הבנים הישירים שתחת אלמנט השורש: Home, Reviews, About, ו-Login. תרשים 7-6 מראה כיצד אמור התפריט להיראות כעת.

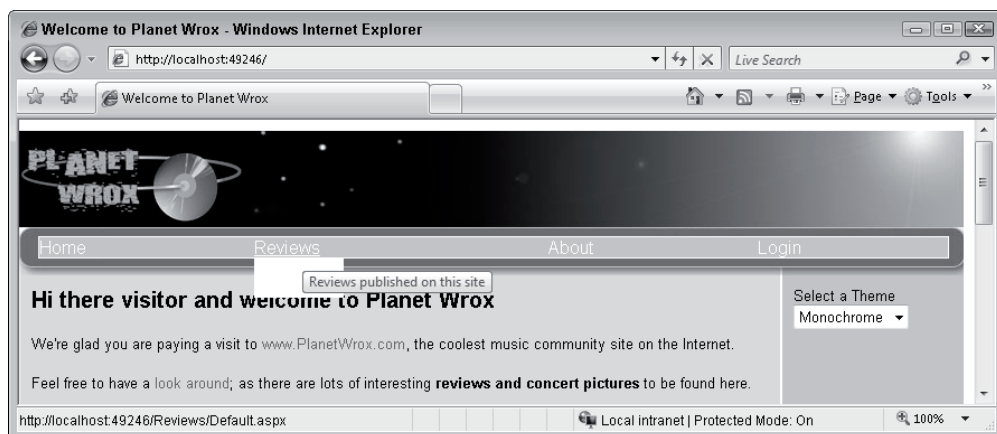


תרשים 7-6

9. לחץ פעם אחת על הפקד Menu כדי לבחור בו. ערוך את השינויים הבאים במאפייני הפקד תוך שימוש בטבלת המאפיינים:

מאפיין	ערך
StaticEnableDefaultPopOutImage	False
Orientation	Horizontal
ItemSpacing	10px
(תת-מאפיין של StaticMenuItemStyle שעליך להרחיב)	

10. שמור את השינויים שנעשו בדף הראשי, ואז בקש בדפדפן את הדף Default.aspx. אם צריך, השתמש ברשימה הנפתחת Theme כדי להפוך את Monochrome לערכת הנושא הפעילה. אתה אמור לראות כעת את התפריט באזור התפריט האופקי. עבור עם העכבר על פני הפריטים ותראה תת-פריטים מופיעים (ראה תרשים 7-7).



תרשים 7-7

שים לב שאינך יכול לקרוא את הטקסט של תת-הפריטים. הסיבה היא שקוד CSS של ערכת הנושא Monochrome שינה את הטקסט של כל העוגנים (anchors) בשטח התפריט ללבן. בו-בזמן הפקד Menu (לא קובץ CSS של הערכה) הגדיר את צבע הרקע של התפריט ללבן, ועל ידי זה הפך את התפריטים לבלתי-נראים. לאחר שראית כיצד פועל הפקד Menu, יש לך הזדמנות לתקן את הסגנון או העיצוב שלו.

אל תדאג אם התפריט אינו נראה טוב בערכת הנושא DarkGrey. בהמשך הפרק תראה כיצד ליישם פקד ניווט אחר עבור ערכת נושא זו.

כיצד זה פועל

כשנשלח לדפדפן דף עם התפריט, הפקד Menu מבקש נתונים מ-SiteMapDataSource (שמוגדר באותו דף ראשי). פקד מקור נתונים קורא את הקובץ Web.sitemap ומעביר לפקד Menu את קוד XML. בהתבססו על ה-XML ההיררכי, מסוגל הפקד Menu ליצור את קוד HTML ואת קוד JavaScript הנחוצים. הפקד יוצר טבלת HTML גדולה עם תת-טבלאות מקוננות שכל אחת מכילה אחד או יותר פריטי תפריט. הפקד Menu מסתיר את תת-התפריט באמצעות בלוק CSS שתוכל למצוא בראש הדף:

```
<style type="text/css">
    .ctl00_Menu1_0 { background-color:white;visibility:hidden;display:none;
                    position:absolute;left:0px;top:0px; }
    .ctl00_Menu1_1 { text-decoration:none; }
    .ctl00_Menu1_2 { }
    .ctl00_Menu1_3 { }
    .ctl00_Menu1_4 { }
</style>
```

כשהדף נטען, המחלקה ctl00_Menu1_0 חלה על כל תת-התפריטים, והם הופכים לנסתרים מכיוון שה-selector מגדיר visibility למצב hidden, ואת המאפיין display ל-none.

כשתרחף עם העכבר על פני אחד מפריטי התפריט הראשי, יהפוך תת-התפריט לנראה. הדבר מתבצע בעזרת JavaScript שמצורף לתגיות <td> של התפריט, כך:

```
<td onmouseover="Menu_HoverStatic(this)" onmouseout="Menu_Unhover(this)"
    onkeyup="Menu_Key(this)" title="Go to the homepage" id="ctl00_Menu1n0">
```

כשתחפש במקור הדף את קוד JavaScript שמסתיר או מציג את התפריט, לא תמצא אותו. אם כך, היכן היא פונקציית JavaScript שמשמשת להצגה או להסתרה של פריטי התפריט הרלוונטיים? התשובה טמונה בתגית הסודית <script> בדף, שנראית דומה לזה:

```
<script src="/WebResource.axd?d=7nGQ_0Tk41UEbNn9HVRm-
    A2&amp;t=633181994038144368" type="text/javascript"></script>
```

התגית <script> מפנה למטפל מיוחד של ASP.NET שנקרא WebResource.axd. התווים האקראיים לכאורה במחרוזת השאילתה (החלק של ה-URL שלאחר סימן השאלה) אומרים ל-ASP.NET runtime להביא קובץ JavaScript שמכיל את הפונקציות עבור התפריט. הקובץ אינו קיים בדיסק שלך, אבל הוא מוחזר על ידי המטפל WebResource.axd בזמן הריצה, בהתבסס על מחרוזת השאילתה.

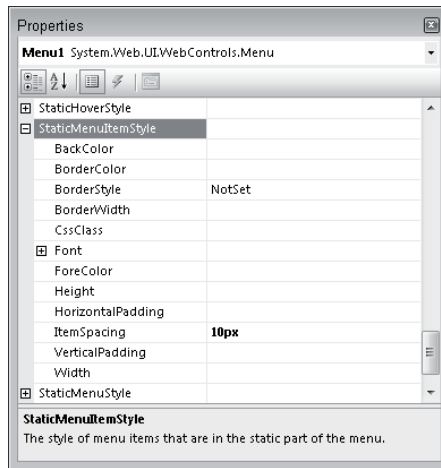
גם פקדים אחרים משתמשים בתחביר של WebResource.axd. למשל, TreeView משתמש בתחביר זה כדי לאחזר את התמונות שהוא משתמש בהן.

כדי לשלב את הפקד Menu בצורה טובה יותר עם העיצוב הקיים של ערכת הנושא Monochrome, אתה יכול לעצב אותו על ידי שימוש במאפייני עיצוב כלול או עם מחלקות CSS נפרדות.

עיצוב הפקד Menu

הפקד Menu חושף כמה מאפייני עיצוב מורכבים שמאפשרים לשנות את המראה של פריטים כמו הפריטים הראשיים ופריטי תת-התפריטים. ביכולתך גם להגדיר כיצד פריטים אלה ייראו כשהם פעילים (נבחרים), או כשאתה מרחף מעליהם עם העכבר. לכל אחד ממאפייני עיצוב אלה יש כמה תת-מאפיינים עבור היבטים חזותיים, כמו גופן, צבע וריווח. תרשים 7-8 מציג את טבלת המאפיינים של `StaticMenuItemStyle`, אשר מגדירים את המראה של פריטי התפריט הראשי שמוצגים כשהדף נטען בפעם הראשונה.

רוב המאפיינים, כמו `BackColor`, `ForeColor` ו-`Font` ממופים ישירות לסגנונות CSS כלולים. הדבר מקשה על שימוש חוזר בעיצוב עבור דפים אחרים או עבור ערכות נושא אחרות. לפיכך, מוטב להשתמש במקומם ב-`CssClass`, ואז להגדיר בקובץ CSS של ערכת הנושא את מחלקות CSS הנחוצות. בסעיף הבא תראה כיצד זה פועל.



תרשים 7-8

נסה זאת עיצוב הפקד Menu

בתרגיל זה תוסיף מחלקות CSS עבור פריטים כמו `StaticMenuItemStyle`, `StaticHoverStyle` ו-`DynamicMenuItemStyle`. אחר כך תוכל להחיל מחלקות אלו על מאפייני העיצוב הרלוונטיים של הפקד Menu.

1. פתח את `Monochrome.css` שבתיקיית ערכת הנושא `Monochrome` והוסף את כללי CSS הבאים. השאר את ההערות שבין `/*` לבין `*/` כפי שהן, מפני שכל תפקידן לתאר את מטרות ה-`selectors`. אם "לא בא לך" להקליד את כל הקוד, באפשרותך להעתיק את קובץ הקוד הזה מקבצי המקור של פרק 7.

```
.StaticMenuItemStyle, .StaticMenuItemStyle a
{
    /* Defines the look of main menu items. */
    color: #707070;
```

```

font-size: 14px;
font-weight: bold;
text-decoration: none;
padding-left: 2px;
}

.StaticHoverStyle, .StaticSelectedStyle
{
    /* Defines the look of active and hover menu items */
    background-color: #c1e4ce;
}

.DynamicMenuItemStyle
{
    /* Defines the sub menu items */
    font-size: 14px;
    color: #707070;
    background-color: #cccccc;
    padding: 4px 2px 4px 3px;
}

.DynamicHoverStyle
{
    /* Defines the hover style of sub menus */
    background-color: #707070;
    color: White;
}

.DynamicHoverStyle a
{
    /* Removes the underline from links in the sub menus */
    text-decoration: none;
}

```

2. שמור וסגור את הקובץ.

3. פתח את הדף הראשי, עבור למצב Design View, ובחר בפקד Menu. הקש F4 לפתיחת טבלת המאפיינים ואז הגדר את CssClass של מאפייני הסגנון הבאים:

שם המאפיין	קבע ערך ל-CssClass	הסבר
StaticMenuItemStyle	StaticMenuItemStyle	הגדרת מראה פריטי התפריט הראשי.
StaticHoverStyle	StaticHoverStyle	הגדרת הסגנון שמוחל על התפריט כשאתה מרחף עם העכבר על פני פריטי התפריט הראשי.

שם המאפיין	קבע ערך ל-CssClass	הסבר
StaticSelectedStyle	StaticSelectedStyle	הגדרת המראה של הפריט שנבחר מהתפריט הראשי.
DynamicMenuItemStyle	DynamicMenuItemStyle	הגדרת המראה של פריטי תת-התפריט.
DynamicHoverStyle	DynamicHoverStyle	הגדרת הסגנון שמוחל על התפריט כשאתה מרחף על פני פריטי תת-התפריט.

שים לב שכדי לפשט דברים, הערך עבור ה-CssClass זהה לשם המאפיין אותו משנים. הדבר אינו דרוש, אבל מקל מאוד על מציאת ה-selectors הרלוונטיים בקובץ ה-CSS.

4. ב-Markup View, הסר את המאפיין ItemSpacing מהאלמנט StaticMenuItemStyle. כשתסיים, הפקד Menu צריך להיראות כך:

```
<asp:Menu ID="Menu1" runat="server" DataSourceID="SiteMapDataSource1"
    CssClass="MainMenu"
    Orientation="Horizontal" StaticEnableDefaultPopOutImage="False" >
    <StaticMenuItemStyle CssClass="StaticMenuItemStyle" />
    <StaticHoverStyle CssClass="StaticHoverStyle" />
    <StaticSelectedStyle CssClass="StaticSelectedStyle" />
    <DynamicMenuItemStyle CssClass="DynamicMenuItemStyle" />
    <DynamicHoverStyle CssClass="DynamicHoverStyle" />
</asp:Menu>
```

5. צור את התיקיות והקבצים הבאים שבהם תשתמש בפרק זה ובפרקים הבאים. השתמש בתבנית MyBasePage כדי ליצור את הקבצים החדשים. בנוסף, במצב Markup View, קבע Title בעל משמעות לכל דף, כדי למנוע שגיאות מאוחר יותר.

תיקייה	שם קובץ	כותרת
/Reviews	Default.aspx	My Favorite Reviews
	All.aspx	All Reviews
	AllByGenre.aspx	Reviews Grouped by Genre
/About	Default.aspx	About this Site
	Contact.aspx	Contact Us
	AboutUs.aspx	About Us

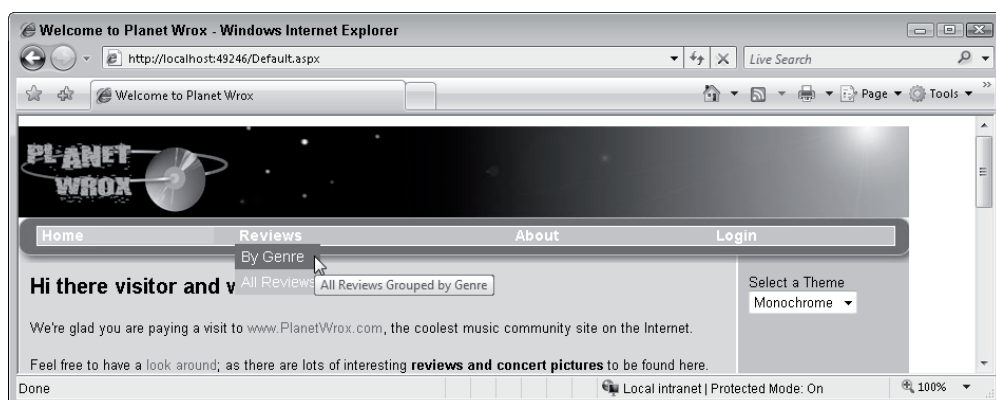
כשתעבוד עם VB, תקבל שגיאה עבור שני הדפים Default.aspx. הסיבה היא שעם התבנית שיצרת קודם אתה מסיים במחלקה בשם Default עבור קובץ בשם Default.aspx. אולם, Default היא מילה שמורה ב-VB, והדבר מוביל לשגיאה. קל לתקן זאת. בקוד ברקע קבע למחלקה Default תחילית של קו תחתון, כמו כאן:

```
Partial Class _Default
Inherits BasePage
```

לאחר שינוי שם המחלקה, אתה צריך לשנות גם את המאפיין Inherits בסימני העריכה של הדף:

```
<%@ Page Language="VB" MasterPageFile="~/MasterPages/MasterPage.master"
AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="_Default"
title="My Favorite Reviews" %>
```

6. שמור את כל השינויים, ופתח את הדף Default.aspx בדפדפן. תפריט האתר נראה כעת טוב בהרבה, ותואם יותר את שאר ערכת הנושא Monochrome. כשתרחף עם העכבר על פני התפריט הראשי יופיעו תת-תפריטים כשהם מציגים את הטקסט על רקע אפור בהיר. כשתרחף על פני תת-תפריט, צבע הרקע ישתנה שוב. תרשים 7-9 מציג את תפריט Reviews המורחב, כשהסגנון hover מוחל על פריט התפריט By Genre.



תרשים 7-9

אם תקבל שגיאה כשתנווט אל אחד הדפים החדשים שיצרת, ודא שקבעת לכולם כותרת תקפה. מכיוון שכולם יורשים מדף הבסיס, הכותרת נבדקת כשהדף נטען.

כיצד זה פועל

הפקד Menu חושף מאפייני סגנון עבור כל החלקים הרלוונטיים של התפריט, כולל פריטי התפריט הראשי (במצבים רגיל, נבחר או ריחוף), והפריטים השונים של תת-תפריט במצבים שונים. ביכולתך להגדיר מאפייני עיצוב סגנון ייחודיים עבור כל האובייקטים, אבל תוצאת הדבר תהיה שליחת כמות גדולה של קוד HTML ו-CSS אל הדפדפן. פתרון נקי יותר הוא ליישם (apply) מחלקות CSS, שביכולתך לנהל באופן ידני, בקובץ CSS של ערכת הנושא.

כשהפקד Menu יוצר את קוד HTML שלו, הוא מיישם את המחלקה הנכונה של CSS על חלקים מהתפריט. לדוגמה, StaticHoverStyle מיושם כך:

```
<table class="StaticMenuItemStyle ct100_Menu1_4 StaticSelectedStyle ct100_Menu1_9"
  cellpadding="0" cellspacing="0" border="0" width="100%">
  <tr>
    <td style="white-space: nowrap;">
      ... Menu items go here; not shown
    </td>
  </tr>
</table>
```

הטבלה מיישמת את המחלקה StaticMenuItemStyle, וזו קובעת את גודל הגופן והצבע של פריטי התפריט, מסירה את הקו התחתון מהקישורים בתפריט, ומוסיפה לפריט מילוי של כמה פיקסלים, כדי ליצור מעט רווח בין הגבול השמאלי של פריט התפריט לבין הטקסט שלו. שאר מחלקות CSS פועלות בדיוק כך. כל אחת מהן משרתת מטרה נפרדת ומשמשת להגדרת המראה של האלמנט שעליו היא מיושמת.

הפקד Menu בצורתו האופקית הוא אידיאלי עבור ערכת הנושא Monochrome, מכיוון שהוא מציג סרגל ניווט אופקי. עבור הערכה DarkGrey תוכל להשתמש באותו Menu ולהגדיר את המאפיין Orientation שלו ל-Vertical. הדבר יוצר תפריט אנכי עם הפריטים העיקריים בערימה זה על גבי זה, בעוד שתת-התפריטים ייפרשו החוצה (fold out) לימין של התפריט הראשי. במקום להשתמש בפקד Menu אפשר להשתמש בפקד TreeView להצגה היררכית של מפת האתר. פקד זה מוצג בסעיף הבא.

שימוש בפקד TreeView

הפקד TreeView יכול להציג רשימה היררכית של פריטים, בדומה למראה העץ של סייר החלונות. פריטים יכולים להיות מורחבים או מכווצים עם הסמלים הקטנים המקובלים "+" ו-"-" לפני שמות הפריטים שמכילים אלמנטים בנים. לכן פקד זה הוא הכלי המתאים ביותר להצגת מפת אתר אינטרנט כאמצעי לניווט בו. TreeView אינו מוגבל לקובץ Web.sitemap. אפשר לקשור אותו לקבצי XML רגילים, ואפשר אפילו ליצור TreeView באמצעות קוד בתכנות.

הטבלה הבאה מציגה את המאפיינים היותר נפוצים של TreeView. גם במקרה זה, העזרה המקוונת של VWD היא מקור טוב לקבלת סקירה מפורטת על כל המאפיינים הזמינים ותיאורם.

מאפיין	תיאור
CssClass	אפשרות להגדיר מאפיין של מחלקת CSS שיחול על הפקד כולו.

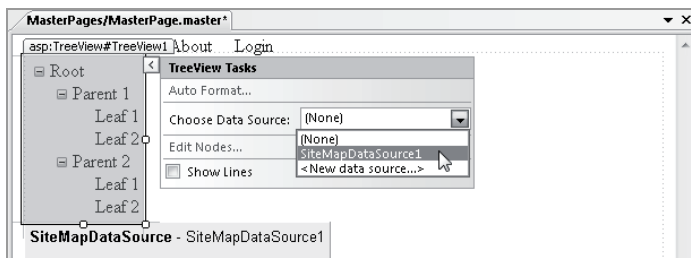
מאפיין	תיאור
CollapseImageUrl	התמונה שמכוצת חלק מהעץ כאשר לוחצים עליה. ברירת המחדל היא סמל עם סימן חיסור (-) עליו.
ExpandImageUrl	התמונה שמרחיבה חלק מהעץ כאשר לוחצים עליה. ברירת המחדל היא סמל עם סימן חיבור (+) עליו.
CollapseImageToolTip	תיאור הכלי שמוצג כשהמשתמש מרחף על פני פריט תפריט שניתן לכיווץ.
ExpandImageToolTip	תיאור הכלי שמוצג כשהמשתמש מרחף על פני פריט תפריט שניתן להרחבה.
ShowExpandCollapse	קביעה אם הפריטים ב-TreeView ניתנים לכיווץ או להרחבה על ידי לחיצת תמונה שנמצאת לפנייהם.
ShowLines	קביעה אם להשתמש בשורות להצגת חיבור הפריטים שנמצאים בעץ.
ExpandDepth	קביעת הרמה שבה פריטים בעץ יורחבו בזמן הטעינה הראשונה של הדף. ברירת המחדל היא FullyExpand, שפירושה הצגת כל הפריטים בעץ. אפשרויות הגדרה אחרות הן ערכים מספריים שמציינים את הרמה שאליה יש להרחיב.

בדיוק כמו לפקד Menu, כך גם לפקד TreeView יש כמה מאפייני סגנון שמאפשרים לשנות את המראה והתחושה של חלקים שונים של העץ. הדרך שבה הם פועלים זהה לדרך פעולתם עם הפקד Menu.

כדי לומר ל-TreeView איזה פריטים להציג, יש לקשור אותו אל הפקד SiteMapDataSource. פעולה זו מודגמת להלן.

נסה זאת בניית מערכת ניווט עם הפקד TreeView

1. פתח את הדף הראשי, ובדיוק תחת הפקד Menu הוסף פקד TreeView על ידי גרירתו מארגז הכלים אל הדף. הפעולה מתבצעת ב- Markup View.
2. עבור ל- Design View, לחץ על TreeView פעם אחת, ולחץ על החץ האפור הקטן כדי לפתוח את לוח Smart Tasks. מהרשימה הנפתחת Choose Data Source בחר SiteMapDataSource1. זהו פקד מקור הנתונים שיצרת עבור הפקד Menu (ראה תרשים 7-10).
- לאחר בחירת מקור הנתונים, מתעדכן ה-TreeView שב- Design View, ומציג את פריטי התפריט הנכונים שנמצאים בקובץ מפת האתר.



תרשים 7-10

3. פתח את טבלת המאפיינים של הפקד TreeView והגדר את המאפיין ShowExpandCollapse ל-False.

4. לחץ במקום כלשהו במסמך כדי שיהיה פעיל, ואז הקש F7 כדי לפתוח את הקוד ברקע של קובץ הדף הראשי. אתר את האירוע Page_Load שהשתמשת בו קודם לבחירה מראש של ערכת הנושא מהרשימה Theme. מייד מתחת לקוד זה, ולפני סוף הקוד של השיטה, הוסף את הקוד המוצלל הבא שמציג, או מסתיר, את הפקדים TreeView ו-Menu על פי ערכת הנושא הפעילה הנוכחית:

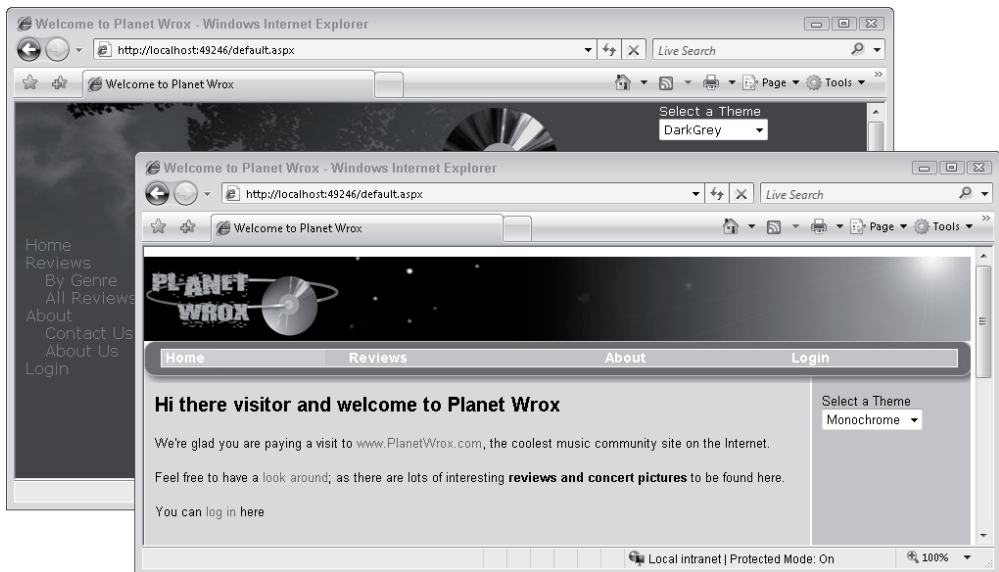
VB.NET

```
End If
End If
Select Case Page.Theme.ToLower()
    Case "darkgrey"
        Menu1.Visible = False
        TreeView1.Visible = True
    Case Else
        Menu1.Visible = True
        TreeView1.Visible = False
End Select
End Sub
```

C#

```
lstPreferredTheme.Items.FindByValue(selectedTheme).Selected = true;
}
}
switch (Page.Theme.ToLower())
{
    case "darkgrey":
        Menu1.Visible = false;
        TreeView1.Visible = true;
        break;
    default:
        Menu1.Visible = true;
        TreeView1.Visible = false;
        break;
}
}
```

5. שמור את כל השינויים ופתח בדפדפן את Default.aspx. כעת תראה את TreeView או את Menu, על פי ערכת הנושא הפעילה הנוכחית. בחר ערכת נושא אחרת מהרשימה ותראה שהדף נטען מחדש, כשהוא מציג עכשיו את הפקד האחר בתור מערכת הניווט של האתר.



תרשים 7-11

כיצד זה פועל

בדיוק כמו הפקד Menu, כך גם הפקד TreeView מקבל את הנתונים שלו מפקד SiteMapDataSource, שבתורו מקבל את המידע שלו מהקובץ Web.sitemap.

הקוד שנמצא בקובץ קוד ברקע של הדף הראשי בוחן את ערכת הנושא הנוכחית בהתאם לחקירתו את המאפיין Theme של Page. כשערכת הנושא הנוכחית היא DarkGrey, הקוד מסתיר את Menu ואז מציג את TreeView. במקרה של בלוק Case Else / default, ההיפך הוא הנכון. פירוש הדבר שעבור ערכת הנושא Monochrome וכל הערכות העתידיות שאתה עשוי להוסיף, TreeView מוסתר ו-Menu משמש במקומו בתור מערכת הניווט.

הפקד TreeView סובל מאותן בעיות כמו הפקד Menu במובן זה שהוא מייצר כמות רבה של קוד HTML מנופח. כדי לגרום ל-TreeView לייצר קוד נקי יותר, אשר מופעל על ידי CSS, אפשר להשתמש במנגנון ה-CSS Control Adapters שהוזכר קודם לכן.

לאחר הדיון בשניים מתוך שלושת פקדי הניווט, הבה נתבונן בפקד השלישי SiteMapPath.

שימוש בפקד SiteMapPath

הפקד SiteMapPath מציג היכן אתה נמצא בתוך מבנה האתר. הוא מציג עצמו כסדרה של קישורים, שלעיתים קרובות מתייחסים אליה כאל פירורי לחם (breadcrumb) שהזכרנו קודם לכן. זהו פקד פשוט למדי ויחד עם זאת רב-עוצמה, עם קרוב ל-50 מאפיינים ציבוריים שבאפשרותך להגדירם על ידי טבלת המאפיינים, כדי להשפיע על המראה שלו. כמו לפקדים Menu ו-TreeView, כך גם לפקד זה יש כמה מאפייני סגנון שבעזרתם תוכל לשנות את המראה של אלמנטים בתצוגה, כמו הצומת הנוכחי, צומת רגיל, ומפריד הנתיבים (path separator).

הטבלה הבאה מציגה כמה מהמאפיינים היותר נפוצים של הפקד SiteMapPath.

מאפיין	תיאור
PathDirection	תמיכה בשני ערכים: RootToCurrent ו-CurrentToRoot. הערך הראשון גורם להצגת אלמנט השורש בצד שמאל, רמות ביניים באמצע ואת הדף הנוכחי בצד ימין של הנתיב. הערך השני גורם להצגת הדפים בסדר הפוך.
PathSeparator	הגדרת הסמל או הטקסט שיוצגו בין האלמנטים השונים של הנתיב. ברירת המחדל היא הסימן "גדול מ-" (>), אבל באפשרותך לשנות זאת לסימן אחר, כמו תו הצינור () למשל.
RenderCurrentNodeAsLink	קביעה אם האלמנט האחרון בנתיב (הדף הנוכחי) יופיע כקישור טקסט או כטקסט רגיל. ברירת המחדל היא False, שמתאימה בדרך כלל, מכיוון שאתה כבר בדף שאלמנט זה מייצג, כך שאין צורך ממשי בקישור.
ShowToolTips	קביעה אם הפקד מציג תיאור כלי כשהמשתמש מרחף על פני האלמנטים בנתיב. התיאור מאוחזר ממאפיין התיאור של האלמנטים siteMapNode בקובץ Web.sitemap.

בדרך כלל אינך צריך להגדיר אף לא אחד מהסגנונות של הפקד SiteMapPath, מותנה כמובן, בהעדפותיך האישיות. בדף הסופי בדפדפן תראה ש-SiteMapPath מורכב בעיקר מתגיות עוגן (<a>) וטקסט פשוט. אם הגדרת בקובץ CSS שלך selector מסוים עבור עוגנים, SiteMapPath יציג את עצמו בקו אחד עם שאר הקישורים בדף.

נסה זאת יצירת Breadcrumb עם הפקד SiteMapPath

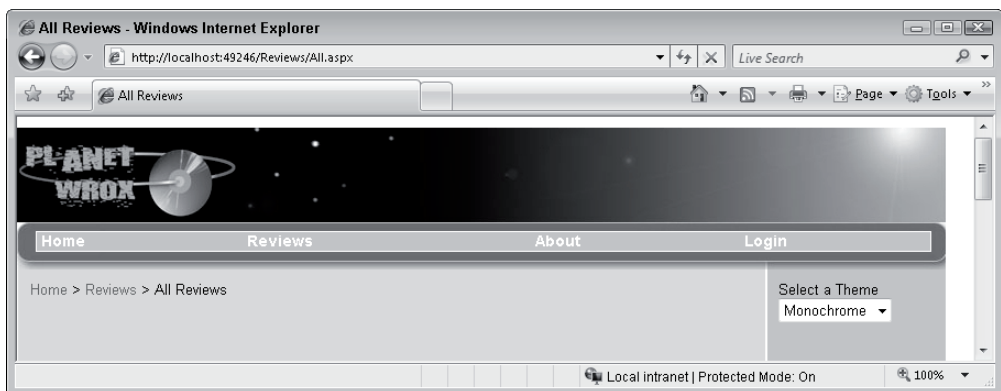
מקום טוב להצבת האובייקט SiteMapPath הוא הדף הראשי הכללי של האתר. בדרך זו הוא הופך לגלוי עבור כל הדפים באופן אוטומטי.

1. פתח את הדף הראשי ב-Markup View ואתר את תגית הפתיחה של <div> MainContent. גזור את SiteMapPath מארגז הכלים ושחרר אותו מייד לאחר תגית ה-div הנזכרת ולפני

התגית `<asp:ContentPlaceHolder>`. מייד לאחר `SiteMapPath` הוסף שני מעברי שורה
הקוד שתקבל ידמה לזה:

```
<div id="MainContent">  
<asp:SiteMapPath ID="SiteMapPath1" runat="server"></asp:SiteMapPath><br /><br />  
<asp:ContentPlaceHolder ID="cpMainContent" runat="server">
```

2. שמור את השינויים ובקש את `Default.aspx` בדפדפן. שים לב שהפקד `SiteMapPath` מראה כעת את הנתיב משורש האתר (מזוהה על ידי הקישור `Home`) אל הדף הנוכחי. לחץ על כמה מהפריטים בפקד `Menu` או בפקד `TreeView` כדי לנווט ברחבי האתר, ותראה את breadcrumb משתנה עבור כל דף. תרשים 7-12 מציג את breadcrumb עבור הדף `All Reviews` שהוא תת-אלמנט של `Reviews`, אשר בתורו מופיע תחת אלמנט השורש `Home`.



תרשים 7-12

כשתנווט אל אחד מתת-הדפים של האתר, תוכל ללחוץ על האלמנטים של הנתיב כדי לעלות רמה אחת או יותר. לחיצה על `Reviews` בדף שמוצג בתרשים 7-12 מחזירה אותך אל הדף הראשי של `Reviews`, בעוד שלחיצה על `Home` מחזירה אותך לשורש האתר.

3. עבור לערכת נושא אחרת על ידי שימוש בבורר `Theme`. שים לב שאין שינוי של ממש במראה `SiteMapPath`, פרט לצבעי הקישורים, שמוגדרים בכל אחד מקבצי קוד `CSS` של הערכה.

כיצד זה פועל

הנתיב `SiteMapPath` מרונדר כסדרה של אלמנטים מסוג `<span>` שמכילים קישור או טקסט פשוט. להלן חלק מקוד `HTML` עבור `SiteMapPath` שמוצג בתרשים 7-12.

```
<span><a title="Home" href="/">Home</a></span>  
<span> &gt; </span>  
<span><a title="Reviews published on this site"  
    href="/Reviews/Default.aspx">Reviews</a></span>  
<span> &gt; </span>  
<span>All Reviews</span>
```

שני האלמנטים הראשונים (Reviews-ו Home) מיוצגים על ידי קישור (<a>) כדי לאפשר ניווט אל דפים שמוגדרים במאפיין href שלהם. האלמנט הסופי - All Reviews - הוא רק טקסט פשוט. בין האלמנטים תראה עם התו המפריד שהגדרת במאפיין PathSeparator.

אם תסתכל בדפדפן בקוד HTML של הדף, תראה גם אלמנט <a> שמאפשר לדלג על קישורים. האלמנט מכיל תמונה קטנה שמאפייני width ו-height שלה הוגדרו ל-0px, דבר שהופך אותה לבלתי נראית. הדבר משמש קוראי מסך שמאפשרים למשתמשים לדלג על הניווט ולעבור ישירות אל תוכן הדף. הפקדים TreeView ו-Menu משתמשים בגישה דומה כדי למנוע מקורא מסך לקרוא בקול רם את כל מבנה האתר בכל פעם שהדף נטען.

שלושת פקדי הניווט שהוצגו מאפשרים ללקוח לנווט באתר האינטרנט בצורה קלה ונוחה. הן הפקד TreeView והן הפקד Menu מאפשרים הצגה מהירה של כל מבנה האתר, וכך המשתמשים יכולים למצוא בו את דרכם בקלות. SiteMapPath מסייע למשתמשים להבין את מקומם באתר ונותן להם דרך קלה לנווט לדפים גבוהים יותר בהיררכית האתר.

בנוסף לניווט מדפדפן הלקוח, נפוץ מאוד גם לנווט את המשתמש אל דף אחר על ידי שימוש בקוד תכנות. הסעיף הבא מתאר כיצד זה נעשה.

הכוונה מחדש על ידי תכנות

הכוונה מחדש על ידי תכנות שימושית מאוד ונפוצה בדפי ASP.NET. לדוגמה, תוכל להניח שיש לך דף שמאפשר למשתמשים להכניס סקירה (ביקורת) שלהם לבסיס נתונים. ברגע שהם לוחצים על הלחצן Save, הסקירה נשמרת, והמשתמשים נשלחים לדף אחר שבו הם יכולים לראות את הסקירה מוצגת במלואה.

ASP.NET תומכת בשתי דרכים עיקריות להכוונת משתמשים לדף חדש על ידי תכנות: שימוש בשיטה Redirect של המחלקה Response ובשיטה Transfer של המחלקה Server. מכיוון שהתנהגותן שונה למדי, יוקדשו שני הסעיפים הבאים לתיאור מפורט של התנהגות זו.

הכוונת הלקוח לדף אחר על ידי תכנות

בתוך כל דף ASPX יש גישה למאפיין בשם Response, שכבר פגשת בזמן שמירת העוגייה עבור ערכת הנושא שנבחרה. האובייקט Response (מסוג HttpResponse) מאפשר גישה למאפיינים שימושיים ולשיטות, שכולם קשורים לתגובת השרת לקריאות של דפדפן המשתמש. אחת מהשיטות האלו היא השיטה Redirect. שיטה זו שולחת הוראה לדפדפן לבקש דף חדש. הדבר שימושי כשברצונך לכוון את המשתמש לדף אחר באתר, או לאתר אינטרנט שונה לחלוטין. ניתן להשתמש בשיטה Redirect בשתי דרכים שונות:

```
Response.Redirect(newUrl)
Response.Redirect(newUrl, endResponse)
```

לאפשרות השנייה יש פרמטר בוליאני נוסף בשם `endResponse`, שמאפשר לך לבצע את המשך הקוד לאחר הקריאה ל- `Response.Redirect`, בתנאי שהפרמטר `endResponse` קיבל את הערך `False`. בדרך כלל אין בזה צורך ועדיף לקרוא לגרסה הראשונה, שמסיימת את התגובה כברירת מחדל.

לעתים קרובות כשרוצים לשלוח את המשתמש לדרך אחר, רוצים גם להעביר מידע נוסף. אפשר להעביר זאת במחרוזת השאילתה (query string) - חלק מהכתובת שנמצא לאחר שם הדרך ומופרד על ידי סימן שאלה. עיין ב-URL (הכתובת) הבא:

```
http://localhost:49246/Demos/Target.aspx?CategoryId=10&From=Home
```

כל החלק בגופן מודגש לאחר סימן השאלה, נחשב למחרוזת השאילתה. המחרוזת מורכבת מזוגות של שם-ערך, כל זוג מופרד מאחר על ידי התו `&`. במקרה זה יש שני זוגות: `CategoryId` עם ערך 10 ו-`From` עם ערך שהוא המילה `Home`. הדרך, `Target.aspx` בדוגמה זו, מסוגל לקרוא ולהתייחס לערכים אלה על ידי השיטה `Request.QueryString`. בתרגיל הבא תלמד איך להשתמש במחרוזת השאילתה.

נסה זאת הכוונת המשתמש לדרך אחר

כדי לקבל מבט מקרוב על דרך הפעולה של ההכוונה, תיצור בתרגיל זה דף שמכוון אותך מדרך אחד לדרך אחר על ידי שימוש ב- `Response.Redirect`.

1. צור שני דפי אינטרנט חדשים בתיקייה `Demos`. הדפים צריכים להתבסס על התבנית שייצאת. קרא להם `Source.aspx` ו-`Target.aspx`. הגדר את ה-`Title` שלהם ל-`Source-Target` בהתאמה.

2. פתח את `Source.aspx` ב- `Design View`. כדי להגדיר מטפל לאירוע `Page_Load`, לחץ לחיצה כפולה במקום כלשהו בשטח האפור, לקריאה בלבד, של הדרך, מחוץ ל-`ContentPlaceHolder`. בתוך מטפל זה כתוב את הקוד שלהלן שמכוון את המשתמש לדרך `Target.aspx`. כדי להראות כיצד להעביר מידע נוסף באמצעות מחרוזת השאילתה, וכיצד לקרוא מידע זה בדרך היעד, הקוד מעביר מחרוזת שאילתה עם שדה בשם `Test` שערכו `SomeValue`.

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Me.Load  
    Response.Redirect("Target.aspx?Test=SomeValue")  
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)  
{  
    Response.Redirect("Target.aspx?Test=SomeValue");  
}
```

3. פתח את Target.aspx, עבור ל- Design View והוסף פקד Label לשומר המקום cpMainContent. השאר את ה-ID שלו מוגדר כ-Label1. צור מטפל לאירוע Page_Load, דומה לזה שיצרת בצעד הקודם, על ידי לחיצה כפולה במקום כלשהו בשטח האפור, לקריאה בלבד, של הדף. הוסף את הקוד הבא:

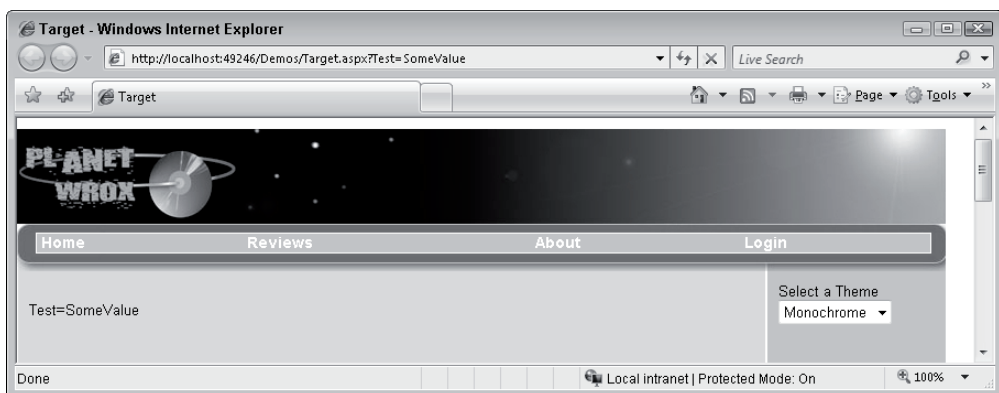
VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Me.Load  
    Label1.Text = Request.QueryString.ToString()  
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)  
{  
    Label1.Text = Request.QueryString.ToString();  
}
```

4. חזור ל-Source.aspx והקש Ctrl+F5 לפתיחתו בדפדפן. במקום לראות את Source.aspx תראה כעת את הדף שהופנית אליו, כפי שמוצג בתרשים 7-13.



תרשים 7-13

שים לב שסרגל הכתובת מכיל כעת Target.aspx?Test=SomeValue, שהוא הדף שאליה הועברת על ידי מטפל האירוע Page_Load של דף המקור Source.aspx. פקד ה-Label בדף היעד מציג את מחרוזת השאילתה שהועברה לדף היעד. שים לב שהשיטה ToString() QueryString. מכילה רק את המחרוזת Test=SomeValue. הכתובת, או אפילו סימן השאלה, אינם חלק ממחרוזת השאילתה עבור הדף.

כיצד זה פועל

כשאתה משתמש ב-Response.Redirect, שולחת ASP.NET הוראה לדפדפן להביא דף חדש. במונחים טכניים היא שולחת קוד סטטוס 302 של HTTP. זה אומר לדפדפן שהדף הנוכחי עבר למיקום חדש. יחד עם הוראה זו נשלח גם ה-URL (הכתובת) החדש, כך שהדפדפן מביין מיהו

הדף הבא שעליו להביא. בתרגיל לעיל הדף הבא היה `Target.aspx?Test=SomeValue` שמכיל גם את שם הדף וגם מחרוזת שאילתה. כשהדף `Target.aspx` מבוקש על ידי הדפדפן, מתרחש האירוע `Page_Load` של דף זה, ומחרוזת השאילתה מוצגת בתווית הדף. כתוצאה מהכוונה מחדש זו של הלקוח, חשופים ללקוח שם הדף החדש ומחרוזת השאילתה במלואם.

בניגוד לשיטה `Response.Redirect` אשר מכוונת לדף אחר על ידי שליחת הוראה לדפדפן שבתורו מבקש את הדף החדש, ישנה השיטה `Server.Transfer` שמכוונת לדף אחר מתוך השרת עצמו.

הכוונה מחדש מצד השרת

הכוונות מחדש מצד השרת הן מעולות כשברצונך לשלוח דף שונה מבלי לשנות את סרגל הכתובת של הלקוח. הדבר מאפשר להסתיר את פרטי שם הדף ואת מחרוזת השאילתה, וכך לקבל URL נקי יותר מנקודת ראות המשתמש. משתמשים בהכוונה זו בעיקר בתרחישי "URL-rewrite" שמשמשים ליצירת כתובות URL קריאות יותר. לדוגמה, משתמש עשוי לבקש דף בדרך זו:

```
http://www.domain.com/Cars/Volvo/850/T5/
```

מאחורי הקלעים עשוי השרת להעביר זאת אל

```
http://www.domain.com/Cars/ShowCar.aspx?Make=843&Model=984&Type=7345
```

ברור שה-URL הראשון קל הרבה יותר להבנה ולהקלדה בדפדפן. הוא גם מאפשר למשתמש לנחש כתובות של אתרים אחרים שתואמות לאותו דפוס או תבנית. לדוגמה, יש סבירות גבוהה שתוכל לבקש דף כמו

```
http://www.domain.com/Cars/Volvo/V70/R/
```

ותקבל את הדף הנכון שמציג את המכונית מסוג R מודל V70 של חברת Volvo.

בנוסף לכך שהן קלות יותר להבנה, הכוונות של צד-שרת עשויות להאיץ מעט את ביצועי האתר. במקום לשלוח תגובה לדפדפן כדי להורות לו להביא דף חדש, פעולה שתוצאתה בקשה חדשה לדף בשרת, אפשר להעביר את המשתמש ישירות לדף חדש, ועל ידי כך לחסוך מעט מתקורת התעבורה ברשת.

הכוונות של צד השרת מבוצעות על ידי המחלקה `Server`. כפי שהאובייקטים `Request` ו-`Response`, שראית קודם, מספקים מידע לגבי הבקשה והתגובה, כך מספק האובייקט `Server` מידע על השרת שהדף מתבצע בו. אתה יכול להשתמש בזה כדי לקבל מידע אודות שם השרת, כתובת IP שלו, וכו'. אחת מהשיטות של המחלקה היא `Transfer`, שמבצעת העברות של צד-שרת.

השיטה `Server.Transfer` משמשת רק ליצור הכוונה לדפים אחרים בתוך האתר שלך. לא ניתן להשתמש בה כדי לשלוח את המשתמש לדפים בתחומים (דומיינים / אתרים) אחרים. אם תנסה לעשות זאת, תפיק ASP.NET runtime הודעת שגיאה.

כדי לעמוד על ההבדל בין Response.Redirect לבין Server.Transfer, תוכל ללמוד בתרגיל הבא כיצד לשנות את הדרך Source באופן שיבצע פעולת Server.Transfer.

נסה זאת Server-Side Redirecting

קל לשנות את קוד ההכוונה מחדש כדי שיעביר את המשתמש לדרך אחר על ידי השרת. מה שצריך הוא להחליף את Response.Redirect ב- Server.Transfer, כפי שמוגם בתרגיל הבא.

1. פתח את קוד ברקע של Source.aspx והחלף את השורה עם Response.Redirect בשורה המוצללת הבאה:

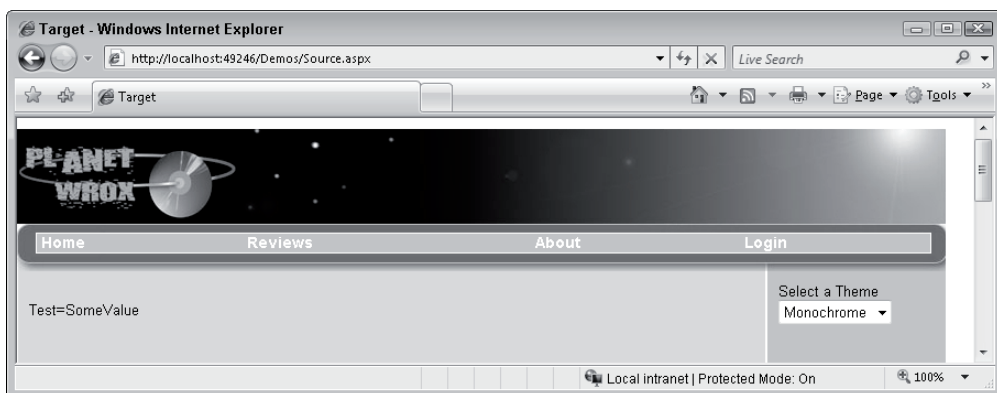
VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Me.Load  
    Server.Transfer("Target.aspx?Test=SomeValue")  
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)  
{  
    Server.Transfer("Target.aspx?Test=SomeValue");  
}
```

2. שמור את השינויים והקש Ctrl+F5 כדי לפתוח את Source.aspx בדפדפן (ראה תרשים 7-14).



תרשים 7-14

הפקד Label מציג את ערכי מחרוזת השאילתה שנשלחו מ-Source.aspx אל Target.aspx, ומדגים את העובדה שאתה אכן צופה בפלט של הדרך Target.aspx. יחד עם זאת, סרגל הכתובת של הדפדפן הושאר ללא שינוי ועדיין מראה Source.aspx, תוך הסתרת שם הדרך החדש וערכי מחרוזת השאילתה מהמשתמש.

כיצד זה פועל

במקום להורות לדפדפן להביא את הדף הבא, השיטה Server.Transfer פועלת לחלוטין בשרת. מתחיל מחזור חיים חדש עבור הדף שאליו מתבצעת ההעברה. דף זה בונה את התוכן שלו ושולח אותו אל הדפדפן, כשהוא משאיר את סרגל הכתובת של הדפדפן ללא שינוי.

כשתסתכל ב-HTML שמתקבל בדפדפן, תראה שפעילות הטופס מוגדרת עבור הדף החדש באופן שכל postback (העברה חוזרת לדף / פעילות לאירועים של פקדים בדף) מתבצע כנגד דף זה, שבתורו ישנה את הכתובת בסרגל הכתובת:

```
<form name="aspnetForm" method="post" action="Target.aspx?Test=SomeValue" id="aspnetForm">
```

```
...
</form>
```

בלימוד הכוונת הדפים באמצעות תכנות, הגענו לסוף של פרק ניווט זה. עם המושגים שהראינו בפרק זה יש בידך את כל הידע הדרוש ליצירת מערכת ניווט יעילה מאוד באתר שלך, הן מדפדפן המשתמש והן מהקוד שבצד השרת.

עצות מעשיות לניווט

הרשימה הבאה מספקת כמה עצות מעשיות לניווט יעיל ונכון:

- ⊙ כשתתחיל לבנות אתר אינטרנט שאתה צופה לו גידול בעתיד, צור עבורו מייד מבנה לוגי. אל תציב את כל הקבצים בשורש האתר, אלא רכז בתיקייה אחת את הקבצים שיש ביניהם קשר לוגי כלשהו. ריכוז כזה של קבצים יקל עליך לנהל את האתר, ועל המשתמשים יקל למצוא את הדפים שהם רוצים. אמנם קל להעביר דף ב-Menu או ב-TreeView על ידי שימוש בקובץ Web.sitemap, אבל הדבר קשה יותר כשאתה משתמש בהכוונה בדרך של תכנות או בהעברה אחרת, מכיוון שאז יש צורך לעדכן את קוד צד השרת כדי שישקף את המבנה החדש של האתר. כדי ליצור מבנה יציב של הדפים, מומלץ לשרטט את המבנה הרצוי על נייר קודם שמתחילים בהקמת האתר.
- ⊙ נסה להגביל את מספר הפריטים הראשיים או תת-הפריטים שתציג בפקדים Menu ו-TreeView. משתמשים עלולים ללכת לאיבוד או להתבלבל כשמוצגות להם רשימות ארוכות של אפשרויות בחירה.
- ⊙ כשתיצור תיקיות לאחסון הדפים, קבע להן שמות קצרים ולוגיים. יותר אינטואיטיבי לנווט לדף תוך שימוש ב- <http://www.PlanetWrox.com/Reviews> מאשר לנווט לתיקייה בעלת שם ארוך שכולל סימני קיצור ומספרים.
- ⊙ במונחי קוד HTML שמגיע אל הדפדפן, הרי שהכלי: "CSS Control Adapters" מציע חלופה טובה יותר מאשר פקדי ברירת המחדל כמו TreeView ו-Menu. טעינה, התקנה וקביעת תצורה למתאמים אינה מאוד קשה, תוכל למצוא דוגמאות קוד וחומר עזר רב

באתר: www.ASP.net. יחד עם זאת, ייתכן שהמתאמים האלה אינם מתאימים להתחלה מיידית. קרוב לוודאי שתמצא להכיר תחילה את ההתנהגות הרגילה של פקדי הניווט לפני שתעבור לשימוש בפקדים מופעלי-CSS. הקדש זמן מה ללימוד פקדי הניווט ודרך התנהגותם. נסה מאפיינים וסגנונות שונים, וראה כיצד הם משפיעים על הפקד בדפדפן. כשתגיע למידת הבנה טובה של פקדים אלה, תמצא שקל יותר לבצע את המעבר אל Control Adapters.

סיכום

בפרק זה למדת על הניווט באתרי אינטרנט מבוססי ASP.NET. המשתמשים בדרך כלל אינם מקלידים באופן ישיר את הכתובת של דף האינטרנט שהם מעוניינים בו, ולכן חשוב להציע להם מערכת ניווט ברורה ופשוטה.

הבנת הדרך שבה פועלות כתובות האתר, URL, היא בסיס חיוני למערכת ניווט טובה. יש שני סוגים של URL: URL יחסי ו-URL מוחלט. ל-URL יחסי יש שני סוגי משנה: URL יחסי מבוסס-מסמך ו-URL יחסי מבוסס-שורש. URL יחסי מבוסס-מסמך מסמך על משאב אחר בשרת באופן יחסי למסמך שמכיל את ה-URL. הדבר מקל על הפנייה למשאבים אחרים באותה תיקייה מבלי שיהיה צורך לציין את מלוא הנתיב אליה. הדבר מאפשר להעביר תיקייה באותה רמה באתר מבלי לשבור קישורים למשאבים אחרים. המגרעת של URL יחסי היא שקשה להעביר תיקיות לרמה אחרת, כמו לדוגמה העברה מהשורש אל תת-תיקייה. כדי להתגבר על בעיה זו אפשר להשתמש ב-URL יחסי מבוסס-שורש, שמציין נתיב שלם אל המשאב, החל משורש האתר. שימוש בתחביר ~ מאפשר לפנות לשורש היישום בקלות יתר.

URL מוחלט מאפשר להצביע על משאבים לפי מיקומו המלא, כולל מידע בדבר הפרוטוקול והתחום. URL מוחלט יהיה שימושי עבורך במיוחד כשתרצה להצביע אל משאבים שמחוץ לאתר האינטרנט שלך.

ASP.NET מציעה שלושה פקדי ניווט לשימוש בממשק המשתמש של אתר אינטרנט. פקדים אלה מאפשרים למשתמשים לבקר בדפים השונים באתר. הפקד Menu מוצג כתפריט אנכי או אופקי עם תת-תפריטים שנפרשים או נפתחים. הפקד TreeView יכול להציג את המבנה השלם של האתר בצורה היררכית. הפקד SiteMapPath מציג נתיב breadcrumb כדי לציין עבור המשתמש את המיקום שהוא נמצא בו באתר.

כל הפקדים האלה יכולים להשתמש בקובץ Web.sitemap, שהינו קובץ XML שמכיל את המבנה הלוגי של האתר. שימוש במפת האתר כמקור עבור פקדי הניווט, מקל על העברת תפריט למקום אחר מבלי להשפיע על המיקום הפיזי של הקובץ. כל שעליך לעשות הוא להעביר את התפריט אל siteMapNode שונה בקובץ Web.sitemap.

הפרק מסתיים במבט על הדרכים לכוון את המשתמש לדף אחר על ידי תכנות. ASP.NET תומכת בשתי דרכים עיקריות לביצוע הדבר: השיטות Response.Redirect ו-Server.Transfer.

האפשרות הראשונה מורה לדפדפן להביא דף חדש מהשרת. הדבר משפיע על סרגל הכתובת של הדפדפן כך שהמשתמש יכול לראות לאן מפנים אותו.

השיטה `Server.Transfer` פועלת בשרת. יתרונה של השיטה בכך שהיא אינה משנה את סרגל הכתובת של הלקוח. יתרון נוסף שלה בהשוואה ל- `Response.Redirect` הוא ש- `Server.Transfer` מקטינה את תקורת השרת הקשורה כאשר מבקשים דף חדש.

בפרק 8 תלמד יותר אודות פקדי `ASP.NET`, שמאפשרים לעשות שימוש חוזר בקוד ובאלמנטים של ממשקי משתמש בדפים שונים באתר.

תרגילים

1. הפקד `Menu` חושף כמה מאפייני סגנון שמאפשרים לשנות פריטים בתפריט. איזה מאפיינים עליך לשנות כשברצונך להשפיע על הדרך שבה מוצגים במסך פריטי התפריט הראשי ותת-התפריטים?
2. מהן האפשרויות שעומדות לרשותך כדי לכוון משתמש לדף אחר על ידי תכנות? מה ההבדל בין שתי הדרכים?
3. אפשר להשתמש בפקדי `TreeView` בשתי דרכים שונות: כרשימה עם פריטים ותת-פריטים שניתן לכווץ או להרחיבם על ידי לחיצה עליהם, או כרשימה סטטית שמציגה את כל הפריטים ללא דרך לכווץ או להרחיב. איזה מאפיין יש להגדיר לפקד כדי למנוע ממשתמשים מלהרחיב או לכווץ פריטים בעץ?

פרק 8

פקדי משתמש

בפרק 6 עסקנו בדפים ראשיים, ערכות נושא ו-skins, אבל אלה אינם המרכיבים היחידים של ASP.NET 3.5, ישנו מרכיב נוסף המאפשר ליצור בלוקים של מידע לשימוש חוזר, ולפיכך עקבי: **פקדי משתמש** (user controls).

פקדי משתמש מאפשרים לקבץ יחד תוכן ופקדים שקשורים באופן לוגי. ניתן להשתמש במקבץ כזה כיחידה אחת בדפי תוכן, בדפים ראשיים ובתוך פקדי משתמש אחרים. פקד משתמש הוא למעשה דף ASPX זעיר, במובן זה שיש לו קטע שמכיל סימני עריכה (markup), ועל פי בחירה גם קובץ קוד ברקע שבו ניתן לכתוב קוד עבור הפקד. עבודה עם פקד משתמש דומה מאוד לעבודה עם דף ASPX רגיל פרט לכמה שינויים בודדים.

בגרסאות ASP.NET שקדמו לגרסה 2.0 שימשו פקדי משתמש לעתים קרובות ליצירת בלוקים של פונקציונליות, או הוראות, לשימוש חוזר. בלוקים אלה היו חייבים להופיע בכל דף באתר. לדוגמה, כדי ליצור תפריט היה עליך ליצור פקד משתמש ואז להוסיף פקד זה לכל דף באתר. מכיוון ש-ASP.NET תומכת בדפים ראשיים, אין עוד צורך בפקדי משתמש בתרחישים אלה. הדבר מקל על עריכת שינויים במבנה האתר. אולם, למרות היתרונות שיש בשימוש בדפים ראשיים, עדיין יש מקום לפקדי משתמש באתרי אינטרנט מבוססי ASP.NET, כפי שתגלה בפרק זה.

בפרק זה תלמד בהרחבה על הנושאים הבאים:

- ⊙ מהם פקדי משתמש ואיך הם נראים?
- ⊙ כיצד יוצרים פקדי משתמש?
- ⊙ איך להשתמש בפקדי משתמש בדפים שלך?
- ⊙ איך לשפר את התועלת שנובעת מהשימוש בפקדי משתמש על ידי הוספת לוגיקה, ולמעשה – קוד?

כשתסיים את הלימוד בפרק זה תהיה לך הבנה ברורה מהם פקדי משתמש, וכיצד הם פועלים. הבנה זו תאפשר לך ליצור בלוקים פונקציונליים של תוכן שניתנים לשימוש חוזר.

היכרות עם פקדי משתמש

פקדי משתמש הינם מצוינים לעטיפה יחד (encapsulating) של סימני עריכה, פקדים וקוד שאתה זקוק להם שוב ושוב בדפים שונים באתר. דפים ראשיים מאפשרים ליצור תוכן שמוצג בכל הדפים באתר, אבל לעתים קרובות דרוש שתוכן מסוים יופיע בכמה דפים, אך לא בכלם. לדוגמה, ייתכן שתמצה להציג באנר (banner) בכמה דפים פופולריים, אבל לא בדף הבית או דפים נפוצים אחרים. ללא פקדי משתמש, היית מוסיף את הקוד עבור הבאנר (תמונה, קישור וכו') לכל דף שזקוק לכך. במצב זה, אילו רצית לעדכן את הבאנר (למשל, להשתמש בתמונה חדשה) היית צריך לערוך שינויים בכל הדפים שמשתמשים בו. לעומת זאת, אם תעביר את הבאנר לפקד משתמש, ותשתמש בו בדפי התוכן שלך, תצטרך לשנות את פקד המשתמש הזה פעם אחת בלבד והשינוי יחול באופן אוטומטי על כל הדפים שמשתמשים בו. זו דרך גמישה ליצירת תוכן לשימוש חוזר.

קווי הדמיון הבאים משותפים לפקדי משתמש ולדפי ASPX רגילים:

- ⊙ יש לשניהם מקטע של סימני עריכה שניתן להכניס בו סימני עריכה רגילים ופקדי שרת.
- ⊙ אפשר ליצור ולעצב אותם בעזרת Visual Web Developer, ב- Markup View וב- Design View.
- ⊙ הם יכולים להכיל לוגיקת תכנות, הן כלולה והן בקובץ קוד ברקע.
- ⊙ בשני המקרים יש גישה למידע ברמת הדף והבקשה כמו Request.QueryString.
- ⊙ הם מעוררים כמה מהאירועים שמעוררת המחלקה Page (אבל לא את כולם), כולל Init, Load ו- PreRender.

יחד עם זאת עליך להיות מודע לכמה הבדלים. לקבצים של פקדי משתמש יש סיומת ascx ואילו לדפים הרגילים יש סיומת של aspx. בנוסף, לא ניתן לבקש את פקדי המשתמש ישירות בדפדפן. לכן אינך יכול ליצור קישור מהדף לפקד משתמש ישירות. הדרך היחידה להשתמש בפקד משתמש באתר היא על ידי הוספתו לדף.

בהמשך הלימוד בפרק זה תראה כיצד ליצור פקד משתמש שמסוגל להציג באנרים. פקד המשתמש מסוגל להציג עצמו כבאנר אופקי או אנכי, כדי להתאים עצמו לאזורים בדפים שיש להם גודל שונה. בסעיף הבא תראה כיצד ליצור פקד משתמש. הסעיפים שיבואו אחריו יראו איך להשתמש בפקד בדף ASPX.

יצירת פקדי משתמש

מוסיפים פקדי משתמש לדף כפי שמוסיפים סוג תוכן אחר: באמצעות תיבת הדו-שיח Add New Item. בדומה לדפים, אתה מקבל אפשרויות שונות לבחירת שפת התכנות ולהצבת הקוד בקובץ קוד ברקע נפרד. תרשים 8-1 מראה את תיבת הדו-שיח Add New Item עבור פקד משתמש.

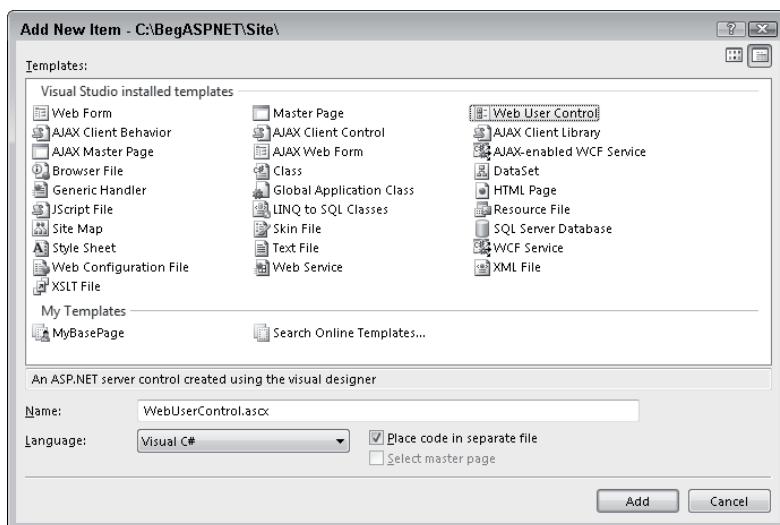
לאחר הוספת פקד משתמש לאתר, הוא נפתח באופן אוטומטי בחלון המסמך. הדבר הראשון שאתה עשוי להבחין בו הוא שלפקד משתמש אין הנחיה @ Page אלא הנחיה @ Control.

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="Banner.ascx.cs" Inherits="Controls_Banner" %>
```

שינוי זה מגדיר את הקובץ כפקד משתמש באופן ש- ASP.NET runtime יודע איך לטפל בו. פרט לכך, יתר המאפיינים בהנחיה זהים למאפייני ההנחיה של דף ASPX שאינו משתמש בדף ראשי.

כשפקד המשתמש פתוח בחלון המסמך של VWD, אתה יכול ליצור דפים ולהסתייע בכל הכלים שבהם השתמשת בשבעת הפרקים הקודמים. תוכל להשתמש בארגז הכלים כדי לגרור פקדים ב- Markup View וב- Design View, להשתמש בחלונות CSS כדי לשנות את המראה והתחושה של פקד המשתמש, וגם להשתמש בטבלת המאפיינים כדי לשנות את המאפיינים של פקדים בדפים. תוכל גם לכתוב קוד שמגיב לאירועים שמעורר הפקד.

כדי לנסות זאת בעצמך, תלמד בתרגיל הבא כיצד ליצור את פקד המשתמש הראשון שלך. בתרגיל מאוחר יותר תלמד כיצד להשתמש בפקד בדפי ASPX באתר.



תרשים 8-1

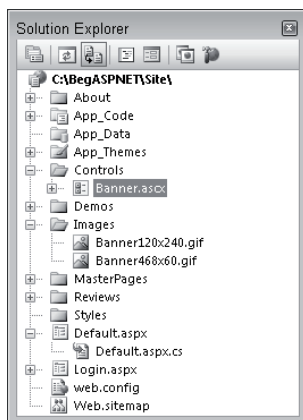
נסה זאת יצירת פקד משתמש

בתרגיל זה תיצור פקד משתמש בסיסי שמציג באנר אנכי יחיד תוך שימוש בפקד `<asp:Image>`. בתרגילים מאוחרים יותר בפרק תראה כיצד להשתמש בפקד זה וכיצד להוסיף תמונה אחרת (אופקית). אחר כך תוסיף לפקד מעט "תבונה" כדי שתוכל לקבוע איזו משתי התמונות להציג בזמן ביצוע.

לצורך התרגיל אתה זקוק לשתי תמונות שמייצגות באנרים - אחת במצב פורטרט (אנכית) בעלת ממדים של 120x240 פיקסלים בקירוב, והשנייה במצב אופקי עם ממדים של 486x60 פיקסלים בקירוב. (שתי התמונות נמצאות כחלק מקוד המקור לפרק זה), אבל אתה יכול ליצור

אותן גם בעצמך, אינך צריך להקפיד על הגודל המדויק של התמונות; כל עוד מידותיהן קרובות לממדים הנזכרים לעיל, לא תהיה בעיה.

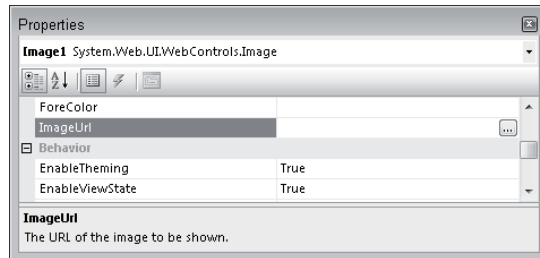
1. פתח את האתר Planet Wrox ב-VWD.
2. צור תיקייה בשם Controls בשורש האתר (אם לא עשית זאת כבר). ניתן להציב פקדים בכל מקום בהיררכיית האתר, אבל הצבתם בתיקייה נפרדת מקלה כאשר צריך למצוא ולנהל אותם.
3. צור תיקייה בשם Images בשורש האתר (אם לא עשית זאת עד כה).
4. פתח את התיקייה Resources (עבור פרק זה) על ידי סייר החלונות (אם עקבת אחר ההוראות בהקדמה, תמצא את התיקייה בכתובת C:\BegASPNET\Resources\Chapter 08). אם לא עשית זאת עד כה, תוכל להוריד את הקבצים מאתר האינטרנט www.wrox.com. גרור את הקובץ Banner120x240.gif ואת הקובץ Banner48x60.gif מסייר החלונות אל התיקייה Images שיצרת בצעד הקודם. אם אתה משתמש בתמונות משלך, גרור אותן לתיקייה Images וקרא להן בשמות הנזכרים לעיל.
5. לחץ לחיצה ימנית על התיקייה Controls ובחר Add New Item. בתיבת הדו-שיח שתפתח לחץ Web User Control, כפי שמוצג בתרשים 8-1. בחר בשפת התכנות שלך. ודא שבחרת גם Place Code In Separate File (אחסון הקוד הקובץ נפרד). קבע את שם הקובץ ל- Banner.ascx, ולחץ Add כדי להוסיף את הפקד לאתר. הפקד אמור להיפתח בחלון המסמך ב- Markup View. סייר הפתרון צריך להיראות כעת כמו תרשים 8-2.



תרשים 8-2

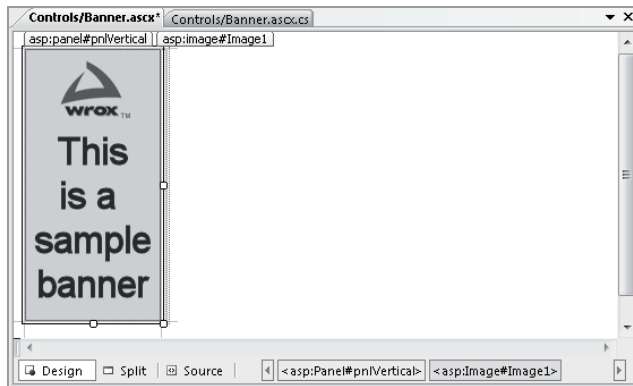
6. העבר את פקד המשתמש ל- Design View, וגרור Panel מהקטגוריה Standard של ארגז הכלים אל משטח העיצוב. על ידי שימוש בטבלת המאפיינים, שנה את ID של Panel ל-pnlVertical. ייתכן כי לרגע קצר תראה את תיבת הדו-שיח Refactoring שבודקת אם יש בקוד שלך התייחסויות כלשהן ל-ID הקודם (Panel1), ואם כן, היא מעדכנת אותן ל-pnlVertical. מכיוון שזה עתה הוספת את Panel, אין כרגע קוד שצריך לעדכן, ואתה יכול להתעלם מתיבת הדו-שיח.

7. גרור פקד Image מארגז הכלים לתוך פקד ה-Panel. לחץ לחיצה על הפקד Image כדי לסמנו, ואז פתח את טבלת המאפיינים. אתר את המאפיין imageUrl ולחץ על לחצן האליפסות שמוצג בתרשים 8-3.



תרשים 8-3

- דפדף אל התיקייה Images, בחר בתמונה Banner120x240.gif, ולחץ OK כדי להוסיף אותה לפקד המשתמש. Design View נראה כעת כמו תרשים 8-4.



תרשים 8-4

8. אתר את המאפיין AlternateText בטבלת המאפיינים והקלד **This is a sample banner**. כמה מהדפדפנים, כמו Firefox, מציגים את הטקסט החלופי כאשר לא ניתן להציג את התמונה בצורה נכונה. דפדפנים אחרים, כמו Internet Explorer, מראים את הטקסט החלופי כתיאור עבור התמונה כשאתה מרחף עליה עם העכבר.
9. עבור ל- Markup View ואם יש ערכים למאפייני הפקד Panel: Height ו-Width שהוספו לו כברירת מחדל כשגררת אותו על הדף, הסר אותם.
10. עטוף את Image באלמנט `<a>` הרגיל והגדר את המאפיין href ל- `http://p2p.wrox.com`. הגדר את מאפיין target של אלמנט `<a>` ל- `_blank` כדי לכפות על הדפדפן לפתוח את הדף בחלון חדש כשילחצו על התמונה. כשתסיים, הקוד של פקד המשתמש יהיה דומה לקוד שלהלן, פרט למאפיין Language שייטכן שהגדרת אותו ל-VB, ול- `AutoEventWireup` שברירת המחדל שלו ב-VB.NET היא `False`:

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="Banner.ascx.cs"
    Inherits="Controls_Banner" %>
<asp:Panel ID="pnlVertical" runat="server">
    <a href="http://p2p.wrox.com" target="_blank">
        <asp:Image ID="Image1" runat="server" AlternateText="This is a sample banner"
            ImageUrl="~/Images/Banner120x240.gif" />
    </a>
</asp:Panel>
```

11. שמור את השינויים על ידי הקשת Ctrl+S. סגור את קובץ פקד המשתמש על ידי הקשת Ctrl+F4.

כיצד זה פועל

חווית עיצוב פקדי משתמש ב- Visual Web Developer IDE זהה לזו של עיצוב דפים. אתה יכול להשתמש במנגנון גרור ושחרר, בארגז הכלים, ב- Markup View וב- Design View. הדבר מקל על העבודה עם פקדי משתמש מכיוון שניתן להשתמש בכל הכלים המוכרים שאתה משתמש בהם גם לפיתוח דפים.

הפקד שיצרת זה עתה מציג תמונה יחידה עטופה באלמנט עוגן. בסעיף הבא תראה איך להוסיף את הבאנר לדף ראשי, כך שיוצג בסרגל הצד <div> של כל דף באתר. בסעיפים מאוחרים יותר בפרק תלמד כיצד להוסיף את התמונה האחרת, שמשתמשת להצגת באנר אופקי בדפי תוכן שונים.

הוספת פקדי משתמש לדף תוכן או לדף ראשי

כדי להשתמש בפקדי משתמש בדף תוכן, או בדף ראשי, או בפקד משתמש אחר, עליך לבצע שתי פעולות. ראשית, צריך לרשום (register) את הפקד על ידי הוספת ההנחיה @ Register לדף או לפקד שברצונך להציב בהם את פקד המשתמש. הפעולה השנייה כרוכה בהוספה של תגיות פקד המשתמש לדף, ובהגדרת כמה מאפיינים שלו לפי בחירה.

ההנחיה @ Register טיפוסית של פקד משתמש תיראה כך:

```
<%@ Register src="../Controls/Banner.ascx" tagname="Banner" tagprefix="uc1" %>
```

ההנחיה מכילה שלושה מאפיינים חשובים המתוארים בטבלה הבאה.

מאפיין	תיאור
src	המאפיין מצביע על פקד המשתמש שברצונך להשתמש בו. כדי להקל על העברת דפים בשלב מאוחר יותר, אפשר להשתמש בתחביר ~ כדי להצביע על הפקד משורש היישום.

מאפיין	תיאור
tagname	שם התגית שמשמשים בה בהכרזת הפקד. אתה חופשי בקביעת השם, אבל בדרך כלל הוא יהיה זהה לשם הפקד.
tagprefix	המאפיין מציין את התחילית של tagname שמשמשים בה בהכרזת הפקד. כפי ש-ASP.NET משתמשת בתחילית asp כדי לפנות לפקדים שלה, כך אתה צריך לספק תחילית עבור פקדי המשתמש שלך. ברירת המחדל היא התחילית uc כשאחריה מספר סדרתי, אבל ביכולתך לשנות זאת בהתאם לטעמך - לדוגמה, לשם החברה שלך או קיצור מותאם.

כשהפקד רשום, תוכל להוסיפו על ידי שימוש במבנה tagprefix:tagname, בדומה לדרך שבה אתה מוסיף פקדי שרת רגילים לדף. כשנתונה ההנחיה Register @ עבור פקד המשתמש Banner.ascx, יש צורך בסימני העריכה הבאים כדי להוסיף את הפקד לדף:

```
<uc1:Banner ID="Banner1" runat="server" />
```

זהו הקוד המינימלי הדרוש לפקד משתמש בדף. שים לב שהפקד מוגדר על ידי שילוב של tagprefix ושל tagname. שני המאפיינים האחרים - ID ו-runat - הם מאפיינים רגילים שקיימים ברוב הפקדים בדף ASP.NET.

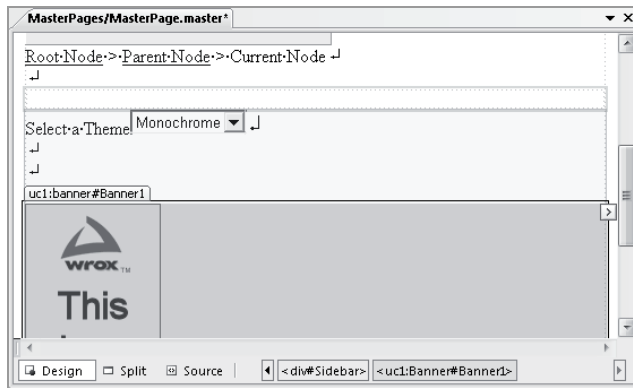
למרבה המזל, ברוב המקרים אינך צריך להקליד את כל הקוד הזה בעצמך. VWD מוסיפה את הקוד הנדרש באופן אוטומטי בזמן שאתה גורר פקד משתמש מסייר הפתרון אל דף שנמצא ב- Design View. התרגיל הבא מדגים כיצד זה פועל.

נסה זאת הוספת פקד משתמש לדף

בתרגיל זה תוסיף את פקד המשתמש Banner.ascx לדף הראשי, כך שבכל דף באתר יוצג באנר בשטח סרגל הצד.

1. פתח את MasterPage.master מהתיקייה MasterPages, ועבור ל- Design View.
2. אתר את פקד הרשימה הנפתחת שמאפשרת לבחור ערכת נושא, הצב את הסמן מייד אחר הרשימה הנפתחת והקש Enter שלוש פעמים, כדי ליצור מעט רווח.
3. גרור את הקובץ Banner.ascx מסייר הפתרון אל המקום הריק שיצרת זה עתה. Design View מתעדכן ונראה עכשיו כמו בתרשים 5-8.

אם Design View אינו נראה כך, אלא נראה הרבה יותר קרוב לצורה שבה הופיע הקובץ בדפדפן, ייתכן ש-StyleSheetTheme עדיין מוגדר בקובץ web.config. לחילופין, ייתכן שבחרת יותר, או פחות, אפשרויות בתפריט View ← Visual Aids או בתפריט View ← Formatting Marks.



תרשים 8-5

4. עבור לתצוגת Markup View ואתר את ההנחיה @ Register בראש הקובץ. שנה את שתי הנקודות שצמודות לתכונה src לתו ~.

```
<%@ Register src="~/Controls/Banner.ascx" tagname="Banner" tagprefix="uc1" %>
```

5. שמור את השינויים שנערכו בדף הראשי, סגור אותו, ואז לחץ לחיצה ימנית על הקובץ Default.aspx שבשורש האתר, ובחר View in Browser.

6. הבאנר מוצג כעת מתחת לרשימה הנפתחת. עבור לערכת הנושא האחרת ותראה שמופיע אותו באנר. כשתלחץ על הבאנר ייפתח חלון חדש שיעביר אותך לאתר שהיית מקושר אליו בתרגיל הקודם.

כיצד זה פועל

כשגררת את פקד המשתמש על פני משטח העיצוב של הדף הראשי, ביצעה VWD שתי משימות: האחת, היא הוסיפה את ההנחיה @ Register כדי לומר לדף היכן לחפש את פקד המשתמש. השנייה, היא הוסיפה את הכרזת הפקד מייד תחת הרשימה הנפתחת.

כשהדף נטען, ASP.NET runtime קורא את הכרזת הפקד ומזריק את הפלט של הפקד במקום שצוין בהכרזה. בדוגמה זו, ה-Panel, האלמנט <a> וה-Image מוכנסים באזור סרגל הצד של הדף. כשתסתכל בקוד HTML של הדף בדפדפן תראה את הקוד הבא:

```
</select>
<br /><br /><br />
<div id="ctl00_Banner1_pnlVertical">
  <a href="http://p2p.wrox.com" target="_blank">
    
  </a>
</div>
```

<asp:Panel> הפך לאלמנט <div> של HTML, ו- <asp:Image> הפך לאלמנט . מכיוון שאלמנט <a> הוגדר בפקד המשתמש ב-HTML פשוט, הוא מופיע בדיוק כפי שכתבת אותו.

שים לב שהזיהוי (id) של Panel שונה מ-pnlVertical ל-ctl00_Banner1_pnlVertical. הדבר נחוץ כדי שהתגית <div> תקבל תכונת id ייחודית עבור צד הלקוח. תכונה זו משמשת בכתיבת תסריטים בצד הלקוח. דבר דומה קרה ל-id של האלמנט . תלמד על כך עוד בסעיף מאוחר יותר בפרק.

בדרך כלל, כששמים אלמנט בתוך אלמנט <a> כדי לקשרו, הדפדפן משרטט גבול מסביב לתמונה. ככלל, הגבול הוא בצבע כחול עבור קישורים שלא ביקרו בהם, ובצבע ארגמן עבור קישורים שביקרו בהם. הדבר אינו נחוץ עם <asp:Image> מכיוון שכברירת מחדל, התמונה מרונדרת עם סגנון כלול שמגדיר את הגבול ל-0 פיקסלים. כשתרצה להסיר את הגבול מתמונות מקושרות שנוצרו על ידי התגית בדפים אחרים, תוכל להשתמש בקוד CSS הבא:

```
img
{
    border: none;
}
```

קוד זה יסיר את הגבולות מכל התמונות המקושרות באתר. כשמוסיפים פקד משתמש לדף, VWD מוסיפה כברירת מחדל הפניה לפקד באמצעות נתיב יחסי. בתרגיל הקודם הכיל נתיב זה שתי נקודות (..) כדי לציין את תיקיית ההורה, אחר כך את התיקייה Controls ולבסוף, את שם הפקד:

```
<% Register src=" ../Controls/Banner.ascx" tagname="Banner" tagprefix="uc1" %>
```

שינוי שתי הנקודות לתו ~, מקל על העברת הדפים באתר, מכיוון שהתכונה src מצביעה כעת תמיד לתיקייה Controls שבשורש היישום, מבלי להתחשב במיקום של הדף שמשמש בפקד.

תחביר ~ אמנם מקל מעט על ניהול הדפים שכוללים פקדי משתמש, אבל קיימת דרך קלה עוד יותר לרישום פקדי משתמש ברחבי האתר.

רישום פקדי משתמש ברחבי האתר

כשהצפי הוא שדפי תוכן נפרדים באתר ישתמשו לעתים קרובות בפקד מסוים, כמו הבאנר שראינו בתרגילים הקודמים, ניתן לרשום את הפקד באופן גלובלי ברמת האתר בקובץ web.config. בדרך זו הוא יהפוך לזמין ברחבי האתר מבלי שיהיה צורך לרשום אותו בכל דף בנפרד. התרגיל הבא מראה כיצד לעשות זאת.

נסה זאת רישום פקדי משתמש בקובץ web.config

בתרגיל הבא תרשום את פקד המשתמש Banner.ascx בקובץ web.config. אחר כך תוכל להסיר את ההנחיה @ Register מהדף הראשי, מכיוון שלא יהיה בה עוד צורך. הוספת אותו פקד משתמש לדפים אחרים לאחר שינוי הקובץ web.config, לא תוסיף עוד את ההנחיה @ Register לדפים.

1. פתח את הקובץ web.config משורש האתר.

2. גלול מעט מטה ואתר את האלמנט <pages> שהשתמשת בו בפרק 6, כדי להחיל את ערכת הנושא.

3. הוסף את הקוד המוצג להלן לאלמנט <controls> שנמצא תחת <pages>:

```
<pages theme="Monochrome">
```

```
...
```

```
<controls>
```

```
<add tagPrefix="Wrox" tagName="Banner" src="~/Controls/Banner.ascx" />
```

4. שמור את השינויים וסגור את הקובץ.

5. פתח שוב את הדף הראשי בתצוגת Markup View ואתר את הפקד Banner בשטח סרגל הצד. שנה את uc1 ל-Wrox:

```
<Wrox:Banner ID="Banner1" runat="server" />
```

אם ההכרזה על פקד המשתמש היא בעלת תגית סגירה משל עצמה, חובה לעדכן גם זאת:

```
<Wrox:Banner ID="Banner1" runat="server"></Wrox:Banner>
```

6. גלול את הדף הראשי עד קצהו, והסר כל שורה שמכילה את ההנחיה @ Register.

7. שמור וסגור את הדף הראשי.

8. פתח שוב את Default.aspx בדפדפן. שים לב שהבאנר עדיין נמצא בשטח סרגל הצד. אם תקבל הודעת שגיאה, ודא שהוספת לקובץ web.config את הקוד הנכון. כמו כן ודא שבהכרזת הפקד בדף הראשי אכן שינית את uc1 ל-Wrox.

כיצד זה פועל

כשההנחיה @ Register אינה קיימת בדף הראשי, ASP.NET runtime סורק את הקובץ web.config כדי למצוא את הפקדים שנרשמו בו. כשהוא מוצא את הרישום של הפקד Wrox:Banner, מסוגל runtime למצוא את הקובץ תוך שימוש במאפיין src, ומוסיף את פלט הקובץ להיררכית הדף.

הוספת רישום הפקד לקובץ web.config מקלה מאוד על העברת הפקד או שינוי שמו. במקום שיהיה צורך למצוא ולהחליף את ההנחיה @ Register בכל הדפים שמשתמשים בפקד, הדבר היחיד שצריך לשנות הוא הרישום בקובץ web.config. השינוי בקובץ זה גורם לכך שכל הדפים שמשתמשים בפקד המשתמש ימצאו באופן אוטומטי את המיקום או את השם החדשים.

אזהרות לגבי פקדי משתמש

כבר ראית קודם ש- ASP.NET runtime שינה את ID הלקוח של הפקד Panel שהוספת לדף. במקום לקבל את האלמנט <div> עם id שמוגדר ל-pnlVertical קיבלת את מזהה ה-id הבא:

```
<div id="ctl00_Banner1_pnlVertical">
```

```
...
```

```
</div>
```

במקרים רבים הדבר אינו בעייתי, מכיוון שלעיתים קרובות אין צורך ב-id של הלקוח. עם זאת, אם תצטרך לגשת לפקד זה מ-JavaScript או מ-CSS של צד הלקוח, חשוב להבין מדוע חל שינוי ב-id זה.

מטבע עיצובם, כל האלמנטים בדף HTML חייבים להיות ייחודיים. אין דרישה שלכל אלמנט יהיה id, אבל אם קובעים קוד זיהוי, חייבים לוודא שהוא ייחודי. כדי למנוע התנגשויות בקוד HTML הסופי של הדף, ASP.NET מוודאת שכל אלמנט של צד-לקוח מקבל id ייחודי על ידי הוספת שם המזהה של הפקד המכיל אותו כתחילית. כל האלמנטים שנמצאים תחת אותו פקד המכיל אותם חייבים להיות בעלי קודי זיהוי (IDs) ייחודיים. כשתנסה להוסיף פקד שאין לו ID ייחודי, תתריע על כך Visual Web Developer IDE. לדוגמה, אם תנסה להוסיף לפקד משתמש Panel שני עם קודי זיהוי בשם pnlVertical, תקבל הודעת שגיאה. אבל מה יקרה אם תשכן שני פקדי Banner באותו הדף? יש אפשרות שתסיים עם שני אלמנטים <div> של לקוח, שלשניהם קודי זיהוי של pnlVertical. כדי להימנע מבעיה זו, מצמידה ASP.NET לכל אלמנט תחילית שהיא ID של מכולת השמות הקרובה ביותר. עבור Panel שבתוך פקד משתמש פירושו שהתחילית תהיה Banner1 - ID צד השרת של פקד המשתמש בדף הראשי. ID חדש זה מקבל אז תחילית שהיא הזיהוי של הדף הראשי - ct100. זיהוי זה נוצר באופן אוטומטי על ידי ASP.NET runtime, אבל באפשרותך לדרוס אותו על ידי הגדרה מפורשת של קוד הזיהוי באירוע Page_Init של הדף הראשי:

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Me.Init  
    Me.ID = "Master"
```

C#

```
protected void Page_Init(object sender, EventArgs e)  
{  
    this.ID = "Master";
```

ביכולתך להשתמש ב-ClientID של פקד כדי לקבל את הזיהוי של צד הלקוח המלא שלו. קטע הקוד הבא מראה כיצד להציג את ClientID של הפקד Panel בפקד Label שבתוך פקד המשתמש Banner.ascx:

VB.NET

```
Label1.Text = pnlVertical.ClientID
```

C#

```
Label1.Text = pnlVertical.ClientID;
```

עם קוד זה, המאפיין Text של הפקד Label יכיל את ct100_Banner1_pnlVertical, שהוא הזיהוי של צד הלקוח של Panel במקרה שלא הגדרת את ה-ID של הדף הראשי במפורש. עם ID מפורש כמפורט בדוגמה הקודמת, Label יציג את Master_Banner1_pnlVertical. עם ID מפורש קל יותר לחזות את קוד הזיהוי הסופי של אלמנט HTML של צד הלקוח. דבר זה בתורו מקל על הפנייה לאלמנטים אלה ב-JavaScript או ב-CSS. בפרקים מאוחרים יותר תלמד איך להשתמש בזה.

שים לב: מכיוון ש- ASP.NET runtime יכול לשנות את תכונות id של צד הלקוח באלמנטים של HTML, עלולים להתעורר קשיים בשימוש ב-selectors מסוג CSS ID בעת פנייה לאלמנטים. הדרך הקלה ביותר לתקן זאת היא להשתמש ב-selectors מסוג Class ולא מסוג ID. לחילופין, באפשרותך להשתמש ב-ClientID של הפקד.

למרות שפקד המשתמש הנוכחי הקל על הצגת באנר במקומות שונים באתר, הוא עדיין אינו מספיק מתוחכם. כל מה שביכולתו לעשות זה להציג תמונה מקושרת בלבד. כדי לשפר את השימוש בו, אפשר לשנות את התנהגותו כשהוא מופיע בדפים שונים באתר.

הוספת לוגיקה לפקדי משתמש

למרות שהשימוש בפקדים עבור תכנים חוזרים כבר שימושי למדי, הרי שפקדים נעשים שימושיים יותר כשמוסיפים להם לוגיקה מותאמת. על ידי הוספת מאפיינים ציבוריים או הוספת שיטות לפקד משתמש ניתן להשפיע על התנהגותו בזמן ביצוע. כשמוסיפים מאפיין לפקד משתמש, הרי שבאופן אוטומטי הוא נעשה זמין ב-IntelliSense ובטבלת המאפיינים של הפקד בדף שפועלים בו. הדבר מאפשר לשנות בקלות התנהגות מתוך קובץ חיצוני כמו דף.

כדי להוסיף מאפיינים או שיטות לפקד משתמש, יש להוסיף אותם לקוד ברקע של הפקד. המאפיינים שתוסיף יכולים ללבוש צורות שונות. בצורתו הפשוטה ביותר נראה מאפיין בדיוק כמו המאפיינים שראית בפרק 5. עבור תסריטים מתקדמים יותר, עליך להוסיף **מאפיינים מסוג ViewState** (ViewState properties) שמסוגלים לשמור את מצבם על פני postbacks שונים. בשני התרגילים הבאים תראה כיצד ליצור את שני סוגי המאפיינים.

יצירת סוגי נתונים משלך עבור מאפיינים

כדי להפוך את פקד הבאנר ליותר שימושי, תוכל להוסיף לו תמונה שנייה שתוצג כבאנר אופקי. על ידי הוספת מאפיין לפקד המשתמש, תוכל לקבוע אם להציג את התמונה האנכית או את האופקית. לשם כך אפשר ליצור מאפיין נומרי מסוג System.Byte, כשהערך 0 מציין תמונה אנכית ו-1 מציין תמונה אופקית, למשל. עם זאת, תתקשה לזכור מה כל מספר מייצג. אפשר להקל במקצת על ידי יצירת מאפיין String שמקבל את הערכים Horizontal ו-Vertical. החיסרון כאן הוא שלא ניתן לבדוק מחרוזות בזמן הפיתוח. לכן אתה עלול לסיים עם שגיאת איות שתוצאתה תהיה שגיאה או הצגת הבאנר הלא נכון. במקום זה אתה יכול ליצור סוג נתונים משלך בצורה של **סוג ספירה** (ספירה, enumeration או enum בקיצור). עם enums אתה מקצה מספרים למחרוזות טקסט ייחודיות לאדם. המתכנתים משתמשים בטקסט הקריא, בעוד שלמעשה יש שימוש בערך המספרי. קטע הקוד הבא מציג דוגמה בסיסית של סוג ספירה:

VB.NET

```
Public Enum Direction
    Horizontal
    Vertical
End Enum
```

C#

```
public enum Direction
{
    Horizontal,
    Vertical
}
```

עם הוראות enums אלו, המהדר מקצה באופן אוטומטי לחברים Horizontal ו-Vertical ערכים מספריים, החל מ-0 ומעלה. באפשרותך גם להגדיר את הערכים המספריים במפורש:

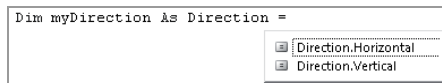
VB.NET

```
Public Enum Direction
    Horizontal = 0
    Vertical = 1
End Enum
```

C#

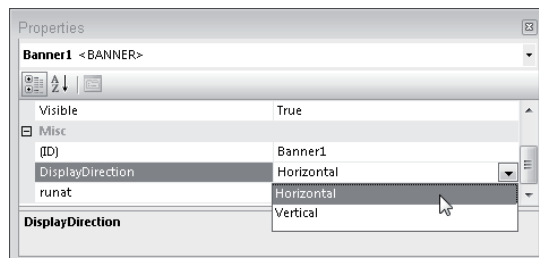
```
public enum Direction
{
    Horizontal = 0,
    Vertical = 1
}
```

הדבר הטוב ב-enums הוא, שכתוצאה מכך תוכל להשתמש ב-IntelliSense בקבצי קוד, בטבלת המאפיינים (properties grid) ואפילו בעורך הקוד של פקדי המשתמש. תרשים 8-6 מראה כיצד מופיע IntelliSense בקבצי קוד של VB.NET.



תרשים 8-6

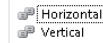
בתרשים 8-7 רואה את אותה רשימה עם ערכים מ-enum שברשת המאפיינים.



תרשים 8-7

בתרשים 8-8 רואה את אותה רשימה מופיעה בתצוגת Markup View עבור המאפיין DisplayDirection של פקד משתמש.

```
<br />
<Wrox:Banner ID="Banner1" runat="server" DisplayDirection="Horizontal" />
```



תרשים 8-8

בדיוק כמו עם קבצי קוד אחרים, כמו מחלקות, גם כאן עליך להציב את הגדרת ה-enum בקובץ תחת התיקייה App_Code. אם יש יותר מרשימה אחת כזו, אפשר לאחסן את כולן בקובץ אחד או ליצור קובץ נפרד עבור כל רשימת enum.

סוגי רשימות (enums) הן מעולות עבור רשימות פשוטות וקצרות. הן עוזרות לך למצוא את הפריט הנכון במהירות מבלי שתצטרך לשנן "מספרי פלא" כמו 0 או 1, אלא להשתמש במחרוזות טקסט קריאות.

בתרגיל הבא תלמד איך ליצור סוג רשימת enum ולהשתמש בה בפקד המשתמש Banner.

נסה זאת יצירת פקדי משתמש מתוחכמים

בתרגיל זה תוסיף באנר שני לפקד המשתמש. באנר זה מוצג כתמונה אופקית בתוך Panel משלה. כדי למנוע ששני הבאנרים יופיעו בו-זמנית, עליך להוסיף מאפיין שיקבע איזה באנר להציג. דפים שמשתמשים בפקד יכולים להגדיר את הבאנר הרצוי.

1. צור סוג רשימה enum שמכיל שני פריטים עבור שני הכיוונים: אנכי ואופקי. לשם כך, לחץ לחיצה ימנית על התיקייה App_Code ובחר Add New Item. הוסף קובץ מחלקה בשם Direction ובחר בשפת התכנות שלך.
2. ברגע שהקובץ ייפתח, נקה את תוכנו והוסף לו את הקוד הבא:

VB.NET

```
Public Enum Direction
    Horizontal
    Vertical
End Enum
```

C#

```
public enum Direction
{
    Horizontal,
    Vertical
}
```

3. שמור וסגור את הקובץ.
4. פתח את הקוד ברקע של פקד המשתמש Banner.ascx והוסף לו את המאפיין הבא. כדי לסייע ביצירת המאפיין כוללת VWD מראש קטע קוד שימושי. כדי להפעיל את הקוד ב-VB.NET הקלד Property (ללא רווח אחרי המילה) והקש Tab. ב-C# הקלד prop ואז הקש Tab פעמיים. בשני המקרים תוסיף עבורך VWD את מבנה הקוד עבור מאפיין;

מאפיין רגיל עבור VB.NET ומאפיין אוטומטי עבור C#. באפשרותך להקיש שוב Tab כדי לעבור משדה לשדה, כשבכל פעם אתה מקליד את סוג הנתונים הנכון: משתנה או שם מאפיין. השלם את הקוד כך שייראה כמו זה שלהלן:

VB.NET

```
Private _displayDirection As Direction = Direction.Vertical

Public Property DisplayDirection() As Direction
    Get
        Return _displayDirection
    End Get
    Set(ByVal value As Direction)
        _displayDirection = value
    End Set
End Property
```

C#

```
public Direction DisplayDirection {get; set;}
```

בשתי השפות הקפד להוסיף את המאפיין מחוץ לשיטה Page_Load, אבל לפני End Class (VB.NET) או לפני הסוגר המסולסל של המחלקה (C#).

שים לב ששם המאפיין הוא DisplayDirection והוא מסוג Direction. כמו כן, שים לב שדוגמת C# משתמשת במאפיין אוטומטי, שראית אותו לראשונה בפרק 5.

5. פתח את Banner.ascx ב-Markup View, העתק את כל הקוד <asp:Panel> המוצלל והדבק אותו מייד מתחת ל-Panel הקיים. קרא ל-Panel בשם pnlHorizontal והגדר את src של התמונה ל- ~/Images/Banner468x60.gif. הקוד שלך אמור להיראות כעת כך:

```
</asp:Panel>
<asp:Panel ID="pnlHorizontal" runat="server">
    <a href="http://p2p.wrox.com" target="_blank">
        <asp:Image ID="Image2" runat="server" AlternateText="This is a sample banner"
            ImageUrl="~/Images/Banner468x60.gif" />
    </a>
</asp:Panel>
```

6. חזור לקוד ברקע של הפקד, והוסף את הקוד הבא למטפל האירוע Page_Load. ב-C# המטפל אמור כבר להיות שם. כדי לקבוע את המטפל ב-VB: מהרשימה הנפתחת השמאלית שבראש חלון המסמך, בחר (Page Events), ומהרשימה הנפתחת הימנית בחר Load. לחילופין, לחץ לחיצה כפולה על פקד המשתמש ב- Design View.

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Me.Load
    pnlHorizontal.Visible = False
    pnlVertical.Visible = False
```

```

Select Case DisplayDirection
    Case Direction.Horizontal
        pnlHorizontal.Visible = True
    Case Direction.Vertical
        pnlVertical.Visible = True
End Select

```

```
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
```

```

{
    pnlHorizontal.Visible = false;
    pnlVertical.Visible = false;

    switch (DisplayDirection)
    {
        case Direction.Horizontal:
            pnlHorizontal.Visible = true;
            break;
        case Direction.Vertical:
            pnlVertical.Visible = true;
            break;
    }
}

```

7. שמור וסגור את שני הקבצים שמרכיבים את פקד המשתמש, מכיוון שסיימת את השימוש בהם בשלב זה.

8. פתח את קובץ הדף הראשי פעם נוספת ב- Markup View, ואתר את הכרזת פקד המשתמש. הוסף את המאפיין DisplayDirection מייד לאחר המאפיין runat="server". המאפיין שהוספת קובע את סוג התמונה הנכון.

```
<Wrox:Banner ID="Banner1" runat="server" DisplayDirection="Horizontal" />
```

IntelliSense יסייע לך לבחור מהרשימה את DisplayDirection הנכון. אם IntelliSense אינו מופיע, המתן כמה שניות עד ש-VWD תשתלט על כל השינויים. לחילופין, סגור את כל הקבצים ופתח שוב את הדף הראשי.

9. שמור את כל השינויים ובקש את הדף Default.aspx בדפדפן. שים לב ששטח סרגל הצד הימני מכיל כעת את התמונה האופקית, אבל העיצוב נשבר מעט משום שהתמונה רחבה מדי עבור שטח סרגל הצד.

10. חזור לדף הראשי ושנה את DisplayDirection מ-Horizontal ל-Vertical. שמור את השינויים ורענן את הדף בדפדפן. סרגל הצד אמור להציג כעת את הבאנר האנכי, כפי שמוצג בתרשים 8-9.



תרשים 8-9

11. פתח את הדף AboutUs.aspx מהתיקייה About שב- Markup View ב-VWD. אם הקובץ אינו קיים, צור אותו תחילה. בתוך הפקד Content אשר ערך המאפיין ContentPlaceHolder שלו הוא cpMainContent, הוסף מעט טקסט שמתאר את הארגון שלך ואת הסיבה לכך שיצרת את האתר. עבור ל- Design View וגרור את הפקד Banner.ascx מסייר הפתרון אל משטח העיצוב, מייד מתחת לטקסט שהוספת זה עתה.
12. ב- Design View, בחר בפקד, פתח את טבלת המאפיינים שלו והגדר את DisplayDirection ל-Horizontal.
13. שמור את כל השינויים. הקש Ctrl+F5 כדי לפתוח את הדף About Us בדפדפן. פרט לבאנר בסרגל הצד הימני, אתה אמור לראות כעת גם את הבאנר האופקי מופיע בשטח Content.

כיצד זה פועל

המאפיין שנקרא DisplayDirection מקנה לפקד המשתמש התנהגות נוספת. דפים שמשתמשים בפקד אמורים להגדיר כעת את DisplayDirection כך:

```
<Wrox:Banner ID="Banner1" runat="server" DisplayDirection="Horizontal" />
```

כשמופע הפקד נוצר על ידי ASP.NET runtime, מוקצה למאפיין הערך שקבעת בהכרזת הפקד. המאפיין בתורו מאחסן את הערך בשדה הפרטי displayDirection (VB.NET), או במשתנה גיבוי שנוצר אוטומטית (C#). למידע נוסף על מאפיינים אוטומטיים פנה לפרק 5.

כשהדף נטען, מתבצע הקוד הבא:

VB.NET

```
pnlHorizontal.Visible = False
pnlVertical.Visible = False

Select Case DisplayDirection
    Case Direction.Horizontal
        pnlHorizontal.Visible = True
    Case Direction.Vertical
        pnlVertical.Visible = True
End Select
```

C#

```
pnlHorizontal.Visible = false;
pnlVertical.Visible = false;

switch (DisplayDirection)
{
    case Direction.Horizontal:
        pnlHorizontal.Visible = true;
        break;
    case Direction.Vertical:
        pnlVertical.Visible = true;
        break;
}
```

הדבר הראשון שקוד זה מבצע הוא הסתרת שני פקדי ה-Panel שמכילים את התמונות. אחר כך הוא משתמש בבלוק הבחירה Select Case / switch (בהתאם לשפת התכנות) כדי לקבוע איזו תמונה להציג. כש-DisplayDirection שווה ל-Horizontal, המאפיין Visible של pnlHorizontal נקבע ל-True. אותו עיקרון חל גם על Vertical.

יישום מאפיין ViewState

פרט למאפיין DisplayDirection, יש מאפיין שימושי נוסף לפקד משתמש, וזהו ה-URL, הכתובת שהבאנר מקושר אליה. בסעיף הבא תראה כיצד ליישם זאת, ותלמד ליצור מאפיין ViewState בשם NavigateUrl שמסוגל לשרוד על פני פניות הלוח ושוב מהשרת לדפדפן ובחזרה (postbacks).

נסה זאת **יישום המאפיין NavigateUrl**

כדי להיות מסוגל להגדיר מקוד השרת את ה-URL שהמשתמש מופנה אליו, עליך להיות מסוגל לגשת לתגית העוגן שמוגדרת בסימני העריכה של הפקד. לשם כך עליך לקבוע לה ID ולהגדיר לה את המאפיין runat="server" כדי שתהיה נגישה עבורך בצד השרת. לחילופין, באפשרותך לשנות את התגית הפשוטה <a> לתגית המקבילה שלו בצד השרת: <asp:HyperLink>. כדי שתוכל להגדיר לפקד המשתמש את המאפיין החדש NavigateUrl בתכנות קוד בצד השרת, ולהבטיח שהוא ישרוד postbacks, עליך ליישם מאפיין ViewState. במהלך התרגיל יובהר מדוע יש צורך במאפיין ViewState.

התרגיל מורכב משני חלקים. בשלושת הצעדים הראשונים תשנה את הדף About Us באופן שיגדיר את DisplayDirection של הפקד משתמש Banner בתכנות קוד. אחר כך תגרום ל-postback ותוכל לראות שהערך של הכיוון אבד, משום שהמאפיין אינו שומר את מצבו. החלק השני של התרגיל מראה כיצד ליישם את המאפיין NavigateUrl שמסוגל לשמור את מצבו.

1. פתח את הקוד ברקע של AboutUs.aspx והוסף את הקוד הבא לאירוע Page_Load. אם המטפל אינו שם עדיין, עבור ל- Design View ולחץ לחיצה כפולה במקום כלשהו בשטח האפור של הדף.

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Me.Load  
    If Not Page.IsPostBack Then  
        Banner1.DisplayDirection = Direction.Vertical  
    End If  
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)  
{  
    if (!Page.IsPostBack)  
    {  
        Banner1.DisplayDirection = Direction.Vertical;  
    }  
}
```

ב- Markup View של הדף, ודא ש-ID של פקד המשתמש Banner הוא Banner1, או שתעדכן את הקוד בהתאם.

2. עבור לתצוגה Design View והוסף פקד Button על ידי גרירתו מהקטגוריה Standard של ארגז הכלים אל הפקד Banner.ascx. הוא אמור להופיע מעל או מתחת לבאנר. אין צורך לכתוב קוד כלשהו עבור האירוע Click של הפקד Button.
3. שמור את הדף ופתח אותו בדפדפן. הקוד ב-Page_Load גורם לכך שכאשר הדף נטען בפעם הראשונה, הפקד Banner שבתחתית המסך מציג את הבאנר האנכי. כעת לחץ על הלחצן כדי שהדף ייטען שנית. הפעם הדף מציג את הבאנר האופקי. מכיוון ש-DisplayDirection של הפקד Banner נקבע באירוע Page_Load רק בפעם הראשונה שהדף נטען (כאשר Page.IsPostBack הוא False), הרי שבזמן postback ההגדרה אינה נשמרת. דבר זה גורם ל-Banner לחזור להגדרת ברירת המחדל שלו שהיא Horizontal.
4. כדי להימנע מבעיה זו עם הפקד NavigateUrl, עליך ליישם אותו כמאפיין ViewState. במקום משתנה גיבוי פשוט שמשתמשים בו עם המאפיין DisplayDirection (השדה _displayDirection ב-VB.NET והמאפיין שנוצר אוטומטית ב-C#), השתמש באוסף ViewState כמחסן הגיבוי. בדרך זו הערך מאוחסן ב-ViewState, ונשלח לדפדפן וחזרה לשרת יחד עם כל בקשה. כדי ליישם את המאפיין, הוסף את הקוד הבא לקוד ברקע של פקד המשתמש Banner.ascx, מייד מתחת למאפיין DisplayDirection:

זכור, אינך חייב להקליד את כל הקוד הזה באופן ידני. אתה יכול להשתמש בקטעי הקוד / Property prop כפי שהראינו קודם. לחילופין, הקוד זמין כחלק מקוד המקור לפרק זה של הספר.

VB.NET

```
Public Property NavigateUrl() As String
    Get
        Dim _navigateUrl As Object = ViewState("NavigateUrl")
        If _navigateUrl IsNot Nothing Then
            Return CType(_navigateUrl, String)
        Else
            Return "http://p2p.wrox.com"
        End If
    End Get
    Set(ByVal Value As String)
        ViewState("NavigateUrl") = Value
    End Set
End Property
```

C#

```
public string NavigateUrl
{
    get
    {
        object _navigateUrl = ViewState["NavigateUrl"];
        if (_navigateUrl != null)
        {
            return (string)_navigateUrl;
        }
        else
        {
            return "http://p2p.wrox.com";
        }
    }
    set
    {
        ViewState["NavigateUrl"] = value;
    }
}
```

5. עבור ל- Markup View של פקד המשתמש והוסף את התכונה `runat="server"` לשני הקישורים. קבע את ID של הלוח האנכי ל- `lnkVertical` ואת ID של הלוח האופקי קבע ל- `lnkHorizontal`. בסיום תקבל קוד כזה:

```
<a href="http://p2p.wrox.com" target="_blank" runat="server" id="lnkVertical">
...
<a href="http://p2p.wrox.com" target="_blank" runat="server" id="lnkHorizontal">
```

6. חזור לקוד ברקע של פקד המשתמש ושנה את מטפל האירוע Page_Load של פקד המשתמש, כך שיגדיר גם את המאפיין HRef של אלמנט העוגן:

VB.NET

```
Case Direction.Horizontal
    pnlHorizontal.Visible = True
    lnkHorizontal.HRef = NavigateUrl
Case Direction.Vertical
    pnlVertical.Visible = True
    lnkVertical.HRef = NavigateUrl
```

C#

```
case Direction.Horizontal:
    pnlHorizontal.Visible = true;
    lnkHorizontal.HRef = NavigateUrl;
    break;
case Direction.Vertical:
    pnlVertical.Visible = true;
    lnkVertical.HRef = NavigateUrl;
    break;
```

7. שמור את השינויים וחזור לקוד ברקע של AboutUs.aspx. שנה את הקוד כדי שיגדיר את המאפיין NavigateUrl של הפקד Banner ל-URL שונה. אתה צריך לכתוב על הקוד שמגדיר את DisplayDirection.

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Me.Load
    If Not Page.IsPostBack Then
        Banner1.NavigateUrl = "http://imar.spaanjaars.com"
    End If
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        Banner1.NavigateUrl = "http://imar.spaanjaars.com";
    }
}
```

8. שמור את כל השינויים ואז בקש בדפדפן את הדף AboutUs.aspx. כדי לוודא שתקבל חלון דפדפן נקי, סגור תחילה את כל החלונות הפתוחים. לחץ על הבאנר האופקי שבצד השמאלי של הדף. כעת יצוץ חלון חדש, שייציג את ה-URL שהגדרת בצעד הקודם.

9. סגור את החלון החדש. כדי לגרום לדף להישלח חזרה לשרת, לחץ על הלחצן שהוספת קודם ל-AboutUs.aspx. ברגע שהדף ייטען שוב, לחץ על תמונת הבאנר. כעת תופנה

לאותו אתר שהופנית אליו בצעד 8 לעיל. הדבר ממחיש את העובדה שהמאפיין `NavigateUrl` מסוגל כעת לשמור על ערכו על פני `postbacks`, בשונה מהמאפיין `DisplayDirection` שהוספת קודם לפקד המשתמש.

כיצד זה פועל

במהלך שלושת הצעדים הראשונים היית עד להתנהגות מאפיינים שאינם מסוג `ViewState`. התחלת בכתיבת קוד במטפל האירוע `Page_Load`, כדי להגדיר באמצעות קוד תכנות את המאפיין `DisplayDirection`:

VB.NET

```
If Not Page.IsPostBack Then
    Banner1.DisplayDirection = Direction.Vertical
End If
```

C#

```
if (!Page.IsPostBack)
{
    Banner1.DisplayDirection = Direction.Vertical;
}
```

כתוצאה מבדיקת `Page.IsPostBack` מתבצע קוד זה רק כשהדף נטען בפעם הראשונה. הוא אינו מתבצע כשהדף נטען פעם נוספת כתוצאה מ-`postback`. כשהקוד מתבצע, הוא מגדיר את המאפיין `DisplayDirection` של הפקד `Banner` באופן שהבאנר יציג את התמונה הנכונה. עם זאת, ברגע שהדף נשלח חזרה (`posted back`) ערך זה אובד, והפקד שב ומשתמש בברירת המחדל של `DisplayDirection` שהיא `Horizontal`. דרך אחת להתגבר על בעיה זו היא לגרום לכך שהקוד יתבצע הן בפעם הראשונה והן ב-`postbacks` שלאחר מכן. כדי להשיג זאת צריך להסיר את הבדיקה של `Page.IsPostBack`. עם זאת, לא תמיד זהו הפתרון הרצוי. נניח שאתה מקבל מבסיס נתונים את כיוון התצוגה הנכון. מכיוון שקריאת נתונים מבסיס נתונים היא פעולה יקרה, ברצונך להקטין את מספר הגישות לבסיס הנתונים. בתסריט כזה מתכנתים נוטים בדרך כלל לקרוא את הנתונים רק כשהדף נטען בפעם הראשונה, ומצפים שהנתונים יהיו זמינים גם ב-`postbacks` הבאים. זה בדיוק מה שעושה המאפיין `NavigateUrl`. אתה קובע את ערכו פעם אחת והוא נשאר זמין, גם אם הדף מוחזר (`posted back`) אל השרת. דבר זה מושג בעזרת המאפיין `ViewState`.

כדי לראות כיצד זה פועל, התבונן תחילה ב-`setter` של מאפיין זה:

VB.NET

```
Public Property NavigateUrl() As String
    ...
    Set(ByVal Value As String)
        ViewState("NavigateUrl") = Value
    End Set
End Property
```

C#

```
public string NavigateUrl
{
    ...
    set
    {
        ViewState["NavigateUrl"] = value;
    }
}
```

כשאתה מקצה ערך למאפיין `NavigateUrl`, הערך מאוחסן באוסף `ViewState`. אפשר להסתכל על האוסף `ViewState` כשק גדול שניתן לאחסן בו נתונים באופן שאפשר לשלפם שוב לאחר `postback`. אתה מזהה ערכים ב-`ViewState` על ידי שימוש במפתח ייחודי. המפתח שבדוגמה זהה לשם המאפיין, וכך קל לראות שהם שייכים זה לזה. ברגע שהקצית ערך למאפיין `ViewState`, הוא מאוחסן בדף בשדה המוסתר `__VIEWSTATE` שאודותיו למדת בפרק 4. פירוש הדבר שערך זה נשלח לדפדפן כשהדף נטען, ונשלח חזרה לשרת כשהדף מוחזר (`posted back`).

בתהליך `postback`, מתבצע שוב הקוד שב-`Page_Load` של פקד המשתמש. בדיוק כמו במקרה של הבקשה הראשונה, הקוד ניגש למאפיין `NavigateUrl` בבליק `Select Case` או בבליק `switch`, על פי שפת התכנות.

VB.NET

```
Case Direction.Horizontal
    ...
    lnkHorizontal.HRef = NavigateUrl
    ...
```

C#

```
case Direction.Horizontal:
    ...
    lnkHorizontal.HRef = NavigateUrl;
    ...
```

הערך של `NavigateUrl` מוחזר על ידי הגדרת חלק ה-`get` של המאפיין:

VB.NET

```
Public Property NavigateUrl() As String
    Get
        Dim _navigateUrl As Object = ViewState("NavigateUrl")
        If _navigateUrl IsNot Nothing Then
            Return CType(_navigateUrl, String)
        Else
            Return "http://p2p.wrox.com"
        End If
    End Get
    ...
End Property
```

C#

```
public string NavigateUrl
{
    get
    {
        object _navigateUrl = ViewState["NavigateUrl"];
        if (_navigateUrl != null)
        {
            return (string)_navigateUrl;
        }
        else
        {
            return "http://p2p.wrox.com";
        }
    }
    ...
}
```

בתחילה הקוד מנסה לקבל את הערך מ-`ViewState` על ידי שימוש ב-`ViewState("NavigateUrl")` ב-`VB.NET`, או שימוש ב-`ViewState["NavigateUrl"]` ב-`C#`. שים לב ש-`C#` משתמשת בסוגריים מרובעים כדי לגשת לפריטים באוסף. אם הערך המוחזר הוא `Nothing` או `null`, הבלוק `get` מחזיר את ערך ברירת המחדל של המאפיין: `http://p2p.wrox.com`.

אם הערך אינו `Nothing` (כלומר יש ערך כלשהו), הוא מומר למחרוזת תוך שימוש בהמרה `CType` ב-`VB.NET` או `(string)` ב-`C#`. בסופו של דבר הערך מוחזר אל הקוד הקורא. לבסוף, `NavigateUrl` שמוחזר מהמאפיין `ViewState` מוקצה שוב למאפיין `HRef` של תגית העוגן, שמשמשת לקביעת URL הנכון עבור התמונה בדפדפן.

שיקולים לשימוש ב-`ViewState`

למרות ש-`ViewState` נועד להתגבר על בעיית שימור המצב שתוארה בתרגיל הקודם, עליך לשקול בזהירות את השימוש בו. הערכים שאתה מאחסן ב-`ViewState` נשלחים לדפדפן וחזרה לשרת עם כל בקשה. כשמאחסנים ערכים רבים או גדולים ב-`ViewState`, הדבר מגדיל את הדף ולפיכך משפיע באופן שלילי על הביצועים. לעולם אל תאחסן אובייקטים גדולים כמו רשומות בסיס נתונים ב-`ViewState`; לעתים קרובות מהיר יותר לקבל בכל בקשה עותק טרי מבסיס הנתונים, מאשר להעביר את הערכים דרך השדה המוסתר `ViewState`.

עצות מעשיות למשתמש בפקדי משתמש

הרשימה הבאה מספקת כמה עצות מעשיות לכתיבה והפעלה של פקדי משתמש:

- ⊙ אל תעשה שימוש יתר בפקדי משתמש. פקדי משתמש מיועדים בעיקר כדי לעטוף תוכן שחוזר על עצמו, אבל הם גם מקשים מעט על ניהול האתר, מכיוון שקוד ולוגיקה נמצאים במספר קבצים. במקרה של ספק לגבי שימוש חוזר בתוכן מסוים בחלק אחר של האתר, התחל עם הטמעתו ישירות בדף. אם יהיה צורך בשלב מאוחר יותר, תוכל תמיד להעבירו לפקד משתמש נפרד.
- ⊙ הקפד שפקדי משתמש יתמקדו במשימה אחת. אל תיצור פקד משתמש שמציג חמישה סוגים שונים של תכנים בלתי-קשורים על ידי שימוש במאפיין שקובע מה להציג. הדבר יגרום לקשיים בניהול ותחזוקה של הפקד. במקום זאת צור חמישה פקדים "במשקל קל" והשתמש בהם בהתאם.
- ⊙ כשתיצור פקדי משתמש שמכילים סימני עריכה מסוגנים, אל תקודד באופן קשיח את מידע הסגנון, כמו `CssClass`, עבור פקדי השרת שמוכלים בפקד המשתמש. במקום זאת שקול יצירת מאפייני `CssClass` נפרדים בפקד המשתמש, אשר ישמשו להגדרת `CssClass` של פקדי השרת. הדבר ישפר את אפשרויות השימוש החוזר בפקד המשתמש, ויקל על שילוב הפקד בעיצובים שונים.

סיכום

פקדי משתמש יכולים לשפר מאוד את תחזוקת האתר. במקום לחזור על אותם סימני עריכה וקוד בדפים רבים ושונים באתר, אתה עוטר את הקוד בפקד אחד, שניתן להשתמש בו בדפים השונים.

כדי לשלב פקד משתמש בדף אינטרנט, יש צורך לרשום אותו תחילה. אפשר לעשות זאת בכל דף בנפרד על ידי שימוש בהנחיה `@ Register` בראש הדף שמגדירה את `tagname`, `tagprefix`, ואת המקור של פקד המשתמש. לחילופין, את הפקדים שהשימוש בהם יהיה ממספר מקומות ניתן לרשום במקום אחד: באלמנט `<controls>` תחת `<Pages>` שבקובץ `web.config`.

לשיפור השימושיות של הפקדים תוכל להוסיף להם ולוגיקה. יצירת פקדים עם מאפיינים שניתן לקבוע את ערכיהם בדפים שצורכים אותם, תאפשר לשנות את התנהגות הפקד בזמן ביצוע. ראית כיצד ליצור מאפיין שמשתמש בסוג ספירה (enumeration) מותאם, כדי לקבוע את סוג הבאנר שיוצג. למדת גם כיצד ליצור מאפיין `ViewState` מתקדם שמסוגל לשמור את מצבו על פני `postbacks`. עם סוגים אלה של מאפיינים יהיה קל יותר לשנות את התנהגות הפקד באופן תכנותי.

למרות שמאפייני `ViewState` פותרים כמה מבעיות המצב שאתה עלול להיתקל בהן, עליך לשקול בזהירות אם אתה באמת זקוק להם. מכיוון שמאפיינים אלה מוסיפים לגודל הדף, עלולה להיות להם השפעה שלילית על ביצועי האתר.

אתה יכול לשפר את הפקד Banner על ידי ניהול מעקב אחר מספר הפעמים שכל תמונה נלחצה. האתר לדוגמה עבור ספר זה אינו מיישם זאת, אבל עם המידע שתקבל בפרקים שעוסקים בעבודה מול בסיסי נתונים, תוכל לעשות זאת בעצמך ללא קושי.

בפרק 9 תיצור פקד משתמש מסוג אחר, אשר משמש כטופס "יצירת קשר". בניית הטופס כפקד משתמש תקל על בקשת משוב מהמשתמשים ממקומות שונים באתר.

תרגילים

1. בפרק זה למדת כיצד ליצור מאפיין רגיל ומאפיין ViewState. מה ההבדל העיקרי בין השניים? מה החסרונות של כל אחד מהם?
2. כיום, המאפיין DisplayDirection של הפקד Banner אינו שומר את מצבו על פני postbacks. שנה את קוד המאפיין כדי שיהיה מסוגל לשמור את מצבו תוך שימוש באוסף ViewState, באופן דומה לדרך שבה NavigateUrl שומר את ערכו.
3. מהן שתי התועלות העיקריות של שימוש בסוג ספירה (enumeration) מותאם כמו Direction, לעומת סוגים מובנים כמו System.Byte או String?

פרק 9

בדיקת אימות הקלט מהמשתמש

שים לב, פרק זה עוסק ב-validation (ולידציה, תיקוף). מכיוון שנהוג בין אנשי המקצוע לומר אימות במקום תיקוף, המילים אימות ותיקוף משמשות לסירוגין בפרק זה.

עד כה יצרת אתר סטטי שבו שלטת בעיצוב ובתוכן. אתה יוצר את הדפים בזמן פיתוח ומוסיף אותם לתפריט כדי שלמשתמשים תהיה גישה אליהם. שילוב נתונים דינמיים יהפוך את האתר למושך ומעניין הרבה יותר. נתונים אלה זורמים בדרך כלל בשני כיוונים: נתונים שבאים מהשרת ונשלחים לדפדפן של משתמש הקצה, או נתונים שמוזנים על ידי המשתמש ונשלחים לשרת לעיבוד או לאחסון.

נתונים שמגיעים מהשרת יכולים להיות מאוחזרים ממקורות נתונים רבים ושונים, ובכלל זה קבצים ובסיסי נתונים, אשר מוצגים לעתים קרובות על ידי פקדי הנתונים של ASP.NET. מפרק 11 ואילך נעסוק בגישה לבסיסי נתונים.

כאשר נתונים מגיעים מכיוון המשתמש ונשלחים אל השרת, טווח המידע רחב, ומשתרע החל מבקשות לדפים פשוטים וטפסי יצירת קשר ועד לתרחישים מורכבים של סלי קנייה וממשקי משתמש דמויי-אשף. העיקרון הבסיסי של זרימת נתונים זו זהה בכל התרחישים – משתמשים מזינים נתונים לטופס אינטרנט ומגישים אותו לשרת.

כדי להבטיח שהמערכת לא תקבל נתונים שגויים או חסרי-תוקף, חשוב לבדוק את תקפות הנתונים לפני שמאפשרים למערכת לפעול עליהם. למרבה המזל, מגיעה ASP.NET 3.5 עם כלים רבים שהופכים את בדיקת תקפות הנתונים (data validation) למשימה פשוטה.

בחלק הראשון של הפרק נכיר את פקדי האימות שנתמכים על ידי ASP.NET. תלמד איזה פקדים זמינים, איך להשתמש בהם ולהתאימם אישית, ובאיזה תרחישים הם ישימים.

בחלק השני של הפרק תלמד על דרכים נוספות לעבודה עם נתונים. תלמד איך לשלוח באמצעות דואר אלקטרוני את המידע שמזין המשתמש למערכת, ואיך להתאים אישית את גוף הודעת הדואר תוך שימוש בתבניות מבוססות טקסט.

בפרק זה תלמד בהרחבה על הנושאים החשובים הבאים:

- ⊙ מהו קלט משתמש ומדוע חשוב לאמת אותו ולבדוק את תקפותו?
- ⊙ איזה סיוע מציעה ASP.NET 3.5 לאימות נתוני משתמש?

- ⊙ איך תפעל עם הפקדים המובנים לאימות, ואיך תיצור פתרונות שאינם נתמכים בכלים התקניים?
- ⊙ איך שולחים דואר אלקטרוני על ידי ASP.NET?
- ⊙ איך קוראים קבצי טקסט?

בסיום הפרק תבין כיצד זורם המידע ביישום אינטרנט מבוסס ASP.NET, ושל הטכניקות השונות העומדות לרשותך לאימות נתונים אלה ולבדיקת התקפות שלהם.

איסוף נתונים מהמשתמש

קלט מהמשתמש הינו חלק בלתי נפרד מכל אתר אינטרנט. ככלל, קלט זה יכול להישלח לשרת האינטרנט בשתי טכניקות שונות, GET ו-POST. בפרק 4 ראינו בקצרה את השוני בין שתי שיטות אלו. נתוני GET מצורפים לכתובת הממשית של הדף המבוקש, בעוד שעם השיטה POST הנתונים נשלחים בגוף הבקשה אל הדף.

על פי השיטה GET, הנתונים מתווספים למחרוזת השאילתה של הדף. תוכל לאחזר אותם על ידי שימוש במאפיין QueryString של האובייקט Request, כפי שלמדנו בפרק 7. תוכל להניח שכאשר מבקשים את הדף הבא:

<http://www.PlanetWrox.com/Reviews/ViewDetails.aspx?id=34&catId=3>

אזי מחרוזת השאילתה במקרה זה היא id=34&catId=3. סימן השאלה משמש להפרדת מחרוזת השאילתה משאר הכתובת. כדי לגשת לפריטים רצויים במחרוזת השאילתה, תוכל להשתמש בשיטה Get של האוסף QueryString:

VB.NET

```
Dim id As Integer = Convert.ToInt32(Request.QueryString.Get("id")) ' id is now 34
Dim catId As Integer = Convert.ToInt32(Request.QueryString.Get("catId")) ' catId is now 3
```

C#

```
int id = Convert.ToInt32(Request.QueryString.Get("id")); // id is now 34
int catId = Convert.ToInt32(Request.QueryString.Get("catId")); // catId is now 3
```

מצד שני, השיטה POST מקבלת את הנתונים שלה מטופס עם פקדים שהוגשו לשרת. נניח שיש לך טופס עם שני פקדים, תיבת טקסט בשם txtAge שתפקידה להכיל את גיל המשתמש, ו-Button למסירת גיל זה לשרת. באירוע Click של הפקד Button תוכל לכתוב את הקוד הבא כדי להמיר את קלט המשתמש למספר שלם:

VB.NET

```
Dim age As Integer = Convert.ToInt32(txtAge.Text) ' age now holds the user's age
```

C#

```
int age = Convert.ToInt32(txtAge.Text); // age now holds the user's age
```

שים לב שבמקרה זה אין צורך לגשת לאוסף כמו Post, כפי שראית קודם עם QueryString. ASP.NET מגינה עליך מהמורכבות של אחזור נתונים באופן ידני מתוך הטופס שהוגש, ובמקום זאת מציבה את הפקדים השונים בדף שלך עם נתונים מהטופס.

הכל טוב כל עוד המשתמשים מזינים לתיבת הטקסט ערך שנראה כמו גיל; אבל מה יקרה כשמשמש יזין נתונים לא תקפים, במתכוון או במקרה? מה אם משתמש מזין את הטקסט "אני בן 36" במקום את המספר 36 בלבד? כשזה יקרה, הקוד ייכשל. כשמעבירים אל השיטה ToInt32 של המחלקה Convert ערך שאינו יכול להיות מיוצג כמספר, היא מוציאה הודעת שגיאה. כדי להימנע מבעיות אלו עליך לאמת את כל הנתונים שנשלחים לשרת, ולבדוק אם הם תקפים. כשהנתונים אינם תקפים, עליך לדחותם ולוודא שהיישום מטפל בהם כנדרש.

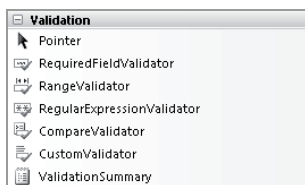
אימות הקלט של המשתמש המוזן לדפי אינטרנט

אנשים שאימות הקלט של המשתמש מדאיג אותם, משתמשים לעתים קרובות במנטרה: "לעולם אל תסמוך על קלט המשתמש". גישה כזו אולי נראית תחילה כפרנואידית, אבל היא בעלת חשיבות בכל מערכת פתוחה. גם אם אתה סבור שאתה מכיר את המשתמשים, ואפילו אם אתה נותן בהם אמון מוחלט, לא תמיד הם יהיו היחידים שמורשים לגשת למערכת. ברגע שהאתר "עולה לאוויר" באינטרנט, הוא מהווה מטרה פוטנציאלית למשתמשים בעלי כוונות זדוניות ולהאקרים שינסו למצוא דרך להיכנס לתוך המערכת. בנוסף למבקרים חורשי רע אלה, גם המשתמשים שאתה נותן בהם אמון עלולים לשלוח בשוגג נתונים שגויים אל השרת.

כדי לעזור לטיפול בבעיה זו בצורה הטובה ביותר האפשרית, כוללת ASP.NET פקדי אימות (validation controls) שונים שמסייעים באימות הנתונים ובדיקת התקפות שלהם לפני שמשתמשים בהם ביישום. בסעיפים הבאים תראה כיצד להשתמש בפקדי האימות התקניים כדי להבטיח שהמערכת תקבל נתונים מאומתים בלבד.

פקדי האימות של ASP.NET

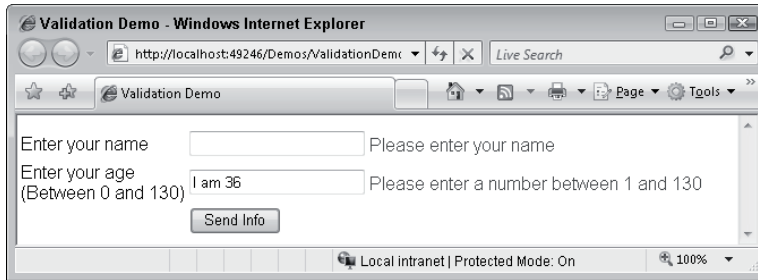
ASP.NET 3.5 כוללת ששה פקדים לביצוע validation באתר. חמישה מהם משמשים לביצוע בדיקת האימות הממשית, בעוד שהפקד האחרון - ValidationSummary - משמש לאספקת משוב למשתמש על שגיאות כלשהן שנעשו בזמן הזנת הנתונים בדף. תרשים 9-1 מציג את הפקדים הזמינים בקטגוריה Validation של ארגז הכלים.



תרשים 9-1

פקדי האימות מועילים ביותר לאימות הנתונים שהמשתמש מזין למערכת. ניתן לחברם בקלות לפקדים כמו TextBox ו-DropDownList; עם זאת, הם גם תומכים בתרחישי אימות מותאמים אישית. תרשים 9-2 מציג שניים מפקדי האימות - RequiredFieldValidator ו-RangeValidator - בפעולה שמטרתה למנוע משתמש מהגשת טופס לשרת מבלי שהזין בו נתונים נדרשים ומאומתים.

חשיבות פקדי האימות היא בכך שהם יכולים לבדוק את הקלט הן בצד הלקוח והן בשרת. כשאתה מוסיף פקד אימות לדף אינטרנט, הפקד מרנדר קוד צד לקוח (JavaScript) שבודק אימות לפקד אליו הוא מקושר. בדיקת אימות זו בצד-לקוח פועלת ברוב דפדפני האינטרנט שנמצאים כיום בשוק כולל Internet Explorer של מיקרוסופט, Firefox ו-Safari. בדיקת האימות מתבצעת בו-זמנית, ובאופן אוטומטי, גם בשרת. הדבר מקל על מתן משוב מיידי למשתמש על הנתונים תוך שימוש בתסריט צד-לקוח, בעוד דפי האינטרנט מוגנים בשרת מפני נתונים שאינם מאומתים.



תרשים 9-2

אזהרה בדבר בדיקת אימות בצד-לקוח

למרות שנראה מספיק בבדיקת האימות בצד הלקוח כדי למנוע משתמשים מלשלוח נתונים לא תקפים למערכת, לעולם אל תסמוך עליה כפתרון היחיד לבדיקה. קל לבטל פעולה של JavaScript בדפדפן, וכך שגרות האימות שבצד הלקוח תהפוכנה לחסרות תועלת. משתמש זדוני יכול בקלות לעקוף את הדף כולו בדפדפן ולשלוח לשרת את הנתונים באופן ישיר. השרת יקבל את המידע הזה ויעבד אותו, אם לא תנקוט אמצעי נגד.

ככלל, עליך לראות בבדיקת האימות שבצד הלקוח מחווה למשתמשים שלך. הדבר נותן להם משוב מיידי, כדי שיידעו ששכחו למלא שדה חובה או שהזינו נתונים שגויים, ואם בדיקת התאימות נכשלה, הם מקבלים חיווי מיידי ללא צורך להגיע לשרת באמצעות postback. עם זאת, בדיקת האימות בצד-שרת היא האמצעי הממשי היחיד לבדיקה. זו למעשה הדרך היחידה למנוע כניסת נתונים שאינם מאומתים אל המערכת.

הסעיף הבא דן בדרך הפעלת פקדי האימות כדי להגן על הנתונים.

שימוש בפקדי אימות

כדי להכריז על פקד אימות בדף ASPX, נשתמש בתחביר ההכרזה המוכר. לדוגמה, כדי ליצור את הפקד RequiredFieldValidator שהוצג בתרשים 9-2 יש צורך בקוד הבא:

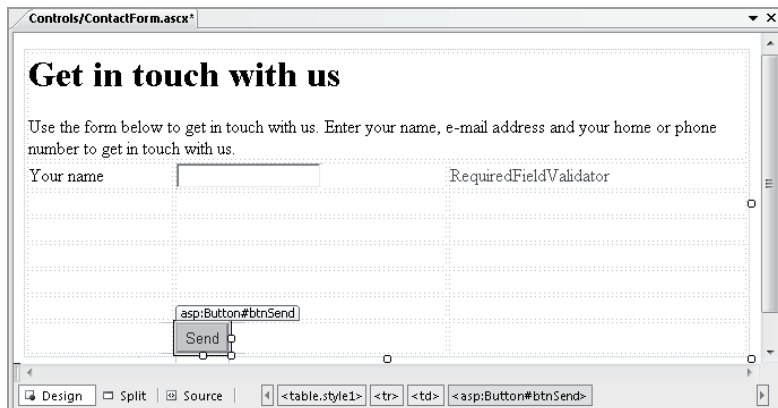
```
<asp:RequiredFieldValidator
  ID="ReqVal1" runat="server"
  ControlToValidate="TextBox1"
  Display="Dynamic"
  ErrorMessage="Please enter your name">
</asp:RequiredFieldValidator>
```

כדי שתקבל מושג כיצד הם פועלים, ינחה אותך התרגיל הבא דרך תהליך השימוש ב-RequiredFieldValidator בטופס יצירת קשר שמוצב בפקד משתמש.

נסה זאת **שימוש ב- RequiredFieldValidator**

בתרגיל זה תיצור פקד משתמש בשם ContactForm.aspx. ניתן להציבו בדף אינטרנט כדי שמבקרים באתר יוכלו להשאיר משוב. בתרגילים מאוחרים יותר תרחיב את הפקד על ידי שליחת תגובת המשתמש באמצעות דואר אלקטרוני אל חשבון הדואר האלקטרוני שלך.

1. פתח את הפרויקט Planet Wrox והוסף פקד משתמש חדש בתיקייה Controls. קרא לפקד בשם ContactForm.aspx. ודא שהוא משתמש בשפת התכנות שלך ובקובץ קוד ברקע.
2. עבור ל- Design View והכנס טבלה על ידי בחירת `Table ← Insert Table`. צור טבלה בת שמונה שורות ושלוש עמודות.
3. מזג את שלושת תאי השורה הראשונה. כדי לעשות זאת, בחר בכל שלושת התאים, לחץ לחיצה ימנית על הבחירה, ובחר `MergeCells ← Modify`.
4. בתא החדש שקיבלת, הקלד טקסט שיאמר למשתמשים שהם יכולים להשתמש בטופס יצירת קשר, כדי להתקשר אתך.
5. בתא הראשון של השורה השנייה הקלד את הטקסט **Your name**. גרור תיבת טקסט אל התא השני באותה שורה וקרא לה `txtName`. גרור פקד `RequiredFieldValidator` מהקטגוריה Validation של ארגז הכלים אל התא האחרון בשורה השנייה. לבסוף, גרור Button אל התא השני של השורה האחרונה. שנה את שם הלחצן ל-`btnSend` על ידי קביעת ה-ID שלו, והגדר את המאפיין `Text` שלו ל-`Send`. כשתסיים תצוגת Design View שלך תהיה כמו תרשים 9-3.



תרשים 9-3

6. ב- Design View, לחץ על `RequiredFieldValidator` פעם אחת ופתח את טבלת המאפיינים שלו על ידי הקשת F4. הגדר לפקד את המאפיינים הבאים:

מאפיין	ערך
ErrorMessage	Please enter your name
Text	*
ControlToValidate	txtName

7. שמור את השינויים שערכת בפקד המשתמש וסגור אותו. בשלב זה סיימת לטפל בו.
8. פתח את Contact.aspx מהתיקייה About ועבור ל- Design View.
9. גרור את פקד המשתמש ContactForm.ascx מסיר הפתרון אל שטח התוכן העיקרי של הדף, שמזוהה על ידי הגבול בצבע ארגמן. עבור ל- Markup View ותראה את הכרזת הפקד הבאה:

```
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" Runat="Server">
  <uc1:ContactForm ID="ContactForm1" runat="server" />
</asp:Content>
```
10. שמור את הדף והקש Ctrl+F5 כדי לפתוח אותו בדפדפן. אם תקבל הודעת שגיאה, ודא ששינית את שם תיבת הטקסט ל-txtName, וכי קבעת למאפיין ControlToValidate של RequiredFieldValidator אותו ID control.
11. השאר את תיבת הטקסט Name ריקה ולחץ על הלחצן Send. שים לב שהדף אינו מבצע פנייה לשרת (Postback). במקום זאת תראה שמופיעה כוכבית (*) אדומה בשורה של שדה השם, לציון שגיאה.
12. הקלד את שמך ולחץ שנית על Send. כעת הדף נשלח חזרה אל השרת בהצלחה.

כיצד זה פועל

כאשר RequiredFieldValidator מצורף אל TextBox באמצעות המאפיין ControlToValidate, נשלח לדפדפן קוד JavaScript שבודק את התוקף של הפקד בצד הלקוח. הפקד RequiredFieldValidator מסוגל לבדוק תוקף של פקד כמו TextBox. הוא עושה זאת על ידי השוואת ערך הפקד עם המאפיין InitialValue שלו עצמו, וכדי לוודא שהערך של הפקד השתנה. ברירת המחדל של מאפיין זה היא מחרוזת ריקה. כלומר, כל דבר פרט למחרוזת ריקה נחשב לערך תקף. בכל פעם שמשתמש מנסה להגיש לשרת טופס על ידי לחיצה על Send, פקד האימות בודק את הפקד שהוא מצורף אליו. כשתיבת הטקסט עדיין ריקה, מוצגת הכוכבית האדומה מהמאפיין Text של תיבת הטקסט, והטופס אינו מועבר. כשהמשתמש מקליד משהו בתיבת הטקסט txtName, האימות מתבצע ועובר בהצלחה והדף מועבר אל השרת.

פרט לפקד RequiredFieldValidator, מכילה הקטגוריה Validation של ארגז הכלים כמה פקדים אחרים שבהם נדון להלן.

פקדי האימות התקניים

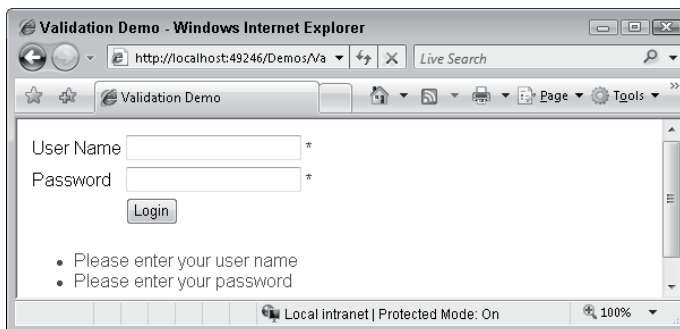
בסופו של דבר, כל חמשת פקדי האימות יורשים מאותה מחלקה בסיסית, ולפיכך חולקים התנהגות משותפת מסוימת. ארבעה מחמשת פקדי האימות פועלים באותה דרך, ומכילים התנהגות מובנית שמאפשרת לברוק את האימות של פקדים מקושרים. הפקד האחרון, CustomValidator, מאפשר לכתוב קוד אימות מותאם אישית. הטבלה הבאה מפרטת את המאפיינים המשותפים לפקדי האימות, שבהם תשתמש בדרך כלל בזמן עבודה עם פקדים מסוג זה.

מאפיין	תיאור
Display	מאפיין זה קובע אם הודעת השגיאה המוסתרת תתפוס שטח או לא. כשהמאפיין מוגדר כ-Static, הודעת השגיאה תופסת שטח מסך אפילו כשהיא מוסתרת. הדבר דומה להגדרת <i>visibility: hidden</i> של CSS שראית בפרקים קודמים. האפשרות Dynamic דומה להגדרת <i>display: none</i> של CSS. הודעת השגיאה אינה נראית כלל עם הגדרת המאפיין כ-None. הסתרת הודעת השגיאה הממשית יכולה להיות שימושית כאשר משתמשים ב-ValidationSummary, כפי שנלמד בהמשך הפרק.
CssClass	מאפיין זה מאפשר לקבוע את מחלקת CSS שמיושמת על טקסט הודעת השגיאה.
ErrorMessage	מאפיין זה מתייחס להודעת השגיאה עבור פקד האימות שמשתמשים בו בפקד ValidationSummary. כשהמאפיין Text ריק, משמש הערך של ErrorMessage גם כטקסט שמופיע בדף.
Text	המאפיין Text משמש כטקסט שפקד האימות מציג בדף. זו יכולה להיות כוכבית לציון שגיאה או שדה חובה, או טקסט כמו "Please enter your name".
ControlToValidate	זהו ID (מזהה) של הפקד אותו יש לאמת.
EnableClientScript	מאפיין זה מקבל או מגדיר הגדרה שקובעת אם הפקד יספק אימות בצד הלקוח. ברירת המחדל היא True.
SetFocusOnError	מאפיין זה קובע אם תסריט צד-לקוח מתמקד בפקד הראשון שיצר הודעת שגיאה. ברירת המחדל היא False.
ValidationGroup	ניתן לקבץ את פקדי האימות יחד. הדבר מאפשר ביצוע אימות כנגד קבוצת פקדים. כל הפקדים בעלי אותו ValidationGroup נבדקים באותו הזמן. פירוש הדבר שפקדים שאינם חלק מקבוצה זו אינם נבדקים. חשוב לדוגמה, על דף התחברות עם לחצן Login ושדות עבור שם המשתמש וסיסמה. אותו דף יכול להכיל גם תיבת חיפוש שמאפשרת לערוך חיפוש באתר. עם ValidationGroup הלחצן Login יברוק את

מאפיין	תיאור
	אימות תיבות שם המשתמש והסיסמה, בעוד שלחצן החיפוש יפעיל אימות לגבי תיבת החיפוש בלבד.
IsValid	בדרך כלל לא תקבע מאפיין זה בזמן העיצוב, אבל בזמן ביצוע הוא מספק מידע אם בדיקת האימות עברה בהצלחה.

ההבדל בין המאפיינים Text ו-ErrorMessage

במבט ראשון נראה ששני מאפיינים אלה משרתים את אותה מטרה. בשניהם ניתן להשתמש כדי לספק למשתמש משוב בצורת הודעת שגיאה. אבל כאשר משתמשים בהם בשילוב עם הפקד ValidationSummary יש הבדל מסוים בין השניים. כשאתה מגדיר את שני המאפיינים באותו הזמן, הפקד Validation מציג את המאפיין Text, בשעה שהפקד ValidationSummary מציג את ErrorMessage. תרשים 9-4 מציג דוגמה של דף Login עם שני פקדי RequiredFieldValidator. המאפיין Text נקבע בשני פקדי האימות לכוכבית (*) כדי לתת למשתמש רמז חזותי שהתעוררה בעיה. ה-ValidationSummary שמתחת לפקד מציג אז את מאפיין ErrorMessage במלואם.



תרשים 9-4

בסעיף הבא תקבל הסבר על שלושת פקדי האימות הנותרים. בתרגיל שיבוא בעקבותיו תראה אותם בפעולה. סעיף מאוחר יותר יראה לך כיצד להשתמש בפקדים CustomValidator ו-ValidationSummary.

אימות טווח - RangeValidator

הפקד RangeValidator מאפשר לבדוק אם ערך נמצא בתוך טווח מסוים. הפקד מסוגל לבדוק סוגי נתונים כמו מחרוזות, מספרים, תאריכים וערכים כספיים. לדוגמה, אפשר להשתמש בו כדי לוודא שמספר נמצא בין 1 ל-10, או שתאריך נבחר יחול בתוך השבועיים הקרובים. הטבלה הבאה מפרטת את המאפיינים היותר חשובים של פקד זה.

מאפיין	תיאור
MinimumValue	מאפיין זה קובע את הערך הנמוך ביותר המקובל. לדוגמה, כשבודקים מספר שלם בין 1 ל-10, תקבע מאפיין זה ל-1.
MaximumValue	מאפיין זה קובע את הערך הגבוה ביותר המקובל. לדוגמה, כשבודקים מספר שלם בין 1 ל-10, תקבע מאפיין זה ל-10.
Type	מאפיין זה קובע את סוג הנתונים שיבדוק פקד האימות. הגדרת הערך ל- String, Integer, Double, Date, או Currency תגרום לבדיקת סוג הנתונים המתאים.

אימות של ביטוי - `RegularExpressionValidator`

הפקד `RegularExpressionValidator` מאפשר לבדוק ערך מול ביטוי מסוג `regular expression`. ביטויי `regular expression` מציעים תחביר דחוס שמאפשר לחפש תבניות במחרוזות טקסט. ביטויי `regular expression` הוא נושא מורכב, אבל התוכנה `Visual Web Developer` כוללת כמה ביטויים מובנים שמקלים על בדיקת האימות של ערכים כמו כתובות דואר אלקטרוני ומיקוד. הביטוי הרגיל מוגדר במאפיין `ValidationExpression` של הפקד.

אימות השוואה - `CompareValidator`

אפשר להשתמש בפקד `CompareValidator` כדי להשוות ערך של פקד אחד לערך אחר. הדבר משמש לעתים קרובות בטפסי חתימה כשהמשתמש נדרש להקליד סיסמה פעמיים כדי לוודא שאכן הוקלדה אותה סיסמה בשני המקרים. לחילופין, במקום להשוות לפקד אחר, ניתן להשוות כנגד ערך קבוע.

הטבלה הבאה מפרטת את המאפיינים הנוספים של הפקד `CompareValidator`.

מאפיין	תיאור
ControlToCompare	מאפיין זה מכיל את ID (מזהה) הפקד שבעת האימות משווים כנגדו. כשמאפיין זה מוגדר, אין ל- <code>ValueToCompare</code> כל השפעה.
Operator	מאפיין זה קובע את סוג פעולת ההשוואה. לדוגמה, כשהמאפיין מוגדר כ- <code>Equal</code> חייבים שני הפקדים להכיל את אותו הערך כדי שבודק התקפות יקבל אותם כתקפים. באופן דומה יש אפשרויות נוספות כמו <code>NotEqual</code> , <code>GreaterThan</code> ו- <code>GreaterThanEqual</code> כדי לבצע פעולות אימות שונות.
Type	מאפיין זה קובע את סוג הנתונים שפקד האימות צריך לבדוק. הגדרת הערך ל- String, Integer, Double, Date, או Currency, תגרום לבדיקת סוג הנתונים המתאים.

מאפיין	תיאור
ValueToCompare	מאפיין זה מאפשר לקבוע ערך קבוע כדי להשוות כנגדו. הדבר משמש לעתים קרובות בהסכמים כשעליך לרשום מילה כמו Yes כדי לציין את הסכמתך לתנאי מסוים. הגדר את המאפיין למילה Yes ואת ControlToValidate לפקד שברצונך רק לבדוק אם הוא תקף. כשמאפיין זה מוגדר, ודא שניקית את המאפיין ControlToCompare, שאם לא כן האחרון מקבל קדימות.

בתרגיל הבא תראה את רוב הפקדים האלה בפעולה, פרט ל-RangeValidator. עם זאת, השימוש בפקד זה דומה לשימוש בשאר פקדי האימות, ועל כן, להוסיפו לדף אינטרנט או לפקד משתמש, קל באותה מידה.

נסה זאת הרחבת טופס יצירת קשר

בתרגיל נסה זאת הקודם בנית את הבסיס של טופס יצירת קשר, כשיצרת פקד משתמש עם טבלה וכמה פקדים שאפשרו למשתמשים להזין את שמותיהם. בתרגיל זה תרחיב את הטופס ותוסיף שדות עבור כתובת הדואר האלקטרוני, מספר הטלפון הביתי ומספר הטלפון העסקי. תשתמש בפקדי האימות כדי להבטיח שכתובת הדואר האלקטרוני היא בעלת מבנה תקף ונכון, וכי נרשם לפחות מספר טלפון אחד מבין השניים. כדי לוודא שמשתמשים הזינו כתובת דואר אלקטרוני נכונה, הם מתבקשים להזינה פעמיים. אם התנהגות זו אינה לרוחק, מחק את השורה עם תיבת הטקסט עבור ההזנה השנייה של כתובת הדואר האלקטרוני והתעלם מ-CompareValidator.

1. פתח את ContactForm.ascx מהתיקייה Controls, והעבר אותו ל- Design View.
2. בעמודה השנייה, גרור חמש תיבות טקסט נוספות לתאי הטבלה הריקים בין תיבת הטקסט עבור השם והלחצן Send. קרא לפקדים בשמות הבאים, מלמעלה למטה:

txtEmailAddress ⊙

txtEmailAddressConfirm ⊙

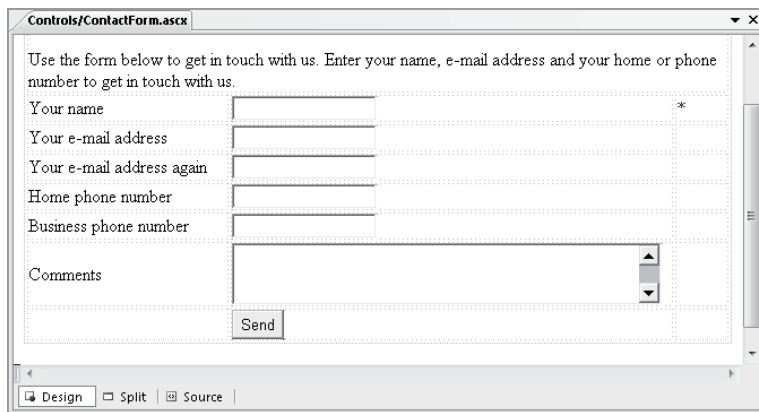
txtPhoneHome ⊙

txtPhoneBusiness ⊙

txtComments ⊙

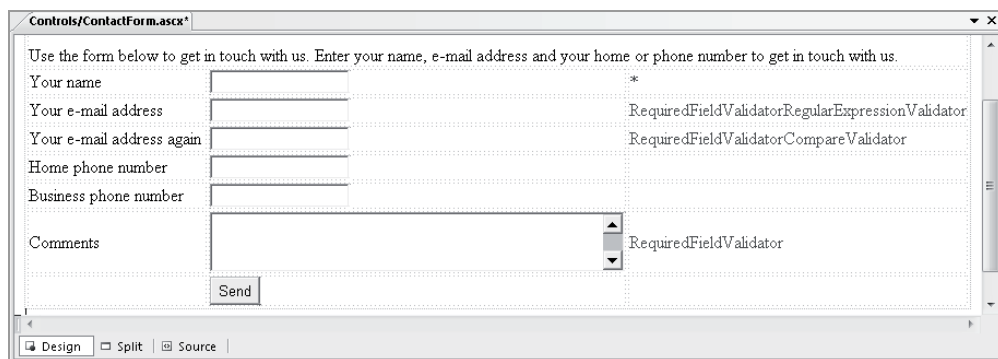
3. הגדר את המאפיין TextMode של txtComments ל-MultiLine ואז השתמש במעצב להגדלה של רוחב הפקד וגובהו, כדי שלמשתמש יהיה קל יותר להוסיף הערה. תוכל גם להתאים את רוחב עמודת הטבלה כדי ליישר את הודעת השגיאה קרוב יותר לפקדי תיבות הטקסט.

4. לתא הראשון של השורות שהוספת להן פקדי TextBox, הוסף את הטקסט שאתה רואה בתרשים 5-9.



תרשים 9-5

5. גרור את `RequiredFieldValidator` ואת `RegularExpressionValidator` אל העמודה האחרונה של השורה, עבור כתובת הדואר האלקטרוני הראשונה. אל העמודה האחרונה של השורה עבור כתובת הדואר האלקטרוני השנייה גרור את `RequiredFieldValidator` ואת `CompareValidator`. לבסוף, גרור את `RequiredFieldValidator` אל התא האחרון של השורה `Comments`. כשתסיים, הטופס שלך יהיה דומה לזה שמוצג בתרשים 9-6.



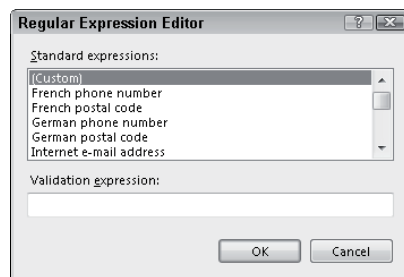
תרשים 9-6

6. פתח את רשת המאפיינים עבור כל אחד מפקדי האימות שהוספת והגדר את המאפיין `Text` לכוכבית (*) ואת המאפיין `Display` ל-`Dynamic`. הגדר את שאר מאפייני הפקדים בהתאם לטבלה הבאה.

מאפיינים שיש לשנות		פקד
הכנס כתובת מייל <code>txtEmailAddress</code>	<code>ErrorMessage:</code> <code>ControlToValidate:</code>	<code>RequiredFieldValidator</code> (עבור כתובת הדוא"ל הראשונה)
הכנס כתובת מייל תקפה <code>txtEmailAddress</code>	<code>ErrorMessage:</code> <code>ControlToValidate:</code>	<code>RegularExpressionValidator</code>

מאפיינים שיש לשנות		פקד
ורא כתובת מייל txtEmailAddressConfirm	ErrorMessage: ControlToValidate:	RequiredFieldValidator (עבור כתובת הדוא"ל השנייה)
הקלד שוב כתובת מייל txtEmailAddressConfirm txtEmailAddress	ErrorMessage: ControlToValidate: ControlToCompare:	CompareValidator
הכנס הערה txtComments	ErrorMessage: ControlToValidate:	RequiredFieldValidator (עבור שדה הערה)

7. כשאתה עדיין ב- Design View, לחץ על RegularExpressionValidator פעם אחת, פתח את טבלת המאפיינים שלו, ואתר את המאפיין ValidationExpression. כשתלחץ על מאפיין זה בטבלת המאפיינים תראה לחצן עם אליפסות. כשתלחץ על לחצן זה תופיע תיבת דו-שיח שמאפשרת לבחור ביטוי רגיל, כפי שמוצג בתרשים 9-7.



תרשים 9-7

8. ברשימה, לחץ על השורה Internet e-mail address ושים לב ש-VWD מכניסה ביטוי *regular expression* בתיבה Validation expression. לחץ OK כדי להוסיף את המאפיין לפקד ושחרר את תיבת הדו-שיח.

9. שמור את השינויים ובקש את Contact.aspx בדפדפן. שחק עם פקדי האימות השונים על ידי כך שתשמיט נתוני חובה או על ידי כך שתזין נתונים לא תקפים, ולא מאומתים. הדף יוגש לשרת רק לאחר שתמלא את כל שדות החובה ותקליד את אותה כתובת דואר אלקטרוני בשתי תיבות הטקסט. בשלב זה תראה רק את הכוכבית האדומה מופיעה כדי לציין בעיה כלשהי. לאחר שראית כיצד פועלים פקדי האימות, תלמד כיצד להשתמש ב-ValidationSummary כדי לספק מידע מפורט יותר למשתמש.

כיצד זה פועל

בדיוק כמו הפקד RequiredFieldValidator כך גם שאר פקדי האימות מפיקים קוד JavaScript הלקוח. קוד JavaScript זה מופעל כשאתה לוחץ על הלחצן Send או כשהערך של אחד מפקדי הלקוח משתנה. הפקד CompareValidator בוחן את הערכים של שני פקדים שונים, ומחזיר true

רק כששני הפקדים המושווים מכילים אותו נתון. חשוב להבין שהפקד CompareValidator אינו מפעיל את קוד האימות שלו כאשר תיבות הטקסט ריקות. מכאן החשיבות של חיבור פקד RequiredFieldValidator. האחרון מוודא תחילה שהשתמש מזין נתונים כלשהם ורק אחר כך הפקד CompareValidator מוודא שהטקסט בשתי תיבות הטקסט זהה.

הפקד RegularExpressionValidator בודק את תבנית הנתונים שהוא מאמת. אם תסתכל במאפיין ValidationExpression של הפקד, תראה מחרוזת ארוכה וסודית. תבנית זו מבטיחה שכתובת הדואר האלקטרוני מכילה איזשהו טקסט, ואחריו, לפי בחירה, יש תו הפרדה כלשהו כמו מקף (-) או נקודה, ואחריו טקסט נוסף. התבנית גם מבטיחה שהסימן @ מופיע בכתובת, ואחריו שם תחום (domain), נקודה, ואחריה לפחות תו אחד נוסף שמייצג את שם התחום העילי כגון .com, .nl, או .co.il. עם ביטוי זה, you@yourprovider.com ייחשב לכתובת דואר אלקטרוני תקפה. כך גם הכיתוב a@a.a ייחשב לכתובת תקפה ונכונה, אך הכיתוב you@you יובן ככתובת בלתי-תקפה.

שים לב שהפקד RegularExpressionValidator בודק את תחביר כתובת הדואר האלקטרוני בקירוב בלבד. בהחלט ייתכן שתזון כתובת דואר אלקטרוני שאינה קיימת, אך רק נראית תקפה, או אפילו כתובת דואר אלקטרוני שאינה תקפה כלל, דוגמת a@a.a. עם זאת, במקרים רבים מספיקה בדיקת תקפות זו כדי לסנן שגיאות הקלדה נפוצות שמשתמשים עושים כשהם מזינים כתובות דואר אלקטרוני.

הפקדים ValidationSummary ו-CustomValidator

פקדי האימות שהכרת עד כה קלים מאוד לשימוש. אתה מוסיף אותם לדף, מגדיר כמה מאפיינים, והם יעשו עבורך את כל "העבודה השחורה". עם זאת, הם אינם תומכים בכל תרחיש אימות אפשרי שאתה עשוי לחשוב עליו. לדוגמה, מה אם תרצה להבטיח שמשתמש הזין לפחות מספר טלפון אחד מתוך שני המספרים? זה המקום שבו נכנס לפעולה הפקד CustomValidator. בודק תקפות זה מאפשר לכתוב פונקציות אימות מותאמות אישית הן עבור הלקוח (ב-JavaScript) והן עבור השרת (ב-VB.NET או C#). הדבר נותן גמישות רבה ביחס למידע שברצונך לאמת.

הפקד ValidationSummary מספק למשתמש רשימת שגיאות שאחזר ממאפייני ErrorMessage של פקדי האימות השונים. הוא יכול להציג שגיאות אלו בשלוש דרכים שונות: תוך שימוש ברשימה מוטמעת בדף, תוך שימוש בתיבת הודעה שכתובה ב-JavaScript או שימוש בשתי הדרכים בו-זמנית. ניתן לשלוט בהגדרות אלו בעזרת שני המאפיינים ShowSummary ו-ShowMessageBox. בנוסף, אפשר לשנות את אופן הצגת רשימת השגיאות בעזרת המאפיין DisplayMod. ברירת המחזר שלו היא BulletedList כשכל שגיאה היא פריט ברשימה לא סדורה. אפשרויות אחרות הן List (ללא תבליטים) ו-SingleParagraph.

בתרגיל הבא תלמד לכתוב שיטות אימות לצד-לקוח ולצד-שרת, וכיצד להשתמש בפקד ValidationSummary.

בתרגיל זה תראה כיצד להשתמש בפקד CustomValidator בדף האתר כדי להבטיח שהזון לפחות מספר טלפון אחד מבין השניים. בנוסף תלמד כיצד לספק למשתמש משוב על ידי שימוש בפקד ValidationSummary.

1. חזור לפקד ContactForm.ascx ב-WVD והעבר אותו לתצוגת Design View. לחץ לחיצה ימנית על השורה עם הפקד Button, ומתפריט ההקשר בחר Insert ← Row Below כדי ליצור שורה חדשה בטבלה. לחילופין, לחץ חץ-מטה+Ctrl+Alt והשורה תוכנס עבורך.
2. בחר בשלושת התאים של השורה שהוספת זה עתה, ומזג אותם על ידי לחיצה ימנית, ובחר Modify ← Merge Cells כדי ליצור תא אחד שמשתרע על פני שלוש העמודות.
3. גרור את הפקד ValidationSummary מהקטגוריה Validation של ארגז הכלים אל התא החדש שיצרת.
4. גרור פקד CustomValidator אל התא הריק שלאחר תיבת הטקסט עבור מספר הטלפון הביתי, והגדר לו את המאפיינים הבאים:

מאפיין	ערך
Display	Dynamic
ErrorMessage	Please enter your home or business phone number
Text	*
ClientValidationFunction	ValidatePhoneNumbers

5. הישאר בטבלת המאפיינים של הפקד CustomValidator, לחץ על הלחצן עם סימן הברק כדי לעבור למצב Events. לחץ לחיצה כפולה על האירוע ServerValidate כדי לגרום ל-WVD לכתוב עבורך מטפל אירוע. הוסף למטפל את הקוד הבא:

VB.NET

```
Protected Sub CustomValidator1_ServerValidate(ByVal source As Object, _
    ByVal args As System.Web.UI.WebControls.ServerValidateEventArgs) _
    Handles CustomValidator1.ServerValidate
    If txtPhoneHome.Text IsNot String.Empty _
        Or txtPhoneBusiness.Text IsNot String.Empty Then
        args.IsValid = True
    Else
        args.IsValid = False
    End If
End Sub
```

C#

```
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args)
{
    if (txtPhoneHome.Text != string.Empty || txtPhoneBusiness.Text != string.Empty)
    {
        args.IsValid = true;
    }
    else
    {
        args.IsValid = false;
    }
}
```

6. עבור ל- Markup View של פקד המשתמש והוסף את הבלוק הבא של קוד JavaScript מייד לפני הטבלה עם הפקדים:

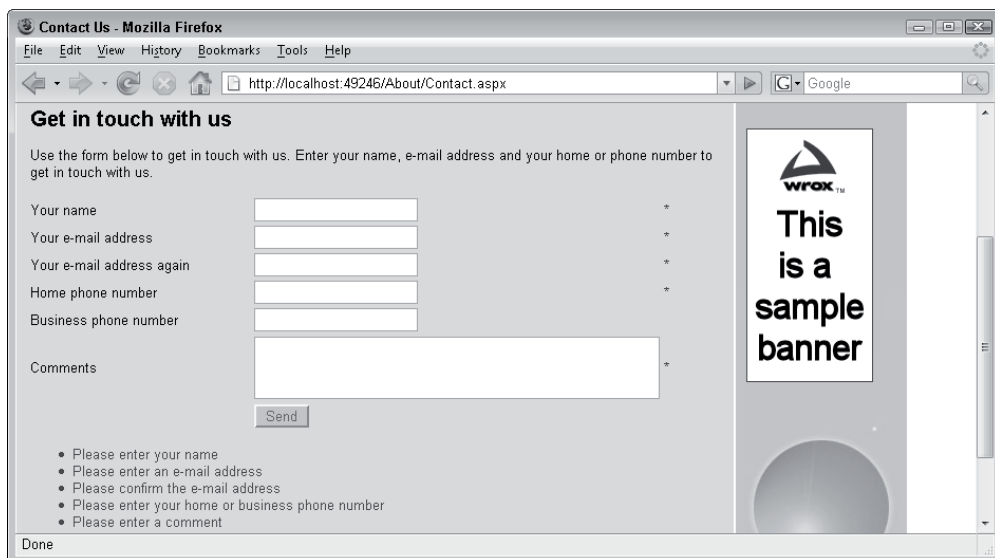
```
<script type="text/javascript">
    function ValidatePhoneNumbers(source, args)
    {
        var txtPhoneHome = document.getElementById('<%= txtPhoneHome.ClientID %>');
        var txtPhoneBusiness = document.getElementById('<%= txtPhoneBusiness.ClientID %>');
        if (txtPhoneHome.value != '' || txtPhoneBusiness.value != '')
        {
            args.IsValid = true;
        }
        else
        {
            args.IsValid = false;
        }
    }
</script>
<table class="style1">
```

7. שמור על כל השינויים על ידי הקשת Ctrl+Shift+S, ואז בקש את הדף Contact.aspx בדפדפן. שים לב שאינך יכול להגיש את הטופס אם לא הזנת לפחות אחד משני מספרי הטלפון. שים גם לב שהפקד ValidationSummary מציג רשימה של כל הבעיות שנובעות מהנתונים שהזנו לטופס. הפונקציה ValidatePhoneNumbers של צד-לקוח (כתובה ב-JavaScript) כעת שיוון מספר טלפון אחד לפחות לפני שתוכל להחזיר את הדף לשרת. תרשים 8-9 מראה איך הדף מופיע בדפדפן Mozilla Firefox.

8. חזור ל-VWD ולחץ על הפקד ValidationSummary ב- Design View. בטבלת המאפיינים שנה את ShowMessageBox ל-True, ואת ShowSummary שנה ל-False. כמו כן, קבע את המאפיין HeaderText של הפקד למחרוזת הבאה:

Please correct the following errors before you press the Send button

9. פתח שוב את הדף בדפדפן ולחץ על הלחצן Send פעם נוספת. שים לב שאתה מקבל כעת התרעת צד-לקוח ולא את הרשימה הכלולה עם השגיאות אשר מוצגת בתרשים 9-9. ה-HeaderText של הפקד ValidationSummary מקדים את רשימת השגיאות.



תרשים 9-8



תרשים 9-9

כיצד זה פועל

כשהוספת את הפקד CustomValidator, הגדרת שני מטפלי איור, אחד עבור הלקוח ואחד עבור בדיקת אימות בצד-שרת. שניהם מופיעים בקטע הקוד הבא באותיות מודגשות:

```
<asp:CustomValidator ID="CustomValidator1" runat="server"
    ErrorMessage="Please enter your home or business phone number"
    ClientValidationFunction="ValidatePhoneNumbers"
    OnServerValidate="CustomValidator1_ServerValidate"
    Display="Dynamic">*</asp:CustomValidator>
```

כשתשתמש ב-VB.NET לא תראה את התכונה OnServerValidate, מכיוון שהיא מוגדרת בקוד ברקע תוך שימוש במילת המפתח Handles.

כשאתה לוחץ על הלחצן Send, מופעלת בצד הלקוח הפונקציה ValidatePhoneNumbers (כתובה ב-JavaScript) שקבעת ב- ClientValidationFunction. פונקציה זו, שהוגדרה בקטע סימני העריכה של פקד המשתמש, מכילה התייחסויות לתיבות הטקסט של מספרי הטלפון:

```
var txtPhoneHome = document.getElementById('<%= txtPhoneHome.ClientID %>');
var txtPhoneBusiness = document.getElementById('<%= txtPhoneBusiness.ClientID %>');
```

שים לב שהקוד txtPhoneBusiness.ClientID עטוף בבלוק צד-שרת <%= %>. קוד זה מתבצע בשרת ומחזיר לשרת את ה-ClientID של הפקד. כשתסתכל בקוד HTML של הדף Contact שבדפדפן, תמצא את הקוד הבא:

```
function ValidatePhoneNumbers(source, args)
{
    var txtPhoneHome =
        document.getElementById('ctl00_cpMainContent_ContactForm1_txtPhoneHome');
    var txtPhoneBusiness =
        document.getElementById('ctl00_cpMainContent_ContactForm1_txtPhoneBusiness');
    if (txtPhoneHome.value != '' || txtPhoneBusiness.value != '')
```

אפשר לראות כאן שמאפיני ClientID של הפקדים בצד השרת הפכו למאפיני id של הלקוחות שלהם. זהו פתרון טוב בהרבה מאשר לקודד את תכונות id של תיבות הטקסט באופן קשיח ב-HTML הסופי, מכיוון ש-ASP.NET runtime עלול לשנות אותן בקלות. בפרק הקודם ראינו את ולמה זה קורה. בשלב זה, הקוד קורא לפונקציה getElementById (כתובה ב-JavaScript) של האובייקט document כדי לקבל הפניה (ב-JavaScript) לתיבת הטקסט. הקוד בוחן את מאפיני הערך של שני פקדי תיבות הטקסט האלה. אם אחד מהערכים אינו מחרוזת ריקה, האימות מצליח. אבל איך מדווחת השיטה ValidatePhoneNumbers חזרה למנגנון האימות שהאימות הצליח, או לא? כשמנגנון האימות של ASP.NET קורא לשיטה ValidatePhoneNumbers הוא מעביר לה שני ארגומנטים, את source שהוא הפניה ל-CustomValidator הממשי ב-HTML, ואת args. האובייקט args חושף מאפיין IsValid שמאפשר לקבוע אם האימות הצליח או נכשל:

```
if (txtPhoneHome.value != '' || txtPhoneBusiness.value != '')
{
    args.IsValid = true;
}
else
{
    args.IsValid = false;
}
```

עם קוד זה, כאשר שתי תיבות הטקסט ריקות, IsValid מוגדר כ-false, כדי שהאימות לא יצליח ויעצור את הגשת הטופס. כשאחת מתיבות הטקסט מכילה ערך, IsValid מוגדר כ-true. בדוגמה זו לא השתמשנו בארגומנט source, אבל תוכל להשתמש בו לסימון, או לשינוי, פקד האימות בהתבסס על תקפות הארגומנט או אי-תקפותו.

בשרת, הפקד CustomValidator קורא לשיטת אימות של צד השרת, שמבצעת את אותה בדיקה.

VB.NET

```
If txtPhoneHome.Text IsNot String.Empty Or txtPhoneBusiness.Text IsNot String.Empty  
Then  
    args.IsValid = True  
Else  
    args.IsValid = False  
End If
```

C#

```
if (txtPhoneHome.Text != string.Empty || txtPhoneBusiness.Text != string.Empty)  
{  
    args.IsValid = true;  
}  
else  
{  
    args.IsValid = false;  
}
```

בדיקת הנתונים הן בצד הלקוח והן בצד השרת מבטיחה שהמערכת תקלוט נתונים תקפים בלבד. גם אם הדפדפן אינו תומך ב-JavaScript (אולי מפני שהמשתמש כבה זאת בכוונה) הנתונים שלך עדיין נבדקים בשרת. עם זאת, חשוב להבין שעדיין צריך לבדוק את תקפות הדף לפני שפועלים עם הנתונים שהוגשו לו. זה מתבצע על ידי בדיקת המאפיין IsValid של הדף.

VB.NET

```
If Page.IsValid Then  
    ' OK to proceed  
End if
```

C#

```
if (Page.IsValid)  
{  
    // OK to proceed  
}
```

המאפיין IsValid מחזיר True כשכל הפקדים בדף, או המאפיין האקטיבי ValidationGroup, תקפים. בדיקת המאפיין IsValid בשרת, לפני שתפעל על הנתונים, תוכל להבטיח שהנתונים מאומתים בהתאם לפקדי האימות שלך, גם אם המשתמש כיבה את JavaScript בדפדפן, ושלח את הטופס לשרת ללא בדיקות צד-לקוח כלשהן. כשנדון בשליחת דואר אלקטרוני, מאוחר יותר בפרק זה, תראה שוב את השימוש במאפיין IsValid.

עיבוד נתונים בשרת

הנתונים שהמשתמש מזין לטפסי האינטרנט אינם בדרך כלל המידע היחיד שהופך את האתר למערכת אינטראקטיבית מונעת-נתונים (data-driven). ברוב אתרי האינטרנט המידע מגיע ממקורות רבים כמו בסיסי נתונים, טקסט, קבצי XML ושירותי web. בנוסף, יש גם נתונים

שיוצאים מהמערכת. אתה עשוי לרצות לשלוח דואר אלקטרוני לבעל האתר בכל פעם שמישהו שלח מידע באמצעות דף "צור קשר", או ייתכן שתרצה להודיע לאנשים בכל פעם שתוסיף לאתר תכונה חדשה או סקירה (ביקורת). עבור תרחישים כגון אלה חשוב להבין כיצד ASP.NET 3.5 מאפשרת לשלוח דואר אלקטרוני. נושא זה נדון בסעיף הבא.

שליחת דואר אלקטרוני מאתר האינטרנט שלך

כתיבת קוד ששולח דואר אלקטרוני מדף ASP.NET היא פעולה ישירה וברורה למדי. בתוך מרחב השמות System.Net.Mail מצויים כמה אובייקטים שמקלים מאוד על יצירת ושליחת הודעות דואר אלקטרוני. אובייקטים אלה מאפשרים ליצור הודעות חדשות; להוסיף כתובות בשדות To, CC, ו-BCC; להוסיף קבצים מצורפים (attachments) כמו קבצי נתונים או תמונות; וכמובן, לשלוח את ההודעות.

הטבלה הבאה מתארת ארבעה אובייקטים שמשתמשים בהם לרוב כשעוסקים בשליחת דואר אלקטרוני מיישום ASP.NET.

מחלקה	תיאור
MailMessage	מחלקה זו מייצגת את ההודעה שאתה עומד לשלוח. יש לה מאפיינים כמו Subject ו-Body שמגדיר את תכני ההודעה; מאפיינים To, CC, ו-BCC לקביעת הכתובות; ואוסף בשם Attachments כדי לצרף קבצים להודעה.
MailAddress	מחלקה זו מייצגת את כתובת השולח או כתובת הנמען של הדואר האלקטרוני. יש לה כמה פונקציות בנייה שמאפשרות לקבוע את כתובת הדואר האלקטרוני ולהציג שם.
Attachment	מחלקה זו מייצגת קבצים שניתן לצרפם ל-MailMessage. כשאתה בונה מופע של Attachment, באפשרותך להעביר לו את שם הקובץ שתוצא לשלוח. את הקובץ המצורף מוסיפים ל-MailMessage תוך שימוש בשיטה Add של האוסף Attachments של המחלקה.
SmtplibClient	מחלקה זו משמשת לשליחת ההודעה הממשית. כברירת מחדל, מופע של מחלקה זו בודק את הקובץ web.config עבור הגדרות כמו שרת ה-SMTP, שם משתמש וסיסמה שמשמשת לשליחת הדואר האלקטרוני.

קביעת תצורת האתר לצורך שליחת דואר אלקטרוני

הקוד לשליחת דואר אלקטרוני הוא אמנם קל למדי, אבל קביעת תצורת היישום והרשת עלולה להיות לעתים קרובות קשה יותר. למנגנון שמשתמשים בו לשליחת הדואר האלקטרוני חייבת להיות גישה לשרת SMTP, שיכול להיות זמין באופן מקומי ברשת שלך או ברחבי האינטרנט. ברוב המקרים עליך להשתמש בשרת SMTP שבו משתמש גם לקוח הדואר האלקטרוני שלך.

(למשל, Outlook של מיקרוסופט). במקרה שיש לך ספקות לגבי שרת SMTP שלך, התקשר עם מנהל הרשת שלך או עם ספק השירות (ISP).

כשבידך כתובת שרת SMTP, ביכולתך לקבוע תצורה גלובלית בקובץ web.config שבאלמנט <system.net>. כשתשתמש בשרת SMTP של ספק השירות שלך, הגדרות התצורה ייראו בדומה לכך:

```
<system.net>
  <mailSettings>
    <smtp deliveryMethod="Network" from="Your Name &lt;you@yourprovider.Com>";">
      <network host="smtp.yourprovider.com" />
    </smtp>
  </mailSettings>
</system.net>
</configuration>
```

את האלמנט <system.net> חייבים להוסיף כבן ישיר של <configuration>, אלמנט השורש של הקובץ web.config. בתוך <system.net> אתה מוסיף אלמנט <mailSettings>, שבתורו מכיל אלמנט <smtp>. לבסוף, האלמנט <smtp> הוא בעל המאפיין host שמצביע אל שרת SMTP שלך.

לאלמנט <smtp> יש מאפיין רשות from שמאפשר לקבוע את השם וכתובת הדואר האלקטרוני של השולח בתבנית שם הדוא"ל <E-mail Address>. מכיוון שלתווים < ו- > יש משמעות מיוחדת ב-XML, עליך להיעזר בתווים המייצגים שלהם: < (עבור '<') ו- > (עבור '>'). בתרגיל הבא תלמד כיצד לדרוס כתובת from זו בזמן שליחת דואר אלקטרוני באמצעות כתיבת קוד.

אם ספק השירות דורש אימות לפני שליחת דואר אלקטרוני, תוכל להוסיף את התכונות של שם משתמש וסיסמה:

```
<smtp deliveryMethod="Network">
  <network host="smtp.yourprovider.com" userName="UserName" password="Password"/>
</smtp>
```

למידע נוסף על ההגדרות השונות עבור האלמנט <system.net>, עיין בתיעוד MSDN המקוון.

יצירת הודעות דואר אלקטרוני

כדי ליצור ולשלוח הודעת דואר אלקטרוני, עליך לבצע ארבעה צעדים. תחילה, עליך ליצור מופע של המחלקה MailMessage. אחר כך עליך לקבוע את תצורת ההודעה על ידי הוספת גוף ונושא. הצעד הבא הוא לספק מידע אודות השולח ואודות נמעני ההודעה. לבסוף, עליך ליצור מופע של המחלקה SmtpClient כדי לשלוח את ההודעה. התרגיל הבא מראה כיצד לקודד את ארבעת הצעדים האלה.

בתרגיל זה תיצור בתיקייה Demos דף פשוט. הקוד בדף זה יוצר הודעת דואר אלקטרוני שנשלחת ברגע שהדף נטען. בתרגיל מאוחר יותר תשנה את טופס יצירת קשר כדי שהמשתמש יוכל לשלוח את תגובתו בדואר אלקטרוני.

1. בתיקייה Demos, צור קובץ חדש בשם Email.aspx. ודא שהוא מבוסס על תבנית דף הבסיס שלך, באופן שיש לו את הדף הראשי הנכון והוא יורש מ-BasePage באופן אוטומטי. שנה את ה-Title של הדף ל- E-mail Demo.
2. עבור לקוד ברקע על ידי הקשת F7. הוסף את המשפט הבא בראש הקובץ, לפני הגדרת המחלקה. תפקידו לגרום לכך שהמחלקות במרחב השמות System.Net.Mail תהיינה זמינות לקוד שלך:

VB.NET

```
Imports System.Net.Mail
```

C#

```
using System.Net.Mail;
```

3. הוסף את הקוד הבא למטפל Page_Load. כשאתה כותב ב-VB.NET עליך להגדיר תחילה את המטפל, תוך שימוש בשתי הרשימות הנפתחות שבראש חלון המסמך:

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Me.Load
```

```
    Dim myMessage As MailMessage = New MailMessage()  
    myMessage.Subject = "Test Message"  
    myMessage.Body = "Hello world, from Planet Wrox"  
    myMessage.From = New MailAddress("you@yourprovider.com", "Sender Name Here")  
    myMessage.To.Add(New MailAddress("you@yourprovider.com", "Receiver Name Here"))  
  
    Dim mySmtpClient As SmtpClient = New SmtpClient()  
    mySmtpClient.Send(myMessage)
```

```
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
```

```
{
```

```
    MailMessage myMessage = new MailMessage();  
    myMessage.Subject = "Test Message";  
    myMessage.Body = "Hello world, from Planet Wrox";  
  
    myMessage.From = new MailAddress("you@yourprovider.com", "Sender Name Here");  
    myMessage.To.Add(new MailAddress("you@yourprovider.com", "Receiver Name Here"));  
  
    SmtpClient mySmtpClient = new SmtpClient();  
    mySmtpClient.Send(myMessage);  
}
```

שנה לכתובות רצויות את כתובות הדואר האלקטרוני ואת השמות שמופיעים בשתי השורות שמגדירות את הכתובות From ו-To. אם יש לך כתובת אחת בלבד של דואר אלקטרוני, השתמש באותה הכתובת עבור השולח והנמען.

4. פתח את web.config וגלול כל הדרך מטה. הוסף את ההגדרות הבאות ממש לפני תגית הסגירה </configuration>:

```
<system.net>
  <mailSettings>
    <smtp deliveryMethod="Network" from="Your Name &lt;you@yourprovider.Com>";">
      <network host="smtp.yourprovider.com" />
    </smtp>
  </mailSettings>
</system.net>
```

</configuration>

אל תשכח לשנות את smtp.yourprovider.com לשם שרת SMTP שלך. ודא גם שבמאפיין from אכן הכנסת את שמך ואת כתובת הדואר האלקטרוני שלך. אם נחוץ, הוסף את המאפיינים userName ו-password לאלמנט <network> כפי שהראינו קודם.

5. שמור את כל השינויים. בקש את הדף Email.aspx בדפדפן על ידי הקשת Ctrl+F5. לאחר זמן מה אתה אמור לקבל הודעת דואר אלקטרוני בכתובת שציינת בצעד 3 של תרגיל זה.

הערה: אם תקבל הודעת שגיאה, תוכל לבדוק כמה דברים. תחילה, ודא שהכנסת ב-web.config כתובת נכונה של שרת SMTP. ייתכן שתצטרך להתקשר אל ספק האינטרנט שלך או אל מנהל הרשת כדי לקבל את הכתובת הנכונה, ואולי גם לקבוע שם משתמש וסיסמה. ודא גם ששרת הדואר שאתה משתמש בו אכן מאפשר לך לשלוח הודעות. לבסוף, אם תקבל הודעה כמו "The SMTP server requires a secure connection or the client was not authenticated." ספק האינטרנט שלך עשוי לדרוש ממך להשתמש בהתקשרות מאובטחת. אם זה המקרה, בדוק את שם המשתמש והסיסמה ב-web.config, או נסה להגדיר את המאפיין EnableSsl של האובייקט mySmtpClient כ-True על ידי הקוד:

VB.NET

```
mySmtpClient.EnableSsl = True
```

C#

```
mySmtpClient.EnableSsl = true;
```

כיצד זה פועל

לקובץ הקוד ברקע הוספת את המשפט Imports או את המשפט using הבא:

VB.NET

```
Imports System.Net.Mail
```

C#

```
using System.Net.Mail;
```

משפט זה גורם לכך שהמחלקות במרחב שמות זה תהיינה זמינות לקוד שלך מבלי שיהיה צורך להוסיף להן כתחילית את השם המלא של מרחב השמות שלהן. הדבר מאפשר, לדוגמה, ליצור מופע של MailMessage בדרך זו:

VB.NET

```
Dim myMessage As MailMessage = New MailMessage()
```

C#

```
MailMessage myMessage = new MailMessage();
```

ללא משפט Imports או using, היית נזקק לקוד רב יותר, כמו זה:

VB.NET

```
Dim myMessage As System.Net.Mail.MailMessage = New System.Net.Mail.MailMessage()
```

C#

```
System.Net.Mail.MailMessage myMessage = new System.Net.Mail.MailMessage();
```

לאחר משפט Imports או using, הקוד יוצר אובייקט MailMessage חדש וקובע את המאפיינים Subject ו-Body שלו. אחר כך הקוד מקצה כתובות דואר אלקטרוני לשולח ולנמען.

VB.NET

```
myMessage.From = New MailAddress("you@yourprovider.com", "Sender Name Here")  
myMessage.To.Add(New MailAddress("you@yourprovider.com", "Receiver Name Here"))
```

C#

```
myMessage.From = new MailAddress("you@yourprovider.com", "Sender Name Here");  
myMessage.To.Add(new MailAddress("you@yourprovider.com", "Receiver Name Here"));
```

המאפיין From של MailMessage הוא מסוג MailAddress, ולכן ביכולתך להקצות כתובת דואר חדשה באופן ישיר. פונקציית הבנייה של המחלקה MailMessage מקבלת את כתובת הדואר האלקטרוני והשם כמחרוזות, ולכן תוכל ליצור ולהקצות את כתובת From בשורת קוד אחת.

המאפיין To של המחלקה MailMessage הוא אוסף, ולכן לא תוכל להקצות כתובת דואר (MailAddress) ישירות. במקום זאת תצטרך להשתמש בשיטה Add כדי להקצות כתובת. הדבר גם מאפשר מספר נמענים על ידי קריאה ל-To.Add מספר פעמים, כשבכל פעם מועבר מופע MailAddress של MailAddress. במאפיינים CC ו-BCC משתמשים בדרך דומה כדי להקצות כתובות דואר אלקטרוני לשדות "העתקים אל" (Carbon Copy) ו"העתקים אל סמויים" (Blind Carbon Copy) של הודעת הדואר האלקטרוני.

שתי השורות האחרונות של הקוד שולחות את ההודעה הממשית:

VB.NET

```
Dim mySmtpClient As SmtpClient = New SmtpClient()  
mySmtpClient.Send(myMessage)
```

C#

```
SmtpClient mySmtpClient = new SmtpClient();  
mySmtpClient.Send(myMessage);
```

כשהשיטה Send נקראת, המחלקה SmtpClient סורקת את הקובץ web.config עבור שרת SMTP שתצורתו הוגדרה. אחר כך היא יוצרת קשר עם שרת זה ושולחת את ההודעה.

פרט לפקדי האימות שראית עד כה, ASP.NET מכילה גם מגננון אימות אחר שבו נרדן להלן.

הבנת הצורך באימות של בקשה

על פי התכנון, דף ASP.NET מתריע על חריג (exception) בכל פעם שאחד הפקדים בדף מכיל תוכן שנראה כמו תגיות HTML. לדוגמה, את השגיאה המוצגת בתרשים 9-10 תראה כשתזין לתיבות הטקסט של השמות בדף "יצירת קשר" תוכן כמו `<h1>Hello World</h1>` או `<script type="text/javascript">alert('Hello World');</script>`.



תרשים 9-10

רכיב ASP.NET runtime עושה זאת במטרה למנוע ממשתמשים להכניס HTML או JavaScript שעלולים להפריע לעיצוב או לפעילות האתר. כשברצונך לאפשר למשתמשים להכניס HTML, בטל את אימות הבקשה על ידי קביעת התכונה ValidateRequest שבהנחיה @ Page למצב False.

```
<%@ Page ... Inherits="Contact" Title="Contact Us" ValidateRequest="False" %>
```

עם הגדרה זו כ-False, משתמשים יוכלו להכניס HTML מבלי לגרום להודעת שגיאה. כשתגדיר את ValidateRequest כ-False, עליך רק להיות בטוח שאמנם ברצונך לאפשר למשתמשים להכניס HTML.

בתרגיל נסה זאת הקודם, הטקסט עבור גוף הודעת הדואר האלקטרוני מקודד באופן קשיח. לא תמיד זהו הפתרון האפשרי הטוב ביותר, מכיוון שפירוש הדבר שבכל פעם שתצרצה לשנות את הטקסט תצטרך לסרוק ולשנות את הקוד בכל מקום שדרוש. לעתים קרובות עדיף להשתמש במקום זאת בתבנית מבוססת טקסט. בסעיף הבא תלמד כיצד לעשות זאת.

קריאה מקבצי טקסט

סביבת ASP.NET כוללת כמה מחלקות ושיטות שימושיות שמקלות מאוד על השימוש בקבצים. לדוגמה, המחלקה File, שנמצאת במרחב השמות System.IO, מאפשרת לקרוא מקבצים ולכתוב לקבצים, ליצור ולמחוק קבצים, ולהעביר קבצים ממקום למקום. מחלקה זו מכילה שיטות סטטיות בלבד, כלומר, אינך צריך ליצור תחילה מופע של המחלקה. במקום זאת אתה קורא ישירות לשיטות מהמחלקה File. לדוגמה, כדי לקרוא את מלוא התוכן של קובץ טקסט, תוכל להשתמש בקוד הבא:

VB.NET

```
Dim myContents As String = System.IO.File.ReadAllText("C:\MyFile.txt")
```

C#

```
string myContents = System.IO.File.ReadAllText(@"C:\MyFile.txt");
```

בדוגמה זו, ב-C#, נקבעת לשם הקובץ התחילית @ כדי למנוע את הצורך לקבוע לכל לוכסן אחורי (i) תחילית של לוכסן אחורי נוסף. הסיבה היא שב-C# יש ללוכסן אחורי משמעות מיוחדת, ולכן, אם משתמשים בו במחרוזת צריכים בדרך כלל להקדים לו לוכסן אחורי נוסף. השימוש בתו @ אומר למהדר שעליו להתייחס לכל לוכסן אחורי שימצא כאל תו, ולהתעלם מהמשמעות המיוחדת שלו.

הטבלה הבאה מפרטת את השיטות היותר נפוצות של המחלקה File שמאפשרות עבודה עם קבצים.

שיטות	תיאור השיטה
AppendAllText	צירוף של מחרוזת מסוימת אל קובץ טקסט. אם הקובץ אינו קיים, השיטה יוצרת את הקובץ.
Copy	העתקת קובץ ממקום אחד לאחר.
Delete	מחיקה מהדיסק של קובץ מסוים.
Exists	בדיקה אם קובץ מסוים קיים בדיסק.
Move	העברת קובץ מסוים למקום אחר.
WriteAllText	כתיבת תוכן של מחרוזת אל קובץ חדש שנוצר. אם קובץ היעד כבר קיים, תוכן המחרוזת ידרוס את הכתוב.

אפשר להשתמש בשיטות אלו למטרות שונות. לדוגמה, כשמשתמש מעלה קובץ לאתר, ניתן להשתמש בשיטה Move כדי להעביר את הקובץ לתיקייה אחרת. בדומה, כשתרצה להיפטר מקבצים שהועלו לאתר ושאינך עוד צורך בהם, תשתמש בשיטה Delete.

השיטה ReadAllText משמשת לקריאת מלוא התוכן של קובץ טקסט. לדוגמה, כששולחים טקסט באמצעות דואר אלקטרוני, אפשר לאחסן את גוף הטקסט של הודעת הדואר בקובץ טקסט. כשאתה עומד לשלוח את הדואר האלקטרוני, קרא ל-ReadAllText והקצה את התוכן שמחזירה שיטה זו לגוף הודעת הדואר. בתרגיל הבא תראה כיצד זה פועל.

נסה זאת שליחת דואר אלקטרוני מפקד המשתמש ContactForm

תרגיל זה מראה כיצד להשתמש בדואר אלקטרוני כדי לשלוח את נתוני המשתמש מטופס יצירת קשר אל תיבת הדואר הנכנס שלך. הקוד קורא קובץ טקסט שמכיל שומרי מקום, ומשתמש בו כגוף הודעת הדואר. הקוד ממלא את שומרי המקום בנתוני המשתמש הממשיים מהטופס, ואז שולח אותם בדואר האלקטרוני.

1. צור קובץ חדש בתיקייה App_Data שבאתר. אם התיקייה כבר אינה קיימת, לחץ לחיצה ימנית על האתר ובחר Add ASP.NET Folder ← App_Data. צור את קובץ הטקסט על ידי לחיצה ימנית על התיקייה App_Data ובחירת Add New Item. קרא לקובץ הטקסט בשם ContactForm.txt.
2. הקלד את הטקסט הבא בקובץ הטקסט, כולל שומרי המקום כשהם עטופים בזוגות של התו סולמית (#):

```
Hi there,  
  
A user has left the following feedback at the site:  
  
Name:          ##Name##  
E-mail address: ##Email##  
Home phone:    ##HomePhone##  
Business phone: ##BusinessPhone##  
Comments:     ##Comments##
```

3. פתח את הקוד ברקע של פקד המשתמש ContactForm.ascx והוסף את מרחב השמות הבא לראש הקובץ:

VB.NET

```
Imports System.Net.Mail
```

```
Partial Class Controls_ContactForm  
    Inherits System.Web.UI.UserControl
```

C#

```
using System.Net.Mail;
```

```
public partial class Controls_ContactForm : System.Web.UI.UserControl
```

4. עבור למצב Markup View והוסף את המאפיין `runat="server"` ואת המזהה `id="FormTable"` לטבלה עם פקדי השרת. בדרך זו תוכל להסתיר את הטבלה כולה לאחר הגשת הטופס. כדי לבצע זאת, אתר את תגית הפתיחה של הטבלה ושנה אותה כך:

```
<table class="style1" runat="server" id="FormTable">
```

5. גלול מטה עד לקצה הקובץ, ומייד לאחר תגית הסגירה `</table>` הוסף תווית בשם `lblMessage`. הגדר את המאפיין `Text` של התווית ל- `Message Sent`. הסתר את התווית על ידי הגדרת המאפיין `Visible` לערך `false`:

```
</table>
```

```
<asp:Label ID="lblMessage" runat="server" Text="Message Sent" Visible="false"></asp:Label>
```

6. העבר את הפקד לתצוגת Design View והגדר את `ShowSummary` של `ValidationSummary` חזרה כ-`True`, ואת `ShowMessageBox` הגדר כ-`False`. עכשיו לחץ לחיצה כפולה על הלחצן `Send`. במטפל האירוע שהוסיפה עבורך `VWD` הוסף את הקוד הבא:

VB.NET

```
Protected Sub btnSend_Click(sender As Object, e As EventArgs) Handles btnSend.Click
```

```
    If Page.IsValid Then
```

```
        Dim fileName As String = Server.MapPath("~/App_Data/ContactForm.txt")
```

```
        Dim mailBody As String = System.IO.File.ReadAllText(fileName)
```

```
        mailBody = mailBody.Replace("##Name##", txtName.Text)
```

```
        mailBody = mailBody.Replace("##Email##", txtEmailAddress.Text)
```

```
        mailBody = mailBody.Replace("##HomePhone##", txtPhoneHome.Text)
```

```
        mailBody = mailBody.Replace("##BusinessPhone##", txtPhoneBusiness.Text)
```

```
        mailBody = mailBody.Replace("##Comments##", txtComments.Text)
```

```
        Dim myMessage As MailMessage = New MailMessage()
```

```
        myMessage.Subject = "Response from web site"
```

```
        myMessage.Body = mailBody
```

```
        myMessage.From = New MailAddress("you@yourprovider.com", "Sender Name Here")
```

```
        myMessage.To.Add(New MailAddress("you@yourprovider.com", "Receiver Name Here"))
```

```
        Dim mySmtpClient As SmtpClient = New SmtpClient()
```

```
        mySmtpClient.Send(myMessage)
```

```
        lblMessage.Visible = True
```

```
        FormTable.Visible = False
```

```
    End If
```

```
End Sub
```

C#

```
protected void btnSend_Click(object sender, EventArgs e)
```

```
{
```

```
    if (Page.IsValid)
```

```

{
    string fileName = Server.MapPath("~/App_Data/ContactForm.txt");
    string mailBody = System.IO.File.ReadAllText(fileName);

    mailBody = mailBody.Replace("##Name##", txtName.Text);
    mailBody = mailBody.Replace("##Email##", txtEmailAddress.Text);
    mailBody = mailBody.Replace("##HomePhone##", txtPhoneHome.Text);
    mailBody = mailBody.Replace("##BusinessPhone##", txtPhoneBusiness.Text);
    mailBody = mailBody.Replace("##Comments##", txtComments.Text);

    MailMessage myMessage = new MailMessage();
    myMessage.Subject = "Response from web site";
    myMessage.Body = mailBody;

    myMessage.From = new MailAddress("you@yourprovider.com", "Sender Name Here");
    myMessage.To.Add(new MailAddress("you@yourprovider.com", "Receiver Name Here"));

    SmtplibClient mySmtplibClient = new SmtplibClient();
    mySmtplibClient.Send(myMessage);

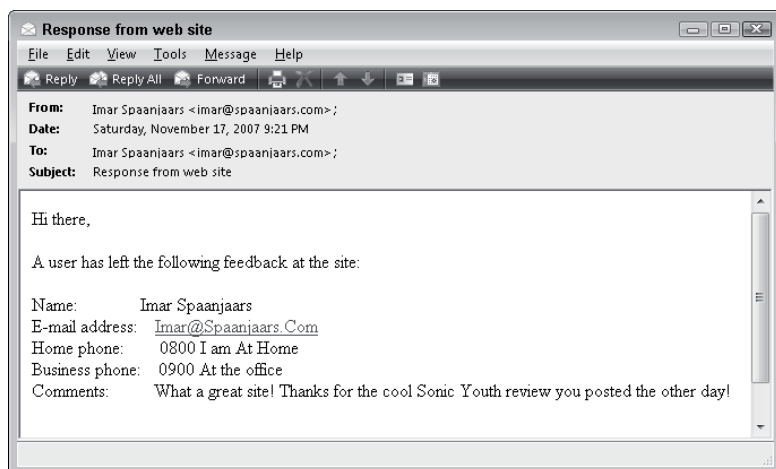
    lblMessage.Visible = true;
    FormTable.Visible = false;
}
}

```

גם כאן, ודא שהחלפת באלו שלך את כתובות הדואר עבור המאפיינים From ו-To של MailMessage.

7. שמור את כל השינויים, ופעם נוספת בקש את Contact.aspx בדפדפן. הקלד את פרטיך ולחץ על הלחצן Send. תראה שמופיע הטקסט Message Sent.

8. לחץ על חשבון הדואר ששלחת אליו את ההודעה, ולאחר זמן מה תקבל הודעת דואר דומה לזו שמוצגת בתרשים 9-11.



תרשים 9-11

כיצד זה פועל

חלק התרגיל שעסק בשליחת דואר אלקטרוני דומה למדי לדף ההדגמה שיצרת קודם. השוני, עם זאת, הוא במקור גוף הטקסט של הודעת הדואר. במקום לקודד את גוף הטקסט באופן קשיח בפקד `ContactForm`, העברת את הטקסט לקובץ נפרד. קובץ זה בתורו מכיל כמה שומרי מקום שמוחלפים בזמן ביצוע בפרטי המשתמש. כדי לקרוא פנימה את כל הקובץ "במכה", השתמשת בקוד הבא:

VB.NET

```
Dim fileName As String = Server.MapPath("~/App_Data/ContactForm.txt")
Dim mailBody As String = System.IO.File.ReadAllText(fileName)
```

C#

```
string fileName = Server.MapPath("~/App_Data/ContactForm.txt");
string mailBody = System.IO.File.ReadAllText(fileName);
```

השורה הראשונה משתמשת ב-`Server.MapPath` כדי לתרגם נתיב וירטואלי למקבילו הפיזי. שימוש בנתיב וירטואלי מקל על העברת האתר למקום אחר, מכיוון שהוא אינו תלוי בנתיבים כלשהם שמקודדים באופן קשיח.

`C:\BegASPNET\Site\` כמו `Server.MapPath("~/App_Data/ContactForm.txt")` מחזיר נתיב פיזי. `App_Data\ContactForm.txt` זה מוזן לשיטה `ReadAllText` של המחלקה `File` אשר פותחת את הקובץ ומחזירה את תוכנו, אשר מוקצה למשתנה `mailBody`. הקוד משתמש כעת בכמה קריאות לשיטה `Replace` כדי להחליף את שומרי המקום הסטטיים שבגוף ההודעה בפרטים שהמשתמש רשם בטופס יצירת קשר. הערך המוחזר של השיטה `Replace` - הטקסט החדש עם המחרוזות המוחלפות - מוקצה מחדש למשתנה `mailBody`. לאחר הקריאה הסופית ל-`Replace`, גוף ההודעה (`mailBody`) אינו מכיל עוד את שומרי המקום, אלא מכיל במקומם את פרטי המשתמש.

VB.NET

```
mailBody = mailBody.Replace("##Name##", txtName.Text)
...
mailBody = mailBody.Replace("##Comments##", txtComments.Text)
```

C#

```
mailBody = mailBody.Replace("##Name##", txtName.Text);
...
mailBody = mailBody.Replace("##Comments##", txtComments.Text);
```

השיטה `Replace` רגישה לרישיות ולכן, אם תמצא שכמה משומרי המקום לא הוחלפו כראוי, ודא שאתה משתמש באותן אותיות רישיות בקוד ובגוף ההודעה.

שומרי המקום עטופים בזוג של כפולות התו סולמית (`##`). התו סולמית נבחר באופן שרירותי, אבל עוזר לזהות את שומרי המקום, ומקטין את הסיכון להחלפה מקרית של טקסט שאמור להופיע בהודעה הממשית.

ברגע שהוגדר גוף הטקסט, הוא מוקצה לאובייקט MailMessage, שנשלח באמצעות SmtClient. מהלך זה דומה למה שראית בתרגיל קודם.

כשרשמת את פרטיך בטופס יצירת קשר ולחצת על הלחצן Send, ייתכן ששמת לב לחוסר יציבות (הבהוב) מסוים של הדף, כשהדף מועבר אל השרת ואז נטען מחדש עם הודעת ההצלחה. ניתן בקלות למזער, או אפילו לסלק לחלוטין, הבהוב זה של הדף תוך שימוש בטכנולוגיות Ajax, שיידונו בפרק הבא.

עצות מעשיות לאימות נתונים

הרשימה הבאה מספקת כמה עצות מעשיות בנושא אימות נתונים (data validation):

- ⊙ חובה עליך לערוך תמיד בדיקות אימות ותקפות של הקלט מהמשתמש. כשיש לך אתר ציבורי באינטרנט אין לך כל אפשרות לשלוט במשתמשים ולעצור משתמשים חורשי-רע מלהזין נתונים לא אמינים למערכת. כדי למנוע פגיעה באתר, עליך לבצע תמיד בדיקות אימות ותקפות של הקלט מהמשתמשים שלך על ידי שימוש בפקדי האימות של ASP.NET.
- ⊙ פקדי האימות צריכים לספק תמיד הודעות שגיאה שימושיות. באפשרותך להשאיר את המאפיין Text ריק ולהקצות את הודעת השגיאה למאפיין ErrorMessage, או להשתמש בפקד ValidationSummary כדי להציג רשימה של הודעות שגיאה.
- ⊙ שקול שימוש במאפיין CssClass של פקדי האימות כדי להעביר את הגדרות הסגנון עבור הודעות השגיאה לקובץ CSS נפרד. אם משתמשים ב-CssClass יש לזכור לנקות את המאפיין ForeColor של הפקד המתקף, אחרת תמשיך הודעת השגיאה להופיע באדום.
- ⊙ כשאתה כותב קוד ששולח הודעות דואר אלקטרוני, שקול את העברת גוף ההודעה לקובץ טקסט נפרד. כפי שראית, הקוד לקריאת הקובץ הוא בן שורה אחת, והיישום יהיה קל יותר לתחזוקה. במקום התקדמות קשה ואיטית דרך הקוד בחיפוש אחר גוף הטקסט, שנה את התבנית בתיקייה App_Data והקוד יחיל את השינויים באופן אוטומטי.
- ⊙ כשמאחסנים נתונים בקבצי טקסט או XML, אחסן אותם תמיד בתיקייה App_Data שמעוצבת במיוחד למטרה זו. בדרך זו יהיו כל קבצי הנתונים שלך ארוזים ביחד בצורה נאה. חשוב מכך, שרת האינטרנט חוסם את הגישה לקבצים בתיקייה זו באופן שמבקר באתר אינו יכול לבקש אותם ישירות.

קלט משתמש הוא היבט חשוב של רוב האתרים האינטראקטיביים באינטרנט. הקלט מגיע ממקורות שונים באתר שלך, מטופס יצירת קשר שערכת בפרק זה, ממחרוזת השאילתה וממקורות אחרים. כדי לעצור משתמשים מלהזין למערכת תוכן לא תקף או אפילו מסוכן, חשוב לערוך בדיקות אימות של כל הקלט לפני שפועלים עליו.

פקדי האימות ש-ASP.NET תומכת בהם מקלים על אימות כל הנתונים שמתקבלים מפקדי טופס כגון TextBox, DropDownList, Calendar וכד'. הפקד RequiredFieldValidator עוזר להבטיח שהמשתמש מילא את כל שדות החובה. הפקדים CompareValidator, RangeValidator ו-RegularExpressionValidator משמשים לבדיקת התוכן של הנתונים כנגד הגדרות שהוגדרו בזמן העיצוב. הפקד CustomValidator מקנה גמישות מרבית מכיוון שהוא מאפשר כתיבת קוד לבדיקות אימות שאינן כלולות בפקדי האימות המובנים. הפקד ValidationSummary משמש למתן משוב למשתמשים על שגיאות שביצעו בזמן הזנת הנתונים. הפקד יכול להציג את השגיאות ברשימת תבליטים פשוטה בדף, או כחלון התרעה ב-JavaScript.

התועלת הגדולה ביותר של פקדי האימות נובעת מכך שהם יכולים לפעול הן בצד-לקוח והן בצד-שרת. הדבר מאפשר יצירת טפסי תגובה שבהם המשתמש יקבל משוב מיידי על שגיאות כלשהן שביצע, ללא צורך ב-postback מלא. בו-בזמן עוברים הנתונים בדיקת אימות בשרת, שמבטיחה גם את תקפות הנתונים שבאים מלקוחות שאינם תומכים ב-JavaScript.

קיימות כמה אפשרויות לאחסון המידע שהמשתמשים מעבירים לאתר. המידע יכול להיות מאוחסן בבסיס נתונים, או בקובץ טקסט, או להישלח בדואר אלקטרוני. האפשרות האחרונה שימושית במיוחד עבור טפסים ליצירת קשר, כך שתקבל התרעה מיידית כשמישהו משאיר הערה באתר. הודות למחלקות במרחב השמות System.Net.Mail קל מאוד לשלוח דואר. מחלקות אלו מאפשרות לך ליצור הודעת דואר, להוסיף נושא, גוף, שולח ומידע על הנמען, ואז לשלוח את ההודעה תוך שימוש במחלקה SmtptClient.

כדי שהודעת דואר תהיה קלה לתחזוקה, מומלץ לאחסן את התוכן שלה בקובץ טקסט נפרד, ולא ישירות בקוד. אם תחליט מאוחר יותר לשנות את ההודעה, הרי כל מה שתצטרך לשנות הוא אותו קובץ טקסט יחיד. קריאת הקובץ מתבצעת בעזרת השיטות הסטטיות שבמחלקה System.IO.File. מחלקה זו מאפשרת לא רק לקרוא קבצים, אלא גם להשתמש בשיטות שלה כדי ליצור, להעתיק, להעביר ולמחוק קבצים באתר.

הוספת שומרי מקום להודעות הטקסט מאפשרת למלא אותם בקלות מתוך הקוד ששולח את הודעות הדואר. הפעלת השיטה Replace על מחרוזת הגוף מאפשרת להטמיע בהודעה את המידע שהוזן על ידי המשתמש לפני שתשלח אותה.

1. כדי שפקד המשתמש `ContactForm.ascx` יתאים עוד יותר לשימוש חוזר, תוכל ליצור עבורו מאפיין מסוג `PageDescription` שיאפשר לקבוע את שם הדף שמשתמש בפקד. הוסף את המחרוזת הזו להכרזת הפקד בדף שמכיל אותו. לבסוף, תוכל להוסיף את תוכן מאפיין זה לנושא של ההודעה שתשלח. בדרך זו תוכל לראות מאיזה דף נקרא טופס יצירת קשר. איזה קוד תצטרך לכתוב כדי לגרום לכל זה לקרות?
2. מדוע כה חשוב לבדוק את הערך של המאפיין `IsValid` של `Page` בזמן עיבוד נתונים? מה עלול לקרות אם תשכח לבצע בדיקה זו?
3. מה ההבדל בהתנהגות בין המאפיינים `To` ו-`From` של המחלקה `MailMessage`?
4. כשאתה מפעיל את הפקד `CustomValidator` אתה יכול לכתוב קוד של בדיקת אימות הן עבור צד הלקוח והן עבור צד השרת. איך אתה אומר ל-`ASP.NET runtime` איזו שיטת אימות של צד הלקוח צריך לקרוא במהלך האימות?
5. כיצד מנגנון האימות מזהה כי פעולת האימות הצליח או נכשלה כאשר אתה משתמש בפקד `CustomValidator`?

פרק 10

ASP.NET AJAX

הנושא "החם" בשנים האחרונות בתחום פיתוח web הוא, ללא ספק, **Ajax** (קיצור של **Asynchronous JavaScript And Xml**). למרות שהטכנולוגיות שמניעות את Ajax אינן חדשות ונמצאות בשוק כבר כמה שנים, הרי שרק בתחילת 2005 הן זכו לשם הרשמי. שימוש ב-Ajax מאפשר לדפי צד-לקוח להחליף נתונים עם השרת על ידי קריאות אסנכרוניות. קרוב לוודאי שהתכונה הפופולרית ביותר של Ajax היא נטרול ההבהוב שמתרחש עקב פעולת ה-postback, הדבר מתאפשר בזכות העברת נתונים חלקית לשרת מבלי לרענן את הדף כולו.

כדי לשפר את האתר עם תכונות Ajax, באפשרותך לבחור מבין כמה סביבות Ajax שונות. רבות מסביבות אלו מציעות מערכת של תכונות וכלים שכוללת סביבת JavaScript לצד-לקוח כדי לאפשר Ajax בדפדפן, קוד JavaScript לתקשורת עם השרת, ופקדי שרת שמיועדים לשילוב עם דפי ASP.NET. במערכת ASP.NET זמינות כמה סביבות שונות של Ajax, אבל הבולטת ביניהן היא **ASP.NET AJAX**, מכיוון שהיא מגיעה יחד עם סביבת העבודה ASP.NET 3.5 ו-Visual Web Developer ומכילה כמה פקדים נוספים ליצירת ממשקי משתמש עשירים ואינטראקטיביים.

בנוסף לחלק של Ajax שמבוסס על פקדי שרת, סביבת העבודה ASP.NET AJAX כוללת גם סביבת צד-לקוח עשירה. סביבת עבודה זו מאפשרת ל-JavaScript לתקשר עם השרת על ידי שירותי Web. היא גם מאפשרת גישה לדף צד-לקוח בשלמותו, תוך שימוש בדגם קוד אינטואיטיבי שפועל באופן שאינו תלוי בדפדפן היעד שלך.

בגרסאות קודמות של ASP.NET ושל Visual Web Developer הייתה Ajax זמינה כתוסף שנטען בנפרד. כיום משולבת Ajax לחלוטין ב-VWD וב-ASP.NET, והדבר מקל על הפעלתה.

בפרק זה תלמד בהרחבה על הנושאים הבאים הקשורים ל-Ajax:

- ⊙ שימוש בפקד `UpdatePanel` למניעת הבהוב דף (page flicker).
- ⊙ הבנת הפקד `ScriptManager` שמאפשר את הפונקציונליות של Ajax.
- ⊙ שימוש בפקד `UpdateProgress` כדי ליידע את המשתמשים לגבי התקדמות פעולת Ajax.
- ⊙ שימוש במזנקים (triggers) ובפקד `Timer` כדי לעורר את העדכון של פקדים מסוג `UpdatePanel`.
- ⊙ יצירת שירותי Web נגישים לתסריט (סקריפט) צד-לקוח.

עם סיום הלימוד בפרק זה, תכיר היטב את פקדי השרת השונים שמציעה סביבת העבודה ASP.NET AJAX ואת דרך הפעלתם. בנוסף, תהיה לך הבנה בסיסית על יצירת שירותי Web בעולם של ASP.NET, וכיצד ניתן לקרוא להם מקוד JavaScript של צד-לקוח תוך שימוש בסביבת העבודה Ajax של צד-לקוח.

מבוא ל-Ajax

המושגים שעומדים מאחורי Ajax אינם חדשים, וכבר ידועים מזה שנים. עוד מאז גרסת הדפדפן Internet Explorer 5 כללו הדפדפנים את האובייקט XMLHttpRequest שמאפשר ל-JavaScript לקרוא לשרת לצורך שליחה וקבלה של נתונים.

הדברים תפסו תאוצה כשהוכנס המונח Ajax לשימוש. במטרה להקדים את השאר, החלה מיקרוסופט בבניית ASP.NET AJAX, שהינה סביבת העבודה של Ajax שמשולבת כיום לחלוטין ב-ASP.NET וב-Visual Web Developer 2008. סביבת עבודה זו מציעה כמה תועלות שמתכנת אתרי אינטרנט יכול לנצל כדי ליצור יישומים שמגיבים היטב. ASP.NET AJAX מאפשרת במיוחד את הפעולות הבאות:

- ⊙ יצירת דפים חופשיים מהבהוב שמאפשרים רענון חלקי שלהם מבלי לטעון את הדף כולו מחדש, ומבלי להשפיע על חלקים אחרים שלו.
- ⊙ אספקת משוב למשתמשים תוך כדי רענון הדף.
- ⊙ עדכון קטעים בדף וקריאה לקוד צד-שרת בהתאם ללוח זמנים קבוע.
- ⊙ גישה לשירותי Web בצד-שרת, ועבודה עם הנתונים שהם מחזירים.
- ⊙ שימוש בסביבת העבודה העשירה של צד-לקוח כדי לגשת ולשנות אלמנטים בדף.

במהלך הפרק תלמד כיצד להשתמש בסביבת העבודה ASP.NET AJAX כדי ליצור יישומי אינטרנט בעלי ממשק משתמש עשיר וידידותי. Ajax עצמה היא נושא רחב שלא ניתן למיציא בפרק אחד. אם ברצונך לדעת יותר על ASP.NET AJAX, תמצא מידע רב ברשת, ובכתובת: www.asp.net/AJAX תוכל למצוא מאמרים וקטעי וידאו בנושא.

היפה ב-ASP.NET AJAX הוא שקל מאוד להתחיל להשתמש בתוכנה. יצירת דף חופשי מהבהוב היא עניין של גרירת כמה פקדים מארגז הכלים ושחרורם בדף. כשתבין את יסודות סביבת העבודה Ajax, תוכל להרחיב את הידע שלך על ידי התבוננות בנושאים מתקדמים יותר, כמו קריאה לשירותי Web ושימוש בסביבת העבודה העשירה של צד-לקוח לעבודה אינטגרטיבית עם הדף.

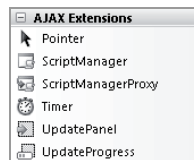
שימוש ב-ASP.NET Ajax בפרויקטים שלך

בגרסאות קודמות של VWD ושל ASP.NET AJAX היית צריך לשנות את הקובץ web.config באופן ידני כדי לאפשר את פעולת Ajax ביישום מבוסס ASP.NET. אם עבדת עם Ajax בעבר בפרויקטים של ASP.NET, תשמח ללמוד שהדבר אינו נחוץ עוד. בכל פעם שאתה יוצר אתר

אינטרנט חדש על ידי שימוש בתיבת הדו-שיח File ← New Web Site, האתר כבר מוגדר לעבודה מול Ajax, כלומר, אתה יכול להתחיל להשתמש ב-Ajax מייד. ארגז הכלים מכיל כיום קטגוריה נוספת שכוללת את פקדי Ajax הזמינים, ואפשר להשתמש בהם מייד מההתחלה.

יצירת דפים חופשיים מהבהוב

ניתן להשתמש בפקד השרת UpdatePanel כדי למנוע postbacks מלאים של דפי ASPX, וכדי לעדכן רק חלקים נבחרים בדף. כדי שפקד זה יפעל נכון, אתה זקוק גם לפקד ScriptManager. אם בכוונתך להשתמש בפונקציונליות של Ajax בדפי ASPX רבים, כדאי להציב את הפקד ScriptManager בדף הראשי. בדרך זו הוא יהיה זמין לכל הדפים שמבוססים על דף ראשי זה. פקד זה ופקדים אחרים שקשורים ל-Ajax, נמצאים בקטגוריה AJAX Extensions של ארגז הכלים, שמוצגת בתרשים 10-1. סמוך לסיום הפרק תקבל הסבר מפורט על ScriptManager, ותגלה מה הוא עושה ולמה הוא משמש.



תרשים 10-1

אם אינך רואה את הפקדים ScriptManager ו-UpdatePanel בארגז הכלים, לחץ לחיצה ימנית על הקטגוריה AJAX Extensions של ארגז הכלים ובחר Choose Items. אחר כך בחר בשני הפקדים מהרשימה שבכרטיסייה .NET Framework Components.

בשני הסעיפים הבאים תכיר את הפקדים ScriptManager ו-UpdatePanel. אחר כך תראה כיצד להשתמש בפקדים אלה בדפים שלך. בסעיפים מאוחרים יותר תכיר את הפקדים UpdateProgress, Timer ו-ScriptManagerProxy.

הפקד UpdatePanel

הפקד UpdatePanel הוא רכיב מפתח ביצירת דפים חופשיים מהבהוב. כדי ליישמו בצורה הבסיסית ביותר, עטוף את התוכן שברצונך לעדכן בפקד, והוסף לדף פקד ScriptManager, ובזה סיימת. בכל פעם שאחד הפקדים שבתוך UpdatePanel גורם postback לשרת, תתבצע פנייה חלקית לשרת ורק התוכן שרשום בתוך UpdatePanel יעודכן.

הבה נראה מהן הבעיות שפותר הפקד UpdatePanel ואיך הוא מתנהג בדף לקוח. תרגיל נסה זאת הבא מציג דוגמה פשוטה של שימוש בפקד כדי למנוע הבהוב של דף במהלך postbacks.

בתרגיל זה תוסיף את הפקדים Label ו-Button לדרך. בכל פעם שתלחץ על הלחצן בדפדפן, יעודכן המאפיין Text של Label עם התאריך והשעה הנוכחיים של השרת. בדרך כלל מלווים postbacks בהבהוב של הדרך. כדי למנוע זאת, עטוף את הפקדים שבדרך ב-UpdatePanel, ותראה כיצד פקד זה משפיע על ההתנהגות.

1. ב-VWD, פתח את הפרויקט Planet Wrox בכתובת C:\BegASPNET\Site.
2. צור קובץ חדש בשם UpdatePanel.aspx בתיקייה Demos. ודא שהקובץ מבוסס על דף הבסיס המרכזי שיצרת קודם, ושהוא משתמש בשפת התכנות שלך. קבע את Title של הדרך ל- UpdatePanel Demo.
3. העבר את הדרך החדש למצב Design View וגרור פקד Label ופקד Button מארגז הכלים אל שומר המקום cpMainContent. אם ContentPlaceHolder נהיה קטן פתאום לגודל של Label, שחרר את Button מעל הפקד Label. הפקד Button יוצב במקרה זה לפני Label, אבל אם תגרור כעת שוב את Label על הפקד Button, השניים יחליפו מקומות. לחילופין, בחר את Label על ידי לחיצה אחת עליו, ואז לחץ על מקש החץ הימני פעם אחת כדי להציב את הסמן ליד Label. אם תקיש כעת Enter כמה פעמים, תיצור יותר מקום ב-ContentPlaceHolder.
4. השתמש בטבלת המאפיינים כדי לנקות את המאפיין Text של הפקד Label. לביצוע, לחץ לחיצה ימנית על המאפיין Text בטבלת המאפיינים, ובחר Reset. כעת תקבל את סימני העריכה הבאים:

```
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" runat="Server">
  <asp:Label ID="Label1" runat="server"></asp:Label>
  <asp:Button ID="Button1" runat="server" Text="Button" />
</asp:Content>
```

5. ב-Design View, לחץ לחיצה כפולה על שטח הדרך שצבעו אפור ומצביע על קריאה בלבד. הפעולה תגרום להגדרת מטפל לאירוע Load של הדרך. הוסף את הקוד הבא למטפל שיצרה עבורך VWD:

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Me.Load
    Label1.Text = System.DateTime.Now.ToString()
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = System.DateTime.Now.ToString();
}
```

6. שמור את כל השינויים והקש Ctrl+F5 כדי לפתוח את הדף בדפדפן. Label מציג כעת את התאריך והשעה הנוכחיים. לחץ כמה פעמים על הפקד Button. שים לב שבכל פעם שאתה לוחץ על הלחצן, הדף **מהבהב** ואז הוא מוצג מחדש ומראה את התאריך והשעה הנוכחיים (המעודכנים על ידי השרת).

7. סגור את הדפדפן, חזור ל-VWD והעבר את הדף UpdatePanel.aspx למצב Markup View. גרור UpdatePanel מהקטגוריה AJAX Extensions של ארגז הכלים אל הקוד, והצב אותו מייד לפני הפקד Label שהוספת בצעד 3.

8. הוסף אלמנט <ContentTemplate> בין תגיות הפתיחה והסגירה של UpdatePanel. שים לב ש-IntelliSense מופיע ברגע שאתה מקליד את סוגר הפתיחה, והדבר מקל על השלמת האלמנט. עכשיו גזור את תגיות הסגירה </ContentTemplate> ו- </UpdatePanel> והדבק אותם מתחת ללחצן שיצרת בצעד 3. אתה אמור לסיים עם סימני העריכה הבאים:

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
  <ContentTemplate>
    <asp:Label ID="Label1" runat="server"></asp:Label>
    <asp:Button ID="Button1" runat="server" Text="Button" />
  </ContentTemplate>
</asp:UpdatePanel>
```

9. גרור ScriptManager מהקטגוריה AJAX Extensions של ארגז הכלים, והצב אותו מייד לפני תגית הפתיחה של UpdatePanel:

```
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" runat="Server">
  <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
  <asp:UpdatePanel ID="UpdatePanel1" runat="server">
```

10. שמור את השינויים ובקש שוב את הדף בדפדפן. לחץ על הלחצן כמה פעמים כדי לעדכן את התווית עם התאריך והשעה הנוכחיים. שים לב שכעת הדף **אינו מהבהב** ורק התווית מעודכנת בדף.

כיצד זה פועל

עטיפת התוכן ב-UpdatePanel מגדירה בדף אזור שברצונך לרענן מבלי להשפיע על הדף כולו. בדוגמה הקודמת גרם הלחצן Button שבתוך UpdatePanel ל-postback, ועל ידי כך לרענון האזור. רק אותו חלק של הדף שעטוף ב-UpdatePanel מרוענן. פעולה זו גורמת לטעינה מחדשת של הדף ללא הבהוב. כפי שתראה מאוחר יותר, גם פקדים שמוצבים מחוץ ל-UpdatePanel יכולים לעדכן אזור.

כדי שרוב הפונקציונליות של Ajax תפעל כהלכה, חובה להציב את ScriptManager בדף הראשי. הוא משמש כגשר בין דף הלקוח וסביבת העבודה של Ajax, ורואג לדברים כמו רישום הקבצים הנכונים של JavaScript שמשתמשים בהם בדפדפן. בסעיפים הבאים תכיר את שני הפקדים ביתר פירוט.

מבט מקרוב על הפקד UpdatePanel

החלק היחיד בדף שמתעדכן כשלווחצים על הלחצן, הוא הפקד UpdatePanel והתוכן שלו (כפי שדנו בתרגיל הקודם). זוהי התנהגות ברירת המחדל של UpdatePanel, כשרק התוכן הפנימי שלו מרוענן על ידי פקדי שרת אחרים שמוגדרים בתוך האלמנט <ContentTemplate>. עם זאת, UpdatePanel יכול לעשות יותר מזה. הטבלה הבאה מפרטת כמה מהמאפיינים החשובים של UpdatePanel שמאפשרים להשפיע על התנהגותו.

מאפיין	ערך
ChildrenAsTriggers	מאפיין זה קובע אם פקדים שנמצאים בתוך UpdatePanel יכולים גם לגרום לרענון שלו. ברירת המחדל של הערך היא True, כפי שראית בתרגיל הקודם. כשאתה קובע ערך זה כ-False, עליך להגדיר את UpdateMode כ-Conditional. שים לב שכאשר הערך הוא False, פקדים שמוגדרים בתוך UpdatePanel עדיין גורמים ל-PostBack לשרת, הם רק מפסיקים לעדכן את הלוח באופן אוטומטי.
Triggers	האוסף Triggers מכיל את האלמנטים PostBackTrigger ו-AsyncPostBackTrigger. הראשון שימושי כשברצונך לכפות רענון מלא של הדף, והאחרון שימושי כשברצונך לעדכן את UpdatePanel על ידי פקד שמוגדר מחוץ ללוח.
RenderMode	אפשר להגדיר מאפיין זה כ-Block או כ-Inline כדי לציין אם UpdatePanel מרנדר (render) את עצמו כאלמנט <div> או כאלמנט .
UpdateMode	מאפיין זה יכול לקבל שני ערכים: Always קובע שהפקד ירוענן תמיד, ואילו Conditional קובע שהפקד ירוענן רק בהתקיים תנאי מסוים. לדוגמה, תנאי: כשאחד הפקדים שמוגדרים באלמנט <Triggers> גורם ל-PostBack.
ContentTemplate	מאפיין זה הוא חובה, למרות שאינו מופיע בטבלת המאפיינים של הפקד. זהו המיכל שכתוכו אתה מציב את הפקדים ככנים של UpdatePanel. אם לא תגדיר אותו, VWD תיתן אזהרה.

UpdatePanel מסוגל לרענן חלקים של דף, כפי שהודגם בתרגיל הקודם. פקדים שמוגדרים בתוך UpdatePanel, או מחוצה לו, יכולים לגרום לרענון של UpdatePanel. עם זאת, כדי לתפקד, UpdatePanel זקוק לפקד ScriptManager שמנהל, בין השאר, את קוד ה-JavaScript בצד הלקוח.

הפקד ScriptManager

הפקד ScriptManager משמש כגשר בין דף הלקוח לבין השרת. הוא מנהל משאבי תסריט (קבצי JavaScript בהם משתמשים אצל הלקוח), דואג לעדכונים חלקיים של דף, כפי שהראינו קודם, ומנהל אינטראקציה עם האתר שלך עבור נושאים כמו שירותי Web ושירותי היישום של ASP.NET כמו הגדרות גישה והזדהות המשתמש. בפרקים 15 ו-16 נעסוק בהרחבה בשירותים אלה מנקודת מבט השרת.

כשהיכולות של Ajax נדרשות בדפים בודדים בלבד, מציבים בדרך כלל את הפקד ScriptManager ישירות בדף התוכן. בתרגיל נסה זאת האחרון ראית כיצד זה פועל. עם זאת, ניתן להציב את ScriptManager בדף ראשי כדי שיהיה זמין ברחבי האתר. תבצע זאת בתרגיל מאוחר יותר בפרק.

למחלקה ScriptManager יש כמה מאפיינים שרובם משמשים בתרחישים מתקדמים. במצבים רבים, כמו בעדכון קטעים של דף תוך שימוש ב-UpdatePanel שראית זה עתה, אין צורך לשנות מאפיין כלשהו של המחלקה ScriptManager. בתרחישים אחרים ייתכן שתצטרך לשנות או להגדיר כמה ממאפייני המחלקה. הטבלה הבאה מפרטת כמה מהמאפיינים היותר נפוצים של הפקד ScriptManager.

מאפיין	ערך
AllowCustomErrorsRedirect	מאפיין זה קובע אם שגיאות שהיו במהלך פעולת Ajax יגרמו לטעינת דף השגיאות המותאם אישית. ברירת המחדל של הערך היא True. אם הערך מוגדר כ-False, השגיאה מוצגת בדפדפן כחלון התרעה של JavaScript, או שהיא מוסתרת מהלקוח כשאפשרות הניפוי מבוטלת בדפדפן. שים לב שאם לא קבעת תצורה עבור דף שגיאות מותאם אישית, השגיאה תוצג תמיד כהתרעת JavaScript, מבלי להתחשב בערך של מאפיין זה. בפרק 17 נדון בהגדרת דף שגיאות מותאם אישית ובניפוי היישום.
AsyncPostBackErrorMessage	כשאינך משתמש בדפי שגיאות מותאמים אישית, יאפשר לך מאפיין זה להתאים אישית את הודעת השגיאה שהמשתמשים יראו כשתרחש שגיאת Ajax. הוא מאפשר להסתיר מהמשתמש פרטים מיותרים, ולהציג לו הודעת שגיאה יותר ידידותית.
EnablePageMethods	מאפיין זה קובע אם קוד לקוח מורשה לקרוא לשיטות שמוגדרות בדף. יותר מאוחר תראה כיצד זה פועל.
EnablePartialRendering	מאפיין זה קובע אם ScriptManager תומך ברינדור של חלקי של הדף תוך שימוש בפקדי UpdatePanel. השאר הגדרה זו כ-True, אלא אם כן ברצונך לחסום את העדכונים החלקיים עבור הדף כולו.

מאפיין	ערך
Scripts	אלמנט הבן <Scripts> של הפקד ScriptManager מאפשר להוסיף קבצי JavaScript שחייבים להיטען על ידי הלקוח בזמן ביצוע.
Services	האלמנט <Services> מאפשר להגדיר שירותי Web שיהיו נגישים מדפי צד-לקוח. בהמשך הפרק תראה כיצד להשתמש בשירותי Web.

למרות שגם UpdatePanel וגם ScriptManager הם כל מה שצריך כדי ליצור דפים חופשיים מהבהוב, ASP.NET AJAX מציעה יותר מכך כדי לשפר את חווית המשתמש באתר מאופשר-Ajax. דרך אחת לשיפור חווית המשתמש היא שימוש בפקד UpdateProgress שנדון בו מיד. אפשרות אחרת היא להשתמש בפקד Timer, שבו נדון מאוחר יותר בפרק.

מתן חשוב למשתמשים

חרף בעיות התצוגה שגורמים postbacks בדרך כלל, יש להם יתרון גדול אחד: המשתמש יכול לראות שמהו קורה. UpdatePanel מקשה על כך מעט. למשתמשים אין רמז חזותי שמהו קורה עד לאחר שקרה. כדי לומר למשתמשים להמתין כמה שניות בעוד בקשתם מעובדת, השתמש בפקד UpdateProgress.

הפקד UpdateProgress

מחברים את פקד UpdateProgress ל-UpdatePanel על ידי המאפיין AssociatedUpdatePanelID. התוכן של הפקד שמוגדר באלמנט <ProgressTemplate>, מוצג בכל פעם שהפקד UpdatePanel הקשור אליו עסוק ברענון. כדי להודיע למשתמש שמהו קורה, התבנית מכילה בדרך כלל טקסט כמו "Please wait" או תמונת הנפשה, אם כי כל סימן עריכה אחר קביל גם הוא.

בנוסף למאפיינים AssociatedUpdatePanelID ו- <ProgressTemplate>, הפקד UpdateProgress כולל את המאפיינים הבאים שבהם תשתמש בדרך כלל.

מאפיין	ערך
DisplayAfter	מאפיין זה קובע את הזמן באלפית השנייה שהפקד ממתין לפני שהוא מציג את תוכנו. הדבר שימושי כאשר משך הרענון כה קצר עד שהצגת הודעה היא בבחינת מאמץ-יתר. ברירת המחדל היא 500 (חצי שנייה).
DynamicLayout	מאפיין זה קובע אם הפקד תופס שטח מסך כשהוא מוסתר. הדבר ממופה ישירות למאפייני CSS אלה: display: none; או visibility: hidden; שכבר ראינו.

בתרגיל הבא תלמד לשלב את UpdatePanel, ScriptManager ואת UpdateProgress כדי להפוך את פקד המשתמש ContactForm לחופשי מהבהובים.

נסה זאת דפים חופשיים מהבהוב עבור כל הרכיבים יחד

בתרגיל זה תשנה את פקד המשתמש ContactForm.aspx שיצרת קודם, תעטוף את כל הפקד ב-UpdatePanel כדי שהדף לא יבצע postback מלא כאשר תזין הודעה, ותלחץ על הלחצן Send. כדי שהמשתמשים יבינו שהדף עסוק כשההודעה נשלחת, תוסיף פקד UpdateProgress. בתוך הפקד תציב תמונת הנפשה (GIF) (המצורפת לקוד המקור של הספר). לחילופין, תוכל לפנות לאתר www.ajaxload.info וליצור תמונת הנפשה משלך.

1. פתח את פקד המשתמש ContactForm.aspx מהתיקייה Controls. ב- Markup View, עטוף עם UpdatePanel ו- <ContentTemplate> את האלמנט <table> כולו ואת Label שבתחתית. תוכל לבצע זאת על ידי הקלדה ישירה של הקוד ב- Markup View, או על ידי גרירת הפקד מהקטגוריה AJAX Extensions של ארגז הכלים. אתה אמור לסיים עם הקוד הבא:

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
  <ContentTemplate>
    <table class="style1" runat="server" id="FormTable">
      ....
    </table>
    <asp:Label ID="lblMessage" runat="server" Text="Message Sent"
      Visible="false"></asp:Label>
  </ContentTemplate>
</asp:UpdatePanel>
```

2. שמור את השינויים שערכת בפקד. פתח את הקובץ MasterPage.master מהתיקייה MasterPages. בין תגית הפתיחה <form> לבין <div> של PageWrapper הוסף פקד ScriptManager, על ידי גרירתו מארגז הכלים אל הדף. אתה אמור לסיים עם הקוד הבא:

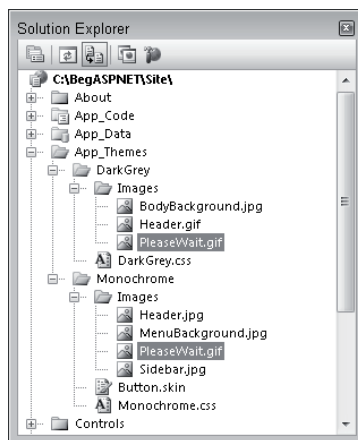
```
<body>
  <form id="form1" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
    <div id="PageWrapper">
```

3. שמור את השינויים שערכת בדף הראשי וסגור אותו.

4. פתח את הדף UpdatePanel.aspx שיצרת בתרגיל נסה זאת קודם, והסר את האלמנט ScriptManager. מכיוון שפקד זה מוכרז כעת בדף הראשי, אינך יכול לשוב ולהגדירו בדפים שמבוססים על אותו דף ראשי. שמור את השינויים וסגור את הדף.

5. פתח בדפדפן את הדף Contact.aspx מהתיקייה About, ומלא את הטופס יצירת קשר. שים לב שברגע שאתה לוחץ על הלחצן Send, הטופס נעלם והוא מוחלף בתווית שאומרת שההודעה נשלחה. כמו בדוגמה הקודמת, לא תבחין בהבהוב של דף כשהדף נטען מחדש ומציג את הטקסט Message Sent.

6. כדי לשמור את המשתמש מעודכן על ההתקדמות בעוד ההודעה נשלחת לשרת הדואר, צריך להוסיף לך את הפקד `UpdateProgress`. בתוך פקד זה תוסיף תמונת הנפשה ומעט טקסט שמודיע למשתמש שההודעה נשלחת. כדי להוסיף את התמונה, אתר בעזרת סייר החלונות את התיקייה שבה ממוקמים הקבצים שמגיעים עם ספר זה (`C:\BegASPNET\Resources`). פתח את התיקייה Chapter 10 ואחר כך את התיקייה Images. `Monochrome`. גרור את הקובץ `PleaseWait.gif` מסייר החלונות אל התיקייה Images של ערכת הנושא `Monochrome` שתחת `App_Themes`. חזור על תהליך זה, אבל עכשיו גרור `PleaseWait.gif` מהתיקייה `DarkGrey` אל התיקייה Images של ערכת נושא זו. תרשים 10-2 מראה כיצד תופענה שתי התמונות.



תרשים 10-2

7. פתח את הקובץ `Monochrome.css`, גלול כל הדרך מטה עד לקצה והוסף את הכלל הבא:

```
.PleaseWait
{
    height: 32px;
    width: 500px;
    background-image: url(Images/PleaseWait.gif);
    background-repeat: no-repeat;
    padding-left: 40px;
    line-height: 32px;
}
```

8. העתק כלל זה אל קובץ `DarkGrey.css` של ערכת הנושא `DarkGrey`.

9. עבור חזרה לפקד המשתמש `ContactForm.ascx`, וגרור פקד `UpdateProgress` אל מתחת לתגית הסגירה של `UpdatePanel` שבקצה הקובץ. הגדר את `AssociatedUpdatePanelID` שלו ל- `UpdatePanel1`.

10. בין התגיות `<UpdateProgress>` צור `<ProgressTemplate>`, ובתוך תבנית זו צור אלמנט `<div>` כשהמאפיין class שלו מוגדר ל- `PleaseWait`, שהיא מחלקת CSS שיצרת

בצעד 7. בתוך האלמנט <div> הקלד מעט טקסט כדי להודיע למשתמשים שעליהם להמתין מעט. אתה אמור לסיים עם הקוד הבא:

```
</asp:UpdatePanel>
<asp:UpdateProgress ID="UpdateProgress1" runat="server"
    AssociatedUpdatePanelID="UpdatePanel1">
    <ProgressTemplate>
        <div class="PleaseWait">
            Please Wait...
        </div>
    </ProgressTemplate>
</asp:UpdateProgress>
```

11. כדי לראות את הפקד UpdateProgress, עליך לחקות מצב של עיכוב ממושך בעת שליחת ההודעה. לשם כך הוסף את שורת הקוד הבאה לקוד ברקע של הפקד, מייד לאחר השורות שמשנות את המראה של הפקדים בשיטה ששולחת את הודעת הדואר:

VB.NET

```
lblMessage.Visible = True
FormTable.Visible = False
System.Threading.Thread.Sleep(5000)
```

C#

```
lblMessage.Visible = true;
FormTable.Visible = false;
System.Threading.Thread.Sleep(5000);
```

12. שמור את כל השינויים ופתח שוב את הדף Contact.aspx מהתיקייה About. מלא את הפרטים הנדרשים ולחץ על הלחצן Send. זמן קצר לאחר לחיצת הלחצן אמור להופיע הפקד UpdateProgress שמציג טקסט ותמונת הנפשה מתחת לטופס, כפי שמוצג בתרשים 10-3.

Get in touch with us

Use the form below to get in touch with us. Enter your name, e-mail address and your home or phone number to get in touch with us.

Your name:

Your e-mail address:

Your e-mail address again:

Home phone number:

Business phone number:

Comments:

 Please Wait...

תרשים 10-3

כיצד זה פועל

כל מה שנמצא בין גבולות ContentTemplate של UpdatePanel שבתוך פקד המשתמש, יעודכן בזמן postback מבלי להשפיע על חלקים אחרים בדף. בדרך זו תוכל להסתיר את הטופס עם פקדי השרת ולהחליפו בתווית Message sent מבלי לגרום להבהוב כלשהו בדף.

כדי להודיע למשתמש שההודעה נשלחה, הוספת לאתר את הפקד UpdateProgress. כברירת מחדל, פקד זה מוצג כשהעדכון לצורך רענון של UpdatePanel של Ajax שהוא קשור אליו נמשך יותר מחצי שנייה. האלמנט <ProgressTemplate> של הפקד הכיל אלמנט <div> פשוט שה-class שלו הוגדר כ-PleaseWait. הוספת את כלל CSS הבא לשני קבצי CSS של ערכות הנושא:

```
.PleaseWait
{
    height: 32px;
    width: 500px;
    background-image: url(Images/PleaseWait.gif);
    background-repeat: no-repeat;
    padding-left: 40px;
    line-height: 32px;
}
```

קוד זה קובע תחילה את ממדי ההודעה Update לרוחב של 500 פיקסל ולגובה של 32 פיקסל. זה מספיק עבור הרוחב של בלוק התוכן, ויש בו די מקום עבור הודעה ארוכה יותר.

אחר כך הקוד מוסיף את תמונת ההנפשה כתמונת רקע. כדי למנוע מהתמונה לחזור על עצמה ברקע, המאפיין repeat מוגדר ל-no-repeat. המילוי משמאל הוגדר ל-40 פיקסל, כדי להזיז את הטקסט ב-<div> ימינה, באופן שיופיע סמוך לתמונת ההנפשה. לבסוף, line-height של הטקסט נקבע ל-32 פיקסל, אשר זהה לזה של כל ה-<div>. הדבר ממרכז את בלוק הטקסט כולו באופן אנכי בתוך האלמנט <div>.

לבסוף, הוספת את שורת הקוד הבאה למטפל ששולח את ההודעה:

```
System.Threading.Thread.Sleep(5000);
```

קוד זה עוצר את ביצוע הדף למשך 5 שניות (המספר שמועבר לשיטה Sleep מבטא באלפית השנייה) כדי שתספיק להתבונן היטב בהודעה שבפקד UpdateProgress. כאשר תעביר את המערכת לסביבת הייצור, יהיה עליך להסיר שורה זו, מכיוון שהיא תאט את הדף בצורה משמעותית מבלי לתרום לו כל ערך.

כאמור, המשתמש יכול ליזום עדכוני דף על ידי שימוש בלחצן Send. בנוסף, אפשר ליזום רענון דפים בתכנות במרווחי זמן מסוימים, נושא שנדון בו בסעיף הבא.

שימוש בפקד Timer

את הפקד Timer תמצא בקטגוריה AJAX Extensions של ארגז הכלים. הוא מעולה לביצוע קוד בצד-שרת שיש לחזור עליו שוב ושוב. לדוגמה, תוכל להשתמש בו לעדכון התוכן של UpdatePanel בכל 5 שניות. התוכן של UpdatePanel יכול לבוא ממקורות שונים, כמו בסיס נתונים להצגת תוכן חדש שהתווסף לאתר, שירותי חברות להצגת מספר המשתמשים המקוונים כעת, וגם משירותי Web שמציגים מידע על מחירי מניות או את המחיר הנמוך ביותר עבור מוצרים מסוימים שהמשתמשים עשויים לגלות בהם עניין.

הפקד Timer פשוט למדי לשימוש. הפקד מעורר את האירוע Tick במרווחי זמן קבועים. בתוך מטפל האירוע עבור אירוע זה תוכל לבצע כל קוד שתמצא לנכון. אם תחבר את Timer לפקד UpdatePanel ותשתמש באוסף Triggers של הפקד, תוכל לעדכן אזור יחיד בדף בכל פעם שהפקד Timer יעורר את האירוע Tick שלו.

בנוסף למאפיינים הרגילים שיש לרוב הפקדים כמו ID ו-EnableViewState, יש לפקד שני מאפיינים שקרוב לוודאי שתשתמש בהם. הטבלה הבאה מציגה אותם.

מאפיין	ערך
Enabled	מאפיין זה קובע אם הפקד Timer יעורר את האירוע Tick. כאשר הערך הוא True, הפקד מעורר את האירוע Tick במרווח הזמן שמצוין במאפיין Interval. כשהערך הוא False, הפקד אינו עושה כלום ואינו מעורר אירועים.
Interval	מאפיין זה קובע את המרווח (באלפית השנייה) בין אירועי Tick שהפקד מעורר. לדוגמה, אם ברצונך שהפקד יעורר אירוע כל דקה, קבע ערך זה ל-60,000.

מטרת התרגיל הבא היא להראות את הפקד Timer בפעולה. התרגיל יראה כיצד ליצור דף שמתעדכן ברציפות מבלי לגרום ל-postback מלא. תוכל להשתמש באותם עקרונות להצגת כל סוג של נתונים דינמיים שיעלה בדעתך.

נסה זאת שימוש בפקד Timer בדף ASPX

בתרגיל זה תראה כיצד להשתמש בפקד Timer לעדכון אזור בדף. Timer יכוון לעורר אירוע בכל 5 שניות ואז יעדכן תווית מסוימת. בנוסף ל-Timer הדף כולל גם לחצן רגיל שלחיצה עליו מציינת בקשה לרענון המסך. הלחצן מאפשר למשתמשים לכפות רענון כשאין ברצונם להמתין לעדכון האוטומטי הבא.

- צור דף חדש בתיקייה Demos וקרא לו Timer.aspx. הדף צריך להיות מבוסס על התבנית המותאמת אישית שלך. קבע את Title של הדף ל-Timer Demo.
- מהדף UpdatePanel.aspx שיצרת קודם, העתק את סימני העריכה של הפקד UpdatePanel כולו. כעת הדבק אותם בתוך הפקד Content עבור הבלוק cpMainContent של הדף Timer.aspx שיצרת זה עתה. הסר את הלחצן מ-ContentTemplate. לבסוף, כדי

ליצור מקום, הוסף שלושה שוברי שורה (האלמנטים `<br />`) אחרי `UpdatePanel`. אתה אמור לסיים עם הקוד הבא:

```
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" Runat="Server">
  <asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
      <asp:Label ID="Label1" runat="server"></asp:Label>
    </ContentTemplate>
  </asp:UpdatePanel>
  <br /><br /><br />
</asp:Content>
```

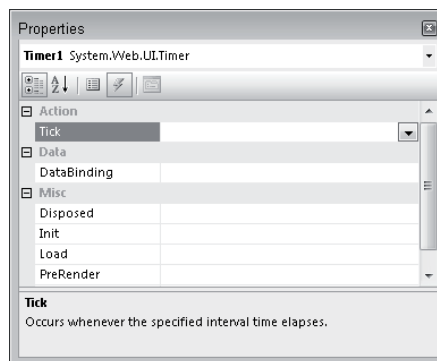
3. כשאתה עדיין ב- Markup View, גרור פקד `Timer` מהקטגוריה `AJAX Extensions` של ארגז הכלים. שחרר אותו מייד תחת `UpdatePanel` ושוברי השורות.

4. גרור פקד `Button` אל מתחת ל-`Timer`. ודא שהן `Timer` והן `Button` מופיעים מחוץ לפקד `UpdatePanel`, לשם כך בדוק אם הקוד נראה כך:

```
</asp:UpdatePanel>
<br /><br /><br />
<asp:Timer ID="Timer1" runat="server"></asp:Timer>
<asp:Button ID="Button1" runat="server" Text="Button" />
</asp:Content>
```

5. עבור ל- `Design View`, העלה את טבלת המאפיינים עבור הפקד `Timer` והגדר את המאפיין `Interval` שלו ל-5000, כדי שהפקד יתקתק בכל 5 שניות.

6. עבור מטבלת המאפיינים לקטגוריה `Events`. לחץ לחיצה כפולה על האירוע `Tick`, שמוצג בתרשים 10-4.



תרשים 10-4

7. צור שיטה בקוד ברקע (ב-`Sub` ;VB.NET וב-`void` ;C#) וקרא לה `UpdateLabel`. השיטה תגדיר את `Text` של `Label1` כשווה לתאריך והשעה הנוכחיים. קרא לשיטה החדשה ממטפל האירוע שהוסיפה `VWD` בצעד הקודם. אתה אמור לסיים עם הקוד הבא:

VB.NET

```
Protected Sub Timer1_Tick(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Timer1.Tick
```

```
    UpdateLabel()
```

```
End Sub
```

```
Private Sub UpdateLabel()
```

```
    Label1.Text = DateTime.Now.ToString()
```

```
End Sub
```

C#

```
protected void Timer1_Tick(object sender, EventArgs e)
```

```
{
```

```
    UpdateLabel();
```

```
}
```

```
private void UpdateLabel()
```

```
{
```

```
    Label1.Text = System.DateTime.Now.ToString();
```

```
}
```

8. חזור ל- Design View. לחץ לחיצה כפולה על Button כדי להגדיר מטפל לאירוע Click. קרא לשיטה UpdateLabel מתוך מטפל זה:

VB.NET

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Button1.Click
```

```
    UpdateLabel()
```

```
End Sub
```

C#

```
protected void Button1_Click(object sender, EventArgs e)
```

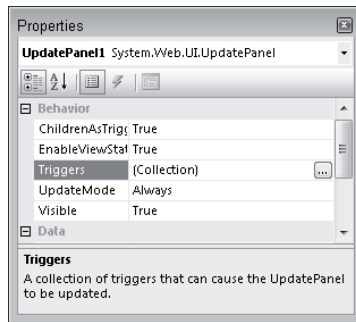
```
{
```

```
    UpdateLabel();
```

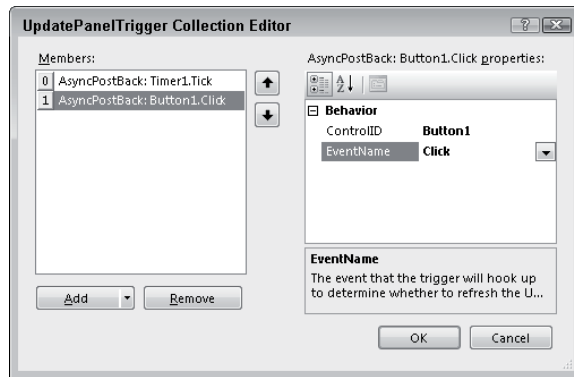
```
}
```

9. חזור ל- Design View פעם נוספת ובחר UpdatePanel. פתח את טבלת המאפיינים שלו (על ידי F4) ולחץ על הלחצן עם האליפסות עבור המאפיין Triggers, שמוצג בתרשים 10-5.

10. בתיבת הדו-שיח שמופיעה לחץ על הלחצן Add כדי להכניס מזניק (trigger) מסוג AsyncPostBack חדש. מצד ימין של המסך קבע את ControlID שלו ל-Timer1 ואת EventName שלו קבע ל-Tick. חזור על צעד זה, אבל הפעם הוסף ל-Button1 מזניק (trigger) עבור האירוע Click שלו. כשתסיים, תיבת הדו-שיח אמורה להיראות דומה לתרשים 10-6.



תרשים 10-5



תרשים 10-6

11. לחץ OK לשחרור תיבת הדו-שיח, עבור חזרה אל תצוגת Markup View ובדוק אם הקוד שלך נראה כמו זה:

```
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" runat="Server">
  <asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
      <asp:Label ID="Label1" runat="server"></asp:Label>
    </ContentTemplate>
    <Triggers>
      <asp:AsyncPostBackTrigger ControlID="Timer1" EventName="Tick" />
      <asp:AsyncPostBackTrigger ControlID="Button1" EventName="Click" />
    </Triggers>
  </asp:UpdatePanel>
<br />
<br />
<br />
  <asp:Timer ID="Timer1" runat="server" OnTick="Timer1_Tick" Interval="5000"></asp:Timer>
  <asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click" />
</asp:Content>
```

אם אתה משתמש ב-VB.NET לא תראה את קוד המטפל עבור OnTick ועבור OnClick. הסיבה - כדי לחבר את השיטות לאירועים בקוד ברקע, Visual Basic משתמשת במילת המפתח Handles.

12. שמור את כל השינויים ואז קרא ל-Timer.aspx מתוך הדפדפן. שים לב שהתווית מתעדכנת מדי 5 שניות בתאריך ובשעה הנוכחיים. כשתלחץ על הלחצן תראה שהפקד Label מעודכן מייד, במקום להמתין שיקרה האירוע Tick הבא בפקד Timer.

כיצד זה פועל

בתרגיל זה העתקת את הפקד UpdatePanel מדף קודם, והוספת מחוץ לו את הפקדים Timer ו-Button. המאפיין Interval של הפקד Timer נקבע ל-5000, ופירושו שהפקד Timer מעורר את האירוע Tick כל 5 שניות, ואז הקוד בקוד ברקע של הדף Timer.aspx מבצע את השיטה UpdateLabel.

VB.NET

```
Protected Sub Timer1_Tick(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Timer1.Tick  
    UpdateLabel()  
End Sub
```

C#

```
protected void Timer1_Tick(object sender, EventArgs e)  
{  
    UpdateLabel();  
}
```

השיטה UpdateLabel בתורה מעדכנת את Label שבתוך UpdatePanel על ידי הגדרת המאפיין Text שלו לתאריך ולשעה הנוכחיים.

כדי שהפקד Timer יעדכן את התוכן של UpdatePanel בלבד, ולא יגרום רענון מלא של הדף, חשוב לרשום אותו באלמנט <Triggers> של UpdatePanel בדרך זו:

```
<Triggers>  
    <asp:AsyncPostBackTrigger ControlID="Timer1" EventName="Tick" />  
    <asp:AsyncPostBackTrigger ControlID="Button1" EventName="Click" />  
</Triggers>
```

קוד זה רושם את שני הפקדים Timer ו-Button, כפקדים שיכולים לכפות על UpdatePanel עדכון חלקי.

הגדרת המאפיין Interval של הפקד Timer לערך 5000 msec גורמת לפקד לעדכן את המסך שלך באופן אוטומטי בכל 5 שניות. אם אינך יכול להמתין זמן ממושך כל כך, באפשרותך ללחוץ על הלחצן Button שהוספת לדף. כשהלחצן נלחץ, הוא גורם ל-postback אל השרת, שבו מטפל האירוע Click של הפקד Button וקורא לשיטה UpdateLabel. מכיוון שהפקד Button רשום כמזניק עבור UpdatePanel, יהיה Label מעודכן מבלי לטעון מחדש את כל הדף. אם ברצונך להשתמש בפקד Button כדי לעדכן את הדף כולו, ולהתעלם מהעדכונים החלקיים של הדף שמציעה Ajax, הסר את AsyncPostBackTrigger מהאלמנט <Triggers>. במצב זה, כשתלחץ על הפקד Button, יקרה postback רגיל, והדף יעודכן כולו.

התרחיש של לוח שמתעדכן באופן אוטומטי יחד עם היכולת לרענן את התוכן בלחיצת לחצן, הינו נפוץ למדי. הלוח שמתרענן באופן אוטומטי הוא דרך לא פולשנית לספק למשתמש מידע עדכני ביותר מהשרת. בו-בזמן, אם משתמשים רוצים לרענן את הנתונים בכל רגע שיבחרו, כל שעליהם לעשות הוא ללחוץ על לחצן. מבחינת הקוד אין הבדל גדול בין שני המקרים, כי בשניהם הפעולה מסתכמת בקריאה לשיטה אחת - UpdateLabel.

הדיון בפקד Trigger מסיים את הדיון בפקדי צד-שרת שכלולים בסביבת העבודה ASP.NET AJAX. כעת נותר לנו להציג את השימוש בשירותי Web בדפים שמאפשרים עבור Ajax, ולהציג את המבוא לסביבת עבודה מבוססת JavaScript בצד הלקוח. במהלך הדיון בשירותי Web ובסביבת העבודה של הלקוח, תראה כיצד להשתמש בפקד ScriptManagerProxy, שיהיה האחרון בקטגוריה Ajax Extensions של ארגז הכלים שנדון בו.

שימוש בשירותי Web באתרי אינטרנט מבוססי Ajax

תוספת חשובה לערכת כלי הפיתוח באינטרנט היא היכולת שמקנה Ajax לקרוא לשירותי Web מאתר אינטרנט מבוסס ASP.NET. המשמעות של יכולת הקריאה לשירותי Web שרירותיים שנגישים באינטרנט, מצביעה על כך שעכשיו קל הרבה יותר לגשת לנתונים מצד הלקוח ממקורות אחרים, דוגמת האתר שלך, או מאתרי אינטרנט חיצוניים שמאפשרים גישה לנתונים שלהם על ידי שירותי Web ציבוריים כמו Google Search, Google Maps, Amazon ו-Virtual Earth של מיקרוסופט. לפני שתוכל ליצור ולצרוך שירותי Web ביישום שלך, חשוב להבין מהו שירות Web, וכיצד להגדיר שירות כזה בפרויקט ASP.NET.

מהם שירותי Web?

שירותי Web הם למעשה שיטות (methods) שאתה יכול לקרוא להן באינטרנט, ואשר יכולות, לפי בחירה, להחזיר נתונים אל הקוד הקורא. הדבר עושה אותן לכלי אידיאלי להחלפת נתונים בין מערכות שונות. מכיוון ששירותי Web מבוססים על תקנים מוצקים ומובנים היטב, הם מקלים על החלפת נתונים בין סוגים שונים של פלטפורמות. לדוגמה, עם שירות Web קל להחליף נתונים בין אתר אינטרנט מבוסס ASP.NET, שפועל תחת מערכת חלונות של מיקרוסופט, לבין אתר מבוסס PHP, שפועל תחת מערכת Linux. בו-בזמן אפשר גם להחליף נתונים בין אתר אינטרנט מבוסס ASP.NET לבין דפדפן לקוח שמשתמש ב-JavaScript.

שירותי Web בפרויקט Planet Wrox ישמשו רק כדי לגרום לדרך לקוח בדפדפן "לדבר" עם השרת ולהחליף נתונים. באתר זה נמצאים השרת והלקוח באותו פרויקט אינטרנט - האחד מתבצע אצל הלקוח (ה-JavaScript שקורא לשרת האינטרנט), והשני נמצא בשרת (שירות Web עצמו). זהו הפתרון הקל ביותר מבחינת אבטחה, מכיוון ששני החלקים נותנים אמון זה בזה.

אם ברצונך שדפי צד-לקוח ידברו עם שירות Web בתחום אחר, עליך להגדיר בדפדפן אבטחה שתאפשר זאת. בנוסף, ביכולתך להשתמש בשירותי Web כדי לגרום לשני שרתים, או יישומים אחרים (יישומי שולחן העבודה למשל), לתקשר זה עם זה. כך, יישום אחד (יישום שולחן עבודה של חלונות, PHP או יישום מבוסס ASP הקלאסית, או אפילו שירות Web אחר מבוסס ASP.NET) פועל ברשת עם שירות Web של ASP.NET כדי להחליף נתונים. שני תרחישים אלה הם מחוץ לתחום הדיון בפרק זה.

שירותי Web שתיצור בפרויקטי Planet Wrox ייראו דומים לשיטות רגילות שכבר יצרת. השוני הוא בצורך לכלול בכל שיטה את המאפיין `WebMethod`. הסימון של המאפיין דומה לתגית קטנה או תווית שאתה מדביק לקטע קוד כמו שיטות, מאפיינים וכו', כדי לסמן אותם כמשהו מיוחד. קוד אחר שפועל באינטראקציה עם קטע הקוד בעל המאפיין, יכול לראות איזו סוג מאפיין יש לקטע הקוד בעל המאפיין, ולקבל החלטות שמבוססות על מידע זה. אל תדאג יותר מדי בקשר לכך, מפני שאינך צריך לקרוא מאפיינים אלה בעצמך כשאתה משתמש בשירותי Web. כל מה שעליך לעשות הוא להדביק את המאפיין לשיטה, כדי להפוך אותה ל-`WebMethod` ואתה מוכן למשימה. לדוגמה, כדי להפוך שיטה רגילה שמחזירה מחרוזת ל-`WebMethod`, עליך להחיל עליה את המאפיין הבא:

VB.NET

```
<WebMethod()> _  
Public Function HelloWorld(ByVal yourName As String) As String  
    Return String.Format("Hello {0}", yourName)  
End Function
```

C#

```
[WebMethod]  
public string HelloWorld(string yourName)  
{  
    return string.Format("Hello {0}", yourName);  
}
```

שים לב שבדוגמת VB.NET משתמשים בסוגריים זוויתיים כדי לעטוף את המאפיין. בנוסף, הקוד משתמש בקו תחתון () כתו המשך לשורה. ב-C# משתמשים בסוגריים מרובעים, ואפשר להציב את המאפיין בשורה נפרדת.

הגדרת המאפיין מאותתת ל- ASP.NET runtime שאתה רוצה לחשוף את השיטה כ- `Web Method`. הדבר גם מאפשר ליצור שיטות אחרות באותה מחלקה שאינן נחשפות באופן אוטומטי כשירותי Web. הדבר מקנה גמישות לקבוע מה יפתח בפני העולם החיצון.

בנוסף למאפיין `WebMethod` לסימון השיטה כ- `Web Method`, נהוג גם להציב את השיטה בקובץ בעל סיומת `asmx` ובתוך מחלקה שיוורשת מ-`System.Web.Services.WebService`. בסעיף הבא תראה כיצד זה פועל.

יצירת שירותי Web

קל מאוד ליצור שירותי Web על ידי VWD. גם במקרה זה, בדיוק כמו בסוגי הקבצים האחרים, VWD כוללת תבנית עבור שירותי Web. מוסיפים לאתר שירותי Web על ידי שימוש בתבנית הדו-שיח Add New Item. אחר כך תוכל לשנות את השירות ולבדוק אותו בדפדפן אינטרנט תוך שימוש בדף המבחן הרגיל שיוצר ASP.NET runtime באופן אוטומטי. כששירותי Web מתפקד כהלכה, תוכל להשתמש בו מקוד JavaScript של צד-לקוח, כפי שתראה לאחר התרגיל הבא.

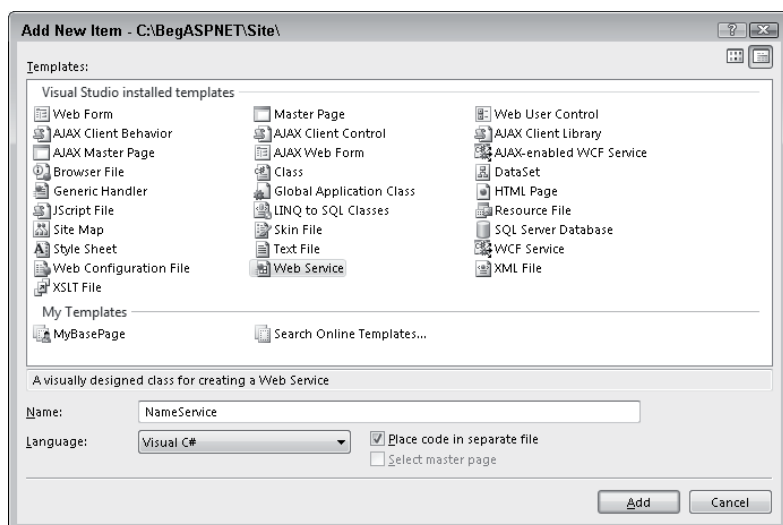
נסה זאת יצירת שירותי Web

בתרגיל זה תיצור שירותי Web בשם Hello World. שירות זה מקבל את שמך כפרמטר קלט ומחזיר ברכה אישית ידידותית. בעולם האמיתי קרוב לוודאי שאין שימוש רב לשירות זה. עם זאת, בגלל פשטות השירות יהיה קל יותר להתמקד במושגים הבסיסיים.

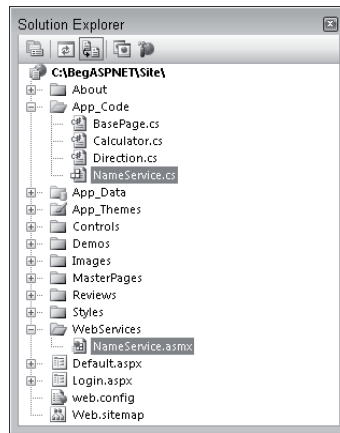
1. צור בשורש האתר תיקייה חדשה בשם WebServices, כדי לקבץ בה את כל שירותי ה-Web מהאתר.

2. לחץ לחיצה ימנית על התיקייה החדשה ובחר Add New Item. לחץ על הפריט Web Service, וקרא לשירות בשם NameService.aspx. ודא שבחרת בשפת התכנות המועדפת עליך וב- Place Code in Separate File, כפי שמוצג בתרשים 10-7.

3. לחץ Add להוספת השירות לאתר. שים לב שקובץ .asmx מתווסף לתיקייה WebServices, בעוד קובץ הקוד ברקע מוצב בתיקייה App_Code של האתר, כפי שמוצג בתרשים 10-8.



תרשים 10-7



תרשים 10-8

4. פתח את הקובץ NameService בתיקייה App_Code ושנה את הקוד של השיטה HelloWorld כדי שיקבל מחרוזת ויחזיר ברכה אישית. אתה אמור לסיים עם קוד כמו זה:

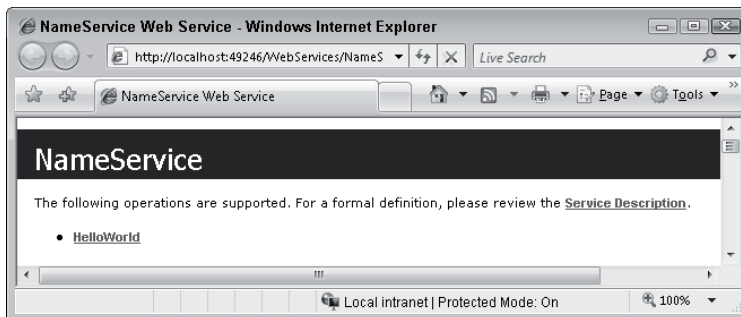
VB.NET

```
<WebMethod()> _
Public Function HelloWorld(ByVal yourName As String) As String
    Return String.Format("Hello {0}", yourName)
End Function
```

C#

```
[WebMethod]
public string HelloWorld(string yourName)
{
    return string.Format("Hello {0}", yourName);
}
```

5. שמור את כל השינויים. בסייר הפתרון, לחץ לחיצה ימנית על NameService.asmx, ובחר View in Browser. ברגע שהדפדפן יסיים את הטעינה, תקבל דף שמפרט את כל שירותי Web הציבוריים שמוגדרים בשירות NameService.asmx. בתרגיל זה תראה רק את HelloWorld, שמוצג בתרשים 10-9.



תרשים 10-9

6. לחץ על הקישור HelloWorld, ותועבר לדף שבו תוכל לבדוק את השירות. הקלד את שמך בשדה yourName ולחץ Invoke. כעת יפתח חלון חדש (ראה תרשים 10-10) שמראה את קוד XML שהוזר על ידי שירות Web.



תרשים 10-10

כיצד זה פועל

כפי שכבר נאמר, שירותי Web הם למעשה שיטות שניתן לקרוא להן על פני הרשת, כמו האינטרנט או הרשת המקומית שלך. הם מתוכננים לאפשר ליישומים להתקשר זה לזה ולהחליף ביניהם נתונים. הפורמט הבסיסי של ההודעה הוא בקוד XML, כפי שרואים בתרשים 10-10 שמציג את התוצאה של השיטה HelloWorld.

כשאתה מוסיף שירות Web לפרויקטים שלך, לא כל השיטות שבקובץ הופכות באופן אוטומטי לשיטות שניתן לקרוא להן ברשת. כדי לחשוף שיטה כשירות רשת, עליך להחיל עליה את המאפיין WebMethod.

VB.NET

```
<WebMethod()> _
```

```
Public Function HelloWorld(ByVal yourName As String) As String
```

C#

```
[WebMethod]
```

```
public string HelloWorld(string yourName)
```

לאחר שנקבע לשיטה מאפיין זה, היא גלויה לעולם החיצון ולפיכך נגישה למערכות חיצוניות. כשאתה פותח קובץ asmx בדפדפן, תקבל באופן אוטומטי דף שמאפשר להעמיד את השירות במבחן. במקרה של השירות HelloWorld, הקלדת את שמך ולחצת על הלחצן Invoke כדי לשלוח את השם כפרמטר לשירות. השירות הגיב על ידי הוספת שמך להודעת הפתיחה והחזיר את ההודעה כמחרוזת על ידי שימוש ב-String.Format:

VB.NET

```
Public Function HelloWorld(ByVal yourName As String) As String
```

```
Return String.Format("Hello {0}", yourName)
```

```
End Function
```

C#

```
public string HelloWorld(string yourName)
```

```
{
```

```
return string.Format("Hello {0}", yourName);
```

```
}
```

השיטה String.Format מקבלת כארגומנט ראשון מחרוזת שיכולה להכיל שומרי מקום מספריים, עטופים בזוג סוגריים מסולסלים ({}). בפרמטרים העוקבים תספק עבור כל ערך מספרי ערך מחרוזת. בדוגמה הקודמת יש שומר מקום אחד בלבד, אבל תוכל בקלות להרחיב את הקריאה לשיטה Format שיש לה יותר פרמטרים. לדוגמה, אם תרצה לעצב מחרוזת עם שם פרטי ושם משפחה, תשתמש בקוד הבא:

VB.NET

```
Return String.Format("Hello {0} {1}", firstName, lastName)
```

C#

```
return string.Format("Hello {0} {1}", firstName, lastName);
```

השימוש בשיטה String.Format מאפשר קוד קריא ומובן על פני שרשרת מחרוזות "מבולגן", תוך שימוש בתווים & או +, אתה מגדיר במחרוזת שומרי מקום, ומספק את הערכים בזמן ביצוע.

לבסוף, השיטה Web Service מחזירה את הודעת הפתיחה במחרוזת. הרכיב Web Service runtime אחראי לשליחת הערך המוחזר הזה אל הקוד הקורא. בדוגמה שלנו מציג דף המבחן את הערך המוחזר כמחרוזת XML גולמית.

למרות שהדוגמה היא פשוטה, העקרונות שראית כאן פועלים גם עבור שירותי Web מורכבים שמחליפים נתונים נרחבים שמעבר למחרוזת פשוטה. מובן מאליך שדף המבחן משמש רק כדי לבדוק אם השירות פועל בצורה נכונה. בשירותי Web ממשיים, הנתונים נצרכים בדרך כלל על ידי קוד אחר, כמו יישום אינטרנט או JavaScript של צד-לקוח. בסעיף הבא תראה כיצד זה פועל.

שימוש בשירותי Web באתר אינטרנט מבוסס Ajax

לפני ASP.NET AJAX, היו הקריאה לשירותי Web מדפדפן לקוח והפעולות על הנתונים המוחזרים על ידם כרוכים בכתיבה מרובה של קוד, בפרט אם רצית שהקריאה תפעל בדפדפנים העיקריים כמו Explorer, Firefox ואחרים. למרבה המזל, סביבת העבודה Ajax מגינה עליך מכל המורכבות והקידוד שנדרשים כדי לצרוך שירותי Web. כל שעליך לעשות הוא להוסיף מאפיין לשירותי Web כדי לסמן אותו כשירות שיכול להיקרא מתסריט מצד הלקוח. אחר כך תרשום את השירות ב-ScriptManager ותקודד כמה שורות קוד ב-JavaScript, כדי לעורר את השירות ולקבל את הערך שיוחזר על ידו. כל זה יפעל רק עבור שירותים שמוגדרים בתוך אתר האינטרנט שלך, כפי שתראה להלן. אם ברצונך לקרוא לשירותים שאינם נמצאים באותו תחום כמו הדף שקורא להם, עליך לכתוב קוד נוסף. דבר זה הוא מחוץ לתחום הדיון בספר זה. אם תרצה לדעת יותר על קריאה לשירותי Web חיצוניים, ישנם מספר מאמרים מעניינים בנושא ברשת.

בסעיף הבא תלמד כיצד להגדיר את תצורת שירותי Web שלך באופן שתוכל לקרוא להם מתסריט צד-לקוח. בתרגיל נסה זאת שיבוא אחר כך תלמד כיצד להשתמש בידע זה כדי לקרוא לשירותי Web מדף לקוח.

קביעת תצורת שירות Web

למדת קודם כיצד לסמן שיטה כ- Web Method על ידי הוספת מאפיין. הדבר חושף את השיטה לעולם החיצון. כדי לחשוף שירות Web לתסריט צד-לקוח, יש להוסיף מאפיין למחלקת השירות. כשתבונן במחלקה NameService ב-App_Code, תראה שהתבנית כבר הוסיפה עבורך את המאפיין, אבל הפכה אותו להערה.

VB.NET

```
' <System.Web.Script.Services.ScriptService()> _
```

C#

```
// [System.Web.Script.Services.ScriptService]
```

הסר את תגיות ההערה כדי להפוך את המאפיין לזמין, וכך לחשוף את השירות כולו כשירות צד-לקוח.

קביעת תצורת ScriptManager

בסעיף קודם בפרק זה כבר ראית שהפקד ScriptManager הוא רכיב הכרחי כמעט בכל הפעולות הקשורות ב-Ajax. הוא מנהל רישום של קבצי JavaScript של צד-לקוח (אלה שסביבת העבודה של Ajax משתמשת בהם, ולפי בחירה גם אלה שלך), הוא דואג, יחד עם הפקד UpdatePanel, לעדכונים של חלקי דף, והוא מטפל באינטראקציה עם שירותי Web שהוגדרו באתר. אפשר להוסיף ScriptManager לדף כלשהו, או לדף הראשי שממנו הוא יהיה זמין בכל האתר.

הצעד הבא לאחר הוספת ScriptManager הוא לומר לפקד שברצונך לחשוף את שירות Web שלך לתסריט לקוח. יש שתי דרכים לעשות זאת:

⊙ ב-ScriptManager בדף הראשי.

⊙ בדף תוכן שמשמש בשירות Web, ובמחלקה ScriptManagerProxy.

אם הצפי הוא שתשתמש בשירות Web בכל הדפים, או ברובם, ברור שמוטב להכריז על שירות Web ב-ScriptManager של הדף הראשי. תעשה זאת על ידי הגדרת אלמנט <Services> בפקד ScriptManager. האלמנט בתורו מכיל פקד ServiceReference שמצביע על השירות הציבורי שלך. לדוגמה, כדי להפוך את השירות NameService.aspx במחלקת הדפים באתר, הוסף לדף הראשי את שורות הקוד המוצגות להלן:

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
  <Services>
    <asp:ServiceReference Path="~/WebServices/NameService.aspx" />
  </Services>
</asp:ScriptManager>
```

על ידי הוספת ההפניה לשירות בדף הראשי, הוא הופך זמין לכל הדפים שמבוססים על אותו דף ראשי. משמעות הדבר שכל דף תוכן יטען את קבצי JavaScript שדרושים להפעלת השירות שהוגדרו בדף הראשי. אם דף התוכן כלל אינו משתמש בשירות, ברור שזה בזבוז משאבים

והגדלת התעבורה בין השרת לדפדפן. מכאן, שאם יש שירותים שאתה משתמש בהם בדפים בודדים בלבד, עדיף לפנות אליהם מהדף עצמו. בדף רגיל, שאינו משתמש בדף ראשי (אינו דף תוכן), אתה יכול להוסיף לדף ScriptManager משלו. עם זאת, כשתשתמש בדף תוכן שמוגדר לעבוד עם דף ראשי בעל ScriptManager משלו (זה המקרה עם הדפים באתר Planet Wrox) תצטרך להשתמש בפקד ScriptManagerProxy. בדף שמגיע לדפדפן יכול להיות ScriptManager אחד בלבד, לפיכך לא תוכל להוסיף עוד אחד לדף תוכן שמשתמש בדף ראשי שהוגדר בו ScriptManager. זו הסיבה שאתה זקוק ל-ScriptManagerProxy שישמש כגשר בין דף התוכן וה-ScriptManager שבדף הראשי. הדבר נותן גמישות רבה בקשר למקום שבו תרשום את השירותים שלך.

הגדר את ה-ScriptManagerProxy ולאחר מכן הוסף לו את האלמנט <Services>, בדיוק כמו שראית עם ScriptManager עצמו.

```
<asp:ScriptManagerProxy ID="ScriptManagerProxy1" runat="server">
  <Services>
    <asp:ServiceReference path("~/WebServices/NameService.asmx" />
  </Services>
</asp:ScriptManagerProxy>
```

הפקד ScriptManagerProxy מקל על רישום שירותי Web וקבצי JavaScript שמשתמשים בהם בדפי תוכן ספורים בלבד.

התרגיל הבא מדגים כיצד לרשום שירות Web וכיצד לגשת אליו מקוד צד-לקוח תוך שימוש ב-ScreenManagerProxy.

נסה זאת קריאת שירותי Web מקוד לקוח

בתרגיל זה תרשום את שירות Web שלך בפקד ScriptManagerProxy, כדי שהשירות יהיה זמין בדף אחד בלבד. בנוסף, תשנה את השירות כדי שהשירות שלו תהיינה נגישות מתסריט צד-לקוח. לבסוף תכתוב קוד JavaScript של צד-לקוח שמטרתו לגשת לשירות ולהציג את הערך המוחזר על ידי השירות.

1. הוסף את המאפיין ScriptService למחלקת השירות שלך כדי לציין שניתן לקרוא לה מתסריטי צד-לקוח. כדי לבצע זאת, פתח את הקובץ NameService.vb או את NameService.cs מהתיקייה App_Code. הסר את תגיות ההערה מהשורה שמגדירה את התכונה. אתה אמור לסיים עם הקוד הבא:

VB.NET

```
<System.Web.Script.Services.ScriptService> _
<WebService(Namespace="http://tempuri.org/")> _
<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Public Class NameService
  Inherits System.Web.Services.WebService
```


C#

```
[System.Web.Script.Services.ScriptService]
public class NameService : System.Web.Services.WebService
{
```

2. כבירת מחדל, מרחב השמות של השירות נראה כך:

VB.NET

```
<WebService(Namespace="http://tempuri.org/")> _
```

C#

```
[WebService(Namespace = "http://tempuri.org/")]
```

שם זה טוב דיו בזמן הפיתוח של שירותי Web, אולם ברגע שהשירות מועבר לסביבת ייצור, השם צריך לשקף את השם הייחודי של השירות. אם יש לך שם תחום משלך, תוכל לשנות את מרחב השמות למשהו כמו <http://www.yourdomain.com/>. אם אין לך תחום משלך, אל תדאג בקשר לכך. הדברים יפעלו היטב גם אם Namespace יוגדר לכבירת המחדל שלו <http://tempuri.org/>.

3. הצעד הבא הוא יצירת דף שמשמש בשירות ומתן אפשרות לקרוא לשירות על ידי הפקד ScriptManagerProxy. הוסף דף חדש בתיקייה Demos וקרא לו WebServices.aspx. ודא שהדף מבוסס על התבנית המרכזית BasePage. בדרך זו יהיה לדף שלך דף ראשי נכון, והוא יירש מהמחלקה BasePage שבתיקייה App_Code. קבע לדף Title כמו Web Services Demo. ברגע שהוספת את הדף, גרור פקד ScriptManagerProxy מארגז הכלים אל סימני העריכה של שומר המקום cpMainContent.

4. בתוך האלמנט ScriptManagerProxy, הוסף אלמנט <Services>, שבתורו מכיל ServiceReference שה-Path שלו מוגדר לקובץ NameService שיצרת קודם. שים לב שלאחר הקלדת "Path=" תקבל עזרה של IntelliSense כדי לבחור את הקובץ הנכון על ידי הצגת רשימת קבצים. לחץ על Pick URL שבתחתית הרשימה ודפדף לקובץ השירות בתיקייה WebServices. אתה אמור לסיים עם הקוד הבא בדף WebServices.aspx שלך:

```
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" runat="Server">
  <asp:ScriptManagerProxy ID="ScriptManagerProxy1" runat="server">
    <Services>
      <asp:ServiceReference Path="~/WebServices/NameService.asmx" />
    </Services>
  </asp:ScriptManagerProxy>
</asp:Content>
```

5. מהקטגוריה HTML של ארגז הכלים גרור Input (Text) ו- Input (Button) והצב אותם מייד תחת לתגית הסגירה של <ScriptManagerProxy>. על ידי שימוש באלמנטים של HTML פשוט, במקום שימוש בפקדי שרת של ASP.NET, תוכל לראות בוודאות כי הקוד שאתה עומד לקודד ממש יתבצע אצל הלקוח ולא בצד השרת. הגדר id של תיבת

הטקסט ל-txtYourName ואת id של הלחצן קבע ל-btnSayHello. הגדר את value של הלחצן ל- Say Hello. אתה אמור לסיים עם סימני העריכה הבאים:

```
</asp:ScriptManagerProxy>
<input id="txtYourName" type="text" />
<input id="btnSayHello" type="button" value="Say Hello" />
```

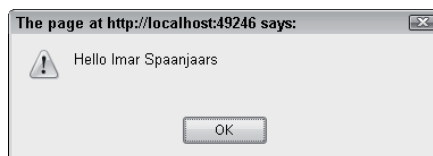
6. מתחת לשתי השורות המוצגות האלו הוסף בלוק JavaScript של צד-לקוח עם הקוד הבא:

```
<input id="btnSayHello" type="button" value="Say Hello" />
<script type="text/javascript">
    function HelloWorld()
    {
        var yourName = $get('txtYourName').value;
        NameService.HelloWorld(yourName, HelloWorldCallback);
    }

    function HelloWorldCallback(result)
    {
        alert(result);
    }

    $addHandler($get('btnSayHello'), 'click', HelloWorld);
</script>
```

7. שמור את כל השינויים על ידי הקשת Ctrl+Shift+S. בקש בדפדפן את הדף WebServices.aspx. הקלד את שמך ולחץ על הלחצן Say Hello. אם הכל יפעל כשורה, תקבל משירות Web את הודעת הברכה הכוללת את שמך. תרשים 10-11 מציג את חלון ההתרעה ב- Firefox של Mozilla.



תרשים 10-11

הערה: אם במקום תיבת הודעה זו תקבל שגיאה, או שתראה משולש צהוב קטן בפינה השמאלית התחתונה של המסך, ודא שהקלדת את קוד JavaScript בדיוק כמו בקטע הקוד לעיל. השפה JavaScript רגישה לרישיות, ולכן רצוי שתוודא שכל האותיות הרישיות מופיעות כהלכה. ודא גם שבלוק JavaScript שהוספת בצעד 6 מופיע לאחר תיבת הקלט והלחצן שהגדרת קודם. לבסוף, ודא שהנתיב אל שירות Web שלך תואם את המסלול הממשי של קובץ השירות asm, תוכל תמיד להעתיק את קטע הקוד מקבצי המקור המגיעים עם הספר.

כיצד זה פועל

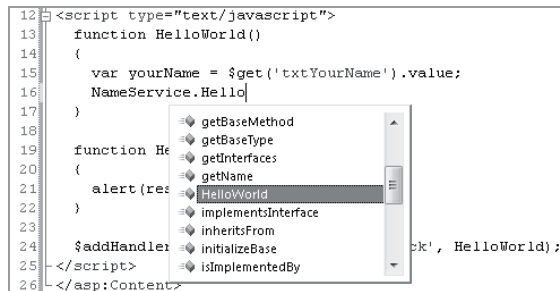
שירות Web שהשתמשת בו בדוגמה זו כמעט וזהה לשירות שהשתמשת בו בדף המבחן שבתרגיל קודם. ההבדל היחיד הוא המאפיין ScriptService שמסמן את השירות כנגיש לקוד של תסריטי צד-לקוח.

כדי לחשוף את השירות לתסריט צד-לקוח ביישום שלך, עליך לרשום אותו. תוכל לעשות זאת בדף הראשי באלמנט <Services> של <ScriptManager>. המגרעת של רישום שירות Web בדף הראשי היא, שכל דף באתר פונה אל JavaScript שבצד הלקוח. עבור שירות שבו תשתמש פעם או פעמיים בלבד, עדיף בהרכבה להוסיף ScriptManagerProxy לדף (או לדפים) הרצויים ולרשום בהם את השירות:

```
<asp:ScriptManagerProxy ID="ScriptManagerProxy1" runat="server">
  <Services>
    <asp:ServiceReference path("~/WebServices/NameService.asmx" />
  </Services>
</asp:ScriptManagerProxy>
```

כל שעליך לעשות הוא לפנות לשירות על ידי הגדרת המאפיין Path. כמו עם כל URL אחר של צד-שרת שראית בספר עד כה, באפשרותך להשתמש בתחביר גל (~) כדי לפנות לשורש היישום.

לאחר רישום השירות, הוא נעשה זמין לקוד צד-לקוח. שים לב ש-IntelliSense ב-VWD מתוכם דיו כדי לגלות את שירותי Web שהגדרת ורשמת. לאחר שתקליד בבולוק תסריט צד-לקוח את המילה NameService ואחריה נקודה, ייכנס IntelliSense לפעולה ויציג את השיטות הציבוריות שמצא. תרשים 10-12 מציג את הרשימה של IntelliSense ומבליט בה את השיטה HelloWorld.



תרשים 10-12

מציאת השירותים הנכונים שהוגדרו באתר הופכת בדרך זו לקלה ביותר. זהו שיפור עצום לעומת גרסאות קודמות של Visual Studio שרשימת IntelliSense שלהן הייתה בעלת מספר קבוע של פריטים שקשורים ל-JavaScript. רכיב IntelliSense של Visual Web Developer 2008 מסוגל למעשה לסרוק את הקוד שלך ולמלא את הרשימה בשמות המשתנים הנכונים, השיטות, השירותים וכך הלאה, כפי שהוא מוצא בזמן סריקת הקוד.

כדי לראות כיצד פועל הדף הממשי ולגשת לשירות Web, התבונן בקוד הבולוק <script>.

הקוד הראשון להתבונן בו הוא השיטה HelloWorld:

```
function HelloWorld()  
{  
    var yourName = $get('txtYourName').value;  
    NameService.HelloWorld(yourName, HelloWorldCallback);  
}
```

בקוד זה יש פנייה לתיבת הטקסט שיצרת קודם. כדי לגשת לתיבת הטקסט עם JavaScript רגיל, היית משתמש בדרך כלל, ב- `document.getElementById('txtYourName')`. עם זאת, סביבת העבודה של Ajax של צד-לקוח מציעה קיצור דרך בשם `$get` שבמהותו מבצע את אותה הפונקציה. ברגע שיש פנייה לתיבת הטקסט, באפשרותך לגשת למאפיין הערך (value) שלה כדי לקבל את השם שהשתמש הזין.

שם זה נשלח לשיטה HelloWorld (שירות Web) על ידי הקוד הבא:

```
NameService.HelloWorld(yourName, HelloWorldCallback);
```

הארגומנט הראשון בקריאה ל-HelloWorld הוא הארגומנט שהשיטה מצפה לו, מחרוזת שמכילה את שמך. הארגומנט השני, HelloWorldCallback, הוא פנייה לשיטה שמוזנקת כאשר השירות מחזיר את התוצאה שלו. שיטה זו קרויה שיטת הקריאה חזרה (callback), מכיוון שהיא זו שמתבצעת לאחר שהשירות מחזיר את המידע שבידו.

מטבע הדברים, הקריאה לשירות Web היא אסנכרונית. פירוש הדבר שהקריאה לשירות נעשית מתוך הליך ביצוע נפרד מתוך היישום, והשיטה HelloWorld מתקיימת זמן קצר אחר כך. מכיוון שתגובת שירות Web עלולה להשתנות זמן רב, עליך לציין שיטה שתהיה אחראית לטיפול בתגובה כשזו תחזור מהשירות. במקרה שלנו, השיטה האחראית לקבלת התשובה נקראת כאן HelloWorldCallback, אך ניתן לקבוע לה שם כלשהו.

בנוסף לשיטה שאחראית לטיפול כאשר השירות מתבצע בהצלחה, באפשרותך להוסיף שיטה שתעורר כאשר השירות נכשל מסיבה כלשהי. למשל, שירות יכול להיכשל משום שחיבור הרשת קרס או משום שהשירות הוציא שגיאה. במקרה זה תיראה הקריאה ל-HelloWorld כך:

```
NameService.HelloWorld(yourName, HelloWorldCallback, ErrorCallback);
```

הפונקציה ErrorCallback יכולה אז להיראות כך:

```
function ErrorCallback(error)  
{  
    alert(error.get_message());  
}
```

לאובייקט השגיאה שמועבר לשיטה זו יש שיטות ומאפיינים שנוחים להצגת המידע אודות השגיאה. כמו בדוגמה, השתמש ב- `get_message()` כדי להגיע לתיאור השגיאה המקורית שקרתה בשרת.

אם הכל מתנהל כשורה, הקריאה ל-HelloWorld בצד-הלקוח תעורר את השיטה HelloWorld (שירות Web). שיטה זו מקבלת את השם ומחזירה הודעת פתיחה ידידותית, כפי שראית קודם.

כששירות Web מחזיר את הערך שלו, מוזנקת השיטה HelloWorldCallback. לשיטה זו יש פרמטר בשם result שמכיל את הערך המוחזר על ידי שירות Web:

```
function HelloWorldCallback(result)
{
    alert(result);
}
```

הפרמטר result של התרגיל הקודם הוא מחרוזת פשוטה, לכן יכולת להשתמש ב- alert(result) כדי להציג את התוצאה באופן ישיר בחלון ההתראה של JavaScript.

במצבים אחרים, יכול הפרמטר result להכיל אובייקטים מורכבים יותר שמספקים גישה למאפיינים שלו. הדבר המגניב לגבי סביבת העבודה ASP.NET AJAX הוא, שרוב העבודה הכרוכה בטיפול באובייקטים מורכבים אלה שקופה מבחינתך. שירות Web עשוי להחזיר אובייקט מורכב כמו Person, Order או Review. סביבת העבודה ASP.NET AJAX עושה את האובייקט הזה לזמין לתסריט צד-לקוח באופן אוטומטי, מבלי שתצטרך לקודד קוד מותאם להפיכת האובייקט ממופץ צד-שרת למשהו שמובן על ידי JavaScript של צד-לקוח.

הדבר האחרון שראוי לתת עליו את הדעת הוא כיצד הכל התחיל. כשלחצת על הלחצן, הופעלה הפונקציה HelloWorld של צד-לקוח באופן אוטומטי. איך זה ייתכן? התשובה נעוצה בקריאה המסתורית ל-\$addHandler:

```
$addHandler($get('btnSayHello'), 'click', HelloWorld);
```

\$addHandler היא למעשה קיצור דרך לשיטה addHandler של המחלקה Sys.UI.DomEvent שמוגדרת בסביבת העבודה Ajax. אפשר להשתמש בה לרישום מטפלי אירוע עבור אירועים מסוימים של האובייקטים בדפים. הדבר דומה למטפלי האירוע שראית עד כה בקוד צד-שרת הן ב-VB.NET והן ב-C#.

בדוגמה שלפנינו השתמשנו שוב ב-\$get כדי לפנות ללחצן. הפרמטר השני הינו האירוע Click של הלחצן שמחובר לשיטה HelloWorld (הפרמטר השלישי). פירוש הדבר שבכל פעם שתלחץ על הלחצן יתבצע הקוד בפונקציה HelloWorld.

בדיוק כמו \$get כך גם השיטה \$addHandler היא קיצור דרך נוח לרישום אירועים, ופועלת ברחבי הדפדפן. אמנם, רישום אירוע כמו Click של הלחצן אפשרי גם ללא סביבת העבודה של Ajax בצד הלקוח, אבל השיטה \$addHandler מקלה מאוד על רישום אירועים ברחבי הדפדפן בדרך נקייה ותמציתית, וכשהטבה נוספת מאפשרת ליהנות גם מ-IntelliSense.

הדבר המגניב לגבי \$get ו-\$addHandler הוא שאפשר להשתמש בהן בכל אתר. כל שעליך לעשות הוא לכלול את הפקד ScriptManager בדף ראשי או בדף תוכן, ואתה יכול להשתמש בסביבת העבודה של צד-לקוח. כדי שזה יקרה אינך צריך להשתמש בדף בשירותי Web או בפקדים קשורים-Ajax אחרים. הדבר מקל מאוד על כתיבת פונקציונליות של JavaScript מתקדמת עם כמות מזערית של קוד.

ברור שלשירות NameService שראית בפרק זה יש שימוש מועט בעולם האמיתי. עם זאת, העקרונות של שירותי Web שלמדת בפרק זה יכולים להיות מיושמים בקלות על ידי שירותים מורכבים יותר, שיאפשרו גישה אל נתונים בשרת מתוך JavaScript של צד-לקוח שאורכו שורות קוד ספורות.

כשנדון בניפוי (debugging), בפרק 17, תפגוש שוב את NameService. בפרק 17 תצטרך דרך הקוד שורה אחת שורה כדי שתוכל לראות איזה קוד מתבצע ובאיזה סדר.

זו רק ההתחלה

עד כה ראית כמה מהשיטות והאובייקטים שמציעה סביבת העבודה Ajax של צד-לקוח. עם זאת, \$get ו-\$addHandler הן רק קצה הקרחון. סביבת העבודה של צד-לקוח מכילה גם ששה מרחבי שמות וגם מרחב שמות גלובלי שבתורם נותנים גישה ליותר מ-30 מחלקות עם מאות שיטות שימושיות שיכולות לסייע בבניית ממשקי צד-לקוח עשירים.

באפשרותך להשתמש בסביבת העבודה של צד-לקוח כדי להגדיר ולטפל באירועי דף כמו load ו-unload, לבקש דפי אינטרנט אחרים תוך שימוש במחלקה WebRequest, להציג נתונים בעיצובים שונים המתייחסים להגדרות אזוריות ותאימות שפה של הלקוח על ידי קוד מהמחלקה CultureInfo, ולהפעיל רבות מהשיטות שמרחיבות את הסוגים הרגילים של JavaScript, כמו startsWith ו-endsWith עבור משתני מחרוזת.

כדי לקבל מידע מפורט על כל סביבת העבודה Ajax של צד-לקוח, פנה לתיעוד הרשמי ב- www.asp.net/AJAX/Documentation/Live/ClientReference/.

עצות מעשיות להפעלת Ajax

עצות אלה תעזורנה לך ליצור חוויות משתמש טובות יותר:

⊙ קשה למדי לעצב דף Ajax, מכיוון שתוכן ה-UpdateProgress ניתן לצפייה רק בעת טעינת הדף. אפשר לראות את התוכן במשך מספר שניות, או פחות, ורק לאחר גרימת postback אל השרת. כדי להקל על עיצוב ה-UpdateProgress אתה חייב להתחיל בעיצוב ההודעה מחוץ ללוח UpdateProgress. לדוגמה, בתרגיל שבפרק זה עליך להזיז את `<div id="PleaseWait">` אל מחוץ לכל הפקדים האחרים כדי שתוכל לראות זאת כל הזמן. תוכל אז לשנות את קוד HTML ואת קוד CSS של `<div>` עד שהתצוגה תהיה לשביעות רצונך. בנקודה זו תוכל להעביר את `<div>` חזרה ל-UpdateProgress כדי שיוצג רק במשך עדכון חלקי של הדף.

⊙ בכל פעם שתשתמש ב-UpdatePanel, חשוב גם על הוספת הפקד UpdateProgress הקשור אליו. גם אם אינך רואה צורך בכך מפני ש-UpdatePanel מתרענן ממש במהירות, עדיין כדאי להוסיף את UpdateProgress עבור אנשים בעלי מחשבים איטיים או רשתות איטיות.

עדיף מכך: הוסף UpdateProgress לדף הראשי בשטח גלוי ונוח של הדף (למשל, באזור הכותרת התחתונה). אל תגדיר את המאפיין AssociatedUpdatePanelID כדי שה-UpdateProgress יוצג בכל מקרה של קריאה חזרה. בדרך זו לא תזדקק למספר רב של מחוונים בשטחים שונים של הדף.

⊙ אל תעשה שימוש יתר בפקדים UpdatePanel. במצבים רבים, ביצועי היישום משתפרים רק לכאורה כתוצאה משימוש בפקדים UpdatePanel, בעוד הביצועים האמיתיים נשארים בעינם. מצד אחד, זה טוב מכיוון שהמשתמשים סבורים שהיישום מהיר יותר מאשר ללא UpdatePanel. מצד שני, שימוש בפקדי UpdatePanel רבים מדי עלול ליצור מצב בו הדף חוזר מהשרת והמשתמש לא הבחין בכך, במיוחד כשאינך משתמש בפקד UpdateProgress שמעדכן אותם לגבי ההתרחשויות.

סיכום

Ajax היא טכנולוגיה רחבה ומאוד מעניינת שיכולה להוסיף ערך רב לאתר שלך. ניתן לחלקה לשני תחומים שונים, פקדי צד-שרת וסביבת העבודה JavaScript של צד-לקוח.

כדי להתחיל, הפקד UpdatePanel מאפשר ליצור, באפס זמן, דפים חופשיים מהבהוב. כל שעליך לעשות הוא לעטוף חלק מהתוכן על ידי הפקד UpdatePanel ולהוסיף ScriptManager. כדי להימנע מהוספת ScriptManager לכל דף שזקוקים לו, אתה יכול להוסיף אותו לדף הראשי של האתר, כדי שיהיה זמין בכל דפי האתר.

למרות ש-UpdatePanel פותר חלק מהבעיות הקשורות לעדכון "מכוער" של הדף, הוא מביא אתו בעיות חדשות משלו. מכיוון שהדף אינו נטען בפועל, המשתמש עלול לא להיות מודע לכך שהדף עסוק ומעובד ברגע זה, ועשוי להתחיל שוב ללחוץ על לחצנים. הדבר עלול לגרום לבעיות בשרת, כמו שליחת אותה הודעת דואר אלקטרוני יותר מפעם אחת. כדי להתגבר על בעיה זו, שקול חיבור פקד UpdateProgress לכל UpdatePanel שאתה משתמש בו. התכנים של הפקד UpdateProgress מוצגים במהלך הרענון של דף Ajax, כדי שלמשתמש יהיה רמז חזותי שמהשו מתרחש במערכת. כדי למנוע את הופעת UpdateProgress במהלך רענוני דף קצרים מאוד, עליך להגדיר את המאפיין DisplayAfter, אשר דוחה את הופעת UpdateProgress למשך פרק הזמן שצוין.

בנוסף, ASP.NET AJAX כוללת גם סביבת צד-לקוח עשירה, שמאפשרת גישה לשירותי Web באתר בעזרת שורות קוד ספורות. לעתים קרובות שירותי Web שתיצור ייראו דומים לשיטות אחרות שיצרת עד כה. ההבדל הגדול ביותר הוא שאתה מגדיר אותם בתוך מחלקה שיורשת מ-System.Web.Services.WebService שבתוך קובץ ASMX בעל קוד ברקע נפרד, שמוצב בתיקייה App_Code.

לאחר יצירת שירות Web, סמן בשיטות מסוימות את המאפיין WebMethod, כדי לעשות אותן זמינות לקוד שיקרא להן.

בנקודה זו תוכל לקרוא לשירות Web על ידי שימוש בפורמט `ServiceName.MethodName`. מכיוון ששירותי Web נקראים באופן אסנכרוני, עליך לציין שיטה שתמתין ותטפל בתגובה החוזרת משירות Web. בדוגמאות שראית זו הייתה השיטה `HelloWorldCallback`. בתוך שיטה זו יש גישה לפרמטר `result`, שמועבר משיטת Web לקוד הלקוח.

כדי להפעיל פקדים בדף, כמו למשל לאיסוף קלט משתמש או להצגת תוצאות משירות Web, עומדת לרשותך סביבת העבודה העשירה של Ajax עבור צד-לקוח. סביבת עבודה זו מציעה מאות שיטות שימושיות כדי לטפל ברוב ההיבטים של הצגת הדף בדפדפן. כל שעליך לעשות הוא לכלול פקד `ScriptManager` בדף הראשי או בדף התוכן שלך, ותוכל להשתמש בכל סביבת העבודה Ajax של צד-לקוח בדפי האינטרנט שלך.

למרות ש-Ajax עצמה היא טכנולוגיה משכנעת מאוד, היא שימושית עוד יותר בתרחישים מונעי-נתונים (data-driven). לדוגמה, שימוש בפקד `UpdatePanel` סביב הרשומות המוחזרות מבסיס נתונים, כדי למנוע הבהוב בזמן מיון, סינון או דפדוף הנתונים, משפר מאוד את חווית הגלישה של המשתמש. בפרק 11 תלמד כיצד לעבוד עם בסיסי נתונים. הידע שצברת בפרק זה לגבי Ajax יסייע לך ליצור במהירות דפי אינטרנט שמונעי מבסיס-נתונים וחופשיים מהבהוב.

תרגילים

1. הקטגוריה `AJAX Extensions` של `Toolkit` מגדירה את `ScriptManager` ואת `ScriptManagerProxy`. הסבר את ההבדל בין שני פקדים אלה, והסבר מתי להשתמש בכל אחד מהם.
2. איך תוכל להודיע למשתמשים שלך שמתבצע עדכון חלקי של הדף?
3. כדי לחשוף שיטה באתר כשירות Web שיכול להיקרא על ידי תסריט צד-לקוח, עליך ליצור מחלקה מיוחדת ולהחיל בה שני מאפיינים. איזו מחלקה עליך ליצור, ואיזה מאפיינים תצטרך להחיל?

פרק 11

מבוא לבסיסי נתונים

היכולת להשתמש בבסיס נתונים (database) באתר אינטרנט מבוסס ASP.NET 3.5 חיונית בדיוק כמו הבנת HTML ו-CSS. ללא שימוש בבסיסי נתונים כמעט בלתי-אפשרי לבנות אתר אינטרנט מודרני ובעל אפשרויות מתקדמות. השימושיות של בסיסי נתונים נובעת מכך שהם מאפשרים לאחסן ולאחזר נתונים בדרך מובנית ונוחה לגישה. התועלת הגדולה ביותר של בסיסי נתונים היא בכך שניתן לגשת אליהם בזמן ביצוע בנחיות ובמהירות. משמעות הדבר, שאינך מוגבל עוד לקבצים הסטטיים יחסית שיצרת בזמן העיצוב ב- Visual Web Developer. ניתן לאחסן בבסיס נתונים טקסטים מסוגים שונים, סגנונות מוזיקליים, תמונות, מידע אודות המשתמשים (שמות המשתמשים, כתובות דואר אלקטרוני, סיסמאות ועוד), מידע אודות מי שקורא את הסקירות/ביקורות וכו', והעיקר - לגשת לנתונים אלה מדפי ASPX.

הדבר נותן גמישות רבה בהצגת הנתונים, ומאפשר יצירת אתרי אינטרנט דינמיים שמסוגלים להתאים עצמם להעדפות המבקרים, לתוכן שיש לאתר להציע, וגם לתפקידים או להרשאות הגישה שיש למשתמשים.

כדי שתוכל לפנות לבסיס הנתונים מתוך דף ASPX, תלמד בפרק זה כיצד לגשת לבסיס נתונים תוך שימוש בשפת השאילתות **SQL** (Structured Query Language). שפה זו מאפשרת לאחזר נתונים שמאוחסנים בבסיס נתונים ולפעול עליהם. תלמד גם להשתמש בכלים המיועדים לעבודה מול בסיס נתונים שמספקת VWD כדי ליצור טבלאות ושאילתות.

ASP.NET וסביבת העבודה .NET. מציעות טכנולוגיות וכלים רבים לפעולה על בסיסי נתונים, שהפעלתם אינה דורשת הבנה מעמיקה של מושגי היסוד. למרות זאת, הבנה כזו עדיין חשובה. ברגע שתדע לגשת לבסיס נתונים, תמצא שיותר קל להבין טכנולוגיות אחרות, כמו LINQ to SQL, שמספקות גישה קלה יותר לפעולות בסיס נתונים ישירות מהקוד.

פרק זה משמש כמבוא לפעולה עם בסיסי נתונים. תלמד בו בהרחבה על הנושאים הבאים:

- ⊙ מהו בסיס נתונים, ובאיזה בסיסי נתונים משתמשים דפי ASP.NET בדרך כלל?
- ⊙ מהי SQL, איך היא נראית וכיצד להשתמש בה?
- ⊙ איך לאחזר נתונים מבסיס נתונים באמצעות המשפט SELECT?
- ⊙ איך לפעול על נתונים בבסיס נתונים על ידי מבני SQL אחרים?
- ⊙ מהם קשרים בבסיס נתונים ומדוע הם חשובים?

⊙ איזה כלים זמינים ב-VWD לניהול אובייקטים כמו טבלאות ושאליות, ואיך להשתמש בכלים אלה?

בפרקים הבאים תיישם את מה שתלמד בפרק זה. בפרק 12 תלמד כיצד להשתמש בפקדים המובנים כדי לבצע פעולות שונות על הנתונים בבסיס הנתונים שלך. בפרק 13 תלמד להשתמש ב-LINQ כשכבה נוספת על בסיס הנתונים כדי לגשת לנתונים בדרך מוכוונת-עצמים על ידי שימוש במעט קוד.

בסעיפים הבאים תלמד מהו בסיס נתונים, ואיזה סוגים שונים של בסיסי נתונים עומדים לרשותך.

מהו בסיס נתונים?

בסיס נתונים, לפי ההגדרה הפשוטה ביותר, הוא אוסף של נתונים שמאורגנים באופן שיהיה קל לגשת אליהם, לנהל ולעדכן אותם.

הסוג הנפוץ ביותר של בסיס נתונים הוא **בסיס נתונים טבלאי** (relational database). זהו סוג בסיס הנתונים שבו משתמשים לרוב באתרי אינטרנט, ובו נשתמש במהלך הלימוד בספר.

בסיס נתונים טבלאי מיישם את הרעיון של **אוסף טבלאות** שבהן מאוחסנים הנתונים בשורות ועמודות בדומה לגיליון אלקטרוני. כל שורה בטבלה מכילה את כל המידע אודות פריט שמאוחסן בטבלה על פי נושא הטבלה (שכר, מלאי, ציוני תלמידים וכד'). כל עמודה מכילה מידע אודות מאפיין מסוים של הרשומה הזו (שם, מס' קטלוגי, ציון במקצוע וכד').

המונח **"טבלאי"** מתייחס לדרך שבה טבלאות שונות בבסיס הנתונים יכולות להיות קשורות זו לזו. במקום לשכפל את אותם הנתונים שוב ושוב, אתה מאחסן נתונים חוזרים בטבלה נפרדת, ויוצר קישור לנתונים אלה מטבלאות אחרות. עיין בטבלה הפשוטה הקרויה Review שמוצגת בתרשים 1-11. טבלה זו יכולה לאחסן סקירות/ביקורות על תקליטורים או קונצרטים שמוצגות באתר האינטרנט Planet Wrox.

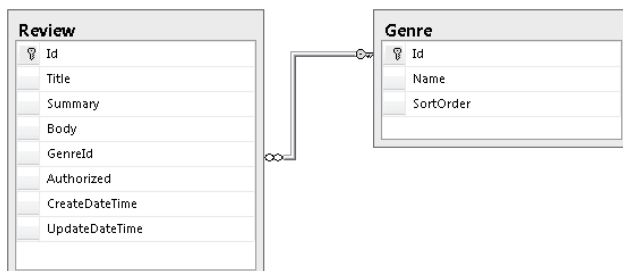
כפי שאפשר לראות בתרשים 1-11, כל סקירה (שורה) משויכת לסגנון (ז'אנר) מוזיקלי כמו Rock או Techno. מה יקרה אם תרצה לשנות את שם הסגנון מ-Techno ל-Hardcore Techno? במקרה כזה תצטרך לעדכן את כל הרשומות שסגנון זה מוקצה להן. אם גם טבלאות אחרות מאחסנות סגנון, תצטרך לערוך גם בהן את השינוי הנדרש באופן ידני.

Review: Query...LANETWROX.MDF			
Id	Title	Genre	CreateDateTime
13	Editors - An End Has A Start	Indie Rock	12/23/2007 10:38:58 PM
14	Blackout by Britney Spears: Should I laugh or should I cry?	Pop	12/17/2007 11:19:24 PM
15	Opgezwolle - Vloeistof	Rap and Hip-Hop	12/18/2007 8:50:38 PM
16	Amy Winehouse - Back to Black	Blues	1/3/2008 12:52:53 PM
17	Seether - Finding Beauty in Negative Spaces	Hard Rock	1/16/2008 5:25:52 PM
18	Muse - Black Holes and Revelations	Rock	1/20/2008 8:57:25 PM
19	Nirvana - With the lights out	Grunge	1/29/2008 2:51:13 PM
20	Norah Jones - Not Too Late	Jazz	2/15/2008 9:05:54 PM
21	DJ Tiesto - In Search of Sunrise 6	Techno	3/4/2008 2:12:37 PM
22	DJ Tiesto - Elements of Life	Techno	3/5/2008 9:12:02 PM

תרשים 1-11

פתרון עדיף יהיה להשתמש בטבלה נפרדת בשם Genre, למשל. טבלה זו יכולה לאחסן שם של סגנון וגם קוד זיהוי (ID) עבור כל סגנון באופן ייחודי. הטבלה Review מקושרת לטבלה Genre, ומאחסנת רק את ה-ID של הסגנון במקום אחסון שמו המלא. תרשים 11-2 מציג את המודל המושגי של שינוי זה.

עכשיו, כשרק ID של הסגנון מאוחסן בטבלה Review, קל לשנות את שם הסגנון. כל שעליך לעשות הוא לשנות את שם הסגנון בטבלה Genre. כל הטבלאות עם קשר לסגנון זה יאספו את השינוי באופן אוטומטי. מאוחר יותר בפרק זה תראה כיצד ליצור ולהשתמש בקשרים בבסיס נתונים טבלאי.



תרשים 11-2

סוגים שונים של בסיסי נתונים טבלאיים

בפרויקטים של ASP.NET ניתן להשתמש בסוגים רבים ושונים של בסיסי נתונים מיצרנים שונים וביניהם Access, SQL Server, Oracle ו-MySQL. עם זאת, קרוב לוודאי שבסיס הנתונים הנפוץ ביותר באתרי אינטרנט מבוססי ASP.NET 3.5 הוא SQL Server של מיקרוסופט. ספר זה מציג את השימוש בגרסה SQL Server 2005 Express Edition מכיוון שזו גרסת חינם שמצורפת לתוכנה VWD, ויש לה הרבה להציע מן המוכן. יתר על כן, מגוון בסיס הנתונים של גרסה זו זהה לזה של הגרסאות המסחריות של שרת SQL, ולפיכך קל לשדרג לגרסאות אלו בשלב מאוחר יותר של מחזור הפיתוח. נתיב שדרוג זה מתואר ביתר פירוט בנספח ב'.

כדי לפעול על נתונים בבסיס נתונים, תומך SQL Server (וכל שאר מערכות בסיסי הנתונים הטבלאיים) בשפת שאילתות הקרויה SQL, אם כי ייתכנו גם שינויים קלים כפי שנראה להלן.

הערה: בזמן כתיבת שורות אלו מיקרוסופט שחררה את **SQL Server 2008 Express Edition**, אל דאגה! ממשק המשתמש והפונקציונליות שבה נשתמש זהים לחלוטין.

שימוש ב-SQL לעבודה עם נתוני בסיס נתונים

כדי להכניס ולהוציא נתונים מבסיס נתונים יש צורך להשתמש ב-SQL. זו השפה שמשתמשים בה למעשה עבור שאילתות מבסיס נתונים טבלאי, והיא מובנת על ידי רוב מערכות בסיסי הנתונים הטבלאיים. יש כמה תקנים ידועים, כשהמקובל והנפוץ מביניהם הוא התקן ANSI 92 SQL. ספקי

בסיסי נתונים רבים הוסיפו הרחבות משלהם לשפה הבסיסית שבה תומך התקן. דבר זה נותן לשפה יותר גמישות ועוצמה במחיר הקטנת היכולת לעבור מבסיס נתונים אחד לאחר.

SQL Server 2005 של מיקרוסופט אינו חריג מבחינה זו. המוצר תומך ברוב הדקדוק שהוגדר ב-ANSI 92 SQL. עם זאת, על תקן זה הוסיפה מיקרוסופט כמה הרחבות משלהם. מכלול השפה החדש קרוי T-SQL או Transact SQL. בהמשך הלימוד בספר נתייחס למונח SQL בלבד.

בסעיפים הבאים תלמד כיצד להשתמש ב-SQL מול בסיס נתונים SQL Server 2005 כדי לאחזר ולפעול על נתונים בבסיס הנתונים. עם זאת, לפני שתכתוב את המשפט הראשון ב-SQL, עליך לדעת תחילה איך להתחבר לבסיס הנתונים. התרגיל הבא מראה כיצד להתחבר לבסיס הנתונים שאיתו נעבוד במהלך הספר והקוד שלו מצורף לספר זה (לאחר שהורדת מהאתר, ראה הקדמה).

נסה זאת התחברות לבסיס הנתונים SQL Server

בתרגיל זה תלמד איך להתחבר ולפעול עם בסיס נתונים מתוך VWD. הקוד המצורף עבור פרק זה מכיל בסיס נתונים לדוגמה שמכיל כמה טבלאות. כדי שתוכל לגשת לבסיס הנתונים מתוך VWD, חייב החשבון שבו אתה משתמש כדי להתחבר למערכת Windows שלך להיות מורשה קריאה וכתיבה לתיקייה שבה מאוחסן בסיס הנתונים. אם אתה מחובר כמנהל, קרוב לוודאי שזה המצב ויש לך הרשאות אלו. ייתכן שתקבל שגיאות בזמן גישה לבסיס הנתונים בתרגיל הבא. במקרה כזה, פנה לסעיף "האבטחה ב-IIS" בפרק 18, כדי לקבל הוראות מפורטות להגדרת ההרשאות המתאימות.

1. לצורך תרגיל זה דרוש לך אתר אינטרנט נקי וריק. אפשר ליצור אחד כזה על ידי בחירת New Web Site ← File (או Web Site ← New ← File) ב-Visual Web Developer. אל תבצע תרגיל זה עם האתר Planet Wrox שפעלת בו עד כה. אתה עלול להסתבך מאוחר יותר כשתשתמש בבסיס נתונים בעל אותו שם. זכור היכן אתה שומר את האתר החדש, כי תזדקק לו בצעד הבא.

2. לאחר יצירת האתר החדש, ודא שהחשבון שלך הוא בעל הרשאות מספיקות לכתיבה לתיקייה App_Data שלו. עליך לוודא שהחשבון שלך הוא בעל הרשאות נכונות, מכיוון ש-Visual Web Developer, ולפיכך גם שרת web המובנה, פועלים תחת החשבון שאתה משתמש בו כדי להתחבר ל-Windows. כדי לוודא כי זהו אכן המצב, השתמש בסייר החלונות כדי לאתר את התיקייה App_Data שיצרת בצעד 1, לחץ עליה לחיצה ימנית ובחר Properties. עבור לכרטיסייה Security, ודא שהחשבון שלך (או של הקבוצה שאתה חלק ממנה) הוא בעל ההרשאה Modify לפחות, כפי שמוצג בתרשים 3-11.

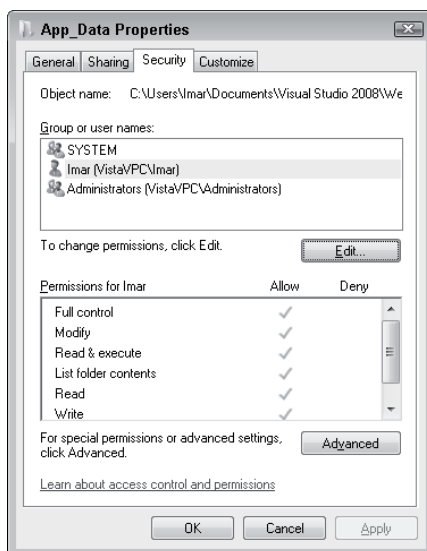
אם הכרטיסייה Security אינה קיימת, או שהחשבון שלך אינו רשום, פנה לסעיף נסה זאת "קביעת תצורת מערכת הקבצים" בפרק 18, לקבלת הוראות מפורטות להוספת החשבון שלך לרשימה.

3. פתח את התיקייה Resources ב-Resources \BegASPNET\ C:. אם תיקייה זו אינה קיימת, פנה להוראות בפרק ההקדמה של הספר. פתח את התיקייה Chapter 11 ובחר

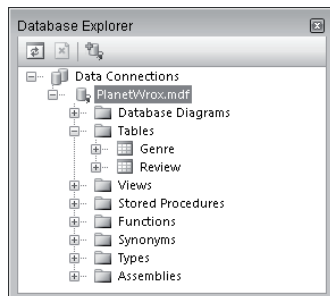
בקבצים PlanetWrox.mdf ו- PlanetWrox_log.ldf. הצג על המסך את VWD ואת סייר החלונות זה לצד זה, וב-VWD גרור את שני הקבצים שנבחרו אל התיקייה App_Data של האתר שלך. לאחר שתשחרר את קובץ mdf בתיקייה App_Data, תראה ש-VWD יצרה באופן אוטומטי את קובץ ldf. כשתישאל אם לשכתב קובץ זה, לחץ Yes. קובץ mdf הוא בסיס הנתונים הממשי ומכיל את הטבלאות והרשומות הממשיות, בעוד שקובץ ldf משמש למעקב אחר השינויים שמבוצעים בבסיס הנתונים.

4. בסייר הפתרון, לחץ לחיצה כפולה על קובץ בסיס הנתונים. פעולה זו פותחת את בסיס הנתונים בסייר בסיס הנתונים (Database Explorer), או סייר השרת, Server Explorer, בגרסאות המסחריות של Visual Studio).

5. כעת תוכל להרחיב את בסיס הנתונים המחובר כדי לגשת לאובייקטים כמו הטבלאות שהוא מכיל. ראה תרשים 11-4.



תרשים 11-3



תרשים 11-4

כיצד זה פועל

כדי שתוכל לגשת לבסיס נתונים מתוך VWD, בסיס הנתונים צריך להיות רשום בחלון סייר בסיס הנתונים תחת הכיתוב Data Connections. ברוב המקרים, רישום בסיס נתונים היא פעולה פשוטה אשר כוללת הוספת בסיס הנתונים לתיקייה App_Data ולחיצה כפולה עליו. עם זאת, כשהאבטחה במערכת שלך הדוקה, או כשאין מתחבר לקובץ נתונים פיזי מסוג SQL Server שנמצא באתר שלך, ההתחברות לבסיס הנתונים עשויה להיות שונה. במקרים אלה פנה לסעיף "האבטחה ב-IIS" בפרק 18 ולנספח ב', לקבלת פרטים נוספים על קביעת תצורת המערכת. השתמש בתיקייה App_Data רק עבור נתונים שמשתמשים בהם בשרת, כמו בסיסי נתונים וקבצי טקסט. מכיוון ששרת web חוסם את הגישה לתיקייה זו מדפדפנים מרוחקים, אינך יכול להשתמש בה כדי לאחסן תמונות שחייבות להיטען על ידי הלקוח.

כשהחיבור לבסיס הנתונים בסייר בסיס הנתונים קיים, אפשר לפעול על האובייקטים בבסיס הנתונים. שים לב שתרגיל נסה זאת הקודם לא גרם לשינויים כלשהם באתר. בתרגיל מאוחר יותר תלמד כיצד לאחסן את המידע אודות החיבור לבסיס הנתונים (נקרא מחרוזת החיבור, connection string) בקובץ web.config.

עם זאת, תחילה עליך לדעת כיצד תוכל לגשת ולשנות את הנתונים בבסיס הנתונים.

אחזור וביצוע פעולות על נתונים

חלק נכבד מהפעילות של המשתמש עם בסיס הנתונים מוקדש לאחזור נתונים ולביצוע פעולות עליהם. ארבעת סוגי הפעולה העיקריים הם: יצירה (Creation), קריאה (Reading), עדכון (Updating), מחיקה (Deleting), ריכוז פעולות אלו נקרא **CRUD**.

מכיוון שפעולות אלו כה קריטיות, תלמד בסעיפים הבאים בפירוט כיצד לבצע אותן.

קריאת נתונים

על מנת לקרוא נתונים מבסיס נתונים, יש צורך ביכולת לציין את העמודות שרוצים לשלוף מתוך הטבלה שהשאלתה הופנתה אליה. דבר זה מבוצע על ידי משפט **SELECT**. את הטבלה, או הטבלאות, שרוצים לבחור מהן נתונים מציינים על ידי מילת המפתח **FROM**. אחר כך יש צורך לסנן את הנתונים כדי להבטיח שתוחזרנה רק הרשומות שרוצים בהן. תוכל לסנן את הנתונים על ידי שימוש בפסקת **WHERE** במשפט SQL. לבסוף, תוכל לסדר את התוצאות בסדר רצוי (מיון) על ידי שימוש בפסקת **ORDER BY**.

בחירת נתונים

כדי לקרוא נתונים מטבלאות בסיס נתונים השתמש במשפט **SELECT**. בצורתו הבסיסית ביותר נראה משפט **SELECT** כך:

```
SELECT ColumnName [, AnotherColumnName] FROM TableName
```

החלקים שבין הסוגריים המרובעים הם רשות. לדוגמה, כדי לאחזר את כל השורות מהטבלה Genre תוך בחירת העמודות Id ו-Name בלבד, תשתמש במשפט SQL הבא:

```
SELECT Id, Name FROM Genre
```

מייד לאחר המילה SELECT באה רשימה של שמות העמודות מופרדות זו מזו בפסיק. מספר העמודות יכול להיות מאחת ועד לכמה שדרוש. כדי לציין שברצונך לשלוף את כל העמודות, תוכל להשתמש בתו כוכבית (*) במקום בשמות המפורשים של העמודות. עם זאת, שימוש ב- `SELECT *` נחשב לנהל תכנות נחות ועדיף להגדיר במפורש כל עמודה שברצונך לשלוף.

את שם הטבלה שממנה רוצים לשלוף נתונים מציינים מייד לאחר רשימת העמודות שלאחר מילת המפתח FROM. הדוגמה הקודמת הראתה טבלה אחת בלבד (הטבלה Genre), אבל מאוחר יותר תראה שניתן לציין טבלאות רבות תוך שימוש בציורפים (joins).

הערה: SQL אינה רגישה לרישיות. למרות זאת, הנוהל המקובל היא לכתוב את כל מילות המפתח, כמו SELECT ו-FROM, באותיות רישיות בלבד. בנוסף, ספר זה משתמש ב- Pascal Casing - כשכל מילה חדשה בשמות עמודות, טבלאות וכו' מתחילה באות רישית. לדוגמה, התאריך והשעה שבהם נוצרה סקירה מסוימת מאוחסנים בעמודה CreateDateTime בטבלה Review.

סינון נתונים

כדי לסנן נתונים, תשתמש במילה WHERE שבה באפשרותך לציין את הקריטריון שהנתונים צריכים לענות עליו. לדוגמה, כדי לשלוף מטבלת Genre את עמודת המפתח (ID) אשר עמודת השם (Name) מכילה את הערך Grunge, השתמש במשפט SQL הבא:

```
SELECT Id FROM Genre WHERE Name = 'Grunge'
```

שים לב שהמילה Grunge עטופה בגרשים יחידים. הדבר נדרש עבור נתונים מסוג טקסט וגם עבור תאריכים, כאשר מסננים נתונים או כשרוצים לשלוח ערכים למשפטי INSERT או UPDATE שמאפשרים ליצור רשומות חדשות או לעדכן רשומות קיימות, כפי שיוסבר להלן. אסור להשתמש בגרשים יחידים עבור סוגים מספריים או בוליאניים. לדוגמה, כדי לקבל רשומות שבהן ערך ID הוא 8, השתמש במשפט הבא:

```
SELECT Name FROM Genre WHERE Id = 8
```

שתי הדוגמאות לעיל הראו שימוש במילה WHERE עם סימן השוויון (=) לקבלת התאמה מדויקת. עם זאת, ניתן להשתמש באופרטורים אחרים עבור קריטריונים אחרים. הטבלה הבאה מפרטת כמה אופרטורים נפוצים של השוואה, שאפשר להשתמש בהם בפסקה WHERE:

אופרטור	תיאור
=	האופרטור "שווה" תואם, רק כאשר הצד הימני והצד השמאלי של הביטוי זהים.
>	האופרטור "גדול מ-" תואם, כאשר הצד השמאלי של הביטוי מייצג ערך גדול יותר מאשר צדו הימני.

אופרטור	תיאור
>=	האופרטור "גדול או שווה" תואם, כאשר הצד השמאלי של הביטוי גדול מ- או שווה ל- צדו הימני.
<	האופרטור "קטן מ-" תואם, כאשר הצד השמאלי של הביטוי מייצג ערך קטן יותר מצדו הימני.
<=	האופרטור "קטן או שווה" תואם, כאשר הצד השמאלי של הביטוי גדול מ- או שווה ל- צדו הימני.
<>	האופרטור "לא שווה" הוא היפוכו של אופרטור השוויון, ותואם כאשר שני צדי ביטוי ההשוואה שונים זה מזה.

SQL תומכת בכמה אופרטורים לוגיים, כמו AND ו-OR, שמאפשרים לשלב כמה קריטריונים של WHERE. השפה SQL תומכת גם באופרטורים נוספים לחיפוש טקסט ולציון טווחים. הטבלה הבאה מפרטת כמה מהאופרטורים ומתארת למה הם משמשים.

אופרטור	תיאור
AND	אופרטור זה מאפשר לצרף שני ביטויים. לדוגמה, <pre>...WHERE Id > 20 AND Id < 30</pre> פסקה זו גורמת לאחזור של כל הרשומות שה-ID שלהן נמצא בטווח שבין 20 ל-30 (20 ו-30 עצמם אינם כלולים).
OR	האופרטור מאפשר להגדיר קריטריונים רבים שמתוכם לפחות אחד חייב לתאום, אך אפשר שתהיינה התאמות נוספות. לדוגמה, <pre>...WHERE Id = 12 OR Id = 27</pre> גורם לאחזור הרשומות 12 ו-27.
BETWEEN	מאפשר לציינ טווח של ערכים שברצונך שיהיו בתחום שבין גבול עליון לתחתון. לדוגמה, <pre>...WHERE Id BETWEEN 10 AND 35</pre> גורם לאחזור של כל הרשומות שה-ID שלהן הוא בין 10 ל-35 (כולל 10 ו-35 עצמם, אם הם קיימים בבסיס הנתונים).
LIKE	משמש לקבוע אם ערך תואם לתבנית מסוימת. אפשר להשתמש בתווי הכללה כמו % (כל מחרוזת של אפס או יותר תווים) וקו תחתון (כל תו יחיד) כדי להתאים חלקים מסוימים של הערך. לדוגמה, <pre>...WHERE Name LIKE '%rock%'</pre> פסקה זו מחזירה את כל הרשומות בטבלה שהמחרוזת rock כלולה בשם, כולל Indie Rock, Hard Rock וכדומה.

לאחר הגדרת דרישות הסינון שלך על ידי פסקת WHERE, אתה עשוי לרצות לשנות את הסדר שבו מוחזרות התוצאות מבסיס הנתונים. הדבר מתבצע על ידי הפסקה ORDER BY.

סידור נתונים

הפסקה ORDER BY מופיעה בסופו של משפט SQL ויכולה להכיל שם של עמודה אחת או יותר, וביטויים ASC או DESC כדי לקבוע אם הפריטים ימוינו בסדר עולה (ASC), שהיא ברירת המחדל, אם אינך מציין את מילת המפתח) או בסדר יורד (DESC).

לדוגמה, כדי לאחזר את כל הרשומות מהטבלה Genre, ולמיין אותן אלפביתית לפי השמות בסדר עולה, תוכל להשתמש במשפט SQL הבא:

```
SELECT Id, Name FROM Genre ORDER BY Name
```

מכיוון שסדר עולה זו ברירת המחדל, אין צורך לציין את מילת המפתח ASC, אם כי ניתן לעשות זאת כמובן. הדוגמה הבאה שוות-ערך לקודמת מבחינה פונקציונלית:

```
SELECT Id, Name FROM Genre ORDER BY Name ASC
```

אילו רצית להחזיר את אותן הרשומות, אבל בסדר מיון הפוך על פי העמודה SortOrder, היית משתמש בתחביר הבא:

```
SELECT Id, Name FROM Genre ORDER BY SortOrder DESC
```

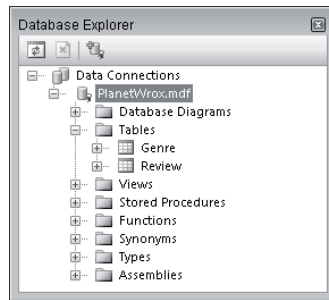
שים לב שבעזרת הפסקה ORDER BY באפשרותך לסדר לפי עמודות שנמצאות ברשומה (שורה של הטבלה) אך אינן חלק ממשפט SELECT, כמו במקרה של העמודה SortOrder לעיל. לסיכום נקודה זו: גם אם עמודה מסוימת אינה חלק מקבוצת התוצאה הסופית, עדיין ניתן להשתמש בה ולמיין לפיה.

בתרגיל הבא תראה כיצד לבצע כמה שאילתות על בסיס הנתונים לדוגמה. שאילתות אלו יתנו לך מושג על הדרך שבה יכולות שאילתות שונות להשפיע על התוצאות המוחזרות מבסיס הנתונים.

נסה זאת בחירת נתונים מבסיס הנתונים לדוגמה

בתרגיל זה תשתמש בבסיס הנתונים שהתחברת אליו בתרגיל קודם. בסיס נתונים זה משמש רק עבור הדוגמאות של פרק זה, ולכן אל תדאג אם הנתונים משתבשים בו.

1. פתח את סייר בסיס הנתונים (או סייר השרת, בגרסאות המסחריות של Visual Studio), אתר את הכיתוב Data Connection שהוספת קודם, הרחב אותו, והרחב את הצומת Tables. אתה אמור לראות שתי טבלאות, Genre-1 Review, כפי שמוצג בתרשים 5-11.
2. לחץ לחיצה ימנית על הטבלה Genre ובחר Show Table Data. בחלון העורך הראשי אתה אמור לראות כעת את הרשימה של כל הסגנונות שזמינים בטבלה Genre, כפי שמוצג בתרשים 6-11.



תרשים 11-5

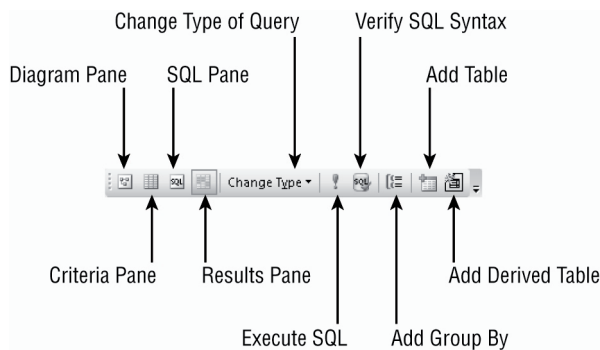
Genre: Query(...ANETWROX.MDF)

	Id	Name	SortOrder
▶	1	Rap and Hip-Hop	14
	2	Pop	12
	3	Jazz	8
	4	Hard Rock	3
	5	Indie Rock	7
	6	Punk	1
	7	Rock	2

1 of 12 Cell is Read Only.

תרשים 11-6

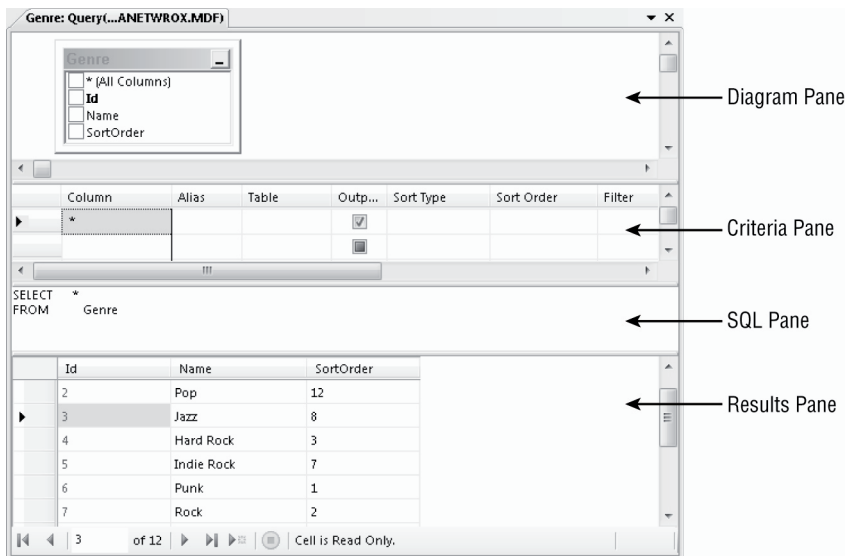
⊙ שים לב שזו אינה רק רשימת כל הרשומות בטבלה Genre. זוהי למעשה התוצאה של שאילתת SELECT של SQL שבוצעה כשפתחת את החלון. כדי לראות את השאילתה שמאחורי הרשימה, ודא שסרגל הכלים Query Designer, שמוצג בתרשים 11-7, נראה במסך. אם סרגל הכלים אינו נראה, לחץ לחיצה ימנית על סרגל כלים קיים ואז לחץ Query Designer.



תרשים 11-7

⊙ בסרגל הכלים, לחץ על הלחצנים של החלוניות Criteria, Diagram ו-SQL כדי לפתוח את החלוניות שלהם בהתאמה. שים לב שהם מוצגים בחלון המסמך זה על יד זה. ארבעת הלחצנים הראשונים בארגז הכלים אמורים להיות כעת במצב "לחוך", וחלון המסמך אמור להיות מפוצל לארבעה אזורים, כשכל אזור מתאים

לאחד מארבעת הלחצנים בסרגל הכלים. תרשים 8-11 מציג את חלון המסמך במלואו, עם ארבע החלוניות.



תרשים 8-11

חלונית SQL מציגה את משפט SQL שמשמש לשליפת הסגנונות שמוצגים בתחתית המסך. כאן משפט SQL הוא: `SELECT * FROM Genre`. שפירושו: שליפת כל העמודות והרשומות מהטבלה. כמובן שאתה יכול לשנות זאת בקלות.

3. בחלונית SQL, הצב את הסמן מייד לאחר המילה `Genre`. הקש `Enter` פעם אחת ואז הקלד `WHERE Id > 4`. משפט SQL השלם אמור להיראות כעת כך:

```
SELECT * FROM Genre WHERE Id > 4
```

בחלונית SQL שלך, השאילתה מפוצלת על פני כמה שורות ללא צורך בתו המשך שורה. זה בסדר מכיוון ש-SQL מאפשרת לפרוש את המשפטים על פני כמה שורות ללא צורך בתו המשך שורה.

4. כדי לוודא שמשפט SQL שכתבת אמנם תקף, לחץ על הלחצן `Verify SQL Syntax` בסרגל הכלים, ואז תקן את השגיאות במשפט SQL (אם יש כאלו). עכשיו לחץ על הלחצן `Execute SQL` (בעל סימן הקריאה האדום עליו) או הקש `Ctrl+R`. בשני המקרים מבוצע משפט SQL והחלונית Results תציג את כל רשומות הסגנון בעלות ID גדול מ-4.

5. בנוסף להצגת התוצאות, VWD גם שינתה את השאילתה, ובמקום `SELECT *` היא רשמה כל עמודה בטבלה במפורש. כעת הבט בחלונית Diagram, החלק העליון של תיבת הדו-שיח בתרשים 8-11, שמציגה את הטבלה כולה. בחלונית Diagram אתה יכול לסמן את שמות העמודות כדי לקבוע אילו עמודות תופענה בשאילתה, או לבטל סימון כזה. בטל את בחירת העמודה `SortOrder` (היזהר ואל תשנה במקום זאת בטעות את

סימון העמודה Output בחלונית Criteria). שים לב שהיא גם הוסרה מהחלונית Criteria (מוצגת בתרשים 9-11) וממשפט SQL שנראה בחלונית SQL.

6. התבונן בחלונית Criteria. היא מציגה את שתי העמודות שבחרת. בעמודה Filter היא מציגה את הביטוי שמסנן את כל הסגנונות בעלי ID גדול מ-4, כפי שמוצג בתרשים 9-11.

Column	Alias	Table	Output	Sort Type	Sort Order	Filter	Or...	Or...	Or...
Id		Genre	☒			> 4			
Name		Genre	☒						
			☐						

תרשים 9-11

בחלונית זו אפשר לשנות את השאילתה מבלי לכתוב קוד רב. כדי לראות כיצד להחיל מסנן נוסף, הקלד 'LIKE "%rock%" בתא Filter של השורה Name. הדבר מגביל את התוצאות לכל רשומות הסגנון שמכילות את המילה rock ובהן ה-ID גדול מ-4.

7. כדי לקבוע את סדר המיון, אפשר להשתמש בעמודה Sort Type. אפשר לעשות זאת עבור עמודות נראות (לדוגמה, אלו שתיתבת הסימון Output שלהן מסומנת, ומופיעות בקבוצת התוצאות הסופית), אבל אפשר לעשות זאת גם עבור עמודות אחרות. כדי למיין לפי העמודה SortOrder, לחץ פעם אחת על התא שתחת Name. הוא משתנה ומציג כעת רשימה נפתחת. מהרשימה הנפתחת בחר SortOrder. כשאתה לוחץ על השדה או מדלג ממנו והלאה, VWD מסמנת את העמודה Output. באפשרותך ללחוץ על סימן זה כדי להסיר את העמודה שוב מהפלט, באופן שהיא תישאר זמינה לסידור וסינון, אבל לא תוצג בתוצאות השאילתה. עם זאת, עבור תרגיל זה אפשר להשאיר את בחירת העמודה.

8. בעמודה Sort Type בחר Descending מהרשימה שנפתחה עבור SortOrder. חלונית Criteria הסופית תיראה עכשיו כמו תרשים 10-11.

Column	Alias	Table	Output	Sort Type	Sort Order	Filter	Or...	Or...	Or...
Id		Genre	☒			> 4			
Name		Genre	☒			LIKE "%rock%"			
SortOrder		Genre	☒	Descendi...	1				
			☐						

תרשים 10-11

בעורך עורך את השינויים ומשתמש בחלוניות Diagram ו-Criteria-1, מעדכנת VWD באופן רציף את חלונית SQL. משפט SQL הסופי אמור כעת לכלול את פסקה WHERE ואת משפט ORDER BY הנוספים.

```
SELECT Id, Name, SortOrder
FROM Genre
WHERE (Id > 4) AND (Name LIKE '%rock%')
ORDER BY SortOrder DESC
```

9. הקש שוב Ctrl+R (או לחץ על הלחצן Execute SQL בסרגל הכלים) והחלונית Results תציג את הרשומות מהטבלה Genre שתואמות את הקריטריון שלך, כמו בתרשים 11-11.

	Id	Name	SortOrder
▶	5	Indie Rock	7
	7	Rock	2
*	NULL	NULL	NULL

תרשים 11-11

שים לב שהרשומות ממוינות עכשיו בסדר יורד על פי האמור בעמודה SortOrder.

כיצד זה פועל

Query Designer שכלול ב-VWD הוא כלי מועיל מאוד ליצירת שאילתות חדשות מתוך בסיס הנתונים. במקום לקודד ידנית את משפט SQL בחלונית SQL, השתמש בחלוניות Diagram ו-Criteria כדי ליצור את השאילתות באופן חזותי. כמובן, תמיד תוכל להשתמש בחלונית SQL כדי לשנות באופן ידני את קוד SQL שנוצר עבורך על ידי VWD.

השאילתה האחרונה שביצעת החזירה את כל הרשומות שהכילו את המילה rock ושהיו בעלות ID גדול מ-4. השאילתה שהוצגה בצעד 9 הייתה בעלת פסקת WHERE שהורכבה משני חלקים: החלק הראשון הגביל את הרשומות שהוחזרו לאלו שהן בעלות ID גדול מ-4; החלק השני סינן את הרשומות לאלו שהכילו את הטקסט rock. שני הקריטריונים מוחלים בו-זמנית תוך שימוש במילת המפתח AND, כדי שתוחזרנה רק רשומות שהן גם בעלות ID גדול מ-4 וגם המילה rock מופיעה בשמן. למעשה מוחזרים הסגנונות Indie Rock ו-Rock. הסגנון Hard Rock לא נכלל, מכיוון שהיה בעל ID של 4, למרות שהמילה Rock כלולה בו (זכור - התנאי הוא "וגם").

בסופו של דבר, קבוצת התוצאות ממוינת בעמודה SortOrder בסדר יורד, על פי התחביר ORDER BY SortOrder DESC. שים לב ש-SortOrder הוא שם שנבחר באופן שרירותי. אפשר בקלות לקבוע לעמודה זו שם אחר, או למיין לפי עמודה אחרת, כמו העמודה Name, כדי לאחזר את רשומות הסגנון בסדר אלפביתי.

בדוגמה זו ראית כיצד לאחזר נתונים מטבלה יחידה. עם זאת, ברוב היישומים בעולם הממשי מתקבלים הנתונים מטבלאות רבות שקשורות זו לזו באופן כלשהו. הקשרים האלה מוגדרים בתחביר SQL תוך שימוש במילת המפתח JOIN.

צירוף נתונים

שימוש ב-JOIN בשאילתה מאפשר לבטא קשרים בין טבלה אחת או יותר. לדוגמה, אפשר להשתמש ב-JOIN כדי למצוא בטבלה Review את כל הסקירות/ביקורות שמתייחסות לסגנון מסוים, ואז לבחור בכמה עמודות מהטבלה Review יחד עם Name של הסגנון.

התחביר הבסיסי של JOIN נראה כמו השורה המוצללת בקוד הבא:

```
SELECT
    SomeColumn
FROM
    LeftTable
INNER JOIN RightTable ON LeftTable.SomeColumn = RightTable.SomeColumn
```

החלק הראשון הוא החלק הרגיל של SELECT בשאלתה שראית קודם, בעוד החלק השני מציג את מילות המפתח INNER JOIN כדי לבטא את הקשר בין שתי הטבלאות. שאלתה זו מחזירה מהטבלה LeftTable רק את הרשומות שיש להן רשומה מתאימה בטבלה RightTable. לדוגמה, כדי לקבל את ה-ID ואת ה-Title של סקירה יחד עם שם הסגנון שהיא שייכת אליו, השתמש במשפט SQL הבא:

```
SELECT
    Review.Id, Review.Title, Genre.Name
FROM
    Review
INNER JOIN Genre ON Review.GenreId = Genre.Id
```

שים לב שבמשפט SELECT יש לכל עמודה תחילית המציינת את שם הטבלה. הדבר מבהיר לאיזו טבלה שייכת העמודה, וגם מונע התנגשויות כאשר בטבלאות שונות יש עמודות בעלות שמות זהים (למשל, העמודה ID שנמצאת בשתי הטבלאות).

בנוסף למשפט הצירוף INNER JOIN שמחזיר רשומות תואמות בלבד, אפשר להשתמש גם במשפט הצירוף OUTER JOIN. הצירוף OUTER JOIN מאפשר לאחזר רשומות מטבלה אחת מבלי להתחשב בכך שיש או אין להן רשומות תואמות בטבלה אחרת. הדוגמה הבאה מחזירה רשימה של כל הסגנונות במערכת יחד עם הסקירות של כל אחד מהם.

```
SELECT
    Genre.Id, Genre.Name, Review.Title
FROM
    Genre
LEFT OUTER JOIN Review ON Genre.Id = Review.GenreId
```

עבור כל סקירה שמוקצית לסגנון מוחזרת שורה ייחודית שמכילה את כותרת הסקירה. עם זאת, גם אם לא הוקצו סקירות לסגנון מסוים, השורה עדיין מתקבלת ומוצגת.

SELECT Genre.Id, Genre.Name, Review.Title FROM Genre LEFT OUTER JOIN Review ON Genre.Id = Review.GenreId			
	Id	Name	Title
	7	Rock	Eyes Open by Snow Patrol - As beautiful as before?
	7	Rock	Muse - Black Holes and Revelations
	8	Grunge	Nirvana - With the lights out
	9	Blues	Amy Winehouse - Back to Black
	10	Reggae	NULL
	11	Industrial	NULL
	12	Techno	DJ Tiesto - In Search of Sunrise 6

תרשים 11-12

אפשר לראות בתרשים 11-12 שהסגנון Rock חוזר כמה פעמים, כמספר הסקירות בטבלה Review שהוקצו לסגנון זה. לסגנונות כמו Grunge ו-Blues הוקצתה סקירה אחת בלבד לכל אחד, ולכן הם מופיעים רק פעם אחת ברשימה. לבסוף, אף סקירה לא הוקצתה לסגנונות Reggae ו-Industrial. עם זאת, מכיוון שמשפט SQL השתמש ב-LEFT OUTER JOIN, שני סגנונות אלה (רשומים בטבלה

משמאל למילה JOIN) עדיין מוחזרים. במקום הכותרת של סקירה מכילה עמודה זו כעת את הערך NULL כדי לציין שלא נמצאה אף סקירה קשורה.

בנוסף למשפט הצירוף LEFT OUTER JOIN, יש גם משפט צירוף RIGHT OUTER JOIN שמחזיר את כל הרשומות מהטבלה שרשומה בצד הימני של ה-JOIN. משפטי הצירוף LEFT OUTER JOIN ו-RIGHT OUTER JOIN דומים מאוד, וברוב המקרים תראה את LEFT OUTER JOIN.

בנוסף, קיימים צירופים אחרים כולל צירופים צולבים וצירופים עצמיים. תוכל למצוא ברשת תיאור מפורט של סוגי צירופים אלה.

בתרגיל נסה זאת הבא תראה כיצד להשתמש בצירוף נפוץ מאוד, ה- INNER JOIN.

נסה זאת צירוף נתונים

כדי לצרף נתונים משתי טבלאות, יש לכתוב בקוד את המשפט JOIN. כדי לסייע בכתיבת הקוד VWD מנסה להוסיף JOIN בכל פעם שאתה מוסיף טבלה לחלונית Diagram. עם זאת, ברוב המקרים JOIN זה אינו נכון, ותצטרך לשנות זאת באופן ידני.

1. בחלון סייר בסיס הנתונים או סייר השרת, לחץ לחיצה ימנית על הטבלה Review ובחר Show Table Data. תראה שהסקירות בטבלה מוצגות. אפשר את החלונות Criteria, Diagram ו-SQL על ידי לחיצה על הלחצנים המתאימים שלהן בסרגל הכלים של Query Designer.

2. לחץ לחיצה ימנית בנקודה ריקה של החלונית Diagram בסמוך לטבלה Review, ובחר Add Table. לחלופין, בחר Add Table ← Query Designer מהתפריט הראשי.

3. בתיבת הדו-שיח שמופיעה לחץ על הטבלה Genre, ואז לחץ על הלחצן Add. לבסוף לחץ Close.

4. התבונן במשפט SQL שהוסיפה VWD. לא נראה ל-VWD הקשר שבין העמודה GenreId של הטבלה Review לבין העמודה Id של הטבלה Genre, ולכן במקום זאת היא צרפה את שתי הטבלאות לפי שדות Id השייכים להן, בהתאמה.

```
SELECT * FROM Review
INNER JOIN Genre ON Review.Id = Genre.Id
```

5. כדי לתקן שגיאה זו, לחץ לחיצה ימנית על הקישור בין שתי הטבלאות ובחר Remove.

6. בחלונית Diagram לחץ פעם אחת על העמודה GenreId של הטבלה Review, וגרור אותה אל העמודה Id של הטבלה Genre. ברגע שתשחרר את העכבר VWD תיצור משפט JOIN חדש בחלונית SQL:

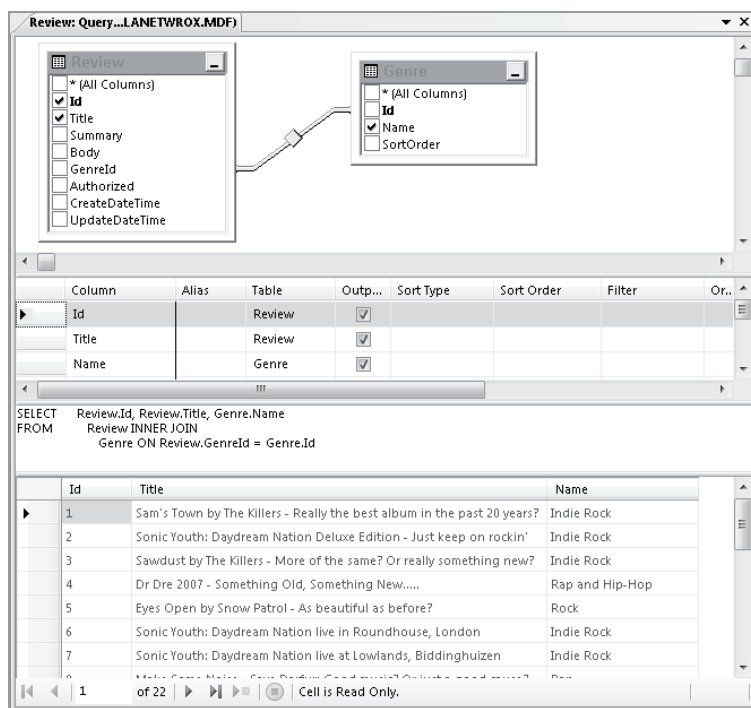
```
SELECT * FROM Review
INNER JOIN Genre ON Review.GenreId = Genre.Id
```

7. בחלונית Criteria, לחץ על השוליים השמאליים של השורה הראשונה שמכילה את הסמל כוכבית (*) כדי לבחור בשורה במלואה, ואז הקש על המקש Delete או לחץ

לחיצה ימנית בשוליים השמאליים ובחר Delete. פעולה זו מסירה את הכוכבית ממשפט SQL. לחילופין, מחק את הכוכבית ישירות מחלונית SQL.

8. בחלונית Diagram, הצב סימן אישור בחזית העמודות Id ו-Title של הטבלה Review, ובחזית העמודה Name של הטבלה Genre.

9. לבסוף, הקש Ctrl+R כדי לבצע את השאילתה. חלון המסמך אמור להיראות כעת בדומה לתרשים 11-13, כשהוא מציג את תוצאות השאילתה בחלונית Results שבתחתית המסך.



תרשים 11-13

כיצד זה פועל

הצירוף JOIN במשפט ה-SQL אומר לבסיס הנתונים איך לקשר את הרשומות זו לזו. בדוגמה הקודמת יצרת קשר בין העמודה GenreId של הטבלה Review וה-Id הממשי של הטבלה Genre.

```
SELECT
    Review.Id, Review.Title, Genre.Name
FROM
    Review
INNER JOIN Genre ON Review.GenreId = Genre.Id
```

עם צירוף JOIN זה אתה יכול לשלוף נתונים מטבלאות רבות ולהציגם בקבוצת תוצאות אחת. מעבד השאילתה בשרת SQL מחזיר עבור כל סקירה/ביקורת את שם הסגנון הנכון, כפי שמוצג בתרשים 11-13.

בחירת הנתונים אינה הפעולה היחידה שתצצה לעשות בבסיס הנתונים שלך. אתה זקוק גם ליכולת להכניס נתונים חדשים לבסיס הנתונים וכשצריך – גם יכולת לעדכן אותם. הוספת נתונים מתבצעת על ידי המשפט INSERT.

יצירת נתונים

כדי להוסיף רשומות לטבלה של SQL Server השתמש במשפט **INSERT**. המשפט יכול להיכתב בכמה דרכים. בצורתו הפשוטה ביותר הוא נראה כך:

```
INSERT INTO TableName (Column1 [, Column2]) VALUES (Value1 [, Value2])
```

ערכי מחרוזת ותאריך צריכים להיות עטופים בזוג גרשים יחידים, בדיוק כמו במקרה של הפסקה WHERE. מספרים וערכים בוליאניים ניתן לכתוב במשפט SQL ללא צורך בגרשים. שורת הקוד הבאה מראה כיצד להכניס שורה חדשה לטבלה Genre:

```
INSERT INTO Genre (Name, SortOrder) VALUES ('Tribal House', 20)
```

לאחר יצירת נתונים אתה עשוי לרצות לערוך אותם. פעולה זו מתבצעת על ידי המשפט UPDATE.

עדכון נתונים

כדי לעדכן נתונים בטבלה השתמש במשפט **UPDATE**.

```
UPDATE TableName SET Column1 = NewValue1 [, Column2 = NewValue2] WHERE Column3 = Value3
```

במשפט UPDATE אנחנו משתמשים במבנים Column = Value כדי לציין את הערך החדש של העמודה שצוינה. לכל עמודה בטבלה יכול להיות מבנה אחד כזה לכל היותר, אבל מספר המבנים אינו מוגבל. אם ברצונך להגביל את מספר הרשומות שיעודכנו, השתמש בפסקה WHERE, בדיוק כמו במקרה של בחירת נתונים שראית לעיל.

הדוגמה הבאה מעדכנת את הרשומה שהוכנסה עם משפט INSERT לעיל. היא קובעת את הערך Trance לעמודה Name ואת הערך 5 לעמודה SortOrder. הדוגמה מגבילה את העדכון לרשומה אחת בפסקה WHERE של הערך 13 בעמודת ID.

```
UPDATE Genre SET Name = 'Trance', SortOrder = 5 WHERE Id = 13
```

בשלב מסוים של העבודה על בסיס הנתונים עלול להתעורר צורך למחוק רשומות קיימות. אין זו הפתעה ש-SQL משתמשת למטרה זו במשפט DELETE.

מחיקת נתונים

גם במשפט **DELETE**, כמו במשפטי SELECT ו-UPDATE, באפשרותך להשתמש בפסקה WHERE כדי להגביל את מספר הרשומות שברצונך למחוק. פסקה WHERE זו היא בעלת חשיבות רבה, משום שבלעדיה אתה עלול למחוק לחלוטין את הטבלה כולה, במקום למחוק רשימות ספורות בלבד.

במשפט DELETE אין צורך לציין שמות עמודות. עליך רק לציין את שם הטבלה שממנה ברצונך למחוק עמודות, כדאי מאוד גם לציין בפסקת WHERE תנאי שיגביל את פעולת המחיקה

לרשומות שיימחקו. הדוגמה הבאה מוחקת את הרשומה שהוכנסה ועודכנה בשתי הדוגמאות הקודמות.

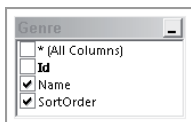
```
DELETE FROM Genre WHERE Id = 13
```

בתרגיל הבא תראה את משפטי SQL אלה בפעולה.

נסה זאת עבודה עם נתונים בבסיס הנתונים לדוגמה

בתרגיל זה תתרגל את כל מה שלמדת עד כה. בסדרה של צעדים פשוטים תראה כיצד ליצור רשומה חדשה בטבלה Genre, לבחור בה שוב, כדי למצוא את ID החדש שלה, לעדכן אותה תוך שימוש במשפט UPDATE, ולבסוף למחוק את רשומת הסגנון מבסיס הנתונים. למרות שהדוגמאות עצמן פשוטות למדי, הן הליכה של דרך פעולת SQL. אם תבין את הדוגמאות שבסעיף זה, תוכל להפעיל את שאר משפטי SQL, בפרק זה ובפרקים שיבואו.

1. פתח את חלון סייר בסיס הנתונים באתר הניסוי הזמני שלך, ואתר את הטבלה Genre בבסיס הנתונים. לחץ עליה לחיצה ימנית ובחר Show Table Data. אם הטבלה כבר הייתה פתוחה, עליך לסגור אותה תחילה על ידי הקשת Ctrl+F4. בפעולה זו תיפטר ממשפט SQL של השאילתה הקודמת.
2. לחץ על שלושת הלחצנים הראשונים בסרגל הכלים של (Diagram) Query Designer, Criteria וחלונית (SQL) כדי לפתוח את החלוניות המתאימות.
3. בחלונית Diagram, סמן את העמודות Name ו-SortOrder. ודא שהעמודה Id נשארה בלתי מסומנת, כפי שמוצג בתרשים 11-14.



תרשים 11-14

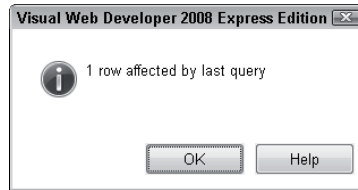
4. בסרגל הכלים Query Designer, לחץ על הלחצן Change Type ואז בחר באפשרות השלישית, Insert Values. השאילתה בחלונית SQL מתעדכנת, ומכילה כעת תבנית עבור משפט INSERT.

```
INSERT INTO Genre (Name, SortOrder) VALUES (,)
```

5. בתוך הסוגריים של VALUES הכנס שם (בין גרשים) עבור עמודה Name ומספר עבור עמודה SortOrder, כשהם מופרדים על ידי פסיק:

```
VALUES ('Folk', 15)
```

6. הקש Ctrl+R כדי לבצע את השאילתה. אתה אמור לקבל תיבת דו-שיח שתאמר שפעולתך השפיעה על שורה אחת, כפי שמוצג בתרשים 11-15.



תרשים 11-15

7. לחץ OK כדי לשחרר את תיבת הדו-שיח.
8. בחלונית SQL נקה את משפט SQL כולו (למשל, בחר במשפט SQL כולו על ידי הקשת Ctrl+A ואז הקש על המקש Delete כדי למחוק אותו) והחלף אותו עם הקוד הבא ששולף את כל הרשומות בטבלה בסדר יורד.

```
SELECT Id, Name FROM Genre ORDER BY Id DESC
```

9. הקש Ctrl+R כדי לבצע את משפט SELECT זה. החלונית Results תציג רשימה מהטבלה כאשר הרשומה שהכנסת זה עתה מוצגת בראש הרשימה. שים לב ל-ID של הרשומה החדשה. הוא צריך להיות 13, אם לא הכנסת רשומה אחרת לפניו, אם כי זה בסדר גם אם ה-ID שונה.

10. לחץ פעם נוספת על הלחצן Change Type שבסרגל הכלים, והפעם בחר UPDATE. השלם את משפט SQL שיצרה עבורך VWD באופן שייראה דומה לזה (שים לב: אפשר לכתוב את המשפט בשורה אחת, או לפצל למספר שורות, כפי שנוח):

```
UPDATE
  Genre
SET
  Name = 'British Folk',
  SortOrder = 5
WHERE
  Id = 13
```

11. אל תשכח להחליף את המספר 13 במשפט SQL בערך ה-ID שקיבלת על ידי הכנסת הרשומה בעת בצעד 9. הקש שוב Ctrl+R כדי לבצע את השאילתה. כעת תופיע תיבת דו-שיח שתודיע כי רשומה אחת שונתה.
12. נקה את חלונית SQL פעם נוספת, הכנס את השאילתה הבאה, ובצע אותה על ידי הקשת Ctrl+R:

```
SELECT Id, Name FROM Genre WHERE Id = 13
```

החלף את קוד הזיהוי בפסקה WHERE ב-ID של הרשומה שקבעת בצעד 9. הרשומה המעודכנת אמורה להופיע.

13. בסרגל הכלים של Query Designer, לחץ על הלחצן Change Type ובחר DELETE. כעת VWD משנה את משפט SQL באופן שהוא מוכן למחיקת הרשומה שערך ID שלה הוא 13.

```
DELETE FROM Genre WHERE (Id = 13)
```

14. הקש Ctrl+R כדי לבצע את השאילתה ולמחוק את הרשומה מבסיס הנתונים.
15. כדי לוודא שהרשומה אכן נמחקה, לחץ על Change Type פעם נוספת ובחר Select. בחלונית Diagram, בחר עמודה אחת או יותר מהטבלה Genre והקש שוב Ctrl+R. כעת תראה שהפעם לא מוחזרת רשומה כלשהי, והדבר מאשר שהרשומה החדשה שהוכנסה אכן נמחקה מבסיס הנתונים.

כיצד זה פועל

בתרגיל קצר זה ביצעת את ארבע פעולות היסוד מול בסיס נתונים: יצירה של רשומה, קריאה, עדכון ומחיקה. סדרת פעולות זו גם מבטאת את מחזור החיים של נתונים בבסיס נתונים מסוג SQL Server.

התחלת במשפט INSERT:

```
INSERT INTO Genre (Name, SortOrder) VALUES ('Folk', 15)
```

משפט זה יצר רשומה חדשה בטבלה Genre. כפי שתראה בסעיף הבא, העמודה Id של הטבלה Genre היא עמודת זהות (identity column) שפירושה שלכל רשומה חדשה מוקצה באופן אוטומטי ID חדש, עוקב.

כדי לשלוף את הרשומה לפי ID זה, השתמשת במשפט SELECT עם פסקה ORDER BY שמסדרת את כל ערכי ID בסדר יורד, באופן שה-ID החדש ביותר מוצב בראש הרשימה. ביישום בעל פעילות מרובה, שליפת ה-ID החדש בהסתמך על מקומו בראש הרשימה אינו צעד אמין, ואתה עלול להגיע אל ID של רשומה אחרת. בהמשך הספר תלמד כיצד לשלוף את מספרי הזיהוי בדרך אמינה, אבל למטרות תרגיל זה, השיטה ORDER BY פועלת כראוי.

מצויד ב-ID החדש ביצעת משפט UPDATE לשינוי Name ו-SortOrder של הרשומה החדשה שהוכנסה. אם ברצונך לעדכן עמודה אחת בלבד בעזרת משפט UPDATE – נניח שברצונך לשנות רק את Name – אתה יכול להשמיט את שמות העמודות האחרות. לדוגמה, משפט UPDATE הבא משנה רק את Name, ומשאיר את כל העמודות האחרות עם הערכים המקוריים שלהן.

```
UPDATE Genre  
SET Name = 'British Folk'  
WHERE Id = 13
```

לבסוף, בסיום התרגיל, ביצעת משפט DELETE כדי להיפטר מהרשומה החדשה. כשמבצעים משפטי DELETE או UPDATE חשוב תמיד לציין פסקה WHERE, כדי למנוע מחיקה מקרית של הטבלה כולה, או שינוי השמות של כל הרשומות לאותו ערך.

```
DELETE FROM Genre WHERE (Id = 13)
```

משפט SQL זה מוחק את הרשומה בעלת ערך 13 בעמודת ID. אם הרשומה קיימת, היא נמחקת. אם הרשומה אינה קיימת, הדבר אינו נחשב לשגיאה, אבל תיבת הדו-שיח ב-VWD מראה שאף רשומה לא הושפעה.

עד לנקודה זו ראית כיצד לעבוד עם טבלאות קיימות בבסיס נתונים. עם זאת, חשוב גם להבין כיצד ליצור טבלאות חדשות בעלות קשרים. נדון בכך בסעיף הבא.

יצירת טבלאות משלך

יצירת טבלאות בבסיס נתונים מסוג SQL Server 2005 (או 2008) היא פעולה קלה, ומתבצעת תוך שימוש בכלי בסיס נתונים שמובנים ב-VWD. בסעיף הבא תערוך היכרות קצרה עם סוגי הנתונים שעומדים לרשותך בתוכנה SQL Server 2005 ובגרסאות מתקדמות יותר. תלמד גם כיצד ליצור טבלאות משלך בבסיס הנתונים.

סוגי נתונים במערכת SQL Server

כמו שפות תכנות דוגמת Visual Basic ו-C#, גם בסיס הנתונים מסוג SQL Server משתמש בסוגי נתונים שונים כדי לאחסן את הנתונים שלו. התוכנה SQL Server 2005 תומכת בכ-30 סוגי נתונים שונים, רובם נראים דומים לסוגים שבהם משתמשת .NET. הטבלה הבאה מפרטת את סוגי הנתונים הנפוצים ביותר של שרת SQL ומציגה את תיאורם וציון מקביליהם ב-.NET.

SQL 2005 Data Type	תיאור הייעוד	.NET Data Type
bit	אחסון ערכים בוליאניים בפורמט 0/1.	System.Boolean
char / nchar	טקסט בעל אורך קבוע. כשתאחסן טקסט קצר מהאורך שנקבע, הוא ירופד ברווחים. סוג הנתונים nchar משתמש בפורמט Unicode, שמאפשר לאחסן נתונים בשפות זרות רבות.	System.String
datetime	אחסון תאריך ושעה.	System.DateTime
decimal	אחסון מספרים: שברים גדולים.	System.Decimal
float	אחסון מספרים: שברים גדולים.	System.Double
image	אחסון אובייקטים בינאריים גדולים כמו קבצים. מהשם עלול להשתמע שניתן להשתמש בסוג זה כדי לאחסן תמונות בלבד, אבל זה אינו המצב. אפשר להשתמש בסוג זה גם לאחסון כל סוג של מסמך או אובייקט בינארי אחר.	System.Byte[]
tinyint	אחסון מספרים שלמים בטווח 0 עד 255.	System.Byte
smallint	אחסון מספרים שלמים בטווח -32,768 עד 32,767.	System.Int16

SQL 2005 Data Type	תיאור הייעוד	.NET Data Type
int	אחסון מספרים שלמים בטווח -2,147,483,648 עד 2,147,483,647.	System.Int32
bigint	אחסון מספרים שלמים בטווח -9,223,372,036,854,775,808 עד 9,223,372,036,854,775,807.	System.Int64
text / ntext	אחסון כמויות גדולות של טקסט.	System.String
varchar / nvarchar	אחסון טקסט בעל אורך משתנה. סוג הנתונים nvarchar מאחסן את הנתונים בפורמט Unicode שמאפשר לאחסן נתונים בשפות זרות רבות.	System.String
uniqueidentifier	אחסון מזהים בעלי ייחודיות גלובלית.	System.Guid

כמה מסוגי נתונים אלה מאפשרים לציין את האורך המרבי שלהם. כשתגדיר עמודה מסוג char, varchar, nvarchar או nchar, עליך לציין את מספר התווים. לדוגמה, nvarchar(10) מאפשר לאחסן לכל היותר 10 תווים. החל מ-SQL Server 2005 מאפשרים סוגי נתונים אלה גם לציין MAX כגודל המרבי. עם MAX אתה יכול לאחסן בעמודה אחת נתונים עד 2GB תווים. עבור קטעי טקסט גדולים, מומלץ להשתמש בסוג הנתונים nvarchar(max). אם יש לך ידיעה ברורה לגבי האורך המרבי עבור עמודה (כמו מספר המיקוד או מספר טלפון) או שברצונך להגביל את האורך של העמודה במפורש, עליך לציין את האורך. לדוגמה, שם העובד יכול להיות מאוחסן בעמודה שמסומנת על ידי nvarchar(10) כדי לאפשר עד 10 תווים.

הבנת מפתחות ראשיים וזהויות

כדי לזהות רשומה בטבלה באופן ייחודי, באפשרותך להגדיר **מפתח ראשי** (primary key). מפתח ראשי מורכב מערכים שנמצאים בעמודה אחת או יותר בטבלה. עושים זאת כאשר צריך לקבל ערך ייחודי על פני כל הרשומות. כשאתה מזהה עמודה כמפתח ראשי, מנוע בסיס הנתונים יבטיח שלא תהיינה שתי רשומות בעלות אותו ערך עצמו. מפתח ראשי יכול להיות מורכב מעמודה אחת בלבד (לדוגמה, עמודה ספרתית בשם Id שמכילה מספרים ייחודיים עבור כל רשומה בטבלה), או שהוא יכול להתפרש על פני עמודות רבות, כשכל העמודות יחד מהוות את ה-ID הייחודי של הרשומה כולה (למשל שם רחוב ומספר בית כמזהה ייחודי).

שרת SQL תומך גם **בעמודות זהות** (identity columns). עמודות זהות היא עמודה ספרתית שערכיה נוצרים באופן אוטומטי בכל פעם שמתווספת רשומה חדשה. ערכים אלה משמשים לעתים קרובות כמפתח ראשי עבור טבלה. בסעיף הבא, כשתיצור טבלאות משלך, תראה כיצד זה פועל.

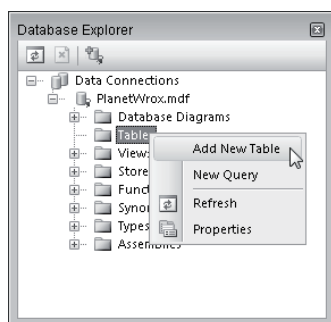
קביעת מפתח ראשי לטבלה אינה חובה, אבל הדבר יכול להקל מאוד על חיך כמתכנת בסיס נתונים. לפיכך מומלץ תמיד להוסיף מפתח ראשי לטבלאות.

יצירת מפתחות ראשיים ועמודות זהות היא פעולה קלה, הודות לכלי בסיס הנתונים הכלולים ב-VWD. תראה זאת בתרגיל נסה זאת הבא.

נסה זאת יצירת טבלאות על ידי Table Designer

בתרגיל זה תוסיף בסיס נתונים חדש לפרויקט Planet Wrox, ואליו תוסיף שתי טבלאות. בתרגיל מאוחר יותר תלמד כיצד ליצור קשרים בין שתי הטבלאות בבסיס הנתונים. שים לב שעליך לבצע את התרגילים באתר של Planet Wrox שבנית עד כה. אתה יכול לסגור ולמחוק את אתר הניסוי שבנית בתחילת פרק זה, מכיוון שלא תזדקק לו יותר.

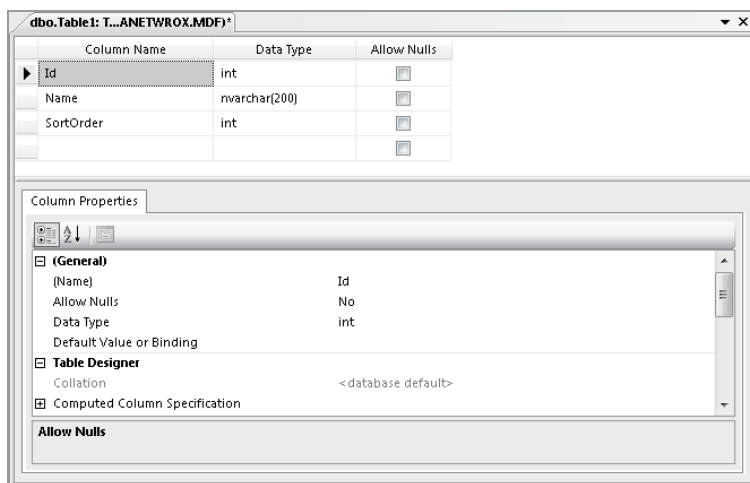
1. לחץ לחיצה ימנית על התיקיה App_Data ובחר Add New Item. בתיבת הדו-שיח שמופיעה לחץ SQL Server Database, הקלד PlanetWrox.mdf עבור השם, ולחץ Add כדי להוסיף את בסיס הנתונים לאתר שלך. סייר בסיס הנתונים (או סייר השרת) אמור להיפתח באופן אוטומטי ולהציג את בסיס הנתונים החדש. אם הדבר אינו קורה, לחץ לחיצה כפולה על PlanetWrox.mdf בסייר הפתרון.
2. בסייר בסיס הנתונים, לחץ לחיצה ימנית על הצומת Tables, ובחר Add New Table, כפי שמוצג בתרשים 11-16.



תרשים 11-16

3. בתיבת הדו-שיח שמופיעה אתה יכול להכניס שמות עמודות וסוגי נתונים שמהווים יחד את הגדרת הטבלה. צור שלוש עמודות עבור Name, Id, ו-SortOrder של הטבלה Genre. תיבת הדו-שיח תראה כפי שמוצג בתרשים 11-17.

ודא שבעמודה Allow Nulls ניקית את תיבות הסימון של כל שלושת הפריטים. עמודה זו קובעת אם שדות הם חובה או רשות. במקרה של הטבלה Genre, כל שלוש העמודות הן חובה, ולכן עליך לנקות את העמודה Allow Nulls.



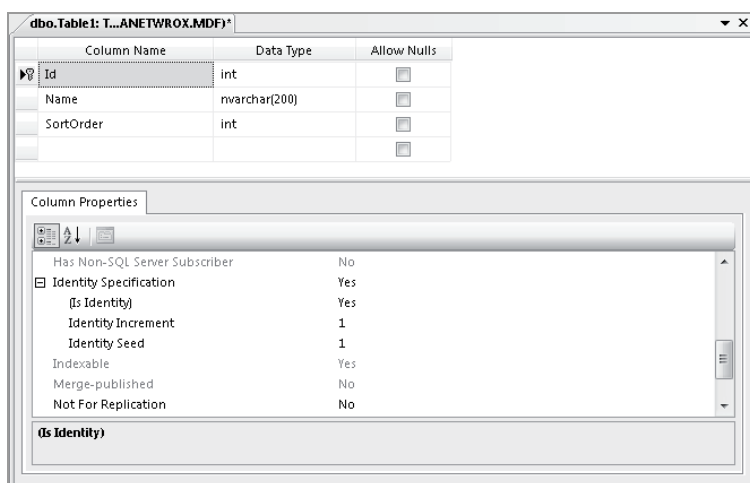
תרשים 11-17

4. בחר בכל השורה בעלת ה-Id על ידי לחיצה בשוליים שמשמאל (מזווה בתרשים 11-17 על ידי החץ השחור). כדי להפוך את העמודה Id למפתח ראשי, לחץ על הלחצן השני שמאל (זה שעליו משורטט מפתח צהוב) מתוך סרגל הכלים Table Designer, שמוצג בתרשים 11-18.



תרשים 11-18

5. מתחת להגדרת הטבלה תראה את מאפייני העמודה (Column Properties), בלוח שנראה דומה לטבלת המאפיינים של VWD. כשהעמודה Id עדיין מסומנת, גלול מעט מטה בטבלת מאפייני העמודה עד שתראה את Identity Specification. הרחב את הפריט והגדר את (Is Identity) ל-Yes, כפי שמוצג בתרשים 11-19.

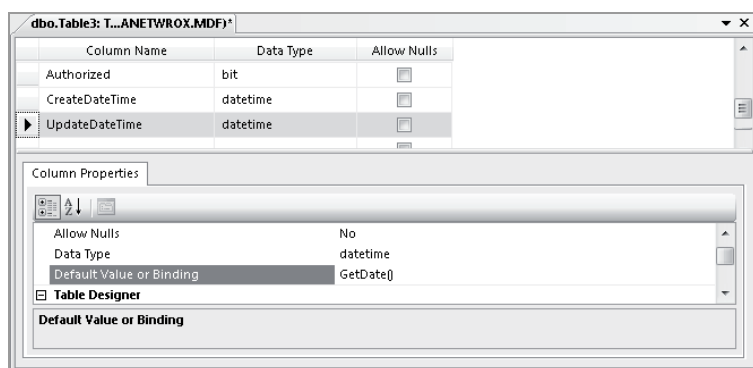


תרשים 11-19

6. הקש Ctrl+S כדי לשמור על השינויים. כעת תופיע תיבת דו-שיח שתאפשר לקבוע שם טבלה. הקלד Genre ולחץ OK כדי להחיל את השינויים.
7. צור טבלה נוספת על ידי חזרה על צעדים 2 עד 6, אלא שהפעם צור טבלה בעלת המפרט הבא, שתכיל את סקירות התקליטורים והקונצרטים של אתר האינטרנט Planet Wrox.

שם עמודה	סוג עמודה	רשות / חובה	תיאור
Id	int	חובה	המפתח הראשי והזהות של הטבלה.
Title	nvarchar(200)	חובה	כותרת הסקירה.
Summary	nvarchar(max)	חובה	תיאור קצר או טקסט טיזר (teaser) של הסקירה.
Body	nvarchar(max)	רשות	גוף הסקירה.
GenreId	int	חובה	ID של הסגנון שהסקירה מתייחסת אליו.
Authorized	bit	חובה	קביעה אם הסקירה מותרת לפרסום לציבור. האתר אינו מאפשר לראות סקירות שאין להן הרשאה כזו.
CreateDateTime	datetime	חובה	התאריך והשעה של יצירת הסקירה.
UpdateDateTime	datetime	חובה	התאריך והשעה של העדכון האחרון של הסקירה.

8. קבע שוב את העמודה Id כמפתח הראשי, והגדר את המאפיין (Is Identity) שלה ל-Yes.
9. לחץ על העמודה CreateDateTime פעם אחת, ואז הקלד GetDate() בטבלת מאפייני העמודה, בשדה המאפיין Default Value or Binding. התוצאה מוצגת בתרשים 11-20.



תרשים 11-20

10. חזור על הצעד הקודם עבור העמודה UpdateDateTime.

11. כשתסיים, הקש Ctrl+S כדי לשמור על הטבלה בשם Review.

כיצד זה פועל

רכיב Table Designer ב-VWD הוא ברור ומובן למדי. אתה מקליד שמות עמודות חדשות ומגדיר את סוג הנתונים לעמודה, וכמעט סיימת. כמה עמודות, כמו העמודה Id בטבלאות Genre ו-Review, דורשות מעט יותר עבודה. עבור עמודות אלו הגדר את Identity לערך Yes. פירוש הדבר ש-SQL Server יקצה באופן אוטומטי מספר עוקב חדש לכל רשומה חדשה שתוסיף לבסיס הנתונים. כברירת מחדל, הרשומה הראשונה בטבלה מקבלת ID שערכו 1, הרשומות העוקבות תקבלנה ID שגדל ב-1 בכל פעם. באפשרותך לשנות את התנהגות ברירת המחדל על ידי שינוי ההגדרה Identity Increment וההגדרה Identity Seed באלמנט Identity Specification של הרשומה.

גם לעמודות CreateDateTime ו-UpdateDateTime הקצית ערך ברירת מחדל. כשאינך מספק ערך מפורש, בסיס הנתונים מכניס את ערכי ברירות המחדל. פירוש הדבר שאם המשפט INSERT אינו מכיל ערך עבור העמודות CreateDateTime או CreateDateTime, בסיס הנתונים יכניס עבורן ערך ברירת מחדל באופן אוטומטי. בתרגיל נסה זאת הקודם היה ערך ברירת המחדל getDate(), זה שמכניס באופן אוטומטי את התאריך והשעה הנוכחיים בעת ההכנסה.

אם אינך בטוח שעקבת אחר כל הצעדים לעיל בצורה נכונה, התבונן בבסיס הנתונים המקורי שכלול בקוד המקור של פרק זה. הוא מכיל את הטבלאות הנכונות, שכבר מכילות בתוכן כמה רשומות. כדי להתבונן בבסיס הנתונים, צור ב-VWD אתר אינטרנט זמני חדש, וגרור את קבצי בסיס הנתונים לתיקייה App_Data. תוכל אז לגשת לבסיס הנתונים תוך שימוש בסייר בסיס הנתונים.

בנוסף לקשרים שמוגדרים בשאילתות SQL בלבד, כפי שראית קודם במשפטי SELECT ו-JOIN, ביכולתך ליצור קשרים גם בבסיס הנתונים. בסעיף הבא נדון בתועלות שבקשרים ובדרך יצירתם בבסיס הנתונים.

יצירת קשרים בין טבלאות

חשוב על הטבלאות שיצרת עד כה. יצרת את הטבלה Genre בעלת העמודה Id שתפקידה לזהות באופן ייחודי כל רשומת סגנון. יצרת גם את הטבלה Review בעלת העמודה GenreId. ברור שעמודה זו צריכה להכיל Id שמצביע על רשומה בטבלה Genre כדי שתדע לאיזה סגנון שייכת הסקירה. כעת תוכל להניח שמחמת מהטבלה Genre רשומה שקשורות אליה סקירות. אם לא קיימים קשרים בין שתי הטבלאות, בסיס הנתונים יאפשר את פעולת המחיקה, אולם הדבר יגרום ללא מעט בעיות. למשל, אם תנסה כעת להציג את הסגנון יחד עם סקירה שלו, הפעולה

תיכשל משום שאין עוד סגנון תואם. באופן דומה, אם תרצה לבנות רשימה של כל הסקירות במערכת כשהן מקובצות לפי סגנונות, לא תוכל להציג את אלו ששייכות לסגנון שנמחק.

כדי למנוע בעיות מסוג זה, וכדי לשמור את בסיס הנתונים במצב בריא ועקבי, באפשרותך ליצור קשרים בין שתי טבלאות. אם הוגדר קשר, בסיס הנתונים ימנע ממך מלמחוק רשומות בטבלה אחת כשטבלה אחרת יש רשומות שקשורות אליהן.

בנוסף להגנה על נתונים, קשרים גם עושים את מודל הנתונים לברור יותר. אם תתבונן בבסיס הנתונים בעזרת תרשים (דבר שתעשה בתרגיל הבא) תמצא שקשרים בין טבלאות תורמים להבנה טובה יותר של החיבור של הטבלאות זו לזו ושל הנתונים שהן מייצגות.

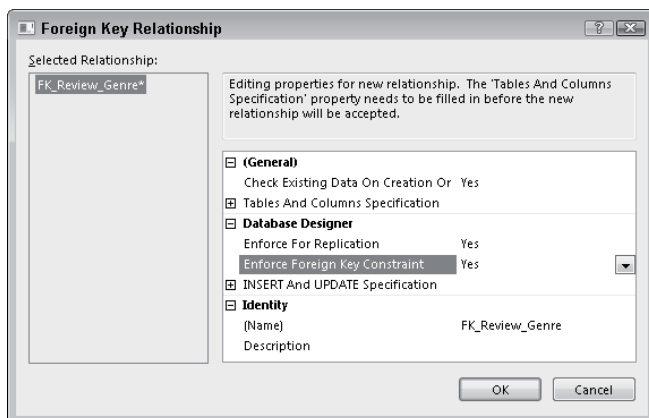
אפשר להגדיר קשר על ידי יצירת קשר בין המפתח הראשי של טבלה אחת לבין עמודה כלשהי בטבלה אחרת. אל העמודה בטבלה השנייה מתייחסים לעתים קרובות כמפתח זר (foreign key). במקרה של הטבלאות Review ו-Genre, העמודה GenreId של הטבלה Review מצביעה על עמודת המפתח הראשי Id של הטבלה Genre. פעולה זו קובעת את GenreId כמפתח זר. בתרגיל הבא תלמד כיצד ליצור קשר בין שתי טבלאות. אחר כך תבצע משפט SQL שמראה כיצד הקשר תומך בהגנה על הנתונים.

נסה זאת יצירת קשר בין שתי טבלאות

לפני שתוכל להוסיף קשר בין שתי טבלאות, עליך להוסיף תרשים לבסיס הנתונים. תרשים הוא כלי חזותי שעוזר להבין ולהגדיר את בסיס הנתונים. בתרשים תוכל לגרור עמודה מטבלה אחת לאחרת, כדי ליצור את הקשר הרצוי. בתרגיל זה תיצור קשר בין הטבלאות Review לבין הטבלה Genre.

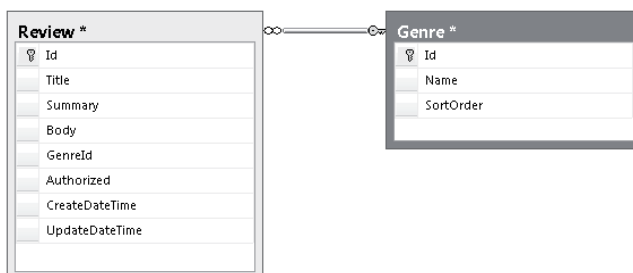
1. פתח את סייר בסיס הנתונים עבור האתר Planet Wrox. לחץ לחיצה ימנית על האלמנט Database Diagrams (נראה בתרשים 11-16) ולחץ Add New Diagram. בפעם הראשונה שמוסיפים תרשים לבסיס הנתונים, עשויה להופיע תיבת דו-שיח שתאפשר לך לשאול אם ברצונך ש-WVD תקבע אותך כבעלים של בסיס הנתונים. לחץ Yes כדי להמשיך. לאחר בקשת הנחיה זו עשויה להופיע הודעה נוספת שתציין שכדי לפעול עם תרשימים WVD צריכה ליצור כמה אובייקטים. לחץ גם הפעם Yes לאישור ולהמשך.
2. בתיבת הדו-שיח Add Table שמופיעה אחר כך בחר בשתי הטבלאות שיצרת בתרגיל נסה זאת הקודם (החזק את מקש Ctrl בזמן שאתה לוחץ על כל אחת מהן), לחץ Add להוספת הטבלאות לתרשים, ואז לחץ Close לשחרור תיבת הדו-שיח Add Table.
3. בשיטת גרור-ושחרר סדר את הטבלאות בתרשים באופן שתוצבנה זו לצד זו.
4. בטבלה Genre, לחץ בשוליים השמאליים של העמודה Id (צריך להיות בעמודה סמל מפתח צהוב לציון העובדה שהיא המפתח הראשי של הטבלה), ואז גרור אותה אל העמודה GenreId של הטבלה Review ושחרר את העכבר.
5. תופענה כעת שתי תיבות דו-שיח שמאפשרות להתאים את ברירות המחדל עבור הקשר. לחץ OK לשחרור החלון העליון. שים לב שבתיבת הדו-שיח שנשארה ומוצגת

בתרשים 11-21, מוגדר Enforce Foreign Key Constraint למצב Yes. מאפיין זה מבטיח שלא תוכל למחוק רשומה מהטבלה Genre אם עדיין קיימות רשומות שמקושרות אליה (בטבלה Review). לחץ OK לשחרור תיבת דו-שיח זו גם כן.



תרשים 11-21

6. חלון התרשים צריך להציג כעת שורה בין שתי הטבלאות. לצד הטבלה Genre אתה אמור לראות מפתח צהוב שמציין שטבלה זו מכילה את המפתח הראשי עבור הקשר. בקצה השני אתה אמור לראות את סימן האינסוף (∞) שמציין כי הטבלה Review עשויה להכיל רשומות רבות שמשמשות באותו GenreId. התרשים מופיע בתרשים 11-22.



תרשים 11-22

7. הקש Ctrl+S כדי לשמור את כל השינויים שערכת בתרשים. אתה יכול להשאיר את שם ברירת המחדל Diagram, או לכתוב שם אחר כרצונך, כגון Reviews and Genres. לחץ OK לאישור. תקבל אזהרה נוספת שאומרת שאתה עומד בפני גרימת שינוי לטבלאות Review ו-Genre. לחץ OK כדי להחיל את השינויים.

8. חזור לסייר בסיס הנתונים, לחץ לחיצה ימנית על הטבלה Genre ובחר Show Table Data. הכנס מספר רשומות לטבלה על ידי הקלדת Name ו-SortOrder. כשאתה בשדה SortOrder הקש על המקש Tab כדי לצאת מהשורה הנוכחית, השורה תוכנס לבסיס הנתונים, ובעמודה Id יופיע מספר עוקב ייחודי. אתה אמור לסיים עם רשימה דומה לזו שמוצגת בתרשים 11-23.

Genre: Query(...LANETWROX.MDF)

	ID	Name	SortOrder
	1	Rock	1
	2	Indie Rock	2
	3	Jazz	3
▶▶	NULL	NULL	NULL

4 of 4 Cell is Read Only.

תרשים 11-23

9. פתח את הטבלה Review והכנס כמה רשומות. עבור GenreId ספק כמה מערכי ID החדשים שקיבלת כשהוספת רשומות לטבלה Genre. בשלב זה מספיק שתמלא את השדות Title, Summary ו-Body ותגדיר את Authorized ל-True. זכור שאינך חייב להכניס ערך לעמודות התאריך. אם תשאיר אותן ריקות, בסיס הנתונים יכניס את ערך ברירת המחדל. שים לב שאינך יכול להכניס בעצמך ערך לעמודה Id. מכיוון שעמודה זו הוגדרה כסוג עמודת זהות חד-ערכי ובסיס הנתונים יספק עבורך ערכים באופן אוטומטי. אם תקבל הודעת שגיאה לגבי ערכים חסרים בעמודות התאריך, ודא שבתרגיל הקודם הכנסת ערך ברירת מחדל הולם. כשתסיים למלא את השדות, לחץ מחוץ לשורה (למשל על השורה הריקה החדשה שמתחת לה) ואז הקש Ctrl+R כדי להכניס את השורה לטבלה. רשימת הרשומות שלך צריכה להיראות דומה לתרשים 11-24, אם כי תוכן העמודות, כמובן, עשוי להיות שונה.

Review: Query(...ANETWROX.MDF)

	ID	Title	Summary	Body	GenreId	Authorized
▶	2	Editors - An End Has A Start	Discusses the new Editors CD	NULL	2	True
	4	Sonic Youth - Daydream Nation Live	A review of the concert that Sonic Youth ga...	NULL	2	True
*	NULL	NULL	NULL	NULL	NULL	NULL

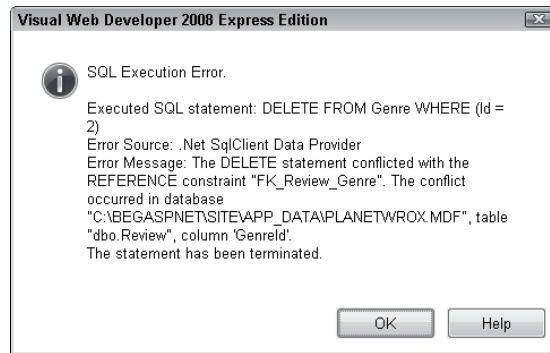
1 of 2 Cell is Read Only.

תרשים 11-24

10. לחץ שוב לחיצה ימנית על הטבלה Genre ובחר Show Table Data. בסרגל הכלים של Query Designer, לחץ על לחצן חלונית SQL, והשתמש בלחצן Change Type באותו סרגל כלים, כדי ליצור שאילתת מחיקה. שנה את השאילתה כדי שתיראה כמו זו:

```
DELETE FROM Genre WHERE Id = 2
```

ודא שה-ID בפסקה WHERE תואם את ה-ID של אחת הרשומות שהשתמשת בהן בצעד 9 לקישור הרשומות אליהן. הקש Ctrl+R כדי לבצע את השאילתה. כעת, במקום למחוק את הרשומה מהטבלה Genre, תציג VWD את תיבת הדו-שיח שמופיעה בתרשים 11-25.



תרשים 11-25

כיצד זה פועל

כשאתה יוצר קשר בין שתי טבלאות, בסיס הנתונים כופה קשר זה בשעה שאתה מנסה להכניס, לעדכן או למחוק נתונים. בדוגמה הקודמת, קיימות רשומות בטבלה Review שיש להן רשומה עם מזהה מקביל בטבלה Genre. כשאתה מנסה למחוק רשומה מהטבלה Genre, בסיס הנתונים רואה שהרשומה נמצאת בשימוש של רשומה מהטבלה Review, ומבטל את פעולת המחיקה.

עצות מעשיות לעבודה עם בסיסי נתונים

להלן כמה עצות מעשיות לעבודה עם בסיסי נתונים:

- ⊙ בסיס הנתונים הוא לעתים קרובות לב אתר האינטרנט. לפיכך עליך להיות זהיר וקפדן כשאתה קובע את עיצובו. במיוחד עליך לחשוב מראש על עיצוב טוב, לפני שאתה מתחיל לבנות את האתר המתבסס עליו. כשיהיו לך כבר כמה דפים שניגשים לבסיס הנתונים, יהיה קשה יותר לערוך שינויים במודל הנתונים, כמו הסרת טבלאות ושינוי שם עמודות.
- ⊙ חשוב תמיד על קביעת מפתח ראשי לטבלה שלך. ההעדפה שלי היא לקבוע בכל טבלה עמודה בשם Id. סוג הנתונים עבורה יהיה int, והזהות קובעת לכל רשומה באופן אוטומטי ID ייחודי. במקום int אפשר לחשוב גם על סוג הנתונים uniqueidentifier, אשר מבטיח ייחודיות גם במקרה שתחרוג מגבול בסיס הנתונים והערך יהיה מזהה ייחודי ברמה עולמית (דוגמה מערכת בינלאומית שאוספת נתונים ממערכות מידע ברחבי העולם – אם תגדיר את סוג העמודה כ- uniqueidentifier תבטיח כי הרשומה תהייה ייחודית לעומת ערך מספרי).
- ⊙ קבע שמות לוגיים לאובייקטים של בסיס הנתונים, כמו טבלאות ועמודות. הימנע מתווים כמו רווחים, קווים תחתונים ומקפים. שם כמו GenreId קל לקריאה הרבה יותר משם כמו colGen_ID_3.

- ⊙ הימנע משימוש בכוכבית במשפטי השליפה כשתרצה לקבל את כל העמודות מבסיס נתונים. על ידי שימוש ב- `SELECT *` אתה עלול לבחור יותר עמודות מכפי שצריך באמת. על ידי הגדרה מפורשת של העמודות שברצונך לאחזר, אתה מבהיר לאחרים את כוונותיך, ובו בזמן משפר את ביצועי השאילתות שלך.
- ⊙ כשהדבר מתאים, צור תמיד קשרים בין טבלאות. אמנם, השאילתות לגבי סקירות וסגנונות שראית בפרק זה פועלות היטב גם ללא קשרים בין שתי הטבלאות, אבל הקשרים הם שעוזרים לכפות את איכות הנתונים. עם קשרים מתאימים, אתה מנטרל סיכונים של נתונים שגויים או "יתומים" (כלומר, שאין להם רשומה מקבילה בטבלה האחרת).

סיכום

היכולת לעבוד עם בסיסי נתונים היא תוספת חשובה לכישורי תכנות האינטרנט שלך. רוב אתרי האינטרנט הדינמיים והמודרניים של ימינו משתמשים בבסיסי נתונים, ולכן חשוב להבין כיצד לעבוד מול בסיסי הנתונים.

בסיס נתונים טבלאי הוא המבנה המקובל והנפוץ ביותר של בסיס נתונים שבשימוש. SQL Server של מיקרוסופט הוא בסיס נתונים טבלאי שבאופן טבעי מתמזג היטב עם סביבת הפיתוח של ASP.NET, והדבר מקל על שליפת הנתונים שמאוחסנים בו ועל פעולה עליהם. מכיוון ש- Visual Web Developer 2008, SQL Server 2005 Express Edition כלול ללא תשלום בתוכנה, ומכיוון שבסיס נתונים זה יכול להיות משודרג בקלות לגרסאות המסחריות המלאות של שרת SQL, הרי שבסיס נתונים זה הוא הבחירה הטבעית עבור אתרי אינטרנט מבוססי ASP.NET 3.5.

כדי לגשת ולפעול על נתונים בבסיס נתונים טבלאי, השתמש בשפה בשם Structured Query Language, או בקיצור SQL. בין שאר האלמנטים, שפה זו מגדירה ארבע מילות מפתח חשובות שמאפשרות לבצע את הפעולות הדרושות בבסיס הנתונים: `Create`, `Read`, `Update`, `Delete`, ובראשי תיבות - CRUD.

המשפט `SELECT` מאפשר לשלוף נתונים מטבלה אחת או יותר. כדי לגשת ליותר מטבלה אחת, תוכל להשתמש באחד הסוגים הזמינים של `JOIN` כדי להגדיר קשר בין הטבלאות. כדי להגביל את מספר הרשומות המוחזרות על ידי שאילתה, תוכל להשתמש בפסקה `WHERE`. כדי למיין את הפריטים בקבוצת התוצאות שהוחזרה על ידי השאילתה, תשתמש בפסקה `ORDER BY`.

כדי ליצור רשומות חדשות בבסיס הנתונים - השתמש במשפט `INSERT`. כדי לשנות רשומות קיימות - השתמש במשפט `UPDATE`. בשני המשפטים אתה קובע ערכים עבור העמודות שברצונך להוסיף או לעדכן. ערכי טקסט ותאריך צריכים להיות עטופים בגרשים. ערכים ספרתיים ובוליאניים אפשר לכתוב באופן רגיל בשאילתה.

כדי למחוק רשומות שאינך זקוק להן יותר, השתמש במשפט `DELETE`. מומלץ מאוד להשתמש בפסקה `WHERE` שמאפשרת להגביל את מספר הרשומות שיימחקו.

בחלק השני של הפרק ראית כיצד להשתמש בכלים המובנים של בסיס הנתונים, כדי ליצור טבלאות חדשות וכדי ליצור ביניהן קשרים. בנוסף, ראית כיצד קשר בין שתי טבלאות מאפשר להגן על הנתונים מפני השחתה או יתמות.

פרק זה התמקד בהוראות SQL שצריך לכתוב כדי לגשת לבסיס הנתונים. בפרק הבא תראה שבמקרים רבים הופכת VWD את הגישה לבסיס הנתונים לקלה למדי, מכיוון שהיא יוצרת את רוב הקוד הנדרש. עם זאת, ידע מוצק של SQL יעזור לך בהבנת ובהתאמת הקוד ש-VWD כותבת עבורך.

תרגילים

1. משפט DELETE ייכשל אם תנסה למחוק מהטבלה Genre רשומה שיש לה רשומות תואמות בטבלה Review. כיצד זה ייתכן?
2. משפט DELETE יצליח כשתנסה למחוק מהטבלה Review רשומה שה-GenreId שלה מוגדר ל-Id של רשומה שקיימת בטבלה Genre. מדוע?
3. נניח שברצונך לנקות את בסיס הנתונים שלך, ולכן החלטת למחוק את כל רשומות הטבלה Review שיש להן Id שערכו 100 או פחות. איך ייראה משפט SQL שלך?
4. נניח שברצונך למחוק את הסגנון בעל ID שערכו 4. לפני שאתה מוחק את הסגנון, ברצונך להקצות מחדש את הסקירות שמוקצות לסגנון זה לסגנון אחר בעל ID שערכו 11. איזה משפטי SQL דרושים כדי לבצע זאת?

פרק 12

הצגה ועדכון של נתונים

לאחר שלמדת כיצד לפעול באופן אינטראקטיבי מול בסיסי נתונים תוך שימוש ב-SQL, הגיע הזמן להתחיל להיות מעשי ולהשתמש בכל הידע הזה כדי לבנות כמה דפים שימושיים שיאפשרו לך לנהל את הנתונים באתר האינטרנט שלך.

בפרק זה תלמד כיצד להציג נתונים, להכניס נתונים לבסיס הנתונים, וגם לעדכן ולמחוק נתונים תוך שימוש בפקדי הנתונים שכלולים בחבילת התוכנה ASP.NET. נוסף לעבודה עם הפקדים החזותיים שמשמשים להצגה ועריכה של נתונים בדף אינטרנט, תלמד גם להשתמש בפקד SqlDataReader שמשמש כגשר בין בסיס הנתונים ודפי ASPX שלך.

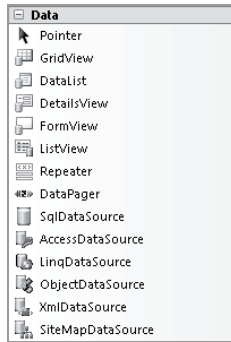
בפרק זה תלמד בהרחבה על הנושאים הבאים:

- ⊙ פקדים שמציגים ומתחזקים נתונים באתר אינטרנט.
- ⊙ כיצד להציג נתונים במבנה טבלאי תוך שימוש בפקד GridView.
- ⊙ כיצד לגשת לבסיס נתונים כדי לשלוף ולעדכן נתונים תוך שימוש בפקד SqlDataReader.
- ⊙ איך להכניס, לערוך ולמחוק נתונים תוך שימוש בפקדי נתונים כמו GridView ו-DetailsView.
- ⊙ כיצד ליצור באמצעות פקדי אימות התוקף של ASP.NET ממשק עשיר, שיאפשר למשתמש להכניס ולערוך נתונים ולשמור על שלמות הנתונים.
- ⊙ הדרך הטובה ביותר לאחסון מחרוזות החיבור ביישום, באופן שתהיינה ניתנות לעדכון בקלות.

פקדי נתונים (Data Controls)

ASP.NET מציעה שתי קבוצות פקדי נתונים, שמאפשרים לפעול ביעילות על נתונים במערכת: **פקדים קשורי-נתונים** (data-bound controls) ו**פקדי מקור נתונים** (data source controls).

הקבוצה הראשונה מכילה פקדים שמשתמשים בהם להצגה ועדכון של נתונים, כמו הפקדים Repeater, GridView ו-ListView. פקדי מקור נתונים מאחזרים נתונים ממקור נתונים, כמו בסיס נתונים או קבצי XML, ואז מגישים אותם לפקדים קשורי-נתונים. תרשים 12-1 מציג את הרשימה המלאה של פקדי הנתונים שזמינים בקטגוריה Data של ארגז הכלים.



תרשים 12-1

שני הסעיפים הבאים מספקים סקירה מהירה של כל הפקדים בקטגוריה Data. בהמשך הפרק תלמד בפירוט רב יותר על כמה מהפקדים האלה, וכיצד להשתמש בהם.

פקדים קשורי-נתונים (Data-Bound Controls)

שבעת הפקדים הראשונים בארגז הכלים שמוצגים בתרשים 12-1 נקראים בשם פקדים קשורי-נתונים. הם משמשים להצגה ולעריכה של נתונים בדפי אינטרנט. הפקדים GridView, DataList, ListView ו- Repeater יכולים להציג רשומות רבות בו-זמנית, בעוד שהפקדים DetailsView ו- FormView מתוכננים להצגת רשומה אחת בכל פעם. הפקדים DataPager ו- ListView הם חדשים ב- ASP.NET 3.5, והראשון מספק לשני יכולות דפדוף.

פקדי רשימה (List Controls)

מכיוון ש-ASP.NET מציעה פקדים רבים להצגת רשימות של רשומות, מתעוררת השאלה מתי לבחור ובמי מהם.

GridView הוא פקד רב-גוני שתומך בדפדוף, מיון, עריכה, מחיקה ובחירה אוטומטיים. הוא מציג את הנתונים שלו כטבלה בעלת שורות ועמודות. למרות שקיימות אפשרויות רבות לעיצוב המראה של שורות ופקדים אלה (תלמד על כך יותר בפרק 14), אין אפשרות לשנות באופן יסודי את צורת הצגת הנתונים. בנוסף, ה-GridView אינו מאפשר להכניס רשומות ישירות למקור הנתונים. תרשים 12-2 מציג GridView אופייני.

	Id	Title	Date Created
Edit Delete Select	9	The National - Boxer	11/29/2007 10:10 PM
Edit Delete Select	10	P.J. Harvey - White Chalk: Wow: really something new...	12/12/2007 1:08 PM
Edit Delete Select	11	Interpol - Our Love to Admire	12/12/2007 1:29 PM
Edit Delete Select	12	Yeah Yeah Yeahs - The Is Is EP	12/20/2007 10:36 PM
1 2 3 4 5 6			

תרשים 12-2

הפקד GridView משמש לעתים קרובות להצגת רשימות נתונים מורכבות, כמו רשימה של סקירות בחלק management של האתר Planet Wrox, רשימת פריטים בעגלת קניות וכו'.

הפקד DataList מאפשר להציג נתונים במעין מטריצה. בנוסף, הפקד גם מאפשר להגדיר את המראה והתחושה של הנתונים על ידי קבוצת תבניות. בצד החסרונות אפשר למנות את העובדה שהפקד אינו תומך באופן טבעי בדפדוף ומיון, ואינו מאפשר להכניס רשומות חדשות או לעדכן ולמחוק רשומות קיימות.

הפקד DataList משמש לעתים קרובות להצגת נתונים חוזרים בשורות ועמודות - לדוגמה, תמונות באלבום תמונות, או מוצרים בקטלוג מוצרים.

הפקד Repeater נותן את הגמישות הרבה ביותר במונחי קוד HTML שנשלח כפלט לדפדפן, מכיוון שהפקד עצמו אינו מוסיף HTML כלשהו לפלט הדף. אתה מגדיר את מלוא סימני העריכה של הלקוח בעזרת המספר הרב של תבניות שהפקד חושף. עם זאת, לגמישות זו יש מחיר. לפקד אין יכולות מובנות לדפדוף, מיון ושינוי נתונים.

הפקד Repeater שימושי ביותר במצבים שדרושה בהם שליטה מדויקת בסימני העריכה שנוצרים על ידי הפקד. כך הוא משמש לעתים קרובות עבור רשימות HTML, סדרות או לא-סדרות (ו-). בפרק הבא תלמד יותר על הפקד Repeater.

הפקד ListView חדש ב- ASP.NET 3.5, והינו השילוב הטוב של תכונות הפקדים האחרים GridView, DataList ו- Repeater. בדומה ל- GridView הוא תומך בעריכה, במחיקה ובדפדוף של נתונים. הוא תומך בעיצוב רב-שורות ורב-עמודות בדומה ל- DataList, והוא גם מאפשר שליטה מלאה בסימני העריכה שנוצרים על ידי הפקד, בדיוק כמו Repeater. בפרק 13 תלמד הרבה יותר על הפקד ListView.

פקדי פריט יחיד (Single Item Controls)

הפקדים DetailsView ו- FormView דומים במידת מה, בכך ששניהם יכולים להציג רשומה אחת בכל פעם. הפקד DetailsView משתמש בעיצוב טבלאי מובנה כדי להציג את הנתונים, בעוד שהפקד FormView משתמש בתבניות כדי לאפשר את הגדרת המראה והתחושה של הנתונים. FormView הינו מבוסס-תבנית ויכול להיראות בדומה לזה שמוצג בתרשים 12-3.



Id:	1
Title:	Sam's Town by The Killers - Really the best album in
Summary:	Is the new album by the Killers - Sam's Town, really the best album of the last 20 years? Read this review to find out!
CreateDateTime:	2007/11/23
Update Cancel	

תרשים 12-3

שני הפקדים מאפשרים להגדיר תבניות עבור מצבים שונים, כמו הצגת נתונים לקריאה בלבד, והכנסה ועדכון של נתונים. בהמשך הלימוד בפרק זה, תלמד איך להתאים תבניות אלו לצרכיך.

פקדי דפדוף (Paging Controls)

הפקד `DataManager` הינו חדש בארגז הכלים של ASP.NET 3.5. זהו פקד פשוט אך שימושי, שמאפשר לפקד אחר להשתמש בו כדי לבצע פעולת דפדוף על נתונים. הבעיה היא שבשלב זה הוא יכול לשמש להרחבת פעולת הדפדוף לפקד `ListView` בלבד, אך ככל הנראה בגרסאות עתידיות של סביבת העבודה .NET. הוא יתמוך בפקדים נוספים. הפקדים `ListView` ו-`DataManager` מתוארים שוב בפרק 13.

כדי שהפקדים קשורי-נתונים יוכלו להציג משהו שימושי, עליך להקצות להם מקור נתונים. שתי שיטות עיקריות עומדות לרשותך כדי לקשור מקור נתונים זה לפקדים: באפשרותך להקצות מקור נתונים למאפיין `DataSource` של הפקד, או להשתמש באחד מפקדי מקור נתונים. בפרקים מאוחרים יותר תלמד להשתמש במאפיין `DataSource`. פקדי מקור נתונים השונים יידונו בסעיף הבא.

פקדי מקור נתונים (Data Source Controls)

הקטגוריה `Data` של ארגז הכלים מכילה ששה פקדי מקור נתונים שונים שבאפשרותך להשתמש בהם כדי לקשור נתונים לפקדים קשורי-נתונים. אלה הם הפקדים האחרונים בקבוצה: `XmlDataSource` ו-`SiteMapDataSource`. שניהם משמשים לקשירת נתונים היררכיים מבוססי-XML אל פקדי מקור הנתונים. את `SiteMapDataSource` כבר ראית בעבודה, כשיצרת את מפת האתר בפרק 7.

הפקד `AccessDataSource` משמש להצגת נתונים בדפי האינטרנט מבסיס הנתונים `Access` של מיקרוסופט בלבד.

הפקד `ObjectDataSource` מאפשר לחבר את הפקדים קשורי-נתונים אל אובייקטים נפרדים ביישום. במקום לקשור את הפקדים ישירות לבסיס נתונים, אתה קושר אותם לאובייקטים מותאמים אישית משכבה נפרדת.

שני הפקדים האחרים, `SqlDataSource` ו-`LinqDataSource`, נדונים בפרק זה ובפרק 13 בהתאמה.

פעולה משותפת של פקדים קשורי-נתונים ופקדי מקור נתונים

הפקד `SqlDataSource` מאפשר לך ליצור במהירות דפי אינטרנט פונקציונליים מונעי בסיס נתונים (data-driven). ביכולתך ליצור דפי אינטרנט שיאפשרו ביצוע ארבע הפעולות - יצירה, קריאה, עדכון ומחיקה - מבלי לכתוב קוד רב. למרות שעשוי להשתמע משם הפקד שהוא יכול לגשת לשרת SQL בלבד, אין זה נכון. הפקד יכול לגשת גם לבסיסי נתונים אחרים כגון Oracle או MySQL.

הצגה ועריכה של נתונים על ידי GridView

מטרת התרגיל הבא היא לתת הסבר על פעולת הפקד `SqlDataSource` בשילוב עם פקדים קשורי-נתונים (`data-bound`). לשם כך מראה התרגיל כיצד ליצור דף אינטרנט מונע-נתונים פשוט, שמאפשר לנהל את הנתונים שמאוחסנים בטבלה `Genre` בבסיס הנתונים. פרק זה מניח שבסיס הנתונים `PlanetWrox.mdf`, שכולל את הטבלאות `Genre` ו-`Review`, נמצא בתיקייה `App_Data` שלך. הנחה נוספת היא שכל אחת מהטבלאות האלו מכילה כמה רשומות. אם לא עקבת אחר הצעדים בפרק הקודם, או שאינך בטוח אם עקבת אחריהם כהלכה, גרור את הקבצים `PlanetWrox.mdf` ו-`PlanetWrox_log.ldf` מהתיקייה `Resources` שבקוד עבור פרק זה (`C:\BegASPNET\Resources\Chapter 12`), ושחרר אותם בתיקייה `App_Data` של האתר `Planet Wrox`, תוך דריסת בסיס הנתונים הקיים (אם יש כזה). גם במקרה שהעותק שלך של בסיס הנתונים אינו מכיל מספר רב של רשומות וסקירות וסגנונות (`genres`, ז'אנר), מומלץ שתעביר עותק של בסיס הנתונים `PlanetWrox.mdf` מהתיקייה `Resources`. בדרך זו תהיה לך קבוצה נאה של רשומות לדוגמה, כדי לפעול עליהן.

נסה זאת שימוש בפקדים `GridView` ו-`SqlDataSource`

בתרגיל זה תתחיל לבנות את החלק `Management` של אתר האינטרנט. זו תהיה נקודת הכניסה הראשית שלך לניהול דברים באתר, כמו סקירות וסגנונות. בינתיים יהיו הדפים שתיצור בסעיף זה נגישים לכל המשתמשים באתר. בפרק 15 תלמד כיצד לחסום את הגישה לתיקייה זו לכל משתמש שאינו מינהלן.

תלמד כיצד לגרור טבלה מסייר בסיס הנתונים (סייר השרת בגרסאות המסחריות של `Visual Studio`) אל הדף, ולגרום ל-`VWD` ליצור ממשק משתמש לניהול פריטים בבסיס הנתונים על ידי יצירת הקוד הנדרש עבור `GridView` ו-`SqlDataSource` באופן אוטומטי. בתרגילים מאוחרים יותר בספר תלמד כיצד לשעתק התנהגות זו באופן ידני, והדבר ייתן לך שליטה רבה יותר על הקוד שנוצר.

1. ב-`VWD`, פתח את הפרויקט `Planet Wrox` מתוך `C:\BegASPNET\Site`.
2. לחץ לחיצה ימנית על התיקייה `MasterPages`, בחר `Add New Item`, והוסף לאתר דף ראשי חדש בשם `ManagementMaster.master`. ודא שהוא משתמש בשפת התכנות שלך ושאינו מבוסס על דף ראשי קיים.
3. שנה את קוד `HTML` שבתוך האלמנט `<form>` לקוד שלהלן, שיוצר שני אלמנטים `<div>` שצפים סמוכים זה לזה. הראשון מכיל תפריט פשוט מבוסס-רשימה עבור החלק `Management`, בעוד השני מכיל את הפקד `ContentPlaceHolder` שמאפשר לדפי תוכן לספק תוכן מותאם אישית.

```
<form id="form1" runat="server">
<div>
```

```

<div style="width: 200px; float: left;">
    <ul>
        <li><a href="~/Management/Default.aspx" runat="server">Management Home</a></li>
        <li><a href="~/Management/Genres.aspx" runat="server">Manage Genres</a></li>
    </ul>
</div>
<div style="width: 750px; float: left;">
    <asp:ContentPlaceHolder ID="cpMainContent" runat="server"></asp:ContentPlaceHolder>
</div>
</div>
</form>

```

אתה יכול להתעלם מהאזהרה שמציגה VWD בדבר הדפים החסרים, מכיוון שהם תיוספו מאוחר יותר. שמור וסגור את הדף הראשי.

4. הוסף תיקייה חדשה לשורש האתר וקרא לה Management. לחץ לחיצה ימנית על תיקייה חדשה זו, בחר Add New Item, וצור דף חדש בשם Default.aspx. ודא שהדף מתבסס על הקובץ החדש ManagementMaster.master. אין צורך לבסס אותו על המחלקה המרכזית BasePage שבתיקייה App_Code. ברגע שהקובץ נפתח בחלון המסמך, הוסף לבלוק התוכן של הפקד cpMainContent מעט טקסט, שמקדם בברכה את המשתמש בחלק Management של אתר האינטרנט:

```

<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" runat="Server">
    <h1>Planet Wrox Management Section</h1>
    <p>Welcome to the Management section of this web site. Please choose an item from the
        menu on the left to continue.</p>
</asp:Content>

```

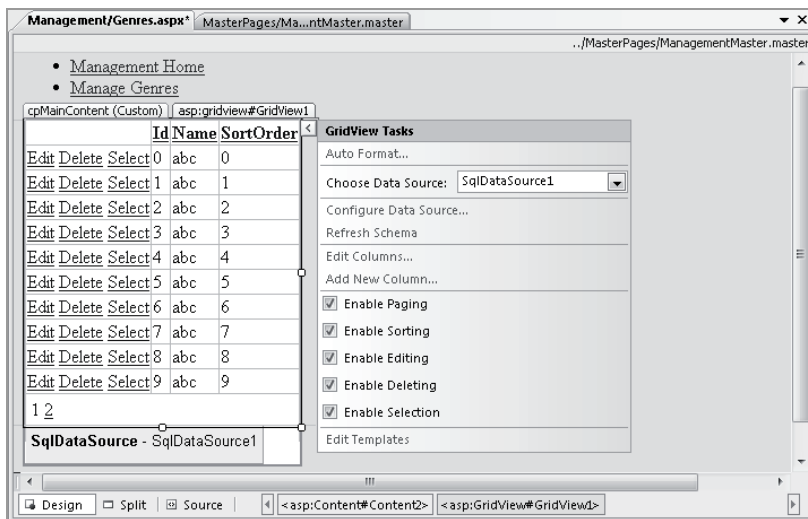
קבע לדף את הכותרת הזו: Planet Wrox - Management - Home.

5. צור דף נוסף בתיקייה Management וקרא לו Genres.aspx. בסס אותו על הדף הראשי שיצרת וקבע לו את הכותרת: Planet Wrox - Management - Genres.

6. העבר את הדף ל- Design View. ודא שחלון סייר בסיס הנתונים פתוח. אם בסיס הנתונים Planet Wrox אינו רשום שם, פנה לתרגיל נסה זאת הראשון בפרק הקודם שמסביר איך ליצור את החיבור. זכור שתיקיית המשאבים עבור פרק זה מכילה בסיס נתונים שכולל את הטבלאות שתפעיל במהלך הלימוד בפרק.

7. הרחב את בסיס הנתונים PlanetWrox.mdf, ואחר כך הרחב את הצומת Tables. גרור את הטבלה Genre מסייר בסיס הנתונים ושחרר אותה בשטח cpMainContent של הדף Genres. VWD יוצרת GridView ו- SqlDataSource באופן אוטומטי.

8. בלוח Smart Tasks של הפקד GridView, שאמור להיפתח באופן אוטומטי, בחר בכל האפשרויות הזמינות, כפי שמוצג בתרשים 4-12. אם הפקד לא ייפתח באופן אוטומטי, לחץ על החץ האפור שבפינה הימנית העליונה של הפקד, או לחץ לחיצה ימנית על הפקד ובחר Show Smart Tag, והמשך בבחירה.

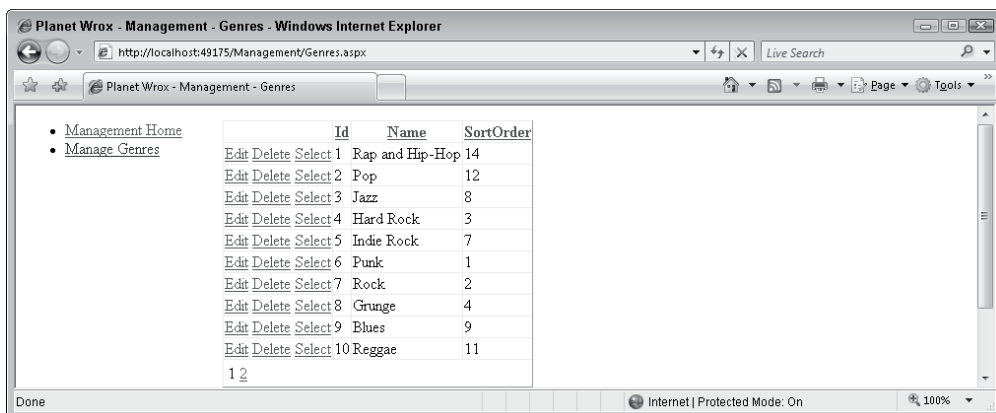


תרשים 12-4

9. בסייר הפתרון, לחץ לחיצה ימנית על התיקיה Management ובחר Add New Item. לחץ אחת על Web Configuration File ואז לחץ Add כדי להוסיף קובץ web.config שחל על התיקיה Management בלבד. לקובץ שייפתח הוסף אלמנט <pages> שמגדיר את ערכת הנושא למחרוזת ריקה:

```
<configuration>
  <appSettings/>
  <connectionStrings/>
  <system.web>
    <pages theme=""></pages>
  </system.web>
</configuration>
```

10. שמור את כל השינויים ובקש בדפדפן את הדרך Genres.aspx מהתיקיה Management (ראה תרשים 12-5).



תרשים 12-5

אתה אמור לראות רשת עם הסגנונות (genres) מהטבלה Genre. הקישורים בעמודות השמאליות מאפשרים לך לערוך את הסגנונות הרלוונטיים, למחוק אותם או לבחור בהם. שים לב שאינך יכול למחוק סגנונות שיש סקירה אחת או יותר שמקושרת אליהם. אם תנסה זאת, תקבל הודעת שגיאה. פרק 14 מרחיב את ההסבר על שינוי ממשק המשתמש (UI) במטרה לנטרל את הקישורים מסוג Delete, כדי שמשתמשים לא יוכלו ללחוץ עליהם בטעות.

אם רשימת הסקירות (ביקורות) מסתיימת מתחת לתפריט שמשמאל, תצטרך להרחיב מעט את חלון הדפדפן.

11. באפשרותך ללחוץ על כותרות העמודות שמופיעות בתרשים 5-12, כדי למיין את הנתונים לפי אותה עמודה. אם תלחץ על אותה כותרת פעם נוספת, הנתונים ימוינו בסדר הפוך.

12. לחץ על הקישור Edit עבור אחד מהסגנונות, בתיבת הטקסט שהופיעה, שנה את השם ולחץ על הלחצן Update. הפקד GridView אמור להציג כעת את השם החדש.

כיצד זה פועל

בתרגיל זה לא כתבת קוד רב, ובכל זאת קיבלת פונקציונליות רבה, על ידי גרירה ושחרור של טבלת בסיס נתונים. כדי לראות כיצד זה פועל, עיין בקוד המקור שנוצר על ידי VWD. הבט תחילה בסימני העריכה עבור הפקד `SqlDataSource`:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%= $ ConnectionStrings:PlanetWroxConnectionString1 %>"
    ProviderName="<%= $ ConnectionStrings:PlanetWroxConnectionString1.ProviderName %>"
    DeleteCommand="DELETE FROM [Genre] WHERE [Id] = @Id"
    InsertCommand="INSERT INTO [Genre] ([Name], [SortOrder]) VALUES (@Name, @SortOrder)"
    SelectCommand="SELECT [Id], [Name], [SortOrder] FROM [Genre]"
    UpdateCommand="UPDATE [Genre] SET [Name] = @Name, [SortOrder] = @SortOrder
        WHERE [Id] = @Id">
    <DeleteParameters>
        <asp:Parameter Name="Id" Type="Int32" />
    </DeleteParameters>
    <InsertParameters>
        <asp:Parameter Name="Name" Type="String" />
        <asp:Parameter Name="SortOrder" Type="Int32" />
    </InsertParameters>
    <UpdateParameters>
        <asp:Parameter Name="Name" Type="String" />
        <asp:Parameter Name="SortOrder" Type="Int32" />
        <asp:Parameter Name="Id" Type="Int32" />
    </UpdateParameters>
</asp:SqlDataSource>
```


כדאי לבחון כאן כמה דברים מעניינים. שים לב שהמאפיינים `ProviderName=1` `ConnectionString` שים לב שהמאפיינים `web.config`. בסעיף הבא נעסוק יותר בעניין זה, מצביעות על מחרוזת חיבור שהוגדרה בקובץ `web.config`. בסעיף הבא נעסוק יותר בעניין זה, כולל הסבר לתחביר `<%$ %>` שמשמשים בו במחרוזת החיבור ובמאפיין `ProviderName`. אז תראה ארבע פקודות, שכל אחת מהן מכילה משפט SQL שמשמש לאחת מארבע הפעולות - יצירה, קריאה, עדכון ומחיקה. הפקודות `INSERT`, `UPDATE` ו-`DELETE` מכילות פרמטרים שמזוהים על ידי התחילית `@`. בזמן ביצוע, כשהפקד מתבקש לבצע את הפעולות הרלוונטיות על הנתונים, מוחלפים פרמטרים אלה בערכי זמן-ביצוע. הפקד `SqlDataSource` משתמש באוספים ממשפחת `Parameters` כדי לנהל מעקב אחר הפרמטרים הרלוונטיים. לדוגמה, האלמנט `<DeleteParameters>` מכיל פרמטר יחיד עבור ה-`Id` (המפתח הראשי) של הסגנון.

```
<DeleteParameters>
  <asp:parameter Name="Id" Type="Int32" />
</DeleteParameters>
```

שים לב שה-`Name` של הפרמטר מייצג את הפרמטר במשפט SQL עבור ה-`DELETE`.

```
DeleteCommand="DELETE FROM [Genre] WHERE [Id] = @Id">
```

בשלב זה, כשהוא לעצמו, הפקד `SqlDataSource` אינו יכול לעשות הרבה. הוא זקוק לפקד קשור-נתונים שיאמר לו אילו פעולות לבצע על הנתונים. בתרגיל לעיל, הפקד קשור-הנתונים הוא `GridView`, שמוגדר על ידי הקוד הבא:

```
<asp:GridView ID="GridView1" runat="server" AllowPaging="True" AllowSorting="True"
  AutoGenerateColumns="False" DataKeyNames="Id" DataSourceID="SqlDataSource1"
  EmptyDataText="There are no data records to display.">
  <Columns>
    <asp:CommandField ShowDeleteButton="True" ShowEditButton="True"
      ShowSelectButton="True" />
    <asp:BoundField DataField="Id" HeaderText="Id" ReadOnly="True" SortExpression="Id" />
    <asp:BoundField DataField="Name" HeaderText="Name" SortExpression="Name" />
    <asp:BoundField DataField="SortOrder" HeaderText="SortOrder"
      SortExpression="SortOrder" />
  </Columns>
</asp:GridView>
```

הפקד `GridView` מכיל כמה מאפיינים חשובים. המאפיין `DataKeyNames` אומר ל-`GridView` מהו המפתח הראשי של הרשומה בבסיס הנתונים. הפקד זקוק לזה כדי שיוכל לזהות באופן פרטני את הרשומות המוצגות בו.

המאפיין `DataSourceID` מצביע על הפקד `DataSourceID` שראית קודם, בעוד `AllowPaging` ו-`AllowSorting` (כששם הם) מאפשרים ב-`GridView` מיון ודפדוף.

תחת האלמנט `<Columns>` מוגדרים כמה שדות. תחילה אתה רואה את `CommandField`. זו עמודה ב-`GridView` שמאפשרת למשתמש לבצע פעולה אחת או יותר עבור השורה ש-`CommandField` חל עליה. היא מופיעה בדפדפן כקישורי טקסט, אחד או יותר, או כלחצנים. בדוגמה שלנו הוגדרו הלחצנים `ShowDeleteButton`, `ShowEditButton` ו-`ShowSelectButton` כ-`True`. דבר זה נותן לפקד

את הפונקציונליות שאתה רואה בתרשים 5-12. כשתלחץ על אחד הקישורים שנוצרו על ידי GridView, הם יעוררו פקודה בשרת. לדוגמה, לחיצה על הקישור Edit תעביר את GridView למצב Edit, כדי שתוכל לערוך את הרשומה שנבחרה. שים לב שלכאורה לחיצה על הקישור Select אינה גורמת לשינוי כלשהו ב-GridView. בפרק 14 תלמד כיצד ליצור סגנונות עבור GridView כדי שתוכל לשנות את המראה של הפקד מן היסוד, כולל הבחנה חזותית בין שורה שנבחרה לבין האחרות.

אם ברצונך ש-GridView יציג לחצנים במקום קישורים, עליך להגדיר את ButtonType ל-Button.

```
<asp:CommandField ShowDeleteButton="True" ShowEditButton="True" ShowSelectButton="True"
ButtonType="Button"></asp:CommandField>
```

שלושת השדות האחרים נקראים שדות-קשורים (bound fields), וממופים ישירות לעמודות של הטבלה Genre בבסיס הנתונים על ידי המאפיין DataField שלהם, באופן ש-GridView יודע איזה נתונים להציג, והיכן.

הפקדים GridView ו-1 SqlDataSource פועלים בשיתוף פעולה הדוק כדי לאחזר ולשנות נתונים במקור הנתונים הבסיסי. כדי שתקבל מושג כיצד זה פועל, לפניך תקציר האירועים שהתרחשו כשביקשת את הדף Genres בדפדפן, וערכת רשומת סגנון יחידה.

1. ביקשת בדפדפן דף, אשר החל את מחזור החיים שלו.

2. הפקד GridView יודע שהוא מוכן לשלוף ולהציג נתונים מכיוון שהתכונה DataSourceID שלו מוגדרת. הוא מתחבר ל-1 SqlDataSource שמזוהה על ידי המאפיין DataSourceID ומבקש ממנו את הנתונים שלו. כעת, SqlDataSource בתורו מתחבר לבסיס הנתונים ומפעיל את המשפט SelectCommand. זהו משפט SQL שבוחר את ה-ID, Name ו-SortOrder מהטבלה Genre שבבסיס הנתונים.

```
SelectCommand="SELECT [Id], [Name], [SortOrder] FROM [Genre]"
```

3. כש-1 SqlDataSource מקבל את הרשומות המבוקשות מבסיס הנתונים, הוא מוסר אותן ל-GridView. האחרון יוצר מהן טבלת HTML תוך שימוש בשדות הקשורים שהוגדרו באלמנט <Columns>. עבור כל קישור Edit, Delete ו-Select בעמודה CommandField, הפקד GridView עוקב אחר ה-ID הייחודי של כל רשומה שמוצגת בדף על ידי אחסונה ב-ViewState.

4. ברגע שאתה לוחץ על הלחצן Edit, מתבצע postback של הדף. ה-GridView יודע על איזה שורה לחצת על פי ה-DataKeyName הקשור. הוא מבקש אז מה-1 SqlDataSource לבצע שוב את הפקודה SelectCommand שלו ומקבל את הרשומה מבסיס הנתונים. בנקודה זו, GridView מכניס את השורה הנבחרת למצב עריכה, כדי שתוכל לשנות בה את הפרטים הרלוונטיים. כשתלחץ על הקישור Update, יאסוף GridView את הערכים החדשים מפקדי TextBox ויתחבר שוב ל-1 SqlDataSource.

5. הפקד GridView מספק ערך לכל אחד מהפרמטרים שבאלמנט <UpdateParameters> של SqlDataReader. הפקד מאחזר את ה-Id של הסגנון מהשורה שנבחרה ואז מאחזר את הערכים החדשים של Name ו-Id SortOrder מפקדי TextBox שברף.
6. כשהוא מצויד בנתונים הרלוונטיים של Id, Name ו-Id SortOrder, מבצע עכשיו SqlDataReader את הפקודה UpdateCommand כנגד בסיס הנתונים.

```
UpdateCommand="UPDATE [Genre] SET [Name] = @Name, [SortOrder] =  
@SortOrder WHERE [Id] = @Id"
```

- ⊙ כל אחד מהפרמטרים בעלי תחילית @ ממולא בערכים שסיפק GridView. משפט SQL שנשלח לבסיס הנתונים עשוי להיראות בסופו של דבר כך:

```
UPDATE [Genre] SET [Name] = 'New Name', [SortOrder] = 1 WHERE [Id] = 1
```

- ⊙ שמות הטבלאות והעמודות נעטפו בזוג סוגריים מרובעים (). הדבר אינו נדרש בתרחיש הנוכחי, אבל הוא שימושי במקרים שבהם שם העמודה מכיל רווח, או כששם הטבלה או העמודה תואם למילה שמורה.

7. לבסוף, GridView מרענן את הנתונים בדף על ידי פנייה נוספת אל SqlDataReader בבקשה שיבצע את SelectCommand. בדרך זו מציג GridView את הנתונים העדכניים ביותר שכוללים את העדכון שביצעת.

הפקודות האחרות פועלות בדרך דומה ושולחות לבסיס הנתונים את פקודות SQL שלהן.

בסוף התרגיל הוספת קובץ web.config חדש לתיקייה Management, ואיפסת את ערכת הנושא שמוחלת על כל הדפים בחלק Management. ברגע שערכת הנושא הוסרה, קל יותר להתמקד בפונקציונליות של חלק Management מבלי שהדעת תוסח על ידי בעיות עיצוב. בפרק 14 תיצור ערכת נושא שלישית, במיוחד עבור התיקייה Management, ותחיל אותה בקובץ web.config שבתיקייה Management. בדרך זו יהיו דפי management שלך בעלי מראה ותחושה שונים מאלה של דפי ממשק המשתמש. פרק 14 מעמיק הרבה יותר בנושא שינוי המראה של פקדי הנתונים באתר האינטרנט Planet Wrox.

הכנסת נתונים על ידי DetailsView

הכנסת נתונים על ידי הפקד DetailsView קלה בדיוק כמו הצגת הנתונים, עדכונם ומחיקתם על ידי הפקד GridView. כמו GridView כך גם DetailsView תומך בכמה תבניות שמאפשרות להתאים אישית את המראה והתחושה של הפקד במצבים שונים. לדוגמה, לפקד יש אלמנטים <HeaderTemplate>, <FooterTemplate> ו- <PagerTemplate> שמאפשרים לך להגדיר את המראה של החלק העליון והתחתון של הפקד. בנוסף, לפקד יש אלמנט <Fields> שמאפשר להגדיר את השורות שחייבות להופיע בפקד, עם דמיון רב לאלמנט <Columns> של GridView.

הפקד DetailsView יכול להציג נתונים בכמה אופנים שונים. הוא יכול להציג רשומה קיימת במצב קריאה-בלבד. בנוסף, הוא יכול לשמש להכנסת רשומות חדשות לחלוטין ולעדכון רשומות קיימות. אתה שולט במצב של DetailsView על ידי המאפיין DefaultMode שתוכל להגדירו ל-ReadOnly, Edit-1, Insert, להלן תראה כיצד לקבוע את תצורת DetailsView, ולהגדיר את המאפיין DefaultMode.

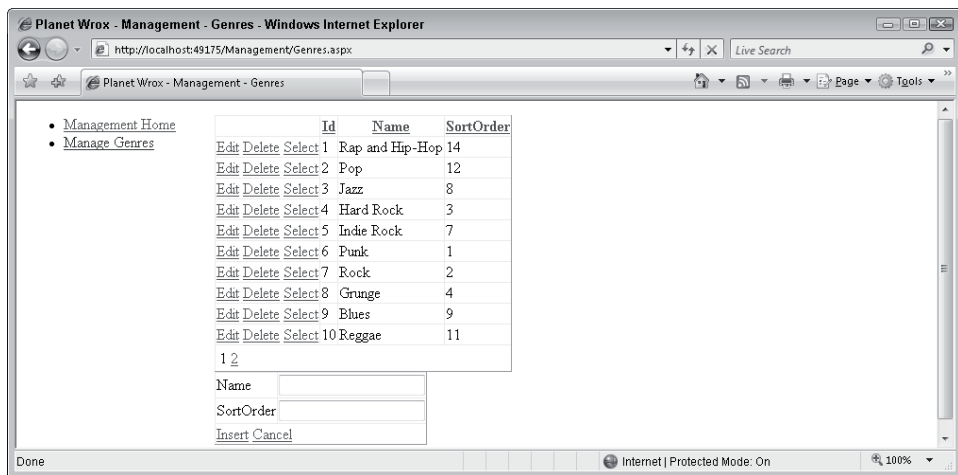
נסה זאת הכנסת נתונים בעזרת הפקד DetailsView

בתרגיל זה תלמד להשתמש בפקד DetailsView כדי לאפשר למשתמשים להכניס רשומות חדשות לטבלה Genre. כמו בדוגמה של GridView כך גם כאן, אינך צריך לקודד כלום. כל שעליך לעשות הוא לגרור ולשחרר כמה פקדים, ולהגדיר כמה מאפיינים. ברור שלדפים חופשיים מקוד יש מגבלות שהופכות אותם לפחות שימושיים בתרחישים יותר מתקדמים. לכן, תלמד בהמשך הפרק כיצד להרחיב ולהתאים פקדים אלה לצרכיך.

1. ב-VWD, חזור לדף Genres.aspx. ודא שהוא ב-Design View.
2. מהקטגוריה Data של ארגז הכלים, גרור את DetailsView ושחרר אותו מייד מתחת ל-GridView. אם מתעוררים קשיים בשחרור הפקד מתחת ל-GridView אבל מעל לפקד SqlDataReader, תוכל לשחרר את DetailsView על SqlDataReader. כעת, VWD תוסיף את סימני העריכה של הפקד המשוחרר לפני אלה של הפקד שעליו שחררת.
3. פתח את לוח Smart Tasks של הפקד (שאמור להיפתח באופן אוטומטי) וחבר אותו אל SqlDataReader1 על ידי בחירת שם זה מהרשימה הנפתחת Choose Data Source.
4. מאותו לוח Smart Tasks בחר בפריט Enable Inserting.
5. פתח את טבלת המאפיינים של הפקד על ידי הקשת F4, ואתר את המאפיין DefaultMode בקטגוריית Behavior. הגדר את DefaultMode ל-Insert. הקוד עבור DetailsView אמור להיראות כעת בדומה לזה:

```
<asp:DetailsView ID="DetailsView1" runat="server" AutoGenerateRows="False"
    DataKeyNames="Id" DataSourceID="SqlDataSource1" DefaultMode="Insert"
    Height="50px" Width="125px">
    <Fields>
        <asp:BoundField DataField="Id" HeaderText="Id" InsertVisible="False"
            ReadOnly="True" SortExpression="Id" />
        <asp:BoundField DataField="Name" HeaderText="Name" SortExpression="Name" />
        <asp:BoundField DataField="SortOrder" HeaderText="SortOrder"
            SortExpression="SortOrder" />
        <asp:CommandField ShowInsertButton="True" />
    </Fields>
</asp:DetailsView>
```

6. שמור את השינויים שערכת בדף. הקש Ctrl+F5 כדי לפתוח את הדף בדפדפן. מתחת ל-GridView אתה אמור לראות כעת את הפקדים שמאפשרים להכניס סגנון חדש (ראה תרשים 12-6).



תרשים 12-6

7. הכנס סגנון חדש. ודא שהכנסת הן את השם והן את סדר המיון, ואז לחץ על הקישור Insert.

8. סגור את הדפדפן, חזור ל-VWD והעבר את הדף ל-Markup View. את סימני העריכה שבתוך הבלוק Content cpMainContent עטוף ב-UpdatePanel של Ajax, שביכולתך לגרור אותו מארגז הכלים אל עורך הקוד. אל תשכח להוסיף ContentTemplate:

```
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" runat="Server">
  <asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
      <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
        DataKeyNames="Id" DataSourceID="SqlDataSource1"
        EmptyDataText="There are no data records to display."
        AllowPaging="True" AllowSorting="True">
        ...
      </asp:SqlDataSource>
    </ContentTemplate>
  </asp:UpdatePanel>
</asp:Content>
```

9. פתח את הדף הראשי של חלק Management (ManagementMaster.master) והוסף ScriptManager מייד לאחר תגית הפתיחה <form>.

```
<form id="form1" runat="server">
  <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
  <div>
```

10. שמור את כל השינויים ופתח שוב את Genres.aspx. הכנס כמה סגנונות נוספים. שים לב שהוספת UpdatePanel עשתה את פעולת ההכנסה לחופשייה מהבהוב.

11. אם תלחץ על אחד המספרים בסרגל Pager שבתחתית המסך, תראה ש-GridView יפנה לדף זה ויצגי את קבוצת הרשומות הבאה. שים לב שהחלפת דפים זו (קרויה דפדוף), המיון והעריכה, מתרחשים כולם בתוך UpdatePanel, וגורמים לרענון הדף ללא postback מלא.

בדיוק כמו עם שאר הפקדים קשורי-הנתונים, חיברת את DetailsView לפקד מקור נתונים על ידי הגדרת המאפיין DataSourceID. מכיוון שבדרך כבר נמצא פקד פעיל SqlDataSource, אתה יכול להשתמש בו. DetailsView משנה את התצוגה שלו בדף בהתאם למצבי עריכה, הכנסה וקריאה-בלבד. הגדרת DefaultMode ל-Insert כופה על הפקד לעבור למצב Insert. פירוש הדבר שבאופן אוטומטי תקבל ממשק משתמש (UI) שיאפשר לך להכניס פרטים עבור הסגנון וקישורים עבור Insert ו-Cancel.

הפקד DetailsView הינו חכם למדי. כשאתה מכוון אותו אל הפקד SqlDataSource, הוא מסוגל להבין את המאפיין DataKeyNames שמוגדר ל-Id:

```
<asp:DetailsView ID="DetailsView1" runat="server" AutoGenerateRows="False"
    DataKeyNames="Id" DataSourceID="SqlDataSource1" DefaultMode="Insert" Height="50px"
```

פקד זה גם מבין שהעמודה Id היא עמודת זהות בבסיס הנתונים, ולפיכך הוא מסתיר אותה במסך Insert שבתרשים 6-12 על ידי הגדרת InsertVisible ל-False. מכיוון שבסיס הנתונים יוצר את ה-ID הזה באופן אוטומטי, אין טעם להרשות למשתמש להכניס ערך עבורו. כשאתה מכניס כמה ערכים ולוחץ על הקישור Insert, מתרחש תהליך שדומה לעדכון על ידי GridView. ה-DetailsView אוסף את הערכים הרלוונטיים מפקדי הדף (Name ו-SortOrder) ומעביר אותו הלאה אל SqlDataSource. פקד זה בתורו מכניס את הערכים החדשים לתוך הפרמטרים של משפט INSERT, ושולח את הפקודה לבסיס הנתונים. בסיס הנתונים מכניס את הרשומות החדשות לטבלה Genre.

בסוף התרגיל עטפת את תוכן הדף בתוך UpdatePanel. במאמץ מוערי שיפרת את התנהגותו הוויזואלית של הדף, ואת ביצועיו בדפדפן. לאחר שנפטרת מ-postbacks, הדף נראה כמתנהג במהירות גבוהה יותר, ולכן משאיר על המשתמש רושם רב יותר.

אחסון מחרוזות החיבור (Connection Strings) ב-web.config

בפעם הראשונה ששחררת את הטבלה Genre בדף שלך, יצרה עבורך VWD פקד SqlDataSource. כדי לומר לפקד זה לאיזה בסיס נתונים לגשת, יצרה VWD גם מחרוזת חיבור בקובץ web.config, תחת האלמנט <connectionStrings>, וכיוונה את SqlDataSource לאותה מחרוזת חיבור. ההגדרות ב-web.config נראות בדומה לזה:

```
<connectionStrings>
  <add name="PlanetWroxConnectionString1" connectionString="
    Data Source=.\\SQLEXPRESS;
    AttachDbFilename=|DataDirectory|\\PlanetWrox.mdf;
    Integrated Security=True;User Instance=True"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

לפיכך, ה-SqlDataSource ניגש למחרוזת חיבור זו:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%%$ ConnectionStrings:PlanetWroxConnectionString1 %>"
```

כדי לפנות למחרוזת החיבור ב-web.config משתמש קוד זה בתחביר קשירת ביטוי (expression binding syntax). למעשה הוא מבקש מקובץ web.config את מחרוזת החיבור שנענית לשם PlanetWroxConnectionString1.

בנוסף לתחביר קשירת ביטוי שמשתמש ב-`<%%$ %>` כדי לקשור ערכי פקד למשאבים כמו ערכים ממחרוזת החיבור, תפגוש תחביר דומה שמשתמש ב-`<#% %>`. האחרון נקרא תחביר ביטוי קשור-נתונים (data-binding expression syntax). הוא מאפשר לקשור ערכי פקד עם נתונים שבאים ממקורות נתונים, כמו בסיסי נתונים. בהמשך פרק זה, ובשני הפרקים הבאים, תמשיך לפגוש בתחביר ביטוי קשור-נתונים.

אחסון מחרוזות החיבור ב-web.config נחשב לנזהר נכון וטוב. בדרך זו אתה מרכז את מחרוזות החיבור במקום אחד, והדבר מקל מאוד על שינוי המחרוזות כשבסיס הנתונים משתנה (לדוגמה, כשאתה עובר מסביבת פיתוח לשרת ייצור). לעולם אל תאחסן את מחרוזות החיבור ישירות בקבצי קוד ברקע, או בין סימני העריכה של הדף. אם תעשה זאת, תתחרט על כך מאוד ביום שיהיה עליך לשנות את מחרוזת החיבור, ואז תצטרך לעבור דרך כל הדפים באתר בחיפוש אחר מחרוזות חיבור.

הגרסה Express Edition של שרת SQL, שבה השתמשת עד כה, מאפשרת לעבוד עם בסיסי נתונים שמצורפים לשרת SQL ברגע שאתה זקוק להם בזמן ביצוע. התבונן במחרוזת החיבור הממשית, כדי להבין כיצד זה פועל:

```
Data Source=.\SQLEXPRESS;
AttachDbFilename=|DataDirectory|\PlanetWrox.mdf;
Integrated Security=True;
User Instance=True"
```

מחרוזת חיבור זו מורכבת מארבעה חלקים. הראשון מכיל את מקור הנתונים לזיהוי שרת SQL שהוא היעד. חלק זה עצמו יכול להיות מפוצל לשני חלקים. החלק שלפני הלוכסן האחורי (במקרה שלנו זו נקודה) מזהה את השרת. הנקודה מפנה למחשב המקומי, אם כי יכולת להשתמש גם בביטוי (local). בנוסף, באפשרותך להכניס כאן את שם השרת במקום הנקודה. החלק שלאחר הלוכסן הוא רשות ומתייחסים אליו כשם המופע (instance name) או בפשטות instance. מכיוון שניתן להתקין מופעים רבים של שרת SQL באותו מחשב זה לצד זה, משמש שם המופע כדי לפנות למופע ייחודי של שרת SQL. בהתקנת SQL Server Express Edition 2005, ברירת המחדל של שם המופע היא SqlExpress. לכן, כדי לפנות למופע של SQL Server 2005 Express Edition במחשב המקומי, עליך לפנות אליו על ידי SqlExpress. או על ידי (local)\SqlExpress.

AttachDbFileName מכיל נתיב אל בסיס הנתונים SQL Server Express. בזמן ביצוע מורחב שומר המקום |DataDirectory| לנתיב המלא של התיקיה App_Data. לפיכך, כשהדפים נטענים, ו-SqlDataSource צריך להתחבר לבסיס הנתונים, הוא יפתח את הקובץ PlanetWrox.mdf

שב- C:\BegASPNET\Site\App_Data\ כחלופה ל-AttachDbFileName ייתכן שבמחזרות חיבור
 Initial Catalog. ה- Initial Catalog מצביע על בסיס נתונים זמין בשרת SQL
 שאתה משתמש בו, בדומה ל-PlanetWrox. תלמד על כך יותר בנספח ב'.

שני החלקים האחרונים במחזרות החיבור עוסקים באבטחה. עם Integrated Security, החשבון
 ששרת האינטרנט משתמש בו מיועד גם להתחברות לבסיס הנתונים. במקרה של VWD ושרת
 האינטרנט המובנה, חשבון זה הוא אותו חשבון שאתה משתמש בו כדי להתחבר (log on) למחשב
 שלך. במקרה של שימוש ב-IIS, חשבון זה הוא חשבון מיוחד של ASP.NET. נוסף ונתעמק
 בנושאי אבטחה בפרק 18, וגם בנספח ב' שיש בו הסבר על החיבור לשרת SQL.

עד כה הכרת את רוב מושגי בסיס הנתונים שהוצגו בפרק הקודם. ראית פעולות יצירה (עם
 GridView במצב DetailsView), קריאה (עם SelectCommand ו-GridView), עדכון (כלול בתוך GridView
 ו-UpdateCommand), ומחיקה (עם אפשרות המחיקה ב-GridView ו-DeleteCommand). ראית גם
 מיון שיכול להיות מאופשר ב-GridView על ידי מאפיין אחד בלבד. עדיין לא ראית סינון, שהוא דרך
 להגביל את כמות הנתונים שמוצגים בדף. בסעיף הבא תראה כיצד ליצור מסנן שמאפשר להציג
 סקירות ששייכות רק לסגנון מסוים. תבצע זאת בחלק Management של דף חדש בשם
 Reviews.aspx. דף זה יהיה נקודת הכניסה הראשית שלך לניהול הסקירות באתר. הסעיפים הבאים
 ייבנו על דף זה, ובסופו של דבר ירחיבו בהדרגה את הדף Reviews על ידי תכונות שימושיות נוספות.

סינון נתונים

כפי שלמדת בפרק הקודם שהציג את פקודות SQL, סינון נתונים (data filtering) מתבצע בעזרת
 הפסקה WHERE. למרבה המזל, VWD ו-ASP.NET מגיעות עם צרור כלים שמקלים מאוד על
 יצירת מסננים. כדי לסנן נתונים יש לפקד SqlDataReader (ולפקדי מקור נתונים אחרים) אלמנט
 <SelectParameters>, שמאפשר לספק בזמן ביצוע ערכים שימשו כמסננים. ערכים אלה
 יכולים לבוא ממקורות שונים, שמוצגים בטבלה הבאה:

על ידי	הערך מאוחזר מתוך
ControlParameter	פקד בדף, כמו DropDownList או TextBox.
CookieParameter	עוגייה שמאוחסנת במחשב המשתמש, ונשלחת לשרת עם כל בקשה.
FormParameter	ערך שנמצא בטופס שנשלח לשרת.
Parameter	מגוון מקורות. עם פרמטר זה, הערך נקבע בדרך כלל על ידי קוד.
ProfileParameter	מאפיין בפרופיל המשתמש. נושא הפרופיל ב-ASP.NET יידון בהרחבה בפרק 16.
QueryStringParameter	שדה של מחזרות שאילתה.
SessionParameter	ערך שמאוחסן ב-session שהוא מחסן נתונים ייחודי למשתמש, ומתקיים למשך ביקור המשתמש באתר.

מכיוון שכל הפרמטרים האלה מתנהגים בצורה דומה, פחות או יותר, קל להשתמש בהם בקוד. ברגע שתבין כיצד להשתמש באחד מהם, תדע להשתמש גם באחרים בקלות רבה. בתרגיל הבא תראה את ControlParameter בפעולה. בתרגיל תשתמש ב-DropDownList עם כל הסגנונות, כדי לסנן רשימה של סקירות ששייכות לסגנון שנבחר.

נסה זאת הגדרת המסנן

כדי שרשימות נתונים ארוכות תהיינה קלות לניהול, כדאי להציע אותן למשתמש בחלקים "ניתנים לעיכול". לדוגמה, כשצריך להציג רשימה של סקירות מבסיס הנתונים, יהיה זה רעיון טוב לאפשר למשתמש לסנן אותן לפי הסגנון ולהציג למשתמש כל רשימה בנפרד. פתרון מושלם לכך יהיה שימוש בפקד DropDownList עם הסגנונות בדיוק מעל ל-GridView שיכיל הסקירות. מייך תראה כיצד לבנות זאת.

1. בתיקייה Management, צור דף חדש בשם Reviews.aspx. ודא שהוא מבוסס על הדף הראשי של Management.

שנה את הכותרת של הדף ל- Planet Wrox - Management - Reviews.

2. בדף הראשי של החלק Management, הוסף קישור לדף שיצרת.

```
<li><a href="~/Management/Genres.aspx" runat="server">Manage Genres</a></li>
<li><a href="~/Management/Reviews.aspx" runat="server">Manage Reviews</a></li>
</ul>
```

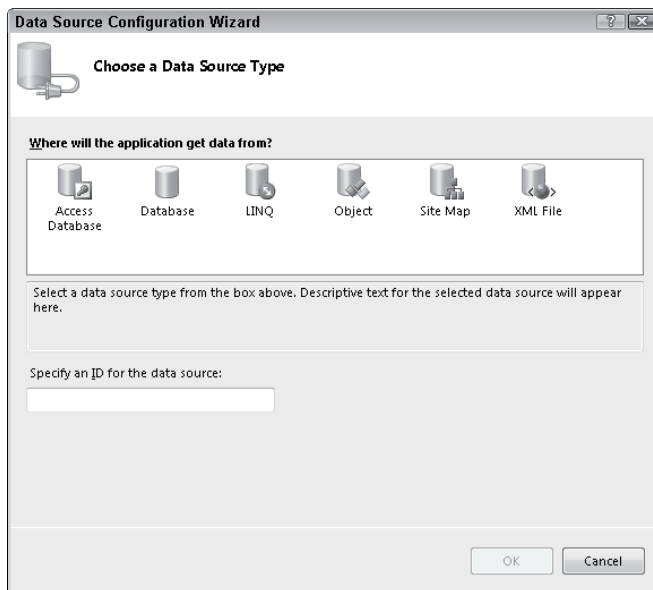
3. חזור לדף Reviews.aspx והעבר אותו ל- Design View. גרור פקד DropDownList מהקטגוריה Standard של ארגז הכלים אל הדף. בלוח Smart Tasks של הפקד בחר Enable AutoPostBack, ואז לחץ על הקישור Edit Items. הכנס פריט כשה-Text שלו מוגדר ל- Please make a selection, ואז נקה את Value שהוכנס עבורך באופן אוטומטי.

4. כשתחזור מתיבת הדו-שיח ListItem Collection Editor, הלווח Smart Tasks של הרשימה הנפתחת עדיין פתוח. לחץ על תיבת הדו-שיח Choose Data Source ומהרשימה הנפתחת שבראש המסך בחר <New data source>. כעת יופיע Data Source Configuration Wizard שמוצג בתרשים 7-12.

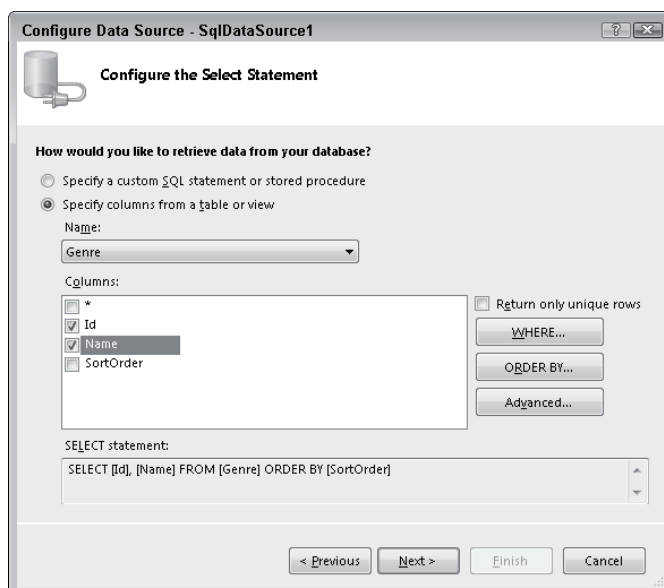
5. לחץ Database, השאר את ה-ID מוגדר ל-SqlDataSource1 ולחץ OK.

6. מהרשימה הנפתחת שבתיבת הדו-שיח שמופיעה, בחר במחרוזת החיבור שנקראת PlanetWroxConnectionString1, ולחץ Next.

7. ודא שנבחר לחצן האפשרויות עבור Specify Columns from a Table or View. ודא שמהרשימה הנפתחת עם שמות הטבלאות נבחר Genre, ובחלק Columns בחר בעמודות Id ו-Name. לחץ על הלחצן ORDER BY ומהרשימה הנפתחת Sort by, בחר SortOrder. לחץ OK. כשתסיים אמור Configure Data Source Wizard להיראות בדומה לתרשים 8-12.



תרשים 12-7



תרשים 12-8

8. לחץ Next ואחר כך Finish, כדי לגרום ל-VWD ליצור את SqlDataSource הנכון. חזור ל-Data Source Configuration Wizard עבור הרשימה הנפתחת שבה תוכל כעת להגדיר שדה שיוצג ברשימה הנפתחת של הסגנונות, ושדה ישירות כערך הבסיס ברשימה זו. בחר ב-Name מהרשימה הנפתחת שמשמשת לקביעת שדה הנתונים שיוצג ברשימה הנפתחת של רשומות הסגנונות. את הרשימה הנפתחת, שמשמשת לקביעת

הערך הבסיסי של הפריטים ברשימה הנפתחת של רשומות הסגנונות, השאר מוגדרת ל-Id. אתה אמור לסיים עם המסך שמוצג בתרשים 9-12.



תרשים 9-12

9. לחץ OK כדי לסגור את תיבת הדו-שיח ולסיים את הגדרת מקור הנתונים עבור הרשימה הנפתחת.

10. כשהפקד DropDownList עדיין נבחר, הקש F4 כדי לפתוח את טבלת המאפיינים שלו, והגדר את המאפיין AppendDataBoundItems ל-True. עבור ל- Markup View. אם המאפיין Value אינו מופיע ב- ListItem הסטטי שהוספת בצעד שלוש (המורה למשתמשים לבחור פריט), הוסף אותו באופן ידני והגדר אותו למחרוזת ריקה. הקוד הסופי אמור להיראות כך:

```
<asp:DropDownList ID="DropDownList1" runat="server" DataSourceID="SqlDataSource1"
    DataTextField="Name" DataValueField="Id" AppendDataBoundItems="true"
    AutoPostBack="True">
    <asp:ListItem Value="">Please make a selection</asp:ListItem>
</asp:DropDownList>

<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:PlanetWroxConnectionString1 %>"
    SelectCommand="SELECT [Id], [Name] FROM [Genre] ORDER BY [SortOrder]">
</asp:SqlDataSource>
```

11. שמור את כל השינויים והקש Ctrl+F5 כדי לפתוח את הדף בדפדפן. כעת תראה רשימה נפתחת שמכילה את רשומות הסגנונות מבסיס הנתונים. ברגע שתבחר בסגנון חדש מהרשימה, הדף נשלח חזרה (posts back) אל השרת. פרט לכך לא קורה דבר, מפני שלא קשרת לפקד DropDownList לוגיקה כלשהי. בתרגיל הבא תלמד כיצד לעשות זאת.

כיצד זה פועל

סיימת את התרגיל עם קוד דומה לזה ש-VWD יצרה באופן אוטומטי בתרגיל קודם, בשעה שגררת GridView אל הדף. יש לך פקד קשור-נתונים (הפקד DropDownList) שמקבל את הנתונים שלו מפקד מקור נתונים (הפקד SqlDataReader). השוני הוא בכך שהדרך שבה הגדרת את הדברים נותנת גמישות רבה יותר ביחס לקוד שנוצר. במקום לסמוך על VWD שתיצור משפט SQL עבור כל העמודות בבסיס הנתונים, באפשרותך לבחור כעת רק בשתי העמודות שאתה זקוק להן. בנוסף, מכיוון ש-SqlDataSource אינו דורש עדכונים כלשהם למקור הנתונים, הצטרפת לספק רק SelectCommand. השתמשת גם בלחצן ORDER BY כדי לשלוט בסדר שבו מוצגים הפריטים ברשימה.

כשהפקד SqlDataReader מוכן, קל למדי להציג את הנתונים שהוא מחזיר על ידי פקד DropDownList. הפנה את DropDownList למקור הנתונים הנכון על ידי שימוש במאפיין DataSourceID, והגדר את DataTextField ו-DataValueField באופן שיאמרו לפקד באילו עמודות להשתמש עבור הטקסט המוצג ובאילו עמודות להשתמש עבור הערך. על ידי הגדרת AppendDataBoundItems ל-True תוכל לשמור את הפריט שהוספת בקוד באופן ידני. כשהגדרה זו היא False, השדה שהוספת יימחק מייד ברגע שמתווספים הפריטים קשורי-הנתונים.

הצעד הבא לאחר שהמסנן מוכן הוא ליצור GridView שיציג רשומות של סקירות שקשורות לסגנונות שנבחרו. בתרגיל שלהלן תראה כיצד לעשות זאת.

נסה זאת הפעלת המסנן

בתרגיל זה תוסיף SqlDataReader שמקבל את הנתונים שלו מהטבלה Review. על ידי יצירת מסנן (הפסקה WHERE במשפט SQL) ביכולתך להגביל את מספר הפריטים שיוצגו בטבלה לאלה ששייכים לסגנון מסוים. הסגנון שנבחר ברשימה הנפתחת שיצרת בסעיף הקודם נשלח לאוסף SelectParameters של הפקד SqlDataReader תוך שימוש ב-`<asp:ControlParameter>`.

1. העבר את הדף Reviews.aspx ל-Design View. גרור GridView מארגז הכלים ושחרר אותו על פקד SqlDataReader הקיים. GridView מתווסף מייד מעליו, ונפתח לוח Smart Tasks שלו.
2. מהרשימה הנפתחת Choose Data Source בחר <New data source>. ב-Data Source Configuration Wizard, לחץ Database (בדיוק כמו באשף עבור הסגנונות שמוצג בתרשים 7-12) ולחץ OK.
3. מופיעה תיבת דו-שיח. מהרשימה הנפתחת שלה בחר במחרוזת החיבור Planet Wrox ולחץ שוב Next.
4. מהרשימה הנפתחת Name בחר בטבלה Review, וברשימה Columns לחץ כוכבית (*) כדי לבחור בכל העמודות.

5. לחץ על הלחצן WHERE שמאפשר להגדיר פסקת WHERE תוך שימוש ב-SelectParameters. בתיבת הדו-שיח שמופיעה הכנס את הפרטים כדי שהמסך יראה בסופו של דבר בדומה לתרשים 12-20.

תרשים 12-10

מסיבה כלשהי, כל אחד מהפקדים עשוי להופיע פעמיים ברשימה הנפתחת Control ID, ולא חשוב באיזו אפשרות DropDownList1 בחרת.

6. לחץ על הלחצן Add כדי להוסיף את הבחירה לרשימת הפסקה WHERE שמופיעה בתחתית המסך. לחץ OK.

7. חזור ל- Configure Data Source wizard ולחץ Next. כדי לבדוק את השאילתה לחץ על הלחצן Test Query. אם הגדרת את הפרמטרים כהלכה, תופיע תיבת דו-שיח שתאפשר לך להכניס ערך. הכנס מספר שידוע לך שהוא קיים בטבלה Genre, ולחץ OK. אם קיימות בטבלה Review רשומות ששייכות לסגנון שנבחר, הן תוצגנה בחלון של Test Query. לבסוף, לחץ Finish כדי לסיים עם האשף. אם תקבל תיבת דו-שיח עם שאלה בדבר רענון פרמטרים, לחץ Yes כדי שהקוד ב- Markup View יעודכן.

8. בדיוק כפי שעשית עם דפי הסגנונות, עטוף גם כאן את התוכן שבתוך הבלוק cpMainContent ב- UpdatePanel עם ContentTemplate. כך תבטיח שרענון הדף בזמן שייבחר סגנון חדש יהיה חלק ככל האפשר.

9. שמור את כל השינויים ופתח את הדף Reviews.aspx ברפדפן. אם תקבל שגיאה, ודא שברשימה הנפתחת הגדרת את Value של ListItem הסטטי כמחרוזת ריקה ("").

10. בחר פריט חדש מהרשימה הנפתחת. הדף מתרענן ומציג כעת את הסקירות ששייכות לסגנון שנבחר. אם הדף אינו מתרענן, ודא שבתרגיל הקודם הגדרת את המאפיין AutoPostBack ל-True עבור פקד הרשימה. בשלב זה הדף נראה מבולגן למדי מכיוון

שה-GridView מכיל עמודות רבות, שחלקן עלולות להיות רחבות מאוד (כמו העמודה Body). בתרגיל נסה זאת הבא תלמד כיצד לתקן זאת.

כיצד זה פועל

חלק גדול בתרגיל דומה ופועל כמו התרגיל הקודם שבו הצגת את רשימת הסגנונות הזמינים. השוני הפעם הוא בדרך שבה SqlDataSource מסוגל לסנן את רשומות הטבלה Review על פי הבחירה שעשית ברשימה הנפתחת. התבונן בקוד של SqlDataSource כדי להבין כיצד זה פועל:

```
<asp:SqlDataSource ID="SqlDataSource2" runat="server"
    ConnectionString="<%$ ConnectionStrings:PlanetWroxConnectionString1 %>"
    SelectCommand="SELECT * FROM [Review] WHERE ([GenreId] = @GenreId)">
    <SelectParameters>
        <asp:ControlParameter ControlID="DropDownList1" Name="GenreId"
            PropertyName="SelectedValue" Type="Int32" />
    </SelectParameters>
</asp:SqlDataSource>
```

משפט SQL עבור SelectCommand מכיל פרמטר עבור GenreId. פירוש הדבר שמשפט SQL מחזיר רק את הרשומות של הטבלה Review שמשייכות לסגנון מסוים. בזמן ביצוע מאוחזר הערך של פרמטר זה מהפקד שמוגדר באלמנט ControlParameter. בדוגמה שלנו, הקוד מוגדר באופן שיקבל את הערך מהפקד DropDownList1. כעת, VWD יודעת שכדי לקבל מ-DropDownList את הערך שנבחר, עליה לגשת למאפיין SelectedValue שלו, לפיכך היא מוסיפה אותו כ-PropertyName עבור ControlParameter. אם צריך להשתמש במאפיין אחר, שנה אותו בהכרזת האלמנט ControlParameter.

כשקוד זה מוכן, GridView מבקש מ-SqlDataSource את הנתונים שלו. מקור נתונים זה, בתורו, מבקש מ-DropDownList את הפריט שהמשתמש בחר מתוך הרשימה. ערך זה מוכנס למשפט SQL ומועבר אל בסיס הנתונים. התוצאות שמוחזרות מבסיס הנתונים נשלחות דרך מקור הנתונים חזרה ל-GridView, אשר משתמש בהן כדי ליצור טבלת HTML ברפדפן.

כשבחרת מהרשימה הנפתחת ב- Please Make a Selection Item, קיבלת דף ריק ללא רשומות. במקרה זה הפקד DropDownList מחזיר מחרוזת ריקה כערך שלו (שמוגדר במאפיין Value), מחרוזת זו מומרת ל-null, בסיס הנתונים שקול ל"כלום". דבר זה בתורו גורם שהשאלתה לא תחזיר רשומות מהטבלה Review.

עד כה סמכת על הכלים יוצרי הקוד של VWD שיגדירו עבורך את GridView ואת DetailsView. כברירת מחדל, VWD יוצרת עמודה (עבור GridView) או שדה (עבור DetailsView) עבור כל עמודה שהיא מוצאת במקור הנתונים. VWD חכמה במידה מספקת כדי לזהות כמה מסוגי הנתונים הבסיסיים במקור הנתונים. כך תקבל עמודה מסוג הפקד asp:CheckBoxField עבור שדות בוליאניים בבסיס הנתונים, אבל בכך מתמצה יכולתה. כדי להתאים עוד יותר את המראה והתחושה של פקדי נתונים אלה, עליך להתאים אישית את האוספים Columns ו-Fields שלהם.

התאמה אישית של מראה פקדי הנתונים

כברירת מחדל מציגים GridView ו-DetailsView עמודות ושורות באופן אוטומטי על פי הנתונים שהם מקבלים. לחילופין, כשתצרך פקד למקור נתונים, תוכל לגרום ל-VWD ליצור עבורך כמה שדות או עמודות. ברוב המקרים תרצה לשנות את מה שמוצג במסך, כמו למשל להקטין את מספר העמודות, לשנות את כותרות העמודות, או להציג את הנתונים על ידי פקדים אחרים. למרבה המזל, הדבר קל מאוד לביצוע על ידי עורך השדות של VWD. בסעיף הבא תלמד כיצד להשתמש בעורך זה כדי ליצור ולשנות סוגים שונים של עמודות ושדות מובנים. בסעיפים בהמשך תראה כיצד להתאים שדות אלה עוד יותר באמצעות תבניות שמוגדרות על ידי המשתמש.

קביעת תצורת עמודות או שדות של פקדים קשורי-נתונים

בתוך האלמנטים <Columns> או <Fields> של GridView ו-DetailsView תוכל להוסיף את סוגי השדות שמפורטים בטבלה הבאה.

סוג השדה	תיאור
BoundField	זהו שדה ברירת המחדל עבור רוב סוגי הנתונים הבסיסיים. במצב קריאה-בלבד הוא מרונדר כטקסט פשוט, ובמצב עריכה הוא מרונדר כ-TextBox.
ButtonField	סוג זה מרונדר כקישור או לחצן שמאפשרים לבצע פקודה בשרת.
CheckBoxField	במצב קריאה-בלבד מרונדר סוג זה כתיבת סימון לקריאה-בלבד. במצב עריכה הוא מרונדר כתיבת סימון שניתנת לעריכה.
CommandField	סוג זה מאפשר להגדיר פקודות שונות, כולל עריכה, הוספה, עדכון ומחיקה.
HyperLinkField	סוג זה מרונדר כקישור (האלמנט <a>). באפשרותך להשפיע על התנהגות HyperLink על ידי הגדרת מאפיינים כמו DataNavigateUrlFields, DataNavigateUrlFormatString ו-DataTextField. תלמד על כך יותר בתרגיל הבא.
ImageField	סוג זה מרונדר בדפדפן כאלמנט .
TemplateField	סוג זה מאפשר להגדיר בהתאם לטעמך את המראה והתחושה עבור תבניות שונות כגון, ItemTemplate, InsertItemTemplate ו-EditItemTemplate.

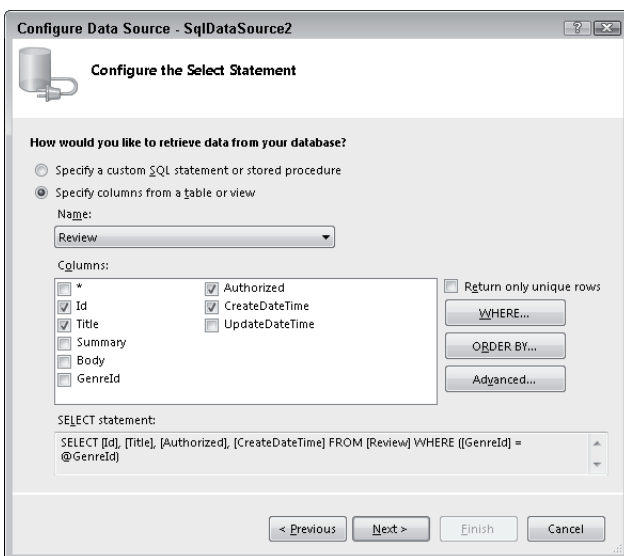
ברור שכל סוג שדה משרת מטרה מסוימת, כך שניתן לבחור בסוג שמתאים ביותר לצרכים. בתרגיל הבא תראה כמה מסוגי שדות אלה ביתר פירוט. בתרגיל זה תלמד כיצד לבצע בדף Reviews.aspx את הדברים הבאים:

- ⊙ להשתמש בעורך השדות כדי להתאים את השדות של GridView להצגת סקירות.
- ⊙ להשתמש בעמודה HyperLink כדי ליצור קישור לדף פרטים שיאפשר לנהל את פרטי הסקירה.

- ⊙ לעצב את הפלט של עמודות BoundField קיימות.
 - ⊙ להשתמש בפונקציה מותאמת אישית שנמצאת בקוד ברקע, כדי לשלוט באופן מלא בפלט שב-TemplateField.
- תרגיל מאוחר יותר יראה כיצד ליצור דף פרטים כדי להכניס סקירות חדשות ולערוך סקירות קיימות.

נסה זאת התאמה אישית של עמודות GridView

1. בדף Reviews.aspx, פתח את לוח Smart Tasks עבור הפקד SqlDataSource2 ולחץ על Configure Data Source. לחץ Next כדי לדלג על מסך מחרוזת החיבור, ואז השלם את המסך בהתאם למוצג בתרשים 11-12, על ידי בחירה בעמודות Title, Id, Authorized ו-CreateDateTime מהטבלה Review.

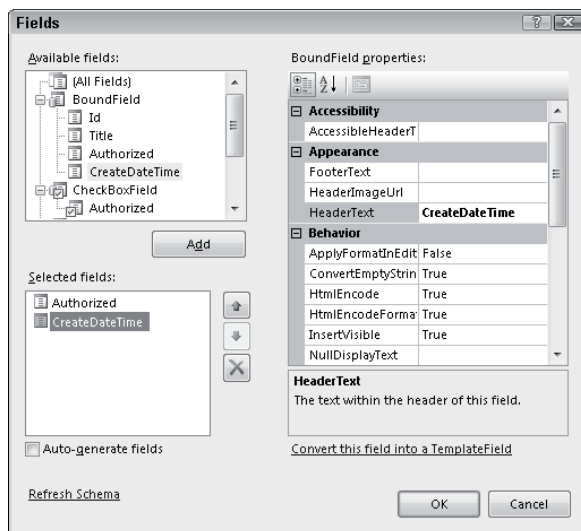


תרשים 11-12

לחץ על Advanced וגרום ל-VWD ליצור פקודות עבור משפטי UPDATE, INSERT ו-DELETE. אתה יכול להשאיר את תיבת הסימון Optimistic Concurrency נקייה (התיבה עוסקת בגילוי שינויים שנערכו ברשומה מאז שנטענה בפעם האחרונה ממקור הנתונים). לחץ OK כדי לסגור את תיבת הדו-שיח Advanced SQL Generation Options. לחץ Next ולבסוף לחץ Finish כדי לעדכן את משפט SQL במקור של הדף. כשתישאל אם ברצונך לאפס את השדות והמפתחות של GridView, לחץ Yes.

2. בשלב זה VWD עדיין לא יצרה ב-Markup View את העמודות עבור GridView. כדי לעשות זאת, פתח את לוח Smart Tasks של GridView ולחץ Edit Columns. הדבר מעלה את תיבת הדו-שיח Fields. אם הרשימה Selected Fields כבר מכילה פריטים, השתמש בלחצן Delete (זה בעל X גדול ואדום) כדי לנקות את הרשימה תחילה.

3. ברשימה Available Fields, בחר Authorized ואז לחץ על הלחצן Add כדי להעתיק את הפריט לרשימה Selected Fields. חזור על צעד זה גם עבור השדה CreateDateTime. תיבת הדו-שיח נראית כעת דומה לתרשים 12-12.



תרשים 12-12

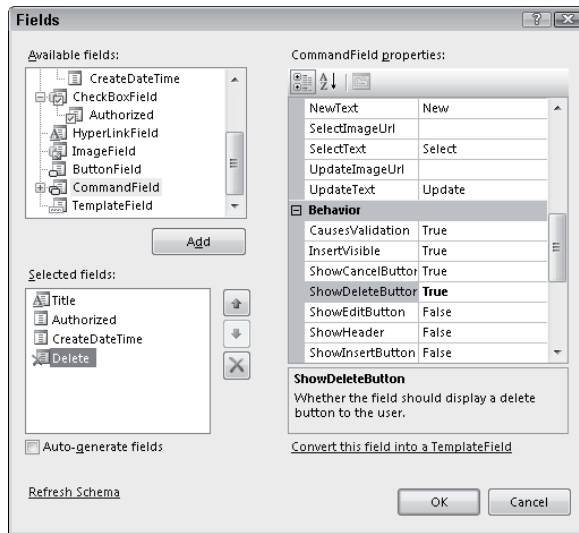
4. ברשימה Available Fields שבראש המסך בחר HyperLinkField, ואז לחץ על הלחצן Add כדי להוסיף את הפריט גם לרשימה Selected Fields. העבר את HyperLinkField לראש הרשימה על ידי לחיצה פעמיים על הלחצן שמסומן עליו חץ מכוון מעלה. תוך שימוש בטבלת המאפיינים שמימין, הגדר את המאפיינים הבאים של HyperLinkField.

מאפיין	קבע ערך זה
HeaderText	Title
DataNavigateUrlFields	Id
DataNavigateUrlFormatString	AddEditReview.aspx?Id={0}
DataTextField	Title

5. ברשימה עם Available Fields, לחץ על CommandField ולחץ שוב על הלחצן Add. תוך שימוש בטבלת המאפיינים, הגדר את HeaderText של הפריט שזה עתה הכנסת ל-Delete ואת ShowDeleteButton הגדר ל-True. מאוחר יותר הדבר יאפשר לך למחוק סקירות מבסיס הנתונים על ידי GridView.

תיבת הדו-שיח Fields אמורה להיראות כעת דומה לתרשים 12-13.

6. ברשימה Selected Fields, לחץ על העמודה Authorized, ואז לחץ על הקישור הכחול Convert This Field into a TemplateField שבתחתית תיבת הדו-שיח (מצד ימין).



תרשים 12-13

7. לחץ על העמודה CreateDateTime שמשמאל, והגדר את DataFormatString ל-{0:g}.
8. לחץ OK כדי להחיל את השינויים על קוד המקור.
9. עבור אל Markup View והסר את <EditItemTemplate> עבור השדה Authorized. כעת GridView מציג סקירות במצב קריאה-בלבד, ולכן אינך זקוק לתבנית זו.
10. שנה את הפקד Label שב-ItemTemplate של השדה Authorized באופן שייראה לבסוף כך:

```
<asp:Label ID="lblAuthorized" runat="server"
    Text='<%# GetBooleanText(Eval("Authorized")) %>' />
```

11. עבור לקוד ברקע של הדף, והוסף את הפונקציה הבאה שמחזירה את הטקסט Yes או No בהתאם לערך הבוליאני שאתה מעביר לראש קובץ המחלקה (ב-VB.NET - מייד לאחר השורה Inherits, ב-C# - לאחר סוגר הפתיחה המסולסל).

VB.NET

```
Protected Function GetBooleanText(ByVal booleanValue As Object) As String
    Dim authorized As Boolean = CType(booleanValue, Boolean)
    If authorized Then
        Return "Yes"
    Else
        Return "No"
    End If
End Function
```

C#

```
protected string GetBooleanText(object booleanValue)
{
```

```

bool authorized = (bool)booleanValue;
if (authorized)
{
    return "Yes";
}
else
{
    return "No";
}
}

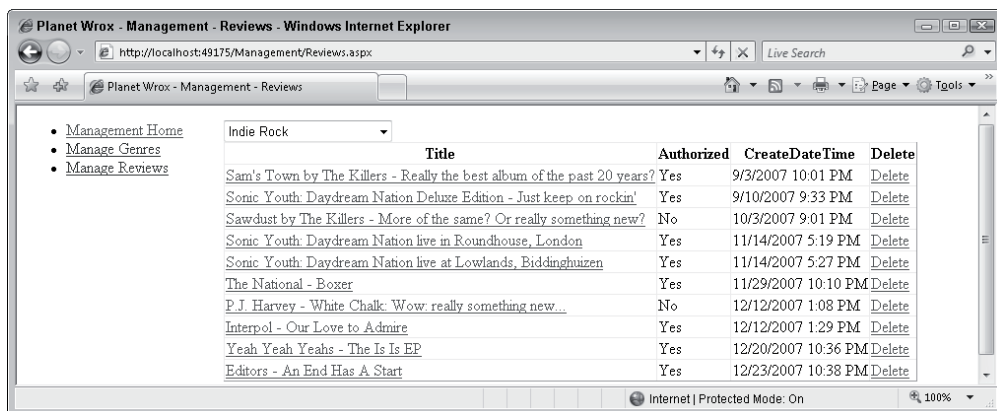
```

12. שמור את כל השינויים (הקש Ctrl+Shift+S) ואז הקש Ctrl+F5 כדי לפתוח את הדף Reviews.aspx בדפדפן. מהרשימה הנפתחת בחר בסגנון ותראה שמופיעה רשימת סקירות. שים לב שהעמודה Authorized מציגה כעת את הטקסט Yes או No. העמודה CreateDateTime מציגה את התאריך והשעה בצורה מקוצרת. תרשים 12-14 מציג את התוצאה עבור הסגנון Indie Rock.

שים לב שהלינק בעמודה הראשונה של פקד GridView מפנה כעת לדף ID-וה של הסקירה מועבר בשדה Id של מחרוזת השאילתה.

http://localhost:49394/Management/AddEditReview.aspx?Id=10

בהמשך הפרק תיצור את הדף הזה. בפרק 14 תלמד על נוהל המחיקה של הדף Reviews מהמערכת.



תרשים 12-14

כיצד זה פועל

התחלת עם שינוי SelectCommand של SqlDataReader. במקום לבחור בכל העמודות על ידי SELECT *, משפט SQL מכיל עכשיו קבוצה חלקית של העמודות, והדבר מזרז במקצת את טעינת הדף:

```

<asp:SqlDataSource ID="SqlDataSource2" runat="server"
    ConnectionString="<%$ ConnectionStrings:PlanetWroxConnectionString1 %>"

```

```
SelectCommand="SELECT [Id], [Title], [Authorized], [CreateDateTime]
FROM [Review] WHERE ([GenreId] = @GenreId)">
```

```
...
</asp:SqlDataSource>
```

אחר כך השתמשת בתיבת הדו-שיח Fields כדי לשנות את השדות השונים שמוצגים על ידי GridView. יצרת עמודה Title תוך שימוש ב-HyperLinkField:

```
<asp:HyperLinkField
    DataNavigateUrlFields="Id" DataTextField="Title"
    DataNavigateUrlFormatString="AddEditReview.aspx?Id={0}" HeaderText="Title">
</asp:HyperLinkField>
```

השדות DataNavigateUrlFields מכילים רשימה של שדות, מופרדים בפסיק, שברצונך להשתמש בהם במאפיין DataNavigateUrlFormatString. במקרה שלנו השתמשנו בשדה אחד בלבד. כדי להציג את הערך של שדה זה, השתמשת במאפיין DataNavigateUrlFormatString בשומרי מקום כמו {0}. לדוגמה, סקירה בעלת ID של 10 תיצור עמודת HyperLink בעלת NavigateUrl: AddEditReview.aspx?Id=10. עם הגדרה כזו, {0} מוחלף בערך של השדה הראשון של המאפיין DataNavigateUrlFormatString. אם הגדרת שדות נוספים, מופרדים על ידי פסיק, תיגש אליהם על ידי {1}, {2} וכו'.

השדה DataTextField מוגדר לעמודה Title. הדבר גורם ל-HyperLink לרנדר את המאפיין Text שלו עם כותרת הסקירה, כפי שמוצג בתרשים 12-14.

כמו כן, הגדרת את המאפיין DataFormatString של BoundField לעמודה CreateDateTime.

```
<asp:BoundField
    DataField="CreateDateTime" HeaderText="CreateDateTime"
    SortExpression="CreateDateTime" DataFormatString="{0:g}">
</asp:BoundField>
```

המחרוזת DataFormatString מאפשרת להגדיר את העיצוב שבו יוצגו הנתונים הבסיסיים. במקרה שלנו משמשת האות הקטנה g להצגת התאריך והשעה בצורה מקוצרת (ללא שניות). מידע נוסף אודות מחרוזות העיצוב השונות תוכל למצוא בתיעוד MSDN ב-

[http://msdn2.microsoft.com/en-us/library/97x6twsz\(VS.90\).aspx](http://msdn2.microsoft.com/en-us/library/97x6twsz(VS.90).aspx)

אחר כך המרת את העמודה Authorized בעמודת תבנית. עמודת תבנית מאפשרת חופש מלא וגמישות רבה ביחס לתוכן שיוצג. בעצם, תוכל להוסיף כמעט כל דבר שתמצא לנכון כתוכן לעמודה, כולל קוד HTML ופקדי ASPX. בתרגיל הקודם שינית את Label כדי שיקבל את הטקסט שלו מפונקציה שהותאמה אישית, תוך שימוש בתחביר של ביטוי קשור-נתונים <%= %>.

```
<asp:Label ID="lblAuthorized" runat="server"
    Text='<%= GetBooleanText(Eval("Authorized")) %>'></asp:Label>
```

כדי שקוד זה יפעל, השתמשנו כאן בשני דברים. הבט תחילה במשפט Eval("Authorized"). הוא נקרא **ביטוי קשור-נתונים חד-סטרי** (a one-way data binding expression), ותוצאתו היא העברת הערך של העמודה Authorized כאובייקט לשיטה המותאמת אישית GetBooleanText.

שיטה זו בתורה ממירה ערך בוליאני המתקבל כפרמטר שנכנס ומחזירה Yes או No בהתאם לערך של העמודה Authorized שבבסיס הנתונים. זו רק דוגמה פשוטה שבאה להדגים כיצד לקרוא לשיטה מותאמת אישית, שנמצאת בקוד ברקע במשך קשירת נתונים. עם זאת, העיקרון נשאר בעינו גם עבור שיטות מורכבות יותר. מעבירים ארגומנט אחד, או יותר, לשיטה בקוד ברקע תוך שימוש ב- Eval("ColumnName"). השיטה שנמצאת בקוד ברקע מקבלת את הארגומנטים האלה כאובייקטים, מעצבת אותם לסוג המתאים, ואחר כך משתמשת בהם בהתאם. בסוף, השיטה יכולה להחזיר מחרוזת עם טקסט או עם קוד HTML כלשהם.

הערך HyperLink שהגדרת עבור העמודה Title מצביע על הדף AddEditReview.aspx. דף זה מאפשר ליצור סקירות חדשות ולעדכן סקירות קיימות. בסעיף הבא תלמד כיצד ליצור דף זה.

עדכון והכנסת נתונים

מוקדם יותר בפרק זה עסקנו בעדכונים פשוטים באמצעות הפקדים GridView ו-1 SqlDataSource. למרות שהתנהגות עדכון מובנה זו מספקת במצבים רבים, הרי שלא תמיד היא מספיק מקיפה כדי לענות על כל הדרישות.

למרבה המזל, מאפשרים פקדים כמו FormView ו-1 DetailsView להתאים את המראה והתחושה שלהם, והדבר מקנה גמישות רבה יותר לדרך עבודת משתמשי הקצה עם הנתונים שלהם. בסעיף הבא תלמד כיצד להשתמש ב-DetailsView כדי לתת למשתמש ממשק קל הרבה יותר לעריכת סקירות ולהכנסתן לבסיס הנתונים.

שימוש ב-DetailsView להכנסת נתונים ועדכון

מוקדם יותר בפרק למדת כיצד להגדיר פקד DetailsView פשוט ולסמוך לחלוטין על VWD ועל הפקד עצמו שירגורו את ממשק המשתמש הרלוונטי בדפדפן. ללא ספק, לעתים קרובות אין התנהגות ברירת מחדל זו מספקת. מה יקרה אם תרצה להשפיע על הפקדים שמשתמשים בהם בממשק? לדוגמה, מה היה קורה אילו רצית להשתמש עבור הסגנונות ב-DropDownList במקום ב-TextBox. ומה אילו רצית להוסיף עוד פקדי בדיקת תקפות שלמדת עליהם בפרק 9? ומה אם רצית לנהל בתכנות חלק מהנתונים שנשלחים לבסיס הנתונים? כל הדברים האלה אפשריים עם הפקד DetailsView, עם עמודותיו מבוססות-התבנית והמספר הרב של האירועים שהוא מעורר בשלבים שונים של מחזור חיו.

את העמודות מבוססות-התבנית תראה בתרגיל הבא. עם זאת, עליך להכיר תחילה את האירועים השונים שמעוררים הפקדים קשורי-הנתונים ופקדי מקור-הנתונים. הטבלה הבאה מפרטת כמה מהאירועים שחושפים ומעוררים הפקדים DetailsView, FormView ו-1 ListView במהלך חייהם. ל-GridView יש אירועים דומים, אבל הם מתחילים במילה Row ולא במילה Item. מכיוון

שהפקדים Repeater ו- DataList אינם תומכים באופן טבעי בעריכת נתונים, אין להם אף לא אחד מהאירועים האלה.

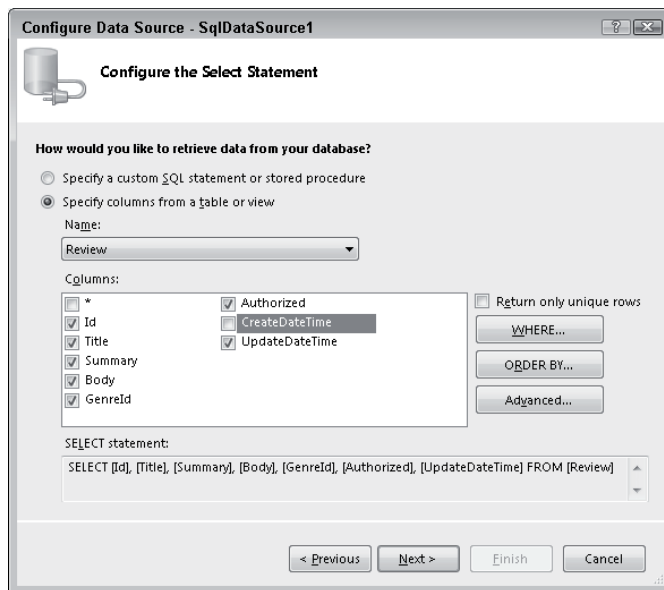
אירוע	תיאור
ItemInserting	אירוע זה מתעורר ממש לפני שמבוצעת הפקודה Insert כנגד מקור הנתונים. זהו מקום אידיאלי לשינוי הנתונים שעומדים להישלח לבסיס הנתונים.
ItemInserted	אירוע זה מתעורר מייד לאחר שבוצעה הפקודה Insert כנגד מקור הנתונים.
ItemUpdating	אירוע זה מתעורר ממש לפני שמבוצעת הפקודה Update כנגד מקור הנתונים. מקום זה הוא המתאים ביותר לשינוי הנתונים שעומדים להישלח לבסיס הנתונים.
ItemUpdated	אירוע זה מתעורר מייד לאחר שבוצעה הפקודה Update כנגד מקור הנתונים.
ItemDeleting	אירוע זה מתעורר ממש לפני שמבוצעת הפקודה Delete כנגד מקור הנתונים.
ItemDeleted	אירוע זה מתעורר מייד לאחר שבוצעה הפקודה Delete כנגד מקור הנתונים.

ששה אירועים אלה מתעוררים ברגעים מאוד נוחים בחיי הפקד: מייד לפני ומייד אחרי שהנתונים עבור הפעולה נשלחים אל בסיס הנתונים. בתרגיל נסה זאת שלהלן תלמד כיצד להשתמש בהם.

נסה זאת ניהול נתונים על ידי הפקד DetailsView

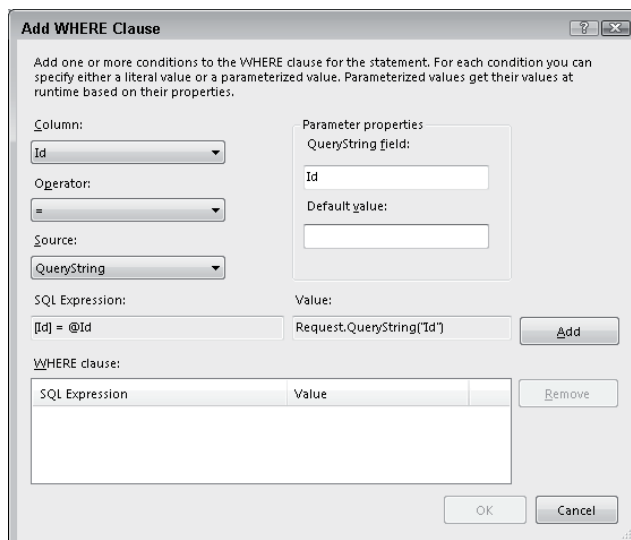
בתרגיל זה תיצור את הדף AddEditReview.aspx שכבר יצרת אליו קישור מהדף Reviews. בדף החדש תיצור פקד DetailsView, ותתאים לפי רצונך את רוב השדות שלו, על ידי יישום שדות תבנית. לאחר מכן תטפל בכמה מהאירועים של הפקד כדי לשנות את התנהגותו. כשתסיים יהיה בידך כל מה שנחוץ כדי ליצור, לרשום, לעדכן ולמחוק סקירות באתר.

1. בתיקייה Management, צור טופס אינטרנט חדש בשם AddEditReview.aspx. גם במקרה זה, בסס אותו על הדף הראשי של החלק Management ובחר את שפת התכנות שלך. קבע לדף את הכותרת Planet Wrox - Management - Insert and Update Reviews.
 2. העבר את הדף ל- Design View ושחרר פקד DetailsView על הדף. בלוח Smart Tasks שנפתח אוטומטית בחר <New data source> מהרשימה הנפתחת Choose Data Source. לחץ על הסמל Database ואז לחץ OK לאישור. מהרשימה הנפתחת בתיבת הדו-שיח שמופיעה, בחר במחרוזת החיבור ולחץ Next.
 3. הכנס את הפרטים כפי שהם מוצגים בתרשים 12-15.
- שים לב שכל השדות של Review נבחרים במפורש, פרט לשדה CreateDateTime.



תרשים 12-15

4. לחץ על הלחצן WHERE כדי להגדיר SelectParameter שיאחזר את ה-ID של סקירה ממחרוזת השאילתה. ההגדרה מתבצעת על ידי השלמת תיבת הדו-שיח כפי שמוצג בתרשים 12-16. אל תשכח להקליד **Id** בשדה QueryString.
5. לחץ על הלחצן Add כדי להוסיף את הפרמטר, ולחץ OK כדי לסגור את תיבת הדו-שיח.

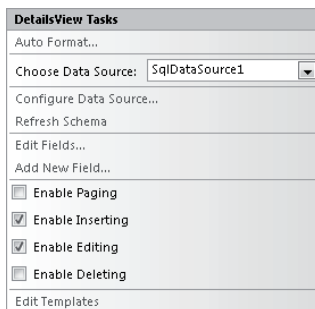


תרשים 12-16

6. חזור לאשף Configure Data Source (מוצג בתרשים 12-15), לחץ על הלחצן Advanced, בחר באפשרות ליצור משפטי INSERT, UPDATE ו-DELETE, ולחץ OK כדי

לסגור את תיבת הדו-שיח. לבסוף, לחץ Next ואחר כך לחץ Finish כדי לסיים עם אשף מקור הנתונים.

7. בלוח Smart Tasks של DetailsView, שמוצג בתרשים 12-17, בחר באפשרויות עבור Editing-1 Inserting.



תרשים 12-17

8. בטבלת המאפיינים של DetailsView, הגדר את DefaultMode ל-Insert.

9. ב- Design View, לחץ לחיצה כפולה במקום ריק בדף כדי להגדיר את מטפל האירוע Page_Load, והכנס את הקוד הבא:

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Me.Load
```

```
    If Request.QueryString.Get("Id") IsNot Nothing Then  
        DetailsView1.DefaultMode = DetailsViewMode.Edit  
    End If
```

```
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
```

```
{
```

```
    if (Request.QueryString.Get("Id") != null)  
    {  
        DetailsView1.DefaultMode = DetailsViewMode.Edit;
```

```
    }
```

```
}
```

10. פתח את הדף Reviews.aspx ב- Design View. גרור אליו את הדף AddEditReview.aspx מסייר הפתרון, ושחרר אותו מתחת ל-GridView. פעולה זו יוצרת קישור אל דף זה, כדי שתוכל להכניס סקירות חדשות. עבור ל-Markup View ושנה את הטקסט שבין התגיות <a> ל- Insert New Review:

```
<a href="AddEditReview.aspx">Insert New Review</a>
```


11. שמור את כל השינויים ופתח את הדף AddEditReview.aspx בדפדפן. אתה אמור לראות את עיצוב ברירת המחדל של הפקד, עם תיבות טקסט פשוטות עבור כל העמודות במקור הנתונים. אם תקבל מסך ריק, ודא שהגדרת את DefaultMode ל-Insert. מלא את השדות בהתאם למוצג בתרשים 12-18 (סדר הופעת השדות בדף שלך עשוי להיות שונה במקצת).

תרשים 12-18

ודא שה-GenreId שאתה מכניס תואם לאחד מהסגנונות בטבלה Genre שבבסיס הנתונים, אחרת תקבל שגיאה כשתנסה להכניס את הפריט. ודא גם שהכנסת תאריך תקף.

לחץ Insert כדי להכניס את הפריט לבסיס הנתונים. בתחילה לא נראה שקורה הרבה, אבל אם תעקוב אחר שני הצעדים הבאים תוכל לאתר את הסקירה החדשה באמצעות :Reviews.aspx

⊙ בתפריט שמשמאל לחץ על הקישור Manage Reviews.

⊙ מהרשימה הנפתחת שבראש הדף בחר את הסגנון הנכון. אם אתה משתמש בבסיס הנתונים שמגיע עם החומר הנטען של פרק זה, וכשהכנסת את הסקירה קבעת את GenreId ל-1, הסגנון הוא Rap and Hip-Hop.

כשתמצא את הסקירה שלך, אתה יכול ללחוץ על כותרתה ותועבר אל AddEditReview.aspx. במקום זה תוכל לשוב ולשנות את פרטי הסקירה.

כיצד זה פועל

רוב התרגיל הזה צריך להיות מוכר לך עכשיו. פעולת DetailsView להכנסת רשומה חדשה זהה לפעולתו עם הסגנונות, שכבר ראית קודם. השוני הוא בטיפול בעדכונים. הקוד שנמצא בחלק קוד ברקע מסתכל במחרוזת השאילתה, ואם הוא מוצא Id כפרמטר של מחרוזת השאילתה, הוא מעביר את DetailsView למצב Edit.

VB.NET

```
If Request.QueryString.Get("Id") IsNot Nothing Then
    DetailsView1.DefaultMode = DetailsViewMode.Edit
End If
```

C#

```
if (Request.QueryString.Get("Id") != null)
{
    DetailsView1.DefaultMode = DetailsViewMode.Edit;
}
```

כשהפקד במצב Edit, הוא יודע מה לעשות. הוא קורא ל-`SqlDataSource` ומבקש ממנו את הנתונים שלו. ה-`SqlDataSource` בתורו מאחזר את ה-ID של הסקירה ממחרוזת השאילתה, ניגש לבסיס הנתונים, ומחזיר את הסקירה הנכונה אשר מוצגת אז בדף. אחר כך, כשאתה לוחץ על הקישור Update, מעורר `SqlDataSource` את `UpdateCommand` שלו כדי לשלוח את השינויים אל בסיס הנתונים.

תרגיל זה מהווה בסיס טוב לתרגיל הבא, שבו תרחיב את `DetailsView` על ידי יישום תבניות מותאמות אישית עם פקדי אימות ובדיקת תוקף, ותגדיר מטפלי אירוע שונים שיגיבו על אירועי הפקד.

נסה זאת התאמה אישית של DetailsView וטיפול באירועים שלו

הדף עם `DetailsView` נראה ברגע זה משעמם למדי. הוא ייראה טוב הרבה יותר ויהיה קל יותר לשימוש אם תהיינה לו התכונות הבאות:

- ⊙ אזור טקסט רחב בו תוכל להזין מלל במקום תיבת טקסט בת שורה אחת עבור השדות Summary ו-Body.
- ⊙ רשימה נפתחת עבור הסגנונות, כשמפורטים בה הסגנונות הזמינים מבסיס הנתונים.
- ⊙ עדכון אוטומטי של העמודה UpdateDateTime.
- ⊙ פקדי אימות שימנעו מהמשתמש להשאיר את השדות הדרושים ללא תוכן.
- ⊙ הכוונה אוטומטית אל הדף Reviews.aspx לאחר שפריט הוכנס או עודכן.

תרגיל זה מראה כיצד ליישם את כל התכונות האלו.

1. ודא ש-`AddEditReview.aspx` נמצא ב-`Design View`. בלוח `Smart Tasks` של `DetailsView`, לחץ על `Edit Fields` כדי להעלות את עורך השדות. אתר את העמודה `UpdateDateTime` והגדר את המאפיין `Visible` שלה ל-`False`.
2. ברשימה `Selected Fields`, לחץ על העמודה `Title` ואחר כך לחץ על הקישור הכחול עם הטקסט `Convert This Field into a TemplateField`. חזור על כך עבור השדות `Summary`, `Body` ו-`GenreId`. לחץ OK כדי לסגור את תיבת הדו-שיח `Fields`.
3. עבור ל-`Markup View` של הדף ועבור ארבעת פקדי תיבות הטקסט הוסף את המאפיין `TextMode` כשהערך שלו מוגדר ל-`MultiLine`, עבור השדות `Summary` ו-`Body`. בנוסף,

הגדר את המאפיינים Width ו-Height שלהם ל-500 ו-100 פיקסל, בהתאמה. ודא שביצעת זאת הן עבור EditItemTemplate והן עבור InsertItemTemplate. אתה אמור לסיים עם הקוד הבא שמוצג בשדה Summary:

```
<asp:TemplateField HeaderText="Summary" SortExpression="Summary">
  <ItemTemplate>
    <asp:Label ID="Label2" runat="server" Text='<## Bind("Summary") %>'></asp:Label>
  </ItemTemplate>
  <EditItemTemplate>
    <asp:TextBox ID="TextBox2"
      TextMode="MultiLine" Width="500" Height="100" runat="server"
      Text='<## Bind("Summary") %>'></asp:TextBox>
  </EditItemTemplate>
  <InsertItemTemplate>
    <asp:TextBox ID="TextBox2"
      TextMode="MultiLine" Width="500" Height="100" runat="server"
      Text='<## Bind("Summary") %>'></asp:TextBox>
  </InsertItemTemplate>
</asp:TemplateField>
```

4. הוסף RequiredFieldValidator הן לתבנית EditItemTemplate והן לתבנית InsertItemTemplate של השורות Title ו-Summary. באפשרותך להכניס את הקוד ישירות, או לגרור אותו מארגז הכלים ולשחררו ב- Markup View. קשר את הפקדים אל פקדי TextBox שבתבנית על ידי הגדרת המאפיין ControlToValidate, וספק הודעת שגיאה שימושית. קבע לכל ארבעת הפקדים RequiredFieldValidator שמות ייחודיים על ידי הגדרת ID שלהם ל-reqVal1, reqVal2, reqVal3 ו-reqVal4.

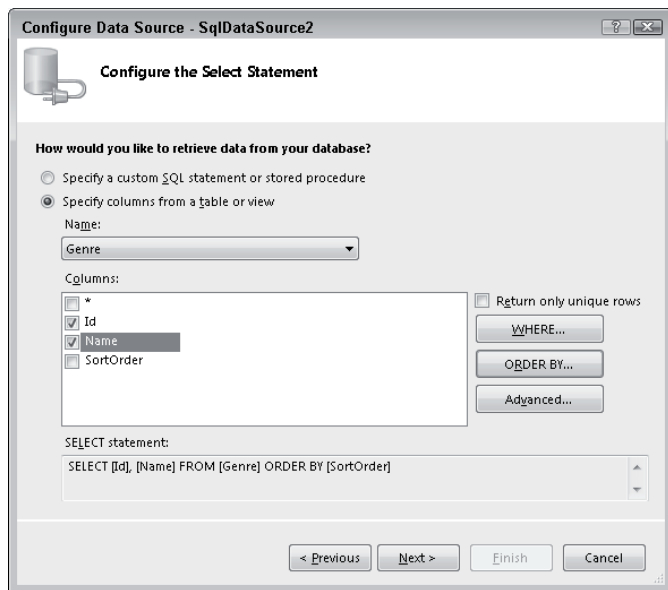
☉ כשתסיים, השדה Summary אמור להיראות כך:

```
<asp:TemplateField HeaderText="Summary" SortExpression="Summary">
  <ItemTemplate>
    <asp:Label ID="Label2" runat="server" Text='<## Bind("Summary") %>'></asp:Label>
  </ItemTemplate>
  <EditItemTemplate>
    <asp:TextBox ID="TextBox2" TextMode="MultiLine"
      Width="500" Height="100" runat="server"
      Text='<## Bind("Summary") %>'></asp:TextBox>
    <asp:RequiredFieldValidator ID="reqVal3" ControlToValidate="TextBox2"
      runat="server" ErrorMessage="Please enter a summary">
    </asp:RequiredFieldValidator>
  </EditItemTemplate>
  <InsertItemTemplate>
    <asp:TextBox ID="TextBox2" TextMode="MultiLine"
      Width="500" Height="100" runat="server"
      Text='<## Bind("Summary") %>'></asp:TextBox>
    <asp:RequiredFieldValidator ID="reqVal4" ControlToValidate="TextBox2">
```

```
runat="server" ErrorMessage="Please enter a summary">
</asp:RequiredFieldValidator>
</InsertItemTemplate>
</asp:templatefield>
```

⊙ השדות Title ו-Body אמורים להיראות דומים לזה. המאפיינים TextMode, Width ו-Height לא הוחלו על TextBox של השדה Title, בעוד שבשדה Body לא נמצא RequiredFieldValidator. פרט לכך השדות צריכים להיראות דומים למדי לשדה Summary.

5. עבור ל- Design View של הדף, גרור פקד SqlDataReader חדש ושחרר אותו סמוך ל-SqlDataSource1 שכבר נמצא בדף. פתח את לוח Smart Tasks שלו ולחץ על Configure Data Source. מהרשימה הנפתחת בחר במחרוזת החיבור Planet Wrox ולחץ Next. מהטבלה Genre בחר בעמודות Id ו-Name לחץ על הלחצן ORDER BY ובחר SortOrder מהרשימה הנפתחת Sort by. כשתלחץ OK ייראה המסך Configure Data Source בדומה לתרשים 12-19.



תרשים 12-19

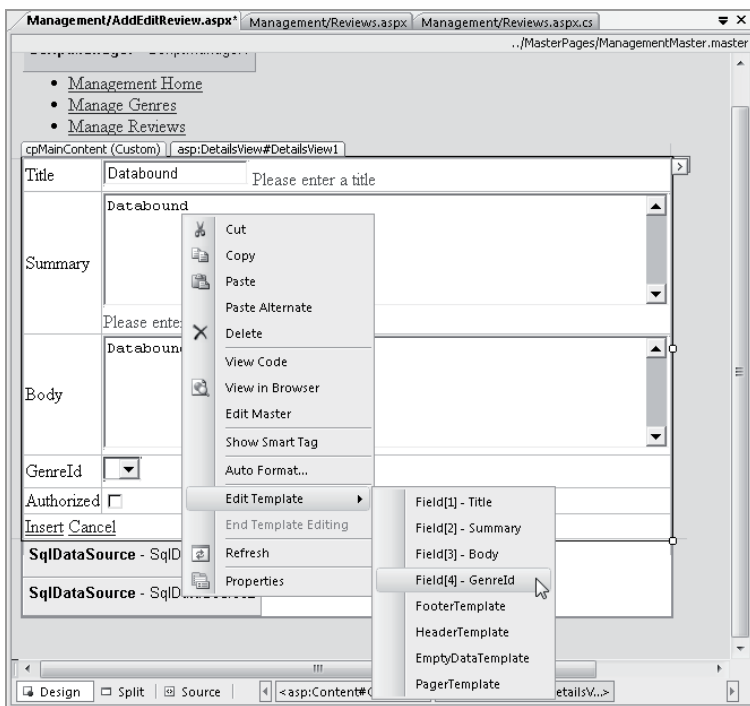
- 6. לחץ Next ואחר כך Finish כדי לסיים עם האשף Configure Data Source.
- 7. ב- Design View, בחר ב-SqlDataSource החדש (נקרא SqlDataReader2) ושנה את ה-ID שלו ל-sdsGenres על ידי טבלת המאפיינים.
- 8. עבור ל- Markup View. אתר את InsertItemTemplate עבור GenreId של הפקד DetailsView והסר את כל התוכן (הפקד TextBox). במקום שממנו הסרת את TextBox הוסף DropDownList על ידי גרירתו מארגז הכלים. הקוד ייראה כמו זה:

```

<asp:DetailsView ID="DetailsView1" runat="server"
    AutoGenerateRows="False" DataKeyNames="Id"
    DataSourceID="SqlDataSource1" DefaultMode="Insert"
    Height="50px" Width="125px">
<Fields>
...
<asp:TemplateField HeaderText="GenreId" SortExpression="GenreId">
...
<EditItemTemplate>
    <asp:TextBox ID="TextBox4" runat="server"
        Text='<%# Bind("GenreId") %>'></asp:TextBox>
</EditItemTemplate>
<InsertItemTemplate>
    <asp:DropDownList ID="DropDownList1" runat="server">
    </asp:DropDownList>
</InsertItemTemplate>
...
</Fields>
</asp:DetailsView>

```

9. עבור ל- Design View. לחץ על DetailsView פעם אחת כדי לבחור בו, ואז לחץ על DetailsView לחיצה ימנית ובחר Edit Template ← GenreId - Field[4], כפי שמוצג בתרשים 12-20.



תרשים 12-20

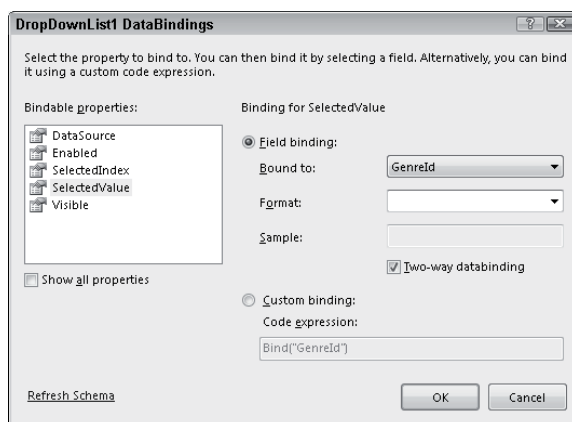


תרשים 12-21

10. כשהפקד נמצא במצב עריכה לפי תבניות, ביכולתך לגשת ישירות אל DropDownList. פתח את הלוח Smart Tasks של DropDownList ובחר Choose Data Source. ב- Data Source Configuration Wizard, בחר sdsGenres מהרשימה הנפתחת של מקור הנתונים, ואת Name ו-Id בחר משתי הרשימות הנפתחות האחרות (ראה תרשים 12-21).

11. לחץ OK כדי לסגור את Data Source Configuration Wizard.

12. חזור ללוח Smart Tasks של הפקד DropDownList ולחץ Edit DataBindings. בתיבת הדו-שיח שמופיעה, לחץ SelectedValue ברשימה שמשמאל, ואז בחר GenreId מהרשימה הנפתחת Bound To שמימין, כפי שמוצג בתרשים 12-22.



תרשים 12-22

13. לחץ OK כדי לסגור את תיבת הדו-שיח. הקוד עבור InsertItemTemplate נראה כעת
ב- Markup View בדומה לזה:

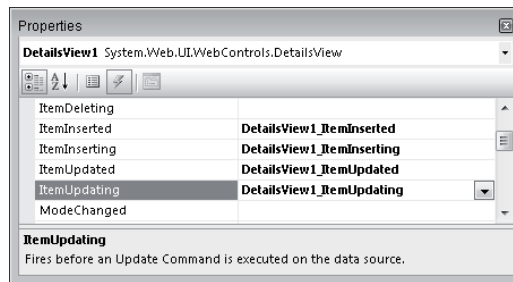
```
<InsertItemTemplate>
  <asp:DropDownList ID="DropDownList1" runat="server" DataSourceID="sdsGenres"
    DataTextField="Name" DataValueField="Id"
    SelectedValue='<%# Bind("GenreId") %>'>
  </asp:DropDownList>
</InsertItemTemplate>
```

14. העתק את התוכן של InsertItemTemplate (הקוד המוצגלל בצעד הקודם), והדבק אותו
ב-EditItemTemplate תוך כתיבה על הפקד TextBox הקיים. הדבר מוסיף את אותה
רשימה נפתחת ל-DetailsView במצב Edit.

15. חזור אל Design View, לחץ על DetailsView, ואז הקש F4 כדי לפתוח את טבלת
המאפיינים. עבור לקטגוריה Events, ולחץ לחיצה כפולה על האירועים שלהלן. בכל
פעם שאתה לוחץ על אירוע, VWD עוברת לקוד ברקע של הדף. על כן, כדי להוסיף את
שאר האירועים עליך לחזור לדף (תוך שימוש ב- Ctrl+Tab).

- ⊙ ItemInserted
- ⊙ ItemInserting
- ⊙ ItemUpdated
- ⊙ ItemUpdating

כשתסיים, אמורה הקטגוריה Events של רשת המאפיינים להיראות כמו תרשים 12-23.



תרשים 12-23

16. גש לקוד ברקע ושנה את הקוד בהתאם לדוגמה שלהלן. שים לב שהמטפלים ItemInserted
ו-ItemUpdated קוראים לשיטה ItemUpdated (שגם אותה עליך להוסיף לקוד), בעוד
שהשיטה ItemInserting וגם ItemUpdating מגדירות את הערך UpdateDateTime.

VB.NET

```
Protected Sub DetailsView1_ItemInserted(ByVal sender As Object, _
    ByVal e As System.Web.UI.WebControls.DetailsViewInsertedEventArgs) _
    Handles DetailsView1.ItemInserted
    EndEditing()
End Sub
```

```

Protected Sub DetailsView1_ItemInserting(ByVal sender As Object, _
    ByVal e As System.Web.UI.WebControls.DetailsViewInsertEventArgs) _
    Handles DetailsView1.ItemInserting
    e.Values("UpdateDateTime") = DateTime.Now
End Sub

Protected Sub DetailsView1_ItemUpdated(ByVal sender As Object, _
    ByVal e As System.Web.UI.WebControls.DetailsViewUpdatedEventArgs) _
    Handles DetailsView1.ItemUpdated
    EndEditing()
End Sub

Protected Sub DetailsView1_ItemUpdating(ByVal sender As Object, _
    ByVal e As System.Web.UI.WebControls.DetailsViewUpdateEventArgs) _
    Handles DetailsView1.ItemUpdating
    e.NewValues("UpdateDateTime") = DateTime.Now
End Sub

Private Sub EndEditing()
    Response.Redirect("Reviews.aspx")
End Sub

```

C#

```

protected void DetailsView1_ItemInserted(object sender,
DetailsViewInsertedEventArgs e)
{
    EndEditing();
}

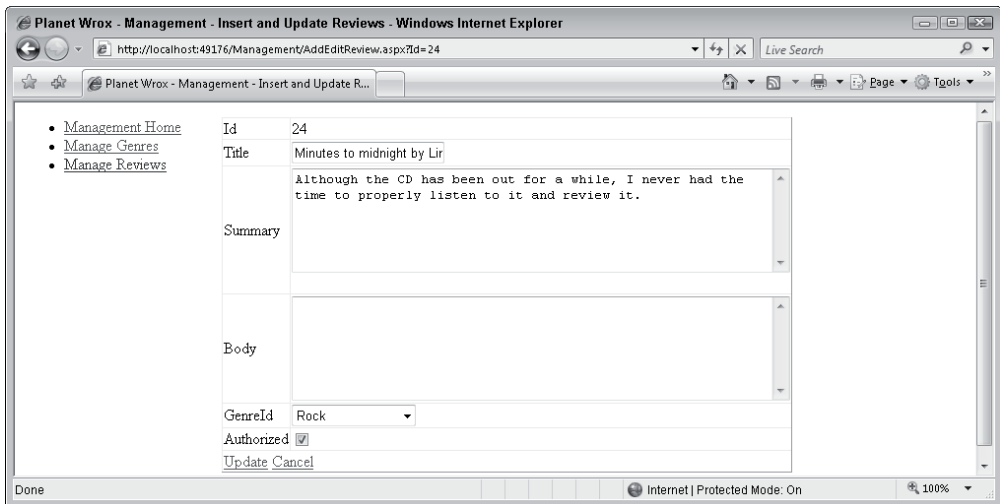
protected void DetailsView1_ItemInserting(object sender, DetailsViewInsertEventArgs e)
{
    e.Values["UpdateDateTime"] = DateTime.Now;
}

protected void DetailsView1_ItemUpdated(object sender, DetailsViewUpdatedEventArgs e)
{
    EndEditing();
}

protected void DetailsView1_ItemUpdating(object sender,
DetailsViewUpdateEventArgs e)
{
    e.NewValues["UpdateDateTime"] = DateTime.Now;
}

private void EndEditing()
{
    Response.Redirect("Reviews.aspx");
}

```

תרשים 12-24

17. לבסוף, שמור את כל השינויים ופתח את AddEditReview.aspx בדפדפן. השאר את כל השדות ריקים, ולחץ על הקישור Insert. שים לב שפקדי האימות ובקרת התוקף נכנסים לפעולה ומונעים ממך מלשלוח לשרת ערכים ריקים. עכשיו, מלא ערכים תקפים ולחץ שוב Insert. כעת תועבר ל-Reviews.aspx. אתר את הסקירה שלך על ידי בחירה בסגנון שלה ואז לחץ על הכותרת כדי לערוך אותה. מבט DetailsView אמור להציג כעת את כל הערכים שהכנסת קודם (ראה תרשים 12-24).

כיצד זה פועל

הפקדים DetailsView ו-SqlDataSource דואגים לרוב העבודה הקשה. הגדרת תבניות שונות שאיפשרו למשתמש להכניס רשומות חדשות ולעדכן רשומות קיימות ואחרי זה דאגו שני הפקדים לכל השאר. הבה נתבונן בכמה דברים מעניינים שיעזרו לך להבין כיצד כל זה פועל. הבט תחילה ב-InsertItemTemplate עבור העמודה Title:

```
<InsertItemTemplate>
  <asp:TextBox ID="TextBox1" runat="server" Text='<%# Bind("Title") %>'></asp:TextBox>
  <asp:RequiredFieldValidator ID="RequiredFieldValidator2" ontrolToValidate="TextBox1"
    runat="server" ErrorMessage="Please enter a title"></asp:RequiredFieldValidator>
</InsertItemTemplate>
```

קטע הקוד החשוב ביותר כאן הוא הדרך שבה קשור המאפיין Text של TextBox. קודם כבר ראית את תחביר הקשירה החד-סטרי משתמש ב-Eval, שבעצם פולט את הערך של עמודה קשורה. עם Bind קורה משהו רב עוצמה הרבה יותר. Bind מאפשר לבטא קשירת נתונים בין עמודה מ-SqlDataSource לבין פקד בדף בשני כיוונים. בדוגמה שלנו העמודה Title של סקירה קשורה אל TextBox. פירוש הדבר שכאשר הפקד חייב להציג את הנתונים שלו (לדוגמה, כאשר הוא מעדכן רשומה קיימת) הוא יודע שהוא חייב להציג את ה-Title של סקירה. אבל חשוב עוד יותר, בזמן postback, לאחר שאתה לוחץ על הקישור Update, הפקד עדיין מבין את הקשר

שבין הפקד TextBox לבין העמודה Title. כך, כשאתה לוחץ Update לאחר עריכת שינויים בסקירה שבדף, ה-DetailsView אוסף מהטופס את כל הנתונים הקשורים (ה-Title, Summary, Body, GenreId ו-Authorized) ואז שולח אותם לפקד SqlDataReader שיש לו פרמטרים מוגדרים עבור כל אחת מהעמודות הרלוונטיות של הטבלה Review.

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server">
```

```
...
```

```
<UpdateParameters>
```

```
  <asp:Parameter Name="Title" Type="String" />
```

```
  <asp:Parameter Name="Summary" Type="String" />
```

```
  <asp:Parameter Name="Body" Type="String" />
```

```
  <asp:Parameter Name="GenreId" Type="Int32" />
```

```
  <asp:Parameter Name="Authorized" Type="Boolean" />
```

```
  <asp:Parameter Name="UpdateDateTime" Type="DateTime" />
```

```
  <asp:Parameter Name="Id" Type="Int32" />
```

```
</UpdateParameters>
```

```
</asp:SqlDataSource>
```

בסופו של דבר, SqlDataReader מרכז את כל ערכי הפרמטרים, "מזריק" אותם אל UpdateCommand ושולח אותם לבסיס הנתונים.

כל זה פועל היטב עבור כל העמודות של הטבלה Review שמצורף להן פקד טופס, אבל מה לגבי כל שאר העמודות? ייתכן ששמת לב לכך שלשדה CreateDateTime לא היה חלק מאף אחת מפקודות SqlDataReader. מכיוון שהגדרת בסיס הנתונים מכוונת להכניס באופן אוטומטי את התאריך והשעה הנוכחיים, אין צורך לכלול את CreateDateTime בקוד. העמודה UpdateDateTime היא סיפור שונה. ברור שאינך רוצה שהמשתמשים יכניסו את הערך עבור עמודה זו באופן ידני. המערכת צריכה לעקוב אחריו באופן אוטומטי. זו הסיבה שהסתרת את הפקד מממשק המשתמש על ידי הגדרת המאפיין Visible שלו ל-False. עם זאת, מכיוון שהפקודות Insert ו-Update עדיין מצפות לערך עבור עמודה זו, עליך למצוא דרך שונה כדי להכניס אותו. זה המקום שבו נכנסים למשחק האירועים Inserting ו-Updating. כדי לקבל הבנה כללית כיצד זה פועל, התבונן באירוע ItemInserting.

VB.NET

```
Protected Sub DetailsView1_ItemInserting(ByVal sender As Object, _
    ByVal e As System.Web.UI.WebControls.DetailsViewInsertEventArgs) _
    Handles DetailsView1.ItemInserting
    e.Values("UpdateDateTime") = DateTime.Now
End Sub
```

C#

```
protected void DetailsView1_ItemInserting(object sender,
DetailsViewInsertEventArgs e)
{
    e.Values["UpdateDateTime"] = DateTime.Now;
}
```

כפי שראית קודם, ItemInserting מתעורר ממש לפני ש-InsertCommand נשלח לבסיס הנתונים. זהו המקום המושלם (ברירת המחדל) כדי לספק ערכים לעמודות בטבלה שלך שאין להן פקד מתאים בממשק המשתמש, כמו במקרה של UpdateDateTime. קוד זה רק מגדיר את הערך של UpdateDateTime לתאריך ולשעה הנוכחיים. ערך זה נשלח לבסיס הנתונים ומשמש בו לעדכון הערך UpdateDateTime של הטבלה Review.

עיקרון זה חל גם על הפקודה ItemUpdating. בתוך האירוע אתה צריך לקבוע אינדקסים לאוסף NewValues ולא לאוסף Values, אבל העיקרון הוא אחד.

אם הקדשת תשומת לב לנושא, אתה עשוי לטעון שבמקרה של הפקודה Insert אין צורך להגדיר את UpdateDateTime. אחרי הכל, בסיס הנתונים מכניס ערך באופן אוטומטי בזמן הכנסת רשומה חדשה. עם זאת, כדי ליצור הבחנה בין הכנסה ועדכון, תצטרך להשקיע עבודה ידנית רבה הרבה יותר. עליך להסיר את העמודה מ-InsertCommand ואחר כך להסיר אותה גם מהאוסף <InsertParameters>. למרות שלכשעצמה זו אינה עבודה רבה, תיקלע לבעיות מאוחר יותר, כשתנסה לשנות את פקודות SQL עבור SqlDataReader, מכיוון שהפקודות Insert ו-Update יצאו מסנכרון. הגדרת UpdateDateTime על ידי קוד, כמו במקרה שלנו, פותרת רבות מבעיות אלו.

כשהפקד SqlDataReader מסיים עם הכנסת רשומה או עדכונה, הוא מעורר את האירועים ItemInserted ו-ItemUpdated, בהתאמה. אירועים אלה קוראים מתוכם לשיטה EndEditing אשר מחזירה את המשתמש לדרך Reviews.aspx:

VB.NET

```
Private Sub EndEditing()  
    Response.Redirect("Reviews.aspx")  
End Sub
```

C#

```
private void EndEditing()  
{  
    Response.Redirect("Reviews.aspx");  
}
```

הדין במגוון האירועים שמעורר הפקד DetailsView, סוגר את הפרק. יש לך כעת הבנה טובה למדי כיצד לבצע את פעולות CRUD (יצירה, קריאה, עדכון ומחיקה) תוך שימוש בפקדים GridView, DetailsView ו-SqlDataSource. בפרק הבא תלמד כיצד להשתמש בפקד LinqDataSource ובפקדים ListView ו-DataPager כדי לבצע פעולות דומות מבלי לכתוב בקוד שלך משפטי SQL מוטמעים.

עצות מעשיות להצגת נתונים ועדכון נתונים

הרשימה הבאה מספקת כמה עצות מעשיות להצגה ועדכון של נתונים.

- ⊙ אחסן תמיד את מחרוזות החיבור בקובץ `web.config`. למרות שהדבר עשוי להיראות קל לאחסן אותן ישירות ב-`SqlDataSource` בדף, תיקלע לבעיות מאוחר יותר כשתצטרך לשנות את מחרוזות החיבור.
- ⊙ חשוב תמיד על הוספת פקדי אימות ובדיקת תוקף לדפי הזנת הנתונים שלך. הדבר מקל מאוד על המשתמשים למצוא איזה נתונים נדרשים ובאיזה עיצוב עליהם למסור אותם. בו-בזמן אתה מגן על המערכת מפני קבלה ועיבוד של נתונים לא תקפים או שגויים.
- ⊙ כשעליך להציג רשימות נתונים ארוכות, חשוב תמיד על הפעלת דפדוף עבור פקדים כמו `GridView`. משתמשים נוטים ללכת לאיבוד כשמוצגות להם רשימות עם יותר מ-50 פריטים. ככלל, גודל דף שבין 10 ל-20 פריטים הוא הטוב ביותר.
- ⊙ שקול את החלפת שמות הפקדים בדף לשמות משמעותיים עבורך, אשר שונים מערכי ברירות המחזל שלהם. לדוגמה, בתרגיל נסה זאת הקודם שינית את השם `SqlDataSource2` ל-`sdsGenres`. הדבר מקל מאוד על ראיית מקור הנתונים הנדרש כדי לקבל מידע אודות הסגנונות. כשמספר הפקדים בדף מועט זו אינה בעיה, אבל ברגע שהדף גדל גוברת החשיבות של בחירת שמות אופייניים עבור הפקדים.

סיכום

פרק זה התבסס על הידע הכללי שרכשת בנושא הגישה לבסיס נתונים באמצעות `SQL`. בפרק זה התחלנו בדיון על הפקדים הרבים שבקטגוריה `Data` של ארגז הכלים ב-`Visual Web Developer`.

פקדים אלה ניתן לפצל לשתי קבוצות: פקדים קשורי-נתונים ופקדי מקור-נתונים. קבוצת הפקדים הראשונה – שכוללת את `GridView`, `DetailsView` ו-`ListView` – משמשת להצגת נתונים בדף אינטרנט. רובם, אם כי לא כולם, מאפשרים גם לתחזק את הנתונים, מכיוון שהם בעלי יכולת להכניס נתונים, לעדכןם או למחוק אותם.

פקדי הקבוצה השנייה, פקדי מקור-נתונים, הם חסרי יכולת תצוגה חזותית משל עצמם. הם משמשים כגשר בין ממשק המשתמש לבין בסיס הנתונים. יש פקדי מקור-נתונים שונים, כל אחד מספק גישה לסוג מיוחד של מחסן נתונים. בפרק זה הכרת את הפקד `SqlDataSource` שמאפשר לאחזר נתונים מסוגים רבים ושונים של בסיסי נתונים טבלאיים.

כדי להציע למשתמשים דרך קלה לניהול נתונים פשוטים, כמו הטבלה `Genre` במערכת, אפשר להשתמש ב-`GridView` בצירוף עם `DetailsView`. כשהוא מחובר ל-`SqlDataSource`, מציע `GridView` תכונות מתקדמות כמו הצגת נתונים, מיון, דפדוף נתונים, עדכונים ומחיקתם וזאת עם תוספת מעט קוד או ללא תוספת כלל. אחר כך מאפשר `DetailsView` למשתמשים להכניס רשומות חדשות, מכיוון של-`GridView` אין תמיכה טבעית בפעולה זו.

כדי לסנן את הנתונים שאתה מציג בדפי האינטרנט, עליך להגדיר פסקת WHERE במשפט SQL של SqlDataReader. אחר כך יש לחבר את הפרמטרים של פסקת WHERE זו לאיזשהו פרמטר זמין, כולל SqlParameter שמקבל את הערך שלו מפקד קיים בדף, או QueryStringParameter שמקבל את הנתונים שלו ממחרוזת השאילתה.

מכיוון שלא סביר שיכולות העריכה הכלולות ב-GridView עונות על כל תרחישי הנתונים שלך, עליך לחפש דרכים אחרות להכנסת נתונים ועדכוןם. אחת מדרכים אלו היא DetailsView ביחד עם הפקד SqlDataReader. פקד DetailsView מאפשר להתאים אישית את התבניות עבור עמודות הבסיס של מקור הנתונים, ועל ידי כך מאפשר ליצור ממשק משתמש עשיר תוך שימוש בפקדים מורכבים יותר מה-TextBox הרגיל. בנוסף הוא נותן הזדמנות לשנות את המראה של הפקדים, להוסיף פקדי אימות, וכו'.

הפקד SqlDataReader אינו הדרך היחידה להכניס ולהוציא נתונים מבסיס הנתונים. בפרק 13 תכיר את LINQ to SQL - טכנולוגיה חדשה של מיקרוסופט שמאפשרת גישה לבסיס נתונים בדרך מוכוונת-עצמים טובה יותר.

תרגילים

1. אם אתה צריך ליצור ממשק משתמש שיאפשר למשתמשים להציג, לסנן, לערוך ולמחוק נתונים שמגיעים מבסיס הנתונים, מיהו הפקד הטוב ביותר לביצוע פעולות אלו? איך אתה מחבר פקד זה לבסיס הנתונים?
2. באיזה פקד תבחר אם תרצה להציג בעיצוב הבא רשימה פשוטה של הסגנונות שבבסיס הנתונים:

```
<ul>
  <li>Punk</li>
  <li>Hard Rock</li>
  <li>Jazz</li>
  <li>Techno</li>
</ul>
```

3. מה ההבדל בין BoundField לבין TemplateField? מתי תשתמש במי מבין השניים?
4. מהו המקום הטוב ביותר לאחסן בו מחרוזות חיבור? איך אתה ניגש למחרוזות החיבור מאותו מקום? ומדוע אסור לך לאחסן אותן בדף?

פרק 13

LINQ

אם היה עלי לבחור בתכונה החדשה המועדפת בעיני ב- ASP.NET 3.5, קרוב לוודאי שהייתי בוחר ב-LINQ, שפת השאילתות החדשה שמשולבת באופן הדוק בשפות התכנות שמשתמשים בהן בסביבת העבודה .NET. המילה LINQ מייצגת ראשי תיבות של **L**anguage-**I**Ntegrated **Q**uery, ומאפשרת לתשאל נתונים מתוך שפות התכנות של .NET. באופן דומה לזה שבו מאפשרת SQL לתשאל נתונים מבסיס נתונים. תחביר SQL שימש למעשה כדגם חלקי לתחביר LINQ, והדבר מקל על מתכנתים שמכירים את SQL להתחיל להשתמש ב-LINQ.

שפת השאילתות LINQ מאפשרת לגשת ולתשאל מגוון רחב של מקורות הבנויים על בסיס VB.NET או C#. המקורות כוללים אוספים בקוד שלך, קבצי XML, XDataSets, ובסיסי נתונים. בסעיף הבא תקבל סקירה קצרה של עמודי התווך של LINQ, שאר הפרק יעסוק ב-LINQ to SQL, אותו רכיב של LINQ שמאפשר לתשאל בסיסי נתונים מסוג SQL SERVER של מיקרוסופט.

בנוסף לתחביר LINQ שהוא כעת חלק משתי שפות התכנות הפופולריות ביותר, C# ו-Visual Basic, כוללת סביבת העבודה ASP.NET 3.5 גם כמה כלים שהופכים את השימוש ב-LINQ בפרויקטים לחלקה ופשוטה.

בנוסף, הפקד החדש `LinqDataSource` מאפשר לחבר ישירות פקדים קשורי-נתונים עם מקורות נתונים תואמי LINQ. בחלק השני של הפרק תלמד איך לקבוע את תצורת הפקד ואיך לצרוך את שירותיו.

בפרק זה תלמד ותכיר את שפת LINQ והתחביר שלה, וגם את הדרכים הרבות שבהן תוכל להשתמש בנתוני LINQ בפרויקטים של ASP.NET. פרק זה עוסק במיוחד בנושאים הבאים:

- ⊙ מהי LINQ ואיך נראה התחביר שלה?
- ⊙ אילו צורות עבודה שונות של LINQ זמינות, ומתי הן מתאימות לשימוש?
- ⊙ איך יוצרים מחלקות מסוג LINQ to SQL ואיך לתשאל אותם בדפי ASPX?
- ⊙ כיצד לקבוע את תצורת הפקד `LinqDataSource`, וכיצד להשתמש בו כדי לגשת למקורות נתונים של LINQ מתוך דפי ASPX?
- ⊙ כיצד להשתמש בפקדים החדשים `ListView` ו-`DataSource` בשילוב עם LINQ כדי ליצור דפי אינטרנט גמישים, מונעי-נתונים, עם כמות מזערית של קוד?

מבוא ל-LINQ

LINQ מאפשרת לתשאל נתונים ממגוון רחב של מקורות נתונים, ישירות מהקוד. היחס בין LINQ ליישום .NET הוא כמו היחס שבין SQL לבסיסי נתונים טבלאיים. עם תחביר הצהרתי ברור, באפשרותך לתשאל מאוספים את האובייקטים שתואמים לקריטריון נתון.

LINQ אינה רק תוספת שמהווה חלק מסביבת העבודה .NET. השפה LINQ תוכננה ויושמה כחלק ממשי של שפות התכנות ב-.NET. פירוש הדבר ש-LINQ משולבת באמת לתוך .NET. ונותנת דרך מאוחדת לתשאל נתונים מבלי להתחשב במקור הנתונים. בנוסף, מכיוון שהיא משולבת לתוך השפה ולא בסוג פרויקט מסוים, LINQ זמינה לכל סוגי הפרויקטים כולל יישומי אינטרנט, יישומי טפסי Windows, יישומי Console, ועוד. התחביר של LINQ דומה לזה של SQL, שפת השאילתות הפופולרית ביותר עבור בסיסי נתונים טבלאיים. פירוש הדבר, שכדי לקבל נתונים ממקור נתונים, משתמשת LINQ במילות מפתח כמו From, Select ו-Where.

כדי לקבל מושג כיצד נראית שאילתת LINQ, לפניך דוגמה שמציגה רשימה של מחברי הספרים של Wrox ששם משפחתם מתחיל באות S.

VB.NET

```
Imports System.Linq
...
Dim authors As String() = New String() { "Homer, Alex", "Sussman, Dave", _
    "Livermore, Shawn", "Bellinaso, Marco", "Spaanjaars, Imar"}
Dim result = From author In authors _
    Where author.StartsWith("S") _
    Select author
For Each author In result
    Label1.Text += author + "<br />"
Next
```

C#

```
using System.Linq;
...
string[] authors = new string[] { "Homer, Alex", "Sussman, Dave", "Livermore, Shawn",
    "Bellinaso, Marco", "Spaanjaars, Imar" };
var result = from author in authors
    where author.StartsWith("S")
    select author;

foreach (var author in result)
{
    Label1.Text += author + "<br />";
}
```

למרות שקל למדי לעקוב אחר התחביר שמשמש בדוגמה זו, הדוגמה עצמה היא באמת רבת עוצמה. כאשר לפניך מערך של מחרוזות שמכילות שמות מחברים, ביכולתך לבחור בכל

המחברים ששם משפחתם מתחיל באות **S**. אינך צריך להיות מופתע מכך שבדוגמה זו מציג הפקד Label את השמות של Dave Sussman ו-Imar Spaanjaars, מכיוון שאלה שני השמות היחידים שתואמים לקריטריון Where. בוודאי שזו רק ההתחלה. סוגי הפקודות השונות של LINQ מאפשרים ליצור שאילתות בעלות עוצמה רבה הרבה יותר כנגד מגוון רחב של מקורות נתונים.

הסוגים השונים של LINQ

מכיוון ש-LINQ כה רבת עוצמה, היא שולבה במקומות רבים ושונים בסביבת העבודה .NET. הסעיפים הבאים יציגו את השימושים השונים של LINQ.

LINQ to Objects

זו הצורה הטהורה ביותר של שילוב השפה. עם LINQ to Objects באפשרותך לתשאל אוספים ביישומי .NET. כפי שראית בדוגמה לעיל. אינך מוגבל למערכים, מכיוון ש-LINQ מאפשרת לתשאל כמעט כל סוג של אוסף שקיים בסביבת העבודה .NET.

LINQ to XML

LINQ to XML היא הדרך החדשה של .NET. לקרוא ולכתוב XML. במקום שפות שאילתה אופייניות ל-XSLT או XPath, ביכולתך כעת לכתוב באופן ישיר ביישום שאילתות LINQ שמכוונות ל-XML.

LINQ to SQL

LINQ to SQL מאפשרת לכתוב בפרויקט .NET. שאילתות מוכוונות-עצמים שמכוונות לבסיסי נתונים מסוג SQL SERVER. יישום LINQ to SQL מתרגם את השאילתות למשפטי SQL שנשלחים לבסיס הנתונים כדי לבצע פעולות CRUD אופייניות.

פרק זה דן אך ורק ב-LINQ to SQL. למידע נוסף על סוגי שימוש ויישום אחרים, בדוק את דף הבית הרשמי של LINQ באתר

<http://msdn2.microsoft.com/en-us/netframework/aa904594.aspx>

הצגת LINQ to SQL

LINQ to SQL היא למעשה LINQ to Microsoft SQL Server. הגרסה הרשמית של .NET 3.5 ששוחררה לציבור מצהירה על תמיכה ב-SQL Server 2000 ו-2005 בלבד, למרות שייתכן שצד שלישי ישחרר יישומים שמיועדים לבסיסי נתונים אחרים.

עם LINQ to SQL אפשר לקחת קבוצת אובייקטים של בסיס נתונים, טבלאות למשל, ולהפוך אותם לאובייקטים של .NET. שאליהם אפשר לגשת מתוך הקוד. ביכולתך להשתמש אז באובייקטים אלה בשאילתות או להשתמש בהם ישירות בתרחישי קשירת נתונים (data binding).

השימוש ב-LINQ to SQL קל וגמיש למדי. על ידי שימוש בכלי העיצוב, אתה גורר אובייקטים מבסיס הנתונים ומשחרר אותם על כלי העיצוב. אובייקטים אלה יכולים להיות טבלאות, תצוגות (Views), פרוצדורות מאוחסנות, ואפילו פונקציות מוגדרות על ידי המשתמש. טבלאות בסיס הנתונים שמשוחררות על כלי העיצוב הופכות זמינות כאובייקטים. לדוגמה, אם תשחרר את הטבלה Review על כלי העיצוב, תקבל את המחלקה Review. באפשרותך ליצור מופעים של מחלקה זו על ידי השימוש בשאילתות LINQ ובאמצעים אחרים, כפי שתראה במהלך הפרק.

כשמשחררים בכלי העיצוב יותר מטבלה קשורה אחת של בסיס הנתונים, המעצב (designer) של LINQ to SQL מגלה את הקשרים שיצרת בין הטבלאות בבסיס הנתונים. המעצב משכפל קשרים אלה במודל האובייקטים (object model). לדוגמה, אם יש מופע של Review שנוצר בקוד על ידי שימוש בשאילתת LINQ כלשהי (כפי שתראה מאוחר יותר), תוכל לגשת למאפיין Genre שלו, שבתורו נותן גישה למאפיינים כמו Name.

VB.NET

```
Label1.Text = myReview.Genre.Name
```

C#

```
Label1.Text = myReview.Genre.Name;
```

באופן דומה תוכל לגשת לאוסף Reviews הקשור עבור סגנון (genre, ז'אנר) מסוים.

VB.NET

```
Repeater1.DataSource = myGenre.Reviews
```

C#

```
Repeater1.DataSource = myGenre.Reviews;
```

ברגע זה, הסר דאגה מלבך בקשר לתחביר הממשי. תפגוש בו הרבה בהמשך הפרק. עם זאת, חשוב להבין ש-LINQ to SQL יוצרת שכבה שהיא מודל האובייקטים בין יישום .NET. לבין בסיס הנתונים של SQL Server. המעצב של LINQ to SQL עושה עבורך את רוב העבודה, ומספק גישה למודל אובייקטים נקי שבו תוכל להשתמש ביישום שלך.

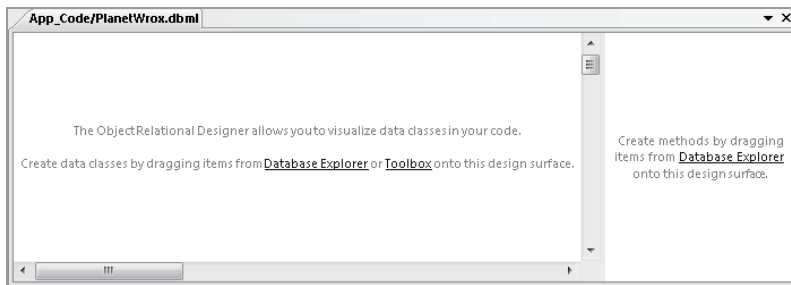
מיפוי מודל הנתונים למודל אובייקטים

עם LINQ to SQL אתה ממפה פריטי בסיס נתונים כמו טבלאות, עמודות, וקשרים בבסיס הנתונים, אל אובייקטים ומאפיינים במודל האובייקטים ביישום. התוכנה VWD מכילה כלים מעולים שתפקידם לגרום שמיפוי זה יהיה קל ככל האפשר, כפי שתראה בתרגיל הבא. סעיפים ותרגילים מאוחרים יותר בפרק יתנו לך מבט מעמיק יותר על הדרך שבה פועלת LINQ to SQL.

נסה זאת דוגמה פשוטה של LINQ

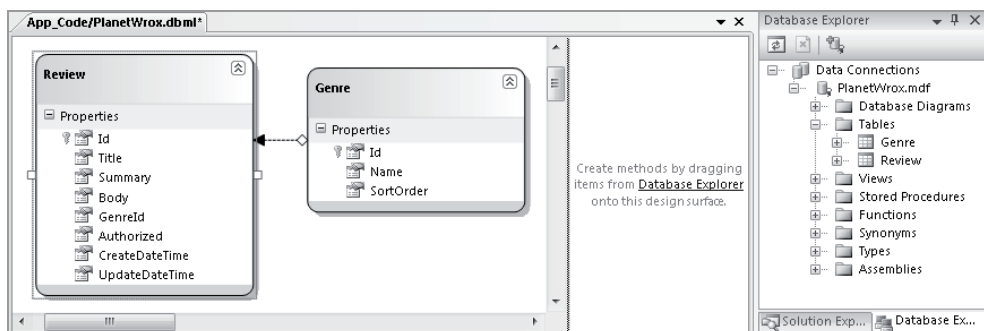
בתרגיל זה תראה כיצד להוסיף קובץ של מחלקה מסוג LINQ to SQL לפרויקט, להוסיף טבלאות בסיס נתונים למודל, ואז לכתוב שאילתת LINQ פשוטה כדי לגשת לנתונים בטבלאות הבסיסיות.

1. פתח את הפרויקט Planet Wrox שאתה כבר מכיר מעבודתך בפרויקט עד כה. לחץ לחיצה ימנית על התיקייה App_Code, בחר Add New Item, ולחץ על הפריט LINQ to SQL Classes. הקלד את השם **PlanetWrox**, ובחר בשפת התכנות שלך. לבסוף, לחץ Add כדי להוסיף את הפריט לפרויקט.
2. VWD מוסיפה לאתר קובץ בשם PlanetWrox.dbml, ואז פותחת את כלי העיצוב Object Relational Designer בחלון הראשי של העורך, כפי שמוצג בתרשים 13-1.



תרשים 13-1

- חלון זה מפוצל לשני חלקים. בחלונית השמאלית אתה יכול לגרור ולשחרר פריטים כמו טבלאות בסיסי נתונים או פריטים מתוך הקטגוריה Object Relational Designer שבתחת הכלים, כדי ליצור מחלקות נתונים עבור היישום. בחלונית הימנית ביכולתך לשחרר פריטים כמו פונקציות ופרוצדורות מאוחסנות מבסיס הנתונים, כדי ליצור שיטות נוספות במודל האובייקטים.
3. ודא שחלון סייר בסיס הנתונים (Database Explorer), או סייר השרת, Database Explorer, בגרסאות המסחריות של Visual Studio) אמנם פתוח (ואם אינו פתוח, הקש Ctrl+Alt+S) ואתר את בסיס הנתונים שלך.
 4. בסייר בסיס הנתונים, הרחב את בסיס הנתונים ואז הרחב את הצומת Tables. גרור את הטבלה Review מהסייר אל הצד השמאלי של התרשים. אחר כך עשה זאת עם הטבלה Genre. שים לב ש-VWD משרטטת קו שמחבר את שתי הטבלאות, כפי שמוצג בתרשים 13-2. הקו מציינ ש-VWD גילתה את הקשר שיצרת בין שתי הטבלאות הללו בפרק 11.



תרשים 13-2

5. שמור את התרשים וסגור אותו.
6. עבור לסייר הפתרון (Solution Explorer), ופתח את הקובץ All.aspx מהתיקייה Reviews. עבור לתצוגה Design View וגרור GridView מארגז הכלים אל הדף.
7. לחץ לחיצה כפולה על הדף, באזור האפור לקריאה-בלבד, כדי לגרום ל-VWD לקבוע מטפל לאירוע Page_Load, והוסף את הקוד המוצלל הבא:

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Me.Load
```

```
Using myDataContext As New PlanetWroxDataContext()
    Dim allReviews = From review In myDataContext.Reviews _
        Where review.Authorized = True _
        Order By review.CreateDateTime Descending _
        Select review

    GridView1.DataSource = allReviews
    GridView1.DataBind()
End Using
```

```
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
```

```
{
    using (PlanetWroxDataContext myDataContext = new PlanetWroxDataContext())
    {
        var allReviews = from review in myDataContext.Reviews
            where review.Authorized == true
            orderby review.CreateDateTime descending
            select review;

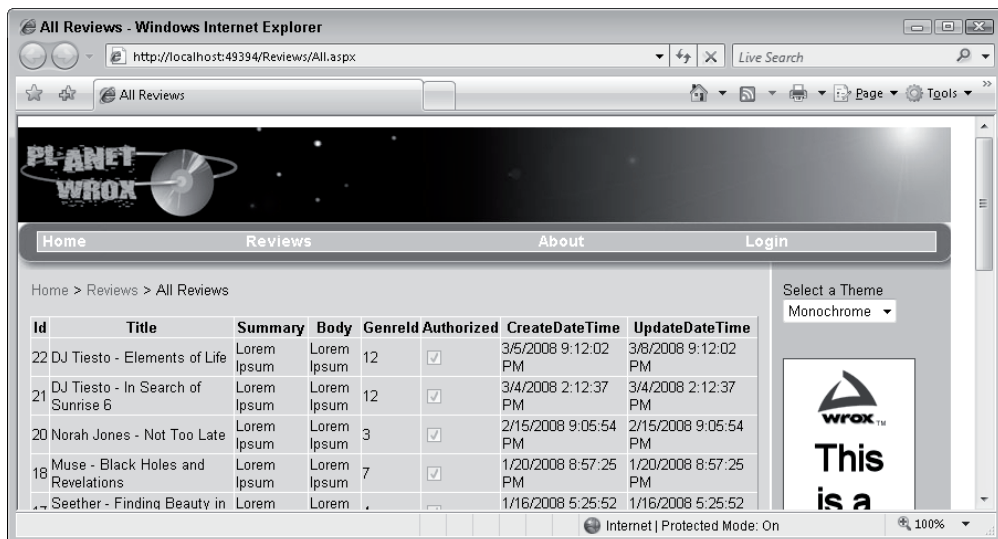
        GridView1.DataSource = allReviews;
        GridView1.DataBind();
    }
}
```

8. שמור את כל השינויים והקש Ctrl+F5 כדי לפתוח את הדף. כעת תקבל מסך עמוס בסקירות שאוחזרו מהטבלה Review שבבסיס הנתונים, כפי שמוצג בתרשים 3-13. הדף נראה מבולגן למדי בגלל הדרך שבה מוצגים הנתונים ב-GridView. בתרגיל מאוחר יותר תלמד איך לשפר את עיצוב הפקד והנתונים.

כיצד זה פועל

התוכנה VWD כוללת את המעצב object-relational שמאפשר ליצור מודל אובייקטים שנגיש דרך קוד שמבוסס על טבלאות בסיס הנתונים. על ידי גרירת טבלאות אל מעצב זה, VWD יוצרת קוד שמאפשר לגשת לנתוני היסוד בבסיס הנתונים מבלי שתצטרך לכתוב בעצמך קוד רב. הטבלאות

שאתה גורר אל המעצב מאוחסנות בקובץ dbml ובקבצי הקוד ברקע שלו. קבצים אלה מכילים מחלקה שיורשת מ-DataContext, הישות העיקרית ב- LINQ to SQL שמספקת גישה לבסיס הנתונים. בדוגמה הקודמת נקראה מחלקה זו בשם PlanetWroxDataContext. היא משמשת לגישה לנתונים בטבלאות שהוספת לתרשים. למרות שבדרך כלל אינך צריך להתבונן בקוד שנוצר, ביכולתך לפתוח את הקובץ PlanetWrox.designer.vb או PlanetWrox.designer.cs ולהתבונן בו.



תרשים 13-3

המעצב חכם דיו כדי לגלות את הקשרים בבסיס הנתונים, ולכן הוא מסוגל ליצור את הקשרים הנחוצים גם בקוד, כפי שראית בתרשים 2-13. המודל יוצר שני סוגים עיקריים של אובייקטים, Review ו-Genre, ולכל אחד מהם יש אוסף מקביל, Reviews ו-Genres, בהתאמה. שים לב שהמעצב קבע עבור שמות הטבלאות Review ו-Genre צורת רבים (Reviews ו-Genres). בהתאמה, על ידי כך קל יותר להבחין בין אוסף (Reviews) לבין מופע של אובייקט (Review).

לאחר שהמודל נוצר, ביכולתך לבצע כנגדו שאילתות LINQ כדי לאחזר נתונים מבסיס הנתונים הבסיסי. כדי לגשת לנתונים דרוש מופע של המחלקה DataContext, אשר נוצרת בקוד בתוך הבלוק Using. בלוק Using (using ב-C#) משמש לעטיפת קוד שיוצר משתנה, שחובה להיפטר ממנו (לנקותו מהזיכרון) לאחר סיום השימוש בו. המשתנה myDataContext מחזיק בחיבור לבסיס נתונים של SQL Server (משאב יקר), ולכן מומלץ לעטוף את הקוד שמשתמשים בו בבלוק Using, כדי שהאובייקט ייהרס בסיום ביצוע הבלוק. אובייקט myDataContext זה חושף את הנתונים (כמו Reviews ו-Genres) שבהם ניתן להשתמש בשאילתה.

VB.NET

```
Using myDataContext As New PlanetWroxDataContext()
    Dim allReviews = From review In myDataContext.Reviews _
        Where review.Authorized = True _
        Order By review.CreateDateTime Descending _
        Select review
```

```
...
End Using
```

C#

```
using (PlanetWroxDataContext myDataContext = new PlanetWroxDataContext())
{
    var allReviews = from review in myDataContext.Reviews
                      where review.Authorized == true
                      orderby review.CreateDateTime descending
                      select review;

    ...
}
```

שים לב ששאילתה זו נראית דומה לשפת SQL שלמדת בפרקים הקודמים. לפיכך, למי שיש הבנה מעטה של SQL יהיה קל למדי להתחיל להשתמש ב-LINQ. למעשה, רכיב runtime ממיר את שאילתת LINQ לשאילתת SQL המקבילה שלה, ואז מבצע אותה כנגד בסיס הנתונים הבסיסי.

המשתנה review מוגדר בפסקת From ונעשה בו שימוש בחלקים האחרים של השאילתה (Where, Select-1 Order By), שמאפשר לציין את הקריטריון לבחירה, לסינון ולסידור (מיון התוצאות).

מלבד האוסף Reviews מכיל המודל גם אוסף Genres. כשתרצה לבחור בכל הסגנונות בבסיס הנתונים, תוכל להשתמש בשאילתה הבאה:

VB.NET

```
Dim allGenres = From genre In myDataContext.Genres _
                 Order By genre.Name _
                 Select genre
```

C#

```
var allGenres = from genre in myDataContext.Genres
                 orderby genre.Name
                 select genre;
```

בנוסף לשני אובייקטים נפרדים אלה והאוספים שלהם, יש לשניהם מאפיינים שמתייחסים האחד לסוג של האחר. לדוגמה למופע של Review יש מאפיין Genre שמספק מידע נוסף אודות הסגנון שאליו הוקצתה הסקירה. ולמופע של Genre, יש מאפיין שהוא האוסף Reviews שנותן גישה לכל הסקירות שפורסמו בסגנון זה. מאוחר יותר תלמד כיצד להשתמש במאפיינים אלה.

ממילות המפתח שהשתמשנו בהן בשאילתה הראשונה בתרגיל זה קל להסיק מה השאילתה עושה: היא מקבלת רשימה של כל הסקירות/ביקורות במערכת שיש להן הרשאה, ומסדרת אותן בסדר יורד בהתאם לתאריך יצירתן. תוצאת השאילתה מוקצית למשתנה allReviews. ייתכן ששמת לב לתחביר משונה בנקודה זו. לדוגמה, הקוד ב-VB.NET אינו משתמש במילת המפתח As כדי להגדיר את סוג המשתנה. באופן דומה, הקוד ב-C# משתמש במילת המפתח החדשה var, גם כאן ללא שם הסוג. אף על פי שאינך יכול להסיק זאת מקטעי הקוד לעיל, הרי שבשתי השפות המשתנה allReviews הוא עדיין בעל סוג מובהק ולא רק משתנה מסוג Object. הרעיון מאחורי זה

נקרא **הסקת הסוג** (type inference), והכוונה היא שהמהדר מסוגל להסיק את סוג המשתנה על ידי התבוננות בצד הימני של ההשמה. במקרה שלנו, המהדר רואה שהשאלתה תחזיר רשימת אובייקטים של Review, ומסווג נכונה את המשתנה allReviews כסוג כללי (generics type), בתחביר VB.NET - IQueryable(Of Review), ובתחביר C# - IQueryable<Review>. בתחילה כל זה נראה מפחיד מעט ובלתי מובן, אבל תראה שהוא נעשה קל יותר להבנה אם תפרש זאת פשוט כמו "קבוצת אובייקטים של Review שאפשר לגשת אליהם בשאלות".

אובייקטים אלה של Review מוקצים אז למאפיין DataSource של הפקד GridView. בפרקים קודמים למדת כיצד להשתמש במאפיין DataSourceID כדי להפנות פקד כמו GridView לפקד מקור נתונים כמו SqlDataReader. במקום זאת, על ידי שימוש במאפיין DataSource, תוכל להקצות את הנתונים הממשיים, שבהם הפקד ישתמש כדי לבנות את ממשק המשתמש.

VB.NET

```
GridView1.DataSource = allReviews  
GridView1.DataBind()
```

C#

```
GridView1.DataSource = allReviews;  
GridView1.DataBind();
```

על ידי קריאה לשיטה DataBind() מתוך GridView אתה מורה לפקד להציג בדף את האובייקטים הנבחרים של Review. מכיוון שהמאפיין AutoGenerateColumns של הפקד GridView מוגדר כ-True (ברירת המחדל), הפקד יוצר עמודה עבור כל מאפיין שהוא מוצא באובייקט של Review. מאוחר יותר תלמד כיצד להתאים אישית את הפקד ואת הנתונים שמוקצים למאפיין DataSource. בסעיף הבא נסיף ונלמד על תחביר השאלתה של LINQ, השפה שמניעה את יכולות התשאול של .NET 3.5.

הצגת תחביר השאלתה

בדוגמה הקודמת ראית שאלתה פשוטה למדי, היא מבקשת מהמערכת את כל הסקירות המורשות, ומחזירה אותן בסדר מיון רצוי. עם זאת, יכולות התשאול של LINQ הן בעלות עוצמה רבה הרבה יותר מכך. בסעיף זה תלמד אודות תחביר שאלתה LINQ שבו תשתמש כדי לתשאל את מודל האובייקטים.

אופרטורי שאלתה תקינים

LINQ תומכת במספר גדול של אופרטורים של שאלתה - מילות מפתח שמאפשרות לבחור, לסדר או לסנן את הנתונים שיוחזרו מהשאלתה. למרות שכל הדוגמאות בפרק זה נידונות בהקשר של LINQ to SQL, ניתן בקלות להחיל אותן גם על סוגים שונים של LINQ. בסעיף הבא תקבל מבט כולל אשר מלווה בדוגמה, על האופרטורים התקינים החשובים ביותר של שאלתה. כל אחת מהדוגמאות משתמשת במודל האובייקטים, תוך שימוש באובייקט DataContext בשם myDataContext שיצרת קודם כמקור הנתונים שכנגדו תבוצענה השאלות.

Select

מילת המפתח Select (select ב-C#) משמשת לאחזור אובייקטים מהמקור שאותו מתשאלים. בדוגמה הבאה תראה כיצד לבחור אובייקט מסוג קיים. מאוחר יותר בפרק תראה כיצד להגדיר צורות אובייקטים חדשות תוך כדי ביצוע.

VB.NET

```
Dim allReviews = From r In myDataContext.Reviews _  
                  Select r
```

C#

```
var allReviews = from r in myDataContext.Reviews  
                  select r;
```

המשתנה 'r' בדוגמה זו הוא משתנה שזמין רק בתוך השאילתה הנוכחית. משתנה זה מוצג בדרך כלל בפסקת From, ומשתמשים בו שוב בפסקת Where ו-Select כדי לסנן נתונים וכדי לציין את הנתונים שברצונך לבחור. למרות שבאפשרותך לבחור למשתנה כל שם שתרצה, הרי שלעתים קרובות תראה משתנים בעלי שם של אות אחת כמו 'r' (עבור Review) או שהשם הוא צורת היחיד של האוסף שאותו מתשאלים (בדוגמאות לעיל, review במקום r).

From

From אינו נחשב לאופרטור שאילתה תקני, מכיוון שאינו פועל על הנתונים אלא מצביע עליהם. למרות זאת, הפסקה From (from ב-C#) היא גורם חשוב בשאילתת LINQ מכיוון שהיא קובעת את שם האוסף או שם מקור הנתונים שהשאילתה חייבת לפעול עליו.

Order By

הפסקה Order By (orderby ב-C#) מאפשרת למיין את הפריטים באוסף התוצאה. לאחר Order By יכולות לבוא (רשות) מילות המפתח Ascending או Descending (או ascending או descending ב-C#) שמציינות את סדר המיון. אפשר לציין קריטריונים רבים על ידי הפרדתם בפסיק זה מזה. השאילתה הבאה מחזירה רשימה של סגנונות שממוינים תחילה לפי sortOrder בסדר יורד, ואז ממוינים לפי Name בסדר עולה.

VB.NET

```
Dim allGenres = From g In myDataContext.Genres _  
                  Order By g.SortOrder Descending, g.Name _  
                  Select g
```

C#

```
var allGenres = from g in myDataContext.Genres  
                  orderby g.SortOrder descending, g.Name  
                  select g;
```

Where

הפסקה Where ב-LINQ (ב-C# where), בדיוק כמו הפסקה WHERE ב-SQL, מאפשרת לסנן את האובייקטים שמוחזרים על ידי השאילתה. השאילתה הבאה מחזירה את כל הסקירות המורשות:

VB.NET

```
Dim allReviews = From r In myDataContext.Reviews _  
                  Where r.Authorized = True _  
                  Select r
```

C#

```
var allReviews = from r in myDataContext.Reviews  
                  where r.Authorized == true  
                  select r;
```

שים לב שהפסקה Where משתמשת באופרטור השוויון הרגיל של השפה, יש סימן שוויון יחיד (=) ב-VB.NET, ושניים ב-C#.

Count, Average, Max, Min, Sum

אופרטורי צבירה אלה מאפשרים לבצע חישובים מתמטיים על האובייקטים שבקבוצת התוצאה. לדוגמה, כדי לאחזר את SortOrder הממוצע עבור כל הסגנונות, תוכל לבצע את השאילתה הבאה:

VB.NET

```
Dim average = (From g In myDataContext.Genres _  
                Select g.SortOrder).Average()
```

C#

```
var average = (from g in myDataContext.Genres  
                select g.SortOrder).Average();
```

שים לב שהשיטה Average חלה על קבוצת התוצאה כולה. לכן, יש צורך לעטוף את המשפט כולו בסוגריים ובעקבותיהם תבוא קריאה לשיטה Average. ללא הסוגריים תקבל שגיאה. לגבי המשתנה average בדוגמה זו, מסיקים שהוא מסוג Double, ויכיל את הערך הממוצע של כל ערכי SortOrder בטבלה Genre. החישוב שעשינו אמנם אינו שימושי ביותר ביישום אמיתי, אבל הוא טוב דיו לשרת כאן כדוגמה להפעלת השיטה Average.

באופן דומה, תוכל לאחזר בדרך זו את מספר הסקירות:

VB.NET

```
Dim numberOfReviews = (From r In myDataContext.Reviews _  
                        Select r).Count()
```

C#

```
var numberOfReviews = (from r in myDataContext.Reviews  
                        select r).Count();
```


SkipWhile, TakeWhile, Skip, Take

Take ו-Skip מאפשרות לבצע בחירות-משנה בתוך קבוצת התוצאה. הדבר מתאים ביותר לתרחישי דפדוף שבהם הרשומות מאוחזרות עבור הדף הנוכחי בלבד. Take אוספת כמות (לפי המספר המבוקש) של אלמנטים מקבוצת התוצאה ומתעלמת מהשאר, בעוד Skip מדלגת על מספר האלמנטים המבוקשים ומחזירה את הבאים.

גם האופרטורים Take ו-Skip מתורגמים בתוך LINQ to SQL למשפטי SQL. פירוש הדבר שהדפדוף מתרחש ברמת בסיס הנתונים, ולא בדף ASP.NET. הדבר משפר את ביצועי השאילתה באופן ניכר, במיוחד עבור קבוצות תוצאה גדולות, מכיוון שאין צורך להעביר את כל האלמנטים מבסיס הנתונים לדף ASP.NET.

הדוגמה הבאה מראה כיצד לאחזר את דף הרשומות השני, כאשר מוגדר שגודל דף הוא 10:

VB.NET

```
Dim allReviews = (From r In myDataContext.Reviews _  
                  Select r).Skip(10).Take(10)
```

C#

```
var allReviews = (from r in myDataContext.Reviews  
                  select r).Skip(10).Take(10);
```

בדיוק כמו במקרה של Average, השאילתה עטופה בזוג סוגריים ובעקבותיהם באה הקריאה ל-Skip ו-Take להשיג את הרשומות המבוקשות.

האופרטורים TakeWhile ו-SkipWhile של השאילתה פועלים בצורה דומה, אבל מאפשרים לקבל או לדלג על רשומות כשמתקיים תנאי נתון.

Single ו-SingleOrDefault

האופרטורים Single ו-SingleOrDefault מאפשרים להחזיר אובייקט יחיד כמופע בעל סוג מובהק. הדבר שימושי כשידוע שהשאילתה מחזירה רשומה אחת בדיוק; לדוגמה, כשמאחזרים רשומה לפי ID (הזיהוי) הייחודי שלה. הדוגמה הבאה מאחזרת מבסיס הנתונים את הסקירה בעלת ID שערכו 22:

VB.NET

```
Dim review22 = (From r In myDataContext.Reviews _  
                Where r.Id = 22 _  
                Select r).Single()
```

C#

```
var review22 = (from r in myDataContext.Reviews  
                where r.Id == 22  
                select r).Single();
```

אם הפריט המבוקש לא נמצא, או כאשר השאילתה מחזירה יותר ממופע אחד, יעורר האופרטור Single שגיאה. אם רצונך שהשיטה תחזיר ריק (null), כמו לדוגמה עבור סקירה או סגנון שלא נמצא(ו) או את ערך ברירת המחדל עבור סוג הנתונים הרלוונטי (כמו 0 עבור Integer, False עבור בוליאני וכו'), השתמש ב-SingleOrDefault.

למרות שבבסיס הנתונים קיימת רק סקירה אחת בעלת Id שערכו 22, הרי שאם תשמיט את הקריאה ל-Single, עדיין תקבל רשימת סקירות (שמחזיקה אלמנט אחד בלבד). השימוש ב-Single כופה החזר של מופע יחיד מהסוג שאתה מתשאל לגביו.

First, FirstOrDefault, Last ו-LastOrDefault

אופרטורים אלה מאפשרים להחזיר את האיבר הראשון או האחרון בסדרה נתונה של אובייקטים. בדיוק כמו עם האופרטור Single, גם First ו-Last מעוררים שגיאה כשהאוסף ריק, בעוד שני האופרטורים האחרים מחזירים את ערך ברירת המחדל עבור סוגי הנתונים הרלוונטיים.

בניגוד ל-Single, האופרטורים First, FirstOrDefault, Last ו-LastOrDefault אינם מעוררים שגיאה כשהשאילתה מחזירה יותר מפריט אחד.

לרוע המזל, האופרטורים Last ו-LastOrDefault אינם נתמכים ב-LINQ to SQL. עם זאת, תוכל בקלות להשלים את אותה התנהגות עם First וסדר מיון יורד. קטע הקוד הבא מראה כיצד לאחזר מבסיס הנתונים את הסקירה הישנה ביותר ואת הסקירה החדשה ביותר:

VB.NET

```
Dim firstReview = (From r In myDataContext.Reviews _
    Order By r.Id _
    Select r).First()

Dim lastReview = (From r In myDataContext.Reviews _
    Order By r.Id Descending _
    Select r).First()
```

C#

```
var firstReview = (from r in myDataContext.Reviews
    orderby r.Id
    select r).First();

var lastReview = (from r in myDataContext.Reviews
    orderby r.Id descending
    select r).First();
```

על ידי סידור מחדש של קבוצת התוצאה לפני ביצוע First, אתה מקבל למעשה את הרשומה האחרונה בסדרה. שים לב שבשני המקרים, הסוג שמוחזר על ידי השאילתה הוא אובייקט Review אמיתי, והדבר מאפשר לגשת ישירות למאפיינים כמו Id ו-Title.

עיצוב נתונים עם סוגים אנונימיים (Anonymous Types)

השאלות שראית עד כה בסעיפים הקודמים החזירו סוגים שלמים. כלומר, השאלות החזירו רשימה של מופעי Review (כמו האופרטור Select), מופע יחיד של First, Single Review או (Last), או ערך מספרי (כמו Count ו-Average).

עם זאת, לעתים קרובות אין צורך בכל המידע מאובייקטים אלה. תרשים 3-13 מציג פקד GridView עם כל המאפיינים של אובייקט Review. כדי לשפר הצגת רשימה זו, תרצה בדרך כלל לדלג על מאפיינים כמו Body ו-Authorized, ובמקום ID של סגנון, קרוב לוודאי שתרצה להציג את שם הסגנון. קל למדי לבצע זאת עם סוגים אנונימיים (anonymous types), מבנים אנונימיים, תכונה חדשה נוספת של השפות C# ו-VB.NET. סוג אנונימי הוא סוג ללא שם שאינך מגדיר אותו מראש, כפי שעושים עם סוגים אחרים כמו מחלקות. במקום זאת אתה בונה את הסוג האנונימי על ידי בחירת נתונים, ומניח למהדר להסיק את סוגם.

אם אינך מגדיר את הסוג הממשי ואינך נותן לו שם, כיצד תוכל לגשת אליו ואל מאפייניו? גם כאן הדבר מתבצע על ידי הסקת הסוג, כשהמהדר יכול לראות איזה נתונים הוקצו למשתנה.

יצירת סוג אנונימי היא פעולה קלה; במקום לבחור באובייקט הממשי תוך שימוש במשהו כמו Select review, השתמש ב-C# במילת המפתח new (ב-New With Visual Basic) ואז תגדיר בין זוג סוגריים מסולסלים את המאפיינים שאתה רוצה לבחור.

VB.NET

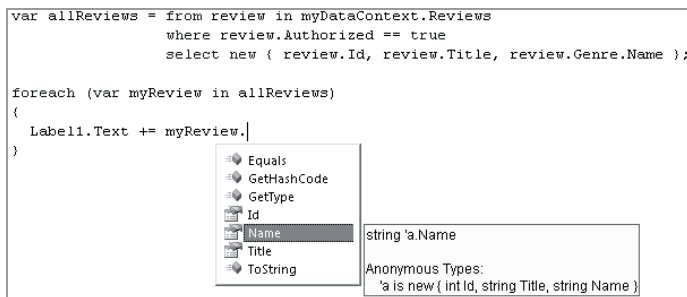
```
Dim allReviews = From myReview In myDataContext.Reviews _
    Where myReview.Authorized = True _
    Select New With {myReview.Id, myReview.Title, _
        myReview.Genre.Name}
```

C#

```
var allReviews = from review in myDataContext.Reviews
    where review.Authorized == true
    select new { review.Id, review.Title, review.Genre.Name };
```

למרות שהסוג הוא אנונימי והמהדר אינו יכול לגשת אליו ישירות לפי שמו, המהדר עדיין יכול להסיק את הסוג ולתת IntelliSense מלא עבור המאפיינים החדשים שנבחרו בשאלתה. תרשים 4-13 מראה כיצד לגשת למאפיינים של הסוג האנונימי במשתנה allReviews, על ידי שימוש במילת המפתח var (C#).

שים לב שהשאלתה הקודמת ניגשה למאפיין הממשי Genre של Review. בנוסף ל-GenreId שלה (שמוגדר כעמודה בטבלה Review בבסיס הנתונים) יש כעת למחלקה Review גם מאפיין Genre בעל סוג מובהק (אובייקט), שמאפשר גישה ישירה למאפיינים שלו, Name למשל, כפי שהודגם בשאלתה הקודמת.



תרשים 13-4

פרט לבחירה ישירה של מאפיינים קיימים - כפי שהוצגה בשאילתה שבחרה ב-Id, Title ו-Name של Genre - באפשרותך גם להחליט על ערכי מאפיינים ולקבוע להם שמות שונים תוך כדי ביצוע. לדוגמה, השאילתה הבאה יוצרת סוג אנונימי חדש שקורא ל-Id בשם החדש Number, מגביל את Title ל-20 התווים הראשונים, ומכיל ערך בוליאני שקובע אם הפריט עודכן קודם לכן בבסיס הנתונים.

VB.NET

```
Dim allReviews = From myReview In myDataContext.Reviews _
                  Select New With {.Number = myReview.Id, _
                                   .Title = myReview.Title.Substring(0, 20), _
                                   myReview.Genre.Name, _
                                   HasBeenUpdated = (myReview.UpdateDateTime > myReview.CreateDateTime)}
```

C#

```
var allReviews = from myReview in myDataContext.Reviews
                  select new
                  {
                      Number = myReview.Id,
                      Title = myReview.Title.Substring(0, 20),
                      myReview.Genre.Name,
                      HasBeenUpdated = (myReview.UpdateDateTime > myReview.CreateDateTime)
                  };
```

שים לב להבדל בין VB.NET לבין C#. בדוגמת VB.NET יש לשמות של המאפיינים החדשים (Number, Title ו-HasBeenUpdated) תחילית של נקודה (.). דרישה זו אינה קיימת ב-C# שמאפשרת לכתוב את שמות המאפיינים החדשים באופן ישיר.

היכולת לבחור מאפיינים נוספים שאינם נוכחים באובייקט המקורי, נותנת גמישות רבה לגבי הנתונים שמוצגים. בדוגמה לעיל, השוואת המאפיינים CreateDateTime ו-UpdateDateTime קובעת אם הסקירה הנוכחית עודכנה. תוצאות השוואה זו (משתנה בוליאני בעל הערך True או False) מאוחסנות במאפיין HasBeenUpdated. באפשרותך לבחור כמעט בכל דבר שתמצא, כולל התאריך והשעה הנוכחיים, חישובים מורכבים, תת-מחרוזות או שילובים של מאפיינים ועוד.

בתרגיל הבא תלמד ליצור סוג אנונימי חדש שיש לו אוסף בשם Reviews כמאפיין. תשתמש בסוג זה כדי ליצור רשימה של כל הסגנונות הזמינים בבסיס הנתונים, וכל הסקירות שנכללות בכל סגנון.

בתרגיל זה תיצור דף פרטים ראשי פשוט, שמציג רשימה של כל הסגנונות הזמינים. לאחר כל סגנון מוצגת רשימת הסקירות שפורסמו אודותיו. הפקד Repeater ישמש להצגת רשימת הסגנונות, ו-BulletedList מקוֹנֵן ישמש להצגת רשימת הסקירות הפנימית. כשתסיים, קרוב לוודאי שתראה רשימה דומה לזו שמוצגת בתרשים 5-13.

1. בתיקייה Reviews, פתח את הדף AllByGenre.aspx. ודא שהדף בתצוגת Markup View ומהקטגוריה Data של ארגז הכלים גרור Repeater לתוך שומר מקום התוכן cpMainContent.

2. בין תגיות הפתיחה והסגירה של Repeater צור אלמנט <ItemTemplate>. בתוך האלמנט צור אלמנט <h3> המכיל Literal. בסיום, צפוי שיהיה בידך הקוד הבא:

```
<asp:Repeater ID="Repeater1" runat="server">
  <ItemTemplate>
    <h3><asp:Literal ID="Literal1" runat="server"></asp:Literal></h3>
  </ItemTemplate>
</asp:Repeater>
```

3. קבע את המאפיין Text של הפקד Literal ל- <%# Eval("Name") %>. במקום הגרשיים הכפולים, ודא שאתה משתמש בגרשים יחידים כדי לתחום את ערך המאפיין:

```
<asp:Literal ID="Literal1" runat="server" Text='<%# Eval("Name") %>'></asp:Literal>
```

4. מתחת לאלמנט <h3>, גרור ושחרר פקד BulletedList מהקטגוריה Standard, והגדר עבורו את המאפיינים הבאים. באפשרותך להזין אותם ישירות במצב Markup View או להשתמש בטבלת המאפיינים.

המאפיין	ערך
ID	lstReviews
DataSource	<%# Eval("Reviews")%> (ודא שהנך משתמש בגרשים יחידים כדי לעטוף ערך מאפיין זה, כפי שמוצג בקטע הקוד הבא)
DataTextField	Title
DisplayMode	Text

אתה אמור לסיים עם הקוד הבא עבור הפקד:

```
<asp:BulletedList ID="lstReviews" runat="server" DataSource='<%# Eval("Reviews")%>'
  DataTextField="Title" DisplayMode="Text"></asp:BulletedList>
```

5. עבור ל- Design View. כדי לקבוע מטפל לאירוע Load של הדף, לחץ על הדף לחיצה כפולה במקום כלשהו בשטח האפור שהוגדר על ידי הדף הראשי לקריאה-בלבד. הכנס את הקוד הבא לתוך המטפל Page_Load:

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Me.Load
```

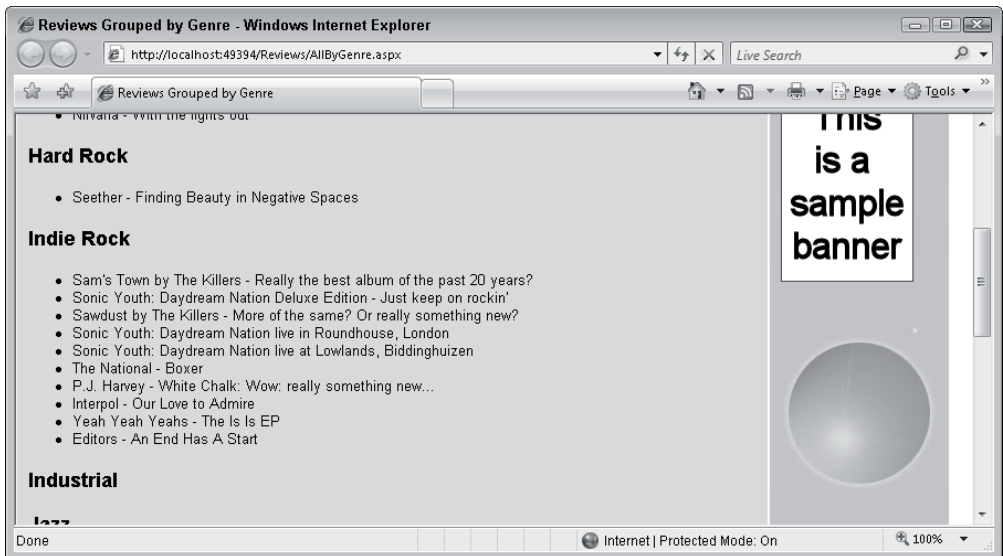
```
    Using myDataContext As New PlanetWroxDataContext()  
  
        Dim allGenres = From genre In myDataContext.Genres _  
                        Order By genre.Name _  
                        Select New With {genre.Name, genre.Reviews}  
  
        Repeater1.DataSource = allGenres  
        Repeater1.DataBind()  
  
    End Using
```

```
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)  
{
```

```
    using (PlanetWroxDataContext myDataContext = new PlanetWroxDataContext())  
    {  
        var allGenres = from genre in myDataContext.Genres  
                        orderby genre.Name  
                        select new { genre.Name, genre.Reviews };  
  
        Repeater1.DataSource = allGenres;  
        Repeater1.DataBind();  
    }  
}
```

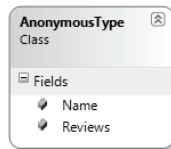


תרשים 13-5

6. שמור את השינויים שערכת בדף, ופתח אותו בדפדפן. כעת תראה תוצאה דומה לזו שמוצגת בתרשים 5-13, שבו כל סגנון מופיע ככותרת קבוצה מעל לרשימת הסקירות.

כיצד זה פועל

כדאי להתבונן בשני דברים חשובים בתרגיל זה. התבונן תחילה בשאילתת LINQ שמשמשת לקבלת הסגנונות והסקירות מבסיס הנתונים. שאילתה זו יוצרת סוג אנונימי חדש בעל שני מאפיינים: ה-Name של הסגנון כ-String, ואוסף אובייקטים של Review בשם Reviews. תרשים המחלקה עבור הסוג האנונימי החדש יכול להיראות דומה לתרשים 6-13.



תרשים 6-13

השדות Name ו-Reviews משמשים בחלק החשוב השני, הפקד Repeater עם רשימת התבליטים המקוננת. תחילה הבט ב-Repeater:

```
<asp:Repeater ID="Repeater1" runat="server">
  <ItemTemplate>
    <h3><asp:Literal ID="Literal1" runat="server"
      Text='%# Eval("Name") %'></asp:Literal></h3>
    <!-- BulletedList here -->
  </ItemTemplate>
</asp:Repeater>
```

למרות שלא השתמשת קודם בפקד Repeater, הוא עשוי להיראות מוכר מכיוון שהוא פועל באופן דומה לפקדי הנתונים האחרים. את סימני העריכה שרצונך שיחזרו על עצמם בכל פריט במקור הנתונים, עליך להגדיר בתוך <ItemTemplate>. באפשרותך לקבל את הערך של Title על ידי שימוש ב-Eval, ולהקצות אותו ל-Literal שעטוף בזוג תגיות <h3>. מבנה דומה משמש עבור BulletedList כדי להזין בו DataSource:

```
<asp:BulletedList ID="BulletedList1" runat="server"
  DataSource='%# Eval("Reviews")%'
  DataTextField="Title" DisplayMode="Text" />
```

בנוסף להקצאת מאפיינים פשוטים, כמו Text של Literal, מתוך Name של פריט הנתונים הבסיסי, ביכולתך להשתמש ב-Eval כדי לקבל מאפיינים מורכבים. בדוגמה שלנו השתמשנו ב-Eval("Reviews") לקבלת האוסף Reviews עבור ה-Genre הנוכחי. הפקד BulletedList יודע כיצד לטפל במקור נתונים זה, ואז הוא מאחזר את Title מכל אובייקט Review אינדיבידואלי ומציג אותו ברשימה. תרשים 7-13 מראה כיצד כל סגנון מכיל סקירה אחת או יותר שכותרותיהן מוצגות מתחת לשם הסגנון.

Name (from AnonymousType(0).Name)
Review Title (from AnonymousType(0).Reviews(0).Title)
Review Title (from AnonymousType(0).Reviews(1).Title)
Review Title (from AnonymousType(0).Reviews(2).Title)
Review Title (from AnonymousType(0).Reviews(3).Title)
Name (from AnonymousType(1).Name)
Review Title (from AnonymousType(1).Reviews(0).Title)
Review Title (from AnonymousType(1).Reviews(1).Title)
Name (from AnonymousType(2).Name)
Review Title (from AnonymousType(2).Reviews(0).Title)
Review Title (from AnonymousType(2).Reviews(1).Title)
Review Title (from AnonymousType(2).Reviews(2).Title)

תרשים 13-7

לאחר שקבעת את מבנה ה-Repeater והגדרת את השאילתה, עליך להתחיל בתהליך קשירת הנתונים. הדבר מתבצע על ידי הקצאת תוצאות השאילתה למאפיין DataSource של Repeater ובעקבות זאת קריאה ל-DataBind():

```
Repeater1.DataSource = allGenres;
Repeater1.DataBind();
```

שתי השורות האלו מניעות את הדברים. ברגע שקוראים ל-DataBind() מתבצעת השאילתה, והסגנונות והסקירות הרלוונטיים מאוחזרים מבסיס הנתונים. בדוגמה שלנו מאוחסנים הסגנונות לפי Name, אבל ברור שניתן לסדרם גם לפי עמודות אחרות, כמו SortOrder. ה-Repeater מבצע לולאה על כל פריט בקבוצת התוצאה (פריט זה הוא הסוג האנונימי שזה עתה ראית) ומשתמש ב-Name של הפריט כדי למלא את האלמנט <h3> בשם הסגנון. בצעד הבא מקצה Repeater את האוסף Reviews למאפיין DataSource של הפקד הפנימי BulletedList. פקד זה מבצע לולאה על פני מופעי Review הזמינים ומשתמש ב-Title שלהם כדי לבנות את רשימת התבליטים.

ייתכן שיעבור זמן מה עד שתבין במלואם את העקרונות שמאחורי שאילתות LINQ, אבל אני בטוח שתתחיל להעריך את כוחן ונגישותן. עם שורות קוד ספורות וכמה פקדים תוכל ליצור דפי ייצוג נתונים רבי-עוצמה. עם זאת, אפשר ליצור שאילתות LINQ ולהשתמש בהן עם פקדי שרת של ASP.NET, אפילו עם פחות שורות קוד. בסעיף הבא, שעוסק בפקדים החדשים ListView, LinqDataSource ו-DataPager, תראה כיצד זה פועל.

שימוש בפקדי שרת עם שאילתות LINQ

עד כה ראית דרך אחת לקשור את תוצאות שאילתת LINQ לפקד בדף ASPX: להקצות את הנתונים למאפיין DataSource של הפקד ואז לקרוא ל-DataBind. לדרך זו של הבאת נתונים לתוך פקדים יש כמה חסרונות. ראשית, שיטה זו אינה תומכת בעריכה, עדכון ומחיקה ישירה של נתונים. שנית, מכיוון שמקור הנתונים מוגדר בקוד ברקע, ה-GridView אינו יודע עד לזמן הביצוע באיזה נתונים הזינו אותו, ולכן אין כלי לתמיכה בהגדרת העמודות שלו. קל להתגבר על חסרונות אלה על ידי שימוש בפקדי השרת החדשים, וביניהם הפקדים ListView ו-LinqDataSource.

פקדים חדשים שהוצגו ב- ASP.NET 3.5

יחד עם LINQ, הציגה מיקרוסופט ב- ASP.NET 3.5 שלושה פקדים חדשים שמאפשרים לבנות ממשקי אינטרנט עשירים בזמן קצר ביותר ועם מעט מאוד קוד, אם בכלל. שניים מהם מספקים ממשק חזותי בדפי ASP.NET, בעוד השלישי פועל כגשר בין הפקדים קשורי-הנתונים ומקור הנתונים הבסיסי. הטבלה הבאה מציגה בקצרה את שלושת הפקדים החדשים.

פקד	תיאור
LinqDataSource	כמו עם SqlDataSource שראית בפרקים קודמים, גם הפקד LinqDataSource פועל כגשר בין הפקדים קשורי-הנתונים לבין מקורות הנתונים הבסיסיים. מקור נתונים זה יכול להיות כל אחד מהסוגים שבהם תומכת LINQ, כולל בסיסי נתונים מסוג SQL SERVER, מסמכי XML, ואפילו מערכים ואוספים. פרק זה עוסק בפקד LinqDataSource מנקודת המבט של LINQ to SQL בלבד.
ListView	הפקד ListView מספק ממשק חזותי שמאפשר להציג, להכניס, לערוך ולמחוק פריטים בבסיס נתונים, תוך מתן שירותי CRUD מלאים.
DataPager	משתמשים בפקד DataPager ביחד עם ListView, אשר מאפשר לעמד את הנתונים במקור הנתונים ולהזין למשתמש מדי פעם כמויות נתונים קטנות יותר, ולא מספר רב של רשומות בבת אחת.

הסעיפים הבאים מספקים מידע נוסף על פקדים חדשים אלה, ובכמה תרגילי נסה זאת הבאים תלמד כיצד להשתמש בהם.

הפקד LinqDataSource

כפי שמשמע משמו, הפקד LinqDataSource הוא קרוב משפחה של SqlDataSource ושל פקדי מקור נתונים אחרים. היחס בין הפקד LinqDataSource לבין LINQ הוא כמו היחס בין הפקד SqlDataSource לבין מקורות נתונים מבוססי-SQL. הפקד מספק דרך הצהרתית לגשת למקורות נתונים מאופשרי-LINQ. בדיוק כמו הפקד SqlDataSource, גם הפקד LinqDataSource מאפשר גישה קלה לפעולות CRUD, והוא גם מקל מאוד על מיון וסינון נתונים. הטבלה הבאה מתארת את המאפיינים העיקריים של פקד חדש זה. התיאורים אמורים לתת לך מושג טוב על יכולות הפקד.

מאפיין	תיאור
EnableDelete EnableInsert EnableUpdate	מציין אם הפקד מספק יכולות אוטומטיות למחיקה, הכנסה ועדכון נתונים אם לאו. כשהוא במצב מאופשר, תוכל להשתמש בפקד יחד עם פקדים קשורי-נתונים כמו GridView כדי לתמוך בניהול נתונים. תראה זאת בפעולה מאוחר יותר.
ContextTypeName	השם של המחלקה DataContext שעל הפקד להשתמש בה. בדוגמאות שבספר זה, שם הסוג הזה הוא PlanetWroxDataContext.

מאפיין	תיאור
TableName	שם הטבלה מתרשים LINQ to SQL שברצונך להשתמש בה, למשל Reviews.

ביחד עם פקד קשור-נתונים מספק LinqDataSource באמצעות LINQ גישה מלאה לבסיס הנתונים SQL SERVER. התרגיל הבא יראה לך כיצד להשתמש בפקד בדפי ASPX.

נסה זאת יישום פשוט של LinqDataSource

בתרגיל זה תתחיל לבנות את היכולת Gig Pics של Planet Wrox, חלק של אתר האינטרנט שבו המשתמשים יכולים להעלות את התמונות שצילמו בזמן הופעות של הלהקות האהובות עליהם. תלמד כיצד לאפשר למשתמש ליצור אלבום תמונות חדש שפועל כשומר מקום עבור התמונות שמועלות. תלמד כיצד להשתמש ב-LinqDataSource וב-DetailsView כדי ליצור ממשק משתמש שמאפשר למשתמש להכניס את שם האלבום למערכת. בתרגילים מאוחרים יותר תראה כיצד להוסיף תמונות לאלבום. בסיס הנתונים יכיל רק הפניה לתמונה; התמונה עצמה תאוחסן בדיסק, כפי שתראה מאוחר יותר.

- הוסף את שתי הטבלאות הבאות לבסיס הנתונים שלך תוך שימוש בחלון סייר בסיס הנתונים. לפרטים נוספים בדבר יצירת טבלאות, מפתחות ראשיים, ועמודות זהות, פנה לפרק 11. אם אין ברצונך ליצור את הטבלאות באופן ידני, תוכל להשתמש בבסיס הנתונים שמגיע עם הקוד המקור של פרק זה.

אלבום תמונות

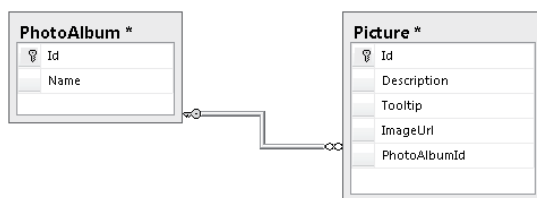
שם עמודה	סוג נתונים	מאפשר Nulls?	תיאור
Id	Int	No	ID ייחודי (זהות ומפתח ראשי) של אלבום התמונות.
Name	nvarchar(100)	No	שם אלבום התמונות.

תמונה

שם עמודה	סוג נתונים	מאפשר Nulls?	תיאור
Id	int	No	ID ייחודי (זהות ומפתח ראשי) של התמונה.
Description	nvarchar(300)	No	טקסט קצר שמתאר את התמונה.
Tooltip	nvarchar(50)	No	תיאור כלי שמוצג כשעוברים על התמונה.
ImageUrl	nvarchar(200)	No	הנתיב הווירטואלי אל התמונה בדיסק.
PhotoAlbumId	int	No	ID של אלבום התמונות שתמונה זו שייכת אליו.

עבור שתי הטבלאות, הגדר את העמודה Id כמפתח ראשי, על ידי לחיצה עליה פעם אחת ואז לחיצה על הסמל עם המפתח הצהוב בסרגל הכלים Table Designer. בנוסף, הפוך עמודה זו לעמודת הזהות (Identity) של הטבלה על ידי הגדרת המאפיין Is Identity שלה ב- Column Properties Grid ל-Yes. פנה לפרק 11 אם אינך זוכר כיצד לעשות זאת.

2. בסייר בסיס הנתונים, פתח את תרשים בסיס הנתונים שיצרת בפרק 11, והוסף אליו את שתי הטבלאות החדשות (כדי לעשות זאת, לחץ לחיצה ימנית על התרשים ובחר Add PhotoAlbumId אל העמודה Id מהטבלה PhotoAlbum אל העמודה PhotoAlbumId של הטבלה Picture. לחץ OK פעמיים כדי לגרום למעצב ליצור קשר בין הטבלה PhotoAlbum לבין הטבלה Picture, כפי שמוצג בתרשים 13-8.

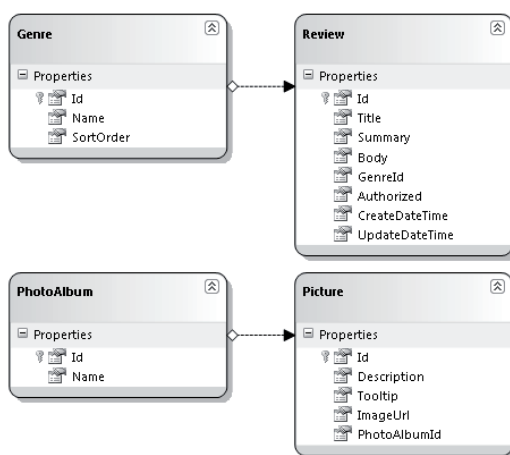


תרשים 13-8

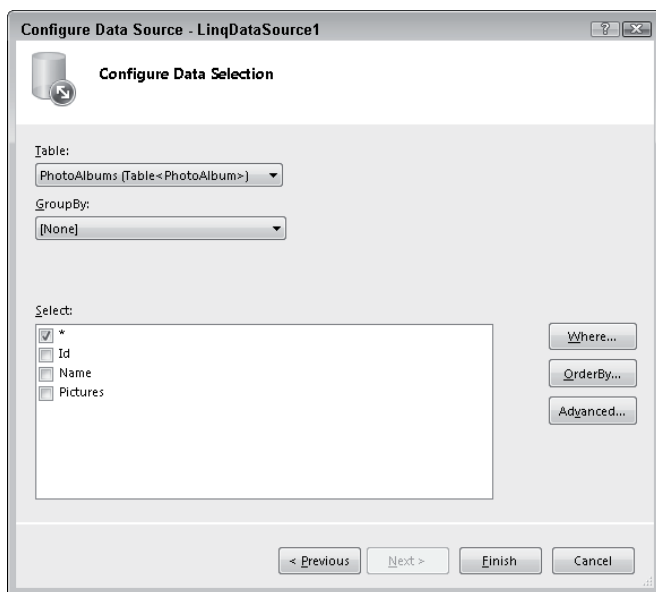
3. שמור וסגור את התרשים. לחץ OK כדי לאשר את השינויים שעומדים להיערך בטבלאות PhotoAlbum ו-Picture.

4. מהתיקייה App_Code, פתח את הקובץ PlanetWrox.dbml של LINQ to SQL Classes. מחלון סייר בסיס הנתונים גרור את שתי הטבלאות החדשות אל התרשים. התרשים שלך אמור להיראות כעת כמו תרשים 13-9. שמור את השינויים וסגור את התרשים.

5. צור קובץ ASPX חדש בשורש האתר, וקרא לו NewPhotoAlbum.aspx. ודא שהוא מבוסס על התבנית המותאמת אישית שגורמת למחלקה קוד ברקע לרשת מהמחלקה BasePage. קבע לדף את הכותרת Create New Photo Album.



תרשים 13-9

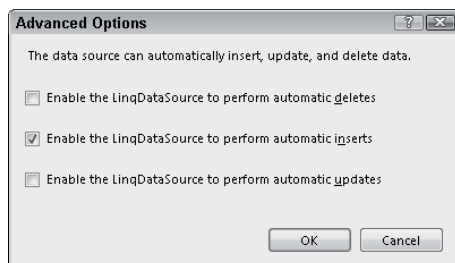


תרשים 13-10

6. העבר את הדף למצב Design View. מהקטגוריה Data של ארגז הכלים גרור פקד DetailsView ושחרר אותו בשומר המקום cpMainContent. פתח את הלוח Smart Tasks של הפקד DetailsView, ומהרשימה הנפתחת Choose Data Source בחר <New data source>. בתיבת הדו-שיח Data Source Configuration Wizard לחץ על הסמל LINQ ואז לחץ OK. בתיבת הדו-שיח Choose a Context Object, ודא שמחלקה PlanetWroxDataContext נבחרת, ולחץ Next כדי לעבור למסך Configure Data Selection, שמוצג בתרשים 13-10. מהרשימה הנפתחת בחר Table PhotoAlbums.

תיבת דו-שיח זו פועלת בצורה דומה לאשף SqlDataSource שראית בפרקים קודמים. היא מאפשרת לבחור טבלה מהרשימה הנפתחת Table. לצורך תרגיל זה ודא שמהרשימה הנפתחת בחרת PhotoAlbums. בצד הימני מופיעים לחצנים שתפקידם ליצור מסננים (תוך שימוש בלחצן Where), לקבוע את סדר המיון (תוך שימוש בלחצן OrderBy), ולומר ל-LinqDataSource אם לאפשר, או לא לאפשר, הכנסה, עדכון ומחיקה של נתונים.

⊙ לצורך תרגיל זה נאפשר הכנסת נתונים. לפיכך, לחץ על Advanced ואפשר הכנסות, כפי שמוצג בתרשים 13-11.



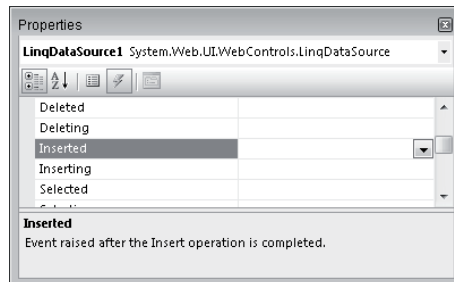
תרשים 13-11

⊙ לחץ OK כדי לסגור את המסך Advanced Options, ולחץ Finish כדי לסגור את אשף התצורה.

7. אפשר לפקד DetailsView לבצע הכנסה, על ידי סימון הפריט Enable Inserting בלוח Smart Tasks של הפקד.

8. פתח את טבלת המאפיינים של הפקד DetailsView ושנה את DefaultMode ממצב Insert ל-ReadOnly.

9. ב- Design View, בחר בפקד LinqDataSource, פתח את טבלת המאפיינים שלו, ועבור לקטגוריה Events. לחץ לחיצה כפולה על האירוע Inserted, כפי שמוצג בתרשים 12-13.



תרשים 12-13

10. הכנס את הקוד הבא אל מטפל האירוע. הקוד מכוון את המשתמש לדף חדש ברגע שבוצעה פעולת Insert לאלבום התמונות בבסיס הנתונים.

VB.NET

```
Protected Sub LinqDataSource1_Inserted(ByVal sender As Object, _  
    ByVal e As System.Web.UI.WebControls.LinqDataSourceStatusEventArgs) _  
    Handles LinqDataSource1.Inserted  
    Dim myPhotoAlbum As PhotoAlbum = CType(e.Result, PhotoAlbum)  
    Response.Redirect(String.Format("ManagePhotoAlbum.aspx?PhotoAlbumId={0}", _  
        myPhotoAlbum.Id.ToString()))  
End Sub
```

C#

```
protected void LinqDataSource1_Inserted(object sender, LinqDataSourceStatusEventArgs e)  
{  
    PhotoAlbum myPhotoAlbum = (PhotoAlbum)e.Result;  
    Response.Redirect(String.Format("ManagePhotoAlbum.aspx?PhotoAlbumId={0}",  
        myPhotoAlbum.Id.ToString()));  
}
```

11. שמור את כל השינויים, ובקש את הדף NewPhotoAlbum.aspx בדפדפן. הזן שם חדש לאלבום התמונות כמו Sonic Youth Playing Daydream Nation Live in London ולחץ על הלחצן Insert. כעת תקבל הודעת שגיאה Page not found (מפני שטרם יצרת את ManagePhotoAlbum.aspx), אבל תוכל לראות את ה-ID של אלבום התמונות בסרגל הכתובת של הדפדפן.

כיצד זה פועל

התחלת תרגיל זה בהוספת הטבלאות Picture ו-PhotoAlbum הן לבסיס הנתונים והן למחלקה LINQ to SQL. טבלאות אלו משמשות לאחסון נתונים אודות אלבומי תמונות והתמונות שהם מכילים. כל אחת מהתמונות שייכת ל-PhotoAlbum שאליו היא פונה על ידי PhotoAlbumId שלה שמצביע על העמודה Id של הטבלה PhotoAlbum בבסיס הנתונים. הטבלה Picture עוצבה כדי להחזיר נתונים אודות התמונה בלבד. קובץ התמונה עצמו מאוחסן בדיסק כפי שתראה מאוחר יותר.

כדי לאפשר למשתמשים ליצור אלבום תמונות חדש, הוספת לדף פקד DetailsView. כדי לוודא שניתן להשתמש בפקד להכנסת אלבומי תמונות חדשים אפשרת הכנסה, ואז הגדרת את DefaultMode ל-Insert. הדבר כופה על הפקד לעבור למצב insert במקום מצב ברירת המחדל שהוא קריאה-בלבד. אחר כך חיברת LinDataSource לפקד DetailsView אשר דואג להכנסת אלבום התמונות לטבלה PhotoAlbum.

הקוד עבור הפקד LinDataSource נראה דומה לזה:

```
<asp:LinDataSource ID="LinDataSource1" runat="server"
    ContextTypeName="PlanetWroxDataContext" EnableInsert="True"
    OnInserted="LinDataSource1_Inserted" TableName="PhotoAlbums">
</asp:LinDataSource>
```

אם אתה משתמש ב-VB.NET, הקוד לא יכיל הגדרה של התכונה OnInserted. שים לב עד כמה ישיר LinDataSource בתרחיש זה: כיוונת אותו אל DataContext, ובדוגמה שלנו אל PlanetWroxDataContext, אפשרת הכנסה על ידי הגדרת EnableInsert ל-True, והגדרת את TableName, כדי שהפקד ידע באיזו טבלה ממודל האובייקטים להשתמש. להכנסת נתונים פשוטה, זה כל מה שעליך לעשות. כשהדף נטען בדפדפן, ה-DetailsView מרנדר ממשק משתמש שמאפשר להזין שם חדש עבור אלבום התמונות. כשלוחצים Insert, נאספים הנתונים שהוזנו ומועברים ל-LinDataSource. פקד זה, בתורו, יוצר מופע חדש של PhotoAlbum ושומר אותו בבסיס הנתונים על ידי שליחת משפט INSERT של SQL לבסיס הנתונים.

התנהגות רגילה זו אינה מספקת במצבים רבים. ייתכן שתצטרך לאמת את הנתונים שמוכנסים, או שתצטרך לשנות את הנתונים הממשיים לפני שהם נשלחים לבסיס הנתונים. דוגמה למקרה האחרון תראה בתרגיל נסה זאת מאוחר יותר, כשתעלה תמונות לשרת. דרישה שכיחה אחרת היא אחזור ID של הפריט החדש שנוצר, שנשלח אל הדף הבא. הדוגמה משתמשת בקוד שלהלן כדי להשיג זאת:

VB.NET

```
Dim myPhotoAlbum As PhotoAlbum = CType(e.Result, PhotoAlbum)
Response.Redirect(String.Format("ManagePhotoAlbum.aspx?PhotoAlbumId={0}", _
    myPhotoAlbum.Id.ToString()))
```

C#

```
PhotoAlbum myPhotoAlbum = (PhotoAlbum)e.Result;
Response.Redirect(string.Format("ManagePhotoAlbum.aspx?PhotoAlbumId={0}",
    myPhotoAlbum.Id.ToString()));
```

הדבר היפה בפקד `LinqDataSource` הוא שביכולתו לפעול עם אובייקטים בעלי סוג מובהק, כשהמבנה ממופה לטבלאות שהוספת למודל האובייקטים. במקרה זה אתה פועל עם מופעים ממשיים של `PhotoAlbum`, המחלקה שמייצגת את אלבומי התמונות באתר האינטרנט. הדבר מאפשר לאחזר את אלבום התמונות שהכנסת לבסיס הנתונים באירוע `Insert` של פקד מקור הנתונים. הארגומנט 'e' חושף מאפיין בשם `Result` שמכיל הפניה לאלבום התמונות החדש. על ידי `casting` (המרה) של המאפיין לאובייקט `PhotoAlbum` ממשי (תוך שימוש ב-`CType` ב-`VB.NET`, או כתיבת שם המחלקה בסוגריים לפניו ב-`C#`), תוכל לגשת למאפיינים של `PhotoAlbum`, כולל ה-ID החדש שלו שנוצר על ידי בסיס הנתונים (על ידי הגדרות `Identity` בעמודה ID) ואוחסן במאפיין `Id` של ה-`PhotoAlbum`. השורה האחרונה במטפל האירוע מעבירה את המשתמש לדף הבא ושולחת את ה-ID של אלבום התמונות החדש במחרוזת השאילתה.

כעת, כשאתה יכול להכניס אלבומי תמונות חדשים, הצעד הלוגי הבא הוא הוספת תמונות לאלבום התמונות. בתרגיל הבא תלמד כיצד ליצור ממשק משתמש עם הפקד `ListView` שמאפשר למשתמש להעלות תמונות חדשות לאלבום התמונות.

הצגת הפקד `ListView`

עד כה ראית כמה פקדים קשורי-נתונים בפעולה. ראית את `GridView` רב העוצמה שתומך בעדכונים, מחיקות, מיון ודפדוף נתונים, אבל חסר יכולת הכנסת נתונים ומייצר כמות גדולה של סימני עריכה של HTML. ראית גם את הפקד `Repeater` שנותן שליטה מדויקת על קוד HTML שנוצר, אבל חסר את רוב התכונות המתקדמות שיש לשאר פקדי הנתונים כמו עדכון ומחיקה, ויכולות מיון וסינון. ראית גם את `DetailsView` שמאפשר להכניס או לעדכן רשומה אחת בכל פעם.

הפקד `ListView` הוא השילוב המנצח: "הטוב שבכל העולמות". הוא משלב את קבוצת התכונות העשירה של `GridView` עם השליטה בסימני העריכה שנותן `Repeater`. פקד `ListView` מאפשר להציג נתונים במגוון עיצובים, כולל טבלה (שורות ועמודות כמו `GridView`), כרשימת תבליטים (בדומה לאופן שבו הגדרת את `Repeater` מוקדם יותר בפרק), ובפורמט `Flow`, שבו כל הפריטים מוצגים ב-HTML בזה אחר זה, וההחלטה בידך אם לכתוב CSS שיעצב את הנתונים.

פקד `ListView` מציג ומנהל את הנתונים שלו באמצעות תבניות שמאפשרות שליטה ברבות מהתצוגות השונות של הנתונים הבסיסיים שהוא מציג. הטבלה הבאה מתארת את כל התבניות הזמינות שביכולתך להוסיף בסימני העריכה של הדף כתולדה ישירה של הפקד `ListView`.

תבנית	תיאור
<LayoutTemplate>	התבנית משמשת את הפקד כמכולה. מאפשרת להגדיר את המקום שיוצבו בו כל הפריטים היחידים (למשל, <code>Reviews</code>). פריטי הנתונים, שמוצגים על ידי <code>ItemTemplate</code> ו- <code>AlternatingItemTemplate</code> מוספים כבנים של מכולה זו. יותר מאוחר תראה כיצד זה פועל.

תבנית	תיאור
<code><ItemTemplate></code> <code><AlternatingItemTemplate></code>	התבניות מגדירות לפקד מצב לקריאה-בלבד. כשמשמשים בהן יחד, הן מאפשרות ליצור "אפקט זכרה" כשורות זוגיות ואי-זוגיות מקבלות מראה שונה (בדרך כלל צבע רקע שונה).
<code><SelectedItemTemplate></code>	התבנית מאפשרת להגדיר את המראה והתחושה של הפריט הפעיל הנוכחי, או של הפריט הנבחר.
<code><InsertItemTemplate></code> <code><EditItemTemplate></code>	תבניות אלו מאפשרות להגדיר את ממשק המשתמש עבור הכנסה ועדכון של פריטים ברשימה. בדרך כלל תציב בתבניות אלו פקדים כמו תיבות טקסט, רשימות נפתחות ופקדי שרת אחרים, ותקשור אותם למקור הנתונים הבסיסי. יותר מאוחר תראה כיצד זה פועל.
<code><ItemSeparatorTemplate></code>	התבנית מגדירה את סימני העריכה שמוצבים בין הפריטים ברשימה. שימושית כשברצונך להוסיף בין הפריטים שורה, תמונה או כל סימן עריכה אחר.
<code><EmptyDataTemplate></code>	הגדרת תבנית הנראית כאשר לפקד אין נתונים להצגה. באפשרותך להוסיף לה טקסט או סימני עריכה ופקדים אחרים, וכך לומר למשתמשים שאין נתונים להצגה.
<code><GroupTemplate></code> <code><GroupSeparatorTemplate></code> <code><EmptyItemTemplate></code>	התבניות משמשות בתרחישי תצוגה מתקדמים, כשהנתונים יכולים להיות מוצגים בקבוצות שונות.

למרות שרשימת תבניות ארוכה זו רומזת כביכול שעליך לכתוב קוד רב כדי להשתמש ב-`ListView`, אין זה כך. ראשית, `Visual Web Developer 2008` יוצרת את רוב הקוד על פי הנתונים שנחשפים על ידי פקדים כמו `LinQDataSource`. שנית, לא תמיד אתה זקוק לכל התבניות, והדבר מאפשר להקטין את כמות הקוד של הפקד. בנוסף לתבניות הרבות, יש לפקד מאפיינים נוספים שתשתמש בהם כדי להשפיע על התנהגותו.

מאפיין	תיאור
<code>ItemPlaceholderID</code>	מציין ID של פקד צד-שרת שמוצב בתוך <code>LayoutTemplate</code> . כשהפקד שמאפיין זה מתייחס אליו מוצג על המסך, הוא מוחלף על ידי כל פריטי הנתונים החוזרים. הפקד יכול להיות פקד שרת כמו <code><asp:PlaceHolder></code> או אלמנט פשוט של HTML בעל ID תקף, שמאפיין שלו מוגדרת ל- <code>server</code> (לדוגמה, <code><ul runat="server" id="MainList"></code>). אם מאפיין זה אינו מוגדר, <code>ASP.NET</code> תנסה למצוא פקד עם ID של <code>itemPlaceholder</code> ולהשתמש בו במקום.

מאפיין	תיאור
DataSourceID	ID של פקד מקור נתונים בדף, כמו LinqDataSource או SqlDataSource.
InsertItemPosition	ערך מאפיין זה נקבע לאחד מערכי המנייה (enumeration) הבאים: FirstItem ו-LastItem, ומאפשר לקבוע את המיקום של InsertItemTemplate: או בתחילת הרשימה, או בסופה או שהתבנית לא תיראה כלל.

לפקד ListView, כמו לפקדים קשורי-נתונים אחרים, יש כמה אירועים שמתעוררים ברגעים מיוחדים במחזור החיים שלו. לדוגמה, האירועים ItemInserting ו-ItemInserted שמתעוררים מייד לפני ומייד אחרי שפריט הוכנס למקור הנתונים הבסיסי. באופן דומה יש לפקד אירועים שמתעוררים מייד לפני ומייד אחרי העדכון והמחיקה של נתונים. בפרק הבא תלמד יותר על הטיפול בסוגי אירועים אלה.

פקד ListView מציע יותר מאשר התבניות, המאפיינים והאירועים שראית זה עתה. להסבר מפורט על הפקד ListView והתנהגותו, פנה לתיעוד MSDN.

התרגיל הבא מראה כיצד לשלב את כל המידע שקיבלת עד כה. תלמד כיצד להגדיר את התבניות השונות ואת המאפיינים הרלוונטיים כדי לשלוט במראה ובתחושה של הפקד ListView.

נסה זאת הכנסה ועדכון של נתונים על ידי הפקד ListView

הכנסת פריטים על ידי ListView קלה בדיוק כמו הכנסתם על ידי DetailsView. כוון את הפקד למקור נתונים, והנח ל-VWD ליצור עבורך את התבניות הנחוצות. עם זאת, באתרים אמיתיים רבים ברשת האינטרנט, תבניות ברירת מחדל אלו אינן עונות לכל הצרכים. אתה עשוי לרצות להציג פחות שדות מכפי שזמינים במקור הנתונים, או שתמצה לאמת את תוקף הנתונים לפני שהם נשלחים לבסיס הנתונים, או שתמצה לאחסן נתונים במקום אחר ולא בבסיס הנתונים. לדוגמה, ייתכן שתמצה לאחסן בדיסק את התמונות שמתקבלות ולא בבסיס הנתונים, ובטבלת בסיס הנתונים לאחסן רק הפניה לקובץ. לכן, התרגיל הבא יראה לך כיצד להתאים אישית את תבניות ListView, וכיצד לטפל באירוע Inserting של LinqDataSource כדי לתפוס את התמונה שתוריד מהרשת ולאחסן אותה בדיסק.

בתרגיל זה תשתמש בקוד רב שנוצר באופן אוטומטי על ידי VWD. רוב פעולותיך בתרגיל זה תהיינה הסרת קוד ולא הוספת קוד חדש. אם תאבד את דרכך בנקודה כלשהי, או אם תרגיש שהקוד שלך אינו נראה כפי שהיה צריך להיראות, זכור שתוכל תמיד להוריד מאתר Wrox ומאתר הוד-עמי את קוד המקור בשלמותו.

1. בשורש האתר, צור דף חדש בשם ManagePhotoAlbum.aspx, ובסס אותו על התבנית המותאמת אישית שלך. קבע את הכותרת שלו ל- Manage Photo Album. עבור אל Design View.

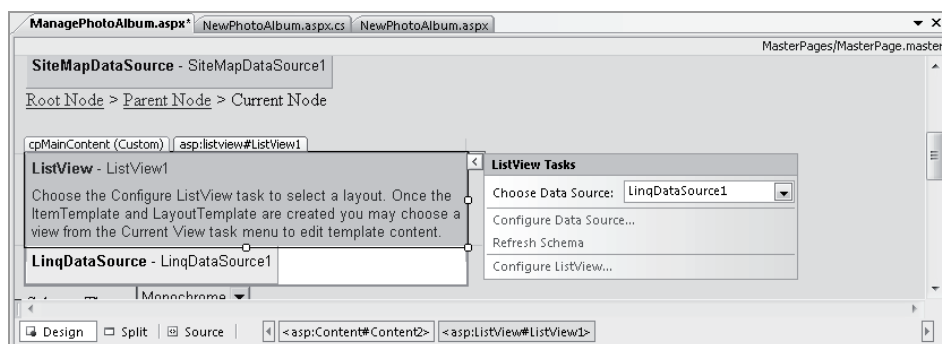
2. גרור פקד ListView מארגז הכלים אל הדף, ושחרר אותו בשומר המקום cpMainContent. חבר את ListView לפקד LinqDataSource על ידי פתיחת לוח Smart Tasks של ListView, ובחירת <New data source> ברשימה הנפתחת Choose Data Source (בדיוק כפי שעשית קודם עם DetailsView). ודא שבחרת ב-PlanetWroxDataContent כאובייקט התוכן. ב-Configure Data Selection, שמוצגת בתרשים 13-13, בחר בטבלה Pictures מהרשימה הנפתחת Table.

3. לחץ על הלחצן Where כדי להגדיר פסוקית Where להגבלת התמונות ברשימה רק לאלו ששייכות לאלבום תמונות בעל ID ייחודי. מכיוון שהדף ManagePhotoAlbum.aspx מקבל את ID של אלבום התמונות באמצעות מחרוזת השאילתה, עליך להגדיר בצעד זה QueryStringParameter. מלא את תיבת הדו-שיח Configure Where Expression שמוצג בתרשים 13-13.

תרשים 13-13

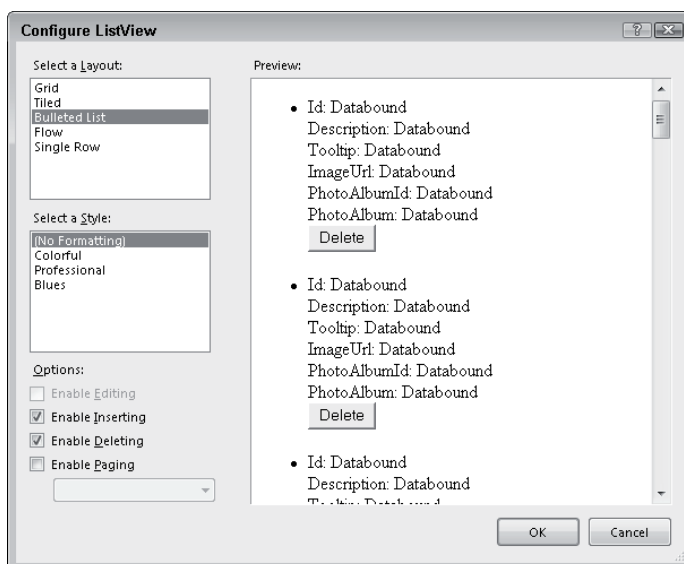
4. כשתסיים, לחץ על הלחצן Add כדי להוסיף את הפרמטר לרשימה שבתחתית המסך, ואז לחץ OK כדי לשחרר את תיבת הדו-שיח.

5. חזור למסך Configure Data Selection, לחץ על הלחצן Advanced, סמן את שני הפריטים הראשונים, שנותנים ל-LinqDataSource תמיכה במחיקה והכנסה, ולחץ OK לאישור. לבסוף, לחץ Finish כדי לסגור את האשף Configure Data Source. חזור אל הדף, וכעת ListView אמור להופיע כמלבן פשוט, כפי שמוצג בתרשים 13-14, מכיוון שטרם סיפקת מידע תבנית כלשהו.



תרשים 13-14

6. בלוח Smart Tasks של ListView בחר Configure ListView. לפניך תיבת דו-שיח שמאפשרת לבחור את העיצוב והסגנון של הפקד, ולציין אם ברצונך לאפשר פעולות כמו הכנסה ועדכון. בחר בעיצוב Bulleted List, וסמן את הפריטים Enable Inserting ו-Enable Deleting. תסיים עם תיבת דו-שיח שדומה לזו שמוצגת בתרשים 13-15.



תרשים 13-15

7. לחץ OK כדי לסגור את תיבת הדו-שיח. אם תקבל תיבת דו-שיח ששואלת אם ליצור מחדש את הפקד ListView, לחץ Yes.
8. עבור ל- Markup View והסר את הקוד עבור התבניות שרשומות מטה. כדי לעשות זאת בדרך הקלה ביותר, לחץ פעם אחת על תגית הפתיחה הרלוונטית, ואז לחץ על התגית שבבורר התגיות שבתחתית חלון המסמך כדי לבחור באלמנט כולו ובתוכנו, ואז הקש על המקש Delete.

⊙ <AlternatingItemTemplate>

- ⊙ <SelectedItemTemplate>
- ⊙ <EmptyDataTemplate>
- ⊙ <EditItemTemplate>
- ⊙ <ItemSeparatorTemplate>

9. אתר את האלמנט ב-LayoutTemplate והסר את המאפיינים ID, runat ו-style. הוסף מאפיין class והגדר אותו ל-itemContainer. תוכל להסיר גם את האלמנט <div> הריק ש-VWD הוסיפה. <LayoutTemplate> מכילה כעת את הקוד הבא:

```
<LayoutTemplate>
  <ul class="itemContainer">
    <li ID="itemPlaceholder" runat="server" />
  </ul>
</LayoutTemplate>
```

10. אתר את ItemTemplate והסר את השורות שיוצרות את העמודות ID, PhotoAlbumId ו-PhotoAlbum, שמוצללות בקטע הקוד שלהלן, מכיוון שאינך זקוק להן. הקפד לא למחוק בטעות את תגית הפתיחה :

```
<li style="">
  Id:
  <asp:Label ID="IdLabel" runat="server" Text='<## Eval("Id") %>' />
  <br />
  Description:
  <asp:Label ID="DescriptionLabel" runat="server" Text='<## Eval("Description") %>' />
  <br />
  Tooltip:
  <asp:Label ID="TooltipLabel" runat="server" Text='<## Eval("Tooltip") %>' /><br />
  ImageUrl:
  <asp:Label ID="ImageUrlLabel" runat="server" Text='<## Eval("ImageUrl") %>' /><br />
  PhotoAlbumId:
  <asp:Label ID="PhotoAlbumIdLabel" runat="server" Text='<## Eval("PhotoAlbumId") %>' />
  <br />
  PhotoAlbum:
  <asp:Label ID="PhotoAlbumLabel" runat="server" Text='<## Eval("PhotoAlbum") %>' />
  <br />
  <asp:Button ID="DeleteButton" runat="server" CommandName="Delete" Text="Delete" />
</li>
```

11. חזור על הצעד הקודם עבור InsertItemTemplate שגם היא חלק מסימני העריכה של הפקד ListView. לתבנית זו אין שדה ID ולכן עליך להסיר רק את PhotoAlbumId ואת PhotoAlbum. הפקד השלם אמור להיראות כעת כך (לאחר עיצוב מסוים של מקור HTML לשיפור הקריאות):

```

<asp:ListView ID="ListView1" runat="server" DataKeyNames="Id"
    DataSourceID="LinqDataSource1" InsertItemPosition="LastItem">
    <LayoutTemplate>
        <ul class="itemContainer">
            <li ID="itemPlaceholder" runat="server" />
        </ul>
    </LayoutTemplate>
    <InsertItemTemplate>
        <li style="">
            Description:
            <asp:TextBox ID="DescriptionTextBox" runat="server"
                Text='<%= Bind("Description") %>' /><br />
            Tooltip:
            <asp:TextBox ID="TooltipTextBox" runat="server" Text='<%= Bind("Tooltip") %>' />
            <br />
            ImageUrl:
            <asp:TextBox ID="ImageUrlTextBox" runat="server" Text='<%= Bind("ImageUrl") %>' />
            <br />
            <asp:Button ID="InsertButton" runat="server" CommandName="Insert" Text="Insert" />
            <asp:Button ID="CancelButton" runat="server" CommandName="Cancel" Text="Clear" />
        </li>
    </InsertItemTemplate>
    <ItemTemplate>
        <li style="">Description:
            <asp:Label ID="DescriptionLabel" runat="server" Text='<%= Eval("Description") %>' />
            <br />
            Tooltip:
            <asp:Label ID="TooltipLabel" runat="server" Text='<%= Eval("Tooltip") %>' />
            <br />
            ImageUrl:
            <asp:Label ID="ImageUrlLabel" runat="server" Text='<%= Eval("ImageUrl") %>' />
            <br />
            <asp:Button ID="DeleteButton" runat="server" CommandName="Delete" Text="Delete" />
        </li>
    </ItemTemplate>
</asp:ListView>

```

12. עבור למצב Design View, בחר בפקד LinqDataSource ופתח את טבלת המאפיינים שלו. עבור לקטגוריית Events ולחץ לחיצה כפולה על האירוע Inserting. הכנס את הקוד הבא למטפל האירוע שנוסף על ידי VWD:

VB.NET

```

Protected Sub LinqDataSource1_Inserting(ByVal sender As Object, _
    ByVal e As System.Web.UI.WebControls.LinqDataSourceInsertEventArgs) _
    Handles LinqDataSource1.Inserting
    Dim myPicture As Picture = CType(e.NewObject, Picture)
    myPicture.PhotoAlbumId = Convert.ToInt32(Request.QueryString.Get("PhotoAlbumId"))
End Sub

```

C#

```
protected void LinqDataSource1_Inserting(object sender, LinqDataSourceInsertEventArgs e)
{
    Picture myPicture = (Picture)e.NewObject;
    myPicture.PhotoAlbumId = Convert.ToInt32(Request.QueryString.Get("PhotoAlbumId"));
}
```

13. בתיקייה Styles, צור קובץ גיליון סגנון חדש בשם Styles.css. החלף את קוד CSS שבקובץ בקוד שמוצג להלן:

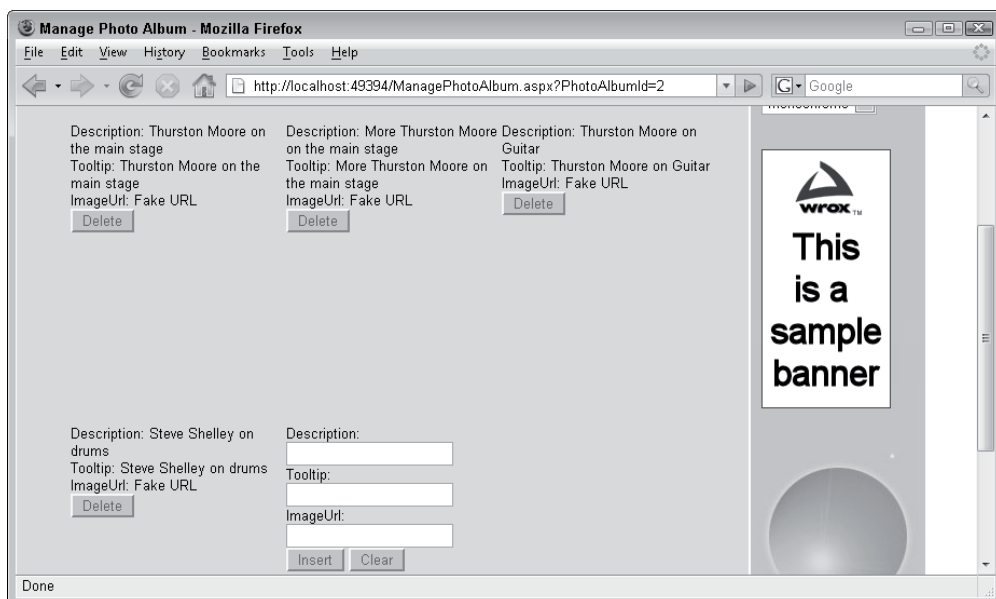
```
.itemContainer
{
    width: 600px;
    list-style-type: none;
}
.itemContainer li
{
    height: 280px;
    width: 200px;
    float: left;
}
.itemContainer li img
{
    width: 180px;
    margin: 10px 20px 10px 0;
}
```

14. שמור וסגור את הקובץ.

15. מהתיקייה MasterPages, פתח את הדף הראשי של האתר. הקפד לפתוח את MasterPage.master ולא את הדף הראשי של החלק Management של האתר. עבור ל- Design View וגרור את הקובץ Styles.css מסייר הפתרון אל הדף הראשי. הדבר מבטיח שכל הדפים באתר יקבלו הפניה לגיליון סגנון חדש זה, מבלי להתחשב בערכת הנושא שבשימוש.

16. שמור את כל השינויים, סגור את כל הקבצים הפתוחים, ואחר כך קרא ל-NewPhotoAlbum.aspx על ידי הדפדפן. ודא שאינך פותח בטעות את ManagePhotoAlbum.aspx מכיוון שהאחרון דורש מחרוזת שאילתה שנשלחת מ-NewPhotoAlbum.aspx. הכנס שם חדש עבור אלבום התמונות ולחץ Insert. תועבר אל ManagePhotoAlbum.aspx שבו תוכל להכניס תמונות חדשות. בינתיים, אתה יכול רק להכניס תיאור של התמונה, תיאור כלי, ו-URL מזוין של התמונה (פשוט הכנס טקסט כלשהו). יותר מאוחר תראה כיצד לשנות זאת ולהניח למשתמש להעלות תמונות ממשיות לאתר. לאחר לחיצת Insert מופיע פריט חדש ברשימה, סמוך לפקדי

Insert. הוסף עוד כמה פריטים ושים לב שפקדי Insert זו לשורה מתחת לאחרים, כפי שמוצג בתרשים 13-16, שמראה דף של Manage Photo Album (ניהול אלבום תמונות).



תרשים 13-16

17. לחץ על הלחצן Delete עבור פריט שתבחר, וראה איך הפריט מוסר מהרשימה באופן אוטומטי.

18. אם אתה צופה כעת באתר בערכת הנושא Monochrome, השתמש ברשימה הנפתחת כדי לעבור ל-DarkGrey. שים לב שהעיצוב מבולגן מעט, מכיוון שאין מספיק מקום להציב שלוש תמונות זו לצד זו. התיקון קל. פתח את הקובץ DarkGrey.css מתיקיית ערכת הנושא DarkGrey והוסף את קוד CSS שלהלן אל סוף הקובץ:

```
.itemContainer
{
    width: 400px;
}
```

הדבר משנה את רוחב רשימת התמונות ל-400 פיקסל, ולכן בערכת נושא זו מוצגות רק שתי עמודות של תמונות.

כיצד זה פועל

התחלת תרגיל זה בצירוף הפקד LinqDataSource אל הפקד ListView. תצורת LinqDataSource נקבעה באופן שיוכל לפעול עם הטבלה Pictures. כפי שראית קודם, כל תמונה מקושרת לאלבום תמונות על ידי PhotoAlbumId שלה. כדי שהדף ManagePhotoAlbum.aspx יציג רק את התמונות ששייכות לאלבום התמונות הנוכחי (מזוהה על ידי מחרוזת השאילתה PhotoAlbumId), הגדרת את WhereParameter.

```

<asp:LinqDataSource ID="LinqDataSource1" runat="server"
    ContextTypeName="PlanetWroxDataContext" EnableDelete="True" EnableInsert="True"
    TableName="Pictures" Where="PhotoAlbumId == @PhotoAlbumId"
    OnInserting="LinqDataSource1_Inserting">
    <WhereParameters>
        <asp:QueryStringParameter DefaultValue="-1" Name="PhotoAlbumId"
            QueryStringField="PhotoAlbumId" Type="Int32" />
    </WhereParameters>
</asp:LinqDataSource>

```

שים לב לשני חלקים חשובים בסימני עריכה אלה. ראשית, האוסף WhereParameters שמכיל QueryStringParameter שצופה לשדה PhotoAlbumId במחרוזת השאילתה. כש-LinqDataSource עומד לקבל את הנתונים שלו, הוא מאחזר את ערך הפרמטר ממחרוזת השאילתה. אם אין ערך, הוא משתמש ב-DefaultValue שערכו '1' בדוגמה שלנו.

החלק החשוב השני הוא המאפיין Where של הפקד LinqDataSource. הוא משתמש בפסקת Where כדי להגביל את כמות הפריטים שמוחזרים מבסיס הנתונים.

```
Where="PhotoAlbumId == @PhotoAlbumId"
```

משפט זה גורם לקבלת כל ה-Pictures מהטבלה Picture שהן בעלות PhotoAlbumId המבוקש. בפעם הראשונה שהדף ייטען לאחר יצירת אלבום תמונות חדש, לא תהיינה תמונות כלל. עם זאת, ברגע שתתחיל להוסיף פריטים תוך שימוש ב-InsertTemplate של הפקד ListView, תראה אותן מופיעות ברשימה.

כדי להציג את התמונות בדף, השתמשת בפקד ListView. כמו פקדים קשורי-נתונים אחרים, גם ListView יכול להציג בדרך מובנית נתונים שחוזרים על עצמם. בדוגמה זו הגדרת את ListView למצב רשימת תבליטים, ולכן הוא מציג את נתוניו כקבוצה של אלמנטים . כדי לציין את המכולה של הפריטים ברשימה, השתמשת ב-LayoutTemplate:

```

<LayoutTemplate>
    <ul class="itemContainer">
        <li ID="itemPlaceholder" runat="server" />
    </ul>
</LayoutTemplat

```

שים לב שה-ID של ה- הזה הוגדר ל-itemPlaceholder. זה אומר לפקד ListView היכן להוסיף את התמונות שמתקבלות. בזמן הביצוע, יוחלף אלמנט זה בפריטים הממשיים מתבניות כמו <ItemTemplate>.

כשהפקד ListView צריך להציג את נתוניו, הוא יוצר פריט שמבוסס על <ItemTemplate> עבור כל פריט נתונים במקור הנתונים. בדוגמה שלנו, כל פריט נתונים הוא אובייקט Picture בעל סוג מובהק שמאפשר גישה למאפיינים כמו ToolTip ו-Description:

```

<ItemTemplate>
    <li>
        ...
        Tooltip:

```



```

<asp:Label ID="TooltipLabel" runat="server" Text='<# Eval("Tooltip") %>' /> />
...
<asp:Button ID="DeleteButton" runat="server" CommandName="Delete" " />
</li>
</ItemTemplate>

```

על ידי קוד זה, כל פריט במקור הנתונים מוצג כסדרה של תוויות שמציגות מאפיינים רלוונטיים של התמונה. לאחר מכן הערך המבוקש מהאובייקט משתמשים ב-`Eval(PropertyName)`. הערך מוצג כטקסט של הפקד `Label`. שים לב שבשלב זה `<ItemTemplate>` מציג רק נתונים אודות התמונה. מאוחר יותר תראה כיצד להעלות ולהציג תמונות ממשיות.

שים לב ל-`CommandName` של הפקד `Button` ב-`ItemTemplate`. הוא מוגדר לביצוע `Delete` וכך לחצן זה הופך לחצן `Delete` אמיתי. כשאתה לוחץ עליו, `ListView` מחליט עבור איזו תמונה לחצת על הלחצן ומורה ל-`LinqDataSource` למחוק אותה מבסיס הנתונים.

קוד `CSS` שהוספת לקובץ `Styles.css` מציג את הפריטים בצורה מאורגנת. כתוצאה מהגדרת התכונה `class` של הפקד `<ul>` ל-`itemContainer`, יוחל קוד `CSS` הבא על רשימה זו:

```

.itemContainer
{
    width: 600px;
    list-style-type: none;
}

```

ההצהרה הראשונה מגדירה את מלוא רוחב הרשימה ל-600 פיקסל, בעוד ההצהרה השנייה מסירה את התבליט מפריטי הרשימה. כל פריט ברשימה מוצג בתוך אלמנט `<li>` שעליו מוחל קוד `CSS` הבא:

```

.itemContainer li
{
    height: 280px;
    width: 200px;
    float: left;
}

```

על כל פריט נכפה רוחב של 200 פיקסל. המאפיין `float` אומר לכל האלמנטים `<li>` לצוף בסמוך זה לזה. בתוך השטח המקורי של 600 פיקסל תוכל להתאים שלושה אלמנטים `<li>` שהרוחב של כל אחד מהם הוא 200 פיקסל. כתוצאה מכך יוצבו האלמנטים הרביעי ואילך בשורה משלהם. זוהי חלופה נהדרת להצגת נתונים על ידי טבלאות `HTML`, שבדרך כלל זקוקות להרבה יותר סימני עריכה כדי להשיג את אותה תוצאה.

לבסוף, על כל תמונה בתוך אלמנט `<li>` נכפים רוחב של 180 פיקסל, שוליים של 10 פיקסל מעל ומתחת לתמונה, 20 פיקסל מימין (כדי ליצור רווח מסוים בין התמונות), וללא רווח משמאל.

```

.itemContainer li img
{
    width: 180px;
    margin: 10px 20px 10px 0;
}

```

בערכת הנושא DarkGrey, הרוחב של itemContainer מאולץ ל-400 פיקסל בלבד. בדרך זו מספיק רוחב ה- <div> בדיוק להצגת שתי תמונות זו לצד זו.

בניגוד לרבים מהפקדים קשורי-הנתונים האחרים, ListView תומך גם בהכנסת נתונים. אתה מגדיר InsertItemTemplate שמכיל פקדים, אחד או יותר, שקשורים למאפיינים באובייקט הבסיסי. לדוגמה, המאפיין Description של התמונה קשור באופן הבא:

```
<InsertItemTemplate>
  <li>
    Description:
    <asp:TextBox ID="DescriptionTextBox" runat="server"
      Text='<%# Bind("Description") %>' /><br />
    ...
  </InsertItemTemplate>
```

במקום **Eval**(ColumnName) קוד זה משתמש ב-**Bind**(ColumnName) כדי להגדיר מנגנון קשירה דו-סיטרי. הדבר מבטיח ש- ASP.NET runtime יוכל להסיק את הקשר בין המאפיין Description של Picture לבין תיבת הטקסט DescriptionTextBox גם לאחר postback. כך, כשאתה מזין כמה פרטים ולוחץ על הלחצן המיוחד Insert (זהו שה-CommandName שלו נקבע ל-Insert) מתרחשת סדרת הצעדים הבאה: נבנה אובייקט Picture חדש, שמאפייניו - כמו Title, Description ו-ToolTip - מתמלאים בערכים מפקדי השרת הקשורים שב-InsertItemTemplate. התמונה נשלחת לפקד LinqDataSource אשר שומר את הפריט בבסיס הנתונים ומרענן את רשימת התמונות שמוצגות בדף.

בתרגיל הבא תלמד כיצד לאפשר למשתמשים לקחת תמונה מהמחשב המקומי ולהעלות אותה לשרת.

נסה זאת התאמה אישית של תבניות הפקד ListView

תבניות ברירת המחדל של הפקד ListView, שיוצרת VWD על פי מידע מ-LinqDataSource, מספקות רק במקרים הפשוטים ביותר. בדרך כלל אתה זקוק לשליטה רבה יותר. לדוגמה, במקום להציג את המאפיין imageUrl הפשוט כטקסט, אתה עשוי לרצות להציג ב-ItemTemplate פקד <asp:Image>. באופן דומה, במקום תיבת טקסט פשוטה אתה עשוי לרצות להציג ב-InsertItemTemplate פקד להעלאת קובץ. בתרגיל הבא תלמד כיצד לשנות את התבניות התקניות באופן שיאפשר לך לכלול בדף את שתי התכונות. בנוסף, תראה כיצד לטפל באירוע Inserting של הפקד LinqDataSource כדי לשמור את הקובץ שהועלה בדיסק, ולעדכן את בסיס הנתונים ב-URL של התמונה.

כדי שדוגמה זו תפעל, החשבון שבו משתמש שרת האינטרנט (אם אתה משתמש בשרת המובנה שמגיע עם VWD, זהו החשבון שבו אתה משתמש כדי להתחבר למחשב שלך) צריך להיות בעל הרשאות קריאה וכתובה לתיקייה GigPics שתיצור במהלך התרגיל. כדי לקבוע את תצורת ההרשאות פנה לסעיף "האבטחה ב-IIS" בפרק 18.

1. בשורש האתר צור תיקייה חדשה GigPics. תיקייה זו מכילה את התמונות שצילמו משתמשים בהופעות חיות שצפו בהן.

2. פתח את הדף ManagePhotoAlbum.aspx ב- Markup View. אתר בו את האלמנט <ItemTemplate>. הסר ממנו את ה-Label שמציג את imageUrl והחליף אותו בפקד imageUrl שהמאפיין שלו מוגדר ל- imageUrl של אובייקט התמונה.

```
<asp:Image ID="ImageUrl" runat="server" ImageUrl='<%# Eval("ImageUrl") %>' />
```

הסר את הטקסט imageUrl שמופיע ממש מעל לתמונה.

3. כדי לאפשר למשתמשים להעלות תמונות, עליך להחליף את ה-TextBox של המאפיין imageUrl בפקד FileUpload. עליך להסיר את הטקסט imageUrl פעם נוספת. בצע זאת ב-InsertItemTemplate:

```
<asp:TextBox ID="TooltipTextBox" runat="server" Text='<%# Bind("Tooltip") %>' /><br />
<asp:FileUpload ID="FileUpload1" runat="server" /><br />
<asp:Button ID="InsertButton" runat="server" CommandName="Insert" Text="Insert" />
```

שים לב שכאן אין צורך לקשור את המאפיין לפקד. מכיוון שהתמונה שמועלית לאינטרנט זקוקה לטיפול מיוחד, עליך לכתוב מעט קוד בקוד ברקע של הדף, במקום לסמוך על היכולות המובנות של קישור נתונים.

4. ב-InsertItemTemplate הגדר את המאפיין CausesValidation של הלחצן Cancel למצב False:

```
<asp:Button ID="CancelButton" runat="server" CommandName="Cancel" Text="Clear"
CausesValidation="false" />
```

5. ב-ItemTemplate הגדר גם את המאפיין CausesValidation של הלחצן Delete למצב False.

6. עבור לקוד ברקע של הדף והרחב את מטפל האירוע Inserting עם הקוד הבא, ששומר את הקובץ בדיסק ומעדכן את המאפיין imageUrl של המופע Picture במיקום החדש שלו.

VB.NET

```
myPicture.PhotoAlbumId = Convert.ToInt32(Request.QueryString.Get("PhotoAlbumId"))
```

```
Dim FileUpload1 As FileUpload = _
    CType(ListView1.InsertItem.FindControl("FileUpload1"), FileUpload)
Dim virtualFolder As String = "~/GigPics/"
Dim physicalFolder As String = Server.MapPath(virtualFolder)
Dim fileName As String = Guid.NewGuid().ToString()
Dim extension As String = System.IO.Path.GetExtension(FileUpload1.FileName)

FileUpload1.SaveAs(System.IO.Path.Combine(physicalFolder, fileName + extension))
myPicture.ImageUrl = virtualFolder + fileName + extension
```

C#

```
myPicture.PhotoAlbumId = Convert.ToInt32(Request.QueryString.Get("PhotoAlbumId"));
```

```
FileUpload FileUpload1 =
(FileUpload)ListView1.InsertItem.FindControl("FileUpload1");
string virtualFolder = "~/GigPics/";
string physicalFolder = Server.MapPath(virtualFolder);
string fileName = Guid.NewGuid().ToString();
string extension = System.IO.Path.GetExtension(FileUpload1.FileName);

FileUpload1.SaveAs(System.IO.Path.Combine(physicalFolder, fileName + extension));
myPicture.ImageUrl = virtualFolder + fileName + extension;
```

7. חזור ל- Markup View והוסף שלושה פקדים לבדיקת תוקף אל InsertItemTemplate: שני פקדים RequiredFieldValidator שמחברים לתיבות הטקסט של Description ושל Tooltip, ופקד CustomValidator שזה ErrorMessage שלו מוגדרת ל- Please select a valid .jpg file. לבסוף, הגדר את המאפיין TextMode של תיבת הטקסט של Description ל-MultiLine.

אתה אמור לסיים עם הקוד הבא:

```
<asp:TextBox ID="DescriptionTextBox" runat="server" TextMode="MultiLine"
Text='<## Bind("Description") %>' /><br />
<asp:RequiredFieldValidator ID="RequiredFieldValidator1"
ControlToValidate="DescriptionTextBox" runat="server"
ErrorMessage="Please enter a description."></asp:RequiredFieldValidator>
Tooltip:
<asp:TextBox ID="TooltipTextBox" runat="server" Text='<## Bind("Tooltip") %>' /><br />
<asp:RequiredFieldValidator ID="RequiredFieldValidator2"
ControlToValidate="ToolTipTextBox" runat="server"
ErrorMessage="Please enter a tooltip.">
</asp:RequiredFieldValidator>
<asp:FileUpload ID="FileUpload1" runat="server" /><br />
<asp:CustomValidator ID="CustomValidator1" runat="server"
ErrorMessage="Please select a valid .jpg file"></asp:CustomValidator>
<asp:Button ID="InsertButton" runat="server" CommandName="Insert" Text="Insert" />
```

8. עבור ל- Design View ובחר בפקד ListView. כדי לקבוע מטפל אירוע עבור האירוע ItemInserting שלו, לחץ לחיצה כפולה על האירוע בקטגוריה Event של טבלת המאפיינים. השלם את מטפל האירוע עם הקוד הבא:

VB.NET

```
Protected Sub ListView1_ItemInserting(ByVal sender As Object, _
ByVal e As System.Web.UI.WebControls.ListViewInsertEventArgs) _
Handles ListView1.ItemInserting
Dim FileUpload1 As FileUpload = _
CType(ListView1.InsertItem.FindControl("FileUpload1"), FileUpload)
If Not FileUpload1.HasFile OrElse _
Not FileUpload1.FileName.ToLower().EndsWith(".jpg") Then
Dim CustomValidator1 As CustomValidator = _
CType(ListView1.InsertItem.FindControl("CustomValidator1"), CustomValidator)
```

```

CustomValidator1.IsValid = False
e.Cancel = True
End If

```

End Sub

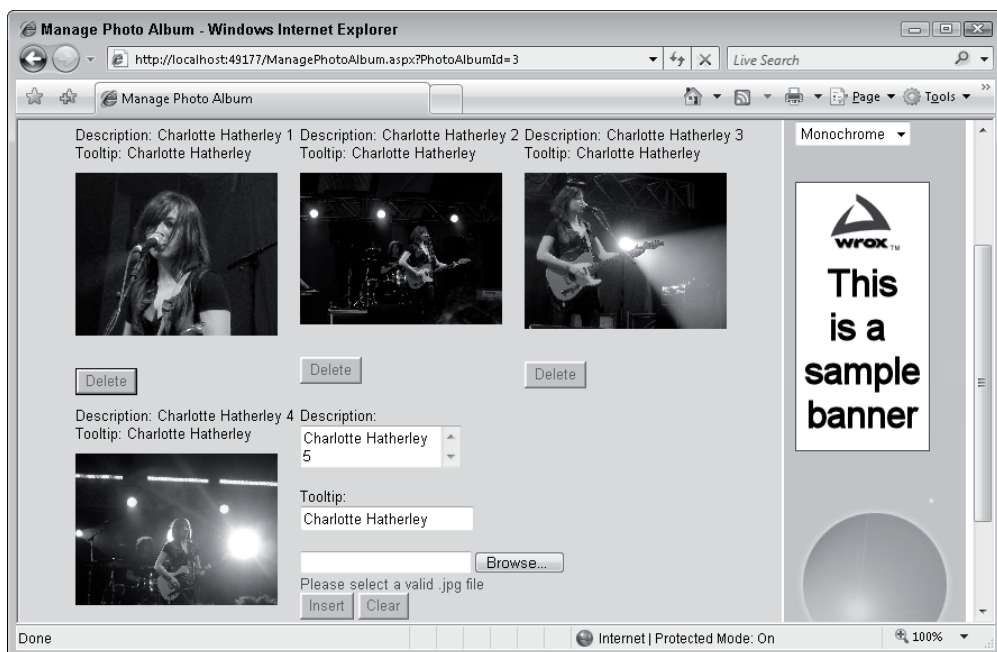
C#

```

protected void ListView1_ItemInserting(object sender, ListViewInsertEventArgs e)
{
    FileUpload FileUpload1 = (FileUpload)ListView1.InsertItem.FindControl("FileUpload1");
    if (!FileUpload1.HasFile || !FileUpload1.FileName.ToLower().EndsWith(".jpg"))
    {
        CustomValidator CustomValidator1 =
            (CustomValidator)ListView1.InsertItem.FindControl("CustomValidator1");
        CustomValidator1.IsValid = false;
        e.Cancel = true;
    }
}

```

9. שמור את כל השינויים ופתח את `NewPhotoAlbum.aspx` על ידי הרפרדפן. הוץ שם חדש עבור אלבום התמונות ולחץ על הקישור `Insert`. הכנס כמה תמונות על ידי הצעדים הבאים: הוסף תיאור ותיאור כלי, בחר תמונה `jpg` מהדיסק הקשית, לחץ על הלחצן `Insert`. חזור על צעדים אלה עבור תמונה נוספת, אבל השאר את תיבת הדר-שיח `File browse` ריקה. כשתלחץ `Insert`, תקבל הודעת שגיאה שמציינת שניסית להעלות קובץ `JPG` שאינו תקף, כפי שמוצג בתרשים 13-17.



תרשים 13-17

10. לחץ על הלחצן Browse של הפקד File upload. דפדף עד שתמצא קובץ JPG תקף, ואז לחץ על Insert פעם נוספת. הקובץ יועלה עכשיו בהצלחה.

כיצד זה פועל

התהליך הממשי של הכנסת Picture לבסיס הנתונים לא עבר שינויים רבים. הפקד ListView עדיין אוסף את כל הנתונים הרלוונטיים מהדף ושולח אותם לפקד LinqDataSource, שמכניס את הפריט לטבלה Picture בבסיס הנתונים. השוני הוא בדרך שבה הגדרת את התבניות והדרך שבה טיפול באירועים של הפקדים LinqDataSource ו-ListView. תחילה הבט בתבניות. בתבנית ItemTemplate הוספת <asp:Image> כדי להחליף את תווית הטקסט הפשוט. כתוצאה מכך התמונה הממשית מוצגת במקום ה-URL שלה, כפי שרואים בתרשים 13-17.

כדי לאפשר למשתמש להעלות את התמונות לדף, החלפת ב-InsertItemTemplate את הפקד TextBox בפקד FileUpload. בנוסף, הוספת כמה פקדי תוקף כדי לכפות על המשתמש למלא את השדות הדרושים. ברגע שלחצת על הלחצן Insert, הדף מוחזר אל השרת (posts back) והפקד ListView מעורר את האירוע ItemInserting שלו. אירוע זה הוא מקום טוב לביצוע בדיקת תוקף מותאמת אישית. אחד מהארגומנטים שמטפל אירוע זה מקבל (הארגומנט e) הוא מסוג ListViewInsertEventArgs, כלומר מחלקה שמספקת מידע תלוי-תוכן אודות פעולת Insert של הפקד ListView. כשתגלה שגיאה תוכל להגדיר כ-False את המאפיין Cancel של הארגומנט 'e' שמועבר למטפל האירוע. הדבר אומר לפקד ListView שברצונך לבטל את פעולת Insert. מטפל אירוע זה כולל הוספת קוד ש"מוצא" את פקד ההעלאה בתבנית InsertItem. מכיוון שקיימת אפשרות שיהיו פקדים רבים בעלי שם זהה (לדוגמה, פקד FileUpload ב-InsertItemTemplate וב-EditItemTemplate), לא תוכל לגשת ישירות לפקד FileUpload. במקום זאת תצטרך להשתמש ב-FindControl באובייקט InsertItem כדי לחפש את הפקד.

VB.NET

```
Dim FileUpload1 As FileUpload = _  
    CType(ListView1.InsertItem.FindControl("FileUpload1"), FileUpload)
```

C#

```
FileUpload FileUpload1 = (FileUpload)ListView1.InsertItem.FindControl("FileUpload1");
```

כשיש לך הפניה לפקד FileUpload, באפשרותך לבדוק את המאפיין HasFile שלו כדי לברר אם הועלה קובץ. בנוסף, כדי לברר אם הועלה קובץ מסוג jpg, תוכל לבדוק זאת כך: FileUpload1.FileName.ToLower().EndsWith(".jpg") כדי לוודא שבדיקה זו מתבצעת רק כאשר המשתמש העלה קובץ, משתמש הקוד ב-OrElse (ב-VB) כדי לגרום קצר בלוגיקה של משפט If, כפי שמוסבר בפרק 5.

כשהמשתמש מעלה קובץ שאינו תקף, מתבצע הקוד בבלוק If. הוא משתמש שוב ב-FindControl כדי למצוא את הפקד CustomValidator, ואז מגדיר את המאפיין IsValid שלו ל-False. הדבר אומר

לפקד להציג את המאפיין ErrorMessage שלו כשהדף מרוגדר. לבסוף, כדי לעצור את ListView מלהמשיך את ההכנסה, עליך להגדיר את המאפיין Cancel של הארגומנט e ל-True.

VB.NET

```
e.Cancel = True
```

C#

```
e.Cancel = true;
```

כשהמשתמש מעלה לאתר קובץ jpg תקף, הפקד ListView ממשיך עם פעולת ההכנסה באמצעות הפקד LinqDataSource. כשהאחרון עומד לשלוח את הנתונים לבסיס הנתונים, הוא מעורר את האירוע Inserting שלו, ונותן לך הזדמנות אחרונה להתחבר לתהליך ולהתבונן בנתונים. בתוך מטפל אירוע זה השתמשת בקוד דומה כדי למצוא הפניה לפקד FileUpload בתוך התבנית InsertItem. אז השתמשת בקוד הבא כדי לקבוע את התיקייה הפיזית והווירטואלית עבור הקובץ, וקבעת את השם ואת הסיומת:

VB.NET

```
Dim virtualFolder As String = "~/GigPics/"
Dim physicalFolder As String = Server.MapPath(virtualFolder)
Dim fileName As String = Guid.NewGuid().ToString()
Dim extension As String = System.IO.Path.GetExtension(FileUpload1.FileName)
```

C#

```
string virtualFolder = "~/GigPics/";
string physicalFolder = Server.MapPath(virtualFolder);
string fileName = Guid.NewGuid().ToString();
string extension = System.IO.Path.GetExtension(FileUpload1.FileName);
```

המשתנה virtualFolder מכיל את המיקום הווירטואלי של התיקייה שבה מאוחסנות התמונות שהועלו לאתר. אז אתה מוצא את התיקייה הפיזית על ידי שימוש ב-Server.MapPath. בהנחה שהפרויקט נמצא במיקום ברירת המחיר של C:\BegASPNET\Site - המשתנה physicalFolder יכיל כעת את C:\BegASPNET\Site\GigPics.

בצעד הבא נוצר שם קובץ חדש אקראי על ידי השיטה Guid.NewGuid(). בפעולה זו מוקצה למשתנה fileName שם ייחודי, כמו למשל f6d8ed05-2dbe-4aed-868a-de045f9462e3. לבסוף, הסיומת של הקובץ מאוחזרת על ידי השיטה הסטטית GetExtension של המחלקה Path ממרחב השמות System.IO.

בשלב זה נמצא בידך כל המידע הדרוש לאחסון הקובץ בדיסק, ולעדכון בסיס הנתונים. האחסון בדיסק מתבצע בקלות על ידי השיטה SaveAs של הפקד FileUpload.

VB.NET

```
FileUpload1.SaveAs(System.IO.Path.Combine(physicalFolder, fileName + extension))
```

C#

```
FileUpload1.SaveAs(System.IO.Path.Combine(physicalFolder, fileName + extension));
```

הקוד לוקח את התיקייה הפיזית, את שם הקובץ ואת הסיומת, ומעביר אותם לשיטה SaveAs, אשר שומרת את הקובץ במיקום הנדרש.

לבסוף, המופע Picture מעודכן עם imageUrl חדש:

VB.NET

```
myPicture.ImageUrl = virtualFolder + fileName + extension
```

C#

```
myPicture.ImageUrl = virtualFolder + fileName + extension;
```

שם דומה ל- ~GigPics/f6d8ed05-2dbe-4aed-868a-de045f9462e3.jpg יוקצה עבור המאפיין imageUrl, שכעת זהו המיקום הווירטואלי של התמונה שהועלתה. פקד ListView מתעדכן מייד לאחר הכנסת התמונה החדשה, ומציג כעת את התמונה החדשה תוך שימוש בפקד Image שה- imageUrl שלו מוגדר לתמונה שהועלתה זה עתה.

אתה יכול לתאר לעצמך שהעלאת מספר גדול של תמונות לאלבום תמונות יחיד תקשה על ניהול הדף. הדבר נכון במיוחד לגבי המשתמשים שמתחברים לאתר ברשת איטית. במצב זה, במקום להציג את כל התמונות שבאלבום התמונות בדף יחיד, עדיף לפצל את האלבום לדפים רבים ולאפשר למשתמש לעבור מדף לדף. בסעיף הבא, שדן בפקד DataPager, תלמד כיצד לעשות זאת.

הפקד DataPager

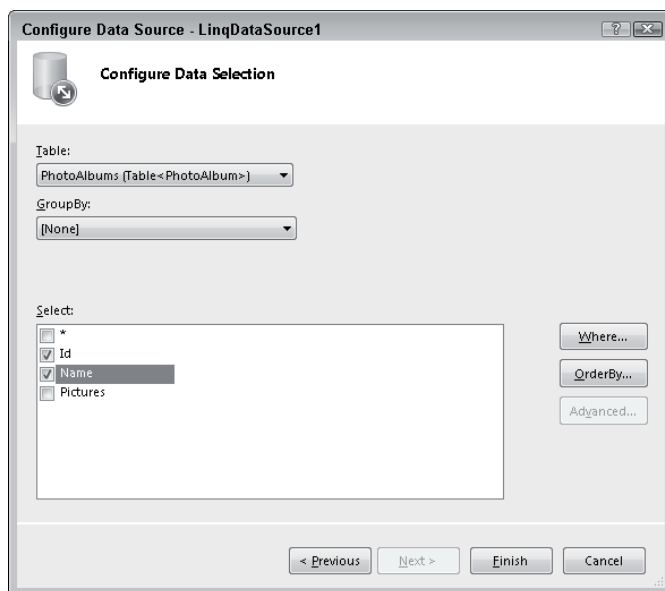
הפקד DataPager חדש ב- ASP.NET 3.5. בגרסאות קודמות של ASP.NET התאפשר דפדוף רק על ידי פונקציונליות שנבנתה ישירות בכמה פקדים, כמו GridView ו-DetailsView, או על ידי כתיבת קוד באופן ידני.

DataPager שונה בכך, שהוא פקד נפרד וניתן להשתמש בו כדי להרחיב פקד אחר, פקד קשור-נתונים. כיום ניתן להשתמש ב-DataPager רק כדי לספק יכולות דפדוף לפקד ListView, אבל הצפי הוא שפקדי צד שלישי, או פקדים עתידיים של ASP.NET יורחבו אף הם.

יש שתי דרכים לחבר את DataPager לפקד ListView. באפשרותך להגדיר אותו בתוך <LayoutTemplate> של הפקד ListView, או להגדירו לגמרי מחוץ לו. במקרה הראשון DataPager יודע לאיזה פקד עליו לספק יכולות דפדוף. במקרה השני עליך להגדיר את המאפיין PagedControlID של DataPager ל-ID של פקד ListView תקף. בסעיף הבא תלמד כיצד להגדיר את תצורת הפקד DataPager, וכיצד להשתמש בו יחד עם ListView. היכולת להגדיר את DataPager מחוץ לפקד ListView הינה שימושית כשצריך להציב אותו במקום שונה בדף, למשל בשטח ה-Footer (החלק התחתון בדף) או ה-SideBar (השטח בשולי הדף).

בתרגיל זה תיצור את דף ממשק המשתמש של Gig Pics. המשתמשים באתר יוכלו לבחור מרשימה נפתחת באחד מאלבומי התמונות הזמינים, ואז לצפות בכל התמונות הזמינות ברשימה ברת-דפדוף שנוצרת על ידי הפקדים ListView ו-DatPager. תרשים 13-21 מציג את התוצאה הסופית של תרגיל זה.

1. בשורש האתר, צור תיקייה חדשה PhotoAlbums. בתוך תיקייה זו צור טופס אינטרנט חדש Default.aspx. ודא שהטופס מבוסס על תבנית הדף המותאמת אישית שלך. הגדר את ה-Title של הדף ל- All Photo Albums. אם אתה משתמש ב-VB.NET, עבור לקוד ברקע ושנה את שם המחלקה ל-Default_ כדי למנוע התנגשות עם המילה השמורה Default. ב- Markup View, ערוך את אותם השינויים בהנחיה Page @ של הדף החדש.
2. עבור ל- Design View ושחרר פקד DropDownList על הדף. בלוח Smart Tasks של הפקד אֶפְשֶׁר AutoPostBack. חבר את הפקד לפקד LinqDataSource חדש על ידי לחיצה על Choose Data Source ובחירת New data source מהרשימה הנפתחת הראשונה. לחץ על הסמל LINQ, לחץ OK, וודא שמהרשימה הנפתחת בחרת PlanetWroxDataContext. לחץ Next. בתיבת הדו-שיח Configure Data Selection בחר PhotoAlbums מהרשימה הנפתחת Table, ובחר בשדות Id ו-Name, כפי שמוצג בתרשים 13-18.



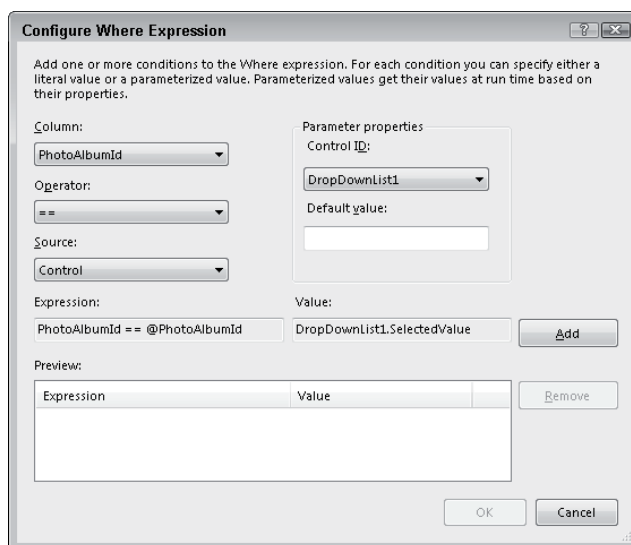
תרשים 13-18

3. לחץ Finish כדי לסגור את Data Source Wizard. בתיבת הדו-שיח Choose Data Source, בחר מהרשימה הנפתחת Data Field to Display את Name, וברשימה הנפתחת Data Field for the Value, השאר את הבחירה של Id, כפי שמתואר בתרשים 13-19.
4. לחץ OK כדי לסגור את האשף Data Source Configuration.



תרשים 13-19

5. הוסף פקד `ListView` חדש מתחת ל-`DropDownList`. חבר אותו אל `LinqDataSource` חדש על ידי בחירת `<New data source>` מהרשימה הנפתחת שכלוח `Smart Tasks` שלו. לחץ על הסמל `LINQ`, לחץ `OK`. ודא שבחרת `PlanetWroxDataContext` ולחץ `Next`. מהרשימה הנפתחת `Table` שבתיבת הדו-שיח שנפתחה, בחר `Pictures (Table<Picture>)`, ואז לחץ על הלחצן `Where`. כעת מוצגת תיבת דו-שיח. כדי להגדיר מסנן שיגביל את התמונות הנבחרות לאלו ששייכות לאלבום התמונות שנבחר ב- `DropDownList`, מלא את שדות תיבת הדו-שיח כפי שמוצג בתרשים 13-20.



תרשים 13-20

6. לחץ Add כדי ליצור את הפרמטר, ואז לחץ OK ו-Finish כדי להכניס את הפקד ל-LinqDataSource לדרך.

7. בלוח Smart Tasks של ListView, לחץ Configure ListView. מוצגת תיבת דו-שיח. מהרשימה Select a Layout שבתיבה, בחר Bulleted List, ובאזור Options בחר Enable Paging. הרשימה הנפתחת שמתחתיה אמורה להכיל את ברירת המחדל Next/Previous Pager שמתאימה לנו בתרגיל זה.

8. לחץ OK כדי ש-VWD תיצור עבורך כמה תבניות.

9. עבור ל-Markup View והסר את התבניות הבאות ואת הקוד שהן מכילות:

- ⊙ <AlternatingItemTemplate>
- ⊙ <InsertItemTemplate>
- ⊙ <SelectedItemTemplate>
- ⊙ <EditItemTemplate>
- ⊙ <ItemSeparatorTemplate>

10. הסר את המאפיינים ID, runat ו-style מה- שב-LayoutTemplate, ואז הוסף מאפיין class והגדר אותו ל-itemContainer. אתר את DataPager בתוך LayoutTemplate והוסף מאפיין PageSize שמוגדר לערך 3:

<LayoutTemplate>

```
<ul class="itemContainer">
  <li id="itemPlaceholder" runat="server" />
</ul>
<div style="">
  <asp:DataPager ID="DataPager1" runat="server" PageSize="3">
    <Fields>
      <asp:NextPreviousPagerField ButtonType="Button"
        ShowFirstPageButton="True"
        ShowLastPageButton="True" />
    </Fields>
  </asp:DataPager>
</div>
```

</LayoutTemplate>

11. שנה את הקוד ב-ItemTemplate כדי שיהיה דומה לזה:

```
<ItemTemplate>
<li>
  <asp:Image ID="Image1" runat="server" ImageUrl='<%= Eval("ImageUrl") %>'
    Tooltip='<%= Eval("ToolTip") %>' />
  <asp:Label ID="DescriptionLabel" runat="server" Text='<%= Eval("Description") %>' />
</li>
</ItemTemplate>
```

הקוד יוצר פקד Image שהמאפיינים imageUrl ו-Tooltip שלו קשורים למאפיינים המקבילים של האובייקט Picture שאליו אתה קושר אותו. Tooltip (תיאור כללי) מופיע כשאתה מרחף עם העכבר על פני התמונה בדפדפן. מתחת לתמונה פקד Label מציג את התיאור (Description) של התמונה.

12. כדי למנוע הבהוב של הדרך בזמן דפדוף ברשימת התמונות, או בזמן בחירת אלבום תמונות חדש מהרשימה, עטוף את כל התוכן שבבלוק התוכן cpMainContent ב-UpdatePanel של Ajax עם אלמנט ContentTemplate.

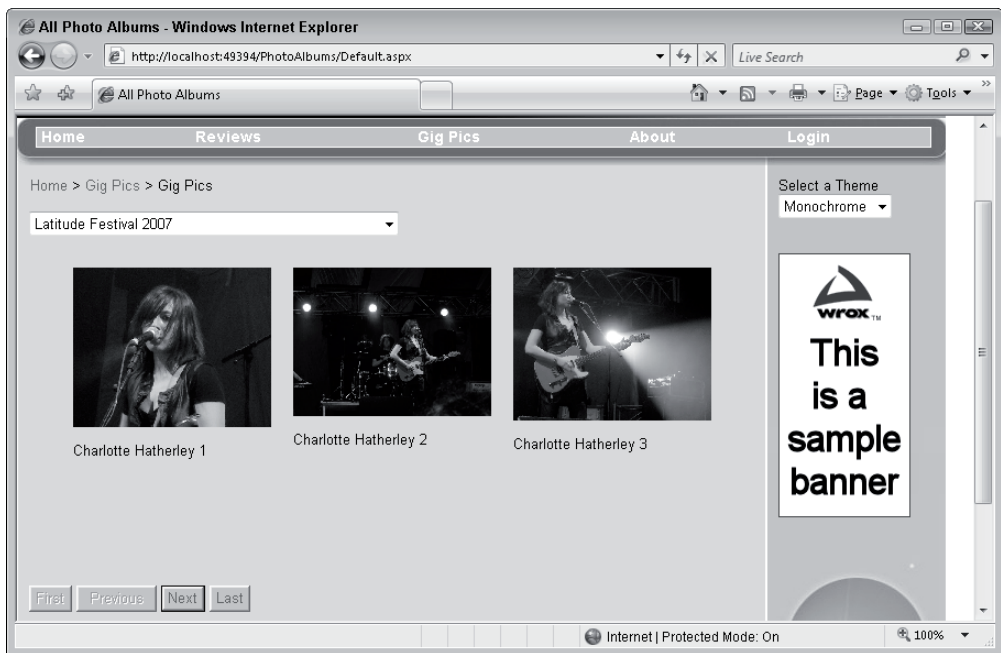
```
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" runat="Server">
  <asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
      <asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"
        DataSourceID="LinqDataSource1"
        DataTextField="Name" DataValueField="Id">
        ...
      </ContentTemplate>
    </asp:UpdatePanel>
  </asp:Content>
```

13. פתח את הקובץ Web.sitemap של האתר. בין הקטעים Review ו-About הוסף תפריט ראשי ושני תת-תפריטים עבור הקטע PhotoAlbum:

```
</siteMapNode>
<siteMapNode url="~/PhotoAlbums/" title="Gig Pics" description="All Gig Pics">
  <siteMapNode url="~/PhotoAlbums/Default.aspx" title="Gig Pics"
    description="All Gig Pics" />
  <siteMapNode url="~/NewPhotoAlbum.aspx" title="New Album"
    description="Create a new Photo Album with Gig Pics" />
</siteMapNode>
<siteMapNode url="~/About/Default.aspx" title="About" description="About this site">
```

14. שמור את כל השינויים ובקש בדפדפן את הדרך Default.aspx מהתיקייה PhotoAlbums. בחר באלבום תמונות מהרשימה הנפתחת, והדרך ייטען מחדש כשהוא מציג את התמונות הרלוונטיות מאלבום התמונות. אם אין לך תמונות כלל באלבום התמונות, או שאין מספיק תמונות כדי למלא את הדרך כולו, בחר New Album מתפריט Gig Pics, צור אלבום תמונות חדש, והוסף לו לפחות ארבע תמונות. עכשיו, לחץ על פריט בתפריט Gig Pics ומהרשימה הנפתחת בחר באלבום התמונות החדש. שים לב שכעת מופיע ממשק משתמש לדפדוף, שמאפשר לך לנוע קדימה ואחורה ברשימת התמונות של אלבום התמונות, באמצעות הלחצנים First, Previous, Next ו-Last שנראים בתחתית המסך שבתרשים 13-21.

שים לב שהודות לתוספת ה-Ajax, פעולת הדפדוף שנעשית כעת חופשייה לחלוטין מהבהוב.



תרשים 13-21

כיצד זה פועל

רוב החומר שהוצג לפניך בתרגיל זה אמור להיות מוכר לך. חיברת את DropDownList ל-LinqDataSource בדרך דומה לזו שבה יצרת את הרשימה הנפתחת Genres בפרק הקודם. הפקד ListView והפקד LinqDataSource הקשור אליו, גם הם דומים לאלה שכבר ראינו מוקדם יותר בפרק זה. עם זאת, במקום הפרמטר Where שמקבל את ערכו ממחרוזת השאילתה, הקוד משתמש כעת בפרמטר Where המקבל את ערכו מהערך הנבחר מפקד ה-DropDownList:

```
<WhereParameters>
  <asp:ControlParameter ControlID="DropDownList1" Name="PhotoAlbumId"
    PropertyName="SelectedValue" Type="Int32" />
</WhereParameters>
```

כשהפקד LinqDataSource עומד לקבל את הנתונים מבסיס הנתונים, הוא מסתכל במאפיין SelectedValue שברשימה הנפתחת, ומאחזר רק את התמונות שתואמות למאפיין PhotoAlbumId הנדרש.

ההבדל הגדול ביותר מהדוגמאות הקודמות הוא בהוספת הפקד DataPager. כפי שהודגם בתרגיל זה, הדפדוף נעשה באופן אוטומטי. כל שעליך לעשות הוא להטמיע פקד DataPager במקום כלשהו ב-LayoutTemplate של ListView, וכל השאר יטופל באופן אוטומטי. אם תציב את DataPager מחוץ ל-ListView, אל תשכח לקשור אותו ל-ListView על ידי הגדרת המאפיין PagedControlID עם ערך שהוא המזהה של ה-ListView. אם אתה מעדיף קישורים או תמונות

על פני לחצנים, תוכל להגדיר את המאפיין ButtonType של האלמנט NextPreviousPagerField ל-Link או Image, בהתאמה.

אם אתה מעדיף מונה מספרי, החלף את האלמנט NextPreviousPagerField ב-NumericPagerField:

```
<asp:NumericPagerField NextPageText="..." PreviousPageText="..." />
```

כמה הערות לגבי ביצועים

לתרגיל זה יש שתי בעיות ביצועים ידועות, שעליך להיות מודע להן. ראשית, למרות ש-ItemTemplate של ListView קובע את גודל התמונות בדפדפן ל-180 פיקסל, על ידי הגדרת רוחבן בקוד CSS, התמונה הממשית נשארת ללא שינוי. פירוש הדבר שכאשר אתה מעלה לאתר תמונה גדולה, כל התמונה נטענת בשלמותה אך מוצגת כתמונת דוגמה קטנה. במקום זאת היה עדיף ליצור תמונת דוגמה אמיתית בשרת ולשלוח רק אותה לדפדפן, ובכך לחסוך בנפח התעבורה ברשת. ניתן למצוא דוגמאות ומאמרים המסבירים כיצד לבצע משימה זו של שינוי גודל התמונות ברשת.

גם לדרך שבה הדפדפן מבצע את הדפדוף עלולות להיות השלכות על הביצועים. כשמשתמשים בפקד DataPager, דפדוף הנתונים נעשה בדף ASPX. פירוש הדבר שכל הנתונים מאוחזרים מבסיס הנתונים ונשלחים לפקד LinqDataSource. הפקד DataPager הוא זה שבחר מהנתונים את הרשומות שמתאימות להצגה בדף הנוכחי. הדבר פועל היטב עבור קבוצות תוצאה שאינן גדולות מכמה מאות רשומות. אולם כאשר מספר הפריטים באלבום תמונות, או באוסף אחר, עולה על מספר זה, אתה עלול לגלות האטה בביצועי הדף. במקרה כזה ייתכן שתרצה לשקול דפדוף ברמת בסיס הנתונים. כזכור מתחילת הפרק, שימוש ב-LINQ עם השיטות Skip() ו-Take() יעזרו לך לבצע משימה זו.

עצות מעשיות לעבודה ב-LINQ

להלן כמה עצות מעשיות לעבודה ב-LINQ.

- ⊙ בפרק זה למדת כיצד ליצור סוגים אנונימיים לעיצוב הנתונים שברצונך להחזיר מהשאליות. המהדר ו-IntelliSense הם כלים הכרחיים בקביעת הנתונים שביכולתך להחזיר והמאפיינים שעומדים לרשותך. הקדש זמן מה ל"משחק" בסוגים האנונימיים, ולהתבוננות באפשרויות השונות שמציגות רשימות IntelliSense.
- ⊙ בדיוק כמו בשיטות גישה אחרות לנתונים, וכמו הפקד SqlDataReader שאותו פגשת בפרק הקודם, נסה לסנן את הנתונים ככל שתוכל. אם, למשל, ידוע לך שאתה צריך רק את הסקירות שנוגעות לסגנון ג'אז, הייה מפורש ושלב בקוד פסקת Where שמגבילה את רשימת הסקירות ברמת בסיס הנתונים. הדבר מאיץ את השאליות ואת אחזור הנתונים, ומשפר את הביצועים הכוללים של היישום.

⊙ השתמש בסוגים אנונימיים כדי להקטין את צריכת הזיכרון של שאילתות LINQ. במקום לאחזר את האובייקט Review בשלמותו, השתמש במילת המפתח New כדי ליצור סוג אנונימי תוך כדי ביצוע. מכיוון שאובייקט חדש זה מכיל רק את המאפיינים שאתה באמת צריך, נחסכת התקורה של הבאת האובייקט כולו.

סיכום

LINQ היא טכנולוגיה חדשה הכלולה ב- ASP.NET 3.5. טכנולוגיית LINQ עתידה לתפוס מקום חשוב בתרחישים רבים של גישה לנתונים, כולל גישה לבסיסי נתונים מיישומי אינטרנט בסביבת ASP.NET תוך שימוש ב- LINQ to SQL.

מכיוון ש-LINQ עומדת למלא תפקיד כה חשוב, היא שולבה במקומות רבים ושונים ב-.NET. היא גם זמינה עבור אובייקטים ומאפשרת לתשאל אוספים בזיכרון. בנוסף, LINQ זמינה עבור XML, Entities ו-DataSets, וכל סוג של LINQ מספק גישה למחסן נתונים שונה, אבל לכולם יש אותה שפת תשאל/שאילתות אחידה. סוג נוסף של LINQ הוא LINQ to SQL שמספק גישה לבסיס הנתונים שרת SQL.

כדי לכתוב שאילתות LINQ, עליך להשתמש באופרטורים של שאילתה (שדומים לאלה של SQL) וכוללים מילות מפתח כמו Select, From ו-Where ואופרטורים של צבירה כמו Sum ו-Count. כדי לקבל נתונים משאילתת LINQ תוכל להשתמש בהסקת הסוג כדי להניח למהדר לקבוע את סוג הנתונים שמתקבל בתוצאה. הדבר גם מאפשר ליצור סוגים אנונימיים, שבהם הסוג הממשי המוחזר על ידי השאילתה הוא חסר שם, אבל הממשק הכללי שלו מוסק על ידי המהדר, שנותן לך את הכלי IntelliSense בזמן הידור, וגישה מלאה לאובייקט בזמן ביצוע.

כדי להפעיל את LINQ to SQL ביישומי אינטרנט בסביבת ASP.NET, עומדות בפניך כמה אפשרויות. האחת, שאפשר לכתוב שאילתות בקוד ברקע של דף, ואז לקשור את התוצאות לפקד קשור-נתונים (data-bound) תוך שימוש במאפיין DataSource ובשיטה DataBind של הפקד. למרות שגישה זו נותנת גמישות רבה במונחי השאילתות שביכולתך לבצע, משמעותה גם שתצטרך לכתוב קוד רב באופן ידני.

אפשרות שנייה להפעיל את LINQ בדפי ASP.NET היא להשתמש בפקד LinqDataSource, הגשר בין הפקדים קשורי-הנתונים לבין מודל LINQ. אפשרות זו מקטינה את כמות הקוד שעליך לכתוב. כשהפקד LinqDataSource משולב עם הפקדים החדשים ListView ו-DataPager, הוא מאפשר ליצור דפי CRUD בעלי פונקציונליות מלאה (כלומר, דפים שמאפשרים ליצור, לקרוא, לעדכן ולמחוק נתונים בבסיס הנתונים), עם קוד מועט ככל האפשר.

הדפים מונעי-בסיס-הנתונים (database-driven) שראית עד כה נראו דלים למדי. לא החלת עליהם כל סגנון ולא סיפקת עיצוב מותנה, היכן שהנתונים מוצגים בצורה שונה בהתבסס על הערכים הפנימיים שלהם. הדבר ניתן להשגה באמצעות סגנונות שליטה, והאירועים הרבים של הפקדים קשורי-הנתונים ופקדי מקור הנתונים. בפרק 14 תלמד כיצד להשתמש בסגנונות ובאירועים אלה.

1. הבה נניח שבתיקייה Reviews נמצא דף בשם MostRecent.aspx. דף זה מציג את 10 הסקירות האחרונות שהתווספו לבסיס הנתונים. איך תיראה שאילת LINQ שלך אם ברצונך להציג רק את המאפיין Title של סקירה ואת שם הסגנון שהסקירה שייכת או? עליך להשתמש בשיטה Take כדי להגביל את קבוצת התוצאה ל-10. אם יהיו לך קשיים בכתיבת הקוד להשגת הסקירות האחרונות, התבונן בסעיף "First, Last, FirstOrDefault ו-LastOrDefault" (שבפרק זה) המסביר כיצד לקבל את הסקירה האחרונה בבסיס הנתונים.
2. מה התועלת העיקרית בפקד ListView החדש לעומת פקדי הנתונים האחרים כמו GridView ו-?Repeater?
3. הדף Default.aspx מהתיקייה PhotoAlbums מציג כיום רק דוגמיות של תמונות. מה עליך לעשות כדי להציג תמונה בגודל מלא בדף משלה, באמצעות שאילתת LINQ?

פרק 14

הצגת נתונים - נושאים מתקדמים

בפרקים הקודמים הצגנו רעיונות חדשים רבים. פרק 11 דן בבסיסי נתונים בכלל ובכלי SQL Server 2005 Express Edition בפרט. הפרק דן גם בפעולות CRUD הבסיסיות, אשר כוללות יצירה, קריאה, עדכון ומחיקה של נתונים שמאוחסנים בבסיס נתונים. פרק 12 התמקד ברובו בעבודה עם הפקד SqlDataSource ובפקדים קשורי-הנתונים השונים שעומדים לרשותך. לבסוף, הפרק הקודם חקר את העולם של LINQ, האסטרטגיה העדכנית של מיקרוסופט לגישה לנתונים, שנועדה להאיץ את הדרך שבה נכתב קוד גישה לנתונים.

כדי לעזור לך להבין באמת את מושגי הליבה של גישה לנתונים, שלושת הפרקים הנזכרים התמקדו בעיקר בפקדי מקור נתונים והעקרונות שעומדים מאחוריהם, ופחות בהצגת הנתונים על ידי פקדים קשורי-נתונים. ללא ספק, ביישום של העולם הממשי אין זה מספיק, ויש צורך להציג נתונים בדרך ברורה ומושכת.

הפקדים קשורי-הנתונים שכלולים ב- ASP.NET 3.5 מספקים את האפשרויות הדרושות לשינוי דרך הצגת הנתונים. הם מאפשרים לשנות לחלוטין את עיצוב הנתונים (גופן, צבעים, ריווח, וכו'). בנוסף, באפשרותך להגדיר פקדים אלה באופן שיסתירו עמודות מסוימות, לשנות את כותרות העמודות, או אפילו לצבוע שורות אי-זוגיות בצבע שונה מזה של השורות הזוגיות. לבסוף, הודות למודל האירועים העשיר שמניע את הפקדים קשורי-הנתונים ואת פקדי מקור הנתונים, ביכולתך לשנות את המראה והתחושה של הפקדים באופן מותנה. לדוגמה, באפשרותך להסתיר את הקישור Delete בדף הניהול של Genres עבור אותם סגנונות שעדיין יש סקירות (ביקורות) שמקושרות אליהם.

בנוסף לדפים שנראים טוב, יכול היישום להפיק תועלת גם ממהירות כדי ליצור רושם חיובי אצל המשתמשים. ביצועים טובים הם תמיד מרכיב חשוב של כל אתר אינטרנט מצליח. ASP.NET מצטיינת ביכולות המיטמון (caching) שלה, שעוזרות להאצת ביצועי היישום.

בפרק זה תראה כיצד עליך ליישם עקרונות אלה באתר אינטרנט. תלמד בו גם בהרחבה על הנושאים הבאים:

- ⊙ שינוי העיצוב של הפקדים קשורי-הנתונים השונים על ידי שימוש בסגנונות.
- ⊙ שילוב סגנונות, ערכות נושא ו-skins ליצירת אתר אינטרנט שנראה טוב וקל לתחזוקה.

- ⊙ שימוש באירועים השונים שמעוררים פקדי הנתונים, כדי לשנות את המראה של פקדי הנתונים לפי התניה. לדוגמה, להסתיר או לבטל את פעולתם של פקדים מסוימים בממשק המשתמש כשלא סביר שהם יהיו גלויים או שיפעלו.
- ⊙ שימוש ביכולות המיטמון המובנות כדי לשפר את ביצועי האתר.

עיצוב הפקדים באמצעות סגנונות

פרקים 12 ו-13 הסבירו איך לעבוד עם הפקדים קשורי-הנתונים הרבים שכלולים ב-ASP.NET 3.5. למדת כיצד להציג ולערוך רשימות נתונים בעזרת פקדים כמו GridView, Repeater ו-ListView, וכיצד להפעיל פקדים שמטפלים ברשומה יחידה כמו DetailsView.

עד כה סמכת על המראה והתחושה המובנים של הפקדים שתוצאתם הייתה HTML משעמם ופשוט למראה. תרשים 14-1 מראה את GridView שיצרת בפרק 12 כדי לנהל את הסגנונות בבסיס הנתונים Planet Wrox.

	Id	Name	SortOrder
Edit Delete Select	1	Rap and Hip-Hop	14
Edit Delete Select	2	Pop	12
Edit Delete Select	3	Jazz	8
Edit Delete Select	4	Hard Rock	3
Edit Delete Select	5	Indie Rock	7
Edit Delete Select	6	Punk	1
Edit Delete Select	7	Rock	2
Edit Delete Select	8	Grunge	4
Edit Delete Select	9	Blues	9
Edit Delete Select	10	Reggae	11
1 2			

תרשים 14-1

ליצור הצגת טקסט וקישורים, מסתמך פקד זה על הגדרות ברירת המחדל של הדפדפן. תוצאת הדבר, בדרך כלל, היא קישורים סגולים או כחולים עם גופן ברירת המחדל, כמו Times New Roman. בנוסף, רוחב העמודות ברשת הוא בדיוק הרוחב הנדרש להצגת הטקסט שהן מכילות. לעין המתבונן יהיה קל הרבה יותר אילו במקום זאת היית יכול להציג את GridView כמו בתרשים 14-2.

	Name	Sort Order
Edit Delete	Rap and Hip-Hop	14
Edit Delete	Pop	12
Edit Delete	Jazz	8
Edit Delete	Hard Rock	3
Edit Delete	Indie Rock	7
Edit Delete	Punk	1
Edit Delete	Rock	2
Edit Delete	Grunge	4
Edit Delete	Blues	9
Edit Delete	Reggae	11
1 2		

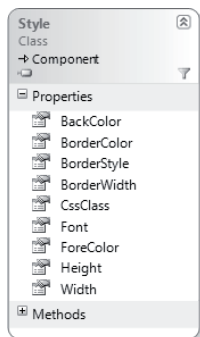
תרשים 14-2

העמודה של קישורי Edit ו-Delete מעט יותר רחבה כעת, ומופרדת בבירור מהתוכן הממשי של הרשת. העמודה Id הוסתרה והעמודה Name הורחבה אף היא. הצבעים השונים של הכותרת העליונה, הכותרת התחתונה, הפריטים, והפריטים המתחלפים הופכים את הנתונים ברשת לקלים הרבה יותר לקריאה.

שינוי GridView המשעמם-למראה של תרשים 1-14 לבעל המראה המטופח שמוצג בתרשים 2-14 ניתן להשגה בקלות על ידי שימוש בסגנונות (styles) של ASP.NET. בסעיף הבא תלמד כיצד להחיל סגנונות אלה על פקד יחיד בדף. בסעיף שאחריו תלמד כיצד להחיל את הסגנונות על ערכת נושא, כדי שכל הפקדים בקטע של האתר יוכלו להשתמש בסגנונות ביתר קלות ולעשות בהם שימוש חוזר. בפרק 7 כבר ראית כמה סגנונות בפעולה, כשהשתמשת בהם כדי לעצב את הפקדים Menu ו-TreeView. עם זאת, מכיוון שנעשה שימוש כה רב בסגנונות כדי לעצב פקדים קשורי-נתונים, הם ראויים להסבר מקיף יותר.

מבוא לסגנונות

לרבים מהפקדים קשורי-הנתונים ופקדי הניווט יש כמה מאפייני סגנון שמאפשרים לשנות את המראה והתחושה של הפקד. לדוגמה, לפקד GridView יש מאפיין RowStyle שמאפשר להתאים אישית את המראה של שורה יחידה בטבלה. באופן דומה, ל-DetailsView יש מאפיין CommandRowStyle שמשמש לשליטה במראה של שורת הפקודה שמכילה פקודות, כמו Insert, Delete, Cancel וכו'. בסופו של דבר, כל מאפיין סגנון יורש מהמחלקה Style שנמצאת במרחב השמות System.Web.UI.WebControls. תרשים 3-14 מציג מראה מסונן של תרשים מחלקה זו, על ידי הצגת המאפיינים היותר שכיחים שלה.



תרשים 14-3

כפי שניתן לראות משמותיהם, המאפיינים של המחלקה Style משמשים לשינוי המידע הקשור לסגנון באובייקטים, שעליהם חלה המחלקה. כל אחד ממאפיינים אלה מומר בסופו של דבר למאפיין CSS או לתכונת HTML, כמו background-color, color וכו'. סגנונות אחרים, כמו הסגנונות עבור הפקד GridView, מוסיפים מאפיינים קשורי-עיצוב שונים, כמו אפשרויות שונות לשליטה ביישור. בטבלה הבאה מוצגים המאפיינים הזמינים החשובים ביותר של המחלקות שנגזרות מ-Style. שים לב שלא כל מאפיין זמין לכל סגנון. IntelliSense מראה בדיוק באילו מאפיינים ניתן להשתמש בסגנון מסוים.

מאפיין	תיאור
BackColor ForeColor	מאפיינים אלה מאפשרים לשנות את צבע הרקע וצבע הטקסט של אלמנטים. הם ממופים למאפיינים background-color ו-color של CSS, בהתאמה.
BorderColor BorderStyle BorderWidth	מאפיינים אלה מאפשרים לשנות את הגבול של האלמנט שעליו חל הסגנון. הם ממופים ישירות למקביליהם ב-CSS, border-color, border-style ו-border-width.
CssClass	מאפיין זה מאפשר להקצות מחלקת CSS במקום מידע סגנון כלול. בתרגיל מאוחר יותר תראה כיצד להשתמש ב-CssClass. עליך לנסות להעדיף מאפיין זה על פני מאפייני הסגנון הפרטיים, מכיוון שהוא מקטין את נפח הדף, כפי שתראה מאוחר יותר.
Font	מאפיין זה מאפשר להגדיר את הגופן עבור האלמנט על ידי תת-המאפיינים השונים כמו Font-Names, Font-Size ו-Font-Bold. מאפיינים אלה נראים כמאפייני גופן שונים של CSS כמו font-size, font-family ו-font-weight.
HorizontalAlign VerticalAlign	מאפיינים אלה הם למעשה תכונות align ו-valign של אלמנט HTML שהם חלים עליו, ומאפשרים לשלוט ביישור של התוכן שבאלמנט. לדוגמה, כדי ליישר את הטקסט של כותרות העמודות של GridView לשמאל, למרכז או לימין תשתמש ב-HorizontalAlign.
Wrap	כשמאפיין זה מוגדר ל-False, הוא למעשה ההכרזה white-space: nowrap; של CSS. הוא קובע אם תא של טבלה מאפשר לעטוף את תוכנו לשורה חדשה.
Height Width	מאפיינים אלה מאפשרים לשלוט ברוחב ובגובה של האלמנטים שעליהם הם חלים. הם ממופים באופן ישיר למקביליהם ב-CSS.

לכל הפקדים קשורי-הנתונים השונים יש קבוצת סגנונות שונה, אם כי יש להם גם כמה סגנונות משותפים. הטבלה הבאה מציגה את הסגנונות הזמינים עבור המחלקות GridView ו-DataList, ומתארת את מטרתם. לשאר הפקדים קשורי-הנתונים יש סגנונות מעט שונים, אבל משמות הסגנונות תוכל להסיק מה הם עושים ולמה הם משמשים. לתיאור מלא של כל מאפייני הסגנון של הפקדים קשורי-הנתונים פנה לתיעוד MSDN. דרך טובה נוספת ללמוד יותר על הסגנונות הזמינים השונים, היא להשתמש ב- Visual Web Developer's Auto Format שמוסיף עבורך כמה סגנונות. מאוחר יותר תראה כיצד להשתמש ולשפר את הסגנונות שנוצרים על ידי התכונה Auto Format.

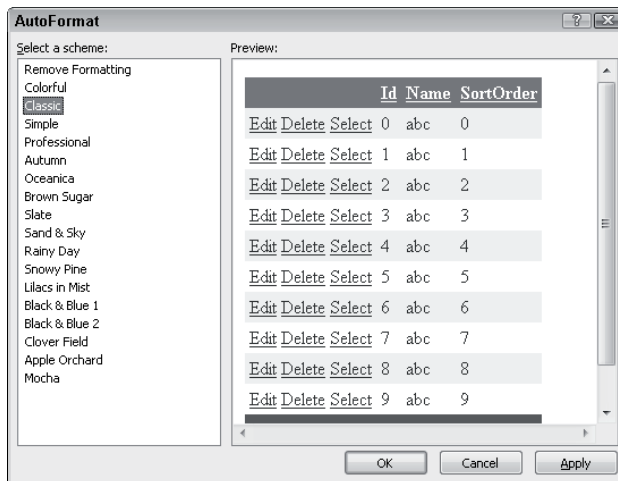
סגנון	תיאור
RowStyle AlternatingRowStyle	סגנונות אלה שולטים במראה של שורה יחידה ב-GridView. כברירת מחדל, RowStyle משפיע על כל השורות בעוד AlternatingRowStyle, כשהוא מוגדר, יכול לדרוס את RowStyle בשורות הזוגיות בלבד. מכיוון ש-DataList אינו פועל על שורות אלא על פריטי נתונים כלליים יותר, יש לו במקום זאת את המאפיינים ItemStyle ו-AlternatingItemStyle, אבל התנהגותם דומה במידה רבה.
SelectedRowStyle	סגנון זה אפשר להחיל על שורות נבחרות, והוא מאפשר להציג אותן באופן שונה משורות שלא נבחרו. SelectedItemStyle של הפקד DataList משמש לאותה מטרה.
EditRowStyle	אפשר להחיל סגנון זה על שורות שהן כרגע במצב עריכה. לדוגמה, כשלוחצים על הקישור Edit עבור שורה ב-GridView שבדף Genres בחלק Management, השורה עוברת למצב עריכה ואז מחילים עליה EditRowStyle זה. כדי להגדיר את העיצוב של פריט בר-עריכה ב-DataList, תשתמש במקום זאת ב-EditItemStyle.
EmptyDataRowStyle	סגנון זה מאפשר להגדיר את מראה השורה שמוצגת כשהפקד קשור למקור נתונים ריק. סגנון זה פועל יחד עם המאפיין EmptyDataText של הטבלה שמכילה את הטקסט שמוצג כשאינן רשומות, או עם EmptyDataTemplate שמאפשר להגדיר תבנית מותאמת אישית שתוצג כשמשתמשים במקור נתונים ריק. ל-DataList אין תמיכה של EmptyDataRowStyle.
HeaderStyle FooterStyle	סגנונות אלה שולטים במראה הכותרת העליונה והכותרת התחתונה של GridView.
PagerStyle	סגנון זה מאפשר להשפיע על המראה של סרגל הדפדוף שמוצג ב-GridView כשהדפדוף מאופשר. DataList אינו תומך בדפדוף לכשעצמו, ולכן אין לו PagerStyle.

לפקדים מסוימים, כמו Repeater ו-ListView, אין סגנונות מובנים. מכיוון שהם לכשעצמם אינם מוסיפים קוד HTML בנוסף לזה הנוצר על ידי הפקדים שהם מכילים, ומניחים לך להגדיר את המראה והתחושה בתבניות הרבות שיש להם, אין טעם שיהיו להם סגנונות נפרדים.

התרגיל הבא ינחה אותך בתהליך שיפור הפקד GridView בדף Genres של החלק Management. בדרך זו תלמד כיצד להשתמש בסגנונות הנזכרים לעיל עם הפקדים. בתרגיל מאוחר יותר בפרק זה תראה כיצד להעביר את המידע הקשור לסגנון אל ערכת נושא ולקובץ CSS, כדי לשפר את השימוש החוזר בקוד העיצוב וכדי להקטין את כמות קוד HTML שנשלחת לדפדפן עם כל בקשה.

בתרגיל זה תשתמש ביכולות העיצוב המובנות של VWD כדי לשנות את המראה של הפקד GridView. תראה כיצד יוצרת את הסגנונות הדרושים, כל אחד עם קבוצת מאפייני עיצוב הסגנון הרלוונטית לו. בתרגיל מאוחר יותר תלמד כיצד לשנות סגנונות אלה כדי שישמשו במאפייני CssClass. תוצאת שימוש זה היא סגנון מוגדר וקל הרבה יותר לניהול.

1. פתח את Genres.aspx מהתיקייה Management של היישום המרכזי של Planet Wrox שהשתמשת בו עד כה.
2. העבר את הדף ל- Design View ופתח את לוח Smart Tasks של הפקד GridView.
3. בראש הלוח, לחץ על הקישור Auto Format.
4. מרשימת סכמות העיצוב שמשמאל, בחר Classic. החלון Preview שמימין מתעדכן, ונראה עכשיו כמו בתרשים 14-4.



תרשים 14-4

5. לחץ OK כדי להחיל את ההגדרות. פקד GridView מתעדכן באופן מיידי, ומציג את סכמת העיצוב שנבחרה.
6. עבור ל- Markup View ובדוק את הסגנונות השונים שנוצרו. אתה אמור לראות את הסגנונות שלהלן, כמה מהם הוצבו לפני האלמנט <Columns> וכמה מהם מתחתיו:

```
<FooterStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />
<RowStyle BackColor="#EFF3FB" />
<PagerStyle BackColor="#2461BF" ForeColor="White" HorizontalAlign="Center" />
<SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True" ForeColor="#333333" />
<HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />
<EditRowStyle BackColor="#2461BF" />
<AlternatingRowStyle BackColor="White" />
```

7. שמור את השינויים שנערכו בדף, וקרא לו מהדפדפן על ידי הקשת Ctrl+F5. אתה אמור לראות את רשימת הסגנונות כשסכמת העיצוב שנבחרה מוחלת עליה.
8. פתח את מקור קוד HTML של הדף על ידי לחיצה ימנית על הדף בדפדפן ובחירת הפקודה View Source או View Page Source. גלול מעט מטה עד שתראה טבלת HTML שה-id שלה מוגדר ל-ctl00_cpMainContent_GridView1. תראה שלטבלה עצמה יש תכונה style שמגדירה את מאפייני צבע הטקסט והגבול.

```
<table cellpadding="4" cellspacing="0" border="0" id="ctl00_cpMainContent_GridView1" style="color:#333333;border-collapse:collapse;">
```

בנוסף, תראה שעל האלמנטים הרבים בתוך הטבלה (שורות הטבלה ואלמנטים של עוגן) מוחלות הגדרות סגנון שונות. לדוגמה, על שורות אי-זוגיות וזוגיות חל כעת הסגנון הבא:

```
<tr style="background-color:#EFF3FB;"> ... </tr>
<tr style="background-color:White;"> ... </tr>
```

כיצד זה פועל

אלמנטי הסגנון השונים שיצרת בצעד 5 מומרים לסגנונות השקולים להם ב-CSS ו-HTML. לדוגמה, אלמנט הסגנון BackColor של RowStyle ושל AlternatingRowStyle הוגדר עבור צבעי רקע שונים:

```
<RowStyle BackColor="#EFF3FB" />
<AlternatingRowStyle BackColor="White" />
```

כשהפקד מרנדר את קוד HTML שלו, הוא מחיל רקעים אלה בטבלה על שורת הפריטים והפריטים החלופיים:

```
<tr style="background-color:#EFF3FB;"> ... </tr>
<tr style="background-color:White;"> ... </tr>
```

אותו עיקרון חל על הסגנונות האחרים ב-GridView. כל סגנון מוחל על שטח אחר של אזור ב-GridView, וכל מאפיין סגנון מרונדר ישירות על אלמנט או במאפיין style כלול או כמאפיין HTML.

אם תתבונן בדפדפן במקור הדף, תראה "ניפוח" רב של הדף, מכיוון שהוגדרו מאפיינים עבור כל שורה בנפרד. הדבר מגדיל את נפח הדף בפרט כאשר ב-GridView מוצגת קבוצת תוצאות גדולה. כדי להקטין את גודל הדף כדי לשפר את ביצועיו, עליך להעביר את הגדרות הסגנון לערכת נושא של הדף ואז להשתמש בקובץ CSS ובמאפיין CssClass. בסעיף הבא תראה כיצד לעשות זאת.

שילוב סגנונות, ערכות נושא ו-Skins

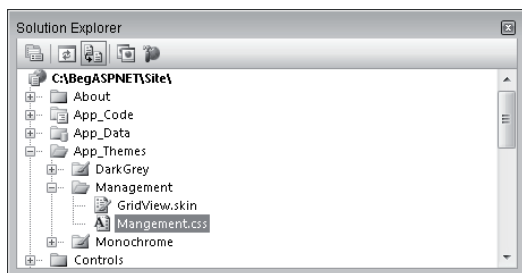
פרק 6 דן ביצירת דפי אינטרנט בעלי מראה עקבי תוך שימוש בדפים ראשיים, ערכות נושא ו-skins. לאחר שהוגדרה התשתית הבסיסית של ערכות הנושא קל כעת להוסיף ערכת נושא חדשה שתחול על כל החלק הניהולי של האתר. הדבר מאפשר במאמץ מזערי להקנות לחלק הניהולי מראה עקבי. כבר למדת כיצד ליצור קובץ skin כדי לשנות את המראה של לחצן. בתרגיל הבא תראה כיצד

להשתמש ברעיון זה כדי ליצור קובץ skin עבור פקד GridView. הקובץ מאפשר לעצב בבת אחת את כל פקדי ה-GridView שבתיקייה Management.

נסה זאת יצירת פתרונות סגנון מתקדמים

בתרגיל זה תעביר את מאפייני Style השונים מהדף Genres.aspx אל קובץ skin נפרד. בנוסף, תחליף את כל המאפיינים הכלולים, כמו ForeColor, BackColor, וכי, במאפיין יחיד של CssClass, ואת כל המידע הקשור לעיצוב תעביר לקובץ CSS של ערכת הנושא.

1. בסייר הפתרון, לחץ לחיצה ימנית על התיקייה App_Themes, בחר Add ASP.NET Theme ← Folder, ואז הקלד **Management** כשם הערכה החדשה.
2. לחץ לחיצה ימנית על התיקייה החדשה הזו ובחר Add New Item. הוסף קובץ skin בשם GridView.skin. עכשיו, הוסף גיליון סגנון בשם Management.css. אתה אמור לקבל סייר פתרון שנראה כמו תרשים 5-14.



תרשים 5-14

3. פתח את הדף Genres.aspx ב- Markup View ומחק את הסגנונות RowStyle, FooterStyle ו-SelectedRowStyle מכיוון שאינך זקוק להם בתרגיל זה. עבור כל אחד מארבעת הסגנונות הנותרים, הסר את כל המאפיינים שלהם והחלף אותם במאפיין יחיד של CssClass שתקרא לו בשם הסגנון בתוספת התחילית GridView. אתה אמור לקבל כעת את הסגנונות הבאים:

```
<AlternatingRowStyle CssClass="GridViewAlternatingRowStyle" />
<HeaderStyle CssClass="GridViewHeaderStyle" />
<PagerStyle CssClass="GridViewPagerStyle" />
<EditRowStyle CssClass="GridViewEditRowStyle" />
```

אל תדאג אם VWD מוסיפה קווי שגיאה מתחת לשמות הערכים של מאפייני CssClass. מכיוון שמחלקות CSS אינן מוגדרות עדיין, VWD לא מצאה אותן. מאוחר יותר תוסיף אותן לקובץ CSS של ערכת הנושא, אבל VWD לא תוכל למצוא אותן גם שם. אל דאגה, הן תפעלנה כהלכה בזמן ביצוע.

4. בעורך הקוד, בחר בכל ארבעת הסגנונות ואז חתוך אותם ל-clipboard על ידי Ctrl+X. עבור לקובץ GridView.skin, מחק את כל הקוד הקיים (טקסט ההערה שראית קודם), והדבק את ארבעת הסגנונות.

5. עטוף את ארבעת הסגנונות באלמנט <asp:GridView> כשהמאפיין runat שלו מוגדר ל-server והמאפיין CssClass מוגדר ל-GridView. אל תוסיף את המאפיין ID מכיוון שקבצי skin אינם זקוקים לו. אתה אמור לסיים עם הקוד הבא:

```
<asp:GridView runat="server" CssClass="GridView">
    <AlternatingRowStyle CssClass="GridViewAlternatingRowStyle" />
    <HeaderStyle CssClass="GridViewHeaderStyle" />
    <PagerStyle CssClass="GridViewPagerStyle" />
    <EditRowStyle CssClass="GridViewEditRowStyle" />
</asp:GridView>
```

6. פתח את הקובץ Management.css והוסף את קוד CSS הבא. מחק את הבורר body והקלד את הקוד הבא:

```
*
{
    font-family: Verdana, Arial, Sans-Serif;
}

body
{
    font-size: 80%;
}

.GridView, .GridView a
{
    color: #000;
}

.GridViewHeaderStyle, .GridViewPagerStyle, .GridViewEditRowStyle
{
    background-color: #bee4cf;
}

.GridViewAlternatingRowStyle
{
    background-color: #e1e1e1;
}

.GridViewHeaderStyle
{
    text-align: left;
}

.GridViewHeaderStyle a
{
    text-decoration: none;
}
```

7. מהתיקייה Management שהוספת קודם, פתח את הקובץ web.config והגדר את theme ל-Management:

```
<system.web>
```

```
<pages theme="Management"></pages>
```

```
</system.web>
```

8. חזור אל Genres.aspx וב- Markup View, תחת האלמנט Columns של הפקד GridView, מחק את העמודה הקשורה של Id. בדרך כלל אין למשתמשים צורך לראות את המזהה של פריטים בממשק המשתמש, מכיוון שלעיתים קרובות זהו פרט חסר משמעות מבחינתם. הגדר מאפיין ItemStyle-Width עבור CommandField ל-100 פיקסל, ועבור העמודה Name הגדר ל-200 פיקסל. לבסוף, הגדר את ShowSelectButton של CommandField ל-False. אתה אמור לקבל עם ה-GridView הבא:

```
<asp:GridView ID="GridView1" runat="server" AllowPaging="True"
```

```
AllowSorting="True" AutoGenerateColumns="False" DataKeyNames="Id"
```

```
DataSourceID="SqlDataSource1" GridLines="None" CellPadding="4"
```

```
ForeColor="#333333" EmptyDataText="There are no data records to display.">
```

```
<Columns>
```

```
<asp:CommandField ShowDeleteButton="True" ShowEditButton="True"
```

```
ShowSelectButton="False" ItemStyle-Width="100px">
```

```
</asp:CommandField>
```

```
<asp:BoundField DataField="Name" HeaderText="Name"
```

```
SortExpression="Name" ItemStyle-Width="200px">
```

```
</asp:BoundField>
```

```
<asp:BoundField DataField="SortOrder" HeaderText="Sort Order"
```

```
SortExpression="SortOrder">
```

```
</asp:BoundField>
```

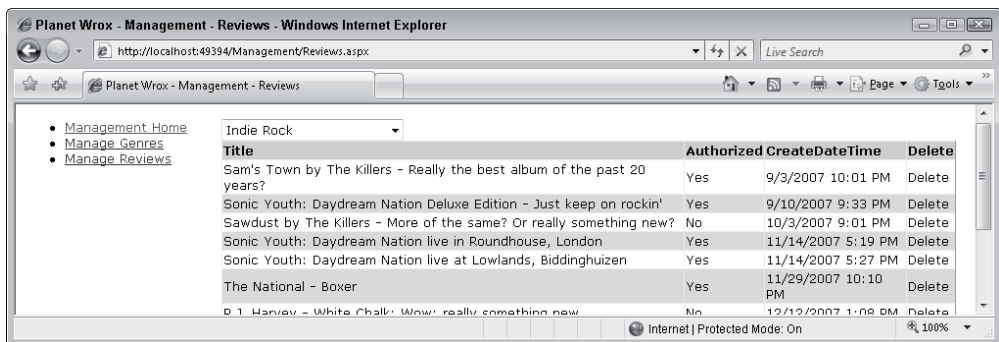
```
</Columns>
```

```
</asp:GridView>
```

9. שמור את כל השינויים ואז פתח את Genres.aspx בדפדפן. אתה אמור לראות כעת את רשימת הסגנונות שהוצגה בתרשים 2-14.

10. בתפריט הראשי של Management, לחץ על Manage Reviews כדי לפתוח את הדף Reviews. מהרשימה הנפתחת בחר סגנון כדי להציג רשימה של סקירות. שים לב שעל הרשימה Reviews שמוצגת בתרשים 6-14 מוחלים כעת אותם סגנונות שחלו על הרשימה Genres שראית קודם.

ברור שאפשר עדיין להתאים את הפקדים ברמת הדף. בעוד שה-skin מגדיר את המראה והתחושה הגלובליים של GridView, אתה עדיין יכול להגדיר מאפיינים יחידים על עמודות, כפי שעשית עם ItemStyle-Width בדף Genres.



תרשים 14-6

כיצד זה פועל

המושגים שנחשפת להם בתרגיל זה אמורים להיות מוכרים לך כעת. בפרק 6 ראית כיצד ליצור ולהחיל ערכות נושא ו-skins, ובתרגיל הקודם ראית כיצד להשתמש בסגנונות הפקדים השונים. הדבר היחיד שהוא באמת חדש בתרגיל, זו הדרך שבה משולבים מושגים אלה. העברת הכרזות סגנון הפקד אל קובץ skin נפרד, שבתורו הופך לחלק מערכת נושא, יצרה פתרון גמיש מאוד וקל לתחזוקה. אם תרצה לשנות את העיצוב של כל פקדי GridView שבחלק Management, כל שעליך לעשות כעת הוא לשנות את קוד CSS הרלוונטי בקובץ Management.css. אם תצטרך לערוך שינויים בסגנונות אחרים, אל תשכח להוסיף אותם תחילה לקובץ GridView.skin. אם ברצונך לראות כיצד מוחלים הסגנונות החדשים, פתח את המקור של הדף בדפדפן על ידי הפקודה View Source. במקום סגנונות כלולים יוחלו כעת מאפייני ה-class הרלוונטיים.

למרות שסגנונות skins וערכות נושא הם כלים רבי עוצמה כדי לעצב דפי אינטרנט, הרי שלעיתים קרובות תמצא שהם פתרון מסוג "הכל, או לא כלום". לדוגמה, כשאתה יוצר אלמנטים מסוג ItemStyle ו-AlternatingItemStyle, הם חלים על כל שורה ושורה בטבלה. אבל, מה קורה כשרוצים לשנות את המראה והתחושה של שורות ספורות בלבד? או כשרוצים לשנות כמה שורות על פי הנתונים הממשיים שהשורות מכילות? בסעיף הבא תראה כיצד לבצע עיצוב מותנה, ויותר מכך, תוך שימוש באירועים מותאמים אישית.

טיפול באירועים

ברוב הפרקים עד כה למדת על הדרך שבה פקדי ASP.NET מעוררים אירועים. למדת כיצד לטפל באירועים אלה בקוד של מטפל אירוע, שמוסיפים אותו בדרך כלל לקובץ קוד ברקע של הדף. לדוגמה, כתבת קוד לטיפול באירוע Click של הלחצן Button. בנוסף, בפרק הקודם למדת כיצד להגיב על אירועים שונים - כמו Inserting ו-Inserted - שקורים בדיוק לפני ואחרי פעולת הכנסת נתונים לבסיס הנתונים. עם זאת, רוב הפקדים חושפים אירועים רבים יותר, שמאפשרים להתחבר לשלבים שונים במחזור החיים של הדף והפקדים.

הבנה טובה של האירועים השונים שמתעוררים במהלך מחזור החיים של פקד, והסדר שבו הם מתעוררים, היא ידע חשוב למתכנת ASP.NET. היכולת להתחבר אל תוך מחזור החיים של פקד, ולהתאים חלקים מהפלט תוך כדי מהלך הדברים, מאפשרת ליצור דפי אינטרנט גמישים ודינמיים שמבצעים בדיוק את מה שברצונך שיבצעו.

כדי שתוכל להבין טוב יותר את האירועים השונים והסדר שבו הם מתעוררים, עיין בסעיף הבא כדי ללמוד על הצעדים הבסיסיים לאורך חייו של פקד ASP.NET. אמנם לא תראה כל אירוע שמתעורר בזמן התהליך, אבל תכיר את אלה שסביר ביותר שתשתמש בהם. בסעיפים מאוחרים יותר תראה כיצד להשתמש באירועים אלה כדי לשנות את התנהגות דפי האינטרנט.

ביקור חוזר במחזור החיים של דפי ופקדי ASP.NET

בפרק 6 למדת אודות השלבים השונים בחיי דף. למדת אודות אירועים שונים כמו PreInit, Load, PreRender ו-Unload. בנוסף לאירועים אלה, שמעוררים על ידי דף ASPX, יכולים גם כל שאר הפקדים בדפי ASPX לעורר את האירועים שלהם. אירועים אלה יכולים להיות פשוטים, כמו האירוע Click של הלחצן Button (שמתעורר כתוצאה מפעולת משתמש), או אירועים מורכבים יותר, כמו Inserting שמתעורר על ידי פקדים כמו LinqDataSource ו-SqlDataSource, או האירוע DataBound שפקדים קשורי-נתונים שונים מעוררים אותו.

מטרת התרגיל הבא היא להציג תמונה מקיפה על האירועים השונים במהלך מחזור החיים של דף או פקד, שאפשר להתחבר אליהם, ועל הסדר שבו מעוררים אותם. על ידי שימוש ב-LinqDataSource, התרגיל יראה כיצד להגדיר דף שמציג נתונים מהטבלה Genres. בנוסף, תציב בדף לחצן שבו תוכל להשתמש כדי לגרום לפעולת postback, במטרה לראות כיצד זה משפיע על מהלך הדברים. אחר כך תחבר מספר מטפלי אירוע לכמה אירועים מעניינים של הפקדים שבדף. בתוך כל מטפל אירוע תעדכן פקד Label באופן שתוכל לראות את סדר התרחשות הדברים.

נסה זאת ראיית מחזור החיים של הדף ושל הפקדים בפעולתו

1. בתיקייה Demos צור קובץ חדש בשם Events.aspx. ודא שהוא מבוסס על תבנית הדף המותאמת אישית שלך כדי שיירש מ-BasePage. קבע את כותרת הדף ל-Events Demo.
2. העבר את הדף ל-Design View, שחרר GridView לתוך שומר המקום cpMainContent, וחבר את GridView לפקד LinqDataSource על ידי לוח Smart Tasks של GridView. קשור את הפקד LinqDataSource אל הטבלה Genres. אין צורך להגדיר פסקת Where או פסקת Order By או להגדיר מאפיינים מתקדמים כלשהם של מקור הנתונים.
3. חזור ללוח Smart Tasks של הפקד GridView ובחר בתיבת הסימון השנייה כדי לאפשר מיון. כשתסיים, דף ASPX שלך צריך להכיל את הקוד הבא:

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
    DataKeyNames="Id" DataSourceID="LinqDataSource1" AllowSorting="True">
  <Columns>
    <asp:BoundField DataField="Id" HeaderText="Id" InsertVisible="False"
```

```

        ReadOnly="True" SortExpression="Id" />
        <asp:BoundField DataField="Name" HeaderText="Name"
            SortExpression="Name" />
        <asp:BoundField DataField="SortOrder" HeaderText="SortOrder"
            SortExpression="SortOrder" />
    </Columns>
</asp:GridView>
<asp:LinqDataSource ID="LinqDataSource1" runat="server"
    ContextTypeName="PlanetWroxDataContext" TableName="Genres">
</asp:LinqDataSource>

```

4. ודא שאתה נמצא ב-Markup View, גרור מארגז הכלים שני פקדי Label ושחרר אותם בסימני העריכה לפני הפקד GridView, ואז מחק את המאפיין Text שלהם. הגדר את ID של הפקדים ל-lblNoPostBack ו-lblPostBack בהתאמה.

5. עטוף את שתי התוויות באלמנט <table> בעל שורה אחת ושני תאים, כדי שהתוויות תופענה זו לצד זו. מכיוון שאתה זקוק לתגיות ספורות בלבד, תוכל להקליד אותן ישירות ב-Markup View, במקום להשתמש בתפריט Table שפועל רק ב-Design View. לבסוף, מעל כל תווית הוסף אלמנט <h1> עם "Postback" או "No Postback" כטקסט. כעת תקבל את הקוד הבא:

```

<table>
<tr>
<td>
<h1>No Postback</h1><asp:Label ID="lblNoPostBack" runat="server"></asp:Label>
</td>
<td>
<h1>Postback</h1><asp:Label ID="lblPostBack" runat="server"></asp:Label>
</td>
</tr>
</table>

```

6. עבור למצב Design View, שחרר פקד Button מתחת ל-GridView ולחץ עליו לחיצה כפולה ב-Design View כדי להגדיר בקוד ברקע מטפל אירוע עבור האירוע Click שלו.

7. עבור חזרה ל-Design View ולחץ על הדף לחיצה כפולה, כדי להגדיר מטפל עבור האירוע Load של הפקד Page. המטפל יופיע בקוד ברקע של הדף Events.aspx.

8. עבור ל-Design View פעם נוספת, לחץ על GridView, ופתח את טבלת המאפיינים שלו על ידי הקשת F4. עבור לקטגוריית Events ולחץ לחיצה כפולה על האירועים הבאים, כדי להגדיר עבורם מטפלים בקוד ברקע.

הערה: על ידי הקשת Ctrl+Tab אפשר לחזור במהירות ממסך קוד ברקע, שם נוצר מטפל האירוע, חזרה אל דף העיצוב כדי ליצור את המטפל הבא:

- ⊙ Sorted
- ⊙ Sorting

- ⊙ RowCreated
- ⊙ DataBinding
- ⊙ DataBound
- ⊙ RowDataBound

9. חזור על הצעד הקודם, אבל כעת הגדר את האירועים הבאים עבור הפקד `LinqDataSource`.

- ⊙ ContextCreating
- ⊙ Selecting

10. ודא שאתה נמצא בקוד ברקע של הדף ומתחת למטפל האירוע האחרון (אבל עדיין בתחום הגדרת המחלקה). הוסף את השיטה הבאה שכותבת מעט טקסט לאחת משתי התוויות, בהתאם לסוג הבקשה הנוכחית לדף, בין אם היא תוצאה של `postback` או לא:

VB.NET

```
Private Sub WriteMessage(ByVal handlerName As String)
    If Page.IsPostBack Then
        lblPostBack.Text &= handlerName & "<br />"
    Else
        lblNoPostBack.Text &= handlerName & "<br />"
    End If
End Sub
```

C#

```
private void WriteMessage(string handlerName)
{
    if (Page.IsPostBack)
    {
        lblPostBack.Text += handlerName + "<br />";
    }
    else
    {
        lblNoPostBack.Text += handlerName + "<br />";
    }
}
```

11. הוסף את הקוד הבא לכל אחד ממטפלי האירוע שהגדרת. תפקיד הקוד לקרוא לשיטה `WriteMessage` ולהעביר לה את שם האירוע, השיטה `WriteMessage` תוסיף כעת את השם לשני פקדי `Label`.

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Me.Load
    WriteMessage("Page_Load")
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
{
    WriteMessage("Page_Load");
}
```

12. לבסוף, הוסף לדף באופן ידני את מטפל האירוע הבא:

VB.NET

```
Protected Sub Page_PreRenderComplete(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Me.PreRenderComplete
    WriteMessage("Page_PreRenderComplete<br />-----")
End Sub
```

C#

```
protected void Page_PreRenderComplete(object sender, EventArgs e)
{
    WriteMessage("Page_PreRenderComplete<br />-----");
}
```

האירוע PreRenderComplete מתעורר מאוחר מאוד במחזור החיים של הפקד Page. עובדה זו הופכת אותו למקום אידיאלי להכנסת שורה בתחתית רשימת האירועים. בדרך זו אפשר לראות בבירור איזו סדרת אירועים שייכת לאחרת, דבר שבתורו עוזר להבין איזה אירועים מעוררים במהלך איזה postback.

13. שמור את כל השינויים ופתח את הדף בדפדפן. בנוסף ל-GridView עם הסגנונות הזמינים, אתה אמור לראות גם רשימה עם שמות אירועים מתחת לכותרת No Postback:

```
Page_Load
LinqDataSource1_Selecting
LinqDataSource1_ContextCreating
GridView1_DataBinding
GridView1_RowCreated
GridView1_RowDataBound
GridView1_RowCreated
GridView1_RowDataBound
...
GridView1_DataBound
Page_PreRenderComplete
-----
```

- ⊙ שים לב שהאירועים RowCreated ו-RowDataBound חוזרים על עצמם מספר רב של פעמים - פעם אחת עבור כל רשומה במקור הנתונים.
- ⊙ לחץ על הלחצן כדי לגרום ל-postback. התוויית No Postback לא תשתנה, אבל התוויית Postback מראה כעת את רשימת שמות האירועים הבאה:

```

...
GridView1_RowCreated
GridView1_RowCreated
GridView1_RowCreated
GridView1_RowCreated
Page_Load
Button1_Click
Page_PreRenderComplete
-----

```

לחץ על אחת מכותרות העמודות של GridView כדי למיין את הנתונים שהיא מציגה. שים לב שהטקסט של התווית השנייה מורחב עם קבוצה שנייה של שמות אירועים.

כיצד זה פועל

תרגיל זה אינו מורכב מבחינה טכנית. הגדרת קבוצה של מטפלי אירועים עבור הפקדים השונים בדף שלך, בתוך מטפל האירוע קראת לשיטה שבודקת האם הדף נטען כעת בפעם הראשונה או שהוא נטען בגלל postback. לבסוף, הקוד מעדכן את אחד משני הפקדים Label עם שם האירוע שעורר את מטפל האירוע.

מה שמעניין בתרגיל זה הוא הסדר שבו קורים האירועים. התבונן ברשימה הראשונה, זו שמוצגת כשהדף נטען לראשונה:

```

Page_Load
LinqDataSource1_Selecting
LinqDataSource1_ContextCreating
GridView1_DataBinding
GridView1_RowCreated
GridView1_RowDataBound
GridView1_RowCreated
GridView1_RowDataBound
...
GridView1_DataBound
Page_PreRenderComplete
-----

```

תחילה מתעורר האירוע Page_Load. אחר כך, פקד GridView רואה שהוא מחובר לפקד Selecting ו-LinqDataSource ומבקש מהאחרון את הנתונים שלו. הדבר גורם להתעוררות האירועים ContextCreating-1. כאשר מתקבלים נתונים מ-LinqDataSource, מעורר GridView את האירוע DataBinding שלו כדי לאותת שהוא עומד לקשור את הנתונים. הפקד GridView מתחיל ליצור שורות. עבור כל פריט במקור הנתונים הוא יוצר שורה, הוא מעורר RowCreated, קושר את נתוני הפריט לשורה, ולבסוף קורא ל-RowDataBound. אם תספור בזירות את מספר הפעמים שנקראים RowCreated ו-RowDataBound, תמצא שהם נקראים פעמיים יותר מהמספר הממשי של פריטים במקור הנתונים. הסיבה היא שאותו אירוע מתעורר גם כשהפקד יוצר את השורות Header ו-Footer שלו. בתרגיל מאוחר יותר תלמד איך להבחין בין שורות אלו במטפל האירוע.

לבסוף, כשה-GridView מסיים ליצור ולקשור את כל השורות במקור הנתונים, הוא מעורר את האירוע DataBound שלו.

הדברים שונים למדי בזמן postback. כשאתה לוחץ על הלחצן כדי לגרום ל-postback, מתעוררים האירועים הבאים:

```
GridView1_RowCreated
GridView1_RowCreated
GridView1_RowCreated
GridView1_RowCreated
Page_Load
Button1_Click
Page_PreRenderComplete
-----
```

שים לב שהאירועים RowDataBound או DataBound אינם מופיעים ברשימה זו, וכך גם LinqDataSource. פקד GridView מסוגל לבנות מחדש את הפקד כולו מ-ViewState, והדבר מבטל את הצורך לגשת שוב לבסיס הנתונים. בעודו מקבל את הנתונים מ-ViewState, פקד GridView עדיין צריך ליצור מחדש כל שורה ברשת, כך שעדיין תראה את האירועים RowCreated. האירוע Page_Load מופיע לקראת סוף הרשימה ואחריו האירוע Click של הפקד Button. חשוב להבין ולזכור כי כשהמשתמש מעורר אירועי פקד - כמו למשל האירוע Click של הפקד Button, או SelectedIndexChanged של DropDownList - האירועים מופעלים לאחר האירוע Load של הדף. שים לב שאירוע Load זה איננו התחלת מחזור החיים של הפקד Page. לפני המופע של Page, האירוע Page נוצר למעשה וגם עורר את האירוע Init שלו.

בסיום התרגיל לחצת על כותרת עמודה כדי למיין את הנתונים בטבלה. במקרה זה, GridView יודע שעליו למיין את הנתונים המוצגים. הוא אינו יכול לבצע זאת בעצמו, ולכן הוא מבקש מ-LinqDataSource עותק רענן של הנתונים, לפי הסדר שהמשתמש ביקש. בדיוק כמו בפעם הראשונה שבה נטען הדף, אתה רואה שמופיעים האירועים השונים RowCreated ו-DataBound.

אם ברצונך לראות אירועים אחרים בפעולה, עליך לחזור על צעדים 8 ו-11 של התרגיל הקודם, ולהגדיר מטפלים עבור האירועים השונים. כדי לראות את השפעת ViewState, נסה לבטל את ViewState ברמת הפקד (לדוגמה, עבור GridView) או ברמת הדף.

למרות שהתרגיל הקודם אמור לעזור לך להגיע להבנת אירועי הפקדים השונים והסדר שבו הם מתעוררים, הרי שהדוגמה עצמה היא חסרת-תועלת (כמעט) ביישום אמיתי. עם זאת, תוכל להשתמש באותו עיקרון בדיוק כדי להתחבר לדף ולערוך שינויים בדף עצמו, או בפקד כלשהו בדף. בתרגיל הבא תראה כיצד לשנות את המראה של שורות במקור הנתונים כתלות בנתונים שמוצגים.

מחזור החיים של דף ASP.NET ואירועים בפקדי נתונים

כפי שהוסבר, הפקד GridView מעורר את האירועים RowCreated ו-RowDataBound עבור כל שורה שהוא מוסיף לפלט שלו. אירועים אלה מתאימים מאוד במצב בו ביכולתך לבחון את הנתונים ואז, בהתבסס על תנאים מסוימים, לנקוט בפעולה המתאימה. לדוגמה, באפשרותך להשתמש באירועים אלה כדי לבדוק אם סקירה שנשלפה אכן מורשית להצגה. אם היא אינה מורשית (כלומר, אסור להציגה באתר), ביכולתך לדוגמה להוריד את השורה כך שלא תוצג. בתרגיל הבא תראה לאיזה קוד אתה זקוק כדי להשיג זאת.

נסה זאת התממשקות לאירוע RowDataBound

בתרגיל זה תכתוב מטפל אירוע עבור האירוע RowDataBound של הפקד GridView בדף Reviews של חלק Management שבאתר. בתוך אירוע זה תוכל לאבחן את פריט הנתונים שנקשר לשורת GridView. הדבר מאפשר לראות אם הפריט מורשה או לא. אם הפריט איננו מורשה, תחיל עליו מחלקת CSS שונה, כדי להבליט את השורה הזו לעומת האחרות. (מטרת התרגיל כאן היא להראות השתלטות על התצוגה בהתאם לתנאים. סביבת הניהול היא בהחלט מקום בו תרצה לדעת כמנהל אילו רשומות מורשות ואילו לא).

1. פתח את הדף Reviews.aspx מהתיקייה Management, ועבור ל- Design View. אתר את GridView ופתח את טבלת המאפיינים שלו. קבע את הגדרת המאפיין DataKeyNames של GridView ל-Id. הדבר אומר ל-GridView מהו המפתח הראשי של הסקירה בבסיס הנתונים, מידע שנחוץ כדי לתמוך בהתנהגות המחיקה למחיקת סקירות מ-GridView.
2. העבר את טבלת המאפיינים לקטגוריה Events. קבע מטפל אירוע לאירוע RowDataBound.
3. אם אתה משתמש ב-VB.NET, הוסף בראש הקוד ברקע של המסמך את שורת הקוד הבאה:

```
Imports System.Data
```

4. הוסף את הקוד הבא בתוך מטפל האירוע שנוצר על ידי VWD:

VB.NET

```
Protected Sub GridView1_RowDataBound(ByVal sender As Object, _  
    ByVal e As System.Web.UI.WebControls.GridViewRowEventArgs) _  
    Handles GridView1.RowDataBound  
    Select Case e.Row.RowType  
        Case DataControlRowType.DataRow  
            Dim myRowView As DataRowView = CType(e.Row.DataItem, DataRowView)  
            If Not Convert.ToBoolean(myRowView("Authorized")) Then  
                e.Row.CssClass = "HighContrast"  
            End If  
        End Select  
    End Sub
```

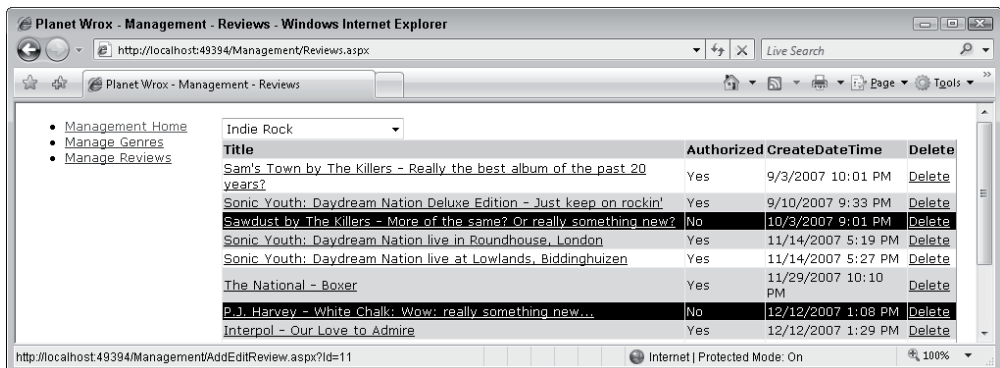
C#

```
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    switch (e.Row.RowType)
    {
        case DataControlRowType.DataRow:
            DataRowView myDataRowView = (DataRowView)e.Row.DataItem;
            if (!Convert.ToBoolean(myDataRowView["Authorized"]))
            {
                e.Row.CssClass = "HighContrast";
            }
            break;
    }
}
```

5. פתח את Management.css מערכת הנושא Management והוסף את הכלל הבא בקוד CSS:

```
.HighContrast, .HighContrast a
{
    background-color: Black;
    color: White;
}
```

6. שמור את השינויים שערכת בכל הקבצים הפתוחים ואז בקש בדפדפן את Reviews.aspx. מהרשימה הנפתחת שבראש הדף בחר פריט כדי להציג את הסקירות של אותו סגנון. כשתמצא סגנון שיש לו סקירה אחת, או יותר, שטרם קיבלה הרשאה (אם אתה משתמש בבסיס הנתונים שכולל את הקוד הנטען של ספר זה, בחר Indie Rock), אתה אמור לראות את הפריט מוצג עם צבעי רקע וטקסט שונים, כפי שמוצג בתרשים 14-7. אם אינך רואה פריט שאינו מורשה, לחץ תחילה על הכותרת של סקירה, נקה את תיבת הסימון Authorized ואחר כך לחץ על הקישור Update.



תרשים 14-7

בתרשים 14-7 אפשר לראות ששתי סקירות אינן מורשות. כדי להדגיש זאת מוצגת כל אחת משתי השורות ברקע שחור וטקסט לבן (נגטיב).

כיצד זה פועל

למרות שהתרגיל קצר, הוא מדגים דרך רבת-עוצמה להתחבר אל אירועים שונים של פקד ולשנות את התצוגה של פקד בסיסי. כדי לראות כיצד זה פועל התבונן בקוד הבא:

VB.NET

```
Protected Sub GridView1_RowDataBound(ByVal sender As Object, _  
    ByVal e As System.Web.UI.WebControls.GridViewRowEventArgs) _  
    Handles GridView1.RowRowDataBound  
    Select Case e.Row.RowType  
        Case DataControlRowType.DataRow  
            ...  
    End Select  
End Sub
```

C#

```
protected void GridView1_RowDataBound (object sender, GridViewRowEventArgs e)  
{  
    switch (e.Row.RowType)  
    {  
        case DataControlRowType.DataRow:  
            ...  
    }  
}
```

האירוע RowDataBound מקבל כפרמטר מופע של GridViewRowEventArgs, מחלקה שמספקת מידע אודות השורה והנתונים הנקשרים בשלב זה. אחד המאפיינים של מחלקה זו הוא Row, שמייצג את השורה הממשית שהתווספה לפקד GridView. שורה זו בתורה מכילה מאפיין enum RowType שאפשר לבחון אותו כדי לראות מהו סוג השורה שמתווספת. enum זה מכיל ששה חברים שונים שממופים ישירות לסוגים השונים של שורות שה-GridView יכול להכיל: DataRow עבור שורות רגילות ומתחלפות, EmptyDataRow עבור שורות נתונים ריקות, Header ו-Footer עבור שורות הכותרת העליונה והתחתונה שמוצבות בראש ובתחתית GridView, וכו'. מכיוון שברצונך לשנות את המראה של שורת נתונים ממשית, הקוד בבלוק Case מתעורר רק עבור שורות רגילות ומתחלפות.

בתוך הבלוק Case מבוצע הקוד הבא:

VB.NET

```
Dim myRowView As DataRowView = CType(e.Row.DataItem, DataRowView)  
If Not Convert.ToBoolean(myRowView("Authorized")) Then  
    e.Row.CssClass = "HighContrast"  
End If
```

C#

```
DataRowView myDataRowView = (DataRowView)e.Row.DataItem;  
if (!Convert.ToBoolean(myDataRowView["Authorized"]))  
{  
    e.Row.CssClass = "HighContrast";  
}
```

המאפיין DataItem מכיל הפניה לאובייקט פריט הנתונים הנקשר. כשמשמשים בפקד LinqDataSource, הפריט DataItem הוא למעשה מופע בעל סוג מובהק של אחד מהאובייקטים של LINQ to SQL, כמו Review או Genre. עוד מעט תראה כיצד זה פועל, אבל כשמשמשים בפקד SqlDataReader, כל שורה מוצגת כ-DataRowView, אובייקט של .NET. שעוטף שורה שמוחזרת מבסיס הנתונים.

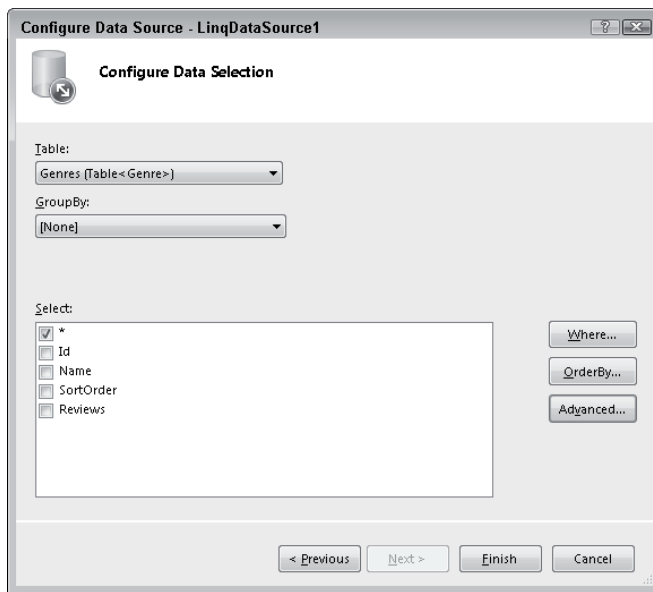
לבסוף, מצרפים את DataItem לאובייקט DataRowView, ואז נקבע אינדקס לאובייקט - תוך שימוש ב- myRowView("Authorized") (עבור VB.NET) או ב-myRowView["Authorized"] (עבור C#) - כדי להגיע לערך הבוליאני של העמודה Authorized. אם Authorized מוגדר כ-False, תראה שמאפיין CssClass חדש חל על המופע Row, וגורם לשינוי צבעי הרקע והטקסט של השורה כולה על ידי CSS.

מכיוון שהדף Reviews משתמש בפקד SqlDataReader, עליך להפעיל כמה תכסיסים כדי להגיע לנתונים הממשיים של הסקירה. תחילה צריך להמיר את DataItem ל-DataRowView, ואז עליך לקבוע אינדקס עבור DataRowView הזה ולנסות למצוא את העמודה Authorized הנכונה. האם לא יהיה זה קל הרבה יותר אם נוכל לגשת באופן ישיר אל אובייקטים בעלי סוג מובהק כמו המופעים Review ו-Genre? בתרגיל הבא תלמד לעשות זאת.

נסה זאת טיפול באירועים עם אובייקטים בעלי סוג מובהק

בתרגיל זה תיצור דף נוסף לרישום הסגנונות השונים שבבסיס הנתונים, אשר דומה לדף הנוכחי Genres.aspx שבתיקייה Management. הדף מפרט את כל הסגנונות הזמינים, וכולל גם לחצן Delete כדי שיאפשר לך למחוק אותם אם תרצה. עם זאת, הלחצן Delete מושבת כל עוד יש סקירות שקשורות לסגנון.

1. הוסף טופס אינטרנט חדש בשם GenresLinq.aspx לתיקייה Management של האתר. ודא שהוא מבוסס על הדף הראשי Management. קבע כותרת דף בשם Events with the LinqDataSource.
2. עבור ל- Design View. גרור פקד GridView מארגז הכלים ושחרר אותו בדף.
3. חבר את GridView אל LinqDataSouece, שבתורו קשור לטבלה Genres. בחר את Genres ברשימה הנפתחת Table. ברשימה Select, בחר את הכוכבית (*). עכשיו לחץ על לחצן Advanced ואפשר מחיקה של סגנונות על ידי סימון תיבת הסימון הראשונה בתיבת הדו-שיח Advanced Options. לחץ OK. תיבת הדו-שיח תיראה כעת כמו תרשים 14-8.



תרשים 14-8

4. לחץ Finish כדי לסגור את תיבת הדו-שיח. כשתסיים אמור להיות בידך הקוד הבא עבור הפקד LinqDataSource:

```
<asp:LinqDataSource ID="LinqDataSource1" runat="server"
    ContextTypeName="PlanetWroxDataContext"
    TableName="Genres" EnableDelete="True">
</asp:LinqDataSource>
```

5. בלוח Smart Tasks של הפקד GridView בחר Enable Deleting, כדי לאפשר מחיקה גם ב-GridView.

6. כשאתה נמצא עדיין בלוח Smart Tasks של הפקד GridView, לחץ על הקישור Edit Columns. ברשימת השדות הקרויה Selected, לחץ על הפריט Delete, ולבסוף לחץ על הקישור הכחול שבצד ימין של תיבת הדו-שיח כדי להמיר את עמודת המחיקה ל-TemplateField. לחץ OK כדי לשחרר את תיבת הדו-שיח. GridView אמור להיות מורכב כעת מהקוד הבא:

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
    DataKeyNames="Id" DataSourceID="LinqDataSource1">
    <Columns>
        <asp:TemplateField ShowHeader="False">
            <ItemTemplate>
                <asp:LinkButton ID="LinkButton1" runat="server" CausesValidation="False"
                    CommandName="Delete" Text="Delete"></asp:LinkButton>
            </ItemTemplate>
        </asp:TemplateField>
        <asp:BoundField DataField="Id" HeaderText="Id" InsertVisible="False"
            ReadOnly="True" SortExpression="Id"></asp:BoundField>
```

```

<asp:BoundField DataField="Name" HeaderText="Name"
    SortExpression="Name"></asp:BoundField>
<asp:BoundField DataField="SortOrder" HeaderText="SortOrder"
    SortExpression="SortOrder"></asp:BoundField>
</Columns>
</asp:GridView>

```

7. השתמש בקטגוריה Event של טבלת המאפיינים של הפקד GridView כדי להוסיף מטפל אירוע עבור האירוע RowDataBound שלו.
8. כתוב את הקוד הבא בתוך מטפל האירוע:

VB.NET

```

Protected Sub GridView1_RowDataBound(ByVal sender As Object, _
    ByVal e As System.Web.UI.WebControls.GridViewRowEventArgs) _
    Handles GridView1.RowDataBound
    Select Case e.Row.RowType
    Case DataControlRowType.DataRow
        Dim myGenre As Genre = CType(e.Row.DataItem, Genre)
        Dim deleteButton As LinkButton = _
            CType(e.Row.FindControl("LinkButton1"), LinkButton)
        deleteButton.Enabled = myGenre.Reviews.Count = 0
    End Select
End Sub

```

C#

```

protected void GridView1_RowDataBound (object sender, GridViewRowEventArgs e)
{
    switch (e.Row.RowType)
    {
        case DataControlRowType.DataRow:
            Genre myGenre = (Genre)e.Row.DataItem;
            LinkButton deleteButton = (LinkButton) e.Row.FindControl("LinkButton1");
            deleteButton.Enabled = myGenre.Reviews.Count == 0;
            break;
    }
}

```

9. שמור את כל השינויים וקרא לדף מתוך הדפדפן על ידי הקשת Ctrl+F5. כעת תראה רשימה של כל הסגנונות במערכת, בדומה לתרשים 9-14. הקישור Delete אמור להיות מנוטרל עבור רוב הסגנונות מכיוון שיש סקירות שקשורות אליהם. עם זאת, הקישור Delete זמין עבור סגנונות כמו Industrial, Reggae, Punk, וכד', שאין סקירות שקשורות אליהם. גש לדף Manage Reviews ומחק את הסקירות של סגנון אחד לפחות. כשתפתח מחדש את הדף GenresLinq.aspx, תראה רשימה אשר דומה לתרשים 9-14. ייתכן שהקישור Delete ייראה עדיין פעיל, כי הדבר תלוי בדפדפן שבו אתה משתמש. עם זאת, לא תוכל ללחוץ עליו.

• Management Home	Id	Name	SortOrder
• Manage Genres	Delete 1	Rap and Hip-Hop	14
• Manage Reviews	Delete 2	Pop	12
	Delete 3	Jazz	8
	Delete 4	Hard Rock	3
	Delete 5	Indie Rock	7
	Delete 6	Punk	1
	Delete 7	Rock	2
	Delete 8	Grunge	4
	Delete 9	Blues	9
	Delete 10	Reggae	11
	Delete 11	Industrial	10
	Delete 12	Techno	5
	Delete 13	Trance	15
	Delete 14	Speed Metal	16
	Delete 15	Trip-hop	17

תרשים 9-14

כיצד זה פועל

חלק גדול מתרגיל זה דומה לתרגיל קודם. הגדרת מטפל לאירוע `RowDataBound`, שמתעורר בכל פעם שמתווספת שורה ל-`GridView`. השונה בתרגיל זה היא הדרך שבה אתה ניגש לנתונים, והדרך שבה אתה משתמש ב-`FindControl` כדי לחפש בתוך הפקד `GridView` את `LinkButton` שגורם למחיקה. תחילה הבט בדרך הגישה לנתונים:

VB.NET

```
If e.Row.DataItem IsNot Nothing Then
    Dim myGenre As Genre = CType(e.Row.DataItem, Genre)
    ...
End If
```

C#

```
if (e.Row.DataItem != null)
{
    Genre myGenre = (Genre)e.Row.DataItem;
    ...
}
```

מכיוון ש-`LinqDataSource` פועל עם אובייקטים בעלי סוג מובהק, המאפיין `DataItem` של `e.Row` מכיל אובייקט `Genre` ממשי. לכן תוכל להמיר מאפיין זה למופע אמיתי של `Genre`, ואז לגשת למאפיינים של המופע. בדוגמה זו, הקוד ניגש לאוסף `Reviews`. לאוסף יש מאפיין `Count` שמציין כמה סקירות מכיל ה-`Genre` הנוכחי. אם המספר הוא אפס, ניתן למחוק את הסגנון בביטחון. בכל המקרים האחרים, חובה להשבית את הלחצן `Delete`. כדי להבין את הקוד שמוצא את הלחצן, עליך להתבונן תחילה בסימני העריכה שלו:

```
<asp:TemplateField ShowHeader="False">
    <ItemTemplate>
        <asp:LinkButton ID="LinkButton1" runat="server" CausesValidation="False"
            CommandName="Delete" Text="Delete"></asp:LinkButton>
    </ItemTemplate>
</asp:TemplateField>
```


המרת עמודת המחיקה ל-TemplateField מאפשרת לראות את תכולת העמודה. במקרה שלנו, העמודה מכילה LinkButton אשר CommandName שלו מוגדר ל-Delete, והדבר הופך אותו ללחצן Delete באופן אוטומטי. שים לב ש-ID של LinkButton מוגדר כ-LinkButton1. שם זה משמש בתוך מטפל האירוע כדי למצוא את הלחצן הרלוונטי.

VB.NET

```
Dim deleteButton As LinkButton = _  
    CType(e.Row.FindControl("LinkButton1"), LinkButton)  
deleteButton.Enabled = myGenre.Reviews.Count = 0
```

C#

```
LinkButton deleteButton = (LinkButton) e.Row.FindControl("LinkButton1");  
deleteButton.Enabled = myGenre.Reviews.Count == 0;
```

מכיוון שכל שורת נתונים ב-GridView מכילה לחצן Delete בשם LinkButton1, אינך יכול לגשת לפקד זה ישירות על ידי שמו. במקום זאת, עליך לחפש אותו בשורה הנוכחית, תוך שימוש ב-e.Row.FindControl. השיטה FindControl מחזירה הפניה אל הפקד LinkButton של אותה שורה ואותה תוכל להמיר למופע LinkButton מתאים.

לבסוף, על ידי התבוננות באוסף Reviews ובמאפיין Count שלו, הקוד קובע אם הלחצן יהיה מופעל או מושבת. כשאין סקירות, הביטוי myGenre.Reviews.Count = 0 יהיה True, והלחצן יופעל. אחרת, המאפיין Enabled מוגדר ל-False, ועל ידי כך מנטרל את האפשרות למחוק סגנון.

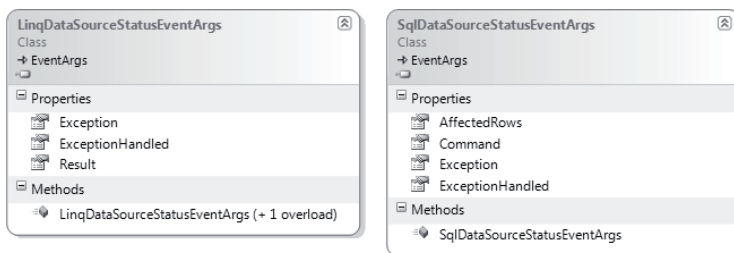
עם קוד זה תוכל בקלות למנוע שגיאות שעשויות לקרות, כשתנסה למחוק סגנון כל עוד יש סקירות שקשורות אליו. עם זאת, ייתכן שלא תמיד תוכל למנוע שגיאות מלקרות בזמן פעולות CRUD מול פקד מקור נתונים. לדוגמה, אתה עלול לנסות למחוק סגנון שבתחילה לא היו שום סקירות שמקושרות אליו. עם זאת, ממש לפני שניסית למחוק את הסגנון, מישהו הכניס סקירה חדשה עבורו. כשתנסה למחוק את הסגנון תקבל שגיאה מכיוון שהסגנון מקושר כעת לסקירה. במקרים כאלה מאפשרים פקדי מקור הנתונים לאבחן את השגיאה שקרתה ולנקוט בצעדים הדרושים, כמו מתן משוב למשתמש על אי-הצלחת פעולת CRUD שלו.

טיפול בשגיאות שקורות בפקדי מקור נתונים

בפרק 17 נרבה לעסוק בזיהוי וטיפול בשגיאות שקורות בדפי ASP.NET. הפרק הנוכחי מדגים כיצד לתפוס שגיאות שעלולות לקרות בקוד, ולטפל בהן על ידי רישומן או על ידי הודעה למשתמש. אבל, מכיוון שפקדי מקור נתונים חושפים גם הם מידע שגיאות, מעניין יהיה להתבונן בפרק זה בשגיאות גישה לנתונים.

הן הפקד LinqDataSource והן הפקד SqlDataSource נותנים מידע על שגיאות (exceptions - חריגות, בשפת .NET). שעלולות לקרות במהלך אחת מארבע פעולות CRUD. כל ארבעת האירועים

שקורים לאחר פעולת CRUD (Deleted, Updated, Inserted, Selected) של `LinqDataSource`, מקבלים מופע של מחלקה בשם `LinqDataSourceStatusEventArgs`. אותם ארבעה אירועים עבור `SqlDataSource` מקבלים את מופע המחלקה `SqlDataSourceStatusEventArgs`. תרשים 10-14 מראה את שתי המחלקות האלו ואת המאפיינים שלהן.



תרשים 10-14

שני מאפיינים חשובים משותפים לשתי מחלקות אלו: `Exception` ו-`ExceptionHandled`. הראשון מכיל את השגיאה הממשית שקרתה, או `Nothing` (VB.NET) או `null` (C#) כשהכל פועל כשורה ואין כל שגיאה. ביכולתך לאבחן את השגיאה ולנקוט בפעולה המתאימה לתיקון. לדוגמה, תוכל להודיע למשתמש שמה שהשתבש לחלוטין, או לשלוח הודעת דואר אלקטרוני למנהל האתר ולדווח על השגיאה להמשך טיפול.

במקרה שתחליט לטפל בשגיאה במטפל האירוע של פקד מקור הנתונים, חובה עליך להגדיר את המאפיין `ExceptionHandled` של האובייקט ל-`True`. כך תאות ל-`ASP.NET runtime` שאתה יודע על השגיאה, וגם טיפלת בה באופן הולם. אם תזניח את הגדרת המאפיין הזה, רכיב `runtime` יעביר את השגיאה הלאה, עד שלבסוף תוצג למשתמש.

בתרגיל הבא תלמד כיצד להשתמש במחלקה `SqlDataSourceStatusEventArgs` בדף `Genres.aspx`. את עקרונות הפעולה שתראה בתרגיל זה תוכל להחיל גם על אירועים שמעורר הפקד `LinqDataSource`.

נסה זאת טיפול בשגיאות בזמן מחיקת שורות

בתרגיל נסה זאת זה תלמד כיצד לטפל בשגיאות שקורות ב-`GridView` בזמן מחיקת שורות. תשנה את הדף `Genres` המקורי ותציג הודעת שגיאה כשהמשתמש ינסה למחוק סגנון שעדיין יש סקירות שקשורות אליו. תרגיל זה משרת בעיקר להדגמת הטיפול בשגיאות שעלולות להופיע בפקדי מקור הנתונים. מנקודת המבט של המשתמש, נטרול הקישור `Delete` כשלא נכון להשתמש בו, כפי שעשית בתרגיל קודם, תענה על הבעיה ברוב הנסיבות. עם זאת, עדיין יש אפשרות שמישהו אחר יכניס סקירה חדשה לפני שתיתן לך ההזדמנות למחוק את הסגנון.

1. פתח את הדף `Genres.aspx` מהתיקייה `Management`.
2. עבור ל-`Design View`. מארגז הכלים גרור פקד `Label` אל `GridView`. הדבר מציב את `Label`, שיכיל הודעת שגיאה, מעל `GridView`. החלף את שם ה-`Label` ל-`lblErrorMessage` ומחק את המאפיין `Text` שלו (בטבלת המאפיינים של הפקד, לחץ לחיצה ימנית על המאפיין

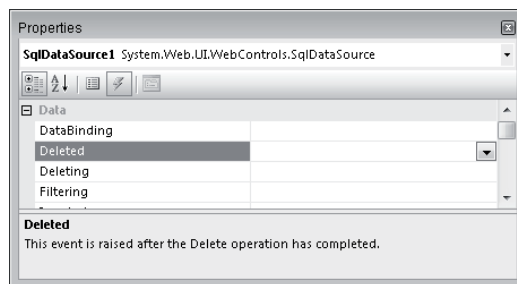
Text ובחר Reset. פעולה זו מסירה את המאפיין Text כולו ואת הערך שלו מסימני העריכה של הפקד). הגדר את ה-CssClass שלו ל-ErrorMessage. לבסוף, הגדר את המאפיין EnableViewState שלו ל-False, כדי להבטיח שהתווית אינה שומרת את הטקסט לאחר postback. כעת תקבל את הקוד הבא:

```
<asp:Label ID="lblErrorMessage" runat="server" CssClass="ErrorMessage"
    EnableViewState="False"></asp:Label>
<asp:GridView ID="GridView1" runat="server" AllowPaging="True"
```

3. פתח את הקובץ Management.css מתיקיית ערכת הנושא Management, והוסף לו את כלל הסגנון הבא:

```
.ErrorMessage
{
    color: Red;
    font-weight: bold;
}
```

4. עבור חזרה ל-Genres.aspx, ודא שהדף נמצא ב- Design View, ואז לחץ על הפקד SqlDataSource פעם אחת כדי לבחור בו. פתח את טבלת המאפיינים שלו, עבור לקטגוריית Events וקבע מטפל אירוע עבור האירוע Delete על ידי לחיצה כפולה על שם האירוע, כפי שנראה בתרשים 14-11.



תרשים 14-11

5. בראש הקוד ברקע הוסף את מרחב השמות הבא:

VB.NET

```
Imports System.Data.SqlClient
```

C#

```
using System.Data.SqlClient;
```

6. כתוב את הקוד הבא בתוך מטפל האירוע שהוסיפה VWD:

VB.NET

```
Protected Sub SqlDataSource1_Deleted(ByVal sender As Object, _
    ByVal e As System.Web.UI.WebControls.SqlDataSourceStatusEventArgs) _
    Handles SqlDataSource1.Deleted
```

```

If e.Exception IsNot Nothing AndAlso _
    TypeOf (e.Exception) Is SqlException Then
    Dim myException As SqlException = CType(e.Exception, SqlException)
    If myException.Number = 547 Then
        lblErrorMessage.Text = "Sorry, you can't delete this genre because " & _
            "it has associated reviews that you need to delete first."
        e.ExceptionHandled = True
    End If
End If
End Sub

```

End Sub

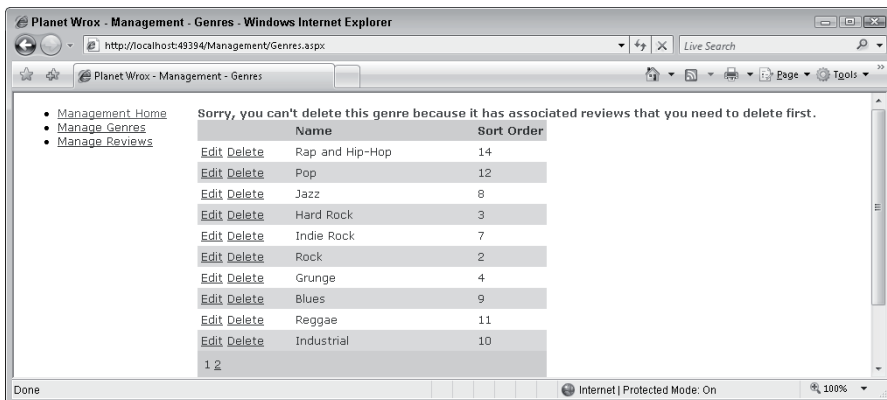
C#

```

protected void SqlDataSource1_Deleted(object sender, SqlDataSourceStatusEventArgs e)
{
    if (e.Exception != null && e.Exception is SqlException)
    {
        SqlException myException = (SqlException)e.Exception;
        if (myException.Number == 547)
        {
            lblErrorMessage.Text = @"Sorry, you can't delete this genre because
                                   it has associated reviews that you need to delete first.";
            e.ExceptionHandled = true;
        }
    }
}

```

7. שמור את כל השינויים ופתח את Genres.aspx בדפדפן על ידי הקשת Ctrl+F5. נסה למחוק סגנון שאתה יודע שיש סקירות שקשורות אליו, כמו Grunge למשל. במקום למחוק את הסגנון, דף ASPX יציג עכשיו הודעת שגיאה מעל ל-GridView. ראה תרשים 14-12.
8. בתפריט Management, לחץ על הפריט Manage Reviews ומהרשימה הנפתחת בחר בסגנון Grunge. מחק את הסקירות עבור סגנון זה, או ערוך והקצה אותן לסגנון אחר.



תרשים 14-12

9. חזור אל Genres.aspx ונסה שוב למחוק את הסגנון Grunge. הפעם הסגנון יימחק בהצלחה מבסיס הנתונים.

10. כדי לראות את השגיאה שהיית מקבל ללא טיפול זה בשגיאה, הפוך להערה את השורה בקוד ברקע שמגדירה את ExceptionHandled ל-True (עשה זאת על ידי שימוש בגרש יחיד (') ב-VB.NET או בשני לוכסנים רגילים (//) ב-C#). בנוסף, הסר את UpdatePanel ואת האלמנט ContentTemplate שלו מ-Markup View של הדף Genres.aspx. כש-UpdatePanel נוכח בדף, הדפדפן עלול לבלוע את השגיאה. ללא UpdatePanel קל הרבה יותר לראות מה קורה. שמור את השינויים, פתח שוב את הדף בדפדפן ונסה למחוק סגנון עם סקירות. כעת תקבל הודעת שגיאה מפורטת של ASP.NET.

כיצד זה פועל

כשאתה לוחץ על הקישור Delete ב-GridView שמוצג בתרשים 12-14, ה-GridView מעורר את הפקודה Delete בפקד SqlDataReader הקשור אליו. פקד זה, כפי שראית בפרקים קודמים, מנסה לשלוח משפט DELETE לבסיס הנתונים. כעת, בסיס הנתונים מנסה למחוק את הסגנון המבוקש, אבל מוצא שלא ניתן לבצע זאת מכיוון שיש סקירות שקשורות לסגנון זה. תוצאת הדבר היא הודעת השגיאה 'Foreign key constraint error', כי לא ניתן למחוק את הסגנון מכיוון ש-ID שלו משמש כמפתח זר בטבלה אחרת.

ההודעה Foreign key constraint error מוחזרת מבסיס הנתונים ומופיעה בסופו של דבר במאפיין Exception של הארגומנט e של המטפל SqlDataReader1_Deleted. אחר כך הקוד בודק אם הייתה שגיאה (e.Exception אינו Nothing / null) ואת סוג השגיאה, כדי למצוא אם מדובר ב-SqlException.

VB.NET

```
If e.Exception IsNot Nothing AndAlso _  
    TypeOf (e.Exception) Is SQLException Then  
    ...  
End If
```

C#

```
if (e.Exception != null && e.Exception is SQLException)  
{  
    ...  
}
```

כשעובדים עם בסיס נתונים מסוג שרת SQL, שזה המקרה בדוגמה של Planet Wrox, השגיאות שמעורר בסיס הנתונים הן מסוג SQLException ממרחב השמות System.Data.SqlClient שייבאת בתרגיל זה. הדבר מאפשר הפרדה ברורה בין שגיאות בסיס הנתונים לבין שגיאות אחרות.

כשרשת SQL מעורר שגיאה, הוא גם מעביר מספר שגיאה שמאוחסן במאפיין Number של השגיאה. כדי שתוכל לגשת למספר זה, עליך לצרף את השגיאה ל-SqlException אמיתי. הדבר נעשה על ידי הקוד הבא:

VB.NET

```
Dim myException As SQLException = CType(e.Exception, SQLException)
```

C#

```
SqlException myException = (SqlException)e.Exception;
```

לבסוף, הקוד בודק את המאפיין Number. אם המספר הוא 547, פירוש הדבר ששרת SQL עורר שגיאה: Foreign key constraint error כדי לציין שלא ניתן למחוק סגנון מכיוון שיש עדיין סקירות שקשורות אליו. במקרה זה מוגדר המאפיין Text של הפקד Label עם הודעת השגיאה המתאימה, ולבסוף הקוד מגדיר את e.ExceptionHandled ל-True. הדבר אומר ל-runtime- שהשגיאה טופלה. כתוצאה מכך המשתמש יקבל הודעת שגיאה נחמדה וידידותית בראש ה-GridView, במקום דף שגיאה עמוס ולא קריא. שים לב שעבור כל סוגי השגיאות האחרים המשתמש עדיין יקבל את מסך ברירת המחדל של הודעת השגיאה של ASP.NET, שנקרא גם Yellow Screen of Death. בפרק 17 תלמד כמה טכניקות לרישום השגיאה במקום מרכזי, ולהצגתה למשתמש בדף שגיאה ידידותי וקריא.

המספר 547 נראה כאילו נבחר באופן שרירותי, אבל זה המספר ששרת SQL מחזיר עבור השגיאה מסוג Foreign key constraint exception. בפרק 17 שעוסק בניפוי, תלמד כמה תחבולות שיאפשרו לך להתבונן בשגיאות שמתעוררות ולאבחן את המאפיין Number עבור סוגי שגיאות שונים.

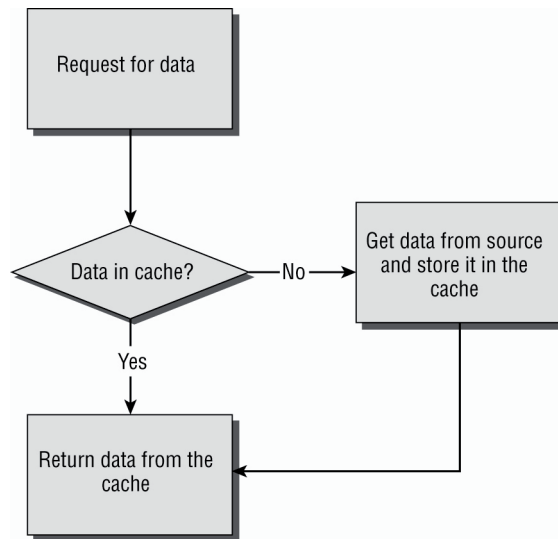
בכל דוגמאות בסיס הנתונים שראית עד כה, הקוד ניגש לבסיס הנתונים עבור כל בקשה ובקשה. בכל פעם שיש צורך להציג נתונים מסוימים הם מאוחזרים מחדש מבסיס הנתונים. בוודאי ברור לך שהדבר כרוך בבזבוז זמן ומשאבים וגם בעומס CPU. הבזבוז בולט במיוחד במקרים שבהם הנתונים לא השתנו מאז הפעם האחרונה שניגשו אליהם.

בסעיף האחרון של פרק זה תכיר טכניקה בשם מיטמון (caching) שיכולה לשפר באופן ניכר את זמן התגובה והביצועים של היישום.

מיטמון (Caching)

מיטמון היא אחת הדרכים הטובות ביותר, ולעתים קרובות הקלות ביותר, לשיפור ביצועים של יישום. מיטמון הוא אחסון עותק של הנתונים במקום שניתן לגשת אליו במהירות רבה. העיקרון של המיטמון הוא ששליפת הנתונים מהמיטמון (cache) חייבת להיות מהירה יותר משליפתם ממקור הנתונים המקורי. לכן, ברוב פתרונות המיטמון מאחסנים נתונים בזיכרון, שהיא הדרך המהירה ביותר לקבלת הנתונים. המטמון של .NET אינו שונה, ומאפשר לאחסן נתונים שניגשים אליהם בתדירות גבוהה, במקום מיוחד בזיכרון המחשב.

תרשים הזרימה של עיקרון המיטמון מוצג בתרשים 13-14.



תרשים 13-14

היישום מבקש נתונים כלשהם, כמו לדוגמה, רשימת סגנונות מבסיס הנתונים. במקום לגשת מייד לבסיס הנתונים, המטמון נבדק תחילה כדי לראות אם הוא מכיל את הנתונים המבוקשים. אם כן, הנתונים מאוחזרים מהמטמון. אם הנתונים אינם מאוחסנים עדיין במטמון, הם מאוחזרים ממקור הנתונים עצמו, כמו למשל מבסיס נתונים SQL Server. עותק של הנתונים מאוחסן במטמון כדי שהנתונים יהיו זמינים לאחזור בפעם הבאה בו הם יידרשו. לבסוף, הנתונים מועברים אל הקוד המבקש.

באופן כללי, מיטמון הוא פתרון מצוין לשיפור היישום, אבל יש לו כמה חסרונות שעליך להיות מודע להם. בסעיף הבא ניתן הסבר על כמה מלכודות נפוצות שעלולים ליפול לתוכן בזמן שימוש במיטמון. לאחר מכן נראה את מנגנוני המיטמון השונים שעומדים לרשותך ביישום אינטרנט בסביבת ASP.NET.

מלכודות נפוצות במיטמון נתונים

כשעובדים עם נתונים מוטמנים יש סיכון לקבלת נתונים ישנים לא עדכניים, או כאלה "שהעלו עובש". לדוגמה, כשמוטמנות תוצאות שאילתה על כל הסגנונות בבסיס הנתונים, ומשתמשים בהם במקום אחזור נתונים ישיר מבסיס הנתונים, עלולים להחמיץ סגנון חדש שמשתמש אחר הכניס בינתיים לטבלה.

סוגיה אחרת עם המטמון היא בכך, שאינך יכול לסמוך על כך שפריט אכן יהיה נוכח בו. כדי למזער את צריכת הזיכרון של היישום הפעיל, מנגנון המיטמון של ASP.NET מסיר מהמטמון מזמן לזמן ובאופן אוטומטי פריטים ישנים, או כאלה שמשתמשים בהם לעתים רחוקות. לכן אסור להסתמך על כך שפריטים יהיו במטמון, אפילו אם הכנסת אותם לשם בזמן אתחול היישום.

בסעיף הבא תלמד כיצד להימנע משתי תקלות אלו.

הימנעות משימוש בנתונים שהתיישנו

כדי להימנע משימוש בנתונים מיושנים שאינם מסונכרנים עוד עם הנתונים המקוריים שבבסיס הנתונים, דרושה דרך של ביטול תוקף הנתונים במטמון (invalidate the cache). ביטול תוקף הנתונים במטמון גורם להסרת פריט מהמטמון, כדי שבפעם הבאה שיידרש, צריך יהיה לאחזר אותו ממקור הנתונים. כמה דרכים עומדות לרשותך כדי לבטל את תוקף נתוני המטמון. ראשית, ביכולתך לקבוע משך שהייה קצר במטמון. לדוגמה, באפשרותך להטמין את רשימת הסגנונות מבסיס הנתונים למשך פרק זמן, 10 דקות למשל. אם משתמש אחר מכניס במשך אותן 10 דקות סגנון חדש, הוא לא יופיע בדפי האינטרנט. עם זאת, לאחר 10 דקות, תוסר הרשימה מהמטמון, ובפעם הבאה שהיא תבוקש, היא תיוצר מחדש עם נתונים חדשים. מאוחר יותר תלמד כיצד להשתמש במיטמון מבוסס-זמן.

אפשרות אחרת לביטול תוקף המטמון היא שימוש בתלות המטמון (cache dependency). לפי שיטה זו אתה יוצר קשר בין פריט המטמון לבין מקור הנתונים המקורי. כשמקור הנתונים הבסיסי משתנה, מבוטל תוקף פריט המטמון כדי שצריך יהיה ליצור אותו מחדש בפעם הבאה שיידרש על ידי היישום. לדוגמה, ביכולתך להטמין את התכנים של טקסט גדול או קובץ XML במקום לקרוא את הקובץ בכל פעם שאתה זקוק לו, תוכל להכניסו למטמון. אחר כך חבר אותו למופע של CacheDependency שמנטר את קובץ המקור. ברגע שהקובץ משתנה, הפריט מוסר מהמטמון, וייוצר מחדש עם הבקשה הבאה. לקראת סוף הפרק תראה כיצד להשתמש ב-CacheDependency (מיטמון מבוסס תלות).

גם כשמשתמשים בבסיס נתונים מסוג SQL Server אפשר להשתמש בתלות-מטמון. פירוש הדבר שברגע שהנתונים שמהווים חלק מהשאלתה המוטמנת משתנים, מבוטל תוקף הפריט המוטמן. מיטמון בסיסי נתונים וביטול תוקפם הוא נושא מתקדם ורחב ולא נדון בספר זה.

אל תסמוך על כך שהנתונים יהיו שם

כפי שלמדת זה עתה, ייתכן שפריטים יוסרו מהמטמון בשלבים שונים של מחזור החיים של היישום. הם יכולים להיות מוסרים על ידי המטמון עצמו מכיוון ש- ASP.NET runtime קבע שלא משתמשים בפריט בתדירות מספקת, ולפיכך הוא תופס מקום יקר-ערך שלא לצורך. המטמון כולו מנוקה כאשר יישום אינטרנט או שרת אינטרנט מאתחלים. אבל פריטים יכולים להיות מוסרים גם בגלל התלויות שלהם. לכן, לעולם אל תסמוך על כך שהפריט יהיה במטמון, גם אם הכנסת אותו לשם בעצמך מוקדם יותר במחזור החיים של היישום.

מאוחר יותר בפרק תלמד כיצד להשתמש בממשק יישום נתונים - Cache API (Application Programming Interface), שהינה הדרך שבה אתה פועל הדדית עם הפונקציונליות של המטמון כדי להשתמש במטמון בתכנות. לפניך דוגמה פשוטה שמאחזרת מופע של Review מהמטמון. הקוד אינו מניח שהפריט נמצא שם, אלא בודק תחילה את המטמון:

VB.NET

```
Dim reviewId As Integer = Convert.ToInt32(Request.QueryString.Get("ReviewId"))  
Dim myReview As Review
```



```

If Cache("MyReview" + reviewId.ToString()) Is Nothing Then
    Using db As New PlanetWroxDataContext()
        myReview = (From r In db.Reviews _
                     Where r.Id = reviewId _
                     Select r).Single()
    End Using

    Cache("MyReview" + reviewId.ToString()) = myReview
End If

myReview = CType(Cache("MyReview" + reviewId.ToString()), Review)

```

C#

```

int reviewId = Convert.ToInt32(Request.QueryString.Get("ReviewId"));
Review myReview;

if (Cache["MyReview" + reviewId.ToString()] == null)
{
    using (PlanetWroxDataContext db = new PlanetWroxDataContext())
    {
        myReview = (from r in db.Reviews
                     where r.Id == reviewId
                     select r).Single();
    }
    Cache["MyReview" + reviewId.ToString()] = myReview;
}

myReview = (Review) Cache["MyReview" + reviewId.ToString()];

```

בקטע קוד זה מחפשים במטמון את הפריט הרצוי לפי המפתח שלו, שהוא שילוב של המילה הסטטית MyReview וה-ID של ה-Review שמחפשים. כשהמטמון ריק (החיפוש מחזיר Nothing או null), יוצרים את הפריט תוך שימוש בשאילתת LINQ ואז מאחסנים אותו במטמון.

זו דוגמה טובה לקוד שאינו סומך על כך שהפריט יימצא במטמון. במקום זאת הקוד בודק את המטמון בזהירות, ואם הפריט אינו קיים עדיין במטמון הקוד יוצר אותו תחילה. בהמשך הפרק תלמד יותר על אפשרויות הגישה למטמון על ידי תכנות.

שימוש ב-API של המטמון אינו הדרך היחידה להשתמש במטמון. בסעיף הבא תלמד על דרכים שונות למיטמון נתונים על ידי ASP.NET.

דרכים שונות להטמנת נתונים ביישומי אינטרנט בסביבת ASP.NET

אפשר להשתמש באסטרטגיות מיטמון שונות ביישומי ASP.NET, וביניהן מיטמון הפלט, מיטמון בעזרת פקדי מקור הנתונים המובנים, ומיטמון על ידי תכנות. כל שלוש האפשרויות נידונות להלן.

מיטמון הפלט (Output Caching)

בשיטת מיטמון הפלט, מטמינים את התוצאה הסופית של דף מרונדר (html). כלומר, בפעם הראשונה שבה הדף מבוקש, התוצאה הסופית שלו מתווספת למטמון. הבקשות הבאות לאותו דף גורמות לשליחת אותו קוד HTML. עבור דפים סטטיים יחסית, כמו הדף About באתר Planet Wrox, זה בסדר, אבל עבור דפים דינמיים אין זה פתרון אידיאלי. לדוגמה, נניח שיש דף דינמי שמציג את הפרטים של Review על פי מחרוזת השאילתה שמועברת לדף. בפעם הראשונה שתבקש את הדף אתה עשוי לדפדף לכתובת כמו זו:

`http://localhost:12345/Reviews/ViewDetails.aspx?Id=23`

ASP.NET יוצרת דף שמציג את סקירה 23 ואז מטמינה את כל הפלט של הדף. אבל מה יקרה כשתבקש את הדף הזה?

`http://localhost:12345/Reviews/ViewDetails.aspx?Id=58`

במקום לקבל את הסקירה בעלת ID שערכו 58 תסיים עם הסקירה הקודמת בעלת ID שערכו 23. כדי להתגבר על מצב זה, ASP.NET מאפשרת להטמין גרסאות מסוימות של דף. לדוגמה, ביכולתך להורות ל-Planet Wrox להטמין עותק של הדף עבור כל שדה מחרוזת שאילתה ייחודי שהוא מאחזר. עבור דף שמציג את הפרטים של סקירה מסוימת זה מושלם. ASP.NET שומרת עותק מוטמן עבור כל סקירה ייחודית. כלומר, פנייה לבסיס הנתונים מתרחשת רק בפעם הראשונה שמבקשים סקירה מסוימת; הבקשות הבאות לאותה סקירה תקבלנה ישירות מהמטמון. בתרגיל הבא תלמד כיצד ליצור דף פרטים שמשתמש באסטרטגיית מיטמון זו כדי למנוע מאותה שאילתת LINQ שמקבלת סקירה מלהיות מבוצעת שוב ושוב.

נסה זאת מיטמון הפלט

1. הוסף דף חדש בשם `ViewDetails.aspx` לתיקייה `Reviews` של יישום האינטרנט. ודא שהוא מבוסס על התבנית המותאמת אישית שלך. אין צורך לקבוע לו כותרת, מכיוון שהיא תיקבע בתכנות.
2. ב- `Markup View`, הוסף שלושה פקדי `Label` לשומר מקום התוכן `cpMainContent`, וקרא להם בשמות אלה:

- ⊙ `lblTitle`
- ⊙ `lblSummary`
- ⊙ `lblBody`

מחק את המאפיין `Text` והערך שלו בכל שלושת הפקדים.

3. עטוף את הפקד `lblTitle` באלמנט `<h1>`. את שני הפקדים האחרים עטוף באלמנט `<div>` והגדר את התכונה `class` שלהם ל-`Summary-1-Body`, בהתאמה. כעת יהיה לפניך הקוד הבא:

```
<h1><asp:Label ID="lblTitle" runat="server"></asp:Label></h1>
```

```
<div class="Summary"><asp:Label ID="lblSummary" runat="server"></asp:Label></div>
<div class="Body"><asp:Label ID="lblBody" runat="server"></asp:Label></div>
```

4. עבור ל- Design View ולחץ לחיצה כפולה על האזור המיועד לקריאה-בלבר של הדף, כדי להגדיר מטפל לאירוע Page_Load.

5. הוסף את הקוד הבא למטפל האירוע Page_Load שנוצר עבורך:

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Me.Load
```

```
Dim reviewId As Integer = Convert.ToInt32(Request.QueryString.Get("ReviewId"))
Dim myReview As Review
```

```
Using db As New PlanetWroxDataContext()
```

```
    myReview = (From r In db.Reviews _
                Where r.Id = reviewId _
                Select r).Single()
```

```
    lblTitle.Text = myReview.Title
```

```
    lblSummary.Text = myReview.Summary
```

```
    lblBody.Text = myReview.Body
```

```
    ' Set the page title programmatically
```

```
    Me.Title = myReview.Title
```

```
End Using
```

```
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
```

```
{
```

```
    int reviewId = Convert.ToInt32(Request.QueryString.Get("ReviewId"));
```

```
    Review myReview;
```

```
    using (PlanetWroxDataContext db = new PlanetWroxDataContext())
```

```
    {
```

```
        myReview = (from r in db.Reviews
                     where r.Id == reviewId
                     select r).Single();
```

```
        lblTitle.Text = myReview.Title;
```

```
        lblSummary.Text = myReview.Summary;
```

```
        lblBody.Text = myReview.Body;
```

```
        // Set the page title programmatically
```

```
        this.Title = myReview.Title;
```

```
    }
```

```
}
```

6. פתח את הדף All.aspx מהתיקייה Reviews, ומחק את GridView שיצרת בפרק הקודם. החלף אותו בפקד Repeater פשוט שמכיל Hyperlink יחיד:

```
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" Runat="Server">
  <asp:Repeater ID="Repeater1" runat="server">
    <ItemTemplate>
      <asp:HyperLink ID="HyperLink1" runat="server"
        NavigateUrl='<## "ViewDetails.aspx?ReviewId=" + Eval("Id").ToString() %>'
        Text='<## Eval("Title") %>'></asp:HyperLink>
      <br />
    </ItemTemplate>
  </asp:Repeater>
</asp:Content>
```

7. עבור אל הקוד ברקע של הדף, והחלף את שתי הקריאות האחרונות שהשתמשו ב-GridView באופן שישתמשו במקומו כעת בפקד Repeater:

VB.NET

```
Repeater1.DataSource = allReviews
Repeater1.DataBind()
```

C#

```
Repeater1.DataSource = allReviews;
Repeater1.DataBind();
```

8. חזור לדף ViewDetails.aspx, ובראשו, מייד תחת ההנחיה @ Page, הוסף את הקוד הבא:

```
<%@ Page Language="VB" MasterPageFile="~/MasterPages/MasterPage.master"
  AutoEventWireup="false" CodeFile="ViewDetails.aspx.vb"
  Inherits="Reviews_ViewDetails" %>
<%@ OutputCache Duration="300" VaryByParam="None" %>
```

9. שמור את כל השינויים ובקש בדפדפן את הדף All.aspx מהתיקייה Reviews. לחץ על כותרת של סקירה כדי שתועבר אל הדף ViewDetails.aspx, עם ה-ID של הסקירה שביקשת במחרוזת השאילתה.

10. לחץ על הלחצן Back ואחר כך לחץ על סקירה אחרת. שים לב שהטקסט של הסקירה אינו מתעדכן ועדיין מציג את הסקירה הקודמת. מהרשימה הנפתחת של ערכת הנושא, בחר נושא חדש. שים לב שהדף ממשיך להציג את הנושא הקודם.

11. עבור חזרה ל-VWD ושנה את ההנחיה OutputCache:

```
<%@ OutputCache Duration="300" VaryByParam="ReviewId" %>
```

12. פתח בדפדפן שוב את All.aspx ולחץ על כותרת של סקירה. מעתה והלאה, לחיצה על סקירה כלשהי תציג אותה כראוי. עם זאת, מתחת לפני השטח עדיין מוטמן הדף כולו, כדי למזער את הצורך בגישה לבסיס הנתונים לאחזור הסקירה. אם אתה עדיין רואה את הדף הקודם, הקש Ctrl+F5 או Ctrl+R בדפדפן כדי לבקש מהשרת עותק רענן של הדף.

אם תעבור לערכת נושא אחרת, תוכל להבחין שהדף עדיין מוצג תוך שימוש בערכת הנושא הקודמת.

כיצד זה פועל

רוב הקוד בתרגיל זה אמור להיות מוכר לך כעת. לדף ViewDetails.aspx הוספת כמה תוויות שתפקידן להכיל מאפיינים רלוונטיים של Review כמו Title, Summary ו-Body. עטפת תוויות אלו באלמנטים מסוג <div> של צד-לקוח עם התכונה class, כדי שתוכל להשפיע על עיצובם מקובץ CSS, אם תרצה (פעולה זו לא הוצגה בתרגיל).

הקוד בקוד ברקע של הקובץ מאחזר מופע יחיד של Review בהתבסס על מחרוזת השאילתה:

VB.NET

```
Using db As New PlanetWroxDataContext()
    myReview = (From r In db.Reviews _
                Where r.Id = reviewId _
                Select r).Single()
    ...
End Using
```

C#

```
using (PlanetWroxDataContext db = new PlanetWroxDataContext())
{
    myReview = (from r in db.Reviews
                where r.Id == reviewId
                select r).Single();
    ...
}
```

שים לב שהשאילתה משתמשת באופרטור Single() כדי להגביל את השאילתה למופע יחיד של Review. הסקירה שמוחזרת מבסיס הנתונים משמשת למלא את פקדי Label שבדף, ואת ה-Title של הדף.

VB.NET

```
lblTitle.Text = myReview.Title
lblSummary.Text = myReview.Summary
lblBody.Text = myReview.Body
```

```
Me.Title = myReview.Title
```

C#

```
lblTitle.Text = myReview.Title;
lblSummary.Text = myReview.Summary;
lblBody.Text = myReview.Body;
```

```
this.Title = myReview.Title;
```

בנסיבות רגילות, קוד זה יתעורר עבור כל בקשה לדף. גם אם ביקשת את אותה סקירה שוב ושוב, הרי שכל בקשה תגרור גישה לבסיס הנתונים. כדי למזער את התקורה הקשורה בגישה חוזרת לבסיס הנתונים, הדף משתמש באסטרטגיה של מיטמון הפלט, שמטמינה את התוצר (דף Html) למשך 5 דקות (300 שניות). בנוסף, עבור כל אובייקט Review נפרד, נוצר עותק של הדף על ידי שימוש בהגדרה הבאה של VaryByParam:

```
<%@ OutputCache Duration="300" VaryByParam="ReviewId" %>
```

התכונה VaryByParam מקבלת רשימת שדות של מחרוזות שאילתה, אשר מופרדים זה מזה בנקודה ופסיק, כי רצונך ליצור עבורם עותקים נפרדים במטמון. הדבר נותן גמישות רבה במונחי הנתונים שיוטמנו. הדבר גם מבטיח ש- ASP.NET runtime יטמין עותקים רק עבור דפים ייחודיים, ועל ידי כך ישפר את הביצועים של סקירות שמבוקשות בתכיפות.

אחת הבעיות עם מיטמון הפלט היא שלעיתים קרובות זהו תרחיש של "הכל או כלום". כפי שראית בתרגיל הקודם, הדף כולו מוטמן עבור המשתמש הראשון שביקש אותו. דף זה מוצג אחר כך לכל משתמש מבלי להתייחס לערכת הנושא שהמשתמש בחר. למרות שניתן להשפיע על התנהגות זו בתכנות, עומד לרשותך פתרון קל יותר כדי להורות לפקד מקור הנתונים להטמין את הנתונים עבורך.

מיטמון עם פקדי מקור נתונים

התועלת הגדולה ביותר במיטמון על ידי פקדי מקור נתונים היא בכך, שהם מטמינים רק נתונים דינמיים מונעי בסיס נתונים (database-driven), ולא את הדף כולו. הדבר מאפשר לשמור על דינמיות בחלקים אחרים של הדף, כמו מודול כרוז או ברכה מותאמת אישית כדי לברך את המשתמש. כל פקדי מקור נתונים, פרט ל-LinqDataSource, מעוצבים באופן שיתמכו במיטמון.

המיטמון עם פקדי מקור-נתונים קל מאוד. עליך רק להגדיר את המאפיין EnableCaching ולציין CacheDuration. קטע הקוד הבא מראה פקד SqlDataReader שמטמין את הנתונים שלו למשך 10 דקות (600 שניות):

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server" CacheDuration="600"
    EnableCaching="True"></asp:SqlDataSource>
```

הדבר שמגניב עם פקדי מקור נתונים הוא שהם מספיק "נבונים" כדי לראות אם מתבצעים עדכונים לנתונים היסודיים. אם הגדרת את הפקד SqlDataReader כדי שיתמין נתונים עבור SelectCommand למשך 20 דקות, אבל שינית את הנתונים (בתוך אותו פרק זמן) על ידי InsertCommand, UpdateCommand או DeleteCommand, הרי שתוקף המטמון יבוטל באופן אוטומטי. הדבר פועל רק כשמבצעים פקודות הכנסה (insert), עדכון או מחיקה כנגד אותו SqlDataReader בדיוק. לעומת זאת, אם יהיה דף אחד שמציג ומטמין רשימה של סקירות (כמו All.aspx לדוגמה), ודף אחר שמשמש להכנסת סקירה חדשה (כמו AddEditReview.aspx בתיקייה Management), הדבר לא יפעל. לאחר הכנסת הסקירה החדשה בחלק הניהולי של האתר, היא לא תופיע ב-All.aspx עד לאחר תפוגת המטמון.

מיטמון ללא קוד על ידי פקדי מקור נתונים שימושי במצבים רבים. עם זאת, לא ניתן להשתמש בפקדי מקור נתונים בכל מצב. מה אם תרצה להטמין את התוצאות של נתונים שקיבלת ממקור שונה לחלוטין? מה אם תרצה להטמין את התוכן של טקסט או קובץ XML שצריך לגשת אליהם תכופות? עבור מקרים אלה נותנת ASP.NET אפשרות גישה אל המטמון בדרך של תכנות.

מיטמון על ידי תכנות

עם מיטמון על ידי תכנות ביכולתך לאחסן פריטים במטמון באמצעות קוד של VB.NET או של C#. תוכל גם לגשת אל הפריטים האלה מאוחר יותר. כדי לאחסן פריטים תשתמש בשיטות Add או Insert, או שתיצור אינדקס עבור האוסף Cache באופן ישיר. השיטה Add היא רבת-עוצמה (ומורכבת), ומאפשרת לציין מספר רב של אפשרויות שקובעות כמה זמן ישהה הפריט במטמון, איזו עדיפות תהיה לו בהשוואה לפריטים אחרים, ובאיזה תנאים ועל פי איזה גורמים תהיה חובה להסיר את הפריט מהמטמון.

השיטה Insert פשוטה הרבה יותר. היא משתמשת בטכניקה הקרויה העמסת-יתר (overloading) כדי לציין את הפריט שמוטמן, ולקשור אותו עם מפתח ייחודי. העמסת-יתר אחרת מאפשרת להגדיר תלויות. בתרגיל הבא תראה כיצד להשתמש בה.

השיטה Remove משמשת להסרת פריטים מהמטמון. שיטה זו מקבלת את המפתח של הפריט המוטמן, אשר מוגדר כשמוטמים את הפריט לאוסף על ידי השיטה Add או Insert.

כדי לגשת לפריט במטמון, עומדות לרשותך כמה אפשרויות. הדרך הישירה היא לגשת ישירות לאוסף Cache:

VB.NET

```
myReview = CType(Cache("MyKey"), Review)
```

C#

```
myReview = (Review) Cache["MyKey"];
```

האינדקס של האוסף Cache מבוסס על המפתח MyKey.

בנוסף, באפשרותך להשתמש בשיטה Get שאף היא מצפה לקבל מפתח:

VB.NET

```
myReview = CType(Cache.Get("MyKey"), Review)
```

C#

```
myReview = (Review) Cache.Get("MyKey");
```

מכיוון ש-Get היא שיטה, דרושים סוגריים סביב ערך מפתח המטמון גם בדוגמה של C#. הדבר גורם לדמיון רב בין שתי הדוגמאות, ובסיום מוחזר בשני המקרים אותו פריט בדיוק.

לבסוף, תוכל לגשת לפריטים במטמון על ידי שימוש במאפיין Item שגם הוא צריך לקבל את המפתח של הפריט המוטמן.

שלוש דרכי הגישה לפריטים במטמון מחזירות תמיד object כללי. כלומר, אם אתה יודע מהו הסוג שמוחזר אליך מהמטמון, אתה צריך תמיד להמיר אותו אל הסוג המתאים לפני השימוש במאפייניו. שתי הדוגמאות הקודמות הראו כיצד פריט מהמטמון מצורף תחילה לאובייקט Review בעל סוג מובהק.

כדי שתלמד כיצד להשתמש במטמון בתכנות, תראה בתרגיל הבא כיצד להכניס למטמון תוכן של קובץ כדי להימנע מהצורך לקרוא את הקובץ בכל פעם שזקוקים לו.

נסה זאת | שימוש ב-API של המטמון

בפרק 9 יצרת את הטופס Contact שמשמש בתוכן של קובץ כגוף התבנית עבור הודעת דואר אלקטרוני. כדי להימנע מקריאת הקובץ בכל פעם שנשלחת הודעת דואר, מומלץ להטמין את תוכנו בפעם הראשונה שהוא נקרא. הקריאות הבאות לשיטות שליחת הדואר משתמשות בגוף הקובץ מהמטמון במקום לקרוא את הקובץ מהדיסק. כדי למנוע התיישנות של העותק במטמון, עליך לציין בעת הכנסת הפריט למטמון את התלות בקובץ המקורי. כלומר, אם הקובץ ContactForm.txt שבדיסק ישתנה, העותק שלו יוסר מהמטמון באופן אוטומטי, ובפעם הבאה שיודקו לו, הוא יילקח שוב מהדיסק.

1. מהתיקייה Controls, פתח את פקד המשתמש ContactForm.ascx.
2. פתח את הקוד ברקע של הפקד, והוסף את משפטי Imports או Using הבאים בראש הקובץ:

VB.NET

```
Imports System.Web.Caching
```

C#

```
using System.Web.Caching;
```

3. אתר את מטפל האירוע btnSend_Click, שמתעורר כשמשתמש מנסה לשלוח את הטופס, ובראש הקוד של המטפל שנה את קטע הקוד שקורא את הקובץ. לבסוף, תקבל את קטע הקוד הזה:

VB.NET

```
If Page.IsValid Then
```

```
    Dim fileName As String = Server.MapPath("~/App_Data/ContactForm.txt")
    Dim mailBody As String = String.Empty
    If Cache("ContactFormMailBody") Is Nothing Then
        mailBody = System.IO.File.ReadAllText(fileName)
        Cache.Insert("ContactFormMailBody", mailBody, New CacheDependency(fileName))
    Else
        mailBody = Cache("ContactFormMailBody").ToString() & vbCrLf & "(File from the cache)"
    End If
    mailBody = mailBody.Replace("##Name##", txtName.Text)
```


C#

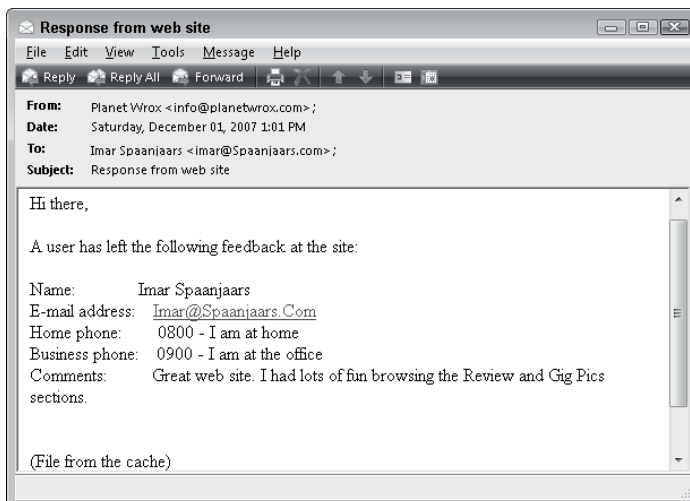
```
if (Page.IsValid)
{
    string fileName = Server.MapPath("~/App_Data/ContactForm.txt");
    string mailBody = string.Empty;
    if (Cache["ContactFormMailBody"] == null)
    {
        mailBody = System.IO.File.ReadAllText(fileName);
        Cache.Insert("ContactFormMailBody", mailBody, new CacheDependency(fileName));
    }
    else
    {
        mailBody = Cache["ContactFormMailBody"].ToString() + "\r\n(File from the cache)";
    }

    mailBody = mailBody.Replace("##Name##", txtName.Text);
}
```

4. שמור את כל השינויים ופתח בדפדפן את הדרך Default.aspx משורש האתר. מהתפריט About בחר Contact Us. מלא את הטופס ולחץ על הלחצן Send. לאחר זמן מה תקבל הודעת דואר בתיבת הדואר של החשבון שמשמש לשליחת דואר אלקטרוני מהקוד ברקע של פקד המשתמש.

5. בחר שוב Contact Us מתפריט About, מלא את הטופס ולחץ על Send פעם נוספת. לאחר זמן מה תקבל הודעה נוספת (ראה תרשים 14-14). עם זאת, הפעם גוף הודעת הדואר מתווג עם הטקסט (File from the cache) כדי לציין שהקובץ אוזרז מהמטמון ולא מהדיסק.

6. חזור אל Visual Studio וערוך שינוי בקובץ ContactForm.txt. באפשרותך לשנות את הטקסט, או להוסיף רווח. שמור את השינויים שעשית, ואז מלא את הטופס שוב. הפעם תקבל את הודעת הדואר ללא ההודעה אודות המטמון. משמעות הדבר היא שהפעם הקובץ נקרא מהדיסק ולא מהמטמון.



תרשים 14-14

כשאתה לוחץ על הלחצן Send כדי לשלוח את הטופס לשרת, מתעורר הקוד ב-`btnSend_Click`. במקום לקרוא את קובץ גוף ההודעה באופן ישיר מהדיסק תוך שימוש ב-`ReadAllText`, כפי שנעשה קודם, הקוד בודק תחילה אם הקובץ נמצא במטמון:

VB.NET

```
Dim fileName As String = Server.MapPath("~/App_Data/ContactForm.txt")
If Cache("ContactFormMailBody") Is Nothing Then
    mailBody = System.IO.File.ReadAllText(fileName)
    Cache.Insert("ContactFormMailBody", mailBody, New CacheDependency(fileName))
Else
    mailBody = Cache("ContactFormMailBody").ToString() & vbCrLf & "(File from the cache)"
End If
```

C#

```
string fileName = Server.MapPath("~/App_Data/ContactForm.txt");
if (Cache["ContactFormMailBody"] == null)
{
    mailBody = System.IO.File.ReadAllText(fileName);
    Cache.Insert("ContactFormMailBody", mailBody, new CacheDependency(fileName));
}
else
{
    mailBody = Cache["ContactFormMailBody"].ToString() + "\r\n(File from the cache)";
}
```

כשהפריט אינו נמצא במטמון - השורה `Cache("ContactFormMailBody")` תחזיר `Nothing` ב-VB.NET, או `null` ב-C#. בשני המקרים הקובץ נקרא מהדיסק תוך שימוש ב-`ReadAllText` ומוקצה ל-`mailBody`. גוף הודעת הדואר (מחרוזת) מאוחסן במטמון על ידי השיטה `Insert`. הפרמטר הראשון של שיטה זו מצפה למפתח המטמון הייחודי כדי לזהות את הפריט (בדוגמה שלנו, `ContactFormMailBody`). כפרמטר שני הוא מצפה שהנתונים הממשיים יוטמנו. לבסוף, הוא מקבל מופע של המחלקה `CacheDependency`. אחת מפונקציות הבנייה של מחלקה זו מצפה לנתיב הפיזי אל הקובץ שהיא אמורה לפקח עליו. בדוגמה שלנו היא מקבלת את המשתנה `fileName` שמכיל נתיב דומה ל- `C:\BegASPNET\Site\App_Data\ContactForm.txt`. כשקובץ זה משתנה, הפריט מוסר מהמטמון באופן אוטומטי. בפעם הבאה שישלח טופס הקשר, ייקרא עותק טרי של הקובץ מהדיסק ויאוחסן במטמון לשימוש עתידי.

עצות מעשיות לגבי נתונים

להלן כמה עצות מעשיות לגבי נתונים.

- ⊙ בכל פעם שתשתמש במאפייני הסגנון המרובים של הפקדים קשורי-הנתונים, חשוב על שימוש במאפיין `CssClass` כתחליף להגדרת מאפייני הסגנון האינדיבידואליים שכלולים באלמנט.
- ⊙ הקטע שעוסק במחזור החיים של פקד הכיל תרגיל שהראה את הסדר שבו מתרחשים האירועים השונים והסביר כיצד להציגם. תוכל להרחיב את הדוגמה ולכתוב קוד עבור אירועים רבים יותר. בנוסף, ביכולתך להוסיף פקדים לדרך ולטפל גם באירועים שלהם כדי להבין טוב יותר את האירועים השונים. מכיוון שהבנה טובה של אירועים אלה וסדרם היא, לעתים קרובות, קריטית לכתיבת יישומי אינטרנט, הזמן שתקדיש להבנת נושא זה ישתלם בעתיד.
- ⊙ בכל פעם שתכתוב דף שניגש לבסיס נתונים או למשאבים אחרים שהם איטיים או נדירים, כמו קבצים או שירותי אינטרנט, חשוב אם ניתן לשפר ולייעל אותם על ידי שימוש במטמון. למרות שאין קושי רב להוסיף את יכולת המיטמון בשלב מאוחר יותר, מומלץ להפעילו מוקדם ככל האפשר. בדרך זו ייבחן קוד המיטמון בצורה יסודית, והיישום יפיק תועלת כבר מההתחלה מהביצועים המשופרים הנובעים משימוש במיטמון.

סיכום

פרק זה עסק בכמה מהנושאים היותר מתקדמים של הצגת נתונים עם פקדי הנתונים, שלא נדונו בפרקים קודמים מתוך כוונה לאפשר לך להתמקד תחילה בעקרונות הליבה של הגישה לנתונים. כעת, כשיש לך הבנה טובה של גישה לנתונים באופן כללי, וגם ל-`SQL`, `LINQ`, פקדים קשורי-נתונים, ופקדי הנתונים השונים כמו `SqlDataSource` ו-`LinqDataSource`, קל הרבה יותר לראות כיצד לשנות את המראה וההתנהגות של הפקדים.

הפרק התחיל במבט מעמיק במספר הרב של אלמנטי הסגנון (עיצוב) שיש לרוב הפקדים קשורי-הנתונים. למדת כיצד לקבוע תצורת סגנונות כמו `RowStyle` ו-`AlternatingRowStyle` כדי ליצור "אפקט זברה" בפקדים קשורי-נתונים, כמו `GridView`. למדת גם להשתמש בסגנונות כמו `EditItemStyle`, `HeaderStyle` ו-`FooterStyle` כדי לשנות את האזורים השונים של פקדי הנתונים.

למרות שמידע העיצוב מוחל לעתים קרובות בבת אחת על כל השורות בפקד הנתונים, יש מצבים שבהם שינוי המראה של פריטים יחידים בפקד הוא מותנה. ראית דוגמה שהשתמשה באירוע `RowDataBound` כדי לשנות את צבעי הרקע והטקסט של סקירות שכרגע אין הרשאה או רצון לפרסם אותן. ראית גם כיצד לאפשר את הקישור למחיקת סגנונות, וכיצד לבטל את פעולתו כדי למנוע ממשתמשים למחוק סגנונות שעדיין יש סקירות שקשורות אליהם.

כדי להבין את האירועים השונים שעומדים לרשותך, ואת הסדר שבו הם מתעוררים, עקבת אחר תרגיל נסה זאת בשם "ראיית מחזור החיים של הדף ושל הפקדים בפעולתו". התרגיל הראה כיצד לכתוב את שם האירוע ב-`Label`. למרות שהדוגמה רחוקה מלהיות מושלמת (יש הרבה יותר

אירועים שבהם תוכל לטפל), היא מראה את העיקרון הבסיסי לקביעת סדר האירועים. בכל פעם שאינך בטוח לגבי אירועים מסוימים, מתי הם מתעוררים, או האם הם בכלל מתעוררים, תוכל להשתמש במושגים שלמדת בתרגיל זה כדי למצוא את האירועים המדויקים וסדרם.

הפרק הסתיים בדיון ביכולות המיטמון השונות שבהן תומכת ASP.NET. ראית דוגמה של מיטמון הפלט שתפקידו להטמין את הפלט של הדף כולו. כדי לאפשר לדף דינמי להיות מוטמן (לדוגמה, דף שמשנה את הנתונים שהוא מציג כתלות במחרוזת שאילתה או במשתנה אחר), תוכל להשתמש ב-VaryByParam כדי להורות ל-ASP.NET runtime להטמין עותק של כל דף שיש לו מחרוזת שאילתה ייחודית.

בנוסף למיטמון הפלט, באפשרותך גם להטמין נתונים ברמת פקד מקור הנתונים. כל שעליך לעשות הוא להגדיר כמה מאפיינים, ולכל השאר ידאג ASP.NET runtime.

אם דרושה שליטה מדויקת במטמון, תוכל לגשת אליו בתכנות. האובייקט Cache חושף שיטות שונות שמאפשרות להכניס נתונים למטמון, לגשת אליהם או למחוק נתונים שמאוחסנים במטמון. שימוש במחלקה CacheDependency מאפשר ליצור קשרים בין משאב, כמו קובץ פיזי בדיסק, לבין הפריט המוטמן. כשהמשאב משתנה (משום שהוספת טקסט כלשהו לקובץ טקסט, למשל), הפריט שבמטמון יאבד את תוקפו באופן אוטומטי. כלומר, בפעם הבאה שיפנו למשאב זה, הוא יילקח ממקומו המקורי בדיסק.

מיטמון הוא הנושא שמסיים פרק זה, שבו למדת כמה נושאים מתקדמים שעוסקים בטיפול בנתונים ביישומי אינטרנט בסביבת ASP.NET. בפרק הבא תלמד כיצד להגן על חלק מנתונים אלה - למשל, התיקייה Management - מפני משתמשים בלתי-מורשים על ידי יישום של מנגנון האבטחה שמספקת ASP.NET.

תרגילים

1. בתרגיל זה תוכל להניח שיש לך טופס אינטרנט פשוט עם Button (לחצן) יחיד עליו. כאשר תלחץ על Button בדפדפן, תגרום ל-postback, והשרת יעורר את האירוע Click שלו. מה קורה קודם, האירוע Load של הפקד Page או האירוע Click של הפקד Button?
2. ה-GridView מאפשר לשנות לסירוגין צבעים (ומידע עיצוב אחר) של השורות שהוא מציג. איך תגדיר את GridView באופן שיציג לסירוגין שורות עם רקע שחור וטקסט לבן, ושורות עם רקע לבן וטקסט שחור?
3. טיפלת בשגיאה באירוע של פקד קשור-נתונים שנמצא בקוד ברקע. מהי הדרך הנכונה למנוע את הצגת השגיאה בדף?

פרק 15

אבטחת אתר אינטרנט

בסביבת ASP.NET 3.5

עד כה יצרת באתר האינטרנט דפים שהיו נגישים לכל המבקרים באתר. לא הייתה לך דרך לחסום משאבים מסוימים כמו קבצי ASPX או אפילו תיקיות שלמות, מפני משתמשים כלשהם. פירוש הדבר, לדוגמה, שכרגע כל אחד יכול לגשת לתיקיית Management שלך ולשנות את הסגנונות ואת הסקירות השונות.

בוודאי לא תרצה שכל אחד יוכל לשנות את אתר האינטרנט שלך. אתה צריך אפוא לחשוב על אסטרטגיית אבטחה טובה שתחסום את הגישה של משתמשים בלתי-רצויים לתכנים מסוימים. אתה צריך גם לחשוב על מנגנון שיאפשר למשתמשים חדשים להצטרף לאתר, ובו-בזמן יאפשר לך לציין משתמשים מסוימים כמנהלים של אתר האינטרנט, ולהעניק להם הרשאות גישה מיוחדות.

ASP.NET 3.5 כוללת את כל הכלים הדרושים ליצירת מנגנון אבטחה יציב ובטוח. בפרק זה תלמד כיצד להשתמש בכלים אלה באתר אינטרנט הפועל בסביבת ASP.NET.

הפרק עוסק בהרחבה בנושאים הבאים:

- ⊙ מונחים חשובים בנושאי אבטחה.
 - ⊙ שירותי היישום של ASP.NET שמאפשרים אבטחה.
 - ⊙ איך לאפשר למשתמשים להירשם לפתיחת חשבון באתר, תוך שימוש בפקדי שרת מובנים.
 - ⊙ מתן אפשרות למשתמשים לאפס את הסיסמה שלהם באופן עצמאי, או לבקש חדשה.
 - ⊙ כיצד לנהל את המשתמשים והתפקידים בבסיס הנתונים בזמן הפיתוח.
 - ⊙ כיצד להציג תכנים שונים למשתמשים שונים על פי הרשאות הגישה שלהם במערכת.
- לפני שתתחיל להתבונן בדרך היישום של אבטחה בסביבת העבודה ASP.NET, עליך להכיר כמה מונחים חשובים שתפגוש בהם בכל דיון בנושא אבטחה.

היכרות עם נושאי האבטחה

למרות שאבטחה עשויה להיות נושא מורכב למדי, היא סובבת לעתים קרובות סביב שלוש שאלות חשובות:

- ⊙ מי אתה?
- ⊙ איך תוכל להוכיח זאת?
- ⊙ מה אתה מורשה לעשות במערכת?

זהות: מי אתה (identity)

זהות (identity) היא מה שמגדיר אותך. התשובה לשאלה מהי זהות תלויה בהקשר שבו נשאלת השאלה. כאזרח של מדינה, זהותך סובבת סביב אישיותך, שמך הרשמי ותאריך הולדתך, ואולי גם מספר תעודת הזהות שלך. עם זאת, עבור אתר אינטרנט כמו p2p.wrox.com, אתר האינטרנט של קהיליית Wrox, זהותך עשויה להיות מורכבת משמך וכתובת הדואר האלקטרוני שלך.

לא חשוב מה אתה כולל במושג זהות, זו הדרך להתייחס אליך. אבל איך מישוהו אחר מכיר אותך? ואיך הוא יכול להיות בטוח שבאמת אתה הוא זה שמנסה להתחבר (log on) לאתר, התשובה לכך היא המושג אימות - authentication.

אימות: איך תוכל להוכיח מי אתה?

אימות (authentication) עוסק במתן ראיות לגבי זהותך. כשתירשם לקבלת כרטיס ספרייה, אתה עשוי להתבקש להציג את תעודת הזהות שלך כדי להוכיח שהשם שביקשת לרשום בכרטיס הוא אכן שמך. באתר אינטרנט כמו p2p.wrox.com, עליך לספק כתובת דואר אלקטרוני וסיסמה. שתי פיסות מידע אלו יחד, מהוות את הראיה שמוכיחה את זהותך. מנגנונים רבים אחרים משמשים לאימות, כולל טביעות אצבעות, סריקת קשתית העין, כרטיסים חכמים ואסימונים (כאשר הראיה מאוחסנת במשהו מוחשי), ועוד. עם זאת, לאור הדיון באבטחת אתרי אינטרנט בסביבת ASP.NET, פרק זה דבק בכתובת דואר אלקטרוני וסיסמה כאמצעי האימות.

הרשאה: מה אתה מורשה לעשות? (authorizations)

המערכת מעניקה הרשאות גישה (authorizations) לאזורים מסוימים באתר בהתאם לתשובה לשאלה מי אתה. חשוב, לדוגמה, על המטה המאובטח של סוכנות לביטחון לאומי בסרט פעולה. גם כשהדמות המרכזית מורשית להיכנס לבניין, הרי שלעתים קרובות האדם מנוע מלהיכנס לאזורים מסוימים מפני שהוא חסר את ההרשאות המתאימות (העובדה שבאותם סרטים הם משיגים בסופו של דבר את ההרשאה בתוך שתי דקות של פריצה למערכת אינה נוגעת לענייננו).

כדי לקבוע מה מותר למשתמש לעשות, המערכת צריכה לדעת שני דברים: מהן ההרשאות של המשתמש הנוכחי, וכללי הרשאה (authorization rules) של המשאב שהמשתמש מנסה לגשת אליו.

הרשאת הגישה של המשתמש מבוססת על שם המשתמש (הזהות שהשם מייצג) והתפקידים (או קבוצות אבטחה) שהמשתמש מוקצה אליהם (רשות). באופן דומה, משאבים יכולים להיפתח או להיחסם למשתמשים או לתפקידים מסוימים. כשיש התאמה בין המשתמש הנוכחי לבין כללי הגישה עבור המשאב שהמשתמש מנסה לגשת אליו, הגישה מוענקת למשתמש. אם המשתמש חסום באופן מפורש, בקשת הגישה נדחת. הנח למשל שיש קובץ שהגישה אליו מוגבלת למשתמש Tom ולקבוצה Developers. המשתמש Tom יכול לגשת לקובץ מבלי לבדוק אם הוא משתייך לקבוצה Developers. בו בזמן, המשתמשת Jacqueline חייבת להשתייך לקבוצת Developers כדי שתורשה לגשת לקובץ.

בהמשך הקריאה בפרק זה תלמד כיצד להשתמש במושגים אלה. היישום של מושגי אבטחה אלה ב-ASP.NET מתבצע על ידי שירותי היישום (application services), ובהם נדון להלן.

מבוא לשירותי היישום של ASP.NET

בגרסאות מוקדמות של ASP.NET הייתה תמיכה כלשהי באבטחה. עם זאת, הן חסרו את הרמה הגבוהה של הפקדים והמושגים שכלולים בגרסה ASP.NET 3.5. ביישומים הפועלים בסביבת ASP.NET 1.x היה צורך לכתוב קוד רב כדי ליישם אסטרטגיית אבטחה יציבה. החיסרון של כתיבת קוד זה - לעתים קרובות היית צריך לחזור עליו בכל אתרי האינטרנט שלך. במילים אחרות, נאלצת להמציא את הגלגל בכל פעם שפיתחת אתר חדש.

בעיות אלו נפתרו עם כניסת ASP.NET 2.0 לשוק, על ידי שירותי היישום שהיו כלולים בה. שירותי היישום הם למעשה שכבת תוכנה החוצצת בין יישום האינטרנט לבין מחסן הנתונים. הם מאפשרים להשתמש בהם עם תוספת של קוד מועט או ללא תוספת קוד בכלל.

ASP.NET כוללת כמה שירותים, שהחשובים ביותר מביניהם:

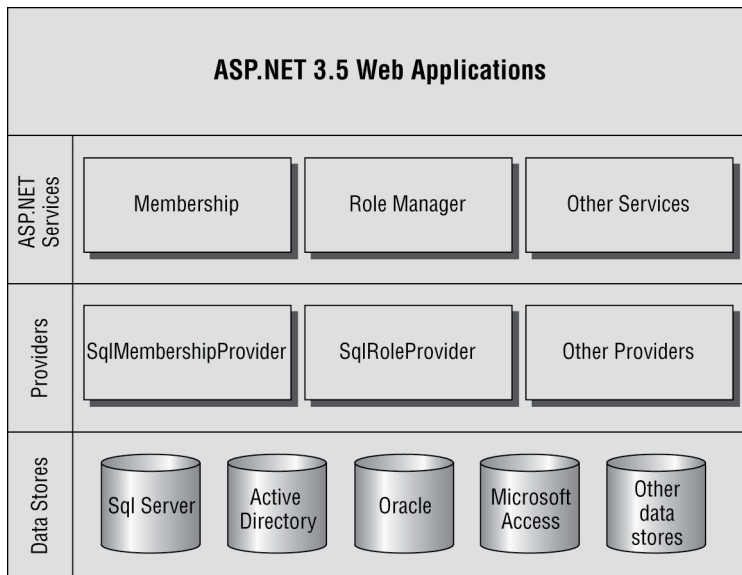
- ⊙ **Membership** (חברות): השירות מאפשר לנהל ולפעול עם חשבונות משתמש במערכת.
- ⊙ **Roles** (תפקידים): השירות מאפשר לנהל את התפקידים שעשויים להיות מוקצים למשתמשים.
- ⊙ **Profiles** (פרופילים): השירות מאפשר לשמור נתונים ייחודיים למשתמש בבסיס נתונים עורפי.

תרשים 1-15 מציג סקירה של שירותים אלה ומראה את קשריהם עם היישום, ואת מחסני הנתונים הבסיסיים שהשירותים עשויים להשתמש בהם.

בראש התרשים מופיעים יישומי ASP.NET 3.5 שמייצגים את יישומי האינטרנט שאתה בונה. יישומים אלה יכולים להכיל פקדים כמו פקדי התחברות או פקדי כניסה (login) שיידונו להלן, שבתורם יכולים לדבר עם שירותי היישום של ASP.NET כמו חברות ופרופילים. כדי ליצור פתרון גמיש, שירותים אלה אינם מדברים ישירות עם מקור נתונים, אלא עם ספק (provider) שעוצב. ספק הוא קטע קוד בר-החלפה שמתוכנן למשימה מסוימת. לדוגמה, במקרה של שירותי חברות, ספק החברות מתוכנן לפעול עם משתמשים במחסן הנתונים הבסיסי. תוכל לעצב ספקים שונים

עבור אותו שירות יישום, בהתאם לצרכיך. לדוגמה, ASP.NET מגיעה עם ספק עבור שרת SQL, שמאפשר לשירותי החברות לדבר עם בסיס נתונים מסוג שרת SQL. זהו ספק אידיאלי עבור אתרי אינטרנט כמו PlanetWrox.com. ASP.NET מגיעה גם עם הספק Active Directory שמאפשר ליצור ולנהל משתמשים ב- Active Directory של Windows. ספק זה מתאים במיוחד לרשתות סגורות כמו אינטראנט.

כל ספק זקוק למחסן נתונים, שמיוצג על ידי החלק התחתון של תרשים 1-15. כל ספק נכתב כדי לפעול מול מחסן נתונים ספציפי. לדוגמה, SqlMembershipProvider ו-SqlRoleProvider תוכננו כדי לפעול עם בסיס הנתונים SQL SERVER של מיקרוסופט.



תרשים 1-15

בהמשך הפרק תלמד כיצד להשתמש בספקי החברות והתפקיד של SQL SERVER. בפרק הבא תכיר את ספק הפרופיל של SQL SERVER. ניתן לעצב את שלושת הספקים באופן ששלושתם ישתמשו בבסיס נתונים SQL SERVER יחיד, דבר שיקל על מרכז כל נתוני המשתמשים.

במצב האידיאלי אינך אמור לעסוק בספקים ישירות. בנסיבות רגילות מוגדרת תצורת הספקים השונים של יישום אינטרנט במקום מרכזי. השימוש בספקים אלה מתבצע אז על ידי פנייה לשירותי היישום. למרות שביכולתך לגשת לשירותים אלה ישירות מהקוד, הרי שבדרך כלל תשתמש בפקדי ההתחברות (login) המובנים של ASP.NET כדי שיעשו עבורך את "העבודה השחורה". בפקדים אלה נדון עכשיו.

הצגת פקדי התחברות (Login)

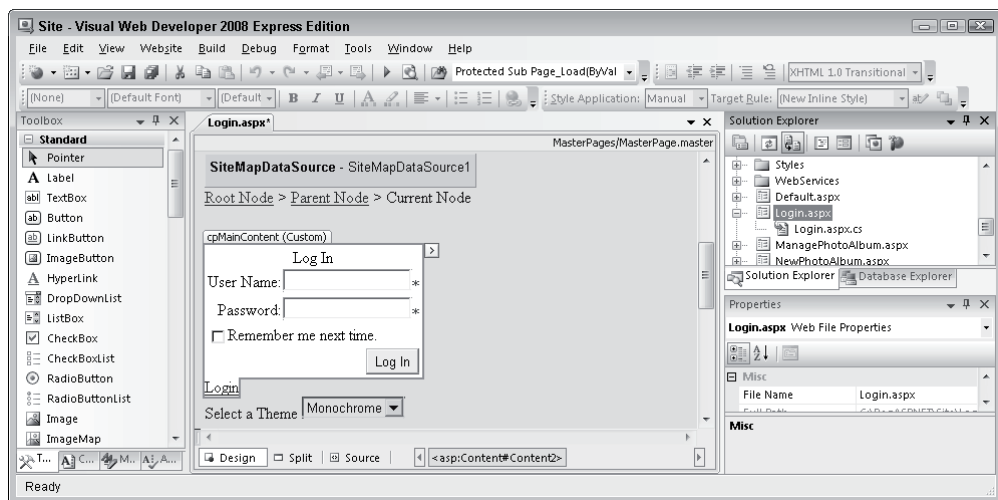
פקדי ההתחברות (login controls) שכלולים ב- ASP.NET 3.5 מסירים חלק ניכר מהמורכבות שכרוכה בדרך כלל בכתיבת שכבת אבטחה עבור יישום אינטרנט. פקדי ההתחברות הזמינים

עוטפים למעשה את כל הקוד שאתה זקוק לו, כדי לנהל משתמשים ולבדוק את תקפותם. במקום לפעול ישירות עם בסיס הנתונים, פועלים פקדים אלה על ידי יצירת קשר עם הספק באמצעות שירותי היישום. התרגיל הבא מראה כיצד ליצור דפי Login ודפי רישום (Sign Up) פשוטים, שמאפשרים למשתמש חדש ליצור חשבון ואז להתחבר. בסעיף שלאחר התרגיל נתבונן בשבעת פקדי ההתחברות.

נסה זאת יצירת דפי Login ו-Sign Up

בתרגיל זה תסיים את בניית דף ההתחברות שיצרת בפרק 6. תיצור גם דף חדש שמאפשר למשתמש להירשם לפתיחת חשבון חדש באתר האינטרנט Planet Wrox.

1. משורש האתר, פתח את הדף Login.aspx שיצרת בפרק 6, והעבר אותו ל- Design View.
2. מהקטגוריה Login של ארגז הכלים, גרור פקד LoginStatus ושחרר אותו בדף.
3. גרור פקד Login ושחרר אותו על LoginStatus באופן שיופיע מייד מעליו. שני הפקדים נראים בתרשים 15-2. LoginStatus מופיע כקישור Login קטן מתחת לפקד Login).



תרשים 15-2

4. פתח את טבלת המאפיינים של הפקד Login והגרר את שני המאפיינים שמוצגים בטבלה הבאה.

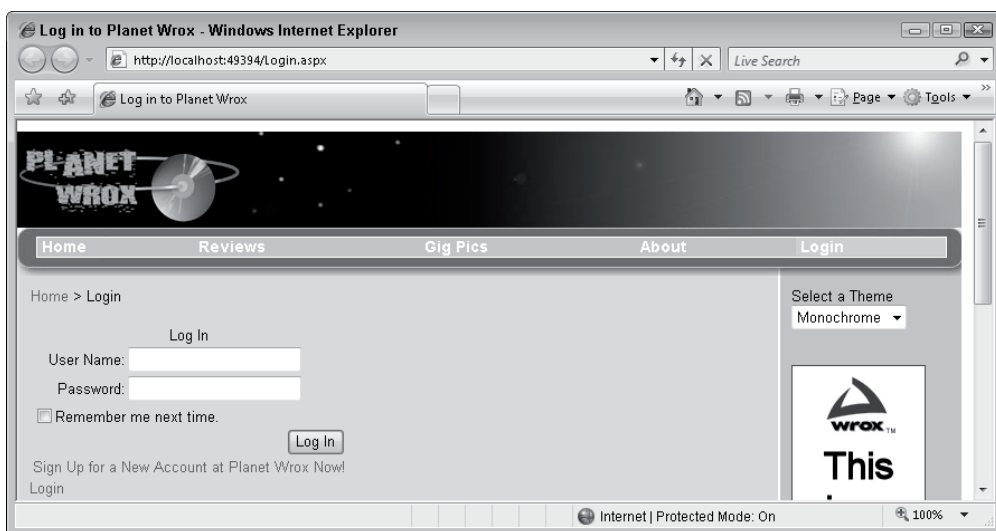
מאפיין	ערך
CreateUserText	Sign Up for a New Account at Planet Wrox Now!
CreateUserUrl	SignUp.aspx

5. בשורש האתר, צור דף חדש בשם SignUp.aspx שמבוסס על התבנית המותאמת אישית שלך, וקבע עבורו את הכותרת Sign Up for a New Account at Planet Wrox.

6. העבר את הדף ל- Design View וגרור פקד CreateUserWizard מארגז הכלים אל אזור התוכן העיקרי של הדף. שמור וסגור את הדף.
7. בשורש האתר, פתח את הקובץ web.config ואתר בו את האלמנט <authentication>. את המאפיין mode שנה מ- Windows ל- Forms:

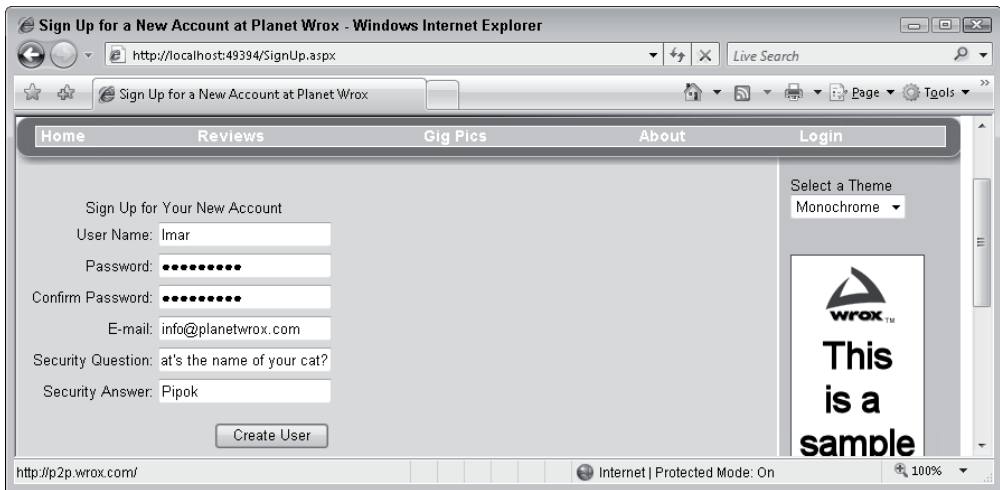
```
<authentication mode="Forms"/>
```

8. חזור אל Login.aspx והקש Ctrl+F5 כדי לפתוח את הדף בדפדפן. כעת תקודם בברכה עם תיבת login כפי שמוצג בתרשים 15-3.
- שים לב שהמזב של Login שנמצא תחת הפקד Login מוגדר כרגע כ-Login, כדי לציין שעדיין לא התחברת. אם במקום זאת הטקסט יאמר Logout, ודא שאכן הגדרת ל-Forms את האימות בקובץ web.config.
9. נסה להתחבר על ידי הזנת שם משתמש וסיסמה אקראיים. פעולה זו תיכשל, מכיוון שעדיין לא יצרת חשבון.



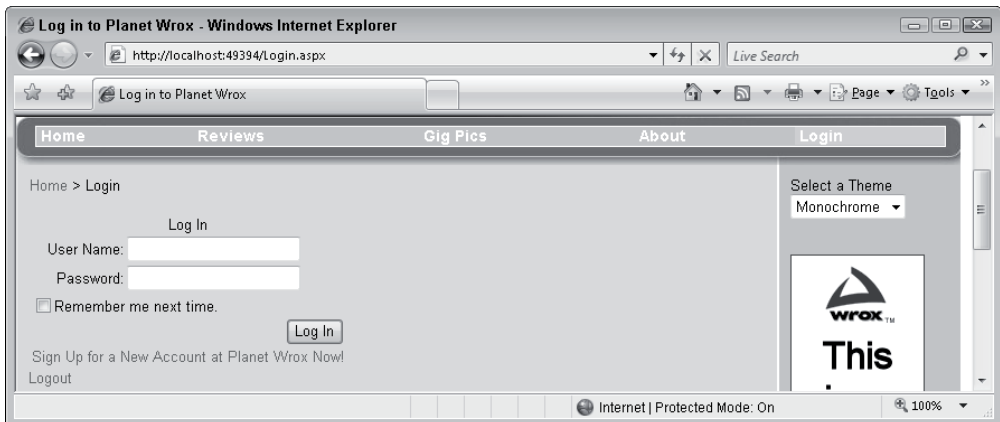
תרשים 15-3

10. לחץ על הקישור Sign Up שתחת הפקד Login כדי לעבור לדף SignUp.aspx, וצור חשבון על ידי הזנת פרטיך האישיים (ראה תרשים 4-15). כברירת מחדל, הסיסמה צריכה להיות בעלת אורך של שבעה תווים לפחות, וחייבת להכיל תו אלפאנומרי אחד לפחות. שים לב שמספר אינו בגדר תו אלפאנומרי, ועליך לוודא שסיסמתך מכילה לפחות תו אחד כמו # או \$ או *. כך, לדוגמה, Password אינה סיסמה תקפה בעוד שהסיסמה Pass##Word תקבל. שים לב שהסיסמה רגישה לרישיות (אותיות גדולות). רשום בצד את שם המשתמש והסיסמה שהזנת, מכיוון שמאוחר יותר תזדקק למידע החשבון.
- את שאלת האבטחה והתשובה עליה אתה יכול ליצור בעצמך. במקרה שתשכח את הסיסמה, תתבקש לספק את התשובה לשאלת האבטחה.



תרשים 15-4

11. לחץ על הלחצן Create User כדי לגרום ליצירת החשבון במערכת. כשהדף ייטען מחדש, תקבל אישור לכך שהחשבון נוצר. בשלב זה התעלם מהלחצן Continue (עדיין לא כתבת עבורו התנהגות כלשהי), ובמקום זאת לחץ באתר האינטרנט על הפריט Login ב-Menu או על TreeView (הדבר תלוי בערכת הנושא הנוכחית שבחרת). תועבר שוב ל-Login.aspx כשהפקד Login.aspx מציין כעת שהתחברת. כשיוצרים חשבון חדש באמצעות CreateUserWizard, ההתחברות מתבצעת באופן אוטומטי. ניתן לשנות התנהגות זו על ידי הגדרת המאפיין LoginCreatedUser של הפקד ל-False. הקישור Logout (יציאה או ניתוק) מופיע בתרשים 15-5 מיד מתחת הפקד Login.



תרשים 15-5

12. לחץ על הקישור Logout ותנותק. כתוצאה מכך LoginStatus יציג שוב את הטקסט Login. בפקד Login, הזן את שם המשתמש והסיסמה שהכנסת בצעד 10, ולחץ על הלחצן Log In. כעת תחובר ותופנה שנית לדף הבית. לחץ על Login בתפריט כדי

לחזור שוב לדף Login. שים לב ש-LoginStatus השתנה, וכעת הוא מציג שוב את Logout, דבר שממחיש את העובדה שהתחברת בהצלחה. בשלב זה, עובדת החיבור אינה מוסיפה ערך רב, כל מה שתראה הוא שינוי הסטטוס של LoginStatus מ-Login ל-Logout. עם זאת, בהמשך הלימוד בפרק תראה כיצד להציע למשתמשים שהתחברו תוכן שונה.

כיצד זה פועל

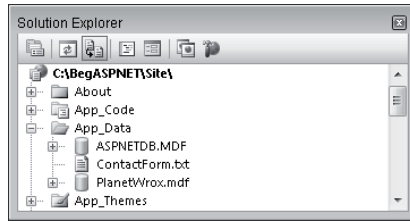
פרט לעובדה שהוספת כמה פקדי שרת של ASP.NET וקבעת את תצורתם, לא כתבת קוד כלשהו בתרגיל זה. עם זאת, הצלחת ליישם תהליך התחברות בעל פונקציונליות מלאה שמאפשר למשתמש להירשם לפתיחת חשבון ואז להתחבר לאתר. בזמן ההרשמה המשתמש מספק שם משתמש וסיסמה שבהם הוא ישתמש מאוחר יותר כדי להתחבר לאתר האינטרנט Planet Wrox. כיצד כל זה פועל? כפי שלמדת קודם, פקדי ASP.NET פונים אל ספקי שירותי היישום (application services providers), שזו שכבת תוכנה שנמצאת בין פקדי ההתחברות לבין בסיס הנתונים של SQL Server, ועוקבת אחר המשתמשים.

בפעם הראשונה שמשתמש מנסה להתחבר (או להשתמש בפקדי התחברות אחרים שדורשים גישה לבסיס הנתונים), הספק (provider) בודק אם היישום משתמש בבסיס נתונים עם האובייקטים החיוניים, כמו טבלאות. כברירת מחדל, הספק בודק את בסיס הנתונים על ידי התבוננות במחרוזת חיבור בשם LocalSqlServer. מחרוזת זו אינה נמצאת בקובץ web.config אלא מוגדרת בקובץ בשם machine.config שנמצא בתיקייה .NET Framework. יותר מאוחר תלמד עוד על קובץ זה. מחרוזת החיבור בקובץ זה נראית כך:

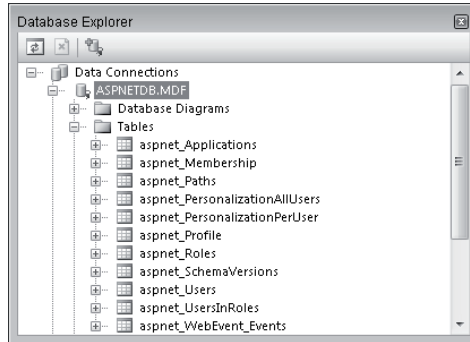
```
<connectionStrings>
  <add name="LocalSqlServer" connectionString="data source=.\SQLEXPRESS;
    Integrated Security=SSPI;AttachDBFilename=|DataDirectory|aspnetdb.mdf;
    User Instance=true"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

מחרוזת חיבור זו מיועדת עבור SQL Server 2005 Express Edition, גרסת החינם של SQL Server של מיקרוסופט (גרסת 2008 נתמכת אף היא). בפרק 18 ובנספח ב' תראה דוגמאות נוספות של מחרוזת חיבור שמיועדת לגרסה המלאה של SQL Server.

בפרק 12 למדת שמחרוזת חיבור שמשתמשת ב-|DataDirectory| מצביעה על בסיס נתונים שנמצא בתיקייה App_Data של האתר. הספקים דואגים לבסיס הנתונים שהם זקוקים לו כדי לפעול כראוי. בפעם הראשונה שנעשה שימוש בפקדי ההתחברות או בשירותי היישום האחרים, נבדק אם בסיס הנתונים aspnetdb.mdf נמצא במקום שצוין, ואם לא, הוא ייווצר באופן אוטומטי. כדי לברר כיצד נראה בסיס הנתונים, חזור אל Visual Web Developer, לחץ על התיקייה App_Data, ואז לחץ על הלחצן Refresh שבסרגל הכלים כדי לרענן את תוכן סייר הפתרון. אתה אמור לראות את בסיס הנתונים החדש כפי שמוצג בתרשים 6-15.



תרשים 15-6



תרשים 15-7

בסייר הפתרון לחץ לחיצה כפולה על בסיס הנתונים החדש כדי לפתוח אותו בחלון סייר בסיס הנתונים. הרחב פריטים כמו Tables ו-Views כדי לקבל את רשימת האובייקטים של בסיס הנתונים שהתווספו. תרשים 15-7 מציג רשימה כזו שמראה כמה מהטבלאות שמשמשות את שירותי היישום.

לאחר שבסיס נתונים זה נוצר בהצלחה, יכולים פקדי ההתחברות להשתמש בו. לדוגמה, כשיוצרים חשבון חדש תוך שימוש בפקד CreateUserWizard, מוכנסות רשומות לטבלאות aspnet_Membership ו-aspnet_Users. באופן דומה, כשמשמש מנסה להתחבר, נבדקים שם המשתמש והסיסמה שלו מול טבלאות אלו.

כדי לכפות על ASP.NET runtime להשתמש באימות מבוסס-טפסים (לדוגמה, שימוש בפקדים כמו הפקד Login בדף, וחלוקת cookies), עליך להגדיר את הערך Forms עבור המאפיין mode של אלמנט האימות שבקובץ web.config.

```
<authentication mode="Forms" />
```

אפשרויות נוספות עבור המאפיין mode הם: Windows (האבטחה מטופלת על ידי מערכת Windows עצמה), Passport (שימוש בשירותי Passport של מיקרוסופט) ו-None שמבטלת את האבטחה בכלל. בשארית הספר נשתמש ב-Forms באופן בלעדי.

האפשרות Remember Me Next Time של הפקד Login מועדת לשכחה יותר מכפי שאתה סבור. כשאפשרות זו מסומנת אזי בפעם הבאה שהמשתמש יבקר באתר הוא יחובר באופן אוטומטי. בפעם הראשונה שמשתמש מתחבר, השרת מגדיר cookie (עוגייה) שנשמרת עבור sessions עתידיים. עם זאת, תוקף העוגייה פג לאחר 30 דקות. כלומר, משתמש שחוזר לאתר לאחר פרק

זמן זה צריך לעבור אימות מחדש. כדי להאריך את משך הזמן שמשתמשים יישארו מחוברים, עליך להגדיר את המאפיין timeout של האלמנט <forms> שהוא עצמו בן ישיר של האלמנט <authentication> שבקובץ web.config. המאפיין timeout מקבל כערך מספר שלם שמייצג את משך פסק-הזמן בדקות. הקוד הבא מגדיר את פסק-הזמן ל-24 שעות (1440 דקות).

```
<authentication mode="Forms">
  <forms timeout="1440" />
</authentication>
```

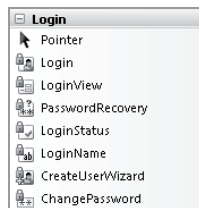
ערכים נמוכים יותר של timeout נחשבים בדרך כלל לבטוחים יותר, מכיוון שאינם מאפשרים גישה למשך זמן רב או בלתי מוגבל.

בסיס הנתונים שנוצר על ידי ASP.NET בתיקייה App_Data מקל מאוד על העבודה עם שירותי היישום השונים בתרחיש פיתוח. עם זאת, כשתצטרך לחשוף את היישום לראשונה בשרת ייצור ברשת שלך, או באמצעות ספק האינטרנט שלך (ISP), ייתכן שתזדקק לשליטה רבה יותר בבסיס הנתונים ובדרך שבה ניגשים אליו. פרק 18 שעוסק בפריסה (deployment), וגם נספח ב' שעוסק בתצורת SQL Server, מראים זאת ביתר פירוט.

עכשיו, משראית את פקדי ההתחברות פועלים בשיתוף עם בסיס הנתונים SQL Server Express שנוצר באופן אוטומטי, הגיע הזמן להתבונן ביתר פירוט בפקדים שבקטגוריה Login בארגז הכלים.

פקדי Login

ASP.NET 3.5 כוללת שבעה פקדי התחברות, שכל אחד מהם משרת מטרה נפרדת. תרשים 8-15 מראה את ארגז הכלים עם פקדי ההתחברות. בסעיפים שלהלן מוסבר כל אחד מפקדים אלה ביתר פירוט.



תרשים 8-15

Login

כפי שראית בתרגיל הקודם, הפקד Login מאפשר למשתמש להתחבר לאתר. הוא מדבר עם ספק החברות (membership provider), באמצעות שירותי היישום, כדי לדעת אם שם המשתמש והסיסמה מייצגים משתמש בר-תוקף במערכת. אם המשתמש עבר את בדיקת התוקף, מונפקת

עוגייה שנשלחת לדפדפן של המשתמש. בבקשות הבאות הדפדפן יעביר את העוגייה לשרת כדי שהמערכת תדע שהיא עוסקת עדיין במשתמש בר-תוקף. כל הגדרות העיצוב השונות עבור ספק החברות (האם הוא משתמש בעוגיות, כפי שתואר לעיל, מדיניות התפוגה של העוגייה, וכמה הגדרות נוספות), נקבעות באלמנט <authentication> שבקובץ web.config.

כדי ליצור דף התחברות בעל פונקציונליות מלאה, אתה זקוק להכרזת הפקד הבאה בלבד:

```
<asp:Login ID="Login1" runat="server" />
```

עם זאת, ברוב המצבים תרצה לשפר את המראה וההתנהגות של הפקד על ידי הגדרת אחד או יותר מהמאפיינים שמוצגים בטבלה הבאה.

מאפיין	תיאור
DestinationPageUrl	מאפיין זה מגדיר את ה-URL שהמשתמש נשלח אליו לאחר ניסיון התחברות מוצלח.
CreateUserText	מאפיין זה שולט בטקסט שמוצג, כדי להזמין משתמשים להירשם לפתיחת חשבון חדש.
CreateUserUrl	מאפיין זה מגדיר את ה-URL שהמשתמשים מועברים אליו כדי להירשם לפתיחת חשבון חדש.
DisplayRememberMe	מאפיין זה מציין אם הפקד יציג את האפשרות Remember Me. כשהמאפיין מוגדר כ-False, או שתיבת הסימון אינה מסומנת בזמן ההתחברות, המשתמשים צריכים לעבור אימות מחדש בכל פעם שהם סוגרים את הדפדפן ופותחים אותו מחדש.
RememberMeSet	מאפיין זה מציין אם האפשרות Remember Me סומנה בהתחלה.
PasswordRecoveryText	מאפיין זה שולט בטקסט שמוצג, כדי לומר למשתמשים שהם יכולים לאפס את הסיסמה או להשיג חדשה.
PasswordRecoveryUrl	מאפיין זה מציין את ה-URL שהמשתמשים מועברים אליו כדי לקבל את הסיסמה (החדשה).
VisibleWhenLoggedIn	מאפיין זה קובע אם הפקד מוצג כשהמשתמש הנוכחי מחובר. ברירת המחדל היא True.

בנוסף למאפיינים אלה, יש לפקד גם מגוון של מאפייני Text כמו LoginButtonText, TitleText, RememberMeText ו-UserNameLabelText. הם משמשים לשליטה בטקסט שמופיע בפקד ובפקדים הבנים השונים שלו, כמו Button ו-Label, שמהווים את ממשק המשתמש.

כמו לפקדים קשורי-הנתונים, כך גם לפקדי ההתחברות יש מאפייני סגנון רבים מאוד שמאפשרים להתאים את המראה שלהם. מומלץ לבדוק את הקטגוריה Styles של טבלת המאפיינים של

הפקדים כדי ללמוד כיצד להגדיר את אפשרויות העיצוב השונות. זכור, כמו עם הפקדים קשורי-הנתונים כך גם כאן, באפשרותך להעביר חלק גדול ממידע העיצוב לקבצי skin ו-CSS.

הפקד Login חושף גם כמה אירועים שבדרך כלל אינך צריך לטפל בהם, אבל עשויים להיות שימושיים מדי פעם. לדוגמה, האירוע LoggedIn מתעורר מייד לאחר שהמשתמש מתחבר. זהו מקום טוב לשלוח ממנו את המשתמש לדף אחר באופן דינמי אם ה-DestinationPageUrl אינו גמיש מספיק.

LoginView

LoginView הוא פקד שימושי שמאפשר להציג נתונים שונים למשתמשים שונים. הוא מאפשר להבדיל בין משתמשים אנונימיים למשתמשים מחוברים, ואפילו בין משתמשים בעלי תפקידים שונים. LoginView הוא מונע-תבנית (template driven), וככזה הוא מאפשר להגדיר תבניות שונות שמוצגות למשתמשים שונים. הטבלה הבאה מתארת את שתי התבניות העיקריות ואת האלמנט המיוחד RoleGroups.

תבנית	תיאור
AnonymousTemplate	תוכן תבנית זו מוצג רק למשתמשים שלא אומתו.
LoggedInTemplate	תוכן תבנית זו מוצג למשתמשים מחוברים בלבד. תבנית זו אינה יכולה להתקיים עם התבנית AnonymousTemplate. רק אחת מהשתיים נראית בכל רגע נתון.
RoleGroups	תבנית זו יכולה להכיל אלמנט RoleGroups אחד או יותר, שבתורו מכיל אלמנט ContentTemplate שמגדיר את התוכן עבור התפקיד המסוים. בעלי התפקיד או התפקידים שמורשים לראות את התוכן מוגדרים במאפיין Roles שיכול לקבל רשימת תפקידים שמופרדים זה מזה בנקודה ופסיק (;). האלמנט RoleGroups לא יכול להתקיים עם LoggedInTemplate. כלומר, אם משתמש הוא חבר באחד התפקידים של RoleGroup אזי התוכן של LoggedInTemplate הינו בלתי-נראה. בנוסף, מוצג רק התוכן של RoleGroup הראשון שתואם את תפקיד המשתמש.

הפקד LoginView עצמו אינו פולט סימני עריכה כלשהם, פרט לתוכן שאתה מגדיר באלמנטים שהם הבנים השונים של הפקד. פירוש הדבר שתוכל בקלות להטמיע אותו בין זוג תגיות HTML כמו `<h1>` ו-`<li>`.

קטע הקוד הבא מראה פקד LoginView שמגדיר תוכן עבור שלושה משתמשים שונים: מבקרים אנונימיים באתר, משתמשים מחוברים, ומשתמשים מחוברים שהם גם חברים בתפקיד Managers.

```
<asp:LoginView ID="LoginView1" runat="server">
  <AnonymousTemplate>
    Hi there visitor. Would you be interested in signing up for an account?
```



```

</AnonymousTemplate>
<LoggedInTemplate>
    Hi there visitor and welcome back to PlanetWrox.com.
</LoggedInTemplate>
<RoleGroups>
    <asp:RoleGroup Roles="Managers">
        <ContentTemplate>
            Hi there manager. You can proceed to the Management section.
        </ContentTemplate>
    </asp:RoleGroup>
</RoleGroups>
</asp:LoginView>

```

בהמשך הלימוד בפרק זה נמשיך ונעסוק ביצירת תפקידים וקביעת תצורתם.

LoginStatus

כפי שהודגם בתרגיל נסה זאת הקודם, הפקד LoginStatus מספק מידע אודות הסטטוס הנוכחי של המשתמש. הוא מספק קישור Login כשהמשתמש אינו מאומת, וקישור Logout כשהמשתמש כבר מחובר. ניתן לשלוט בטקסט הממשי שמוצג על ידי הגדרת המאפיינים LoginText ו-LogoutText. לחילופין, אפשר להגדיר את המאפיינים LoginImageUrl ו-LogoutImageUrl באופן שתוצג תמונה במקום טקסט. לבסוף, אפשר להגדיר את המאפיין LogoutAction באופן שיקבע אם הדף הנוכחי ירוענן כשהמשתמש מתנתק, או שהמשתמש יועבר לדף אחר לאחר ההתנתקות. דף היעד הזה נקבע על ידי הגדרת LogoutPageUrl.

בנוסף למאפיינים אלה, הפקד מסוגל לעורר שני אירועים, LoggingOut ו-LoggedOut, שמופעלים מייד לפני ומייד אחרי ההתנתקות של המשתמש.

LoginName

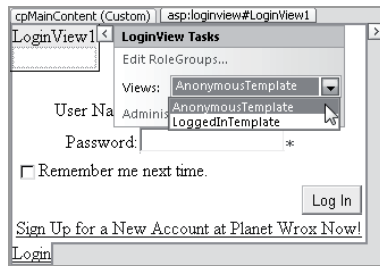
LoginName הוא פקד פשוט ביותר, שתפקידו להציג את שם המשתמש המחובר. כדי להטמיע את שם המשתמש בטקסט כלשהו, כמו You are logged in as Imar, תוכל להשתמש במאפיין FormatString. אם תכלול {0} במחרוזת עיצוב זו, היא תוחלף בשם המשתמש. בתרגיל הבא תראה כיצד זה פועל. בתרגיל תשנה את הדף Login ואת הדף הראשי של האתר כדי שיציגו את המידע הרלוונטי אודות המשתמש.

נסה זאת פעולה עם פקדי Login

בתרגיל זה תסתיר את הפקד Login בדף Login.aspx כשהמשתמש כבר מחובר, ובמקומו תציג הודעה. בנוסף, תוסיף טקסט לכותרת התחתונה של הדף שמציג את שם המשתמש יחד עם אפשרות להתנתק שוב.

1. פתח את Login.aspx ועבור ל- Design View. גרור LoginView חדש מארגז הכלים ושחרר אותו בראש הפקד Login כדי שיוצב ממש מעליו בדף.

2. פתח את לוח Smart Tasks של הפקד LoginView, ודא שמהרשימה הנפתחת Templates נבחרה התבנית AnonymousTemplate, כפי שמוצג בתרשים 9-15.



תרשים 9-15

3. לחץ על הפקד Login פעם אחת ואז הקש Ctrl+X כדי לחתוך אותו ל-clipboard. לחץ בתוך המלבן הלבן הקטן שמייצג את LoginView כדי להציב את הסמן בפקד, ואז הקש Ctrl+V כדי להדביק את הפקד Login לתוך LoginView.
4. פתח שוב את לוח Smart Tasks של LoginView ועבור ל-LoggedInTemplate. לחץ שוב בתוך המלבן הלבן הקטן של הפקד והקלד את הטקסט You are already logged in.
5. עבור ל- Markup View והתבונן בקוד. הפקד Login אמור להיות מוצג בתוך ה-AnonymousTemplate, בעוד הטקסט שהקלדת אמור להיות מוצג בין התגיות :LoggedInTemplate

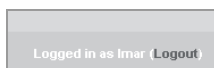
```
<asp:LoginView ID="LoginView1" runat="server">
  <LoggedInTemplate>
    You are already logged in
  </LoggedInTemplate>
  <AnonymousTemplate>
    <asp:Login ID="Login1" runat="server" CreateUserUrl="SignUp.aspx"
      CreateUserText="Sign Up for a New Account at Planet Wrox Now!">
    </asp:Login>
  </AnonymousTemplate>
</asp:LoginView>
```

6. שמור וסגור את הדף מכיוון שסיימת כעת את העיסוק בו.
7. פתח את הדף הראשי MasterPage.master ב- Markup View, ואתר את האלמנט <div> Footer בתחתית הדף. הסר את הטקסט Footer Goes Here והחלף אותו בפקד LoginName חדש על ידי גרירתו מארגז הכלים אל האלמנט <div>. הגדר את המאפיין FormatString שלו ל- {0}. Logged in as {0}.
8. גרור פקד LoginView חדש מארגז הכלים ושחרר אותו תחת הפקד LoginName, אבל עדיין בתחום האלמנט <div> Footer. עבור ל- Design View, ומלוח Smart Tasks של LoginView, מהרשימה הנפתחת Views, בחר LoggedInTemplate. במלבן הלבן של LoggedInTemplate הפעיל, גרור ושחרר פקד LoginStatus חדש.

9. עבור שוב ל- Markup View ועטוף את הקוד של LoginStatus בסוגריים. אתה אמור לקבל כעת את הקוד הבא:

```
<div id="Footer">
  <asp:LoginName ID="LoginName1" runat="server" FormatString="Logged in as {0}" />
  <asp:LoginView ID="LoginView1" runat="server">
    <LoggedInTemplate>
      (<asp:LoginStatus ID="LoginStatus1" runat="server" />)
    </LoggedInTemplate>
  </asp:LoginView>
  ...
</div>
```

10. שמור את כל השינויים ובקש את Login.aspx בדפדפן. התחבר תוך שימוש בחשבון ובסיסמה שיצרת בתרגיל קודם (ייתכן שתצטרך להתנתק תחילה על ידי לחיצה על הקישור Logout). אם אינך יכול להיזכר בשם המשתמש ובסיסמה, לחץ על הקישור Sign Up תחילה כדי ליצור חשבון חדש. שים לב שברגע שתהיה מחובר תציג הכותרת התחתונה את הטקסט שנראה בתרשים 10-15.



תרשים 10-15

11. לחץ על הפריט Login ב-Menu או ב-TreeView כדי לעבור לדרך Login. במקום הפקד Login אתה אמור לראות כעת הודעה שמציינת שאתה כבר מחובר.
12. לחץ על הקישור Logout בכותרת התחתונה שבתחתית הדרך. הדרך מתרענן ומציג שוב את הפקד Login. בנוסף, הטקסט שהופיע בכותרת התחתונה נעלם.

כיצד זה פועל

התחלת בהוספת LoginView לדרך Login כדי לעטוף את הפקד LoginView, ובהוספת הודעת טקסט. אילו רצית רק להסתיר את הפקד Login כשהמשתמש כבר מחובר, יכולת להגדיר את VisibleWhenLoggedIn ל-False.

```
<asp:Login ID="Login1" runat="server" VisibleWhenLoggedIn="false" />
```

עם זאת, במטרה להציג הודעה כשהמשתמש כבר מחובר, ובו-בזמן להסתיר את הפקד Login, עדיף בהרכבה להשתמש ב-LoginView. הצב את התוכן הרלוונטי בתבניות הזמינות, ובכל השאר יטפל הפקד.

הקוד בכותרת התחתונה של הדרך הראשי מכיל פקד LoginName שמציג את שם המשתמש שמחובר. הוא אינו מציג מאומה עבור משתמשים אנונימיים. כדי לשלוט בטקסט שמוצג, תשתמש במאפיין FormatString:

```
<asp:LoginName ID="LoginName1" runat="server" FormatString="Logged in as {0}" />
```

בזמן ביצוע מוחלף הסימון {0} בשם המשתמש.

כברירת מחדל מציג LoginStatus שהוספת את הקישור שמאפשר למשתמש להתחבר ולהתנתק. מכיוון ש-Menu או TreeView כבר מכילים קישור לדף Login, הכותרת התחתונה משתמשת שוב ב-LoginView רק כדי להציג את הטקסט של Logout כשהמשתמש מחובר כרגע. אם ברצונך להוסיף גם קישור ל-Login, תוכל להרחיב את LoginView עם תבנית אנונימית ו-LoginStatus נוסף, או שתסיר את LoginView כולו כדי ש-LoginStatus יוצג עבור כל המשתמשים.

עד כה ראית פקדים שמאפשרים למשתמש להתחבר, ומשתמשים בסטוס ההתחברות הנוכחי של המשתמש כדי להציג או להסתיר תכנים רלוונטיים. בנוסף להם כוללת הקטגוריה Login של ארגז הכלים שלושה פקדים שמאפשרים למשתמשים לבקש פתיחת חשבון חדש באתר, לשנות סיסמה קיימת או להשיג מחדש סיסמה שאבדה. נדון בפקדים אלה עכשיו.

CreateUserWizard

בתרגיל קודם ראית בחטף את CreateUserWizard בפעולה. אבל לפקד זה יש מה להציע הרבה מעבר להתנהגות הרגילה שראית באותו תרגיל.

לשם התחלה, לפקד יש רשימה ארוכה של מאפייני Text, כמו CancelButtonText, CompleteSuccessText, CreateUserButtonText ו-UserNameLabelText, שמשפיעים על הטקסט שמשתמשים בו בפקד. לכל המאפיינים יש ברירות מחדל טובות (עבור השפה האנגלית), אבל באפשרותך לשנות אותן (לעבודה בשפה העברית).

לפקד יש גם קבוצה של מאפיינים ששמן מסתיים ב-ImageUrl כמו CreateUserButtonImageUrl. מאפיינים אלה מאפשרים להגדיר תמונות עבור פעולות שונות של המשתמש במקום לחצני ברירת המחדל שמייצר הפקד. אם מגדירים מאפיין כלשהו מאלה ל-ImageUrl תקף, יש צורך להגדיר גם את ButtonType המתאים. לדוגמה, כדי לשנות את הלחצן Create User לתמונה, עליך להגדיר את המאפיין CreateUserButtonImageUrl לתמונה תקפה, ולהגדיר את CreateUserButtonType ל-Image.

ערך ברירת המחדל של ButtonType הוא Button, שמרונדר כברירת מחדל, ללחצנים הרגילים האפורים. באפשרותך גם להגדיר מאפיינים אלה ל-Link כדי שירונדרו כפקדי LinkButton רגילים.

בנוסף, הפקד חושף כמה מאפיינים שימושיים שביכולתך להגדיר כדי לשנות את התנהגות הפקד ומראהו. הטבלה הבאה מציגה מאפיינים אלה.

מאפיין	תיאור
ContinueDestinationPageUrl	המאפיין קובע את הדף שהמשתמשים מועברים אליו כשהם לוחצים Continue.

מאפיין	תיאור
DisableCreatedUser	המאפיין מציין אם בזמן יצירת החשבון המשתמש מוגדר כ-disabled, אם לאו. כשהוא מוגדר True, משתמשים אינם יכולים להתחבר לאתר עד שחשבונם מופעל. מאוחר יותר תראה כיצד להפעיל או לנתק חשבון משתמש.
LoginCreatedUser	המאפיין קובע אם המשתמש יחובר באופן אוטומטי לאחר יצירת החשבון.
RequireEmail	המאפיין קובע אם הפקד ישאל את המשתמש לכתובת הדואר האלקטרוני שלו, אם לאו.
MailDefinition	המאפיין מכיל כמה תת-מאפיינים שמאפשרים להגדיר את הודעת הדואר האלקטרוני (רשות) שתישלח למשתמשים לאחר שביקשו פתיחת חשבון.

שים לב שלפקד אין מאפיינים להסתרת שאלת האבטחה והתשובה עליה, או לשינוי המדיניות הקשוחה ביחס לסיסמה. המשתמשים חייבים להקליד סיסמה בת שבעה תווים לפחות, שמתוכם אחד לפחות הוא שאינו אלפאנומרי כמו # או *. במקום לשנות הגדרות אלו בפקד, עליך לקבוע את תצורתן בספק הבסיסי. מאוחר יותר בפרק, בסעיף "עיצוב יישום אינטרנט כדי שיפעל עם תפקידים", תראה כיצד זה פועל.

בתרגיל הבא תלמד כיצד לקבוע את תצורת האלמנט MailDefinition באופן ש-CreateUserWizard ישלח הודעת דואר אלקטרוני למשתמש החדש כדי לאשר את חשבונו, וכן את שם המשתמש והסיסמה לצורך פניות עתידיות.

נסה זאת שליחת אישור בדואר אלקטרוני על ידי CreateUserWizard

כדי שתרגיל זה יפעל, עליך לעצב את האלמנט <system.net> בקובץ web.config עם שם שרת דואר תקף (כפי שדנו בפרק 9).

1. הוסף קובץ Text לתיקייה App_Data וקרא לו SignUpConfirmation.txt.

2. הוסף את הטקסט הבא לקובץ, ואז שמור וסגור אותו.

```
Hi <% UserName %>,

Thank you for signing up for a new account at www.PlanetWrox.Com.

To log in to the site, use the following details:

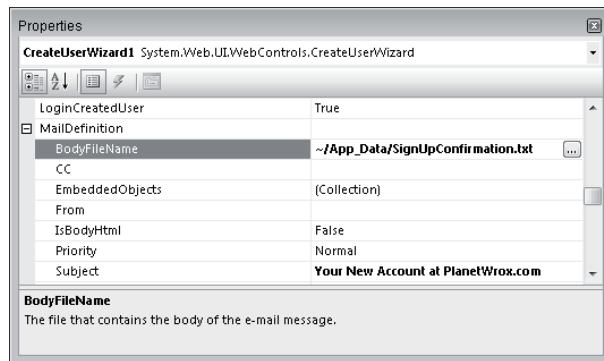
User name:      <% UserName %>
Your password:  <% Password %>

We look forward to your contributions.

The Planet Wrox Team
```

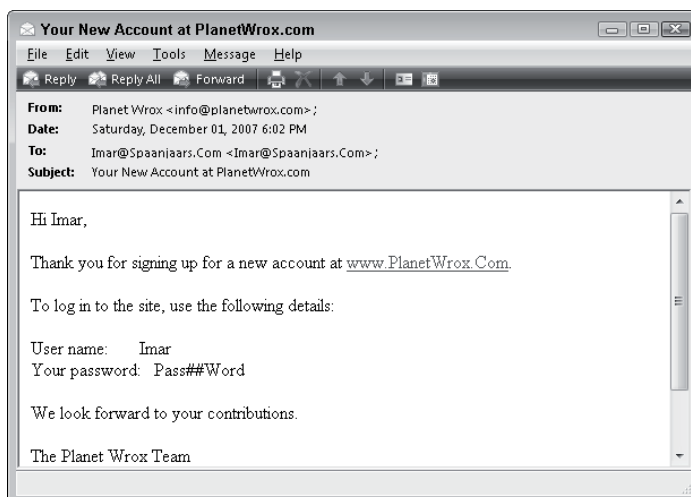
שים לב, שבשעה שאתה מקליד את שומרי המקום UserName ו-Password, הם עטופים בזוג תגיות צד-שרת (<% ו- >%), שמשמשות למתן משמעות מיוחדת לשומרי מקום אלה.

3. פתח את SignUp.aspx במצב Design View. אתר את המאפיין MailDefinition בטבלת המאפיינים של הפקד CreateUserWizard והרחב אותו. לחץ על המאפיין BodyFileName, לחץ על האליפסות לדפדוף בקובץ, ואז בחר SignUpConfirmation.txt שיצרת בתיקייה App_Data.
4. קבע את המאפיין Subject ל- Your New Account at PlanetWrox.com. כשתסיים, אמורה טבלת המאפיינים להיראות כמו בתרשים 11-15.



תרשים 11-15

5. שמור את השינויים ובקש בדפדפן את SignUp.aspx. הזן את הפרטים הנחוצים לחשבון חדש ולחץ Create User כדי לבקש חשבון. אם תקבל שגיאה שמדברת על ציון התכונה from באלמנט <smtp>, פנה לפרק 9. ודא שהזנת כתובת דואר אלקטרוני תקפה, אחרת עלול שרת הדואר להמשיך לדחותה.
- אם שרת הדואר דורש ממך להשתמש ב-SSL, אתה בבעיה. אין דרך להגדיר את המאפיין EnableSsl ישירות, כפי שהראינו בפרק 9. במקום זאת עליך לכתוב קוד משלך באירוע SendingMail של הפקד, אשר ישלח את ההודעה תוך שימוש באותו קוד לשליחת דואר אלקטרוני כמו זה שבפקד ContactForm.ascx. אפשר לגשת ל-e.Message של האירוע SendingMail כדי לקבל הרשאת גישה להודעה שצריכה להישלח.
6. לאחר זמן מה תקבל הודעת דואר אלקטרוני שמכילה את טקסט הברכה שהקלדת בצעד 2 (ראה תרשים 12-15).
- תרשים 12-15 מציג את ההודעה, כששומרי המקום של שם המשתמש והסיסמה הוחלפו בפרטים שהוכנסו בצעד 5.



תרשים 15-12

כיצד זה פועל

האשף CreateUserWizard כולל פונקציונליות מובנה לשליחת הודעת אישור למשתמש. הוא אינו שולח את ההודעה עד שתציין את האלמנט <MailDefinition>. השתמש במאפיין BodyFileName כדי להצביע על קובץ הטקסט או קובץ HTML שמשמש כגוף הודעת הדואר האלקטרוני.

בגוף זה אפשר להשתמש בשומרי המקום המיוחדים <% UserName %> ו- <% Password %> שמוחלפים באופן אוטומטי בשם המשתמש והסיסמה האמיתיים שהמשתמש מכניס בטופס ההצטרפות.

לפקד PasswordRecovery, שבו נדון עכשיו, יש פונקציונליות דומה.

PasswordRecovery

הפקד PasswordRecovery מאפשר למשתמשים לאחזר את הסיסמה הקיימת (כשהמערכת תומכת בכך), או לקבל סיסמה חדשה לחלוטין שמיוצרת באופן אוטומטי. במקרה האחרון הסיסמה נשלחת לכתובת הדואר האלקטרוני שהמשתמש הזין כשהגיש בקשה לפתיחת חשבון.

רוב המאפיינים של PasswordRecovery אמורים להיות מוכרים לך כעת. יש לו כמה מאפייני Text כמו GeneralFailureText שמוצג כשלא ניתן להשיג מחדש את הסיסמה, ו-SuccessText שמאפשר לקבוע את הטקסט שמוצג על ידי הפקד. יש לו גם מאפיינים שמסתיימים ב-ButtonImageURL ו-ButtonImageURL. מאפיינים אלה מאפשרים לשנות את המראה וההתנהגות של לחצני הפעולה השונים של הפקד. אם ברצונך להפנות את המשתמש לדף אחר לאחר שהצליח להשיג שוב את הסיסמה, הגדר את SuccessPageURL לאותו הדף.

כמו עם CreateUserWizard, גם ל-PasswordRecovery יש אלמנט MailDefinition שמאפשר להצביע על קובץ שברצונך לשלוח כגוף הודעת הדואר. תוכל להשתמש באותם שומרי מקום

של שם המשתמש והסיסמה כדי להתאים את ההודעה באופן אישי. אם תשמיט את MailDefinition, הפקד ישתמש בברירת המחדל של גוף ההודעה, כפי שתראה בתרגיל הבא.

ChangePassword

הפקד ChangePassword מאפשר למשתמשים קיימים ומחוברים לשנות את סיסמתם. יש לו מאפיינים רבים שמאפשרים לשנות דברים כמו טקסט, הודעות שגיאה ולחצנים, בדומה לפקדים CreateUserWizard ו-PasswordRecovery. יש לו גם אלמנט MailDefinition שמאפשר לשלוח אישור של הסיסמה החדשה אל כתובת הדואר האלקטרוני של המשתמש.

נסה זאת יישום הפונקציונליות של הסיסמה

בתרגיל זה תוסיף לאתר את הפקדים PasswordRecovery ו-ChangePassword כדי לאפשר למשתמשים לשנות סיסמה או להשיגה מחדש. מכיוון ששינוי סיסמה תופס רק לגבי משתמשים מחוברים, תוסיף פקד ChangePassword אל הדף שלו עצמו. בתרגיל מאוחר יותר בפרק תגן על דף זה כדי שרק משתמשים מאומתים יוכלו לגשת אליו.

1. פתח את Login.aspx ב- Markup View ואתר את תגית הסגירה `</asp:Login>` בתוך `<AnonymousTemplate>`. מייד אחריה הקלד שני אלמנטים `<br />` כדי ליצור שטח מתחת לפקד Login:

```
<asp:Login ID="Login1" runat="server" ... >
</asp:Login>
<br /><br />
```

2. מארגז הכלים, גרור פקד PasswordRecovery ושחרר אותו בעורך הטקסט מייד לאחר שני האלמנטים `<br />` שהוספת בצעד 1.

3. בין תגי הפתיחה והסגירה של הפקד PasswordRecovery הוסף אלמנט `<MailDefinition>`, ואז הגדר את Subject של הודעת הדואר האלקטרוני ל- `Your New Password for PlanetWrox.Com`. כעת, הקוד אמור להיראות כך:

```
</asp:Login>
<br /> <br />
<asp:PasswordRecovery ID="PasswordRecovery1" runat="server">
  <MailDefinition subject="Your New Password for PlanetWrox.Com"></MailDefinition>
</asp:PasswordRecovery>
```

4. שמור את השינויים וסגור את הקובץ.
5. בשורש האתר, צור דף אינטרנט חדש שמבוסס על התבנית המותאמת אישית שלך, וקרא לו `MyProfile.aspx`. קבע את כותרת הדף ל- `My Profile`.
6. עבור ל- Markup View, ובשומר המקום `cpMainContent` של התוכן צור אלמנט `<h1>`, שהטקסט הפנימי שלו מוגדר ל-`Profile`. מייד מתחת לכותרת הקלד טקסט שמסביר שהדף `My Profile` משמש לדברים כמו שינוי סיסמאות. עטוף את הטקסט בזוג תגיות `<p>` כדי לציין פסקה.

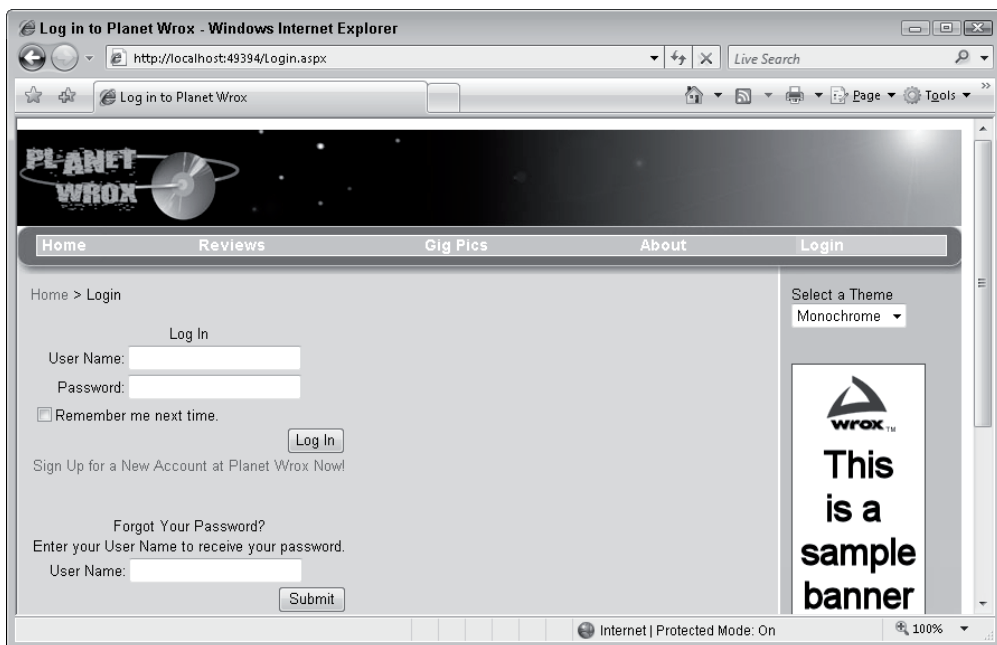
7. מארגז הכלים, גרור פקד ChangePassword ושחרר אותו אחרי תג הסגירה </p>. עליך לקבל כעת קוד דומה לזה:

```
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" Runat="Server">
  <h1>My Profile</h1>
  <p>The My Profile page allows you to make changes to your personal profile.
    For now, all you can do is change your password below.</p>
  <asp:ChangePassword ID="ChangePassword1" runat="server"></asp:ChangePassword>
```

8. פתח את Web.sitemap והוסף אלמנט חדש תחת הקטע About. הנח ל-url להצביע על ~/MyProfile.aspx והגדר את title ו-description ל-My Profile. כעת צפוי שתקבל את ה-XML הבא:

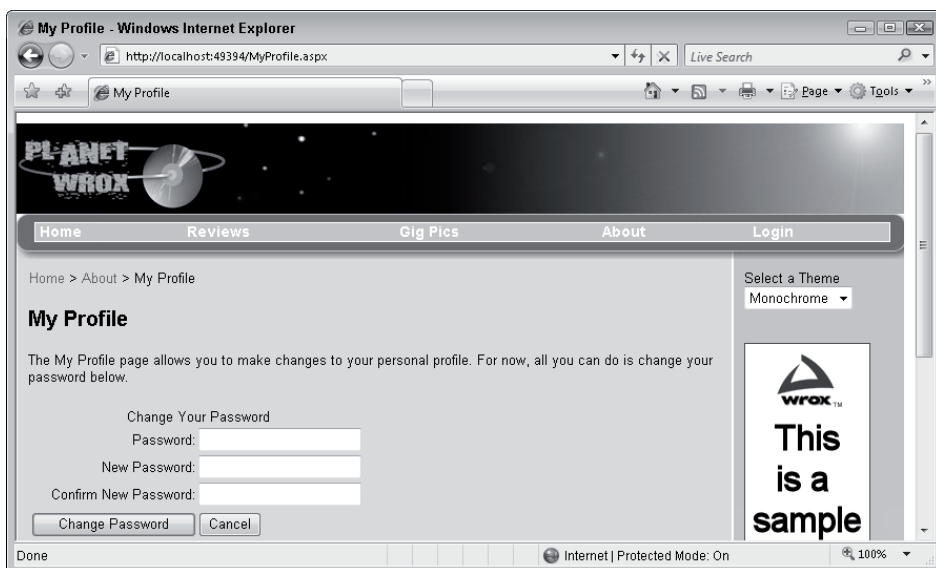
```
<siteMapNode url="~/About/Default.aspx" title="About" description="About this site">
  <siteMapNode url="~/About/Contact.aspx" title="Contact Us" description="Contact Us" />
  <siteMapNode url="~/About/AboutUs.aspx" title="About Us" description="About Us" />
  <siteMapNode url="~/MyProfile.aspx" title="My Profile" description="My Profile" />
</siteMapNode>
```

9. שמור את כל השינויים וסגור את כל הקבצים הפתוחים. בסייר הפתרון, לחץ לחיצה מינית על Login.aspx ובחר View in Browser. מתחת לפקד Login אתה אמור לראות כעת את הפקד PasswordRecovery, שנראה בתרשים 13-15.



תרשים 13-15

אם כבר היית מחובר, עליך ללחוץ תחילה על הקישור Logout.



תרשים 14-15

10. הזן את שם המשתמש שלך ולחץ על הלחצן Submit. תתבקש להזין את התשובה לשאלת האבטחה שסיפקת כשביקשת פתיחת חשבון. הזן את התשובה ולחץ Submit פעם נוספת. לאחר זמן מה אתה אמור לקבל הודעת דואר אלקטרוני עם הסיסמה החדשה שנוצרה באופן אוטומטי.
11. השתמש בסיסמה חדשה זו כדי להתחבר לאתר. כשתהיה מחובר, בחר בפריט My Profile מ-Menu או מ-TreeView. כעת יופיע הפקד ChangePassword שנראה בתרשים 14-15.
12. הזן את הסיסמה שנשלחה אליך בדואר האלקטרוני, הקלד סיסמה חדשה שקל יותר לזכור, ואז חזור והקלד אותה שוב. לבסוף, לחץ Change Password. מעתה והלאה תוכל להתחבר לאתר באמצעות הסיסמה החדשה.

כיצד זה פועל

כברירת מחדל, הסיסמאות מאוחסנות בבסיס הנתונים במצב hashed ופירוש הדבר שלא ניתן לאחזר אותן. **hashing** הוא תהליך בלתי-הפיך שיוצר טביעת-אצבעות ייחודית של הנתונים שלך. מכיוון שהתהליך בלתי-הפיך, לא ניתן ליצור את הסיסמה מחדש מתוך ה-hash, ולכן האחסון בבסיס הנתונים בטוח יותר. כשמשתמש מתחבר, הסיסמה המוזנת עוברת hashing והתוצר משווה עם הרשום בבסיס הנתונים כדי לראות אם המשתמש מורשה להיכנס לאתר. מכיוון שלא ניתן לאחזר את הסיסמה המקורית, הפקד PasswordRecovery יוצר סיסמה חדשה ושולח אותה לכתובת הדואר האלקטרוני הקשורה לשם המשתמש שהוזנת. בגוף ההודעה הפקד משתמש בתבנית הרגילה שמכילה את שם המשתמש ואת הסיסמה החדשה. כדי להתאים אישית את גוף ההודעה, באפשרותך להפנות את BodyFileName של MailDefinition לקובץ טקסט שמכיל שומרי מקום עבור שם המשתמש והסיסמה, בדיוק כפי שראית כיצד לעשות זאת עם CreateUserWizard.

ייתכן ששמת לב שפקדי ההתחברות משתמשים בכמה ברירות מחדל שעד כה לא יכולת לשנותן. לדוגמה, הפקדים CreateUserWizard ו-PasswordRecovery תמיד שואלים את המשתמש את שאלת האבטחה והתשובה כשכבה נוספת של אבטחה. המשתמש גם נדרש להזין סיסמאות קשיחות בעלות אורך של שבעה תווים לפחות. באפשרותך לשנות הגדרות אלו עבור היישום כולו תוך שימוש בקובץ web.config.

קביעת התצורה של יישום האינטרנט

מוקדם יותר בפרק עסקנו בשאלה כיצד ברירת המחדל של מחרוזת החיבור, הקרויה בשם LocalSqlServer, מוגדרת בקובץ machine.config באופן שהיא אפקטיבית כברירת מחדל עבור כל יישומי האינטרנט שאתה בונה. הדבר נכון גם לגבי הגדרות אחרות רבות שרלוונטיות לשירותי היישום של ASP.NET כמו חברות, תפקידים ופרופיל. הקובץ machine.config מכיל את ההגדרות הבאות עבור שירותי חברות (membership services):

```
<membership>
  <providers>
    <add name="AspNetSqlMembershipProvider"
      type="System.Web.Security.SqlMembershipProvider, Version=2.0.0.0,
        Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
      connectionStringName="LocalSqlServer"
      enablePasswordRetrieval="false"
      enablePasswordReset="true"
      requiresQuestionAndAnswer="true"
      applicationName="/"
      requiresUniqueEmail="false"
      passwordFormat="Hashed"
      maxInvalidPasswordAttempts="5"
      minRequiredPasswordLength="7"
      minRequiredNonalphanumericCharacters="1"
      passwordAttemptWindow="10"
      passwordStrengthRegularExpression="" />
  </providers>
</membership>
```

תצורת ספק (provider configuration) זו מציגה כמה תכונות מעניינות שמתוארות בטבלה הבאה.

מאפיין	תיאור
connectionStringName	מאפיין זה מצביע על מחרוזת החיבור של היישום. ברירת המחדל הוא החיבור LocalSqlServer שכבר פגשת בו.
enablePasswordRetrieval	מאפיין זה קובע אם משתמשים יכולים לאחזר את הסיסמה הנוכחית שלהם. לא ניתן להגדיר מאפיין זה כאשר passwordFormat מוגדרת כ-Hashed.

מאפיין	תיאור
enablePasswordReset	מאפיין זה קובע אם משתמש יכול לבקש סיסמה חדשה.
requiresQuestionAndAnswer	מאפיין זה קובע אם פקדים כמו CreateUserWizard ו-PasswordRecovery יכולים לדרוש מהמשתמש להזין שאלת אבטחה ולקבל תשובה עליה.
applicationName	תכונה זו מספקת את השם הייחודי ליישום.
requiresUniqueEmail	מאפיין זה קובע אם המערכת מאפשרת חשבונות משתמש בעלי כתובות דואר אלקטרוני כפולות. כשהוא מוגדר True, כל משתמש חייב לספק שם משתמש ייחודי וכתובת דואר אלקטרוני ייחודית.
passwordFormat	מאפיין זה קובע את הדרך שבה מאוחסנות סיסמאות בבסיס הנתונים. היא תומכת באפשרויות הבאות: Clear – הסיסמאות מאוחסנות כטקסט פשוט. Encrypted – הסיסמאות מוצפנות בפורמט הפיך שמאפשר למערכת לאחזר את ייצוג הסיסמה כטקסט פשוט. Hashed – סיסמאות מוצפנות בעזרת אלגוריתם חד-סטרי ובלתי-הפיך. כש-passwordFormat=Hashed מוגדר כ-Hashed, משתמשים אינם יכולים לאחזר את סיסמתם המקורית, הם יכולים רק לבקש סיסמה חדשה שתיווצר באופן אוטומטי.
maxInvalidPasswordAttempts	מאפיין זה מציין את מספר הפעמים שמשתמש יכול להזין סיסמה לא תקפה או שאלת אבטחה לא תקפה לפני שחשבונו יינעל.
minRequiredPasswordLength	מאפיין זה קובע את האורך המינימלי של מחרוזת הסיסמה.
minRequiredNonalphanumericCharacters	מאפיין זה קובע את המספר המינימלי של תווים שאינם אלפאנומריים שחייבים להיות בסיסמה.

מאפיין	תיאור
passwordAttemptWindow	מאפיין זה קובע את משך הזמן, בדקות, שבמהלכו נספרים נסיונות לא תקפים של הזנת סיסמה.
passwordStrengthRegularExpression	מאפיין זה מאפשר לציין ביטוי רגיל (regular expression) מותאם אישית כדי לכפות סיסמה קשיחה.

כדי להתאים הגדרות אלו ליישום Planet Wrox, תוכל לשנות את machine.config באופן ישיר. עם זאת, מומלץ בחום לא לעשות זאת. מכיוון שקובץ זה חל על המחשב כולו, אתה עלול לסיים עם תופעות-לוואי בלתי-רצויות רבות.

במקום זאת תוכל לעצב מחדש את האלמנט <membership> בקובץ web.config של האתר, ולשנות בו את התכונות הנחוצות עבור היישום הנוכחי בלבד. בתרגיל הבא תראה מה נדרש כדי לעצב מחדש את ספק החברות עבור יישום Planet Wrox.

נסה זאת עיצוב אלמנט חברות <membership>

בתרגיל קצר זה תלמד כיצד לדרוס את התנהגות ברירת המחדל של חברות באתר Planet Wrox. כשיהיה לך עותק של התצורה, תשנה בו את ההגדרות כדי להסיר את האפשרות של שאלת האבטחה והתשובה עליה, וכדי לשנות את כללי הסיסמה.

1. אתר את הקובץ machine.config. כברירת מחדל הוא נמצא ב-

C:\Windows\Microsoft.NET\Framework\v2.0.50727\CONFIG

אם התקנת את Windows בתיקייה אחרת או בכונן אחר, שנה את הנתיב בהתאם. אם מנהל המערכת מונע גישה לקובץ זה במחשב שלך, אל דאגה. תוכל להקליד את הקוד מצעד 6 אל קובץ web.config. שים לב גם לעובדה שהקובץ מאוחסן בתיקייה של גרסה 2.0 של סביבת העבודה .NET. רכיב runtime של סביבת העבודה של ASP.NET 3.5 הוא עדיין זה של גרסת 2.0, ולכן משתמשים בקבצים שבתיקיית התצורה של 2.0.

2. פתח את machine.config עם Notepad ואתר את האלמנט <membership> תחת <system.web>, בערך בשלושת רבעי הדרך אל סוף הקובץ.

3. העתק (Ctrl+C) את האלמנט <membership> בשלמותו אל ה-clipboard.

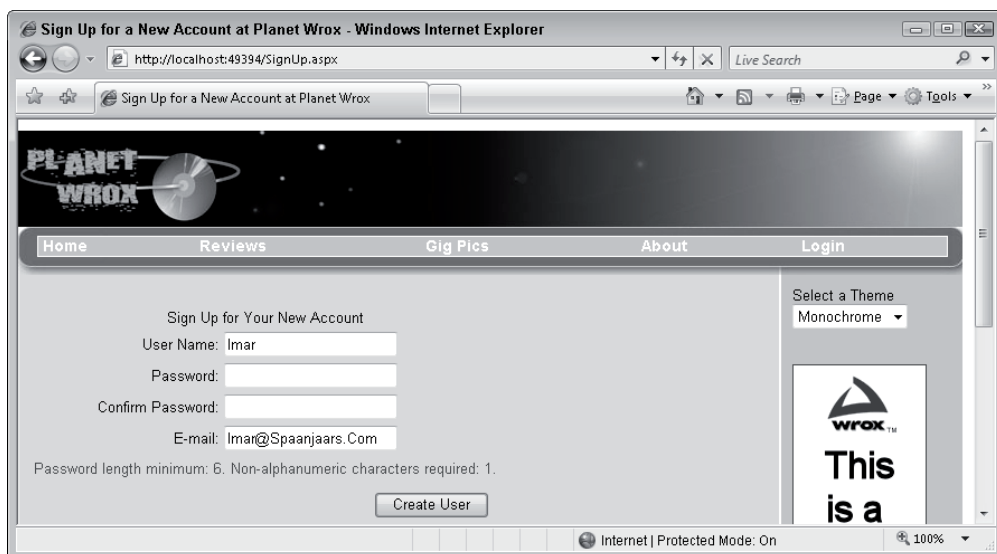
4. חזור ל-VWD, פתח את web.config והדבק את אלמנט membership מיידי לפני האלמנט <authentication>. פרט לכמה הבדלים בעיצוב (ייתכן שכל ההגדרות תהיינה בשורה אחת), אתה אמור לסיים עם הגדרות התצורה שראית בתחילת סעיף זה.

5. שנה את minRequiredPasswordLength ל-6 והגדר את requiresQuestionAndAnswer ל-true.

6. מייד לפני האלמנט <add> הוסף משפט <clear />. כשתסיים אמורות הגדרות התצורה להיראות כך:

```
<membership>
  <providers>
    <clear />
    <add name="AspNetSqlMembershipProvider"
      type="System.Web.Security.SqlMembershipProvider, System.Web,
        Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
      connectionStringName="LocalSqlServer"
      enablePasswordRetrieval="false"
      enablePasswordReset="true"
      requiresQuestionAndAnswer="false"
      applicationName="/"
      requiresUniqueEmail="false"
      passwordFormat="Hashed"
      maxInvalidPasswordAttempts="5"
      minRequiredPasswordLength="6"
      minRequiredNonalphanumericCharacters="1"
      passwordAttemptWindow="10"
      passwordStrengthRegularExpression="" />
  </providers>
</membership>
<authentication mode="Forms"/>
```

7. שמור את כל השינויים ובקש את SignUp.aspx בדפדפן. שים לב ששאלת האבטחה והתשובה אינן מופיעות עוד בטופס Sign Up כפי שהיה בתרשים 4-15 שבתחילת הפרק.



תרשים 15-15

8. מלא את הטופס, אבל בשדות הסיסמה הקלד טקסט קצר, למשל pass.
9. לחץ על הלחצן Create User. שים לב שהפקד כופה עליך כעת להזין סיסמה בת ששה תווים לפחות, כפי שהגדרת ב-web.config, על ידי הצגת הודעת השגיאה הנכונה מתחת לפקד, כפי שמוצג בתרשים 15-15.
10. הזן סיסמה בת ששה תווים לפחות, כולל תו אחד לפחות שאינו אלפאנומרי כמו # או *, ולחץ על הלחצן Create User פעם נוספת. הפעם הסיסמה מתקבלת והחשבון נוצר.

כיצד זה פועל

הגדרות התצורה של ASP.NET פועלות בצורה היררכית. כלומר, הגדרות שמוגדרות ברמה גבוהה (לדוגמה, ב-machine.config) חלות על כל יישומי האינטרנט במחשב. ללא כל שינויים, יורש אתר האינטרנט Planet Wrox את ההגדרות שב-machine.config, כולל אלו של ספק החברות. כדי לשנות את ההתנהגות באתר אינטרנט יחיד, עליך ליצור עותק של ההגדרות המקוריות ואז לשנות את התכונות הנדרשות.

עם זאת, כדי לבצע זאת עליך להסיר (remove) תחילה את ההגדרות המקוריות. תוכל לעשות זאת עם האלמנט:

```
<providers>
<clear />
<add name="AspNetSqlMembershipProvider" ...
```

כאן, משמעות הסימון `<clear />` היא: "הסר את הגדרות הספק שנוספו במקום כלשהו בהיררכית התצורה". לפיכך, מייד לאחר האלמנט `<clear />` נחשב האלמנט `<providers>` לנקי, והדבר מאפשר להוסיף שוב את אותו ספק, והפעם – עם הגדרות שונות.

למרות שהדברים נראים כעבודה מרובה לשינוי הגדרות ספורות בלבד, תמצא שהיכולת לדרוס את ההגדרות באתר שלך היא שימושית ביותר כשצריך שהיישום שלך ידבר עם מערכת בסיס נתונים אחרת. נספח ב' מראה כיצד לבצע זאת.

עד כה ראית כיצד לאפשר למשתמשים להירשם לקבלת חשבון כדי שיוכל להתחבר לאתר. אבל כיצד תוכל להבחין בין המשתמשים השונים במערכת? כיצד תוכל לחסום משתמשים בלתי-מורשים מלגשת לתיקיות מסוימות כמו Management? תעשה זאת עם מנהל התפקיד (role manager), שירות יישום נוסף שמגיע עם ASP.NET.

מנהל התפקיד (role manager)

זה נחמד שהמשתמשים יכולים כעת להירשם ולהתחבר לאתר, אבל זה יהיה הרבה יותר שימושי אם תוכל להבחין בין המשתמשים. הדבר יאפשר הענקת זכויות גישה לתיקייה Management למשתמש אחד בלבד, או לכמה משתמשים, באופן שרק הם יוכלו לשנות את הסקירות

והסגנונות. הדבר קל למדי לביצוע עם רכיב מנהל התפקיד שכלול ב ASP.NET. מנהל התפקיד מאפשר להקצות משתמשים לתפקידים שונים. תוכל אז להשתמש בתפקידים אלה כדי לפתוח, או לחסום, פונקציונליות מסוימת באתר. לדוגמה, תוכל לחסום את הגישה לתיקייה Management לכל המשתמשים פרט לבעלי תפקיד Managers. בנוסף, באמצעות LoginView תוכל להציג תכנים שונים על פי התפקידים שהמשתמשים ממלאים, כפי שראית קודם.

קביעת תצורת מנהל התפקיד

כמו במקרה של חברות (membership), ההגדרות עבור מנהל התפקיד נמצאות בקבצי תצורה (config files). עם זאת, מנהל התפקיד אינו מופעל כברירת מחדל. עליך להפעילו במפורש בקובץ web.config, באלמנט <roleManager> שאמור להיות מוצב בתוך האלמנט <system.web> ב-web.config.

```
<roleManager enabled="true" />
```

בתרגיל מאוחר יותר תראה כיצד לאפשר תפקידים על ידי כלי ניהול שנגיש מתוך VWD, כלומר, מבלי לשנות את הקובץ web.config באופן ישיר.

לאחר שאתה מאפשר תפקידים, עומדות לרשותך כמה דרכים להקצאת משתמשים לתפקידים שונים:

⊙ שימוש בכלי Web Site Administration Tool (או WSAT בראשי תיבות).

⊙ על ידי תכנות, תוך שימוש ב- Role Manager API.

ה-WSAT הוא בעל שימושים רבים הרבה יותר, והוא יידון בפירוט בסעיף הבא. ניהול תפקידים תוך שימוש ב- Role Manager API נמצא מעבר לתחום הספר. אם ברצונך לדעת יותר על תחום זה פנה לאחד המאמרים ברשת.

ניהול משתמשים עם WSAT

כלי התוכנה WSAT כלול בתוכנה VWD והוא זמין מתפריט Website. הכלי משמש למשימות אלו:

⊙ ניהול משתמשים.

⊙ ניהול תפקידים.

⊙ ניהול כללי גישה. לדוגמה, לקבוע איזה משתמש יכול לגשת לאיזה קבצים ותיקיות.

⊙ עיצוב הגדרות יישום, דואר אלקטרוני וניפוי.

⊙ העברת האתר למצב לא-מקוון, כדי שמשתמשים לא יוכלו לבקש דפים כלשהם ויקבלו במקומם הודעת שגיאה ידידותית.

ה-WSAT זמין רק במחשב המקומי כקיצור דרך של תפריט ב- Visual Web Developer, על כן הוא מעולה להגדרת המשתמשים והתפקידים הראשוניים במהלך הפיתוח, אבל אינו מתאים

לניהול משתמשים בסביבת ייצור. לניהול משתמשים בסביבת ייצור תוכל לפתח כלי תוכנה משלך תוך שימוש ב- Membership API וב- Role API, או לחפש במגוון הרחב של פתרונות את הקוד הפתוח שיתאים לצרכיך. שני הפתרונות הם מחוץ לתחום פרק זה.

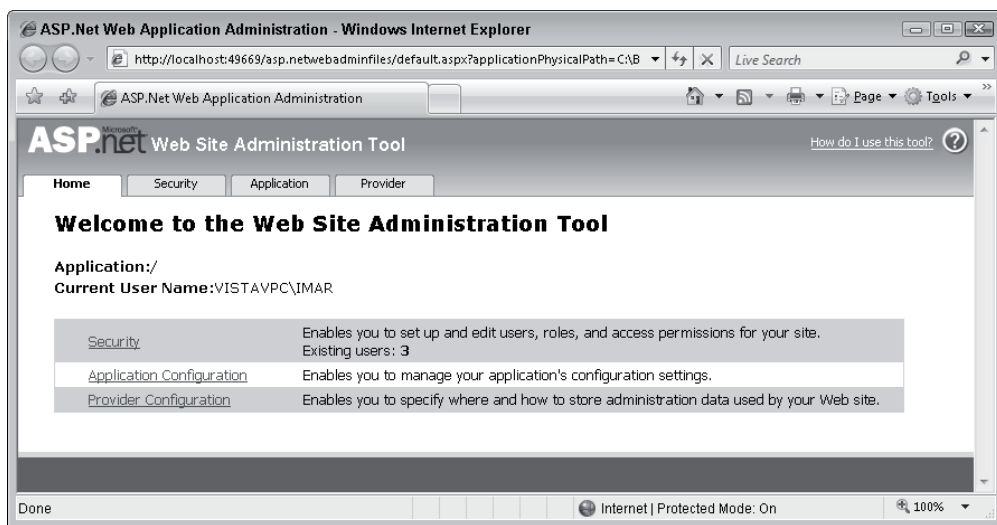
כמה מהשינויים שתבצע עם WSAT יתקבעו בקובץ web.config של היישום. הגדרות אחרות, כמו משתמשים ותפקידים, מאוחסנות בבסיס הנתונים של הספק.

בתרגיל הבא תלמד כיצד להתחיל להשתמש ב-WSAT. תלמד כיצד ליצור תפקיד חדש וחשבון משתמש חדש, וכיצד להקצות משתמש זה לתפקיד.

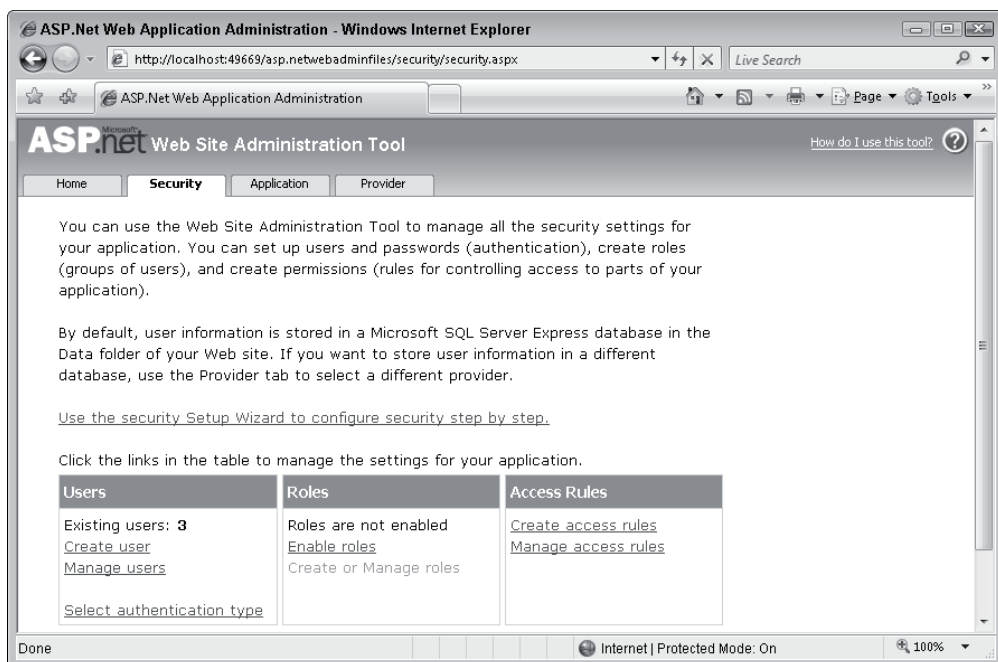
נסה זאת שימוש ב-WSAT לניהול חשבונות ותפקידי משתמשים

כדי להגן על התיקייה Management מפני משתמשים שאינם מורשים לגשת אליה, צריך ליצור תפקיד שמורשה לגשת לתיקייה זו. לאחר הגדרת התפקיד, תוכל להעניק לכל המשתמשים בתפקיד זה גישה לתיקייה, ולמנוע זאת מכל המשתמשים האחרים. בתרגיל זה תלמד כיצד ליצור את התפקיד Managers ולהקצות לו משתמש. בתרגיל אחר תלמד כיצד להגביל את הגישה לתיקייה Management לבעלי תפקיד Managers בלבד.

1. מתוך VWD, בחר Website Configuration ← ASP.NET Configuration. הדפדפן יפתח ויצגי את Web Site Administration Tool, שנראה בתרשים 15-16.



תרשים 15-16



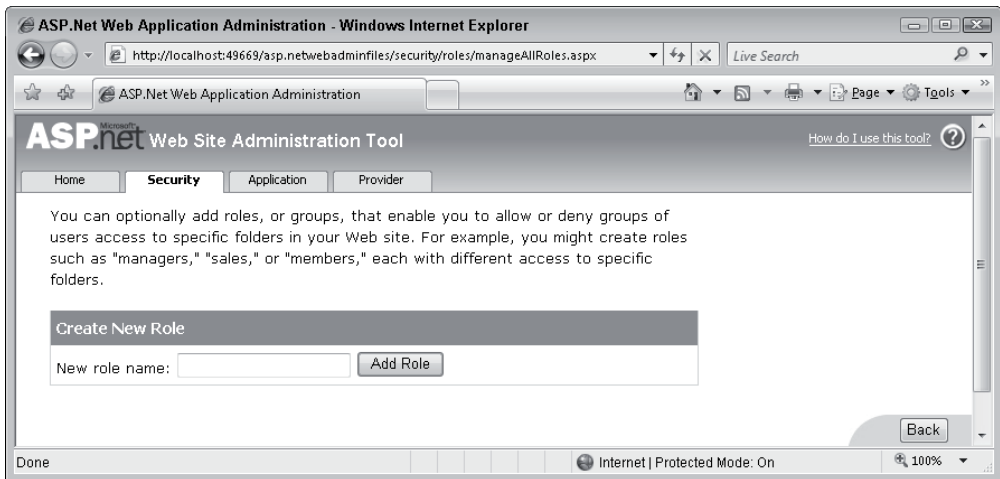
תרשים 15-17

2. בפניה הימנית העליונה תראה קישור Help שיעביר אותך לקובץ עזרה, שמתאר כיצד להשתמש בכלי. מייד תחת הלוגו של היישום מוצגות ארבע כרטיסיות: Home, Security, Application, ו-Provider. הכרטיסייה Home מחזירה אותך לדף הבית שמוצג בתרשים 15-16. הכרטיסייה Application משמשת לעיצוב הגדרות יישום שונות, בעוד הכרטיסייה Provider מאפשרת לעצב מחדש את הספק הנבחר עבור היישום. בתרגיל זה, חשובה לענייננו הכרטיסייה Security, ולכן לחץ עליה. כעת תראה את המסך שמוצג בתרשים 15-17.

3. האזור התחתון של המסך מחולק לשלוש עמודות: Users, Roles, ו-Access Rules. בתרגיל זה תלמד להשתמש ב-Users ו-Roles. פעולות Access Rules משמשות לחסימת או לפתיחת חלקים מסוימים באתר האינטרנט למשתמשים או בעלי תפקידים. בפרק זה לא תלמד כיצד להשתמש בפעולות אלו, אבל במקום זאת תלמד בתרגיל מאוחר יותר כיצד לשנות כמה מהגדרות אלו ישירות ב-`web.config`.

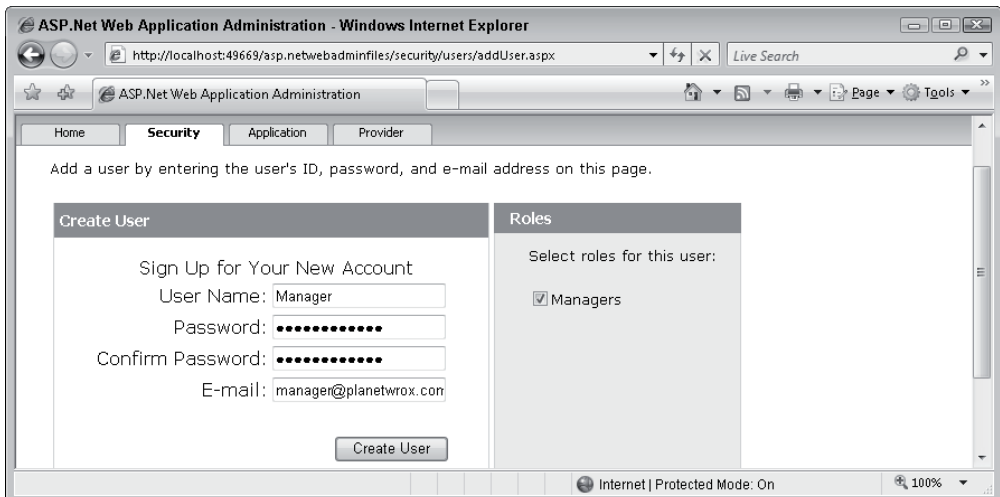
4. ודא שתחת העמודה Users מופיעים הקישורים Create user ו-Manage users. אם אינך רואה אותם, ובמקום זאת מופיעה הערה על האימות של Windows, לחץ על הקישור Select authentication type, בחר From the internet ולבסוף לחץ Done. המסך שלך אמור להיראות כעת כמו בתרשים 15-17.

5. בעמודה Roles, לחץ על הקישור Enable roles. המסך נטען מחדש ומציע כעת קישור עם הטקסט Create or Manage Roles. לחץ על קישור זה כדי לפתוח את החלון New Role שנראה בתרשים 15-18.

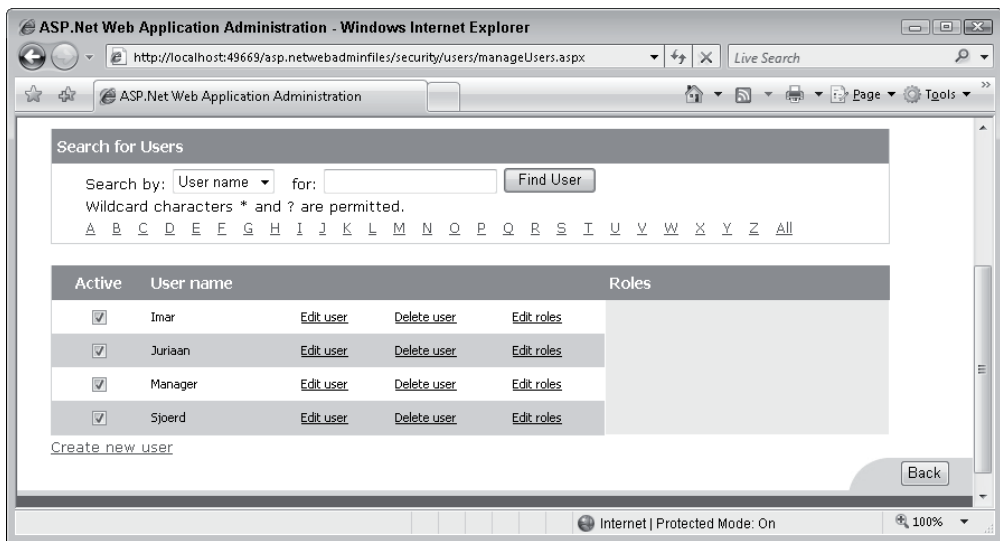


תרשים 15-18

6. הכנס Managers כשם התפקיד החדש ולחץ על הלחצן Add Role. התפקיד החדש אמור להופיע. לחץ על הלחצן Back שבתחתית הדף כדי לחזור לדף הראשי של Security.
7. לחץ על הקישור Create User. כעת תועבר לדף שמאפשר להזין את הפרטים עבור משתמש חדש, ובו-בזמן להקצות משתמש זה לתפקיד Managers. מלא את הפרטים כמו בתרשים 15-19. כסיסמה, הזן מחרוזת טקסט שעונה על כללי הסיסמה שקבעת קודם. סיסמה כמו Manager##123 תתקבל. באפשרותך לבחור בכתובת הדואר האלקטרוני שלך במקום החשבון של PlanetWrox.com. בנוסף, זכור לסמן את שם התפקיד Managers ברשימת התפקידים שמימין.



תרשים 15-19



תרשים 15-20

8. לחץ על Create User כדי להוסיף את המשתמש למערכת, ואז לחץ Continue בדף האישור. לחץ על הלחצן Back שבתחתית הדף עד שתגיע לדף הראשי של Security.
9. בדף Security, לחץ על הקישור Manage users. כעת תועבר לדף שמציג רשימה של כל המשתמשים הזמינים במערכת. ראה תרשים 15-20. מכאן תוכל לערוך, להפעיל, להשבית, או למחוק משתמשים קיימים.
- לדוגמה, אם הגדרת קודם ב- CreateUserWizard את DisableCreatedUser כ-True, תוכל לאפשר כאן את המשתמש על ידי סימון תיבת הסימון שבחזית שם המשתמש, כפי שנראה בתרשים 15-20. תוכל לשנות את התפקידים המוקצים למשתמש על ידי לחיצה על הקישור Edit roles. בנוסף, במקרה שתעסוק במספר רב של חשבונות משתמשים, תוכל לחפש בזריזות אחר משתמש מסוים על ידי שימוש בפקדי הסינון ובאלפון שמעל רשימת המשתמשים.
10. כדי לראות את התמונה הסופית של משתמשים ותפקידים, סגור את הדפדפן וחזור ל-VWD. בסיר הפתרון, לחץ לחיצה כפולה על בסיס הנתונים ASPNETDB.MDF בתיקייה App_Data, כדי לפתוח אותו בחלון סיר בסיס הנתונים. הרחב את הצומת Tables, לחץ לחיצה ימנית על הטבלה aspnet_Roles, ובחר Show Table Data. התפקיד שיצרת בצעד 6 חייב להופיע ברשימה. פתח כמה מהטבלאות האחרות כמו aspnet_Membership ו-aspnet_UsersInRoles ובדוק את הנתונים שהן מכילות. בטבלה הראשונה תראה את חשבון המשתמש שיצרת בצעדים 7 ו-8. הטבלה השנייה מכילה קשר בין התפקיד החדש לבין חשבון משתמש.

כמו פקדי ההתחברות, כך גם WAST מדבר בסופו של דבר עם הספק (provider), שאת תצורתו קבעת בקובץ web.config. במקרה של יישום Planet Wrox, פירוש הדבר שה-WAST מדבר עם `AspNetSqlMembershipProvider` שבתורו מדבר עם בסיס הנתונים `SQL Server Express` שמזוהה על ידי מחרוזת החיבור `LocalSqlServer`. המשתמשים והתפקידים שאתה יוצר מאוחסנים בבסיס הנתונים בטבלאות `aspnet*` השונות. משתמשים שנוצרים תוך שימוש ב-WAST מסיימים בדיוק באותו המקום כמו אלה שנוצרים בעזרת הפקד `CreateUserWizard`. למעשה, WAST משתמש ב-`CreateUserWizard` כדי לאפשר לך ליצור חשבון משתמש חדש. פירוש הדבר שכל משתמש שאתה מכניס ב-WAST מסוגל להתחבר באמצעות דף `Login.aspx` רגיל. בתרגיל מאוחר יותר תשתמש בחשבון `Manager` שיוצרת בתרגיל זה, כדי להתחבר לאתר ולגשת לתיקייה `Management`.

כדי להשתמש בתפקיד שיוצרת בתרגיל זה, עומדות בפניך כמה אפשרויות. ראשית, תוכל להשתמש בשם התפקיד כדי לחסום גישה לתיקיות מסוימות של יישום האינטרנט על ידי הגדרות בקובץ web.config. שנית, תוכל להשתמש בתפקיד בפקדים כמו `LoginView` כדי להציג תוכן שונה למשתמשים שונים. לבסוף, תוכל להשתמש ב-`Role API` כדי לבדוק אם המשתמש הנוכחי הוא בעל תפקיד מסוים. הדבר נותן שליטה טובה על התוכן או על הפונקציונליות שתוכל להציע למשתמשים מסוימים בעלי זכויות מיוחדות. כעת תראה כיצד לחסום את הגישה לתיקייה `Management` ולשנות את `LoginView`. בתרגיל מאוחר יותר נדון בשימוש ב-`Role API`.

עיצוב יישום אינטרנט כדי שיפעל עם תפקידים

בדף `Security` של WAST ראית קטע בשם `Access Rules`. חלק זה של הכלי מאפשר לחסום או לפתוח משאבים באתר. הוא מאפשר לקבוע כללים כמו "תיקייה זו חסומה לכל אחד פרט למשתמשים בתפקיד `Managers`", או, "כל אחד יכול לגשת לקובץ זה פרט לבעלי תפקיד `Members` ולחשבון `Joe`". השימוש בכלי אינטואיטיבי למדי, ולפיכך מאפשר לקבוע בקלות את הכללים השונים. עם זאת, יש לו מגרעה אחת: הוא מאחסן את הגדרות האבטחה בקבצי `web.config` שונים, אחד לכל תת-תיקייה שעיצבת.

דבר זה מקשה על קבלת תמונה כוללת של הגדרות האבטחה השונות. למרבה המזל, `ASP.NET` מאפשרת גם לעצב את ההגדרות ב-`web.config` הראשי, על ידי שימוש באלמנטים מסוג `<location>`. לאלמנט `<location>` יש מאפיין נתיב שמצביע על הקובץ או על התיקייה שברצונך לעצב באופן שונה וייחודי. תוכל להשתמש באלמנט `<location>` גם עבור הגדרות אחרות ב-`web.config` (לדוגמה, תוכל להגדיר ב-`web.config` הראשי את המאפיין `theme` של האלמנט `<pages>` עבור התיקייה `Management`). בתרגיל הבא עליך לקבוע רק את האלמנטים הבנים של `<location>` עבור ההגדרות שנוגעות לאבטחה.

ללא ספק, אינך רוצה שכל גולש לאתר ישבש את הסקירות והסגנונות שפורסמו באתר. לכן חשוב למנוע גישה לתיקייה Management מכל מי שאינו נמנה על Managers. בתרגיל זה תראה כיצד לשנות את web.config, כדי לחסום את התיקייה באופן שרק משתמשים שהוקצו קודם לתפקיד Managers יוכלו לגשת לתיקייה זו ולקבצים שהיא מכילה.

1. פתח את הקובץ web.config משורש האתר. גלול מטה עד לתגית הסגירה `</configuration>`. מייד לפני הקלד אלמנט `<location>`. הוסף לאלמנט מאפיין path וקבע את ערכו ל-Management. שים לב ש-IntelliSense מתפרץ כדי לעזור לך להשלים את האלמנט. הקוד שלך אמור להיראות כעת כך:

```
</mailSettings>
</system.net>
<location path="Management"></location>
</configuration>
```

2. השלם את התצורה על ידי הכנסת הגדרות XML הבאות:

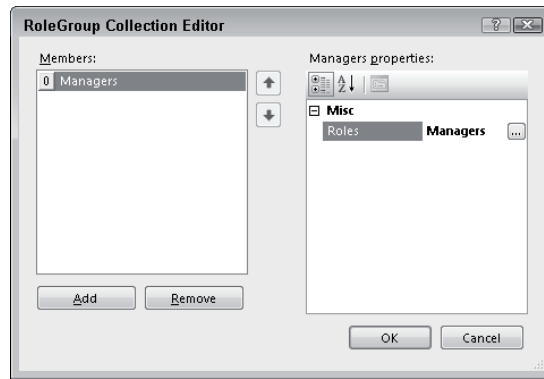
```
</system.net>
<location path="Management">
  <system.web>
    <authorization>
      <allow roles="Managers"/>
      <deny users="*/">
    </authorization>
  </system.web>
</location>
</configuration>
```

3. שמור וסגור את הקובץ web.config.

4. ב-Design View, פתח את הדף הראשי של האתר (MasterPage.master) וגלול מטה עד לסוף הקובץ. בחר בפקד LoginView ופתח את לוח Smart Tasks שלו. בראש הלוח, לחץ על הקישור Edit RoleGroups, שמוצג בתרשים 15-21.



תרשים 15-21



תרשים 15-22

5. בתיבת הדו-שיח שנפתחת, לחץ על הלחצן Add כדי להכניס RoleGroup חדש. הגדר את המאפיין Roles של קבוצה זו ל-Managers, כפי שמוצג בתרשים 15-22.

6. לחץ OK כדי להכניס את RoleGroup וחזור ל- Design View.

7. כשאתה נמצא עדיין בלוח Smart Tasks של LoginView, בחר מהרשימה הנפתחת Views את RoleGroup[0] – Managers. הדבר מעביר את התבנית הנוכחית של הפקד אל RoleGroup עבור Managers, כדי שתוכל להוסיף תוכן שייראה על ידי Managers בלבד.

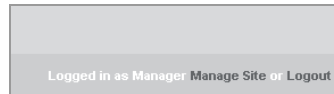
8. גרור פקד HyperLink מארגז הכלים ושחרר אותו לתוך LoginView. השתמש בטבלת המאפיינים כדי לקבוע את המאפיין Text של HyperLink לערך Manage Site, ואת NavigateUrl למצב ~/Management/Default.aspx (תוכל להשתמש ב- URL picker של HyperLink על ידי לחיצה על הלחצן הקטן עם האליפסות עליו). עבור ל- Markup View, ואחרי הפקד HyperLink הקלד את המילה or כשאחריה הפקד LoginStatus. את האחרון תוכל לגרור מערכת הכלים (Toolkit), או להעתיקו מהקוד הקיים ב- LoggedInTemplate.

ה-LoginView צריך להכיל את הקוד הבא:

```
<asp:LoginView ID="LoginView1" runat="server">
  <RoleGroups>
    <asp:RoleGroup Roles="Managers">
      <ContentTemplate>
        <asp:HyperLink ID="HyperLink1" runat="server"
          NavigateUrl="~/Management/Default.aspx">Manage Site</asp:HyperLink> or
        <asp:LoginStatus ID="LoginStatus1" runat="server" />
      </ContentTemplate>
    </asp:RoleGroup>
  </RoleGroups>
  <LoggedInTemplate>
    (<asp:LoginStatus ID="LoginStatus1" runat="server" />)
  </LoggedInTemplate>
</asp:LoginView>
```

9. שמור את כל השינויים, ואז בקש בדפדפן את דף הבית של האתר (Default.aspx). אמת שאינך מחובר כרגע (בדוק את הכותרת התחתונה של הדף, ואם נחוץ לחץ על הקישור Logout).

10. לחץ על הקישור Login ב-Menu או ב-TreeView, ואז התחבר עם החשבון Manager שיצרת מוקדם יותר בפרק. ודא שלא סימנת את האפשרות Remember Me. הדף מתרענן ומציג כעת בכותרת התחתונה של כל דף את הקישור Manage Site (ראה תרשים 15-23).



תרשים 15-23

11. לחץ על הקישור Manage Site כדי לפתוח את הקטע Management של אתר האינטרנט. העתק את ה-URL הנוכחי של הדף מסרגל הכתובת של הדפדפן אל ה-clipboard. לחץ על הלחצן Back של הדפדפן כדי לחזור לדף הבית, ואז לחץ על הלחצן Logout בכותרת התחתונה. סגור את הדפדפן ופתח מופע חדש שלו (תוכל לעשות זאת מתפריט Start או משולחן העבודה, אם יש שם קיצור דרך, או שתלחץ לחיצה ימנית על דף ב-VWD ותבחר View in Browser).

12. הדבק את הכתובת שזה עתה העתקת בסרגל הכתובת של חלון הדפדפן והקש Enter. במקום לעבור לכתובת כמו:

<http://localhost:49666/Management/Default.aspx>

תועבר לדף ההתחברות, ה-Login:

<http://localhost:49666/login.aspx?ReturnUrl=%2fManagement%2fDefault.aspx>

שים לב שהדף שביקשת תחילה (Management/Default.aspx) מצורף כעת למחרוזת השאילתה. האלכסונים הקדמיים (/) שבכתובת הוחלפו באופן אוטומטי למקבילים הבטוחים שלהם %2f, שנהוגים ב-URL. התחבר עם החשבון Manager וכעת תראה שהקטע Management מופיע שוב.

כיצד זה פועל

כדאי להתבונן בכמה נושאים מעניינים בתרגיל זה. הבט תחילה בהגדרות שהוספת לקובץ web.config כדי להגביל גישה לתיקייה Management:

```
<location path="Management">
  <system.web>
    <authorization>
      <allow roles="Managers"/>
      <deny users="*" />
    </authorization>
  </system.web>
</location>
```


כאשר ASP.NET runtime מעבד בקשה לדרך, הוא בודק את קבצי התצורה השונים כדי לראות אם המשתמש הנוכחי מורשה לגשת למשאב זה. עבור בקשות לקבצים בתיקייה Management הוא פוגש בקבוצת הכללים שהוגדרה באלמנט <location>. רכיב runtime מתחיל לסרוק את הכללים השונים (האלמנטים allow ו-deny בעלי התכונות roles ו-users לציון המשתמשים והתפקידים שמושפעים על ידי כלל זה), וברגע שהוא מוצא כלל הוא עוצר את תהליך הסריקה. אם לא סופקו כללים כלשהם, מוענקת הרשאת גישה. לכן, חשוב לסיים את קבוצת הכללים עם כלל דחייה שיחסום את כל המשתמשים האחרים שלא אושרה להם גישה קודם לכן.

כשמתחבר משתמש שאינו מאומת, הכלל הראשון אינו תואם, מכיוון שהמשתמש האנונימי אינו חבר בתפקיד Managers. לפיכך המשתמש לא מקבל גישה בגלל כלל הדחייה שחוסם את כל המשתמשים (מצוין על ידי כוכבית [*]).

לאחר חיבור כ-Manager ובקשת אותו המשאב שוב, מערכת הכללים נסרקת שוב. רכיב runtime מוצא את האלמנט allow שמעניק גישה לתפקיד Managers ומייד מאפשר את הכניסה. הכלל האחרון, שחוסם גישה לכל שאר המשתמשים, אפילו אינו נבדק כי אחד הכללים שלפניו התאים.

בנוסף לתפקידים המיוחדים, שמות המשתמשים, והכוכבית (*) לפנייה אל כלל המשתמשים, תוכל להשתמש גם בסימן השאלה (?) כדי לפנות למשתמשים שאינם מאומתים, או אנונימים. לדוגמה, כדי להגדיר גישה לתיקייה Reviews לכל משתמש מחובר, מבלי להתחשב בתפקיד שלו, ולחסום את הגישה לכל שאר המשתמשים, תוכל להוסיף את האלמנט <location> הבא לקובץ התצורה שלך:

```
<location path="Reviews">
  <system.web>
    <authorization>
      <deny users="?" />
    </authorization>
  </system.web>
</location>
```

זה מונע גישה מכל המשתמשים שאינם מחוברים. בגלל כלל ברירת המחדל, שמעניק גישה למשאב אם המשתמש הנוכחי אינו תואם כלל קודם, יכולים כל המשתמשים המחוברים לגשת לקבצים בתיקייה Reviews.

באלמנטים של תפקידים ומשתמשים אפשר לציין תפקידים רבים ושמות משתתפים רבים על ידי הפרדתם בפסיק זה מזה.

חשוב להבין כיצד פועל האלמנט RoleGroups של LoginView. למרות שביכולתך לציין מספר רב של אלמנטים RoleGroup שכולם יכולים להיות מוחלים על משתמש יחיד, הרי שרק הראשון שתואם יוצג. חשוב על משתמש בשם Marcel שמוקצה לקבוצה WebMasters ולקבוצה Managers, ועל דף אינטרנט עם LoginView הבא:

```
<asp:LoginView ID="LoginView1" runat="server">
  <RoleGroups>
```

```

<asp:RoleGroup Roles="Managers">
  <ContentTemplate>
    <!-- Content for Managers here -->
  </ContentTemplate>
</asp:RoleGroup>
<asp:RoleGroup Roles="WebMasters">
  <ContentTemplate>
    <!-- Content for WebMasters here -->
  </ContentTemplate>
</asp:RoleGroup>
</RoleGroups>

```

על פי קוד זה, המשתמש Marcel יראה רק את התוכן של RoleGroup הראשון, למרות שהוא מוקצה גם לקבוצה WebMasters.

בדיקת תפקידים על ידי תכנות

למרות שקל להשתמש בפקד LoginView כדי לשנות בזמן ביצוע את התכנים שמשתמש מורשה לראות, לא תמיד זה מספיק. יש מצבים שבהם נדרשת שליטה בתכנות על הנתונים המוצגים, על פי ההשתייכות של מישור לתפקיד מסוים. תוכל לגשת למידע אודות תפקידיו של המשתמש הנוכחי בכמה דרכים. תחילה, תוכל לגשת לשיטה IsInRole של המאפיין User מהדף הנוכחי או מפקד משתמש בדרך זו:

VB.NET

```

If User.IsInRole("Managers") Then
  ' This code runs for Managers only
End If

```

C#

```

if (User.IsInRole("Managers"))
{
  // This code runs for Managers only
}

```

לחילופין, תוכל לגשת למחלקה Roles שמכילה כמה שיטות סטטיות שתוכל לגשת אליהן באופן ישיר. הקוד הבא שקול מבחינה פונקציונלית לדוגמה הקודמת:

VB.NET

```

If Roles.IsUserInRole("Managers") Then
  ' This code runs for Managers only
End If

```

C#

```
if (Roles.IsUserInRole("Managers"))
{
    // This code runs for Managers only
}
```

ההבדל הגדול בין השניים הוא, שכדי שקטע הקוד האחרון יפעל, Role Manager חייב להיות מאופשר. מכיוון שזה המקרה באתר האינטרנט של Planet Wrox, תוכל לבחור בכל אחת משתי השיטות. עבור פתרונות מותאמים אישית, כשאיןך משתמש בספק תפקידים אלא בפתרון מותאם משלך, תוכל להשתמש במבנה הקוד הראשון בלבד.

בנוסף לשיטה IsUserInRole, המחלקה Roles מכילה שיטות רבות נוספות שמאפשרות לפעול עם תפקידים בתכנות. לדוגמה, ביכולתך ליצור ולמחוק תפקידים, להקצות משתמשים לתפקידים ולהסירם מתפקידים, ולקבל רשימת משתמשים שמוקצים לתפקיד מסוים. למידע נוסף על Roles API פנה לתיעוד MSDN.

בתרגיל הבא תלמד כיצד לשנות את דף אלבומי התמונות, באופן שמשתמשים שמחזורים כ-Managers יוכלו למחוק תמונות מאלבום תמונות. משתמשים אחרים לא יוכלו למחוק תמונה מכיוון שהלחצן Delete יהיה מוסתר מהם.

נסה זאת בדיקת תפקידים עם IsUserInRole בזמן ביצוע

למרות שניתן ליצור דוגמה זו תוך שימוש ב-LoginView עם תבניות שונות ו-RoleGroups, התרגיל הבא משתמש בבדיקה בתכנות של תפקיד המשתמש, כדי לקבוע אם להציג או להסתיר את הלחצן Delete. הדבר יקל על שינוי הקוד מאוחר יותר, באופן שיאפשר גם לבעלי אלבום תמונות מסוים למחוק את התמונות שלהם עצמם.

1. פתח את Default.aspx מהתיקייה PhotoAlbums ב-Design View ואתר את הפקד LinqDataSource2 שמשמש לניהול התמונות באלבום תמונות. פתח את לוח Smart Tasks שלו ואפשר מחיקה על ידי בחירת הפריט Enable Delete.

2. עבור ל-Markup View ואתר את האלמנט <ItemTemplate> של ListView שמשמש להצגת התמונות.

3. בין הפקד Label לבין תגית הסגירה הוסף Button, אשר המאפיינים CommandName ו-Text שלו מוגדרים כ-Delete. הגדר את ID של הלחצן ל-DeleteButton. בסופו של דבר, הלחצן צריך להיראות כך:

```
<asp:Label ID="DescriptionLabel" runat="server" Text='<# Eval("Description") %>' />
<asp:Button ID="DeleteButton" CommandName="Delete" runat="server" Text="Delete" />
</li>
```

4. עבור שוב ל- Design View. בחר ListView ופתח את טבלת המאפיינים שלו. עבור לקטגוריה Event ולחץ לחיצה ימנית על האירוע ItemCreated כדי ליצור עבורו מטפל אירוע.
5. הוסף את הקוד הבא למטפל האירוע שהוסיפה Visual Web Developer לקוד ברקע של הקובץ.

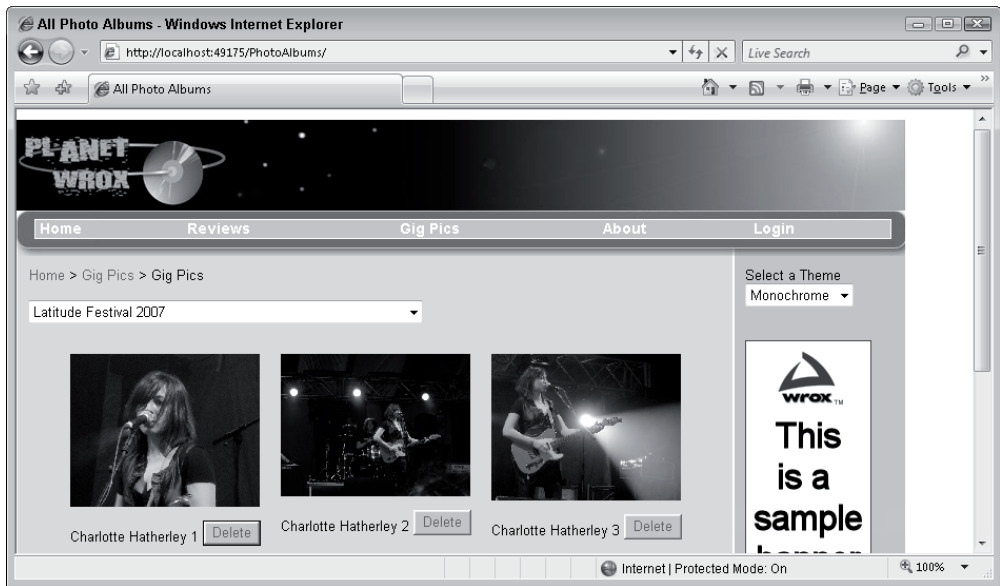
VB.NET

```
Protected Sub ListView1_ItemCreated(ByVal sender As Object, _
    ByVal e As System.Web.UI.WebControls.ListViewItemEventArgs) _
    Handles ListView1.ItemCreated
    Select Case e.Item.ItemType
        Case ListViewItemType.DataItem
            Dim deleteButton As Button = CType(e.Item.FindControl("DeleteButton"), Button)
            deleteButton.Visible = Roles.IsUserInRole("Managers")
    End Select
End Sub
```

C#

```
protected void ListView1_ItemCreated(object sender, ListViewItemEventArgs e)
{
    switch (e.Item.ItemType)
    {
        case ListViewItemType.DataItem:
            Button deleteButton = (Button)e.Item.FindControl("DeleteButton");
            deleteButton.Visible = Roles.IsUserInRole("Managers");
            break;
    }
}
```

6. שמור את השינויים ובקש את הדף בדפדפן על ידי הקשת Ctrl+F5. אם אתה עדיין מחובר, לחץ תחילה על הקישור Logout שנמצא בכותרת התחתונה בתחתית הדף.
7. מהרשימה הנפתחת בחר באחד מאלבומי התמונות. הדף נטען מחדש ומציג את התמונות שבאלבום התמונות. מכיוון שאינך מחובר, הקישור Delete אינו נראה. לחץ על הקישור Login ב-Menu או ב-TreeView, והתחבר עם החשבון Manager שיצרת מוקדם יותר בפרק זה.
8. עבור לדף Photo Albums הראשי (בחר Gig Pics בתפריט) ובחר באלבום תמונות מהרשימה שמוצגת. תרשים 15-24 מראה כיצד כל תמונה קשורה כעת עם לחצן מחיקה שימחק את התמונה כשיילחץ.
9. לחץ שוב על הקישור Logout שבכותרת התחתונה ועבור לדף Login. התחבר על פי החשבון הראשון שיצרת בפרק, ואשר אינו חבר בתפקיד Managers. חזור לדף Gig Pics ובחר באלבום תמונות מתוך הרשימה. גם אם אתה מחובר עם חשבון תקף, עדיין לא תוכל למחוק תמונות מכיוון שאין לך הרשאת Managers.



תרשים 15-24

כיצד זה פועל

רוב הקוד בתרגיל זה אינו אמור להיות חדש מבחינתך. ראית כיצד למחוק פריטים על ידי הפקדים `ListView` ו-`LinqDataSource`. למדת גם כיצד לטפל באירוע `ItemCreated` ולחפש אחר פקדים בפריט תוך שימוש ב-`FindControl`.

החידוש בדוגמה זו היא הדרך שבה אתה בודק אם המשתמש הנוכחי חבר בקבוצה `Managers`.

VB.NET

```
Dim deleteButton As Button = CType(e.Item.FindControl("DeleteButton"), Button)
deleteButton.Visible = Roles.IsUserInRole("Managers")
```

C#

```
Button deleteButton = (Button)e.Item.FindControl("DeleteButton");
deleteButton.Visible = Roles.IsUserInRole("Managers");
```

השיטה `IsUserInRole` מחזירה ערך בוליאני שמציין אם המשתמש הנוכחי הוא מנהל. כשהשיטה מחזירה `True`, פירוש הדבר שהמאפיין `Visible` של `Button` מוגדר ל-`True`. כשהשיטה מחזירה `False`, הלחצן מוסתר, והמשתמש אינו יכול למחוק תמונות מאלבום התמונות.

עצות מעשיות בנושא אבטחה

הרשימה הבאה מספקת כמה עצות מעשיות בנושא אבטחה:

- ⊙ למרות שמושג האבטחה מוצג בשלב די מאוחר בפרק זה, אל תראה בזה כל כוונה מיוחדת. כדי להבטיח שתיצור יישום בטוח ויציב, עליך לחשוב על אבטחה החל מהשלבים הראשוניים של פיתוח אתר האינטרנט. ההחלטות אם לקבוע אזורים שיהיו נגישים למשתמשים מסוימים בלבד, ואם לכפות על המשתמשים לפתוח חשבון באתר לפני שהם מקבלים גישה אליו – ראוי שתתקבלנה מוקדם ככל האפשר. ככל שתאחר בהכנסת מושגים אלה לתהליך, כך תעמוד בפני קשיים גדולים יותר כשתרצה לשלב את הפונקציונליות הזו.
- ⊙ נסה לקבץ משאבים כמו דפי ASPX למשל, בתיקיות שמייצגות תפקידים במערכת. קח לדוגמה, את התיקייה Management באתר האינטרנט Planet Wrox. כל הדפים הקשורים לניהול האתר ארוזים יחד בתיקייה אחת. הדבר מקל מאוד על חסימת התיקייה כולה על ידי אלמנט <location> יחיד בקובץ web.config. כשהקבצים שתוצאה להגן עליהם מפוזרים בכל רחבי האתר, תזדקק לזמן רב יותר לקביעת תצורת היישום, ובסופו של דבר לא תהיה לך תמונה ברורה של הגדרות האבטחה הפעילות.
- ⊙ כשיוצרים תפקידים כדי להבדיל בין משתמשים שונים באתר, נסה להגביל את מספר התפקידים השונים במערכת. תמצא שהמערכת הרבה יותר קלה לניהול עם קומץ תפקידים אשר מקובצים לוגית, מאשר עם מספר גדול של תפקידים שלכל אחד מהם יש משתמשים בודדים בלבד.

סיכום

יש טכניקות שונות ליישם אבטחה באתר מבוסס ASP.NET, ובכלל זה אימות של Windows (שבו שרת web הוא זה שדואג לאימות), או אימות של צד שלישי (שבו שירות חיצוני, כמו Passport של מיקרוסופט, דואג לבדיקת תקפות המשתמש). לבסוף, אפשר להשתמש באימות טפסים, שכיום הוא התקן למעשה של רבים מאתרי האינטרנט מבוססי ASP.NET.

האימות (authentication) מקיף שלושה מושגים חשובים: זהות, אימות והרשאה. זהות (identity) מתארת מי אתה ומייצגת אותך באתר. תהליך אימות (authentication) חושף את ההוכחה שמאשרת שזהותך תקפה (valid). באתרי אינטרנט מבוססי ASP.NET נשלחת הוכחה זו בצורת שם משתמש וסיסמה שתוקפם נבדק מול מחסן נתוני משתמש. **הרשאה (authorization)** עוסקת בדברים שאתה מורשה לעשות בתוך המערכת. לדוגמה, כמשתמש אנונימי אינך מורשה לגשת לקבצים בתיקייה Management של האתר. כשאתה מתחבר עם חשבון שיש לו הרשאות מתאימות, תוכל לדפדף בקבצים אלה בהצלחה.

ספק החברות (membership provider) מאפשר ליצור ולנהל משתמשים בבסיס נתונים מרכזי. משתמשים יכולים ליצור את החשבונות שלהם על ידי פקדים כמו CreateUserWizard, או שחשבונות משתמשים ינוהלו על ידי Web Site Administration Tool המובנה. משתמשים

יכולים לנהל את חשבונותיהם באופן עצמאי בעזרת פקדי ההתחברות: הם יכולים לאחזר סיסמה שאבדה או לשנות סיסמה קיימת. פקדי ההתחברות האחרים כמו `LoginName`, `LoginView` ו-`LoginStatus`, מאפשרים להציג תכנים שונים למשתמשים אנונימיים ולמשתמשים מחוברים, ומאפשרים אפילו להבדיל בין משתמשים בתפקידים שונים.

ניתן לנהל תפקידים גם על ידי `WSAT`, כמו למשל להקצות חשבונות משתמש לתפקידים שונים כפי שראית בפרק זה. תוכל אז לפתוח משאבים מסוימים באתר לחברים בתפקיד מסוים. לדוגמה, עם אלמנט `<location>` פשוט בקובץ `web.config` חסמת את הגישה לתיקייה `Management` לכל המשתמשים שאינם חברים בתפקיד `Managers`.

פקדי ההתחברות השונים מאפשרים להתאים לפי טעם אישי את התוכן שיוצג למשתמשים. בפרק 16 תגלה איך לקחת עניין זה צעד אחד קדימה על ידי יצירת דפים דינמיים שמתאימים עצמם בהתאם למשתמש שניגש אליהם.

תרגילים

1. מה ההבדל בין אימות לבין הרשאה?
2. ברגע זה חסומה התיקייה `Management` לכל המשתמשים, פרט לאלה שבתפקיד `Managers`. איזה שינוי צריך לבצע בקובץ `web.config` אם ברצונך לפתוח את התיקייה למשתמש `John` ולכל האנשים בתפקיד `Editors`?
3. תאר לעצמך שיש לך אתר אינטרנט אשר מציג דף `Login` עם פקד `Login` יחיד. איזה שינוי יש לבצע בפקד `Login` זה כדי לשלוח משתמשים, לאחר שהתחברו, אל דף `MyProfile.aspx` שנמצא בשורש האתר?

פרק 16

התאמה אישית של אתרי אינטרנט

רק דבר טוב אחד עולה על תוכן טוב ברשת - זהו תוכן טוב שמותאם אישית. בעידן של התפוצצות המידע וכמות עצומה של אתרים מתחרים, חשוב לך לדעת מי המבקרים באתר שלך ולהבין את האפשרויות העומדות בפניך להצגת תוכן מותאם אישית. עם אסטרטגיית התאמה אישית טובה, תוכל ליצור אתרי אינטרנט שעונים על ציפיות המשתמשים שלך, על ידי הצגת הנתונים שהם מחפשים ומעוניינים בהם. התאמה אישית שימושית בתרחישים רבים ושונים. באתרי ספורט תשתמש בה כדי להדגיש פעילויות של הקבוצות שהמשתמש אוהד. באתר שעוסק בתכנות, תוכל להתאים את התוכן באופן אישי על ידי הצגת דוגמאות שרלוונטיות לשפת התכנות המועדפת על המשתמש. באתר שעוסק בחדשות תוכל להניח למשתמש לבחור באחת או יותר קטגוריות חדשות (העולם, מקומיות, ספורט, עסקים, פיננסים וכו') ולכוון את התוכן שתציג על פי העדפות אלו. תוכל לקחת זאת צעד אחד קדימה על ידי שליחת עדכונים בדואר אלקטרוני כשמתפרסם מאמר חדש באחת מאותן קטגוריות.

עם זאת, התאמה אישית היא יותר מאשר אחסון של העדפות אישיות והתאמת התוכן אליהן. עם התאמה אישית אפשר לעקוב אחר פרטים נוספים שנוגעים למשתמש כמו שמות, תאריך לידה, ביקורים באתר, פריטים שהמשתמש רכש בחנות מקוונת, וכך הלאה. תוכל להשתמש בפרטים אלה להתאמה אישית נוספת של דפי האינטרנט כדי ליצור מערכת יחסים קרובה עם המבקר בהם.

באתר Planet Wrox ההתאמה האישית מיושמת בפשטות, אך ביעילות. הדף Reviews מציג רק את הסקירות (ביקורות) הנוגעות לאותם סגנונות שהמשתמש מעוניין בהם. משתמשים יכולים עדיין לראות את כל הסקירות הזמינות על ידי ביקור בדף All.aspx, אבל על ידי ביקור בדף שמותאם אישית, הם יראו סקירות בסגנונות שהם אוהבים בלבד.

בנוסף, משתמשים יכולים להכניס פרטים אישיים אודות עצמם, כמו שם פרטי ושם משפחה, וביוגרפיה קצרה. פרטים אלה יוצגו בקטע Photo Albums של האתר כדי שתדע מי העלה אלבום תמונות מסוים.

כדי לאפשר הוספת תכונות התאמה אישית לאתר אינטרנט, כוללת ASP.NET 3.5 גם שירות יישום בשם Profile. השירות Profile מאפשר לאחסן נתונים אודות משתמש מסוים תוך שימוש בשורות קוד מועטות בלבד.

הפרק עוסק בנושאי ההתאמה האישית הבאים:

- ⊙ התכונה Profile שמגיעה עם ASP.NET.
- ⊙ כיצד ליצור ולהתייחס לפרופיל משתמש באתר אינטרנט.
- ⊙ כיצד ניתן לזהות משתמשים ואיך נותנים להם תוכן מותאם אישית.
- ⊙ איך לגשת לפרופיל של משתמשים אחרים באתר.

בסיום הפרק יהיה בידך מידע מספיק אודות תכונות ההתאמה האישית שמספק השירות Profile, כדי ליצור אתרי אינטרנט דינמיים ומותאמים אישית.

הבנת היישום Profile

Profile הוא יישום שירות שכלול ב-ASP.NET. הוא מאפשר לאחסן ולאחזר מידע אודות משתמשי האתר, מעבר למידע הבסיסי כמו כתובת הדואר האלקטרוני וסיסמה שמשתמשים יכולים להכניס כשהם נרשמים לאתר, כפי שלמדת בפרק הקודם. עם Profile אפשר לאחסן מידע כמו שם פרטי ושם משפחה, תאריך לידה, והרבה יותר, כפי שתראה בהמשך הפרק. על ידי מעקב אחר המשתמש שהמידע שייך לו, ASP.NET מסוגלת למפות את המידע למשתמש בפעם הבאה שיבקר באתר, בין אם אלו דקות או שבועות מאוחר יותר. הדבר המגניב ב-Profile הוא בכך שהוא מאפשר לאחסן נתונים הן לגבי משתמשים רשומים והן לגבי משתמשים אנונימיים. לפיכך, גם אם המבקרים לא נרשמו לפתיחת חשבון, תוכל להכיר אותם ולאחסן מידע אודותם.

הגישה למאפייני היישום Profile היא דרך API נקי, ולמעשה ללא קוד. כל שעליך לעשות הוא להגדיר את המידע שברצונך לעקוב אחריו בקובץ web.config המרכזי, והתכונה Profile דואגת לכל השאר. כל הפעילות האינטראקטיבית עם בסיס הנתונים לאחזור או אחסון מידע הפרופיל בבסיס הנתונים נעשית באופן אוטומטי.

הפעלת Profile ביישום אינטרנט היא תהליך פשוט בן שלושה צעדים:

1. בקובץ web.config קבע את המידע שברצונך לאחסן אודות המשתמש.
2. בהתבסס על מידע זה, ASP.NET runtime יוצר ומהדר תוך כדי ביצוע, מחלקה שמאפשרת גישה למאפיינים שהוגדרו בצעד 1. אחר כך מוסיף runtime באופן דינמי מאפיין בשם Profile אל הדפים באתר, כדי שתוכל לגשת ל-Profile מכל דף באתר.
3. ביישום אתה מתכנת ישירות מול המחלקה שנוצרה כדי לקבל ולאחסן את מידע ה-Profile עבור המשתמש הנוכחי.

כברירת מחדל, ה-Profile משויך למשתמש מחובר, למרות שביכולתך גם לשמור מידע פרופיל על משתמשים לא מאומתים, כפי שתראה מאוחר יותר בפרק.

מכיוון שנתונים אודות המשתמש המחובר מאוחסנים בעוגייה (cookie), הרי שכדי שהתכונה Profile של ASP.NET תפעל כהלכה, דפדפני המשתמשים צריכים לתמוך במנגנון העוגיות.

בסעיף הבא תלמד כיצד להגדיר את מאפייני הפרופיל בקובץ web.config, וכיצד להשתמש ב-Profile החדש בדפי האינטרנט שלך.

עיצוב Profile

פרופיל עבור אתר אינטרנט מוגדר בקובץ web.config על ידי יצירת אלמנט <profile> כבן ישיר של האלמנט <system.web>. בין התגיות <profile> עליך ליצור אלמנט <properties> שמשמש להגדרת המאפיינים שברצונך לחשוף מהאובייקט Profile. יש שני סוגי מאפיינים, מאפיינים פשוטים ומאפיינים מורכבים, שפונים אליהם כקבוצות פרופיל.

יצירת מאפייני Profile פשוטים

מאפיינים פשוטים מוגדרים כבנים ישירים של האלמנט <properties> תוך שימוש באלמנט <add />. הדוגמה הבאה מציגה כיצד ליצור מאפיין שיכיל את השם הפרטי של משתמש, ומאפיין שיכיל את תאריך הלידה שלו. המאפיין FirstName ניתן לגישה ולהגדרה גם עבור משתמשים אנונימיים, ואילו מאפיין תאריך הלידה נגיש רק עבור משתמשים מחוברים.

```
<system.web>
```

```
...
```

```
<profile>
  <properties>
    <add name="FirstName" allowAnonymous="True" />
    <add name="DateOfBirth" type="System.DateTime" />
  </properties>
</profile>
```

כברירת מחדל, מאפיינים הם מסוג System.String. לפיכך, עבור מאפיינים פשוטים כמו השם הפרטי, אין צורך להגדיר type מפורש במאפיין. עם זאת, עבור סוגים אחרים כמו DateTime, Boolean או Integer, עליך להגדיר את הסוג במפורש על ידי התכונה type ולציין את השם המלא של סוג המאפיין (כולל את מרחב השמות שלו). הטבלה הבאה מפרטת את המאפיינים הנפוצים ביותר של האלמנט <add /> שמשפיעים על המאפיינים של פרופיל.

מאפיין	תיאור
Name	מגדיר את שם המאפיין, כמו FirstName, DateOfBirth וכו'.
Type	מגדיר את השם המלא של סוג המאפיין ב-.NET, כמו System.String, System.Boolean, System.DateTime וכו'.
allowAnonymous	מציין אם ניתן להגדיר את המאפיין עבור משתמשים אנונימיים. ברירת המחדל היא False. כדי להגדיר תכונה זו ל-True, עליך להפעיל גם את anonymousAuthentication שבדיון מאוחר יותר בפרק.

מאפיין	תיאור
defaultValue	קובע את ערך ברירת המחדל של המאפיין. אם מאפיין זה מושמט, מאפיין הפרופיל מקבל כברירת מחדל את ערך ברירת המחדל של הסוג הבסיסי (לדוגמה, null עבור string, 0 עבור Int32 וכו')
readOnly	מציין אם ניתן לשנות את מאפיין הפרופיל בזמן ביצוע. ברירת המחדל היא False. כלומר, אפשר לקרוא מהמאפיין ולכתוב אליו.

בנוסף למאפיינים הפשוטים, ביכולתך ליצור קבוצות פרופיל שמאפשרות לקבץ יחד מאפיינים פשוטים אחרים.

יצירת קבוצות פרופיל

קבוצות פרופיל משרתות שתי מטרות ברורות. הן מאפשרות לקבץ מאפיינים שקשורים זה לזה באופן לוגי. לדוגמה, תוכל ליצור קבוצה בשם Address שבתורה תהיה בעלת מאפיינים כמו Street, PostalCode ו-City.

קבוצות גם מאפשרות מאפיינים בעלי אותו שם, בתנאי שהם שייכים לקבוצות שונות. לדוגמה, אפשר שתהיינה לך שתי קבוצות, VisitAddress ו-PostalAddress, ובשתיהן יהיו מאפיינים כמו Street ו-PostalCode. הדבר יקל על מתכנת שמשתמש באובייקט Profile למצוא את המידע הרלוונטי.

כדי ליצור קבוצת פרופיל, עליך להוסיף אלמנט <group> לאלמנט <properties> של הפרופיל ולציין בו שם. האלמנט <group> יכול אז מאפיין פשוט אחד או יותר. הדוגמה הבאה מראה קבוצת פרופיל עבור PostalAddress:

```
<properties>
  <add name="FirstName" />
  <group name="PostalAddress">
    <add name="Street"/>
    <add name="PostalCode"/>
    <add name="City"/>
    <add name="Country"/>
  </group>
</properties>
```

בין התגיות <properties> יכולות להופיע קבוצות פרופיל רבות, אבל כולן חייבות להיות באותה רמה. כלומר, אינך יכול לקנן אלמנט <group> בתוך אלמנט <group> או <add /> אחר.

שימוש בסוגי נתונים לא-תקינים

בנוסף לסוגי הנתונים שפורטו לעיל כמו String, DateTime ו-Integer, באפשרותך להשתמש גם בסוגים משלך שאותם הגדרת (לדוגמה, בתיקייה App_Code).

כמו עם הסוגים המובנים של .NET, עליך לפנות לסוג שלך ולציין את שמו המלא שכולל את מרחב השמות ושם המחלקה. תוכל להניח שיש לך סוג בשם Preference שמכיל מאפיינים שונים שקשורים להעדפות המשתמש. כדי לכלול סוג זה ב-Profile, עליך לעטוף אותו תחילה במרחב שמות:

VB.NET

```
Namespace PlanetWrox
    Public Class Preference
        Private _favoriteColor As String
        Public Property FavoriteColor() As String ' You need full properties in VB.NET
            Get
                Return _favoriteColor
            End Get
            Set(ByVal value As String)
                _favoriteColor = value
            End Set
        End Property
        ' Other properties here
    End Class
End Namespace
```

C#

```
namespace PlanetWrox
{
    public class Preference
    {
        public string FavoriteColor {get; set;}
        // You can use automatic properties in C#
        // Other properties here
    }
}
```

עכשיו תוכל להתייחס לסוג במאפיין של Profile:

```
<add name="Prefs" type="PlanetWrox.Preference" />
```

כאשר משתמשים ב-generics דרוש תחביר שונה כדי להתייחס לסוג בהגדרות Profile. פרק 5 דן בשימוש ב-generics כדי לאחסן שמות תפקיד באמצעות רשימת מחרוזות של generics.

VB.NET

```
Dim roles As New List(Of String)
...
roles.Add("Members")
```

C#

```
List<string> roles = new List<string>();
...
roles.Add("Members");
```

כדי לקבוע בפרופיל מאפיין מסוג List גנרי, צריך להשתמש בתחביר מיוחד. ההגדרה הבאה ב-web.config יוצרת מאפיין פרופיל בשם FavoriteGenres, שמאחסן את הסגנונות המועדפים על המשתמש כ- List (Of Integer) ב-VB.NET, או כ- List<int> ב-C#.

```
<add name="FavoriteGenres" type="System.Collections.Generic.List`1[System.Int32]" />
```

החלק הראשון של התכונה type נראה רגיל למדי. המחלקה List חיה במרחב השמות System.Collections.Generic והגיוני שצריך לציין זאת גם כאן. עם זאת, מייד לאחר שם המחלקה (List) מופיע 1. זו אינה שגיאת דפוס, אלא הדרך של .NET. להתייחס לסוגים גנריים בטקסט פשוט. כדי להגדיר מאפיין שמבוסס על סוג גנרי, צריך להשתמש בגרש הפוך (back tick, `), שבדרך כלל נמצא בלוח המקשים לשמאל המקש 1. לאחר 1 מופיע זוג סוגריים מרובעים שמכיל את הסוג הממשי שברצונך להשתמש בו עבור הרשימה. הסוג שמצוין במאפיין זה של Profile זהה למקבילים לו ב-VB.NET ו-C#, שמוגדרים כך:

VB.NET

```
Dim FavoriteGenres As New List(Of Integer)
```

C#

```
List<int> FavoriteGenres = new List<int>();
```

בתרגילים הבאים תלמד כיצד להשתמש במאפיין זה ובמאפיינים אחרים של Profile. תחילה, בתרגיל נסה זאת הבא תלמד כיצד להגדיר את Profile ב-web.config. תרגילים מאוחרים יותר יראו כיצד לפעול עם מאפיינים אלה וכיצד להשתמש בשיטות השונות של המחלקה List.

נסה זאת יצירת פרופיל

בתרגיל זה תראה כיצד ליצור פרופיל שמסוגל לאחסן את השם הפרטי ושם המשפחה של משתמש, את תאריך הלידה, ביוגרפיה קצרה, ואת רשימת המזהים (IDs) של הסגנונות המועדפים על המשתמש.

1. פתח את הקובץ web.config משורש האתר ואתר את תגית הפתיחה <system.web>.
2. תחת תגית זו הוסף אלמנט <profile> חדש.
3. השלם את האלמנט <profile> באופן שבסופו של דבר ייראה כך:

```
<system.web>
  <profile>
    <properties>
      <add name="FirstName" />
      <add name="LastName" />
      <add name="DateOfBirth" type="System.DateTime"/>
      <add name="Bio" />
      <add name="FavoriteGenres"
        type="System.Collections.Generic.List`1[System.Int32]" />
    </properties>
  </profile>
</roleManager enabled="true"/>
```

4. שמור את הקובץ web.config על ידי הקשת Ctrl+S. ברגע שתשמור את הקובץ יתחיל תהליך רקע ליצירת קובץ מחלקה שישמש את Profile. לאחר שקובץ המחלקה נוצר והודר בהצלחה, תוכל לגשת ל-Profile בתכנות באמצעות מאפיין Profile מהמחלקה Page.
5. כדי לבדוק את Profile, פתח את הדף My Profile שיוצרת בפרק הקודם ב- Design View, לחץ לחיצה כפולה על הדף כדי להגדיר מטפל אירוע לאירוע Load, והוסף את הקוד הבא:

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Me.Load
```

```
    Profile.FirstName = "Your First Name"
    Profile.LastName = "Your Last Name"
```

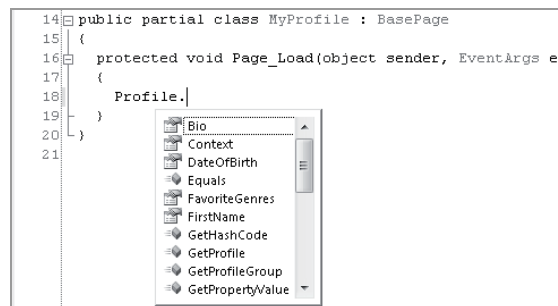
```
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
```

```
{
    Profile.FirstName = "Your First Name";
    Profile.LastName = "Your Last Name";
}
```

6. כאשר תקליד את הנקודה (.) לאחר Profile, תופיע רשימה של IntelliSense שתציג את המאפיינים הזמינים של Profile (ראה תרשים 16-1).



תרשים 16-1

7. כשתסיים להקליד את הקוד, שמור וסגור את הקובץ מכיוון שאינו דרוש לך כעת. אם שום דבר אינו מופיע בעת לחיצת הנקודה, בחר Build Solution ← Build מהתפריט הראשי, או הקש Ctrl+Shift+B. הדבר כופה על VWD להתחיל הידור מחדש של היישום, כולל המחלקה המיוחדת שנוצרה עבור Profile. בפרק 18 תלמד עוד על הידור. לאחר עיכוב של כמה שניות אמורים המאפיינים להופיע ברשימה של IntelliSense עבור המאפיין של המחלקה Page ב-Profile. אם המאפיינים עדיין אינם מופיעים, בדוק את רשימת השגיאות (כדי לפתוח את רשימת השגיאות בחר View ← Error List מהתפריט הראשי) כדי לוודא שלא עשית שגיאות כלשהן בקובץ web.config.

8. חזור לקובץ web.config וגלול עד סופו. צור עותק של האלמנט <location> שחוסם את הגישה לתיקייה Management למשתמשים שאינם מורשים, והדבק אותו מייד תחת האלמנט הקיים. שנה את העותק באופן שיחסום את הגישה לקובץ MyProfile.aspx שבשורש האתר לכל המשתמשים שאינם מאומתים. אתה אמור לסיים עם ההגדרות הבאות:

```
</location>
<location path="MyProfile.aspx">
  <system.web>
    <authorization>
      <deny users="?"/>
    </authorization>
  </system.web>
</location>
</configuration>
```

9. בסייר הפתרון, לחץ לחיצה ימנית על הקובץ MyProfile.aspx ובחר View in Browser. תוכל לראות קובץ זה רק כשאתה מחובר. אם לא היית מחובר קודם, תועבר כעת אל Login.aspx. התחבר עם שם המשתמש והסיסמה שיוצרת בפרק הקודם ולחץ Login. בתגובה תוחזר אל MyProfile.aspx. למרות שלא תראה שום דבר חדש בדף, הקוד ב- Page_Load התבצע ויצר פרופיל בבסיס הנתונים.

10. כדי לראות את הפרופיל שנוצר, סגור את הדפדפן וחזור ל-VWD. פתח את חלון סייר בסיס הנתונים והרחב את האלמנט Tables של בסיס הנתונים ASPNETDB.MDF. אתר את הטבלה aspnet_Profile, לחץ עליה לחיצה ימנית ובחר Show Table Data. אתה אמור לראות תרשים דומה לתרשים 2-16.

UserId	PropertyNames	PropertyValuesString	PropertyValuesBinary	LastUpdatedDate
92a14595-23fb-4443-bb23-2d97be754ff6	LastName:S:0:14;FirstName:S:14:15;	Your Last NameYour First Name	<Binary data>	12/1/2007 7:36:20 PM
NULL	NULL	NULL	NULL	NULL

תרשים 2-16

התרשים מראה נתוני Profile עבור משתמש יחיד. השם הפרטי ושם המשפחה שהכנסת בצעד 5 מאוחסנים בעמודה PropertyValuesString. בגלל העיצוב המיוחד שנתונים אלה מאוחסנים בו, אסור לך לשנותם באופן ידני. במקום זאת, השתמש באובייקט Profile כדי לשנות את הנתונים הבסיסיים.

כיצד זה פועל

כשמגדירים ב-web.config את המאפיינים עבור Profile, יוצר ASP.NET runtime מחלקה ברקע. מחלקה זו נקראת ProfileCommon והיא מאפשרת גישה למאפיינים בעלי סוג מובהק כמו LastName, FirstName ו-FavoriteGenres. המחלקה ProfileCommon נעשית נגישה ל-Page

באמצעות המאפיין Profile שלו. המחלקה ProfileCommon יורשת מ-ProfileBase, שהיא מחלקת הבסיס של סביבת העבודה ASP.NET המכילה את ההתנהגות הדרושה לגישה ל-Profile בבסיס הנתונים, על ידי פנייה לרכיב ספק ה-Profile. הספק בתורו דואג לכל העבודה הקשה הכרוכה בשמירת הנתונים בבסיס הנתונים. כמו ספק החברות וספק התפקיד שראית בפרק הקודם, גם ספק Profile, כברירת מחדל, משתמש בבסיס הנתונים שמוגדר במחרוזת החיבור LocalSqlServer. בנספח ב' תלמד כיצד לשנות התנהגות זו ולגרום לספק להשתמש בבסיס נתונים אחר.

כדי להגדיר מאפיינים, השתמש באלמנטים <add /> עם המאפיין name, והמאפיין type (רשות) אם המאפיין הוא מסוג שונה מאשר System.String.

```
<add name="FavoriteGenres" type="System.Collections.Generic.List`1[System.Int32]" />
```

מאפיין זה מגדיר רשימה שיכולה לאחסן ערכים מסוג Integer עבור הסגנונות המוזיקליים המועדפים על המשתמש. בתרגיל מאוחר יותר תראה כיצד להשתמש במאפיין זה.

לאחר הגדרת Profile ב-web.config והידור מחלקת הרקע, תוכל לגשת ל-Profile מהקוד. לדוגמה, תוכל כעת להגדיר מאפיינים כמו FirstName על ידי קוד:

VB.NET

```
Profile.FirstName = "Your First Name"
```

C#

```
Profile.FirstName = "Your First Name";
```

למרות שלא הראינו זאת בתרגיל הקודם, הגישה למאפיינים בקבוצה נעשית בדרך דומה. כל שעליך לעשות הוא להציב את שם הקבוצה כתחילית של שם המאפיין. על פי הדוגמה של PostalAddress, תאחסן את שם הרחוב עבור אותה כתובת בדרך זו:

VB.NET

```
Profile.PostalAddress.Street = "Some Street"
```

C#

```
Profile.PostalAddress.Street = "Some Street";
```

שינויים שנעשו ב-Profile נשמרים באופן אוטומטי במהלך EndRequest, אירוע שמתעורר מאוחר מאוד במהלך מחזור החיים של דף ב-ASP.NET. לפיכך תוכל לשנות את Profile במהלך שלבים רבים של מחזור החיים מבלי שתצטרך לשמור אותו במפורש באופן ידני.

בתרשים 2-16 אפשר לראות כיצד משתמשים ברשומה יחידה כדי לאחסן את הפרופיל כולו. העמודה הראשונה מכילה ID ייחודי של המשתמש שהפרופיל שייך לו. העמודה השנייה מכילה רשימה של שמות מאפיינים שנשמרים עבור המשתמש הנוכחי, יחד עם אינדקס התחלתי של הערך ואורך בתווים. לדוגמה, עבור שם המשפחה תראה:

```
LastName:S:0:14
```


פירוש הדבר, שהערך של המאפיין LastName, שמאוחסן בעמודה PropertyValuesString, מתחיל במיקום 0 ואורכו 14 תווים. האות שבין LastName והאינדקס ההתחלתי מגדירה את העמודה שבה ניתן למצוא את הנתונים. האות S פירושה העמודה PropertyValuesString (מחרוזת), והאות B מציינת את העמודה PropertyValuesBinary (בינארי), שמשמשת לאחסון אובייקטים מורכבים בפורמט בינארי. עיצוב דחוס זה מאפשר לספק Profile לאחסן מאפיינים רבים ושונים בעמודה אחת, והדבר מבטל את הצורך לעסוק בסכמת בסיס הנתונים בכל פעם שהפרופיל משתנה.

בסעיף הבא תלמד יותר על קריאה מ-Profile ועל כתיבה אליו.

שימוש ב-Profile

כפי שראית בסעיף הקודם, הכתיבה ל-Profile קלה ופשוטה. כדי לשנות מאפיין כמו FirstName דרושה שורה אחת של קוד. ה-Profile עוקב אחר השינויים שביצעת בו, ואם דרוש, הוא שומר את השינויים באופן אוטומטי במהלך EndRequest. קריאה מ-Profile קלה באותה מידה, כי כל שעליך לעשות הוא לגשת לאחד מהמאפיינים שלו. הקוד הבא מראה כיצד למלא TextBox עם השם הפרטי מהפרופיל:

VB.NET

```
txtFirstName.Text = Profile.FirstName
```

C#

```
txtFirstName.Text = Profile.FirstName;
```

אחזור מאפיינים מתוך קבוצה זהה כמעט לחלוטין. כדי לגשת למאפיין Street, שדנו בו בתרגיל קודם, דרוש הקוד הבא:

VB.NET

```
txtPostalAddressStreet.Text = Profile.PostalAddress.Street
```

C#

```
txtPostalAddressStreet.Text = Profile.PostalAddress.Street;
```

הגישה למאפיין FavoriteGenres שונה מעט. מכיוון שמאפיין זה הוא אוסף (collection), אסור לך לגשת אליו ישירות. במקום זאת תשתמש בשיטות ובמאפיינים שלו כדי להכניס ולהוציא ממנו נתונים. הדוגמה הבאה מנקה תחילה את הרשימה כולה, ואז מוסיפה לה את ה-IDs של שני סגנונות:

VB.NET

```
Profile.FavoriteGenres.Clear()  
Profile.FavoriteGenres.Add(7)  
Profile.FavoriteGenres.Add(11)
```

C#

```

Profile.FavoriteGenres.Clear();
Profile.FavoriteGenres.Add(7);
Profile.FavoriteGenres.Add(11);

```

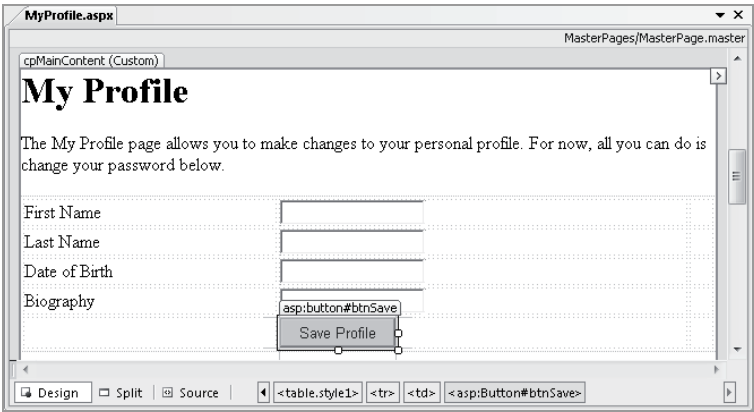
התרגיל הבא מראה לך כיצד לאחסן ב-Profile נתונים בסיסיים. בתרגיל מאוחר יותר תראה יישום אמיתי של שימוש ברשימה FavoriteGenres.

נסה זאת אחסון נתונים בסיסיים של המשתמש ב-Profile

בתרגיל זה תשנה את הדף Profile באופן שמשתמש יוכל להכניס בו את שמו הפרטי ושם משפחתו, תאריך הולדתו וביוגרפיה קצרה.

1. פתח שוב את MyProfile.aspx ועבור לקוד ברקע. ב-Page_Load הסר את שתי שורות הקוד שמגדירות את השם הפרטי ואת שם המשפחה של המשתמש.
2. עבור ל- Design View. מייד מעל לפקד ChangePassword הוסף טבלת HTML בעלת חמש שורות ושלוש עמודות, על ידי בחירת Insert Table ← Table.
3. גרור פקדי TextBox אל העמודה השנייה בכל אחת מארבע השורות הראשונות, ושנה את שמם, מהשורה הראשונה לרביעית, ל-txtFirstName,txtLastName,txtDateOfBirth ו-txtBio בהתאמה. תרשים 3-16 מראה היכן בדיוק צריך להציב את פקדי TextBox.
4. בעמודה הראשונה של כל אחת מארבע השורות הראשונות שחרר פקדי Label והגדר את המאפיינים שלהם, כמו שתראה להלן. כל תווית משויכת ל-TextBox שבאותה שורה.

משיך לפקד בעל מזהה AssociatedControlID	Text	ID
txtFirstName	First Name	lblFirstName
txtLastName	Last Name	lblLastName
txtDateOfBirth	Date of Birth	lblDateOfBirth
txtBio	Biography	lblBio



תרשים 3-16

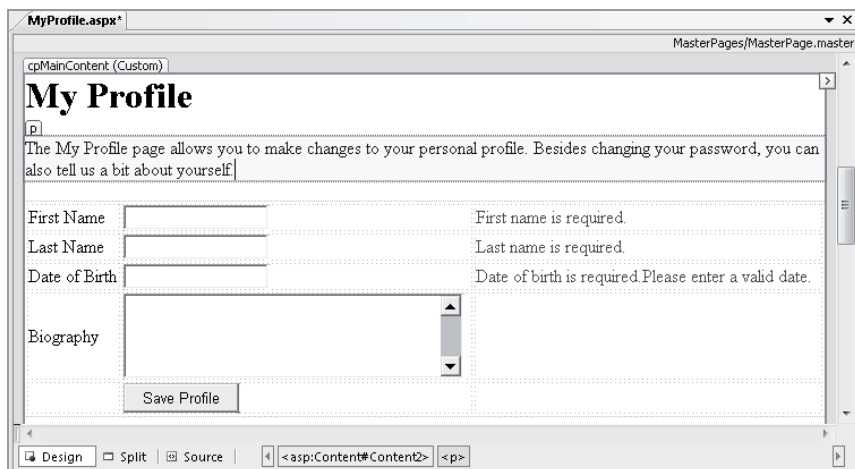
5. בתא השני של השורה החמישית, גרור לחצן, הגדר את ה-ID שלו ל-`btnSave` ואת ה-`Text` שלו הגדר ל-`Save Profile`. כעת `Design View` צריך להיראות כמו תרשים 3-16.
6. גרור פקדי `RequiredFieldValidator` אל העמודה האחרונה של כל אחת משלוש השורות הראשונות. קבע את המאפיינים שלהם כמו בטבלה שלהלן, כדי שכל פקד אימות יסתדר בשורה עם `TextBox` שבאותה שורה.

Error Message	Display	ControlToValidate
First name is required.	Dynamic	txtFirstName
Last name is required.	Dynamic	txtLastName
Date of birth is required.	Dynamic	txtDateOfBirth

7. לצד פקד האימות עבור התיבה `DateOfBirth`, גרור פקד אימות `CompareValidator` והגדר את מאפייניו:

מאפיין	ערך
Display	Dynamic
ErrorMessage	Please enter a valid date.
ControlToValidate	txtDateOfBirth
Operator	DataTypeCheck
Type	Date

8. הגדר את `TextMode` של הפקד `txtBio` ל-`MultiLine`, ואת המאפיינים `Width` ו-`Height` שלו הגדר ל-`300px` ו-`75px` בהתאמה.
9. שנה את הטקסט מעל הטבלה, כדי שיציין שמשתמשים יכולים לעשות כעת יותר מאשר לשנות את סיסמתם בלבד. `Design View` אמור להיראות כמו בתרשים 4-16.



תרשים 16-4

10. לחץ לחיצה כפולה על Button, ובמטפל האירוע Click ש-WVD הוסיפה, כתוב את הקוד הבא:

VB.NET

```
Protected Sub btnSave_Click(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles btnSave.Click
```

```
    If Page.IsValid Then  
        Profile.FirstName = txtFirstName.Text  
        Profile.LastName = txtLastName.Text  
        Profile.DateOfBirth = DateTime.Parse(txtDateOfBirth.Text)  
        Profile.Bio = txtBio.Text  
    End If
```

```
End Sub
```

C#

```
protected void btnSave_Click(object sender, EventArgs e)
```

```
{  
    if (Page.IsValid)  
    {  
        Profile.FirstName = txtFirstName.Text;  
        Profile.LastName = txtLastName.Text;  
        Profile.DateOfBirth = DateTime.Parse(txtDateOfBirth.Text);  
        Profile.Bio = txtBio.Text;  
    }  
}
```

11. במטפל האירוע Page_Load של אותו הדף הוסף את הקוד הבא, שממלא את פקדי תיבות הטקסט עם הנתונים מהפרופיל בעת שהדף נטען:

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Me.Load
```

```
    If Not Page.IsPostBack Then  
        txtFirstName.Text = Profile.FirstName  
        txtLastName.Text = Profile.LastName  
        txtDateOfBirth.Text = Profile.DateOfBirth.ToShortDateString()  
        txtBio.Text = Profile.Bio  
    End If
```

```
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
```

```
{  
    if (!Page.IsPostBack)  
    {  
        txtFirstName.Text = Profile.FirstName;  
        txtLastName.Text = Profile.LastName;  
        txtDateOfBirth.Text = Profile.DateOfBirth.ToShortDateString();
```

```
txtBio.Text = Profile.Bio;
}
```

12. שמור את כל השינויים ובקש את הדף בדפדפן. אם תידרש להתחבר תחילה, הכנס את פרטיך ולחץ על הלחצן Login. אתה אמור לראות כעת שהדף My Profile מופיע מחדש עם הנתונים שהכנסת בתרגיל נסה זאת הקודם עבור השם הפרטי ושם המשפחה. שנה נתונים אלה לשם שלך, ולחץ על הלחצן Save Profile.
13. סגור את הדפדפן ובקש שוב את MyProfile.aspx. שים לב שהנתונים שלך נשמרו בין שתי הפעלות הדפדפן.

כיצד זה פועל

הרבה ממה שראית בתרגיל זה צריך להיות מוכר לך עכשיו. הדף מכיל כמה פקדי TextBox שתקפותם נבדקת בעזרת הפקדים RequiredFieldValidator ו-CompareValidator. בנוסף, פקדי Label מתחברים לפקדי TextBox המתאימים שלהם, על ידי שימוש במאפיין AssociatedControlID. הדבר מאפשר התמקדות בפקדים שבדפדפן, מכיוון שכעת לחיצה על Label תציב את הסמן בתיבת הטקסט TextBox המשויכת לו.

כשלוחצים על הלחצן Save Profile, מאוחזרים הערכים מארבעת פקדי TextBox ומאוחסנים בפרופיל. כשהדף נטען בפעם הראשונה, קורה התהליך ההפוך: הפקדים מתמלאים מראש בערכים מה-Profile. כדי להימנע מדריסת הנתונים שהשתמש הכניס, הקוד מקבל את הנתונים מהפרופיל רק בתחילת טעינת הדף, ולא במהלך postback.

VB.NET

```
If Not Page.IsPostBack Then
    txtFirstName.Text = Profile.FirstName
    ....
End If
```

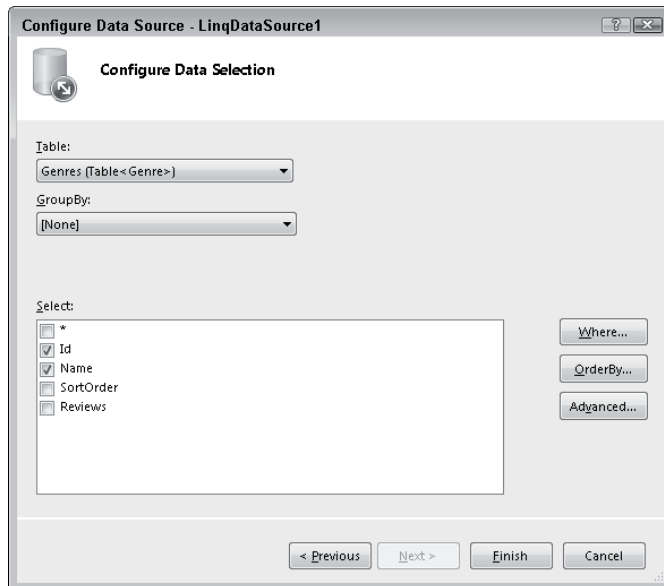
C#

```
if (!Page.IsPostBack)
{
    txtFirstName.Text = Profile.FirstName;
    ...
}
```

למרות שהדוגמה עצמה פשוטה למדי, היא מניחה תשתית נאה לתרחישים מתקדמים יותר על ידי שימוש ברשימה גנרית - List (עבור מספרים שלמים) - לאחסון העדפות המשתמש של סגנונות מוזיקליים מסוימים. אחר כך תוכל להשתמש ברשימה זו של סגנונות מועדפים כדי להגביל את רשימת הסקירות לאלו שהשתמש באמת מעוניין בהן. בתרגיל הבא תלמד כיצד לאחסן את העדפות המשתמש ב-Profile. בתרגיל מאוחר יותר תראה כיצד להשתמש שוב בנתונים אלה.

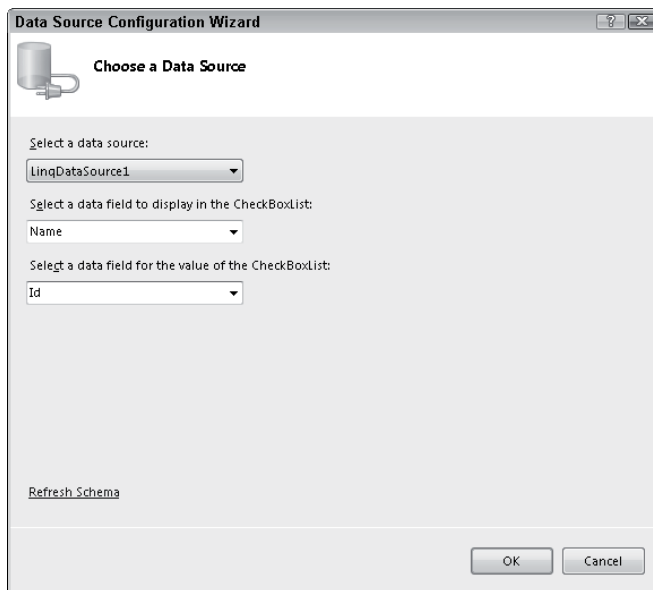
בתרגיל זה תלמד למלא את המאפיין FavoriteGenres של פרופיל המשתמש. כדי לאפשר למשתמש לבחור את הסגנונות המועדפים עליו, וגם תציג CheckBoxList שמחובר ל-LinqDataSource שמאחזר את הסגנונות הזמינים. כשהמשתמש שומר את הנתונים, הפריטים שסימן מאוחסנים ב-Profile.

1. ב-MyProfile.aspx, הוסף שורת טבלה מעל לזו עם הלחצן Save Profile. כדי לבצע זאת, לחץ לחיצה ימנית על השורה עם הלחצן, ומתפריט ההקשר שמופיע בחר Insert ← Row Above.
2. אל התא הראשון של השורה החדשה גרור Label, וקבע את ערך Text שלו ל- Favorite Genres.
3. אל התא השני גרור פקד CheckBoxList מהקטגוריה Standard של ארגז הכלים.
4. חבר את הפקד ל-LinqDataSource חדש תוך שימוש בלוח Smart Tasks שלו. ב- Configure Data Source Wizard, ומהרשימה הנפתחת Table בחר בטבלה Genres. ברשימה Select, סמן רק את הפריטים של העמודות Id ו-Name. לחץ על הלחצן OrderBy, מהרשימה הנפתחת Sort by בחר Name, ולחץ OK. תיבת הדו-שיח Configure Data Source אמורה להיראות כעת כמו תרשים 16-5.



תרשים 16-5

5. לחץ על Finish כדי לסגור את תיבת הדו-שיח. חזור אל Data Source Configuration Wizard של הפקד CheckBoxList. בחר ב-Name כשדה הנתונים להצגה והשאר את Id כשדה הנתונים עבור הערך. שניהם מוצגים בתרשים 16-6.



תרשים 16-6

הקוד עבור הפקד LinqDataSource ועבור CheckBoxList אמור להיראות כך:

```
<asp:CheckBoxList ID="CheckBoxList1" runat="server" DataSourceID="LinqDataSource1"
    DataTextField="Name" DataValueField="Id"></asp:CheckBoxList>
<asp:LinqDataSource ID="LinqDataSource1" runat="server"
    ContextTypeName="PlanetWroxDataContext" OrderBy="Name"
    Select="new (Id, Name)" TableName="Genres"></asp:LinqDataSource>
```

6. ב- Design View, לחץ על הפקד CheckBoxList פעם אחת, פתח את טבלת המאפיינים שלו, ועבור לקטגוריית Events. לחץ לחיצה כפולה על האירוע DataBound, ואז הוסף את הקוד הבא אל הקוד ברקע, כדי לבחור מראש את פריטי הרשימה בהתבסס על הגדרות Profile של המשתמש:

VB.NET

```
Protected Sub CheckBoxList1_DataBound(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles CheckBoxList1.DataBound
    For Each myItem As ListItem In CheckBoxList1.Items
        Dim currentValue As Integer = Convert.ToInt32(myItem.Value)
        If Profile.FavoriteGenres.Contains(currentValue) Then
            myItem.Selected = True
        End If
    Next
End Sub
```

C#

```
protected void CheckBoxList1_DataBound(object sender, EventArgs e)
{
    foreach (ListItem myItem in CheckBoxList1.Items)
```

```

{
    int currentValue = Convert.ToInt32(myItem.Value);
    if (Profile.FavoriteGenres.Contains(currentValue))
    {
        myItem.Selected = true;
    }
}
}

```

7. הרחב את המטפל btnSave_Click עם הקוד הבא, כדי שישמור גם על הסגנונות המועדפים על המשתמש:

VB.NET

```

Profile.Bio = txtBio.Text
' Clear the existing list
Profile.FavoriteGenres.Clear()

' Now add the selected items
For Each myItem As ListItem In CheckBoxLayout1.Items
    If myItem.Selected Then
        Profile.FavoriteGenres.Add(Convert.ToInt32(myItem.Value))
    End If
Next

```

C#

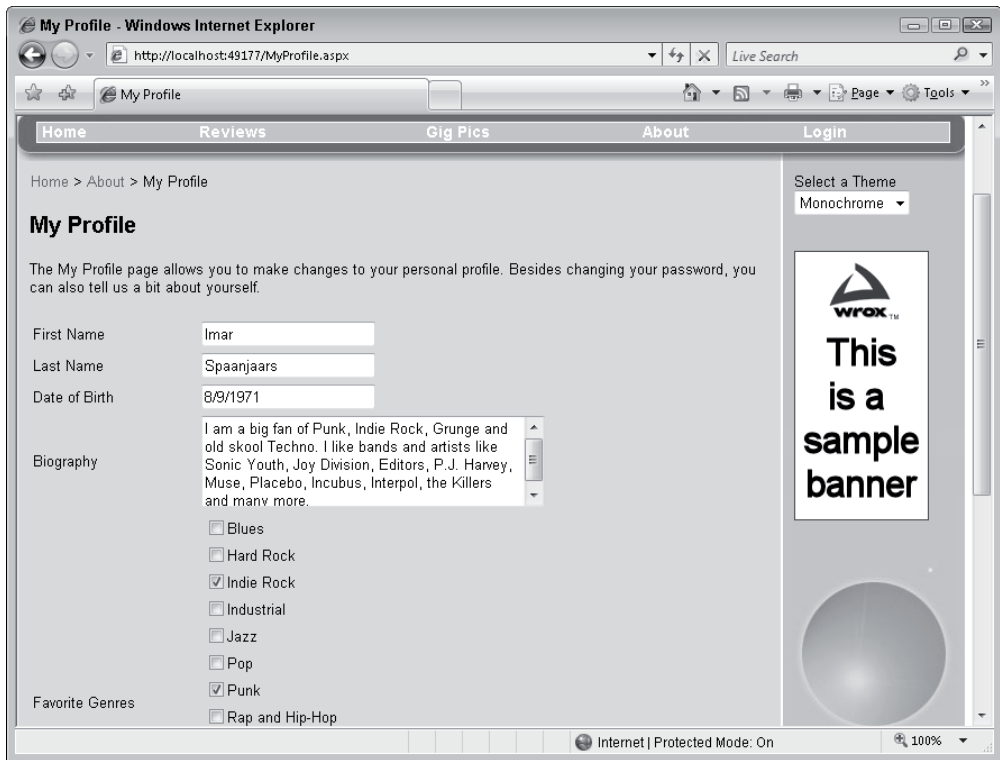
```

Profile.Bio = txtBio.Text;
// Clear the existing list
Profile.FavoriteGenres.Clear();

// Now add the selected items
foreach (ListItem myItem in CheckBoxLayout1.Items)
{
    if (myItem.Selected)
    {
        Profile.FavoriteGenres.Add(Convert.ToInt32(myItem.Value));
    }
}

```

8. שמור את כל השינויים, בקש את הדף Profile בדפדפן, והתחבר כשדרוש. כעת אתה אמור לראות את רשימת הסגנונות מוצגת בדפדפן, כשלפני כל סגנון יש תיבת סימון. בחר בכמה מהסגנונות המועדפים עליך ולחץ על הלחצן Save Profile. דפדף אל דף אחר ובחר שוב My Profile מהתפריט הראשי או מ-TreeView. הסגנונות שבחרת אמורים להיות עדיין נבחרים בדף, כפי שמוצג בתרשים 7-16.



תרשים 16-7

כיצד זה פועל

בסעיף קודם הגדרת את המאפיין FavoriteGenres ב-Profile כרשימה גנרית שיכולה להכיל ערכים של מספרים שלמים. מכיוון שמאפיין זה הוא List, אינך מקצה לו ערכים באופן ישיר. במקום זאת אתה משתמש בשיטות שלו, כמו Add ו-Clear, כדי להוסיף ולהסיר ממנו פריטים. מכיוון שכל ID של סגנון חייב להיות מאוחסן ברשימה פעם אחת בלבד, יש לנקות את הרשימה לפני שמוסיפים לה שוב את הפריטים הנבחרים כשהמשתמש לוחץ על הלחצן Save Profile.

VB.NET

```
Profile.FavoriteGenres.Clear()
```

C#

```
Profile.FavoriteGenres.Clear();
```

עכשיו, כשהרשימה ריקה, מוסיפים IDs של הסגנונות הנבחרים.

VB.NET

```
For Each myItem As ListItem In CheckBoxList1.Items
    If myItem.Selected Then
        Profile.FavoriteGenres.Add(Convert.ToInt32(myItem.Value))
    End If
Next
```

C#

```
foreach (ListItem myItem in CheckBoxList1.Items)
{
    if (myItem.Selected)
    {
        Profile.FavoriteGenres.Add(Convert.ToInt32(myItem.Value));
    }
}
```

קוד זה מבצע לולאה על כל הפריטים ב-CheckBoxList. המאפיין Selected קובע אם המשתמש בחר בפריט בדף Profile. אם הפריט נבחר, הערך של הסגנון מאוחזר, מומר ל-Integer (int) (ב-C#), ומוסיפים אותו לרשימה FavoriteGenres על ידי השיטה Add.

זה כל מה שצריך כדי לאחסן נתונים מורכבים כמו רשימה של סגנונות מועדפים ב-Profile של המשתמש. כל שעליך לעשות הוא להוסיף קבוצת מספרים לרשימה. מאותו רגע, רכיב runtime דואג לשמירת ה-Profile בבסיס הנתונים, ולזמינותו בדפים העוקבים.

כמובן שהרשימה עם הסגנונות המועדפים אינה באמת שימושית, עד שתנצל אותה באופן ממשי באתר. בתרגיל הבא תראה כיצד להשתמש ברשימה כדי להגביל את רשימת הסקירות ההתחלתית שהמשתמשים רואים כשהם מבקרים בדף ברירת המחדל Reviews.

נסה זאת שימוש ב-Profile בדף Reviews

כרגע יש באתר שני דפים בתיקייה Reviews, אשר מסוגלים להציג סקירות - AllByGenre.aspx ו-All.aspx. בתרגיל זה תשנה את הדף Default.aspx באופן שיציג רשימה נוספת של סקירות. עם זאת, הפעם רשימת הסקירות תהיה מוגבלת לאלו ששייכות לסגנונות שהמשתמש בחר בדף My Profile. כשמשתמשים אנונימיים יראו את הדף, הם יקבלו הודעה שטרם קבעו את רשימת הסגנונות המועדפים עליהם.

1. ב- Markup View, פתח את Default.aspx מהתיקייה Reviews.

2. הוסף את הקוד הבא שיוצר Repeater מקונן:

```
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" Runat="Server">
    <asp:Repeater ID="Repeater1" runat="server">
        <HeaderTemplate>
            <p>Below you find a list with Reviews for your favorite music genres.</p>
        </HeaderTemplate>
        <ItemTemplate>
            <h3><asp:Literal ID="Literal1" runat="server"
                Text='<## Eval("Name") %>'></asp:Literal></h3>
            <asp:Repeater ID="Repeater2" runat="server" DataSource='<## Eval("Reviews")%>'>
                <ItemTemplate>
                    <asp:HyperLink ID="HyperLink1" runat="server" Text='<## Eval("Title") %>'>
```

```

        NavigateUrl='<## "ViewDetails.aspx?ReviewId=" + Eval("Id").ToString() %>'>
    </asp:HyperLink><br />
    </ItemTemplate>
</asp:Repeater>
</ItemTemplate>
</asp:Repeater>
<asp:Placeholder ID="plcNoRecords" runat="Server" Visible="False">
    <p>Sorry, no Reviews were found. You either didn't set your favorite genres
        or you may need to log in first. </p>
</asp:Placeholder>
<p>You can change your genre preferences <a href="~/MyProfile.aspx"
    runat="server">here</a>.</p>

```

</asp:Content>

ביכולתך ליצור את פקדי Repeater באופן ידני על ידי כתיבת הקוד הדרוש, או שביכולתך לגרור ולשחרר אותם מהקטגוריה Data של ארגו הכלים. ה-Repeater הפנימי מכיל פקד HyperLink שמצביע על הדף ViewDetails.aspx שיצרת בפרק 14.

3. לחץ לחיצה כפולה על הדף ב- Design View כדי להגדיר מטפל Load, והוסף את הקוד הבא:

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Me.Load
```

```
    Using myDatabaseContext As PlanetWroxDataContext = New PlanetWroxDataContext()
```

```
        Dim favGenres = From genre In myDatabaseContext.Genres _
                        Order By genre.Name _
                        Where Profile.FavoriteGenres.Contains(genre.Id) _
                        Select New With {genre.Name, genre.Reviews}

```

```
        Repeater1.DataSource = favGenres
        Repeater1.DataBind()

```

```
        Repeater1.Visible = Repeater1.Items.Count > 0
        plcNoRecords.Visible = Repeater1.Items.Count = 0

```

```
    End Using
```

```
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
```

```
{
```

```
    using (PlanetWroxDataContext myDatabaseContext = new PlanetWroxDataContext())
```

```
    {
```

```
        var favGenres = from genre in myDatabaseContext.Genres
                        orderby genre.Name
                        where Profile.FavoriteGenres.Contains(genre.Id)

```

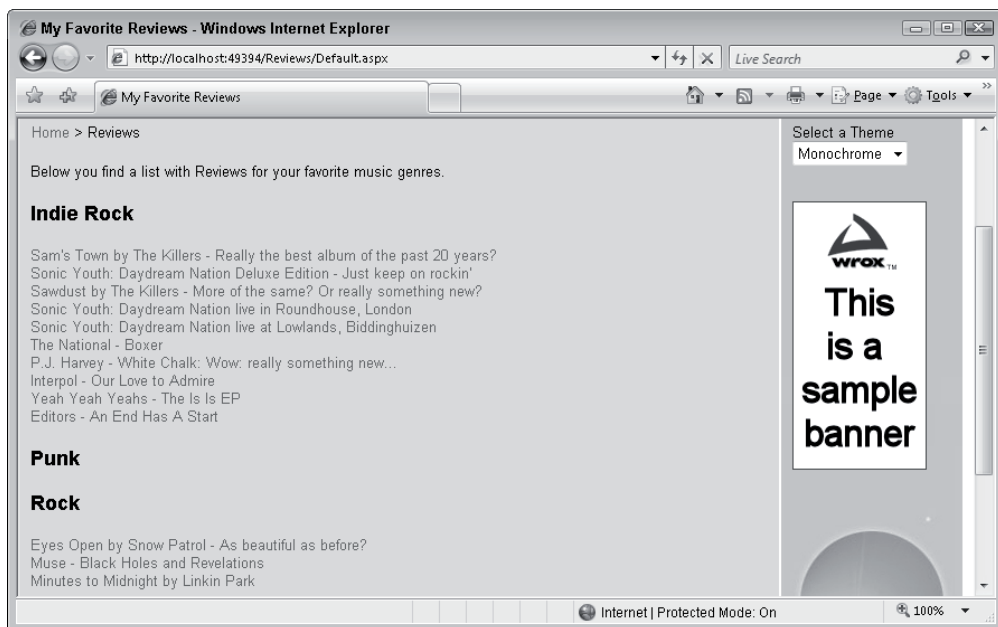
```

        select new { genre.Name, genre.Reviews };
Repeater1.DataSource = favGenres;
Repeater1.DataBind();

Repeater1.Visible = Repeater1.Items.Count > 0;
plcNoRecords.Visible = Repeater1.Items.Count == 0;
}
}

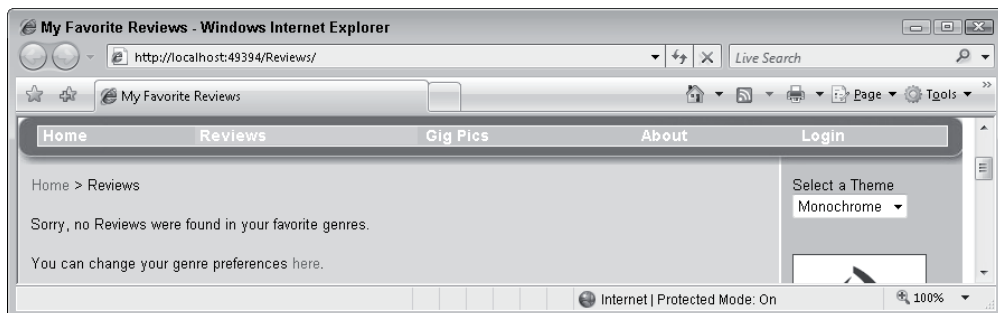
```

4. שמור את כל השינויים ובקש את הדף בדפדפן. אם בחרת קודם לכן סגנון אחד או יותר בדף Profile, ויש סקירות זמינות עבור סגנונות אלה, אתה אמור לראות רשימה דומה לזו שבתרשים 16-8.



תרשים 16-8

אם לא הגדרת סגנונות מועדפים כלשהם, או שאף לא אחד מהסגנונות מכיל סקירות, או שאינך מחובר, תקבל את ההודעה שמוצגת בתרשים 16-9.



תרשים 16-9

אם תלחץ על הקישור שבהודעה תועבר לדף My Profile, ותוכל לקבוע או לשנות את רשימת הסגנונות המועדפים עליך. משתמשים שאינם מורשים, יתבקשו להתחבר או להירשם לקבלת חשבון לפני שיוכלו לגשת לדף Profile.

כיצד זה פועל

הקוד שנמצא בקטע הקוד ברקע מבצע שאילתת LINQ שמאחזרת את כל הסקירות ששייכות לסגנונות המועדפים על המשתמש. עבור משתמשים אנונימיים, רשימת הסגנונות המועדפים תהיה ריקה, ולכן הם תמיד יקבלו את ההודעה לקביעת העדפותיהם בדף Profile.

ה-Repeater המקונן שהוספת לדף Default.aspx בתיקייה Reviews נראה דומה לקוד שבדף AllByGenre.aspx, שיש בו Repeater שמכיל פקד BulletedList. כמו בדף ההוא, ה-Repeater המקונן מקבל את הנתונים שלו מה-Repeater החיצוני באמצעות המאפיין DataSource:

```
<asp:Repeater ID="Repeater2" runat="server" DataSource='<%# Eval("Reviews")%>'>
...
</asp:Repeater>
```

ה-Repeater המקונן משתמש אז ברשימת הסקירות כדי לבנות את הקישורים (hyperlinks) שמעבירים אותך אל דף הפרטים:

```
<asp:HyperLink ID="HyperLink1" runat="server" Text='<%# Eval("Title") %>'
    NavigateUrl='<%# "ViewDetails.aspx?ReviewId=" + Eval("Id").ToString() %>'>
</asp:HyperLink><br />
```

הפקד HyperLink מקבל את ה-Text שלו מהמופיע Review שהוא קשור אליו, ומשתמש ב-Id שלו כדי לבנות את ה-NavigateUrl. השיטה ToString משמשת ב- Eval("Id") כדי להמיר את הערך למחרוזת, לפני שהוא משורשר למחרוזת שמכילה את ה-URL. הדבר נעשה כדי למנוע המרת סוג ב- Visual Basic שבה תוצאת Eval("Id") היא בדרך כלל מספר שאינך יכול לשרשר ישירות למחרוזת.

כדי לראות כיצד פקדים אלה מקבלים את הנתונים שלהם, עליך להתבונן בקוד ברקע שמשתמש בשאילתת LINQ:

VB.NET

```
Dim favGenres = From genre In myDatabaseContext.Genres _
    Order By genre.Name _
    Where Profile.FavoriteGenres.Contains(genre.Id) _
    Select New With {genre.Name, genre.Reviews}
```

C#

```
var favGenres = from genre in myDatabaseContext.Genres
    orderby genre.Name
    where Profile.FavoriteGenres.Contains(genre.Id)
    select new { genre.Name, genre.Reviews };
```

פרט לשורת הקוד המודגשת ולשם המשתנה, שאילתת LINQ זו זהה לזו שמשמשת ב-AllByGenre.aspx. מה שמייחד דוגמה זו היא הפסוקית where, שמגבילה את מספר הסקירות לאלו שהמשתמש באמת מעוניין בהן. שים לב כיצד משתמשים כאן בשיטה Contains של המחלקה הגנרית List. למרות שתחילה זה נראה כאילו כל הסגנונות והסקירות מאוחזרים מבסיס הנתונים אל דף ASPX, ואז מושווים עם הערכים שבמאפיין Profile, הרי שבפועל הדברים הפוכים. מנוע LINQ חכם מספיק כדי לאסוף תחילה את כל ה-IDs מהמאפיין FavoriteGenres, ולכלול אותם במשפט SQL שנשלח לבסיס הנתונים, כדי לאחזר את הסגנונות והסקירות המבוקשים. פירוש הדבר שסינון הסגנונות המבוקשים מתקיים ברמת בסיס הנתונים, ולא בדף ASPX. פירוש הדבר שפחות רשומות מועברות מבסיס הנתונים לדף ASPX (רק אלו שבאמת זקוקים להן), מה שתורם לביצועים משופרים.

עבור משתמשים אנונימיים, המאפיין FavoriteGenres של ה-Profile מחזיר רשימה ריקה, במקום להוציא שגיאה. מכאן שאפילו משתמשים ללא Profile יכולים להסתכל בדף זה בבטחה. במקום לראות סקירות כלשהן, הם מקבלים הודעה שאומרת שעדיין לא הגדירו את הסגנונות המועדפים עליהם, או שעליהם להתחבר תחילה.

בסוף המטפל Page_Load נמצא קוד, שקובע אם להציג או להסתיר את ה-Repeater ואת התוית No Results.

VB.NET

```
Repeater1.Visible = Repeater1.Items.Count > 0  
plcNoRecords.Visible = Repeater1.Items.Count = 0
```

C#

```
Repeater1.Visible = Repeater1.Items.Count > 0;  
plcNoRecords.Visible = Repeater1.Items.Count == 0;
```

אם האוסף Items נשאר ריק גם לאחר קישור הנתונים של ה-Repeater החיצוני, פירוש הדבר שלא נמצאו סגנונות עבור המשתמש הנוכחי. במקרה כזה יוסתר כל ה-Repeater ויוצג תוכן הפקד Placeholder. עם זאת, אם המאפיין Count של האוסף Items גדול מאפס, יוצג ה-Repeater ויוסתר ה-Placeholder.

בפרק 13 יצרת דף בשם NewPhotoAlbum.aspx שהניח למשתמשים להכניס אלבום Gig Pics חדש. ליישום הנוכחי של דף זה יש כמה מגרעות. ראשית, כל אחד יכול להכניס אלבום חדש. אין דרך לחסום משתמשים אנונימיים מיצירת אלבום חדש והעלאת תמונות.

שנית, אין דרך קלה להוסיף או להסיר תמונות מאלבום תמונות קיים. הדבר חל גם על ה-Managers של האתר. לפיכך, גם אם מישו מעלה תמונות שאינן הולמות, אין דרך קלה למחוק אותן. כעת, כשאתה יודע יותר אודות אבטחה, קל למדי ליישם זאת, כפי שתראה בתרגיל הבא.

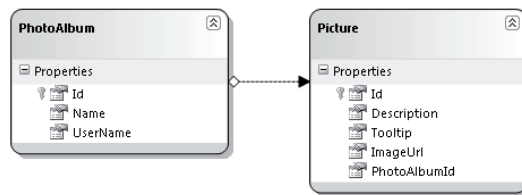
נסה זאת אפשר למשתמשים לנהל את אלבומי התמונות שלהם

בתרגיל זה תראה כיצד לחסום את הדפים NewPhotoAlbum.aspx ו- ManagePhotoAlbum.aspx מפני משתמשים שלא אומתו. בנוסף, תלמד כיצד לרשום את שם המשתמש שיצר את אלבום התמונות ולהשתמש בשם זה מאוחר יותר, כדי לאפשר למשתמשים לשנות את אלבומי התמונות שלהם.

1. פתח את סייר בסיס הנתונים, הרחב את בסיס הנתונים PlanetWrox, ואתר את הטבלה PhotoAlbum. לחץ על הטבלה לחיצה ימנית ובחר Open Table Definition. הוסף עמודה חדשה בשם UserName, הגדר את סוג הנתונים שלה ל-nvarchar(256) והשאיר את האפשרות Allow Nulls נבחרת (בטבלה זו נמצאים כבר אלבומי תמונות ללא UserName תקף, ולכן אינך יכול בשלב זה להפוך את העמודה לנדרשת, אלא אם כן תמחק תחילה את אלבומי התמונות האלה ואת התמונות הקשורות אליהם מבסיס הנתונים). שמור את השינויים וסגור את הטבלה.

2. פתח את הקובץ PlanetWrox.dbml (מסוג LINQ to SQL Classes) מהתיקייה App_Code. בתרשים, לחץ לחיצה ימנית על המחלקה PhotoAlbum ובחר Delete. מסיר בסיס הנתונים, גרור טבלה PhotoAlbum על התרשים. כתוצאה ממחיקת הטבלה והוספתה מחדש, מופיעה כעת העמודה UserName בבסיס הנתונים כמאפיין של המחלקה PhotoAlbum (ראה תרשים 10-16).

3. שמור את השינויים וסגור את קובץ DBML.



תרשים 10-16

4. פתח את הקובץ web.config והוסף את שני האלמנטים <location> הבאים כדי לחסום למשתמשים אנונימיים את הגישה לשני הקבצים שהוזכרו לעיל:

```
</location>
<location path="ManagePhotoAlbum.aspx">
  <system.web>
    <authorization>
      <deny users="?"/>
    </authorization>
  </system.web>
</location>
<location path="NewPhotoAlbum.aspx">
  <system.web>
    <authorization>
```

```

        <deny users="?"/>
    </authorization>
</system.web>
</location>
</configuration>

```

5. שמור את השינויים וסגור את הקובץ web.config.

6. פתח את NewPhotoAlbum.aspx ב- Design View. אתר את הפקד LinqDataSource והגדר מטפל אירוע עבור אירוע Inserting שלו. הוסף את הקוד הבא למטפל שיצרה עבורך VWD:

VB.NET

```

Protected Sub LinqDataSource1_Inserting(ByVal sender As Object, _
    ByVal e As System.Web.UI.WebControls.LinqDataSourceInsertEventArgs) _
    Handles LinqDataSource1.Inserting
    Dim myPhotoAlbum As PhotoAlbum = CType(e.NewObject, PhotoAlbum)
    myPhotoAlbum.UserName = User.Identity.Name
End Sub

```

C#

```

protected void LinqDataSource1_Inserting(object sender, LinqDataSourceInsertEventArgs e)
{
    PhotoAlbum myPhotoAlbum = (PhotoAlbum)e.NewObject;
    myPhotoAlbum.UserName = User.Identity.Name;
}

```

7. פתח את Default.aspx מהתיקייה PhotoAlbums ועבור ל- Markup View. גלול כל הדרך מטה עד לתחתית הקובץ, וממש לפני תגית הסגירה </ContentTemplate> של Edit Photo Album גרור פקד HyperLink. הכנס את הטקסט UpdatePanel של AJAX, גרור פקד HyperLink. הפקד HyperLink שים שני אלמנטים
 בין תגיות הפתיחה והסגירה שלו. בחזית הפקד HyperLink כדי ליצור מעט רווח.

```

<br /><br /><asp:HyperLink ID="HyperLink1" runat="server">Edit Photo Album</asp:HyperLink>
</ContentTemplate>

```

8. עבור ל- Design View, בחר בפקד ListView והגדר מטפל לאירוע DataBound שלו, שמתעורר כשהפקד מסיים את קישור הנתונים. השלם את מטפל האירוע עם הקוד הבא:

VB.NET

```

Protected Sub ListView1_DataBound(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles ListView1.DataBound
    Dim photoAlbumId As Integer = Convert.ToInt32(DropDownList1.SelectedValue)
    Using myDatabaseContext As PlanetWroxDataContext = New PlanetWroxDataContext()

        Dim photoAlbumOwner As String = (From p In myDatabaseContext.PhotoAlbums _
            Where p.Id = photoAlbumId _
            Select p.UserName).Single()
    End Using
End Sub

```



```

If User.Identity.IsAuthenticated And (User.Identity.Name = photoAlbumOwner Or _
    User.IsInRole("Managers")) Then
    HyperLink1.Visible = True
    HyperLink1.NavigateUrl = "~/ManagePhotoAlbum.aspx?PhotoAlbumId=" & _
        photoAlbumId.ToString()
Else
    HyperLink1.Visible = False
End If
End Using

```

End Sub

C#

protected void ListView1_DataBound(object sender, EventArgs e)

```

{
    int photoAlbumId = Convert.ToInt32(DropDownList1.SelectedValue);
    using (PlanetWroxDataContext myDatabaseContext = new PlanetWroxDataContext())
    {
        string photoAlbumOwner = (from p in myDatabaseContext.PhotoAlbums
                                   where p.Id == photoAlbumId
                                   select p.UserName).Single();

        if (User.Identity.IsAuthenticated &&
            (User.Identity.Name == photoAlbumOwner || User.IsInRole("Managers")))
        {
            HyperLink1.Visible = true;
            HyperLink1.NavigateUrl = "~/ManagePhotoAlbum.aspx?PhotoAlbumId=" +
                photoAlbumId.ToString();
        }
        else
        {
            HyperLink1.Visible = false;
        }
    }
}

```

9. פתח את הקוד ברקע של ManagePhotoAlbum.aspx. הוסף את הקוד הבא למטפל Page_Load. אם המטפל עדיין אינו שם, לחץ על הדף לחיצה כפולה ב- Design View כדי לגרום ל-VWD ליצור אותו.

VB.NET

```

Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Me.Load

    Dim photoAlbumId As Integer = Convert.ToInt32(Request.QueryString.Get("PhotoAlbumId"))

    Using myDatabaseContext As PlanetWroxDataContext = New PlanetWroxDataContext()
        Dim photoAlbumOwner As String = (From p In myDatabaseContext.PhotoAlbums _
                                           Where p.Id = photoAlbumId _
                                           Select p.UserName).Single()
    End Using

```

```

If User.Identity.Name <> photoAlbumOwner And Not User.IsInRole("Managers") Then
    Response.Redirect("~/")
End If
End Using

```

End Sub

C#

```
protected void Page_Load(object sender, EventArgs e)
```

```

{
    int photoAlbumId = Convert.ToInt32(Request.QueryString.Get("PhotoAlbumId"));

    using (PlanetWroxDataContext myDatabaseContext = new PlanetWroxDataContext())
    {
        string photoAlbumOwner = (from p in myDatabaseContext.PhotoAlbums
                                   where p.Id == photoAlbumId
                                   select p.UserName).Single();
        if (User.Identity.Name != photoAlbumOwner && !User.IsInRole("Managers"))
        {
            Response.Redirect("~/");
        }
    }
}

```

10. שמור את השינויים לכל הקבצים הפתוחים (הקש Ctrl+Shift+S) ואז בקש את NewPhotoAlbum.aspx בדפדפן. אם נחוצ, התחבר עם חשבון שיצרת קודם.

11. הכנס שם חדש עבור אלבום התמונות ולחץ Insert. בשלב זה, אלבום התמונות נשמר יחד עם שם המשתמש שלך. המשך על ידי הוספת כמה תמונות לאלבום התמונות.

12. לחץ Gig Pics בתפריט הראשי או ב-TreeView, ומהרשימה הנפתחת בחר באלבום התמונות החדש שזה עתה יצרת. לאחר טעינת הדף מחדש, אמור אלבום התמונות החדש להיות מוצג, יחד עם קישור Edit Photo Album בתחתית המסך. לחיצה על הקישור תעביר אותך לדף Edit Photo Album שמאפשר לך להוסיף או להסיר תמונות מאלבום התמונות שלך.

13. לחץ על הקישור Logout בכותרת התחתונה של הדף. עבור שוב לדף Gig Pics, ומהרשימה הנפתחת בחר באלבום התמונות החדש שלך. שים לב שהקישור Edit Photo Album אינו נראה כעת.

כיצד זה פועל

התחלת תרגיל זה בהוספת עמודה עבור שם המשתמש בטבלה PhotoAlbum. העמודה מאפשרת לעקוב אחר המשתמש שיצר את אלבום התמונות, ולהציג נתונים שקשורים אליו יחד עם אלבום תמונות. למרות שיכולת להוסיף את UserName באופן ידני לקובץ LINQ to SQL, הרי שלעתים קרובות קל הרבה יותר ליצור מחדש את המחלקה כולה כפי שעשית בתרגיל זה. האם זה משתלם? התשובה תלויה בקובץ הממשי שאתה פועל עליו. אם כבר ביצעת התאמות אישיות

(לדוגמה, שינוי שמות טבלאות, קשרים וכו'), עדיף שתגדיר את המאפיין החדש באופן ידני על ידי שימוש במשטח מודל האובייקטים LINQ to SQL.

בדף New Photo Album, תוכל לגשת לשם המשתמש הנוכחי בעזרת הקוד הבא:

VB.NET

```
myPhotoAlbum.UserName = User.Identity.Name
```

C#

```
myPhotoAlbum.UserName = User.Identity.Name;
```

למחלקה Page יש מאפיין User שמייצג את המשתמש שמשוך לבקשה הנוכחית. למשתמש זה בתורו יש Identity, שבתורה מכילה את Name של המשתמש. Name מוקצה למאפיין UserName של מופע PhotoAlbum, שמאוחר מ- e.NewObject.

השם מאוחסן כעת בהצלחה בבסיס הנתונים, יחד עם שארית אלבום התמונות. מה שנותר לעשות עם השם הוא להשתמש בו. המקום הראשון שבו תשתמש בשם זה הוא דף ברירת המחדל של התיקיה PhotoAlbums. שם השתמשת בשאילתת LINQ הבאה כדי לאחזר את UserName של אלבום תמונות:

VB.NET

```
Dim photoAlbumOwner As String = (From p In myDatabaseContext.PhotoAlbums _  
                                  Where p.Id = photoAlbumId _  
                                  Select p.UserName).Single()
```

C#

```
string photoAlbumOwner = (from p in myDatabaseContext.PhotoAlbums  
                           where p.Id == photoAlbumId  
                           select p.UserName).Single();
```

קוד זה משתמש בשיטה Single כדי לאחזר את UserName עבור אלבום תמונות יחיד, אותו אלבום שצוין ב-photoAlbumId. יתרת הקוד קובעת אם הקישור Edit ייראה. הדבר מותנה בחיבור של המשתמש הנוכחי ואם הוא הבעלים של אלבום התמונות, או שהוא חבר בקבוצה Managers. בדרך זו הן הבעלים והן כל חברי Managers יכולים לשנות אלבומי תמונות קיימים. הקוד ב- ManagePhotoAlbum.aspx מבצע בדיקה דומה כדי לעצור משתמשים בלתי-מורשים מלגשת לדף באופן ישיר.

דרכים אחרות להתמודדות עם Profile

בסעיף האחרון של פרק זה תראה שתי דרכים שימושיות נוספות להתמודדות עם התכונה Profile של ASP.NET. תחילה תלמד כיצד להשתמש ב-Profile עבור משתמשים אנונימיים, ואחר כך נדון בגישה לפרופיל של משתמש שאינו המשתמש הנוכחי.

זיהוי אנונימי

קל מאוד לקבוע את התצורה של התכונה Profile, שהינה רבת-עוצמה. כל שעליך לעשות כדי לאפשר למשתמשים מחוברים גישה לפרופילים שלהם הוא ליצור כמה אלמנטים ב-web.config, מכאן והלאה, ASP.NET runtime מטפל בכל השאר. אבל מה לגבי משתמשים אנונימיים? מה אם תרצה לאחסן נתונים על מבקרים שלא נרשמו לפתיחת חשבון, או שאינם מחוברים עדיין? עבור משתמשים אלה יש צורך לאפשר זיהוי אנונימי (anonymous identification). על ידי זיהוי אנונימי עבור כל מבקר חדש באתר, ASP.NET יוצרת משתמש אנונימי בטבלה aspnet_Users. למשתמש זה מוקצית עוגייה שמקושרת לחשבון המשתמש האנונימי בבסיס הנתונים. בכל ביקור, הדפדפן שולח את העוגייה עם הבקשה, וכך הוא מאפשר ל-ASP.NET לשייך משתמש, כלומר Profile, למשתמש שהזין את הבקשה הנוכחית.

כדי לאפשר פרופיל אנונימי, עליך לעשות שני דברים, להפעיל זיהוי אנונימי, ולשנות את מאפייני הפרופיל, חלקם או כולם, כדי שייחשפו למשתמשים אנונימיים.

הפעלת זיהוי אנונימי מתבצעת על ידי האלמנט הבא ב-web.config, שמופיע מייד תחת

<system.web>

```
<anonymousIdentification enabled="true" cookieName="PlanetWroxAnonymous" />
```

המאפיין enabled מפעיל את הזיהוי האנונימי, בעוד שהמאפיין cookieName משמש כדי לקבוע ליישום שם עוגייה ייחודי.

הצעד הבא לאחר הפעלת הזיהוי האנונימי, כרוך בשינוי מאפיינים תחת האלמנט <profile> והגדרת המאפיין allowAnonymous שלהם ל-True.

```
<add name="FavoriteGenres" type="System.Collections.Generic.List`1[System.Int32]"
allowAnonymous="true" />
```

כעת ניתן לגשת למאפיין Profile זה גם מהקוד עבור משתמשים אנונימיים. אם תנסה להגדיר מאפיין Profile עבור משתמש שאיננו מחובר, מבלי להגדיר את allowAnonymous ל-True, תקבל שגיאה. לעומת זאת, הקריאה ממאפיין מבלי שהוגדרה תכונה זו פועלת היטב, כפי שראית בתרגיל נסה זאת "שימוש ב-Profile בדף Reviews".

לאחר הפעלת מאפייני Profile עבור משתמשים אנונימיים, תיווכח שקריאה מהם או כתיבה אליהם זהות לאלו של מאפייני Profile רגילים. בסעיף "תרגילים" שבסוף הפרק תמצא קוד לשינוי בורר ערכת הנושא הנוכחית, באופן שישתמש ב-Profile של משתמשים אנונימיים ומחוברים.

ניקוי פרופילים אנונימיים ישנים

אתה עשוי לתהות על מה שקורה לפרופיל של משתמש אנונימי כשהמשתמש המשוך לו נרשם לפתיחת חשבון והופך להיות משתמש רשום. התשובה היא - כלום. הפרופיל הישן מבוטל והמשתמש מקבל פרופיל חדש שמשויך לחשבון הרשום. פירוש הדבר, שהטבלה aspnet_Users עלולה להגיע לממדים גדולים כתוצאה מחשבונות ישנים של משתמשים אנונימיים שאינם

בשימוש. למרבה המזל קל לתקן זאת. בכל פעם שמשתמש הופך מאנונימי למשתמש מאומת (כלומר, כשהוא מתחבר), ASP.NET מעוררת את האירוע Profile_OnMigrateAnonymous שביכולתך לטפל בו. הטיפול באירוע מתבצע בקובץ מיוחד בשם Global.asax שמשמש לקוד שמטפל באירועים של היישום כולו (אירועים שהם ברמת האתר לכלל המשתמשים). מתוך מטפל האירוע תוכל לגשת לשני פרופילים של אותו המשתמש, הפרופיל האנונימי הישן, שעומד להיות מנותק מהמשתמש, והפרופיל החדש שמשויך למשתמש שנמצא כרגע בתהליך התחברות. תוכל אז למחוק את החשבון הישן של המשתמש ואת נתוני הפרופיל הקשור אליו, מכיוון שמעתה ואילך תשתמש בפרופיל החדש. למרות שמטפל אירוע זה אינו בשימוש באתר האינטרנט Planet Wrox, זהו המקום המושלם להעתקת נתוני פרופיל אנונימי מהפרופיל הישן אל החדש, כפי שמודגם על ידי הקוד הבא.

VB.NET

```
Public Sub Profile_OnMigrateAnonymous(sender As Object, args As ProfileMigrateEventArgs)
    Dim anonymousProfile As ProfileCommon = Profile.GetProfile(args.AnonymousID)

    ' Copy over anonymous properties only
    Profile.AnonymousProperty = anonymousProfile.AnonymousProperty

    ProfileManager.DeleteProfile(args.AnonymousID)
    AnonymousIdentificationModule.ClearAnonymousIdentifier()
    Membership.DeleteUser(args.AnonymousID, True)
End Sub
```

C#

```
public void Profile_OnMigrateAnonymous(object sender, ProfileMigrateEventArgs args)
{
    ProfileCommon anonymousProfile = Profile.GetProfile(args.AnonymousID);

    // Copy over anonymous properties only
    Profile.AnonymousProperty = anonymousProfile.AnonymousProperty;

    ProfileManager.DeleteProfile(args.AnonymousID);
    AnonymousIdentificationModule.ClearAnonymousIdentifier();
    Membership.DeleteUser(args.AnonymousID, true);
}
```

שים לב שקוד זה משתמש ב- Profile.GetProfile(args.AnonymousID) כדי לקבל מופע של הפרופיל האנונימי הקודם של המשתמש. בדרך זו משיגים פנייה לפרופיל המשתמש לפני שהוא מחובר.

הקוד ממשיך להעתיק את מאפייני הפרופיל האנונימי הקיים מהפרופיל הישן אל החדש. בדוגמה זו מועתק רק מאפיין אחד, AnonymousProperty. עם זאת, באפשרותך לשנות את הקוד באופן שיעתיק מאפיינים נוספים. שים לב שאין טעם להעתיק מאפיינים שאינם נגישים על ידי משתמשים אנונימיים. סוגים אלה של מאפיינים לא ניתנו להגדרה קודם, לפיכך אין מה להעתיק.

שלוש השורות האחרונות של הקוד מוחקות את הפרופיל הישן, מנקות את העוגייה מה-ID של המשתמש האנונימי, ולבסוף מוחקות מבסיס הנתונים את החשבון הישן של המשתמש האנונימי. כשקוד זה סיים לפעול, הפרופיל הישן הועבר בהצלחה לפרופיל החדש, וכל החומר של הפרופיל הישן נמחק בהצלחה מבסיס הנתונים ומעוגיות המשתמש.

המחלקה ProfileManager נמצאת במרחב השמות System.Web.Profile שעליך לייבא, כדי שהתרגיל הקודם יפעל כהלכה. היא מספקת לך שיטות שימושיות נוספות לפעולה עם Profile. לדוגמה, תוכל להשתמש ב-DeleteInactiveProfiles כדי למחוק פרופילים של משתמשים שלא היו פעילים במשך פרק-זמן מסוים.

הצצה לפרופילים של משתמשים אחרים

הדוגמאות שראית עד כה השתמשו ב-Profile כדי לגשת לנתונים של המשתמש הנוכחי. עם זאת, איך תנהג כשתצטרך להציג נתונים של משתמש אחר? לדוגמה, מה יקרה כשתרצה להציג ביוגרפיה של משתמש מתחת לאלבום Gig Pics? במקרה זה לא תוכל להשתמש באופן ישיר במאפיין Profile של המחלקה Page, מכיוון שהוא מספק מידע אודות המשתמש הנוכחי, ולא אודות המשתמש שיצר את אלבום התמונות.

כדי לפתור בעיה זו ישנה המחלקה ProfileCommon, שהינה מחלקת הבסיס של המאפיין Profile של המחלקה Page, שכוללת את השיטה GetProfile. אם השם שמועבר אליה נמצא, השיטה GetProfile מאחזרת Profile קיים מבסיס הנתונים. אם השם אינו מזוהה, השיטה תיצור פרופיל חדש לחלוטין. לדוגמה, כדי להשיג את הפרופיל Manager תוכל להשתמש בקוד הבא:

VB.NET

```
Dim theProfile As ProfileCommon = Profile.GetProfile("Manager")
```

C#

```
ProfileCommon theProfile = Profile.GetProfile("Manager");
```

ברגע שנוצר מופע של הפרופיל, תוכל לגשת למאפיינים שלו כפי שאתה רגיל. הקוד הבא מקצה את המאפיין Bio של הפרופיל Manager למאפיין Text של הפקד Label:

VB.NET

```
lblBio.Text = theProfile.Bio
```

C#

```
lblBio.Text = theProfile.Bio;
```

היכולת לקרוא את הפרופיל של מישהו אחר היא שימושית ביותר. תוכל להשתמש בה כדי להציג כמה ממאפייני Profile למשתמשים אחרים, כפי שתראה בתרגיל האחרון של פרק זה. עם זאת, תוכל גם להשתמש בקוד דומה, כדי לעדכן את הפרופילים של משתמשים אחרים. לדוגמה, תוכל ליצור דף בקטע Management שיאפשר לך לנהל את הפרופילים של המשתמשים שנרשמו לאתר. כשאתה משנה פרופילים של משתמשים אחרים, ודא לקרוא לשיטה Save בסיום. כפי

שלמדת מוקדם יותר, שינויים ב-Profile נשמרים בבסיס הנתונים באופן אוטומטי. עם זאת, הדבר חל רק על Profile של המשתמש הנוכחי. כדי לשנות ולשמר את ה-Profile של Manager שאוחזר קודם, תוכל להשתמש בקוד הבא:

VB.NET

```
theProfile.Bio = "New Bio for the Manager account here"
theProfile.Save()
```

C#

```
theProfile.Bio = "New Bio for the Manager account here";
theProfile.Save();
```

בתרגיל הבא תראה כיצד להשתמש ביכולת זו. בתרגיל זה תתנסה בכמה מהדברים כשתציג את שם המשתמש שיצר אלבום תמונות מסוים, ואת הביוגרפיה שלו.

נסה זאת התבוננות בפרופילים של משתמשים אחרים

הדף Default.aspx בתיקייה PhotoAlbums מציג כרגע את התמונות שבאלבום תמונות מסוים. אינך יכול לראות איזה משתמש יצר את האלבום, וזו עשויה להיות תוספת נאה לדף. כדי לשפר את הדף עוד יותר, תוכל גם להציג ביוגרפיה של המשתמש יחד עם הדף.

1. פתח את הדף Default.aspx מהתיקייה PhotoAlbums ב- Markup View. גלול עד לתחתית הדף בסוף הקובץ, ואתר את שני האלמנטים `<br />` ואת הפקד `HyperLink` שהוספת בתרגיל הקודם. ממש לפניהם, גרור פקד `Placeholder` מארגז הכלים. גרור שני פקדי `Label` לתוך `Placeholder` זה, ושנה באופן ידני את הקוד כדי שיראה בסופו של דבר כך:

```
<asp:Placeholder ID="plcPhotoAlbumDetails" runat="server">
    <h2>Photo Album Details</h2>
    Created by: <asp:Label ID="lblUserName" runat="server" Text=""></asp:Label><br />
    About this user: <asp:Label ID="lblBio" runat="server" Text=""></asp:Label>
</asp:Placeholder>

<br /><br />
<asp:HyperLink ID="HyperLink1" runat="server">Edit Photo Album</asp:HyperLink>
</ContentTemplate>
```

2. עבור לקוד ברקע של הדף, ואתר בו את מטפל האירוע `DataBound` של הפקד `ListView`. מייד לאחר הבלוק `Else` (ב-`C#` `else`) שמסתיר את הפקד `HyperLink`, הוסף את שורות הקוד הבאות שמאחזרות את הפרופיל של המשתמש שיצר את האלבום התמונות, ואז עדכן את התוויות הרלוונטיות:

VB.NET

```
HyperLink1.Visible = False  
End If
```

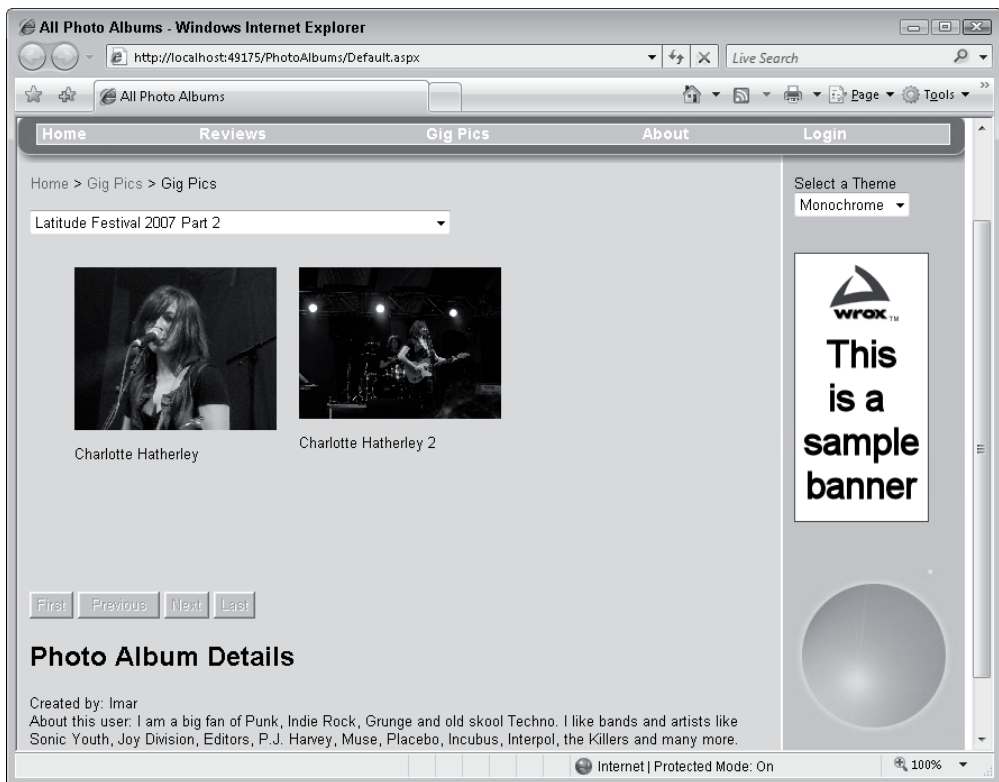
```
' Display user's details  
If Not String.IsNullOrEmpty(photoAlbumOwner) Then  
    Dim ownerProfile As ProfileCommon = Profile.GetProfile(photoAlbumOwner)  
    lblUserName.Text = photoAlbumOwner  
    lblBio.Text = ownerProfile.Bio  
    plcPhotoAlbumDetails.Visible = True  
Else  
    plcPhotoAlbumDetails.Visible = False  
End If
```

C#

```
HyperLink1.Visible = false;  
}
```

```
// Display user's details  
if (!string.IsNullOrEmpty(photoAlbumOwner))  
{  
    ProfileCommon ownerProfile = Profile.GetProfile(photoAlbumOwner);  
    lblUserName.Text = photoAlbumOwner;  
    lblBio.Text = ownerProfile.Bio;  
    plcPhotoAlbumDetails.Visible = true;  
}  
else  
{  
    plcPhotoAlbumDetails.Visible = false;  
}
```

3. שמור את כל השינויים ופתח את Default.aspx מהתיקייה PhotoAlbums שבדפדפן.
4. מהרשימה הנפתחת, בחר באלבום תמונות שיצרת, ותראה את פרטי האלבום שהופיעו. אם אינך רואה אותם, וודא שבחרת מהרשימה אלבום תמונות טרי. מכיוון שאת העמודה UserName הוספת לבסיס הנתונים בשלב מאוחר, הרי שלכמה מאלבומי התמונות אין משתמש שמשויך אליהם. אם הקטע Photo Album Details נשאר מוסתר, צור אלבום תמונות חדש והוסף לו כמה תמונות. הדבר יבטיח את הקיום של אלבום תמונות אחד לפחות שמוגדר עם המאפיין UserName. אם תבחר כעת באלבום התמונות מהרשימה, אתה אמור לראות את Photo Album Details כפי שהם מוצגים בתרשים 11-16.



תרשים 16-11

כיצד זה פועל

ברוב הקוד של תרגיל זה כבר דנו קודם. לאחר הוספת כמה פקדי Label בדף Photo Album, אחזרת את הפרופיל של בעל האלבום בעזרת הקוד הבא:

VB.NET

```
Dim ownerProfile As ProfileCommon = Profile.GetProfile(photoAlbumOwner)
```

C#

```
ProfileCommon ownerProfile = Profile.GetProfile(photoAlbumOwner);
```

קוד זה מאפשר לפנות לפרופיל קיים תוך שימוש ב-`GetProfile`. המחלקה שמוחזרת היא מסוג `ProfileCommon`, סוג הנתונים הבסיסי של המאפיין `Profile`. כאשר קיבלת את ההפניה הרצויה, הרי שהשימוש בה זהה כמעט לחלוטין לפעולה עם פרופילים רגילים. ההבדל היחיד הוא בכך שחובה עליך לקרוא ל-`Save` כדי לשמור בבסיס הנתונים את השינויים שנעשו ב-`Profile`, כפי שראית קודם.

עצות מעשיות בנושא התאמה אישית

הרשימה הבאה מספקת כמה עצות בנושא התאמה אישית:

- ⊙ אל תנסה לגשת ל-Profile בדף Login, מכיוון שהוא אינו זמין עדיין. מופע של Profile נוצר מוקדם במחזור החיים של הדף. לפיכך, כשפקד Login מאמת אותך בדף Login, זה מאוחר מדי כדי לשייך Profile של המשתמש לבקשה הנוכחית. במקום זאת, השתמש בשיטה GetProfile של ProfileCommon, או פנה אל דף אחר.
- ⊙ בחן בתשומת לב מה לאחסן ב-Profile ומה מוטב לאחסן בטבלאות בסיס הנתונים שלך. למרות שמבנה הרשומה-היחידה ש-ASP.NET משתמשת בו כדי לאחסן את הפרופיל מציע פתרון מהיר ונוח, הוא אינו הפתרון היעיל ביותר, בפרט לא כשמדובר בכמויות נתונים גדולות. אל תנסה לאחסן סקירות שלמות או אפילו אלבומי תמונות ב-Profile, אלא השתמש במקום זאת בטבלאות בסיס הנתונים שלך.
- ⊙ היישום הנוכחי של Profile מקשה על אחזור נתונים בשאליות שלך מתוך בסיס הנתונים aspnet_Profile. לדוגמה, קשה לענות על שאליות כמו "תן לי את כל המשתמשים שמעדיפים את הסגנון Rock", מכיוון שכל הנתונים מאוחסנים בעמודה אחת. כדי לעקוף בעיה זו, תוכל להשתמש בספק Profile אחר שתוכל לטעון מסעיף Sandbox של האתר הרשמי של ASP.NET בכתובת: www.asp.net/downloads/sandbox/.

סיכום

בפרק זה למדת כיצד להשתמש בתכונה Profile של ASP.NET 3.5 כדי לאחסן נתונים שקשורים למשתמש. אפשר להשתמש ב-Profile כדי לעקוב אחר נתונים של משתמשים מאומתים, אבל גם של משתמשים אנונימיים.

הגדרת פרופיל היא פעולה ישירה למדי. יש צורך ליצור אלמנט <profile> בקובץ web.config, עם אלמנט <properties> כבן, ואז להוסיף מאפיין אחד או יותר תוך שימוש באלמנטים <add />. כדי לקבץ מאפיינים קשורים, השתמש באלמנט <group>.

לאחר הגדרת הפרופיל ניגשים למאפיינים שלו באמצעות המאפיין Profile של המחלקה Page. גישה זו היא תמיד אל הפרופיל של המשתמש הנוכחי. שינויים כלשהם בפרופיל זה נשמרים באופן אוטומטי בסיום מחזור החיים של ASP.NET.

בהתאם לעיצוב, מאפייני Profile נגישים למשתמשים מחוברים בלבד. עם זאת, תוכל לשנות זאת בקלות על ידי הפעלת זיהוי אנונימי.

כדי לגשת ל-Profile של משתמש שאיננו זה שמשויך לבקשה הנוכחית, תוכל להשתמש בשיטה GetProfile. שינויים כלשהם שיעשו בפרופיל זה לא יישמרו באופן אוטומטי, ותצטרך לקרוא ל-Save כדי לשלוח את השינויים לבסיס הנתונים.

כעת, כשהדף שלך מכיל יותר ויותר קוד, גדל הסיכוי שבאגים ובעיות אחרות יתגנבו לתוך היישום. בפרק הבא תלמד כיצד לטפל בחריגים כדי למנוע בעיות אלו מלהופיע בממשק המשתמש. תלמד גם כיצד לנפות את הקוד כדי לתקן בעיות לפני שהן קורות.

תרגילים

1. התכונה של ערכת הנושא המועדפת שיצרת מוקדם יותר היא מועמדת מצוינת למאפיין Profile. איזה מאפיין תצטרך להוסיף לפרופיל ב-web.config כדי שזה יקרה?
2. כדי להגדיר את ערכת הנושא המועדפת ב-BasePage, עליך לגשת ל-Profile בדרך מיוחדת. במקום לגשת לפרופיל דרך המחלקה Page, תיגש אליו דרך HttpContext בדרך זו:

VB.NET

```
Dim myProfile As ProfileCommon = CType(HttpContext.Current.Profile, ProfileCommon)
```

C#

```
ProfileCommon myProfile = (ProfileCommon) HttpContext.Current.Profile;
```

- על פי קוד זה, איך תוכל לכתוב מחדש את Page_PreInit באופן שישגי את ערכת הנושא המועדפת מ-Profile ולא מהעוגייה?
3. מה עוד תצטרך לשנות, כדי לסיים את אחסון ערכת הנושא ב-Profile ולא בעוגייה מותאמת?

פרק 17

טיפול בחריגים, ניפוי ומעקב

(Exception Handling, Debugging, and Tracing)

לא ניתן להכין חביתה מבלי לשכור ביצים, ולא ניתן לכתוב קוד מבלי ליצור גם באגים. אין זה משנה כמה תתאמץ וכמה אתה טוב, הקוד שלך יכיל בעיות שישפיעו על התנהגות אתר האינטרנט.

ברור מאליה שיש לשאוף למזעור השפעת באגים אלה (שגיאות), כשהמטרה היא "אפס בעיות". כדי לסייע לך בכך, מספקים הן ASP.NET runtime והן Visual Web Developer כמה כלים.

קודם כל, השפות שנתמכות על ידי .NET. מיישמות **טיפול בחריגים** (exception handling), שהינה מתודולוגיה לזיהוי ולטיפול בשגיאות שמתרחשות בזמן ביצוע. על ידי טיפול נכון בשגיאות אלו תוכל להציג למשתמשים הודעות שגיאה ידידותיות. בו בזמן, תוכל גם לנהל רישום של השגיאות, והדבר ייתן לך הזדמנות לתקן אותן לפני שהן מתרחשות שנית. בפרק זה תלמד כיצד הטיפול בחריגים מתבצע וכיצד לתעד את השגיאות ולהעבירן בדואר אלקטרוני.

לפני שהקוד עובר לייצור, עליך לכתוב אותו ולנפות את השגיאות בו (debug). כדי לעזור בניפוי הקוד, VWD מגיעה עם ערכת כלים עשירה שכוללת דרכים למעבר בקוד שורה אחר שורה, להתבונן במשתנים ובאובייקטים בזמן ביצוע, ואפילו להכניס שינויים תוך כדי ביצוע. ערכת הכלים מספקת גם מידע רב-ערך על נתיב הביצוע של היישום בתוך הקוד: נתיבי המעבר בין הוראות קוד שונות, דרך שיטות, מטפלי אירועים, משפטי If ו-Else וכן הלאה.

פרק זה עוסק בהרחבה בנושאים הבאים:

- ⊙ כיצד לכתוב קוד שמסוגל לגלות ולטפל בשגיאות שמתרחשות בזמן ביצוע, ובאותו הזמן להגן על המשתמשים ממסכי שגיאה לא מובנים ולתת להם הסבר מובן אודותיה.
- ⊙ כיצד לגלות שגיאות שמתרחשות בעת הייצור (run time) ולנקוט באמצעי נגד.
- ⊙ מהו ניפוי ואיזה כלי ניפוי מציעה Visual Web Developer.
- ⊙ באיזה כלים תוכל להשתמש כדי להשיג מידע על המערכת שלך כשזו פועלת בסביבת פיתוח או ייצור.

טיפול בחריגים

בכל פעם שאתה כותב קוד, יש סיכוי שדברים לא יפעלו כפי שמצופה מהם, והתוצאה תהיה קוד שאינו תואם, קריסה, או התנהגות בלתי צפויה של היישום. דברים יכולים להשתבש מסיבות רבות: שגיאת כתיב בקוד, קריסה פתאומית של שרת בסיס הנתונים שאתה מתחבר אליו, מחיקה של רשומות מטבלת בסיס נתונים. ייתכן בלכול בלוגיקת היישום שגורם לניסיון למחוק מטבלת בסיס נתונים רשומה שיש עדיין רשומות אחרות שמקושרות אליה, או ניסיון לכתוב קובץ לתיקייה ללא הרשאה מתאימה. יכולים גם להיות נתונים שגויים שהוכנסו על ידי משתמשים ולא נחסמו כראות, וכן הלאה.

כדי להבין את הבעיות האלו ולחשוב על דרכים לצפות, למנוע ולטפל בהן, עליך להבחין תחילה בין הסוגים השונים של שגיאות שעלולות לקרות ביישום אינטרנט. לאחר שתבין את ההבדלים העיקריים, תמשיך בדיון בדרכים למנוע ולפתור את השגיאות שנחשפו.

סוגים שונים של שגיאות

ניתן לסווג את השגיאות לקבוצות הכלליות הבאות:

- ⊙ **שגיאות תחביר** (syntax errors) – שגיאות שנגרמות כתוצאה משגיאות כתיב, או השמטת מילות מפתח, לדוגמה.
- ⊙ **שגיאות לוגיקה** (logic errors) – שגיאות ביישום שלכאורה מתבצע כהלכה אבל מייצר תוצאות בלתי-צפויות או בלתי-רצויות.
- ⊙ **שגיאות זמן ביצוע** (run-time errors) – שגיאות שגורמות ליישום להיכשל או להתנהג באופן בלתי-צפוי בזמן הביצוע.

כל אחת מקטגוריות אלו נדונה בסעיפים הבאים, שכוללים גם מידע על מניעתן ותיקונן.

שגיאות תחביר

שגיאות תחביר, או **שגיאות הידור** (compile errors) הן הקלות ביותר לגילוי ולתיקון מכיוון שהן קורות בזמן פיתוח. ה-IDE מתריע כשמתרחשת שגיאה, ולעתים קרובות הוא מונע ממך להפעיל את היישום. שגיאות תחביר נגרמות בשל שגיאות כתיב פשוטות, מילות מפתח ותווים חסרים או כפולים, וכד'. כל הדוגמאות הבאות מציגות שגיאות שנתפסות על ידי המהדר (compiler) בזמן פיתוח. פעולת ההידור היא שהופכת את הקוד הקריא בידי אדם, שנכתב ב-VB.NET או C#, לקוד מכונה (בינארי) שמוכן על ידי המחשב, ואשר ניתן לבצעו.

VB.NET

```
mailBody = mailBody.Repalce("##Name##", txtName.Text) ' Replace is misspelled  
  
Response.Write() ' Required parameter for the Write method is missing
```

```
If i > 10          ' Missing keyword Then
    ' Do something here
End If
```

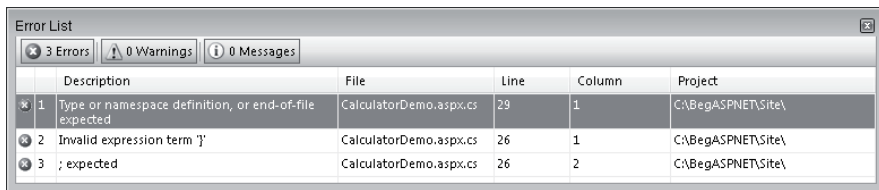
C#

```
mailBody = mailBody.Repalce("##Name##", txtName.Text); // Replace is misspelled

Response.Write();          // Required parameter for the Write method is missing

if (i > 10)                 // Missing opening brace or
                           // extraneous closing brace
    // Do something here
}
```

שגיאות הידור מוצגות תמיד ב- Error List (רשימת השגיאות אשר נגישה דרך התפריט View – Error List) שמוצגת בתרשים 17-1.



תרשים 17-1

באפשרותך לכפות על המהדר להציג רשימה עדכנית של כל שגיאות ההידור באתר. כדי לבצע זאת, בחר מהתפריט הראשי Build Web Site – Rebuild. כדי לעבור למקום שבו קרתה השגיאה במטרה לתקנה, לחץ לחיצה כפולה על השגיאה ב- Error List.

שגיאות לוגיות

שגיאות לוגיות הן לעתים קרובות קשות יותר לגילוי, מכיוון שהן עוברות את ההידור כהלכה ומתרחשות רק בזמן ביצוע הקוד. חשוב על הדוגמה השגויה הבאה:

VB.NET

```
Dim fromAddress As String = "you@yourprovider.com"
Dim toAddress As String = txtEmailAddress.Text
myMessage.From = new MailAddress(toAddress)
myMessage.To.Add(new MailAddress(fromAddress))
```

C#

```
string fromAddress = "you@yourprovider.com";
string toAddress = txtEmailAddress.Text;
myMessage.From = new MailAddress(toAddress);
myMessage.To.Add(new MailAddress(fromAddress));
```

למרות שקל לראות מה השגיאה בדוגמה זו (הכתובות From ו-To הוחלפו), הדבר עשוי להיות קשה יותר בקובץ של 250 שורות קוד. בנוסף, מכיוון שהמהדר קיבל את השגיאה, לא תבחין בה עד שתקבל בתיבת הדואר שלך (Inbox) הודעה שחשבת שנשלחה למבקרים באתר.

הדרך הטובה ביותר לעקוב אחר שגיאות לוגיות ולפתור אותן, היא להשתמש ביכולות ניפוי השגיאות המובנות של VWD. בהמשך הפרק תקבל מבט מעמיק על כלים אלה.

שגיאות זמן ביצוע

שגיאות זמן ביצוע מתרחשות בזמן ביצוע, והדבר עושה אותן לקשות ביותר למעקב. תוכל להניח שבאתר יש באג שמסתתר אי-שם בדף. עדיין לא מצאת אותו, אבל אחד המבקרים נתקל בו וקיבל הודעת שגיאה לא נעימה (נרחיב על כך במועד מאוחר יותר). מה תוכל לעשות? ככל הנראה לא הרבה, מכיוון שיש סבירות גבוהה שהמבקר אפילו לא יטרח לדווח לך על השגיאה.

לפיכך, חשוב שתהיה לך אסטרטגיה טובה לטיפול בשגיאות, שתאפשר להימנע משגיאות כשרק אפשר, לטפל בהן, ולרשום את המידע הרלוונטי כשהן מתרחשות.

הסעיף הבא עוסק בגילוי שגיאות, חריגים (exceptions) בשפה של .NET, ובטיפול בהן. מאוחר יותר בפרק תלמד כיצד לרשום שגיאות ולהגן על המשתמשים מפני דפים "מכוערים" עם הודעות שגיאה מפורטות.

תפיסת חריגים והטיפול בהם

בדרך כלל, כשמתרחשת שגיאה רצינית, מוצגת למשתמש הודעת שגיאה. לדוגמה, אם תנסה לשלוח הודעה לשרת דואר שאינו פעיל, או שאינו מאפשר לך להתחבר אליו, תקבל הודעה שתספק פרטים אודות השגיאה. כברירת מחדל, השגיאה תגיע כל הדרך עד לממשק המשתמש, ושם תוצג במה שנקרא "מסך המוות הצהוב" (Yellow Screen of Death), אנלוגיה ל"מסך המוות הכחול" של Windows שמתקבל כש-Windows קורסת. בתרגיל הבא תראה דוגמה ממשית של שגיאה כזו.

כמובן שהיה טוב הרבה יותר אם תוכל לצפות את השגיאה ולכתוב קוד שימנע את הופעתה בממשק המשתמש. תוכל, לדוגמה, להציג למשתמשים הודעה מעוצבת יפה שתודיע להם שהודעת הדואר אינה יכולה להישלח כרגע.

למרבה המזל, תמיכה בתרחישים מסוג זה משולבת עמוק בשפות התכנות של .NET. כמו C# ו-Visual Basic. בשפות אלו תוכל להשתמש בבלוקים Try Catch Finally (try catch finally ב-C#) שבהם קוד בעל פוטנציאל לשגיאה נעטף בבלוק Try.

כשיש שגיאה במהלך הביצוע, יתרת הקוד שנמצאת בבלוק Try אינה מבוצעת, וניתן להפעיל במקום זה קוד שנמצא בבלוק Catch שמותאם לטיפול בשגיאה. אפשר שיהיו בלוקים Catch רבים שכולם מטפלים בשגיאות מסוימות, אבל רק אחד מהם יופעל, כפי שתראה בתרגיל מאוחר יותר.

לאחר בלוק Try או Catch יכול לבוא בלוק Finally. הקוד בבלוק Finally מופעל תמיד, מבלי להתחשב אם קרתה שגיאה או לא.

הבלוקים Catch ו-Finally הם רשות, אם כי תמיד תצטרך לפחות אחד מהם.

בקוד שנמצא בדוגמה הבאה מנסים לשלוח דואר אלקטרוני וקובעים את הערך של המשתנה userMessage עבור המאפיין Text של פקד Label. למשתנה userMessage מוקצה ערך בבלוק Try (כשהקוד מתבצע כהלכה), או בבלוק Catch (כשמרחשת שגיאה). כך או כך, הערך של userMessage מוקצה תמיד ל-Label בבלוק Finally.

VB.NET

```
Dim userMessage As String
Try
    mySmtpClient.Send(myMessage)
    userMessage = "Message sent"
Catch ex As Exception
    userMessage = "An unknown error occurred."
Finally
    lblMessage.Text = userMessage
End Try
```

C#

```
string userMessage;
try
{
    mySmtpClient.Send(myMessage);
    userMessage = "Message sent";
}
catch (Exception ex)
{
    userMessage = "An unknown error occurred.";
}
finally
{
    lblMessage.Text = userMessage;
}
```

בדוגמת קוד זו, הבלוק Catch מוגדר לטיפול בשגיאה מסוג System.Exception, מחלקת הבסיס של כל החריגים בסביבת העבודה ASP.NET. השגיאה נשלחת לבלוק Catch במשתנה ex (משתנה זה מכיל מידע מפורט על השגיאה). מכיוון שדוגמה זו אינה משתמשת כלל במשתנה ex, אין לך צורך במשתנה ותוכל להשמיטו לחלוטין.

VB.NET

```
Catch
    userMessage = "An unknown error occurred."
End Try
```


C#

```
catch
{
    userMessage = "An unknown error occurred.";
}
```

היכולת לציין סוג של Exception שימושית למדי, כשחושבים על כך שבקוד עלול להיות יותר מחריג אחד. במקרה זה, אפשר שיהיו בלוקים Catch רבים עבור סוגים שונים של חריגים. הקוד הבא מסוגל לטפל בשגיאה מוגדרת מסוג SmtptException שעלולה להתרחש במהלך פעולת שליחת דואר אלקטרוני, ומסוגל גם לתפוס את כל שאר השגיאות תוך שימוש בבלוק Catch כללי.

VB.NET

```
Try
    mySmtptClient.Send(myMessage)
Catch smtpException As SmtptException
    lblMessage.Text = "Sorry, an error occurred while sending your message."
Catch ex As Exception
    ' Something else went wrong.
End Try
```

C#

```
try
{
    mySmtptClient.Send(myMessage);
}
catch (SmtptException smtpException)
{
    lblMessage.Text = "Sorry, an error occurred while sending your message.";
}
catch (Exception ex)
{
    // Something else went wrong.
}
```

יש חשיבות לסדר הבלוקים שמטפלים בשגיאות. NET. סורקת את רשימת הבלוקים Catch ומפעילה רק את הבלוק הראשון שתואם לסוג מסוים של חריג. בדוגמה לעיל, כשמתרחשת שגיאה מסוג SmtptException (שהיא תת-מחלקה של Exception), היא תתפס על ידי הבלוק Catch שמטפל בשגיאות מסוג SmtptException. למרות ש-SmtptException היא גם Exception, הקוד בבלוק Catch האחרון לא יופעל, מכיוון שמופעל אך ורק בלוק Catch הראשון שתואם. לכן, אילו היית הופך את הסדר של הבלוקים Catch שבדוגמה, היה מתבצע בלוק Exception היותר כללי, ואילו בלוק SmtptException לא היה מופעל.

בתרגיל הבא תראה כיצד להשתמש בקוד בבלוקים Try ו-Catch.

בתרגיל זה תנסה את הקוד Try Catch בדף נפרד שנמצא בתיקייה Demos, כדי שתוכל לעקוב מקרוב אחר התנהגותו. כשתבין כיצד הוא פועל, תשנה את פקד המשתמש ContactForm.aspx ותכלול בו את הקוד לטיפול בשגיאה. הסיבה שהקוד אינו חל באופן ישיר על פקד המשתמש היא מכיוון שהוא משתמש ב-UpdatePanel של AJAX, שכברירת מחדל מגן על משתמשים מפני הפרטים "המלוכלכים" של שגיאה.

1. צור קובץ חדש בתיקייה Demos וקרא לו ExceptionHandling.aspx. בסס את הדף על התבנית המותאמת אישית וקבע את ה-Title שלו ל-Exception Handling Demo.
2. הוסף פקד Label לשטח התוכן העיקרי וקרא לו lblMessage.
3. עבור אל הקוד ברקע והוסף לקוד משפט Imports או משפט using:

VB.NET

```
Imports System.Net.Mail
```

C#

```
using System.Net.Mail;
```

4. עבור אל Design View ולחץ לחיצה כפולה על השטח לקריאה-בלבד של הדף, כדי להגדיר מטפל אירוע לאירוע Load של הדף. הוסף את הקוד הבא למטפל האירוע. שים לב שהקוד הזה כמעט לקוד שהוספת לפקד המשתמש ContactForm.aspx. תוכל אפוא לחסוך לעצמך משהו מההקלדה על ידי העתקת חלקים מהקוד שנמצא באותו קובץ. המחרוזת עם "InvalidMailServer" שנמצאת בפונקציית הבנייה של האובייקט SmtplibClient, נועדה בכוונה תחילה ליצור שגיאה. מכיוון שאין לך שרת דואר בשם InvalidMailServer תקבל שגיאה כשתנסה לשלוח אליו הודעה.

VB.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Me.Load
```

```
Dim myMessage As MailMessage = New MailMessage()  
myMessage.Subject = "Exception Handling Test"  
myMessage.Body = "Test message body"  
  
myMessage.From = New MailAddress("you@yourprovider.com")  
myMessage.To.Add(New MailAddress("you@yourprovider.com"))  
  
Dim mySmtplibClient As SmtplibClient = New SmtplibClient("InvalidMailServer")  
mySmtplibClient.Send(myMessage)  
lblMessage.Text = "Message sent"
```

```
End Sub
```

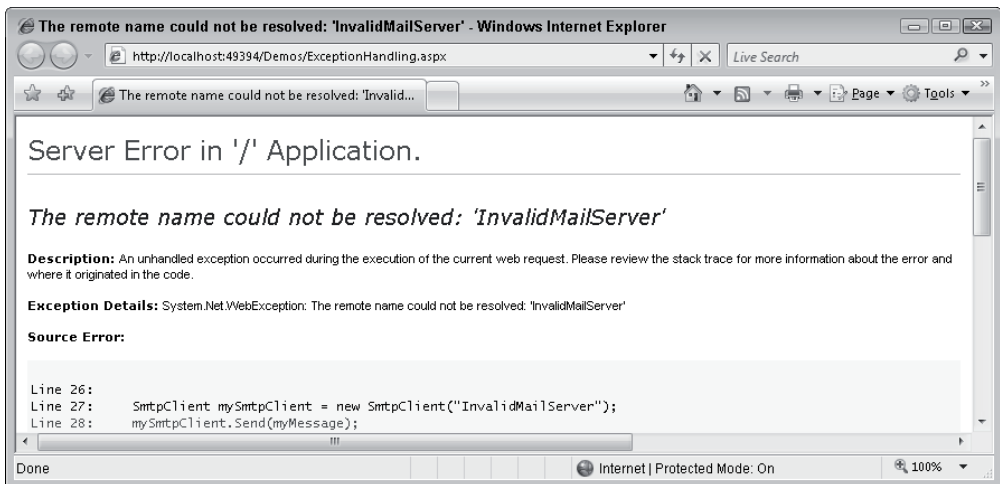
C#

```
protected void Page_Load(object sender, EventArgs e)
```

```
{  
    MailMessage myMessage = new MailMessage();  
    myMessage.Subject = "Exception Handling Test";  
    myMessage.Body = "Test message body";  
  
    myMessage.From = new MailAddress("you@yourprovider.com");  
    myMessage.To.Add(new MailAddress("you@yourprovider.com"));  
  
    SmtpClient mySmtpClient = new SmtpClient("InvalidMailServer");  
    mySmtpClient.Send(myMessage);  
    lblMessage.Text = "Message sent";  
}
```

אל תשכח לשנות את כתובות הדואר האלקטרוני From ו-To.

5. הקש Ctrl+F5 כדי לפתוח את האתר בדפדפן. אתה אמור לראות את "מסך המוות הצהוב" עם הודעת השגיאה, כפי שמוצג בתרשים 2-17. שים לב שהקטע Exception Details אומר לך שהתרחשה שגיאה מסוג System.Net.WebException.



תרשים 2-17

6. חזור ל-VWD ועטוף את הקוד ששולח את ההודעה ומגדיר את Text של הפקד Label, בבלוק Try Catch.

VB.NET

```
Dim mySmtpClient As SmtpClient = New SmtpClient("InvalidMailServer")
```

```
Try
```

```
    mySmtpClient.Send(myMessage)
```

```
    lblMessage.Text = "Message sent"
```

```
Catch ex As Exception
```

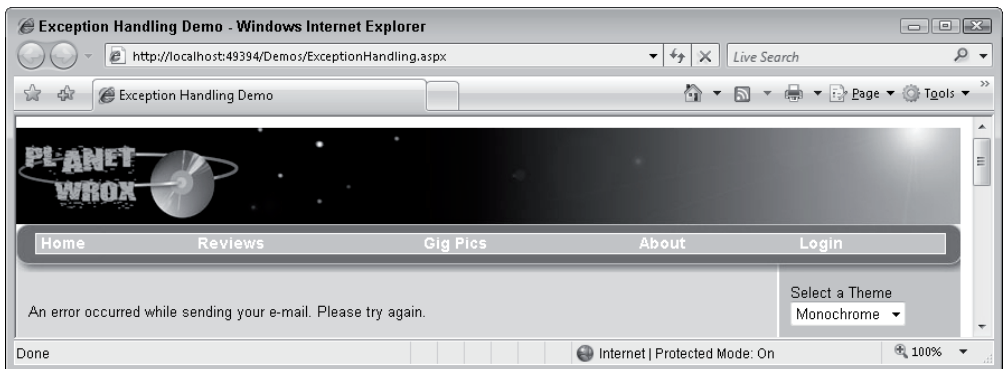
```
    lblMessage.Text = "An error occurred while sending your e-mail. Please try again."
```

```
End Try
```

C#

```
SmtpClient mySmtpClient = new SmtpClient("InvalidMailServer");  
try  
{  
    mySmtpClient.Send(myMessage);  
    lblMessage.Text = "Message sent";  
}  
catch  
{  
    lblMessage.Text = "An error occurred while sending your e-mail. Please try again.";  
}
```

7. שמור את השינויים ובקש את הדף שוב על ידי הדפדפן. אתה אמור לראות כעת הודעת שגיאה ידידותית למשתמש, שמוצגת בתרשים 3-17. השגיאה שמתרחשת נתפסת כעת על ידי הבלוק Catch. במקום לקבל דף שגיאה עם כל הפרטים הטכניים של השגיאה, המשתמש מקבל כעת הודעה ידידותית שמסבירה שמהו השתבש.



תרשים 3-17

8. חזור לקוד שלך והסר את הפרמטר מפונקציית הבנייה `SmtpClient`:

VB.NET

```
Dim mySmtpClient As SmtpClient = New SmtpClient()
```

C#

```
SmtpClient mySmtpClient = new SmtpClient();
```

על ידי הסרת שם השרת מפונקציית הבנייה, הדואר יישלח דרך שרת SMTP שהוא ברירת המחדל שהגדרת בקובץ `web.config` בפרק 9.

9. שמור את השינויים ובקש שוב את הדף. כעת תקבל הודעה שמציינת שההודעה נשלחה בהצלחה.

10. פתח את `ContactForm.ascx` מהתיקייה `Controls`, עבור לקוד ברקע שלו ועטוף את השיטה `Send` בבלוק `Try Catch` הבא:

VB.NET

```
Dim mySmtpClient As SmtpClient = New SmtpClient()
```

```
Try
```

```
    mySmtpClient.Send(myMessage)
```

```
Catch ex As Exception
```

```
    lblMessage.Text = "An error occurred while sending your e-mail. Please try again."
```

```
End Try
```

C#

```
SmtpClient mySmtpClient = new SmtpClient();
```

```
try
```

```
{
```

```
    mySmtpClient.Send(myMessage);
```

```
}
```

```
catch (Exception ex)
```

```
{
```

```
    lblMessage.Text = "An error occurred while sending your e-mail. Please try again.";
```

```
}
```

מעשה ואילך, בכל פעם שתתרחש שגיאה במהלך שליחת הדואר האלקטרוני, המשתמשים יקבלו הודעה רגילה במקום הדף עם הפרטים המלאים של השגיאה ש-NET מציגה כברירת מחדל.

כיצד זה פועל

כשקורית שגיאה בזמן ביצוע, מערכת NET. בודקת אם הקוד עטוף בבלוק Try Catch. אם זה המקרה, היא סורקת את רשימת הבלוקים Catch עבור בלוק שנועד לטפל בסוג השגיאה שהתרחשה. NET. קוראת רק לבלוק Catch הראשון שתואם את השגיאה, ומתעלמת מכל השאר. פירוש הדבר שהקוד בבלוק Try שנמצא לאחר השורה שגרמה לשגיאה - לא יבוצע.

בתרגיל זה הקוד הוגדר באופן שיטפל בסוג הכללי Exception. מכיוון שסוג זה הוא הבסיס לכל סוגי החריגים, הוא יופעל עבור כל השגיאות, מבלי להתחשב בסוגן. אבל כפי שלמדנו לפני התרגיל הזה, יכולים להיות בלוקי Catch רבים, שכל אחד מהם מגיב לסוג שגיאה ייחודי.

עם בלוקים Try Catch Finally תוכל לכתוב קוד שיעזור לך לטפל בשגיאות שצפויות להתרחש בקוד. עטיפת קוד שעלול להיכשל בבלוק Try Catch היא נוהג נכון וטוב שיאפשר לך להתמודד עם השגיאה בחן, עד כמה שניתן. להלן כמה דוגמאות לקוד שעלול להיכשל:

⊙ **שליחת דואר אלקטרוני** - שרת הדואר עלול להיות מושבת, או שאין לך הרשאת גישה אליו.

⊙ **גישה לבסיס נתונים** - השרת עלול להיות מושבת, אין לך הרשאות, אתה עלול לקבל שגיאות כתוצאה מ- foreign key constraint violations כפי שראית בפרק 14, וכן הלאה.

⊙ **ניסיון לכתוב קובץ נטען לדיסק** - הדיסק עלול להיות מלא, אין לך את ההרשאות הדרושות לכתובה לדיסק, סיפקת שם קובץ שאינו תקף.

למרות שבלוקים מסוג Try Catch טובים למניעת הגעה של שגיאות לממשק המשתמש, עליך להשתמש בהם בזהירות, מכיוון שלשימוש בהם יש מחיר. בלוק Try Catch הוא בדרך כלל איטי יותר מקוד ללא הבלוק. לפיכך, אל תשתמש בבלוק Try Catch עבור שגיאות שניתן לאתר ולמנוע בדרך אחרת. חשוב על הדוגמה הבאה לחילוק של שני מספרים:

VB.NET

```
Dim value1 As Integer = Convert.ToInt32(txtValue1.Text)
Dim value2 As Integer = Convert.ToInt32(txtValue2.Text)
Try
    result = value1 / value2
    lblResult.Text = result.ToString()
Catch ex As DivideByZeroException
    lblResult.Text = "Sorry, division by zero is not allowed."
End Try
```

C#

```
int value1 = Convert.ToInt32(txtValue1.Text);
int value2 = Convert.ToInt32(txtValue2.Text);
try
{
    result = value1 / value2;
    lblResult.Text = result.ToString();
}
catch (DivideByZeroException ex)
{
    lblResult.Text = "Sorry, division by zero is not allowed.";
}
```

בדוגמה זו, נקבע בקוד שעליו לצפות לשגיאה מסוג DivideByZeroException. שגיאה זו מתרחשת כש-value2 מכיל את הערך 0 (מצב של חלוקה ב-0!). במקום לאפשר לשגיאה לקרות, עדיף לכתוב קוד שבודק את הערך value2 לפני שהחילוק מתבצע.

VB.NET

```
If (value2 <> 0) Then
    result = value1 / value2
    lblResult.Text = result.ToString()
Else
    lblResult.Text = "Sorry, division by zero is not allowed."
End If
```

C#

```
if (value2 != 0)
{
    result = value1 / value2;
    lblResult.Text = result.ToString();
}
```

```
else
{
    lblResult.Text = "Sorry, division by zero is not allowed.";
}
```

טוב עוד יותר היה, אילו הצבת בדף את הפקד CompareValidator כדי שיוודא ש-txtValue2 לא יכיל לעולם את הערך אפס. בפרק 9 הוסבר כיצד להשתמש בפקד זה.

הבלוקים Try Catch אמנם שימושיים לתפוס שגיאות שאתה צופה אותן, אבל מה יקרה עם שגיאות שאינך צופה? איך תוכל להתמודד עם שגיאות בלתי-צפויות? מכיוון שהן בלתי-צפויות, לא תדע מתי הן תתרחשנה, ולכן קשה לכתוב קוד שיתמודד אתן.

כדי לפתור בעיה זו, תלמד בסעיף הבא כיצד לנהל רישום גלובלי של שגיאות בלתי-מטופלות ולשלוח מידע אודותן בדואר אלקטרוני. בדרך זו תדע שהשגיאות התרחשו, והדבר ייתן לך הזדמנות לתקן אותן לפני שתתרחשנה שוב.

טיפול גלובלי בשגיאות ודפי שגיאות מותאמים אישית

כדי להגן על משתמשים מהפרטים הטכניים של השגיאה, עליך לספק להם דף הודעת שגיאה ידידותית למשתמש, ועדיף שהדף יהיה בסגנון דומה לזה של היישום. למרבה המזל, ASP.NET מאפשרת להגדיר דפי שגיאה מותאמים אישית. אלה הם דפי ASPX שמוצגים למשתמש כשמתרחשת שגיאה. ביכולתך למפות סוגים שונים של שגיאות (שגיאות שרת, דף לא נמצא, בעיות אבטחה ועוד) לדפים שונים.

דף או דפי השגיאה שברצונך להציג מוגדרים באלמנט customErrors בקובץ web.config. אלמנט אופייני נראה כך:

```
<customErrors mode="On" defaultRedirect="~/Errors/Error500.aspx">
  <error statusCode="404" redirect="~/Errors/Error404.aspx" />
  <error statusCode="500" redirect="~/Errors/Error500.aspx" />
</customErrors>
```

המאפיין mode קובע אם המבקר באתר יראה דף שגיאה מפורט או לא. המאפיין תומך בשלושת הערכים הבאים:

- ⊙ On – כשמתרחשת שגיאה, כל מבקר באתר יראה תמיד דפי שגיאה מותאמים אישית.
- ⊙ Off – דפי השגיאה המותאמים אישית לא יוצגו אף פעם, ובדף יוצגו הפרטים המלאים של השגיאה.
- ⊙ RemoteOnly – למשתמשים מקומיים מוצגים הפרטים המלאים של השגיאה, לכל המשתמשים האחרים מוצגים הדפים המותאמים אישית. זו ההגדרה המועדפת, מכיוון שהיא מאפשרת לראות את הודעות השגיאה באתר במשך הפיתוח, בעוד שלמשתמשים מוצג תמיד דף השגיאה שמותאם אישית.

בין תגיות הפתיחה והסגירה של האלמנט `customErrors` הגדר אלמנט `< error />` נפרד, אחד לכל קוד מצב של HTTP שברצונך לתמוך בו. בתצורה הקודמת מוגדרים שני דפים מותאמים אישית: אחד מוצג כאשר הדף המבוקש אינו נמצא (קוד מצב 404), השני מוצג כשמתרחשות שגיאות שרת (קוד מצב 500). עבור כל קודי המצב האחרים של HTTP שלא הוגדרו במפורש, תשתמש במאפיין `defaultRedirect` כדי לציין את דף השגיאה המותאם אישית.

למרות שדפי שגיאה מותאמים אישית מגוננים על המשתמש מפרטי השגיאה, הם אינם עוזרים ליידע אותך שהתרחשה שגיאה. דפים אלה למעשה מסתירים את השגיאה הממשית ומציעים במקומה דף עם הודעת שגיאה מותאמת אישית. כדי שתקבל הודעות על שגיאות אלו, עליך לכתוב קוד שיתבונן בשגיאה ואז ישלח לך את המידע המפורט בדואר אלקטרוני. לחילופין, באפשרותך לכתוב קוד שיכניס את פרטי השגיאה לבסיס נתונים, או יכתוב אותם לקובץ טקסט.

ASP.NET מציעה מיקום מרכזי ושימושי לכתיבת קוד שיופעל כשמתרחשת שגיאה. קוד זה נכתב במטפל אירוע מיוחד בשם `Application_Error` בתוך הקובץ `Global.asax`. זהו קובץ מיוחד (שים לב לסיומת `asax`) שמאפשר לכתוב בו קוד שמתעורר בתגובה לאירועים שקורים ברחבי האתר, כמו התחלת היישום או התחלת בקשה למשאב. בתוך מטפל אירוע זה תוכל לאסוף נתונים רלוונטיים על השגיאה, להדביק אותם להודעת דואר אלקטרוני, ולשלוח אותם לתיבת הדואר שלך. הדבר נותן לך מידע מפורט על שגיאות שקרו באתר, ועוזר לתיקון הבעיה מוקדם ככל האפשר. בתרגיל הבא תלמד כיצד לכתוב קוד זה.

נסה זאת טיפול בחריגים שמתרחשים ברחבי האתר

בתרגיל זה תלמד כיצד לכתוב קוד בקובץ `Global.asax` שישלח את הודעת השגיאה בדואר אלקטרוני. בנוסף, תראה כיצד ליצור דפי שגיאה גלובליים שמוצגים למשתמשים במקרה שגיאה.

1. בסייר הפתרון, לחץ לחיצה ימנית על האתר ובחר `Add New Item`. ברשימת התבניות בחר `Global Application Class`. השאר את שמה מוגדר ל-`Global.asax` ולחץ `Add`.
2. בראש הקוד של הקובץ, מייד מתחת להנחיה `Application`, הוסף את משפט `Import` הבא. שים לב שבעת הוספת משפט `Import` כמו זה, הן `VB.NET` והן `C#` משתמשות במילת המפתח `Import`, ולא ב-`Imports` וב-`using`, שבהן משתמשים בדרך כלל בקוד ברקע:

VB.NET

```
<%@ Application Language="VB" %>  
<%@ Import Namespace="System.Net.Mail" %>
```

C#

```
<%@ Application Language="C#" %>  
<%@ Import Namespace="System.Net.Mail" %>
```

3. בתוך המטפל `Application_Error`, שאמור כבר להיות נוכח בקובץ `Global.asax`, הוסף את הקוד המוצגלל הבא שיופעל בכל פעם ששגיאה בלתי-מסופלת קורית באתר. אם המטפל איננו שם, עליך להקליד את כל הקוד מהקטע הבא, כולל החלקים שאינם מוצללים.

VB.NET

```
Sub Application_Error(ByVal sender As Object, ByVal e As EventArgs)
```

```
    If HttpContext.Current.Server.GetLastError() IsNot Nothing Then
        Dim myException As Exception =
            HttpContext.Current.Server.GetLastError().GetBaseException()
        Dim mailSubject As String = "Error in page " & Request.Url.ToString()
        Dim message As String = String.Empty
        message &= "<strong>Message</strong><br />" & myException.Message & "<br />"
        message &= "<strong>StackTrace</strong><br />" & myException.StackTrace & "<br />"
        message &= "<strong>Query String</strong><br />" & _
            Request.QueryString.ToString() & "<br />"
        Dim mySmtpClient As SmtpClient = New SmtpClient()
        Dim myMessage As MailMessage = New MailMessage("you@yourprovider.com", _
            "you@yourprovider.com", mailSubject, message)
        myMessage.IsBodyHtml = True
        mySmtpClient.Send(myMessage)
    End If
```

```
End Sub
```

C#

```
void Application_Error(object sender, EventArgs e)
```

```
{
    if (HttpContext.Current.Server.GetLastError() != null)
    {
        Exception myException = HttpContext.Current.Server.GetLastError().GetBaseException();
        string mailSubject = "Error in page " + Request.Url.ToString();
        string message = string.Empty;
        message += "<strong>Message</strong><br />" + myException.Message + "<br />";
        message += "<strong>StackTrace</strong><br />" + myException.StackTrace + "<br />";
        message += "<strong>Query String</strong><br />" + Request.QueryString.ToString() +
            "<br />";
        SmtpClient mySmtpClient = new SmtpClient();
        MailMessage myMessage = new MailMessage("you@yourprovider.com",
            "you@yourprovider.com", mailSubject, message);
        myMessage.IsBodyHtml = true;
        mySmtpClient.Send(myMessage);
    }
}
```

אל תשכח לשנות את שתי כתובות הודעת הדואר האלקטרוני שמועברות לפונקציית הבנייה של MailMessage. הכתובת הראשונה מייצגת את כתובת השולח, והכתובת השנייה היא זו של המקבל.

4. שמור את כל השינויים וסגור את הקובץ Global.asax.

5. פתח את הקובץ web.config ואתר את האלמנט customErrors שאמור להיות בקובץ. שים לב כיצד בקובץ שלך האלמנט הופך להערה ברגע שהוא מופיע כטקסט ירוק. גזור

את האלמנט והדבק אותו שוב מחוץ לתגיות ההערה (!< ו- >--> כדי שיהפוך לפעיל.
שנה את האלמנט כך:

```
<customErrors mode="On" defaultRedirect="~/Errors/Error500.aspx">
  <error statusCode="404" redirect="~/Errors/Error404.aspx" />
  <error statusCode="500" redirect="~/Errors/Error500.aspx" />
</customErrors>
```

אל תשכח לשנות את המאפיין mode מ-RemoteOnly ל-On. שמור וסגור את קובץ התצורה.

6. צור באתר תיקייה חדשה בשם Errors.

7. בתוך תיקייה חדשה זו צור שני טפסי web חדשים וקרא להם Error500.aspx ו-Error404.aspx בהתאמה. ודא ששניהם מבוססים על התבנית המותאמת אישית שלך, כדי שישתמשו בדף הראשי העיקרי ויירשו מ-BasePage.

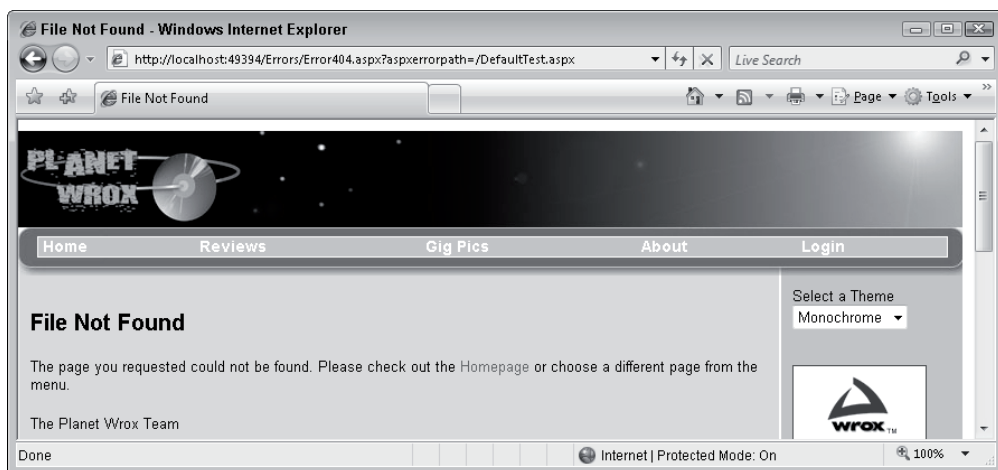
8. קבע את ערך Title של Error404.aspx ל-File Not Found. בתוך שומר מקום התוכן של התוכן העיקרי הוסף את סימני העריכה הבאים:

```
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" Runat="Server">
  <h1>File Not Found</h1>
  <p>The page you requested could not be found. Please check out the
    <a href="/" runat="server">Homepage</a> or choose a different page from the menu.</p>
  <p>The Planet Wrox Team</p>
</asp:Content>
```

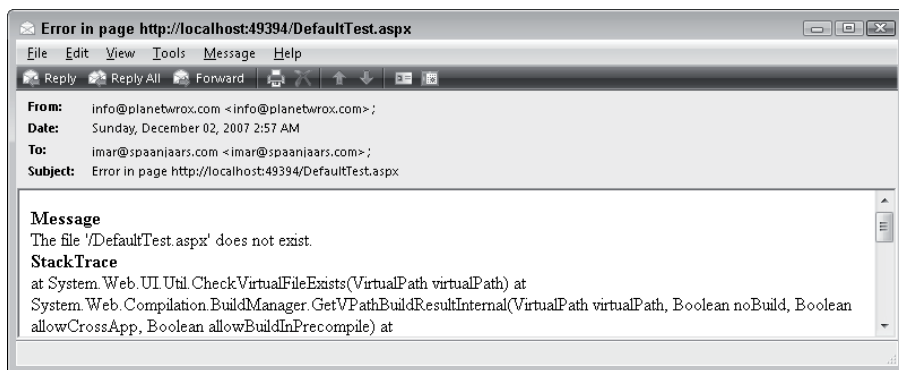
9. חזור על תהליך זה עבור הדף Error500, שנה את ה-Title שלו לערך An Error Occurred, והכנס את התוכן הבא:

```
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" Runat="Server">
  <h1>Un known error occurred</h1>
  <p>An error occurred on the page you requested. The error has been logged and
    we'll fix it ASAP.</p>
  <p>The Planet Wrox Team</p>
</asp:Content>
```

10. שמור את השינויים בכל הקבצים הפתוחים על ידי הקשת Ctrl+Shift+S, וסגור אותם. בסיום הפתרון, לחץ לחיצה ימנית על Default.aspx ובחר View In Browser. לאחר סיום טעינת הדף, בקש דף שאינו קיים, כמו DefaultTest.aspx למשל, על ידי שינוי הכתובת בסרגל הכתובת של הדפדפן למשהו כמו http://localhost:49169/DefaultTest.aspx. כעת תקבל שגיאה מכיוון שהדף DefaultTest.aspx אינו קיים. אבל, במקום דף שגיאה מפורט אתה אמור לקבל את דף השגיאה שהגדרת ויצרת בתרגיל זה, כפי שמוצג בתרשים 4-17. בנוסף, אתה אמור לקבל גם הודעה בדואר האלקטרוני שתספק פרטים נוספים על השגיאה. תרשים 5-17 מציג את ההודעה שתקבל כשתבקש דף שאינו קיים. שים לב שקטע Message מציין את שם הדף שאינו קיים.



תרשים 17-4



תרשים 17-5

כיצד זה פועל

בתרגיל זה יש שני חלקים חשובים שראוי לבחון אותם. החלק הראשון זו הדרך שבה מסתיר ASP.NET runtime את "מסך המוות הצהוב" עם פרטי השגיאה מפני המשתמש, על ידי שימוש בדפי שגיאה מותאמים. הדרך משרתת שתי מטרות. ראשית, היא עוזרת להגן על נתונים בעלי אופי פרטי כמו סיסמאות או מידע על חיבורים לבסיס נתונים, שעלולים להופיע בהודעת השגיאה. שנית, היא מגנה על המשתמשים מהודעות שגיאה נסתרות שקרוב לוודאי שממילא לא יבינו, ונותנת לך הזדמנות להציג במקומן דף שגיאה נאה למראה שמשתלב עם המראה והתחושה של האתר.

הדבר היחיד שעליך לעשות כדי שהדבר יפעל, הוא לאפשר שגיאות מותאמות בקובץ web.config ולספק את הדפים, אחד או יותר, שברצונך להציג עבור השגיאות שעלולות להתרחש באתר. אלמנט התצורה מאפשר לך להגדיר דפים שונים עבור חריגים שונים.

```
<customErrors mode="On" defaultRedirect="~/Errors/Error500.aspx">
  <error statusCode="404" redirect="~/Errors/Error404.aspx" />
  <error statusCode="500" redirect="~/Errors/Error500.aspx" />
</customErrors>
```

כש-.NET נתקלת בשגיאה 404 (לדוגמה, ביקשת דף שלא ניתן למצוא), תועבר לדף Error404.aspx. השם והתוכן של דף זה הם לחלוטין בשליטתך, וניתנת לך האפשרות לספק למשתמש הסבר משלך על מה שהשתבש. שים לב שהדבר נכון רק לגבי סוגי קבצים שרשומים ב-ASP.NET, כמו קבצי .aspx, אך הדבר לא יפעל באופן אוטומטי עבור קבצי html או תמונות.

שגיאת שרת, כמו השגיאות שראית עד כה, משויכת לקוד מצב 500. בכל פעם שהקוד נכשל מסיבה כלשהי, המשתמשים מועברים לדף Error500.aspx שמסביר שמהו השתבש.

החלק השני של תרגיל זה הוא הקוד בקובץ Global.asax שמתעורר כשמתרחשת שגיאה. במקרה זה מתעורר מטפל האירוע Application_Error. בתוך מטפל זה תוכל לאחזר על ידי הקוד הבא את השגיאה שקרתה:

VB.NET

```
If HttpContext.Current.Server.GetLastError() IsNot Nothing Then
    Dim myException As Exception = _
        HttpContext.Current.Server.GetLastError().GetBaseException()
```

C#

```
if (HttpContext.Current.Server.GetLastError() != null)
{
    Exception myException = HttpContext.Current.Server.GetLastError().GetBaseException();
```

כדי לקבל את המידע על השגיאה הבסיסית שגרמה לבעיה מלכתחילה, עליך לקרוא ל-GetBaseException() של ה-Exception שמוחזר על ידי Server.GetLastError(). מופע Exception זה, שמאוחסן במשתנה myException, נותן לך גישה למידע שימושי כמו Message ו-StackTrace. ה-StackTrace שהוצג בהודעת השגיאה שנשלחה בדואר האלקטרוני בתרגיל הקודם, הכיל מידע שאין לך כל עניין בו. עם זאת, עבור שגיאות אחרות, כמו זו שנגרמת כתוצאה מתצורה לא נכונה של שרת הדואר, או שגיאת החלוקה באפס, StackTrace נותן מידע על הקובץ שיצר את השגיאה, השיטה שגרמה לה, ואפילו את מספר השורה בקוד. מידע זה מקל על מציאת השגיאה ותיקונה.

יתרת הקוד יוצרת הודעת דואר אלקטרוני עם פרטי השגיאה, ומוסיפה מידע על מחרוזת השאלתה עם הקוד הבא:

VB.NET

```
message &= "<strong>Query String</strong><br />" & Request.QueryString.ToString() &
    "<br />"
```

C#

```
message += "<strong>Query String</strong><br />" + Request.QueryString.ToString() +
    "<br />";
```

ידיעת מחרוזת השאילתה עוזרת בניפוי בעיה אם משתמשים בערכים מהמחרוזת. תוכל להרחיב את הקוד Application_Error ולהוסיף מידע שימושי נוסף כמו עוגיות ואוספי טפסים.

למידע נוסף, הסתכל בפרויקט ELMAH (Error Logging Modules and Handlers) באתר <http://code.google.com/p/elmah/>, פרויקט קוד פתוח שמנוהל על ידי Atif Aziz ומטרתו טיפול ורישום חריגים.

היכולת לטפל ולרשום שגיאות בזמן ביצוע אמנם שימושית, אבל מוטב למנוע את התרחשות השגיאות מלכתחילה. כדי לכתוב קוד יציב עם מספר מועט ככל האפשר של באגים, אתה זקוק לכלים טובים שיעזרו לך בהבנת תהליכי הביצוע של הקוד כדי שתוכל לנפותו. חבילת התוכנה VWD כוללת כלי ניפוי מעולים שסייעו לך בתהליך זה. בסעיף הבא תכיר את הכלים ותלמד כיצד להשתמש בהם.

יסודות ניפוי השגיאות

ניפוי שגיאות הוא תהליך של גילוי ותיקון באגים בקוד. למרות שהדבר עשוי להישמע כמשהו קל, הרי שלעיתים קרובות אין זה כך. יש באגים ברורים מאוד וקלים לאיתור ולפיכך גם לתיקון. אחרים קשים הרבה יותר לגילוי ודורשים ידע אודות סביבת הביצוע של התוכנית. כלי הניפוי שכלולים בחבילת התוכנה VWD (Visual Web Developer) מסייעים לך להבין סביבת ביצוע זו, מכיוון שהם מאפשרים גישה ישירה לפעולה הפנימית של התוכנית או של דף האינטרנט.

בזמן ניפוי עם VWD הכל נעצר, לא עבורך, וביכולתך אז "להסתובב" בקוד, לחקור משתנים, להתבונן באובייקטים, לנסות שיטות, ואפילו לבצע קוד חדש. כדי לומר ל-VWD היכן לעצור יש צורך להגדיר נקודות עצירה (breakpoints) בקוד. כשהקוד תחת נקודת העצירה עומד להתבצע, VWD עוצרת את ביצוע היישום (בדרך כלל, דף אינטרנט, פקד משתמש, או קוד בתיקייה App_Code) ומחזירה את השליטה ל-VWD כדי שתוכל לאבחן את הקוד וסביבתו.

נקודת עצירה מוגדרת על ידי הקשת F9 בשורת הקוד שבה רוצים לעצור את הביצוע. במקום מקש קיצור הדרך F9 תוכל גם ללחוץ בשולי הקוד, כמו בתרשים 6-17 במקום בו מופיעה נקודה גדולה ואדומה כסימן לנקודת עצירה, או לבחור Debug ← Toggle Breakpoint מהתפריט הראשי. הקשת F9, לחיצה על הנקודה בשולי הקוד, או בחירת פריט התפריט פעם נוספת, מבטלת את נוכחות נקודת העצירה. כדי לנקות את כל נקודות העצירה באתר כולו הקש Ctrl+Shift+F9.

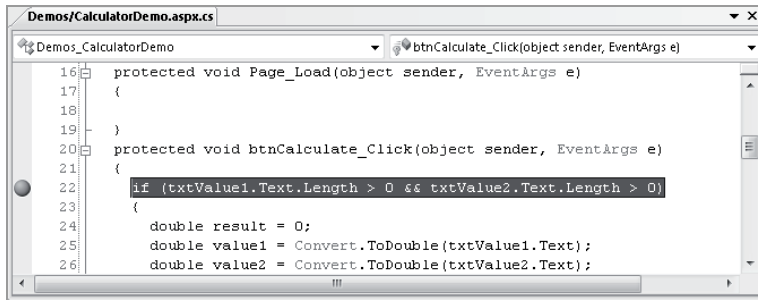
התרגיל הבא מציג את הפעולות הבסיסיות של הניפוי, כדי שתקבל מושג כיצד פועל הניפוי, ואיך ביכולתו לסייע לך. בהמשך הפרק תקבל מבט מפורט על הכלים ועל חלונות הניפוי הרבים שכלולים בתוכנה VWD.

נסה זאת ניפוי היישום

בתרגיל זה תנפה את המחשבון שיצרת בפרק קודם. אם הקובץ אינו בידך, פנה לפרק 5 שמסביר כיצד ליצור אותו, או טען את קוד המקור של הפרק הנוכחי. תרגילי הניפוי בפרק זה מניחים

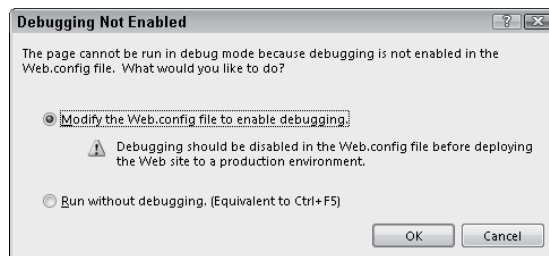
שהדפדפן בשימושך הוא Internet Explorer. כשאתה משתמש בדפדפן אחר, כמו Firefox או Opera, חווית הניפוי תהיה דומה למדי, למרות שאתה עלול למצוא ש-VWD לא תמיד מחזירה לעצמה את השליטה באופן אוטומטי בזמן הפריצה לקוד.

1. פתח את הדף CalculatorDemo.aspx מהתיקייה Demos ועבור לקוד ברקע.
2. לחץ על שורת הקוד הראשונה במטפל btnCalculate_Click שבודק את האורך של שני פקדי TextBox. הקש F9 כדי לקבוע נקודת עצירה. השורה הופכת למודגשת, כפי שמוצג בתרשים 17-6, ונקודה צבועה מופיעה בשולי חלון המסמך משמאל.



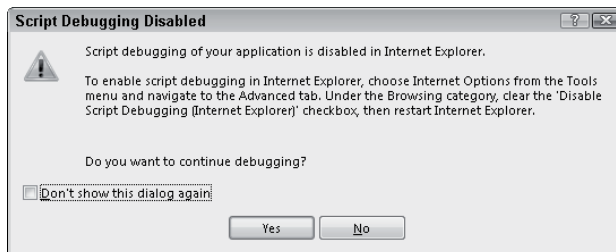
תרשים 17-6

3. הקש F5 כדי לפתוח את האתר בדפדפן, והתחל בתהליך הניפוי. לחילופין, בחר Debug ← Start Debugging מהתפריט הראשי. אם תקבל את תיבת הדו-שיח שבתרשים 17-7, לחץ OK כדי לגרום ל-VWD לשנות את הקובץ web.config.



תרשים 17-7

ייתכן שתוצג בפניך תיבת הדו-שיח שמופיעה בתרשים 17-8, הדבר תלוי בהגדרות הדפדפן שלך.



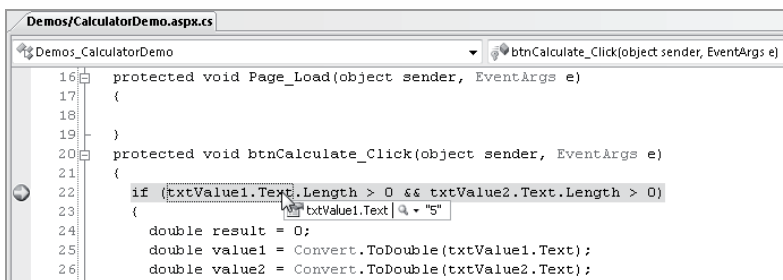
תרשים 17-8

תיבת דו-שיח זו עשויה להופיע אם לא הגדרת את Internet Explorer באופן שיאפשר ניפוי תסריטי צד-לקוח, דבר שתזדקק לו בתרגילים מאוחרים יותר בפרק. אם התיבה מופיעה, עקוב אחר ההוראות שהיא מציגה, וכשתחזור ל-VWD לחץ על הלחצן Yes. אם הגדרת את Internet Explorer בהתאם להוראות, לא תראה עוד את תיבת הדו-שיח הזו. אל דאגה אם התיבה אינה מופיעה כלל, היא מופיעה רק כשיש לשנות משהו בהגדרות.

4. הדף אמור להיטען כרגיל, ולהציג לך את שני פקדי TextBox: הפקדים DropDownList ו-Button. הכנס 5 לתיבת הטקסט הראשונה, ו-7 לשנייה, ולחץ על הלחצן Calculate. במקום לקבל את התשובה בדפדפן, תוחזר אל VWD. אם לא תוחזר ישירות ל-VWD, תצטרך לעבור אל התוכנה באופן ידני. סמל סרגל המשימות של VWD יבהב כדי למשוך את תשומת לבך.

5. ב-VWD השורה עם נקודת העצירה מודגשת כעת בצהוב. בנוסף, בשולי המסמך מופיע סמל צהוב שמציין כי שורת קוד זו עומדת להתבצע. עם זאת, לפני שהיא מתבצעת, ניתנת לך הזדמנות להסתכל בפקדים, במשתנים, ובאלמנטים אחרים שיוצרים את סביבת הביצוע. VB.NET ו-C# מתנהגות כאן באופן שונה. לדוגמה, כדי לראות את הערכים שהכנסת לפקדי TextBox באתר שנכתב ב-C#, רחף עם העכבר על פני מאפייני Text בשורות המודגשות. תראה שמופיע תיאור כלי קטן שמציג את הערך שהכנסת. כשתעשה את אותה פעולה ב-VB.NET, כל מה שתראה הוא האורך של הערך שהכנסת לתיבת הטקסט.

ההתנהגות ב-VB.NET שונה מעט. כדי לראות את הערך שהכנסת לפקד TextBox, השתמש בעכבר כדי להדגיש את הקוד txtValue1.Text ואז רחף עם העכבר על פני בחירה זו. כעת תראה את הערך מופיע בתיאור כלי, בדיוק כמו בתרשים 9-17.



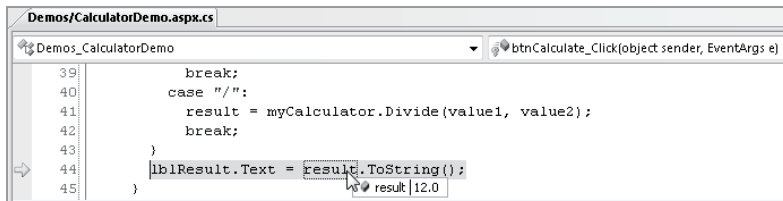
תרשים 9-17

6. רחף עם העכבר על פני כמה מהמשתנים בקוד כמו result ו-value1. שים לב שלא הופיע תיאור כלי, מכיוון שהקוד לא הגיע לנקודה שבה מוכרזים משתנים אלה. לכן, ככל שהדבר נוגע למנפה השגיאות (debugger) - הם אינם קיימים.

7. כדי להתקדם בקוד הקש F10. הפעולה תגרום לבצע את השורה ולהתקדם הלאה. המשך להקיש F10 עד שהשורה שמכריזה על המשתנה value2 תודגש. כשתעבור כעת עם העכבר על value1, יופיע תיאור הכלי, ויציין ש-value1 מכיל כעת את הערך 5.0.

8. רחף עם העכבר על המאפיין SelectedValue במשפט Select Case (VB.NET), או במשפט switch (C#). שים לב שתיאור הכלי מראה את הערך שבחרת ברשימה הנפתחת (סימן החיבור). גם אם שורת קוד זו טרם בוצעה, הרי של-SelectedValue הוקצה ערך מוקדם יותר, ולכן אתה יכול לראותו כאן.

9. הקש F10 כמה פעמים נוספות ושים לב שהקוד עובר לבלוק Select Case או switch. המשך להקיש על F10 עד שתגיע לשורה שמקצה ערך ל-lblResult. רחף עם העכבר על פני המשתנה result (כשמשתמשים ב-VB.NET, יש להדגיש תחילה את המשתנה result בעזרת העכבר) ושים לב שהוא מציג את המספר 12, תוצאת החישוב (כאן, חיבור), כפי שרואים בתרשים 10-17.



תרשים 10-17

10. לבסוף, הקש F5. הקשה זו גורמת לקוד להמשיך עד שימצא את נקודת העצירה הבאה. מכיוון שלא הגדרת נקודת עצירה נוספת בקוד, תבוצע יתרת הקוד במטפל האירוע Click, והתוצאה תוצג בדף.

כיצד זה פועל

בכל פעם שמקישים Ctrl+F5 כדי לראות דף בדפדפן, כפי שעשית עד לפרק זה, לא קורה שום דבר מיוחד. VWD פותחת את הדפדפן, שמבקש את הדף משרת האינטרנט המובנה. אבל, כשמקישים במקום זאת F5, VWD נכנסת למצב ניפוי ומכבדת את נקודות העצירה שקבעת בקוד. כאשר נתקלים בנקודת עצירה, ביצוע הקוד נעצר כדי שתוכל להתבונן בקוד ובסביבת הביצוע שלו, שנותנת לך גישה למשתנים, פקדים, שיטות, והרבה יותר מזה. שים לב שהקוד בשורה עם נקודת העצירה לא בוצע בנקודה זו. כדי לבצעו, עליך לצעוד דרכו תוך שימוש ב-F10, F11 או F5, לדוגמה.

לפני שתוכל לנפות את הקוד עליך להגדיר את תצורת היישום כדי שתתמוך בכך. תוכל לעשות זאת על ידי הגדרת התכונה debug של אלמנט ההידור ב-web.config ל-True.

```
<compilation debug="true">
```

בכל פעם שתתחיל בניפוי, והתכונה debug של <compilation /> מוגדרת ל-false, תופיע תיבת הדו-שיח שמוצגת בתרשים 7-17 ותציע להפעיל את הניפוי. כדי למנוע את התקורה הכרוכה בהגדרה זו, עליך להגדירה תמיד ל-false בשרת הייצור (production).

בתרגיל זה למדת גם להשתמש בתיאורי נתונים (data tips), חלון תיאור הכלי הקטן שמופיע כשמרחפים עם העכבר על פני משתנים שנבחרו. עבור סוגים פשוטים כמו Integer או String,

כל מה שתראה הוא ערך המשתנה. עבור סוגים מורכבים כמו תוצאות שמוחזרות משאילתת LINQ, תקבל תיאור נתונים עשיר בהרבה, שמספק הרבה יותר פרטים.

ככל שתיאורים של נתונים נראים שימושיים בניפוי, הם רק חלק קטן מתכונות הניפוי העשירות. בסעיף הבא תקבל סקירה כללית של כלי הניפוי שכלולים בתוכנה VWD.

תמיכת כלים בניפוי שגיאות

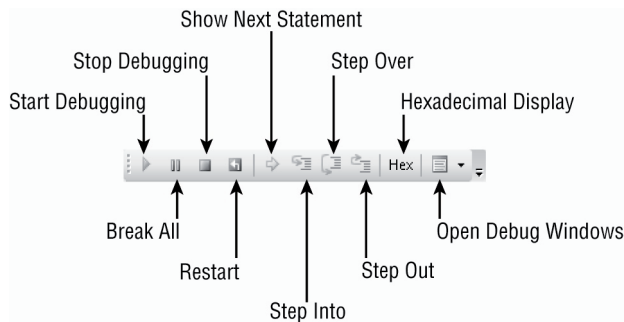
בעזרת מספר מקשי קיצור דרך ופריטי תפריט, VWD מאפשרת "לשוטט" בקוד שמנפים ולבצעו שורה אחר שורה, או לבצע בלוקים גדולים יותר כיחידת ביצוע אחת. בנוסף, IDE מספקת חלונות רבים שמאפשרים לאבחן ולשנות את סביבת הביצוע, כולל הערכים של משתנים, בזמן ביצוע. תחילה תראה איך לשוטט בקוד, ואחר כך נדון בחלונות הניפוי הרבים.

שוטטות בקוד במהלך הניפוי

כשביצוע הקוד מגיע לנקודת עצירה, תוכל להשתמש בכמה קיצורי-דרך של לוח המקשים כדי לקבוע איך להמשיך. הטבלה הבאה מפרטת את קיצורי הדרך הנפוצים ביותר.

מקש	תיאור
F5	התחלת הניפוי, כפי שמודגם בתרגיל הקודם. כשהמקש נלחץ בזמן ניפוי, הקוד ימשיך להתבצע עד לנקודת העצירה הבאה, או עד שכל הקוד יסיים להתבצע.
F11	ביצוע השורה הנוכחית, וכניסה לתוך שיטה הנקראת כעת, אם אפשר. יותר מאוחר תראה כיצד זה פועל.
F10	ביצוע השורה הנוכחית מבלי להיכנס לקוד שנקרא, אלא אם כן קוד זה עצמו מכיל נקודת עצירה.
Shift+F11	סיום ביצוע הקוד בשיטה הפנימית הנוכחית וחזרה לקוד שקרא לה.
Shift+F5	עצירת הניפוי. סגירת הדפדפן וחזרה אל VWD.
Ctrl+Shift+F5	התחלת תהליך הניפוי מחדש.

בנוסף לקיצורי הדרך של לוח המקשים, תוכל גם להשתמש בסרגל הכלים Debugging שמוצג בתרשים 11-17 ומציע פונקציונליות דומה.



תרשים 17-11

סרגל הכלים אמור להופיע באופן אוטומטי כשמתחילים בניפוי. אם הוא אינו מופיע, לחץ לחיצה ימנית על סרגל כלים קיים ובחר Debug.

בזמן שמנפים את הקוד ב-VWD, עומדים לרשותך כמה חלונות ניפוי שיידונו בסעיף הבא.

חלונות ניפוי שגיאות

חלונות הניפוי הרבים מאפשרים להתבונן במשתני היישום ואפילו לשנותם בזמן ביצוע. בנוסף, הם מספקים מידע באשר למיקומך בתוכנית ואיזה קוד בוצע קודם לכן. כל המידע הזה עוזר לך להבין את מהלך הביצוע של היישום.

הגישה לכל חלונות הניפוי היא על ידי בחירת אפשרות התפריט `Debug ← Windows`. לא כל החלונות זמינים בגרסת Express Edition של Visual Web Developer. בנוסף, כדי לגשת לרוב החלונות, על היישום להיות מוגדר לעבודה במצב ניפוי. בסעיף הבא תכיר את החלונות הזמינים השונים. בתרגיל שאחרי תיאור החלונות תינתן לך הזדמנות להפעילם ולהבין כיצד להשתמש בהם ומדוע הם מועילים לניפוי.

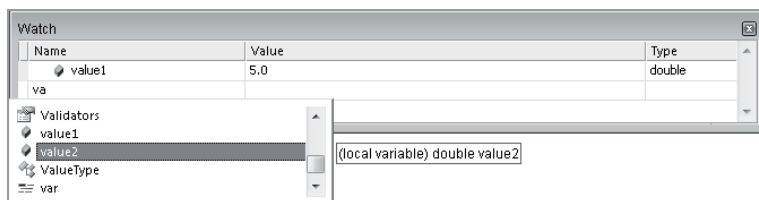
צפייה במשתנים

ידיעת ערכי המשתנים חיונית להבנת היישום. כדי לעזור לך בזה, VWD מציעה שלושה חלונות ניפוי שמספקים מידע שימושי. כל החלונות האלה תומכים בשינוי ערכי המשתנים בזמן ריצה, מאפשרים שימוש בתיאורי נתונים כדי לחדור לשורשיהם של האובייקטים, ומאפשרים גזירה והדבקה של נתונים כדי שניתן יהיה להשתמש בהם במקום אחר.

החלון Watch

חלון הצפייה (Watch) הינו ללא ספק החשוב בין החלונות, וכדאי לעיין בו בתשומת לב. הוא מאפשר להתבונן בכל המשתנים ולהתעמק בהם. תרשים 17-12 מציג את החלון Watch שבו ניתן לראות כרגע את המשתנה value1 שמשמש בדף המחשבון, בעוד שאת המשתנה value2 מוסיפים לרשימה.

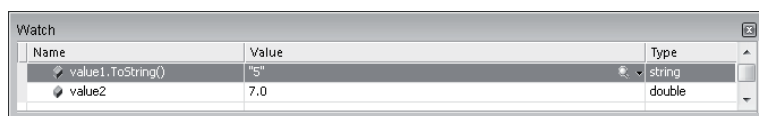
יש כמה דרכים להוסיף משתנים לחלון Watch. אפשר ללחוץ על החלון פעם אחת, ואז להתחיל בהקלדה של שם המשתנה. הקשה על Ctrl+Space תעלה את הרשימה של IntelliSense שתקל עליך את השלמת השם. לחילופין, לחץ לחיצה ימנית על משתנה בחלון המסמך, ובחר Add Watch. לבסוף, תוכל להדגיש משתנה בחלון המסמך ואז לגרור אותו לחלון Watch.



תרשים 12-17

כשהמשתנים נמצאים בחלון Watch, אפשר לשנות את ערכיהם כדי להשפיע על ביצוע הקוד. לדוגמה, תוכל לשנות את הערך של המשתנה value1 למספר אחר, ועל ידי כך תשתנה תוצאת החישוב. כדי לשנות ערך, אתר אותו בעמודה Value שבחלון ולחץ עליו לחיצה כפולה. לחילופין, לחץ לחיצה ימנית על המשתנה הנצפה ובחר Edit Value.

בנוסף להצגת ערכי המשתנים, תוכל להשתמש בחלון Watch כדי לבצע קוד. לדוגמה, תוכל לקרוא ל- ToString() עבור המשתנה value1 כדי לראות איך נראה ייצוגו כמחרוזת. כדי לבצע זאת, לחץ לחיצה כפולה על שם המשתנה בחלון כדי שניתן יהיה לערוך אותו, ואז הוסף ToString() כפי שמוצג בתרשים 13-17.



תרשים 13-17

החלון Watch אינו מגביל לקריאת ToString בלבד. ניתן לבצע בו כל ביטוי שיש לו תוצאה. עם זאת, החלון Immediate, שיידון להלן, מתאים הרבה יותר לביצוע קוד תוך כדי ניפוי.

בנוסף לחלון Watch זמינים גם החלונות Autos ו- Locals, שפועלים בדרך דומה.

החלון Autos

החלון Autos זמין רק בגרסאות המסחריות של Visual Studio ולא ב- Express Edition. הדבר אינו מהווה קושי, בגלל דמיונו הרב לחלון Watch. החלון Autos מציג את המשתנים שבהם משתמשים משפט הקוד הנוכחי ומשפט הקוד הקודם, והוא מתעדכן באופן אוטומטי כשאתה צועד דרך הקוד.

החלון Locals

החלון Locals דומה לחלון Watch ולחלון Autos. הוא מציג את כל המשתנים שנמצאים כרגע בתחום הקוד שמתבצע (דהיינו, שהקוד יכול "להגיע" אליהם). השימוש בחלון נוח מכיוון שהוא מציג את כל המשתנים הרלוונטיים מבלי לדרוש ממך להוסיף אותם באופן ידני.

חלונות אחרים

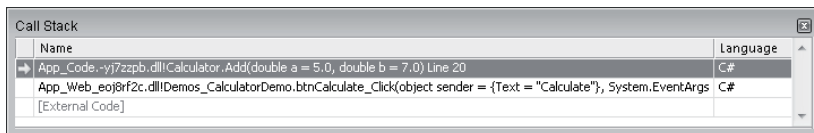
בנוסף לחלונות המיועדים לצפייה במשתנים אשר זמינים ב-VWD, יש חלונות שימושיים אחרים.

חלון Breakpoints

חלון נקודות העצירה (Breakpoints) נותן סקירה כללית על כל נקודות העצירה שהוגדרו בקוד של האתר. לרוע המזל, החלון אינו זמין ב-Express Edition, לפיכך תצטרך למצוא את נקודות העצירה באופן ידני על ידי סריקת הקוד עצמו.

חלון Call Stack

חלון מחסנית הקריאות (Call Stack) מספק מידע אודות מהלך הביצוע או הקריאה של הקוד. כל קריאה מקטע קוד אחד לאחר מוכנסת למחסנית קריאות שבאפשרותך לנווט. היא נראית מסתורית תחילה, אבל לאחר שתכיר את דרך פעולתה, היא תאפשר לך לדלג בקוד בצורה קלה למדי. תרשים 14-17 מציג את חלון מחסנית הקריאות בתוך השיטה Add של המחלקה Calculator.



תרשים 14-17

בשורה המודגשת תוכל לראות שהקוד הפעיל כרגע הוא Calculator.Add. מייד מתחתיו אתה רואה את btnCalculate_Click, שהינו מטפל האירוע בדף Calculator שקרא לשיטה Add. לחיצה כפולה על שורה בחלון מחסנית הקריאות תעביר אותך אל המיקום המתאים בקוד.

החלון Immediate

חלון ניפוי מעניין הוא החלון לביצוע מיידי (Immediate). הוא מאפשר לך לבצע קוד כאילו כתבת אותו בדף. תוכל להשתמש בחלון זה כדי לבדוק ביטויים, לראות מה מחזירות פונקציות וכדומה. לדוגמה, כשאתה במצב ניפוי, תוכל להכניס בחלון Immediate את הפקודה הבאה:

VB.NET

```
? New Calculator().Add(3, 4)
```

C#

```
? new Calculator().Add(3, 4);
```

סימן השאלה משמש לכתיבת פלט לחלון Immediate עצמו. הקוד יוצר אובייקט חדש של המופע Calculator, ומעביר את הערכים 3 ו-4 אל השיטה Add שלו. הקוד מתבצע, והשיטה Add מחזירה את הערך 7 שמוצג בחלון Immediate.

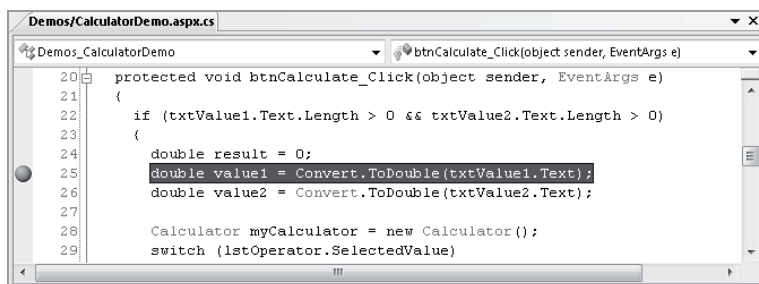
חלון זה מצוין לבדיקה מהירה של קוד. במקום לכתוב את הקוד שברצונך לבדוק בדף עצמו, תוכל להקליד אותו בחלון Immediate ולראות את הפלט שלו.

בתרגיל נסה זאת הבא תראה את חלונות הניפוי האלה בפעולה.

נסה זאת ניפוי שגיאות מקיף

בתרגיל זה תראה את כל חלונות הניפוי שהוצגו לעיל. מכיוון שמספר החלונות והאפשרויות רב, לא תראה דיון פרטני בכל צעד של התהליך. במקום זאת אתה מוזמן להתנסות. נסה להוסיף עוד משתנים לחלון Watch, הקלד קוד משלך בחלון Immediate, וכן הלאה. התנסות היא הדרך הטובה ביותר לגלות את יכולות הניפוי הרבות של VWD.

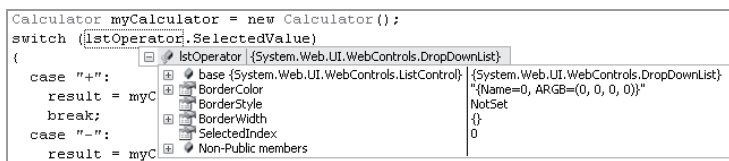
1. פתח שוב את הדף CalculatorDemo.aspx, והקש Ctrl+Shift+F9 כדי לנקות את כל נקודות העצירה שהוגדרו קודם. לחץ Yes לאישור המחקיקה.
2. לחץ על השורה שמכריזה על המשתנה value1, והקש F9 כדי לקבוע נקודת עצירה. חלון המסמך שלך אמור להיראות דומה לתרשים 15-17, שמציג את חלון המסמך עבור פרויקט בשפה C#.



תרשים 15-17

3. כדי להתחיל בניפוי היישום, הקש F5 או בחר Debug ← Start Debugging מהתפריט הראשי. הכנס את המספר 5 בתיבת הטקסט הראשונה, ודא שברשימה הנפתחת נבחר סימן החיבור, והכנס 7 בתיבת הטקסט השנייה. לחץ על הלחצן Calculate. כעת תראה ש-VWD עוצרת בנקודת העצירה שקבעת בצעד הקודם.
4. אם לא הועברת ל-VWD באופן אוטומטי, חזור לשם. רחף עם העכבר על פני המשתנה lstOperator, שמשתמשים בו מתחת לנקודת העצירה הנוכחית, ושים לב כיצד VWD מציגה תיאור נתונים עם סימן חיבור (+) בחזיתו. פירוש הדבר שניתן להרחיב את הפרט כדי לקבל מידע מפורט אודות המשתנה. תרשים 16-17 מציג תיאור נתונים מורחב.

כשמשתמשים ב-VB.NET, עליך ללחוץ תחילה לחיצה כפולה על המשתנה IstOperator שבחלון המסמך, ורק אז לרחרף על פניו עם העכבר.



תרשים 16-17

שים לב שביכולתך להרחיב גם פריטים אחרים (כמו מחלקת הבסיס של הפקד DropDownList) כדי לקבל נתונים רלוונטיים נוספים. הרחב את פריט הבסיס. ברגע שתעשה זאת, תראה מאפיינים רלוונטיים של ListControl, כולל SelectedValue שלו שמוגדר ל-"+", אם לא שינית זאת בצעד 1.

5. לחץ לחיצה ימנית על txtValue1 שנמצא בראש הקוד שבדוק את אורך פקדי TextBox, ובחר Add Watch. המשתנה מתווסף לחלון Watch ותוכל להרחיבו באופן דומה להרחבת תיאור הנתונים לעיל. הרחב את הפריט ותראה, למשל, שהמאפיין Text מוגדר לערך "5".
6. לחץ לחיצה כפולה על הערך "5", שנה אותו ל-"12" (כלול את הגרשיים) והקש Enter.
7. פתח את החלון Locals (אם החלון אינו נראה עדיין, בחר Debug ← Windows ← Locals). הקש F10 כדי לבצע את השורה שתחת נקודת העצירה. הדבר גורם לקבלת הערך מ-txtValue1, המרתו ל-double והקצאתו ל-value1. התבונן במשתנה value1 בחלון Locals. שים לב שהוא מכיל כעת 12.0. זהו הערך שהקצית בצעד הקודם למאפיין Text של תיבת הטקסט, כשהוא מומר כעת ל-double.

Locals		
Name	Value	Type
this	{ASP.demos_calculatordemo_aspx}	Demos_CalculatorDemo {ASP.demos_calculatordemo_aspx}
sender	{Text = "Calculate"}	object {System.Web.UI.WebControls.Button}
e	{System.EventArgs}	System.EventArgs
result	0.0	double
value1	12.0	double
value2	0.0	double
myCalculator	null	Calculator

תרשים 17-17

שים לב לכך שהערך של המשתנה value1 שינה את צבעו. פירוש הדבר שהפריט השתנה לאחרונה. תן דעתך גם לכך, שמכיוון שכל זה קורה בשרת בזמן postback, הדפדפן נשאר ללא שינוי, ותיבת הטקסט עדיין מציגה את הערך 5. הערך החדש יופיע רק לאחר סיום הרינדור של הדף לדפדפן.

בתרשים 17-17 אתה רואה גם את המשתנים האחרים שנמצאים כרגע בתחום של הקוד, כמו myCalculator, result ו-this (Me ב-Visual Basic) שמכילים פנייה לדף שמתבצע כרגע.

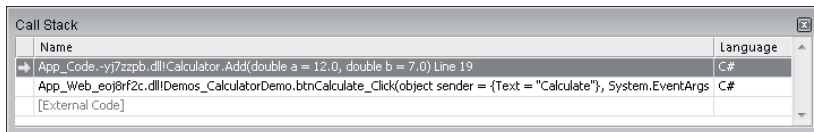
8. הקש F10 פעם אחת כדי שגם value2 יעודכן. שים לב שצבע הערך בחלון Locals השתנה לאדום כדי לציין שהערך השתנה. הדבר מקל עליך לראות איזה משתנים מושפעים על ידי משפט מסוים.

9. הקש F10 עד שתגיע לשורה שקוראת לשיטה Add של המחלקה Calculator. במקום להקיש F10 כדי לבצע שורה זו, הקש F11. כתוצאה מכך תועבר אל השיטה Add ותוכל לראות כיצד היא מבצעת את החישוב. בתוך השיטה Add תוכל לרחף על פני הארגומנטים של השיטה כדי לראות את ערכיהם, כפי שמודגם בתרשים 17-18.

```
public double Add(double a, double b)
{
    return a + b;
}
```

תרשים 17-18

10. בחר Debug ← Windows ← Call Stack כדי להעלות את חלון מחסנית הקריאות. שים לב שהשיטה Add נקראה על ידי מטפל האירוע Click של btnCalculate, שמוצג בתרשים 17-19.



תרשים 17-19

11. לחץ לחיצה כפולה על השורה השנייה בחלון מחסנית הקריאות ותחזור לדף Calculator. שים לב שכתוצאה מכך לא בוצע קוד כלשהו, כל מה שנעשה הוא להציג לך את הקוד הרלוונטי. לחץ לחיצה כפולה על השורה הראשונה ותחזור לקוד של השיטה Add.

12. הקש Shift+F11 כדי לצאת מהשיטה Add ולחזור לקוד שקרא לה בדף המחשבון. אם תעיף מבט נוסף בחלון מחסנית הקריאות תראה שהשורה עבור השיטה Add הוסרה ממחסנית הקריאות.

13. פתח את החלון Immediate (בחר Debug ← Windows ← Immediate) כדי לבדוק מעט קוד. הקלד את הקוד הבא בחלון והקש Enter:

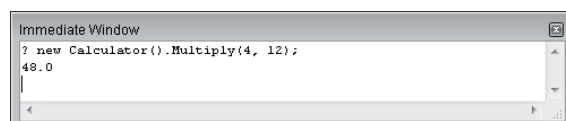
VB.NET

```
? New Calculator().Multiply(4, 12)
```

C#

```
? new Calculator().Multiply(4, 12);
```

החלון Immediate מראה את תוצאת החישוב, כפי שמוצג בתרשים 17-20.



תרשים 17-20

14. לבסוף, הקש F5. פעולה זו גורמת לביצוע יתרת הקוד בדף. השליטה מוחזרת אל הדפדפן שמציג כעת את תוצאות החישוב בפקד Label. אם הכל פועל כמתוכנן, תראה את המספר 19, הסכום של 12 (הערך החדש שהכנסת ל-`txtValue1.Text` בצעד 6) ו-7 (שהכנסת בצעד 3).

כיצד זה פועל

כפי שהודגם בתרגיל נסה זאת קודם, כשקובעים נקודת עצירה בקוד, הביצוע נעצר ברגע שהשורה עם נקודת העצירה עומדת להתבצע. משם תוכל לשווט בקוד כרצונך, לחקור משתנים ולבצע משפטים. בתרגיל זה ראית כיצד לצעוד בתוך ועל פני קוד תוך שימוש במקשי קיצור הדרך F10 ו-F11. בדרך כלל תשתמש ב-F10 כדי לבצע שורה כשאינך מעוניין בקוד הבסיסי שנקרא. ב-F11 תשתמש כשברצונך לראות את הקוד שמתבצע, כפי שראית כיצד לעשות זאת עם השיטה `Add`.

תיאורי הנתונים, החלון `Watch` והחלון `Locals`, הם כלים חשובים לבחינה ולשינוי של משתנים וערכים. לדוגמה, למרות שהכנסת 5 לתיבת הטקסט הראשונה בדפדפן, יכולת לשנות ערך זה ל-12 בשעת הניפוי. כל השינויים שנערכים בזמן הניפוי מופצים לשאר חלקי הקוד שעדיין צריך להתבצע.

החלון `Immediate` מאפשר לנסות קטעי קוד קטנים. הדבר יכול להיות שימושי לבדיקת רעיונות אחדים מבלי שתצטרך לקודד אותם בחלון הקוד ולנפותם. בתרגיל זה כתבת מעט קוד שיצר מופע `Calculator` חדש, קרא לשיטה `Multiply`, והציג את התוצאה מכיוון שכתבת את סימן השאלה.

VB.NET

```
? New Calculator().Multiply(4, 12)
```

C#

```
? new Calculator().Multiply(4, 12);
```

למרות שרוב יכולות הניפוי של ASP.NET 3.5 ושל Visual Web Developer 2008 כבר היו קיימות בגרסאות קודמות, גרסה חדשה זו מגיעה עם תכונת ניפוי חדשה ומרכזית - ניפוי תסריטי צד-לקוח.

ניפוי תסריטי צד-לקוח

עד כה השתמשת בכלי הניפוי כדי לנפות דפי ASPX וקוד ברקע. עם זאת, אין זה כל מה שנ ניתן לעשות. Visual Studio 2008 מכילה כעת תמיכה מעולה גם לניפוי JavaScript של צד-לקוח. ניפוי JavaScript של צד-לקוח דורש שימוש ב-Internet Explorer, ולא יפעל כהלכה עם דפדפנים אחרים כמו Firefox או Opera. הדבר המגניב לגבי ניפוי JavaScript של צד-לקוח ב-VWD, הוא בכך שאתה כבר יודע כיצד לבצעו. תוכל להשתמש באותם כלים מוכרים שראית בפרק זה כדי לנפות הן קוד צד-שרת והן קוד צד-לקוח.

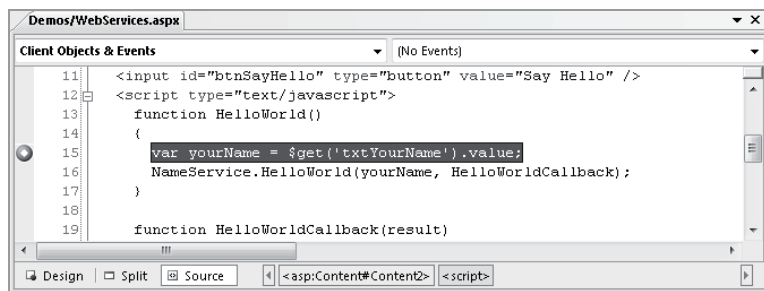
תסריט JavaScript שהדף משתמש בו בסופו של דבר בדפדפן יכול לבוא ממספר רב של מקורות שונים. JavaScript יכול להימצא בקבצי תסריטים חיצוניים, מוטבע בדף, בדף ראשי, ואפילו פקדי שרת יכולים לפלוט JavaScript משלהם. הדבר מקשה לפעמים לעצור בקוד הנכון, מכיוון שלא תמיד אתה יודע מאין הגיע. למרבה המזל יש ל-VWD פתרון נהדר לבעיה זו, התוכנה מאפשרת לקבוע נקודות עצירה ב-HTML הסופי שמוצג בדפדפן. כדי להראות לך באיזה קובץ אתה מוסיף נקודות עצירה, או איזה קוד אתה מנפה, VWD מעדכנת את סייר הפתרון ומציגה רשימה של כל הקבצים שמכילים תסריטי צד-לקוח שדרכם תוכל לצעוד כאשר תהיה במצב ניפוי. נקודות העצירה שתקבע בקבצים אלה במהלך הניפוי נשמרות כשאפשרי, והדבר הופך את הניפוי להתנסות חלקה.

הדרך הטובה ביותר ללמוד את האפשרויות החדשות הגנוזות בניפוי JavaScript של צד-לקוח היא לנסות אותן. התרגיל הבא מעמיק ומראה כיצד לנפות את דף המבחן של שירות האינטרנט שיצרת בפרק 10.

נסה זאת ניפוי תסריט JavaScript ב-Internet Explorer

כדי לבצע תרגיל זה עליך להשתמש בדפדפן Internet Explorer של מיקרוסופט, מכיוון שרבות מהתכונות שמוצגות בתרגיל פועלות רק בו. אם Internet Explorer אינו דפדפן ברירת המחדל שלך כרגע, לחץ לחיצה ימנית על דף בסייר הפתרון ובחר Browse With. מרשימת הדפדפנים בחר Internet Explorer, ואז בחר Set as Default. לחץ Cancel כדי לסגור את תיבת הדו-שיח, VWD עדיין זוכרת את הגדרות דפדפן ברירת המחדל. כשתסיים תרגיל זה תוכל לחזור לדפדפן המועדף עליך על ידי שימוש באותם צעדים בדיוק.

1. בתיקייה App_Code פתח את הקובץ NameService.cs (עבור C#) או את NameService.vb (עבור VB). אתר את שיטת האינטרנט HelloWorld וקבע נקודת עצירה בשורת הקוד הראשונה שמחזירה את הברכה המותאמת אישית. סגור את הקובץ.
2. פתח את WebServices.aspx מהתיקייה Demos, ועבור ל- Markup View. אתר את השיטה HelloWorld ב-JavaScript, לחץ על השורה שמכריזה על המשתנה yourName, והקש F9 כדי לקבוע נקודת עצירה, כפי שמוצג בתרשים 17-21.

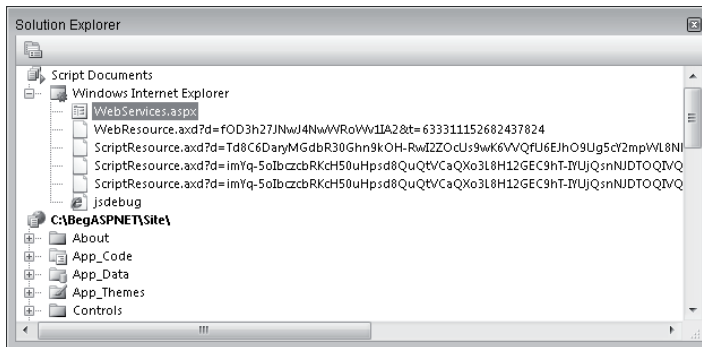


תרשים 17-21

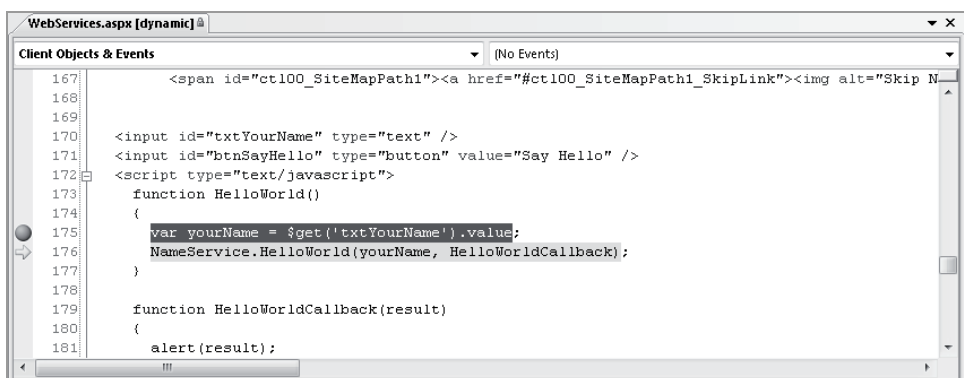
3. הקש F5 כדי להתחיל בניפוי. הדף נטען בדפדפן ומוצגים הפקדים TextBox ו-Button. הכנס את שמך ל-TextBox ולחץ על Button. כאשר תלחץ עליו תוחזר אל Visual Studio, וביצוע הקוד ייעצר בבלוק הקוד של JavaScript. הקש F10 כדי לבצע את השורה המודגשת, ולהקצות את השם שהכנסת למשתנה yourName. עכשיו רחף עם העכבר על פני המשתנה ותראה שמופיע תיאור נתונים.

4. פתח את חלונות הניפוי האחרים, ושים לב לכך שהתנהגותם זהה למה שראית קודם. תוכל להוסיף לחלון Watch משתני JavaScript כדי להתבונן בערכיהם, תוכל להכניס JavaScript לחלון Immediate לצורך הערכה, וכן הלאה. שים לב לכך שסייר הפתרון השתנה, והוא מציג את הקבצים הפעילים של צד-לקוח שמכילים תסריטים מייד מעל לפרויקט האינטרנט (ראה תרשים 17-22).

5. לחץ לחיצה כפולה על WebServices.aspx שמתחת לצומת Windows Internet Explorer שהופיע בסייר הפתרון. במבט ראשון הקובץ עשוי להיראות כמו קודם. אבל אם תתבונן מקרוב, תוכל לראות שזה איננו קובץ המקור המקורי שכלל פקדי ASP.NET מעורבים עם סימני עריכה אחרים, אלא קוד HTML הסופי שמורנדר בדפדפן. כדי להזהירך שאתה מתבונן בקובץ הסופי ולא במקור המקורי, VWD הוסיפה את הטקסט [dynamic] וסמל של מנעול לכרטיסייה של הקובץ שמעל לחלון המסמך. ראה תרשים 17-23.



תרשים 17-22



תרשים 17-23

מה שבאמת נחמד כאן הוא שלמרות שאתה מסתכל בקובץ בזמן ביצוע, VWD עדיין מסוגלת לקשור קובץ זמן-ביצוע זה עם המקור הראשוני. פירוש הדבר, שביכולתך לקבוע נקודות עצירה בקובץ זמן-הביצוע והן תיזכרנה במקור הראשוני ולפיכך תהיינה זמינות בניסיון הניפוי הבא.

6. כדי לראות כיצד זה עובד, קבע נקודת עצירה בשורה עם המשפט alert במטפל HelloWorldCallback שנראה בתרשים 17-23. ברגע שהקוד חוזר משירות האינטרנט שנקרא בתרגיל זה, תחזור שוב למטפל זה, ותוכל לחקור את הערך שהוחזר על ידי השירות.

7. הקש F5 כדי להמשיך בביצוע הקוד. השם שהכנסת בתיבת הטקסט מאוחזר ונשלח אל השיטה HelloWorld של השירות. מכיוון שבצעד 1 קבעת בשיטה נקודת עצירה, ביצוע הקוד ייעצר שוב ותוכל להתבונן במשתנה yourName שהועבר לשיטה. למרות שהתרגיל עצמו פשוט למדי, הרי שלמעשה מתרחשים בו "קסמים" רבים. הגעת מתוך קוד צד-לקוח שמתבצע בדפדפן אל קוד שמתבצע בתוך שירות האינטרנט בשרת, וכל זה נעשה מאותו IDE.

8. הקש F5 שוב ותועבר חזרה משירות האינטרנט שבצעד השרת אל קוד צד-לקוח, וכעת תוכל לראות את התוצאה של השירות במטפל HelloWorldCallback.

9. הקש F5 פעם נוספת. הקוד ישלים את ביצועו ויציג חלון התראה של JavaScript עם ברכה שכוללת את שמך, בדיוק כפי שעשה בפרק 10.

10. סגור את הדפדפן, חזור שוב ל-VWD ופתח את הקובץ WebServices.aspx על ידי לחיצה כפולה עליו בסייר הפתרון. כעת נפתח קובץ המקור הראשוני, ולא הגרסה הדינמית שראית בצעד 5 של תרגיל זה. אתר את המטפל HelloWorldCallback בקובץ WebServices.aspx. שים לב שנקודת העצירה שקבעת בצעד 6 נשמרה.

כיצד זה פועל

בתרגיל זה יש כמה נקודות מעניינות הראויות לציון. עליך להבין תחילה את המושג **קבצים דינמיים** (dynamic files) או **קבצי זמן-ביצוע** (run-time files). קבצים אלה הם התוצאה הסופית של דפי ASPX ונותנים לך תובנה לגבי קוד HTML הסופי, קוד CSS ותסריט JavaScript שמופיעים בדפדפן. זו עזרה חשובה, מכיוון שהיא נותנת מבט כולל על כל התוכן הרלוונטי. זכור, סימני העריכה הסופיים שמוצגים בדפדפן מגיעים ממגוון מקורות, כולל דפים ראשיים, דפי תוכן, קבצי CSS ו-JavaScript חיצוניים, ופקדי השרת השונים שחיים בדף. היכולת להסתכל על התוצאה המשולבת ממקום אחד עוזרת לראות איך הכל משתלב זה בזה.

נקודה חשובה אחרת שיש לזכור מתרגיל זה היא השילוב המלא של תכונות הניפוי שמציע IDE, החל מקוד צד-לקוח בממשק המשתמש וכלה בקוד שבשרת. כדי להקל על קביעת נקודות עצירה, VWD אינה מגבילה אותך להוספתן בדפים בזמן העיצוב בלבד, אלא מאפשרת לקבוע אותן בקבצי זמן הביצוע הדינמיים. כשאתה מפסיק את הניפוי, VWD עוקבת אחר נקודות

העצירה החדשות, מוצאת לאיזה קבצי מקור הן שייכות, ומוסיפה אותן בתוכם, כדי שתהיינה זמינות לסבב הניפוי הבא.

לתשומת לבך, אם לא הבחנת בכך – הכל מתרחש באותו מחשב ובאותו IDE ולכן, כשמנפים בדרך זו נחצים גבולות רבים. VWD מאפשרת לנפות תסריטי צד-לקוח בדפדפן, כדי שתוכל להתחבר אליהם עוד לפני שליחת נתונים כלשהם לדפדפן. כשהקשת F5 בצעד 7 הקוד המשיך להתבצע ושלח את הערך לשרת, כדי שהשיטה HelloWorld של השירות NameService תוכל להשתמש בו. ברגע ששיטה זו בצד-שרת מסיימת, הביצוע חוזר אל הלקוח, אל משפט ההתראה (alert) שמציג את ההודעה משירות האינטרנט.

הניפוי אמנם שימושי מאוד במשך פיתוח האתר, אך הוא חסר יכולות לחקור את התנהגות האתר כשיופעל. למרבה המזל יש ל-ASP.NET פתרון גם לבעיה זו – **מעקב** (tracing).

מעקב (Tracing) אחר דפי האינטרנט של ASP.NET

איתור בזמן ביצוע של ערכי משתנים, אובייקטים, נתיב הביצוע של הקוד וכך הלאה, תהיה בעייתית ללא מעקב ראוי. ככל הנראה היית מוסיף פקד Label לדף, וכותב אליו מידע כך:

VB.NET

```
Dim value2 As Double = Convert.ToDouble(txtValue2.Text)
lblDebug.Text &= "The value of value2 = " & value2.ToString() & "<br />"
```

C#

```
double value2 = Convert.ToDouble(txtValue2.Text);
lblDebug.Text += "The value of value2 = " + value2.ToString() + "<br />";
```

קוד זה בהחלט יפעל, אך הוא מסורבל למדי. ראשית, עליך לכתוב קוד רב כדי שהדבר יפעל. שנית, תסיים עם פקד Label מסורבל שמוצג בדף, ותצטרך לזכור להסירו כשתסיים את ביצוע הניפוי או המעקב. לבסוף, כשתהיה מוכן, תצטרך להסיר את כל הקוד שהגדיר את התווית lblDebug. תוכל לנקוט בדרך הקלה על ידי הגדרת המאפיין Visible של הפקד Label ל-False, אבל עדיין תסבול מהרעת הביצועים שנובעת מהקצאת הטקסט לפקד Label.

המעקב (tracing) ב-ASP.NET פותר את כל הבעיות האלו. הוא מאפשר לדפים, לפקדים ולקוד לכתוב מידע במקום מרכזי שנקרא Trace, שניתן להציגו אחר כך בדפדפן. המעקב בנוי בתוך סביבת העבודה ASP.NET, ופירוש הדבר שביכולתך להשתמש בו מבלי שתצטרך לכתוב קוד באופן ידני. בנוסף, תוכל להוסיף מידע משלך לצורך המעקב. בסעיף הבא תלמד כיצד להשתמש ביכולות המעקב המובנות, שנותנות שפע של מידע אודות הדף. בתרגיל מאוחר יותר תלמד כיצד להוסיף מידע משלך עבור המעקב.

שימוש ביכולות המעקב הרגילות

ללא טרחה מרובה תקבל מידע רב על הדרך שבה מתבצעים הדפים. כל שעליך לעשות הוא להפעיל מעקב אחר הדפים שלך. תוכל לעשות זאת ברמת הדף או ברמת היישום. כשהמעקב מופעל ברמת הדף, תוכל לבחור דפים מסוימים, אחד או יותר, שברצונך לעקוב אחריהם. מעקב ברמת היישום שימושי כשברצונך להתבונן בדפים רבים בו-זמנית. הדבר יכול לסייע לך, לדוגמה, לאתר דפים איטיים באתר.

מעקב אחר דפים יחידים

כדי להפעיל מעקב בדף, עליך לקבוע שערך המאפיין Trace בהנחיה Page יהיה True:

```
<%@ Page ..... Trace="True" %>
```

כשתפעיל דף שלגביו הופעל מעקב, תקבל רשימה ארוכה של פרטים בתחתית הדף. תרשים 17-24 מציג את ASP.NET Trace עבור דף ההדגמה של המחשבון ששימש אותך בפרק זה.

יומן המעקב מספק פרטים רבים על הדף הנוכחי. בראש פלט הדף יש תמצית של פרטי הבקשה, כולל התאריך והשעה הנוכחיים, השיטה ששימשה לאחזור הדף (GET או POST), וקוד המצב (קוד מצב 200 בתרשים 17-24 מציין הצלחה).

מתחת לתמצית מופיע הסעיף Trace Information. המחלקה Page של ASP.NET כותבת ל-trace כשהיא מופעלת. הדבר דומה לדרך ההדגמה שכתבת בפרק 14 שכתב לפקד Label מאירועים רבים שהופעלו במהלך מחזור החיים של הדף.

כברירת מחדל, הנתונים ממוינים לפי הזמן. כלומר, האירועים מופיעים לפי סדר התרחשותם. תוכל למיין אותם בהתאם לקטגוריה (על קטגוריות תלמד יותר בסעיף שיעסוק בהוספת מידע משלך ל-trace) על ידי שינוי TraceMode מ-SortByTime ל-SortByCategory.

```
<%@ Page ..... Trace="true" TraceMode="SortByCategory" %>
```

בהמשך הדף תראה את עץ הפקדים שמציג מבט היררכי על הפקדים בדף ומציין את גודלם.

מתחת לעץ הפקדים תראה את הפרטים של כמה אוספים חשובים, כולל Cookies, Query String, Form, Headers ו-Server Variables. בנוסף, תראה מידע שהיית עשוי לאחסן ב-Session state או ב-Application state. היכולת לראות אוספים אלה מהווה סיוע ממשי להבנת בעיה. לדוגמה, אם יש לך דף שאמור לקרוא מעוגייה, אבל נכשל ברגע שהוא נטען, תוכל להתבונן באוסף Cookies ולראות אם הדף קיבל את הנתונים שציפית שיקבל. אוספים אלה הם כלים יקרי-ערך להבנת ביצוע הדף ויכולים לסייע במציאה ותיקון של הבאגים בקוד.

מעקב ברמת הדף, פירושו שעליך להפעיל את המעקב בכל דף שברצונך לעקוב אחריו. פירושו גם, שעליך לבטל את הפעלתו בכל דף כשתסיים את המעקב. מכיוון שהמעקב הפרטני על כל דף בנפרד עלול להיות מסורבל באתר גדול, ASP.NET מאפשרת גם לעקוב אחר האתר כולו כיחידה שלמה.

Calculator Demo - Windows Internet Explorer			
http://localhost:43394/Demos/CalculatorDemo.aspx			
Calculator Demo			
Request Details			
Session Id:	obdoa1mhd0ss53b5yrhkoqf55	Request Type:	GET
Time of Request:	12/2/2007 1:55:35 PM	Status Code:	200
Request Encoding:	Unicode (UTF-8)	Response Encoding:	Unicode (UTF-8)
Trace Information			
Category	Message	From First(s)	From Last(s)
aspx.page	Begin PreInit	0.000216463519804891	0.000218
aspx.page	End PreInit	0.000477393711491265	0.000260
aspx.page	Begin Init	0.00114930808245182	0.000671
aspx.page	End Init	0.00133145413732751	0.000182
aspx.page	Begin InitComplete	0.00151136527128448	0.000180
aspx.page	End InitComplete	0.00168960021455241	0.000178
aspx.page	Begin PreLoad	0.00345909895194906	0.001769
aspx.page	End PreLoad	0.00364850839981059	0.000189
aspx.page	Begin Load	0.00427121324078898	0.000623
aspx.page	End Load	0.00445615294681307	0.000185
aspx.page	Begin LoadComplete	0.00463801963657392	0.000182
aspx.page	End LoadComplete	0.00482659108909093	0.000189
aspx.page	Begin PreRender	0.00504449587866614	0.000218
aspx.page	End PreRender	0.00522245145681923	0.000178
aspx.page	Begin PreRenderComplete	0.00540068640008716	0.000178
aspx.page	End PreRenderComplete	0.00682908023226416	0.001428
aspx.page	Begin SaveState	0.007588953344629	0.000760
aspx.page	End SaveState	0.00777193749484921	0.000183
aspx.page	Begin SaveStateComplete	0.00795101053346165	0.000179
aspx.page	End SaveStateComplete	0.00813231849299282	0.000181
aspx.page	Begin Render	0.00922938529896956	0.001097
aspx.page	End Render		
Control Tree			
Control UniqueID	Type	Render Size Bytes (including children)	ViewState Size Bytes (excluding children) ControlState Size Bytes (excluding children)
Page	ASP.demos.calculatordemo.aspx	1737	0
ctl03	System.Web.UI.LiteralControl	170	0
ctl03	System.Web.UI.LiteralControl	274	0

תרשים 17-24

מעקב אחר האתר כולו

באפשרותך להפעיל מעקב אחר האתר כולו על ידי שינוי הגדרות המעקב בקובץ web.config. תבצע זאת על ידי יצירת אלמנט `<trace />` תחת `<system.web>`. הטבלה הבאה מפרטת את המאפיינים החשובים ביותר של האלמנט `<trace />`.

מאפיין	תיאור
enabled	קביעה אם מופעל מעקב אחר היישום כולו או לא. כברירת מחדל המעקב אינו מופעל, ולפיכך יש צורך להגדיר מאפיין זה באופן מפורש ל-True.
traceMode	קביעת סדר המיון של הפריטים בפלט של המעקב. הוא פועל באופן זהה למאפיין TraceMode של ההנחיה Page.
requestLimit	קביעה של מספר בקשות המעקב שיכולות להיות פעילות ב-ASP.NET.
pageOutput	ציון אם מידע המעקב יוצג בדף. כשהמאפיין מוגדר כ-False (ברירת המחדל) תוכל לגשת למידע המעקב רק על ידי המטפל Trace.axd שנדון בו מאוחר יותר.
localOnly	קביעה אם המטפל Trace.axd נגיש מהמחשב המקומי בלבד. מנקודת המבט של אבטחה עדיף להשאיר את התכונה מוגדרת כ-True, כלומר המעקב אינו זמין למשתמשים מבחוץ.

מאפיין	תיאור
mostRecent	קביעה אם רשומות מעקב ישנות תסולקנה כשמספר בקשות המעקב יגיע ל-requestLimit. כשהמאפיין מוגדר כ-False, המעקב יבוטל באופן אוטומטי כשמגיעים ל-requestLimit.

לאחר הפעלת המעקב, יש שתי דרכים לקרוא את המידע שלו. אם הגדרת את pageOutput ל-True, מידע המעקב מצורף לכל דף, בדומה למה שראית בעת מעקב ברמת הדף.

עם זאת, באפשרותך לבטל את pageOutput ואז לבקש את מידע המעקב על ידי שימוש בקובץ מיוחד בשם Trace.axd. זהו קובץ וירטואלי, ולכן לא תוכל למצוא אותו באופן פיזי באתר. עם זאת, רכיב ASP.NET runtime יודע שעליו לספק מידע מעקב כאשר אתה מבקש קובץ זה. בתרגיל הבא תראה כיצד זה פועל.

נסה זאת הפעלת מעקב בכל חלקי האתר

בתרגיל זה תראה כיצד להפעיל מעקב ברחבי האתר. תחילה תבצע כמה שינויים בקובץ web.config. אחר כך תדפדף באתר ותגרום ליומן המעקב להתמלא בקריאת הדפים שנעשתה על ידך. לבסוף תבקש את הדף המיוחד Trace.axd כדי לראות את המידע שהצטבר ביומן המעקב.

1. פתח את web.config ואתר את תגית הפתיחה <system.web>. מייד אחריה הוסף את מידע התצורה הבא, כדי לאפשר מעקב:

```
<trace
  mostRecent="true"
  enabled="true"
  requestLimit="100"
  pageOutput="false"
  localOnly="true"
/>
```

כך תוכל לנהל את המעקב, אבל אין זה מוסיף את פלט המעקב לדף. תצטרך לבקש את הדף המיוחד Trace.axd כדי לראות את מידע המעקב. בנוסף, הפכת את המערכת שלך לבטוחה מעט יותר, כי אפשרת בקשות למידע המעקב מהמחשב המקומי בלבד.

2. שמור וסגור את web.config. בסייר הפתרון, לחץ לחיצה ימנית על Default.aspx ובחר View in Browser.

3. לחץ ברחבי האתר, פתח דפים, שנה ערכות נושא, מלא טפסי קשר וכן הלאה.

4. לאחר בקשת חמישה דפים לפחות, שנה את הכתובת בסרגל הכתובת בדפדפן, כדי לקבל את הדף המיוחד Trace.axd:

<http://localhost:49394/Trace.axd>

מספר היציאה (port) שלך עשוי להיות שונה, אבל חשוב שתבקש את הדף Trace.axd מ-localhost. שים לב שכתובת זו מניחה שבתצורת האתר קבעת את הסימן "/" (לוכסן)

כ- Virtual Path, כפי שלמדת בפרק 7. אם לא עשית זאת עדיין, בחר באתר מסייר הפתרון, פתח את טבלת המאפיינים שלו, והגדר את Virtual Path לסימן לוכסן. בקש את הקובץ Virtual Path שוב. אתה אמור לקבל דף דומה לתרשים 17-25.

No.	Time of Request	File	Status Code	Verb	Remaining: 90
1	12/2/2007 1:59:48 PM	Default.aspx	200	GET	View Details
2	12/2/2007 1:59:51 PM	PhotoAlbums/default.aspx	200	GET	View Details
3	12/2/2007 1:59:55 PM	PhotoAlbums/default.aspx	200	POST	View Details
4	12/2/2007 1:59:58 PM	About/Contact.aspx	200	GET	View Details
5	12/2/2007 1:59:59 PM	App_Themes/Monochrome/Images/PleaseWait.gif	200	GET	View Details
6	12/2/2007 2:00:20 PM	PhotoAlbums/default.aspx	200	GET	View Details
7	12/2/2007 2:00:22 PM	PhotoAlbums/default.aspx	200	POST	View Details
8	12/2/2007 2:00:25 PM	Reviews/AllByGenre.aspx	200	GET	View Details
9	12/2/2007 2:00:29 PM	Reviews/All.aspx	200	GET	View Details
10	12/2/2007 2:00:31 PM	Reviews/ViewDetails.aspx	200	GET	View Details

Microsoft .NET Framework Version: 2.0.50727.1433; ASP.NET Version: 2.0.50727.1433

תרשים 17-25

5. רשומות המעקב ממוינות על פי הזמן שתועדו, מהישנה ביותר אל החדשה ביותר. לחץ על הקישור View Details (ראה תרשים 17-25) עבור דף ASPX, למשל Contact.aspx. תקבל דף דומה לזה שמוצג בתרשים 17-24.

כיצד זה פועל

היכולת לראות מידע מעקב עבור דפים שנדרשו היא רבת-ערך ביותר. המידע יכול לעזור לך להבין את זרימת המידע מדף האינטרנט (web) ואליו. לדוגמה, מידע המעקב עבור הדף Contact.aspx מציג גם את המידע שהכניסו משתמשים בפקדי תיבות הטקסט שבדף. כדי לראות כיצד מידע זה נראה, לחץ על הקישור View Details עבור פריט שברצונך להתמקד בו.

למרות שמידע המעקב ש-ASP.NET אוספת עבורך באופן אוטומטי הוא שימושי ביותר, אינך מוגבל למידע זה בלבד, ובאפשרותך להוסיף מידע משלך ליומן המעקב.

הוספת מידע משלך ל-Trace

בנוסף למידע הרגיל ש-ASP.NET runtime מוסיפה, באפשרותך גם להוסיף מידע משלך למעקב. הדבר שימושי אם, לדוגמה, ברצונך לראות את הערך של משתנה, או שברצונך לדעת אם אירוע מסוים התרחש אם לאו, ואם התרחש, מתי בדיוק.

תוכל להוסיף מידע מעקב על ידי שימוש במחלקה Trace. מחלקה זו חושפת שתי שיטות שימושיות, Write ו-Warn שפעולתן דומה למדי. הן מוסיפות ליומן המעקב את המידע שתוכל לשים אותו בקטגוריה שיצרת בעצמך. ההבדל היחיד בין Write לבין Warn הוא בכך שהודעות שנכתבות על ידי Warn מופיעות באדום. תוכל להשתמש בשיטה Warn במצבים בלתי-צפויים, מכיוון שהודעותיה מושכות יותר תשומת-לב.

בתרגיל הבא תראה עד כמה פשוט להוסיף מידע משלך ליומן המעקב על ידי שימוש בשיטות Write ו-Warn.

נסה זאת הוספת נתוני מעקב לדפים שלך

בתרגיל זה תוסיף מידע מותאם אישית ליומן המעקב של ASP.NET. תשתמש בשיטה Write כדי לכתוב מידע מעקב בזמן ביצוע תקין של דף, ובשיטה Warn עבור תרחישים בלתי-צפויים.

1. פתח את CalculatorDemo.aspx, עבור לקוד ברקע שלו, ואתר את המטפל Click של הלחצן Calculate.

2. מייד לפני המשפט Select Case (VB.NET), או המשפט switch (C#), הוסף את הקוד הבא:

VB.NET

```
Trace.Write(String.Format("Performing the calculation with the {0} operator", _  
    lstOperator.SelectedValue))  
Select Case lstOperator.SelectedValue
```

C#

```
Trace.Write(string.Format("Performing the calculation with the {0} operator",  
    lstOperator.SelectedValue));  
switch (lstOperator.SelectedValue)
```

3. בתחתית מטפל האירוע, שנה את משפט Else לבדיקה שמבטיחה ששני פקדי TextBox מכילים ערכים.

VB.NET

```
Else  
    lblResult.Text = String.Empty  
    Trace.Warn("Custom Category", _  
        "TextBox controls are empty; time to add Validation controls?")  
End If
```

C#

```
else  
{  
    lblResult.Text = string.Empty;  
    Trace.Warn("Custom Category",  
        "TextBox controls are empty; time to add Validation controls?");  
}
```

4. הפעל במפורש מעקב אחר דף זה. תוכל לעשות זאת על ידי הגדרת המאפיין Trace של ההנחיה Page ב- Markup View.

```
<%@ Page ... Trace="true" %>
```

5. שמור את כל השינויים ובקש את דף המחשבון בדפדפן על ידי הקשת Ctrl+F5. הכנס שני מספרים ולחץ על הלחצן Calculate. שים לב שהמידע המותאם אישית שלך מתווסף ליומן המעקב בין הכניסה Begin RaisePostBackEvent לבין הכניסה End RaisePostBackEvent. שים לב לשינוי בעמודה Category באותה שורה.
6. בדפדפן, נקה את הטקסט משני פקדי TextBox, ולחץ שוב על הלחצן Calculate. מידע המעקב אמור להיות כעת קל יותר לאיתור בגלל צבעו השונה ושם הקטגוריה שלו, כפי שמוצג בתרשים 17-26.
7. חזור פעם נוספת ל- Visual Web Developer ובטל את המעקב. עליך לשנות זאת ברמת הדף (הסר את המאפיין Trace מההנחיה Page של הדף Calculator), וברמת האתר ב- web.config. שמור את כל השינויים ובקש שוב את הדף Calculator. שים לב שהדף מתפקד עדיין כהלכה, אבל אינו פולט עוד את מידע המעקב.

Category	Message	From First(s)	From Last(s)
aspx.page	Begin PreInit	0.000147225415520688	0.000147
aspx.page	End PreInit	0.000355073060961659	0.000208
aspx.page	Begin Init	0.000645054050165594	0.000290
aspx.page	End Init	0.00118813983341458	0.000543
aspx.page	Begin InitComplete	0.00137531446035739	0.000187
aspx.page	End InitComplete	0.00156640019890796	0.000191
aspx.page	Begin LoadState	0.0019094605599315	0.000343
aspx.page	End LoadState	0.00209663518687431	0.000187
aspx.page	Begin ProcessPostData	0.00237544157148464	0.000279
aspx.page	End ProcessPostData	0.00256177810308293	0.000186
aspx.page	Begin PreLoad	0.00276934638340906	0.000208
aspx.page	End PreLoad	0.00296937180563452	0.000200
aspx.page	Begin Load	0.00326521946225009	0.000296
aspx.page	End Load	0.00345043853338902	0.000185
aspx.page	Begin ProcessPostData Second Try	0.00363565760452795	0.000185
aspx.page	End ProcessPostData Second Try	0.0038203179454372	0.000185
aspx.page	Begin Raise ChangedEvents	0.0041603052901975	0.000340
aspx.page	End Raise ChangedEvents	0.00440977833774963	0.000249
aspx.page	Begin Raise PostBackEvent	0.00462572757152096	0.000216
Custom Category	TextBox controls are empty; time to add Validation controls?	0.00491822284675846	0.000292
aspx.page	End Raise PostBackEvent	0.00597394361573887	0.001056
aspx.page	Begin LoadComplete		

תרשים 17-26

כיצד זה פועל

השיטות Write ו-Warn של המחלקה Trace מאפשרות לכתוב מידע נוסף ליומן המעקב. רכיב runtime של ASP.NET עוקב אחר המידע ומציג אותו יחד עם שאר מידע המעקב. כשמדובר במעקב ברמת הדף, המידע מוצג בדפדפן בתחתית הדף, או על ידי הדף המיוחד Trace.axd שראית קודם.

לכל אחת מהשיטות Write ו-Warn יש שלוש העמסות-יתר. הראשונה (מוצגת רק עבור Write בדוגמה הקודמת) מקבלת מחרוזת יחידה שמוצגת בעמודה Message. העמסת היתר השנייה מקבלת גם שם של קטגוריה, כפי שמודגם עם השיטה Warn. העמסת היתר האחרונה, שאינה מוצגת בתרגיל נסה זאת הקודם, מקבלת גם אובייקט Exception שההודעה שלו תתווסף לפלט המעקב. הדבר שימושי עבור מעקב אחר מידע חריגים בבלוק Catch.

מעקב וביצועים

למרות שנראה שהשארית משפטי Warn ו-Write בקוד עלולה לפגוע בביצועים, אין זה כך. מכיוון שביכולתך לבטל את המעקב על ידי הגדרת המאפיין enabled של האלמנט trace למצב false (בקובץ web.config), הרי שלמעשה אתה מעלים את תקורת הביצועים של משימת המעקב.

הערת אבטחה

מעקב יכול להיות שימושי מאוד, אבל השארית מידע מעקב בסביבת ה-production עלולה להוביל לחשיפת מידע. לכן, עליך תמיד לבטל את המעקב על ידי הגדרת המאפיין enabled שלו בקובץ web.config ל-False, או לפחות הגדר את localOnly ל-True. בפרק 18 תלמד תכסיס שיאפשר לך להפעיל שינוי זה בכל האתרים שבשרת הייצור (production) שלך, והדבר יקל על חסימת הגישה לפונקציונליות המעקב.

עצות מעשיות בנושא ניפוי

הרשימה הבאה מספקת כמה עצות מעשיות שתעזורנה לך לנפות את היישום.

- ⊙ לעולם אל תשאיר debug="true" בקובץ web.config של אתר שנמצא בסביבת production. הגדר אותו תמיד ל-true כדי לשפר בביצועים וכדי להבטיח פעולה יציבה של האתר. בפרק 18 תראה פתרון טוב עוד יותר שיבטיח שהגדרה זו לעולם לא תהיה מוגדרת ל-True בשרת ה-production.
- ⊙ נסה לא להתעלם מחריגים בבלוק Catch. אתה עשוי להתפתות לעטוף את הקוד בבלוק Try/Catch ולהשאיר את הבלוק Catch ריק. הדבר אמנם ימנע הצגת חריגים בממשק המשתמש, אבל כעת אינך מודע לכך שקרתה בעיה, לא תוכל לכתוב קוד שימנע את הבעיה מלכתחילה. הכלל כאן הוא זה: תפוס שגיאות שביכולתך לטפל בהן בהצלחה, לדוגמה, על ידי הצגת הודעה למשתמש. אם אינך יכול לטפל בבעיה בקוד, הנח לה לעלות מעלה ורשום אותה במטפל האירוע Application_Error כדי שתדע שהבעיה קרתה.
- ⊙ אם מתעורר הצורך לגרום להופעת הבעיה מחדש בבלוק Catch, אל תשתמש ב-Throw ex (throw ex ב-C#) אלא ב-Throw (throw ב-C#) בלבד. השימוש ב-Throw ex מקשה על מעקב אחר המסלול שהקוד עובר לפני שקורית התקלה. הנה הקוד:

VB.NET

Try

...

Catch ex As Exception

```
Throw ex ' Bad example; you lose track of the source of the exception
```

```
Throw ' Good example; forwards the exception and maintains the call stack
```

End Try

C#

try

{

...

}

catch (Exception ex)

{

```
throw ex; // Bad example; you lose track of the source of the exception
```

```
throw; // Good example; forwards the exception and maintains the call stack
```

}

⊙ השתדל ככל האפשר להימנע מטיפול בחריגים. כפי שראית בפרק, עדיף (ומהיר יותר) להימנע מחריגים מלכתחילה.

לדוגמה, ניתן בקלות להימנע מהחריג DivideByZeroException על ידי בדיקה עבור הערך אפס לפני ביצוע פעולת החילוק.

סיכום

גם אם תיזהר מאוד בעת התכנות, סביר שהאתר יכיל כמה באגים או שיופיעו חריגים בזמן ביצוע. כדי למזער את מספר השגיאות ולבנות אתר שיפעל בצורה חלקה ככל האפשר, יש כמה דברים שביכולתך לעשות.

ראשית, תוכל להשתמש בטכניקות לטיפול בבעיות, שבהן תכתוב קוד שמסוגל לתפוס את החריגים שצפית מראש, ולטפל בהם כראוי. לעתים מספיק עדכון תווית כדי שתודיע למשתמש שמהו השתבש. במקרים אחרים, למשל כששרת בסיס נתונים אינו פועל, לא תוכל לעשות הרבה. במקרים כאלה תוכל להשתמש בקוד ששולח הודעות דואר אלקטרוני, ומאפשר לך לגלות ולנתח בעיות גם כשאינך מחובר לשרת.

כדי לעזור לך לכתוב קוד נקי מבאגים ככל האפשר, התוכנה VWD מציעה לך מבחר עשיר של כלי ניפוי. היכולת לפרוץ לקוד ולנתח ולשנות את סביבת הביצוע, תגרום לחיסכון ניכר בזמן. נוספת לכך חווית הניפוי החלק שמאפשרת לנפות את היישום החל מהקוד שבצד-לקוח ועד לשרת. אפשרויות הניפוי עושות את VWD למכשיר נהדר בהרפתקאות הניפוי שלך.

גם אם ניפית את היישום באופן יסודי, עדיין יש סיכוי שבאתר תתגלית בעיות ב-production, בין אם הן קשורות בביצועים, שגיאות לוגיות, או סיבות בלתי-צפויות אחרות. במקרים אלה

תוכל להשתמש באמצעי המעקב של ASP.NET שמאפשרים לך לעקוב אחר מידע אודות הדפים המתבצעים. ניתוח מידע מעקב זה יכול לקדם אותך מאוד בתיקון הבעיות הבסיסיות. עכשיו, כשאתר האינטרנט שלך הושלם, ואנו מקווים שללא באגים, השלב הבא הוא להפכו למקוון. בפרק הבא תלמד כיצד להעלות לרשת אתר אינטרנט מבוסס ASP.NET.

תרגילים

1. מה ההבדל בין ניפוי לבין מעקב?
2. הנח שיש לך קוד שאתה חושש שעלולה להיות בו בעיה. לדוגמה, אתה מנסה לשלוח הודעת דואר אלקטרוני לשרת דואר, או כותב קובץ לדיסק כשאינך בטוח שיש לך את ההרשאות המתאימות לביצוע הפעולה. באיזה סוג של אסטרטגיית טיפול בחריגים תשתמש כדי למנוע את הצגת הבעיה בדפדפן? לאיזה קוד תזדקק?
3. אתה לוקח על עצמך אחריות לאתר אינטרנט שנבנה על ידי מתכנת שלא שמע מעולם על טיפול בחריגים. הלקוח שלך מתלונן על האיכות הירודה של האתר, ועל המספר הגדול של "מסכי המוות הצהוב" שהמשתמש רואה. בנוסף לניתוח הקוד של כלל היישום, מה יהיה פתרון מהיר לקבלת מידע על השגיאות והמקומות שבהן התרחשו? איך תוכל להגן על משתמשי האתר מפני הפרטים "המלוכלכים" של הודעות השגיאה?

פרק 18

פריסת אתר אינטרנט

(Deploying Your Web Site)

הערה: בזמן כתיבת שורות אלו מיקרוסופט שחררה את **SQL Server 2008 Express Edition**, אל דאגה! ממשיך המשתמש והפונקציונליות שבה נשתמש זהים לחלוטין.

איחולינו! העובדה שאתה קורא פרק זה פירושה, ככל הנראה, שיש בידך כעת אתר אינטרנט מבוסס ASP.NET על כל תכונותיו, שמקור הנתונים שלו נמצא בבסיס-הנתונים, ומוכן להתפרסם בעולם הרחב. קהל היעד יתחיל במהרה להשתמש באתר וגם... לשפוט אותו.

כדי שהאתר יהיה נגיש למשתמשים באינטרנט, עליך להעבירו לשרת ייצור (production server) שקשור לאינטרנט. איזה סוג שרת זה יהיה והיכן הוא יימצא, הן שתי שאלות שהתשובה עליהן תלויה בדרישותיך ובתקציב שלך. באפשרותך לארח את האתר בשרת ביתי בעליית הגג עם חיבור פרטי לאינטרנט, או שתארח אותו אצל גורם חיצוני (לעתים קרובות מסחרי) בעל חיבור ישיר לשדרת האינטרנט.

כך או כך, עליך להשקיע עבודה מסוימת כדי שתוכל להעביר את האתר ממקום הפיתוח שלו ב- C:\BegASPNET\Site למקום שבו הוא יהיה נגיש לרשת האינטרנט כולה.

פרק זה עוסק בכמה נושאים שקשורים לפריסה (**deploying**) מוצלחת של יישום אינטרנט. תלמד על התהליך החל מהכנת היישום בסביבת הפיתוח ועד להפעלתו הממשית ובדיקתו בשרת ה-production. להלן כמה מהנושאים שפרק זה יעסוק בהם בהרחבה:

- ⊙ כיצד להקל על תהליך הפריסה על ידי שינויים פשוטים בקוד ובתצורה.
- ⊙ כיצד להכין את האתר לפריסה על ידי יצירת עותק שלו בשימוש כלי ההעתקה המובנים של Visual Web Developer.
- ⊙ איך להתקין את שרת האינטרנט ואת האתר במחשב היעד.
- ⊙ איך להימנע משגיאות נפוצות שעלולות להתרחש בעת פריסת אתר.
- ⊙ איך להעתיק נתונים שמאוחסנים בבסיסי הנתונים מסוג SQL Server 2005 אל שרת היעד.

הפרק מסתיים ברשימת דברים שעליך לתת דעתך עליהם בעת פריסה של יישום. תוכל להשתמש ברשימת תיוג זו כדי להבטיח שהגדרת אתר תצורת אתר ה-production בדרך הבטוחה והטובה ביותר.

הכנת האתר לפריסה (deployment)

כשאתה עובד על המהדורה הראשונה של האתר בסביבת פיתוח, ניהול האתר וקוד המקור שלו הינו ישיר למדי. יש רק גרסת מקור אחת של האתר, והדבר מקל על תחזוקתה. עם זאת, ברגע שהאתר עובר לייצור, מייד נוצרות שתי גרסאות שלו: האחת פועלת בסביבת הייצור והשנייה היא זו שמשמשת להמשך הפיתוח. הדבר מקשה על שמירת הסנכרון בין הגרסאות. לדוגמה, קרוב לוודאי שבסביבת הייצור אתה משתמש בבסיס נתונים שונה ובמחזרות חיבור שונה. סביר גם שתשתמש בכתובות דואר אלקטרוני שונות עבור הדואר האלקטרוני שנשלח על ידי האתר. לבסוף, ייתכן שבסביבת הפיתוח תרצה לבטל את השליחה בדואר אלקטרוני של הודעות השגיאה מקבצי Global.asax. אם תעשה את כל השינויים האלה בקוד כשהאתר יעלה לאוויר, יש סבירות גבוהה שבמהלך העדכון הבא תדרוס כמה מההגדרות, והדבר עלול לגרום לתוצאות לא-רצויות.

בסעיף הבא תלמד כיצד להפוך את ניהול הגרסאות השונות של אותו אתר לקל יותר. תלמד גם כיצד להעביר כמה מההגדרות המובנות, כמו כתובות דואר אלקטרוני, אל קובץ web.config. קוד היישום קורא ערכים אלה בזמן ביצוע. ההבדל היחיד בין סביבות הפיתוח (development) והייצור (production) הוא קובץ תצורה יחיד, והדבר מקל על קיום הגדרות שונות בשתי הסביבות.

הימנעות מהגדרות מובנות בתוכנה

עד כה השתמשו הדפים ופקדי המשתמש שבנית בכמה הגדרות מובנות עבור דברים כמו כתובות דואר אלקטרוני. לדוגמה, ContactForm.aspx, פקד המשתמש ששולח דואר אלקטרוני, משתמש בקוד הבא כדי לקבוע את פרטי הנמען והשולח:

VB.NET

```
myMessage.From = New MailAddress("you@yourprovider.com", "Sender Name Here")
myMessage.To.Add(New MailAddress("you@yourprovider.com", "Receiver Name Here"))
```

C#

```
myMessage.From = New MailAddress("you@yourprovider.com", "Sender Name Here");
myMessage.To.Add(New MailAddress("you@yourprovider.com", "Receiver Name Here"));
```

כתיבת הגדרות בצורה מובנית כזו מקשה על שינוי הערכים בסביבות שונות. בכל פעם שתצצה לשחרר את האתר לייצור, תצטרך לוודא שלא תדרוס בטעות את הגדרות ששינית לצורך סביבת הייצור.

למרבה המזל, ASP.NET מכילה פתרון נהדר למניעת סוג זה של בעיות: הקובץ web.config, **תחביר הביטוי** (expression syntax), והמחלקה ConfigurationManager שבה משתמשים לקריאה מהקובץ web.config.

הקובץ web.config

עד כה השתמשת בקובץ web.config מספר פעמים לאחסון מידע אודות מחרוזות חיבור, פרופילים וחברות, ועוד. עדיין לא הכרת את האלמנט <appSettings> שמאפשר לאחסון מידע בצורה של מפתח + ערך על ידי שימוש באלמנטים <add />. האלמנט <appSettings> מאפשר לאחסון מידע פשוט, כמו כתובת דואר אלקטרוני, ולאחזר ערך זה לפי המפתח שלו. לדוגמה, כדי לאחסון כתובת דואר אלקטרוני, תוכל להוסיף את המידע הבא לקובץ web.config:

```
<appSettings>
  <add key="FromAddress" value="webmaster@planetwrox.com" />
</appSettings>
```

האלמנט <appSettings /> מוצב בקובץ web.config מחוץ לאלמנט <system.web>, אך עדיין בתוך אלמנט האב <configuration />.

ברור שזקוקים לדרך גישה לנתוני <appSettings /> בזמן ביצוע. יש כמה אפשרויות לכך ומהן נדון בשתיים: שימוש בתחביר הביטוי ושימוש במחלקה ConfigurationManager.

תחביר הביטוי (expression)

תחביר הביטוי מאפשר לקשור מאפייני פקד אל משאבים, כמו אלה שנמצאים באלמנט <appSettings> שבקובץ web.config, במחרוזות חיבור, ובקבצי משאבים מקומיים. כדי להציג נתונים מהאלמנט <appSettings> השתמש בתחביר הבא, שבו AppSettingKeyName מתייחס למפתח שמגדירים ב-web.config.

```
<%$ AppSettings:AppSettingKeyName %>
```

לדוגמה, כדי להציג בדפים הודעה על זכויות יוצרים באמצעות הפקד Literal, עליך להוסיף את ההגדרה הבאה ל-web.config:

```
<add key="Copyright" value="Copyright by Wrox" />
```

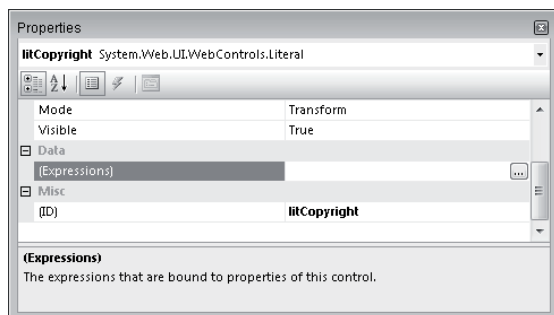
תוכל להציג טקסט זה בפקד Literal בדרך זו:

```
<asp:Literal ID="litCopyright" runat="server" Text="<%$ AppSettings: Copyright %>" />
```

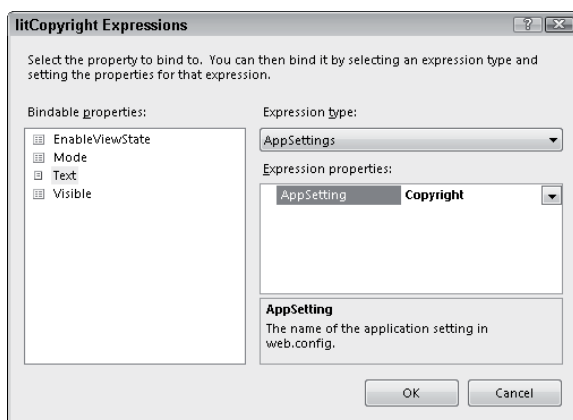
כדי להקל עוד יותר על הגדרת מאפיינים כגון Text, כפי שנעשה בדוגמה לעיל, VWD כוללת Expression Editor (עורך ביטויים). כדי לגשת לתיבת דו-שיח זו, בחר פקד ב- Design View או ב- Markup View, פתח את טבלת המאפיינים שלו, ולחץ על הסימן (...) שמימין לפריט (Expressions), כפי שמוצג בתרשים 18-1. אם הפריט (Expressions) אינו מופיע ב- Markup View (הדבר קורה לעתים), עבור תחילה ל- Split View או ל- Design View.

כעת נפתחת תיבת הדו-שיח Expressions, שמאפשרת לקשור מאפייני פקד לביטויים. כדי לקשור את המאפיין Text של הפקד Literal להגדרת יישום בצע את הפעולות הבאות: לחץ על Text שבצד שמאל של תיבת הדו-שיח, מהרשימה הנפתחת Expression Type שבצד ימין בחר AppSettings, ולבסוף בחר AppSettings נכון מהרשימה הנפתחת שבקטע Expression Properties. תרשים 18-2 מציג את תיבת הדו-שיח Expressions השלמה עבור פקד Literal שמשמש להצגת

טקסט זכויות היוצרים. כשתלחץ OK, תשנה VWD את המאפיין Text של Literal באופן שיכיל הפניה להגדרת היישום הנכונה.



תרשים 18-1



תרשים 18-2

קבלת ערכים מ-web.config על ידי שימוש בתחביר הביטוי היא שימושית למדי, אך עשויה שלא לענות על כל צרכיך. לכן, טוב לדעת שאפשר לאחזר את הערכים גם על ידי תכנות. כדי לבצע זאת, השתמש במחלקה ConfigurationManager.

המחלקה ConfigurationManager

המחלקה ConfigurationManager ממרחב השמות System.Configuration מספקת גישה לנתונים שמאוחסנים בקבצי תצורה. יש לה תמיכה מיוחדת באלמנטים appSettings ו-connectionStrings של הקובץ web.config, שמאפשרת לאחזר נתונים מאלמנטים אלה על ידי שורת קוד אחת. קטע הקוד הבא מראה כיצד לאחזר את הקטע FromAddress מהאלמנט appSettings.

VB.NET

```
Dim fromAddress As String = ConfigurationManager.AppSettings.Get("FromAddress")
```

C#

```
string fromAddress = ConfigurationManager.AppSettings.Get("FromAddress");
```

השיטה Get תמיד מחזירה נתונים כמחרוזת. לכן, כשמצפים למשהו שונה ממחרוזת, צריך להמיר את תשובת Get לסוג הנתונים המתאים. לדוגמה, אם אחסנת ערך בוליאני ב-web.config בדרך זו:

```
<add key="SendMailOnError" value="true" />
```

תצטרך להשתמש בקוד הבא, כדי לאחזר ולהמיר את הערך:

VB.NET

```
Dim sendMail As Boolean = _  
    Convert.ToBoolean(ConfigurationManager.AppSettings.Get("SendMailOnError"))
```

C#

```
bool sendMail = Convert.ToBoolean(ConfigurationManager.AppSettings.Get("SendMailOnError"));
```

אתה יכול לגשת אל המחלקה ConfigurationManager באופן ישיר מהקוד שלך (בהנחה שייבאת את מרחב השמות System.Configuration). למרות זאת, אני מעדיף ליצור מאפיינים סטטיים, לקריאה-בלבד במחלקת תצורה מותאמת אישית, שניגשת אל web.config כדי לקבל את הערכים. בתרגיל הבא תלמד כיצד לעשות זאת.

נסה זאת העברת הגדרות היישום לקובץ התצורה

בתרגיל זה תיצור מחלקה עם כמה מאפיינים שמקבלים את ערכיהם מהקובץ web.config. אחר כך תשתמש במאפייני מחלקה זו בקוד כדי להחליף את הערכים המובנים שהיו בשימוש קודם.

1. בתיקייה App_Code, צור קובץ מחלקה חדש וקרא לו AppConfigurion.vb או AppConfigurion.cs. ב-C#, הסר את קוד פונקציית הבנייה שמוצלל בבלוק הקוד הבא:

```
public AppConfigurion()  
{  
    //  
    // TODO: Add constructor logic here  
    //  
}
```

אינך זקוק לפונקציית הבנייה, מכיוון שלמחלקה יהיו מאפיינים סטטיים בלבד.

2. בתוך מחלקה זו, מייד לאחר ההכרזה, צור מאפיין סטטי חדש לקריאה-בלבד (Shared static ב-C#), בשם FromAddress שמחזיר את הערך FromAddress מהקובץ web.config. להזכיר, בפרק 5 למדנו כי חבר (member) מסוג Shared או static (כמו שיטה או מאפיין) פועל על המחלקה עצמה ולא על מופע של אותה מחלקה.

VB.NET

```
Public Class AppConfigurion  
    Public Shared ReadOnly Property FromAddress() As String  
    Get  
        Dim result As String = ConfigurationManager.AppSettings.Get("FromAddress")  
        If Not String.IsNullOrEmpty(result) Then  
            Return result  
        End If  
    End Get  
End Class
```

```

        Throw New Exception("Appsetting FromAddress not found in web.config file")
    End Get
End Property
End Class

```

C#

```

public class AppConfiguration
{
    public static string FromAddress
    {
        get
        {
            string result = ConfigurationManager.AppSettings.Get("FromAddress");
            if (!string.IsNullOrEmpty(result))
            {
                return result;
            }
            throw new Exception("Appsetting FromAddress not found in web.config file");
        }
    }
    ...
}

```

3. חזור על הצעד הקודם, אבל הפעם צור את שלושת המאפיינים הבאים על ידי יצירת עותק של FromAddress:

```

FromName
ToAddress
ToName

```

אל תשכח לשנות את השם של כל אחד משלושת המקרים של FromAddress לשם המאפיין החדש.

4. כשאתה עדיין בתוך המחלקה AppConfiguration, צור מאפיין בוליאני בשם SendMailOnError:

VB.NET

```

Public Shared ReadOnly Property SendMailOnError() As Boolean
    Get
        Dim result As String = ConfigurationManager.AppSettings.Get("SendMailOnError")
        If Not String.IsNullOrEmpty(result) Then
            Return Convert.ToBoolean(result)
        End If
        Throw New Exception("Appsetting SendMailOnError not found in web.config file")
    End Get
End Property

```

C#

```

public static bool SendMailOnError
{

```

```

get
{
    string result = ConfigurationManager.AppSettings.Get("SendMailOnError");
    if (!string.IsNullOrEmpty(result))
    {
        return Convert.ToBoolean(result);
    }
    throw new Exception("Appsetting SendMailOnError not found in web.config file");
}
}

```

5. כשתסיים עם חמשת המאפיינים, שמור וסגור את הקובץ AppConfiguraton.
6. בתיקייה Controls, פתח את הקוד ברקע של ContactForm.aspx, אתר את הקוד שמגדיר את הכתובות From ו-To, והחלף את הערכים המובנים במקבילים להם ב-AppConfiguration:

VB.NET

```

myMessage.From = New MailAddress(AppConfiguration.FromAddress, AppConfiguration.FromName)
myMessage.To.Add(New MailAddress(AppConfiguration.ToAddress, AppConfiguration.ToName))

```

C#

```

myMessage.From = new MailAddress(AppConfiguration.FromAddress, AppConfiguration.FromName);
myMessage.To.Add(new MailAddress(AppConfiguration.ToAddress, AppConfiguration.ToName));

```

7. שמור את השינויים וסגור את הקובץ.
8. פתח את הקובץ Global.asax, ועטוף את כל הקוד של Application_Error בבדיקת If שמבטיחה כי SendMailOnError מוגדר כ-True:

VB.NET

```

Sub Application_Error(ByVal sender As Object, ByVal e As EventArgs)
    If AppConfiguration.SendMailOnError Then
        If HttpContext.Current.Server.GetLastError() IsNot Nothing Then
            ...
        End If
    End If
End Sub

```

C#

```

void Application_Error(object sender, EventArgs e)
{
    if (AppConfiguration.SendMailOnError)
    {
        if (HttpContext.Current.Server.GetLastError() != null)
            ...
    }
}

```

9. פתח את הקובץ web.config, החלף את האלמנט <appSettings /> הקיים בזה שלהלן, ושנה את כתובות הדואר האלקטרוני של FromAddress ו-ToAddress לאלו שלך.

```
<appSettings>
```

```
<add key="FromAddress" value="webmaster@planetwrox.com" />
<add key="FromName" value="Planet Wrox" />
<add key="ToAddress" value="webmaster@planetwrox.com" />
<add key="ToName" value="Planet Wrox" />
<add key="SendMailOnError" value="true" />
```

```
</appSettings>
```

10. שמור את כל השינויים וסגור את כל הקבצים הפתוחים. הקש Ctrl+F5 כדי לפתוח את דף הבית בדפדפן. גש לדף Contact ומלא את טופס ההתקשרות. אתה אמור לקבל דואר אלקטרוני בכתובת שציינת בצעד 9.

11. בקש בדפדפן דף שאינו קיים. לדוגמה, שנה את הכתובת בסרגל הכתובת של הדפדפן ל-DefaultTest.aspx. אתה אמור לקבל הודעת דואר אלקטרוני עם פרטי השגיאה, בדיוק כמו בפרק 17.

12. חזור ל- Visual Web Developer, פתח את web.config ושנה את הגדרת SendMailOnError מ-true ל-false.

```
<add key="SendMailOnError" value="false" />
```

13. שמור את כל השינויים, ובקש שוב דף שאינו קיים. מכיוון ששינית את הגדרת SendMailOnError, לא תקבל הודעת דואר אלקטרוני עם פרטי השגיאה.

כיצד זה פועל

מאפייני המחלקה AppConfiguration פונים לקובץ web.config לקבלת הגדרות היישום המבוקשות. כשהגדרה אינה מוגדרת, או שאינה מכילה ערך, כל מאפיין זורק חריג (exception). הדבר מסייע לגילוי מוקדם של הגדרות יישום חסרות. במקום החזרה שקטה של ערך ריק, מקבלים כעת חריג שמזכיר להוסיף את הגדרת היישום הנדרשת.

בזמן ביצוע, הקוד ניגש למאפיינים אלה בדרך זו:

VB.NET

```
myMessage.From = New MailAddress(AppConfiguration.FromAddress, AppConfiguration.FromName)
```

C#

```
myMessage.From = new MailAddress(AppConfiguration.FromAddress, AppConfiguration.FromName);
```

מכיוון שהמאפיינים הוגדרו כ-Shared (static ב-C#), תוכל לגשת אליהם ישירות במחלקה AppConfiguration, מבלי שתצטרך ליצור תחילה מופע חדש של AppConfiguration.

באפשרותך לגשת לאלמנט <appSettings> ב-web.config באופן ישיר מהקוד. לדוגמה, תוכל להשתמש ב- ConfigurationManager.AppSettings.Get("FromAddress") כדי לקבל ישירות את כתובת הדואר האלקטרוני שב-ContactForm.aspx. למרות זאת, אני מעדיף לעטוף במחלקה נפרדת

את האלמנטים <appSettings> שבמאפיינים משותפים. פתרון זה מאפשר ליהנות משירותי IntelliSense במחלקה AppConfig, ומקל על גילוי מאפייני התצורה הזמינים. הדבר גם מאפשר לכתוב קוד ממורכז שזורק חריגים כשהגדרות היישום הדרושות אינן נמצאות. אם תיגש ל-web.config ישירות מהקוד, תצטרך לבדוק את תקפות הערכים בכל פעם שתיגש להגדרה.

אותו עיקרון משמש גם בהגדרת SendMailOnError. כשמתרחשת שגיאה בזמן ביצוע, הקוד ב- Application_Error "מתייעץ" עם המאפיין SendMailOnError. מאפיין זה בתורו בודק את האלמנט <appSettings> של web.config כדי לקבוע אם לשלוח הודעת שגיאה בדואר האלקטרוני, או לא. מכיוון שהמאפיין SendMailOnError הוא בוליאני, הקוד משתמש ב- Convert.ToBoolean כדי להמיר את המחרוזת שהוחזרה מהקובץ web.config לערך בוליאני.

על ידי אחסון ערכים ב-web.config במקום לכתוב אותם ישירות בקוד, האתר נעשה קל יותר לתחזוקה ופריסה. כשהאתר ישוחרר לשרת ה-production, כל שתצטרך לעשות הוא ליצור עותק של web.config עבור סביבת ה-production ולשנות כמה הגדרות.

לאחר העברת הגדרות היישום המובנות לקובץ web.config המרכזי, הצעד הבא בתהליך הפריסה הוא יצירת עותק של האתר.

העתקת האתר

בזמן פיתוח האתר השתמשת בשרת האינטרנט המובנה שמגיע עם Visual Web Developer. שרת זה אמנם מצוין לפיתוח מקומי, אבל אינך יכול להשתמש בו בסביבת ייצור מכיוון שהוא מאזין רק לבקשות שבאות מהמחשב המארח המקומי. העברת האתר לייצור פירושה פריסתו במחשב שפועל בו שרת IIS (Internet Information Services), שהינו שרת האינטרנט המקצועי של מיקרוסופט. בסעיף זה תלמד כיצד להכין את האתר לפעולה בסביבת IIS, ואחר כך תלמד כיצד להתקין את IIS ולקבוע את התצורה של תוכנת השרת.

יעדי הפריסה המפורטים בטבלה הבאה זמינים על ידי VWD, לצורך פריסת אתר בשרת ייצור.

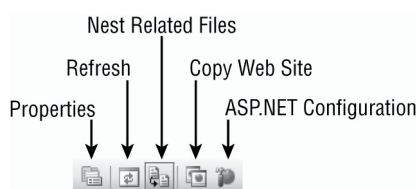
אפשרות פריסה	תיאור
File system	אפשרות ליצור עותק של האתר במערכת הקבצים המקומית של מחשב הפיתוח או של מחשב הרשת. אפשרות זו מיועדת להעברת הקבצים מאוחר יותר באופן ידני אל שרת הייצור.
Local IIS	אפשרות ליצור עותק של האתר שיפעל בסביבת שרת IIS המקומי.
FTP	אפשרות לשלוח את הקבצים שמרכיבים את יישום האינטרנט ישירות לשרת מרוחק על ידי שימוש ב-FTP.

אפשרות פריסה	תיאור
Remote Server	אפשרות לשלוח את הקבצים שמרכיבים את יישום האינטרנט לשרת IIS מרוחק. כדי שאפשרות זו תפעל, התוכנה Front Page Server Extensions צריכה להיות מותקנת בשרת המרוחק. למידע נוסף בקשר לאפשרות זו, בדוק את התיעוד שמגיע עם IIS, או התייעץ עם מנהלן השרת המרוחק.

משתמשי הגרסה המסחרית של Visual Studio, יכולים לגשת לארבע אפשרויות פריסה אלו משתי דרכי הפריסה שמציעה VWD: Copy Web Site ו-Publish Web Site. גרסת החינם Express Edition מאפשרת שימוש ב-Copy Web Site בלבד.

יצירת עותק פשוט של האתר

הפקודה **Copy Web Site** יוצרת עותק של האתר על ידי שימוש באחת מארבע אפשרויות ההעתקה. זוהי דרך מצוינת להעתקה מהירה של האתר למיקום אחר, כולל שרת הייצור או אפילו אל התקן מדיה נישא כמו USB stick (Disk-On-Key). תוכל לגשת לפקודה Copy Web Site מסרגל הכלים של סייר הפתרון, שמוצג בתרשים 3-18, או מהתפריט הראשי **Website**.



תרשים 3-18

לפני שיוצרים עותק של האתר, מומלץ לבדוק את מצבו. עליך לבצע הידור (compilation) מלא של האתר, באמצעות VWD שמהדרת בו את כל הקוד והדפים. הדבר עוזר לגלות בעיות כלשהן באתר לפני פריסתו בסביבת ייצור.

פריסת אתר היא הזדמנות טובה לביצוע עבודות ניהול משק בית. כדי להימנע מהאטת תהליך הפריסה, וכדי לשמור את האתר "רזה ונקי" ככל האפשר, עליך למחוק ממנו את כל הקבצים שאינך זקוק להם בייצור.

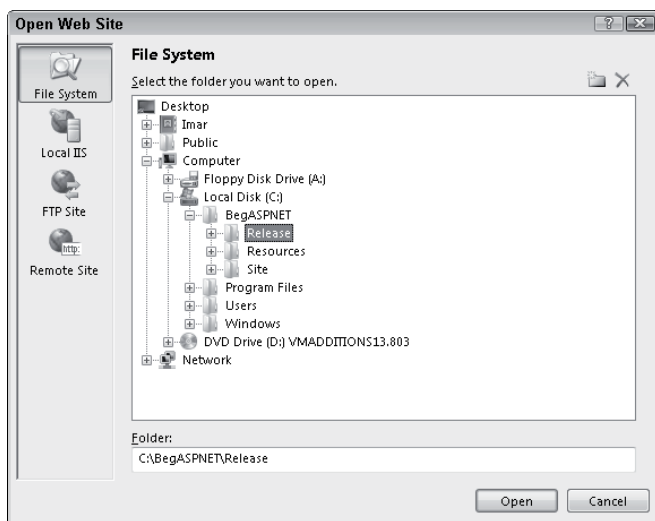
בתרגיל הבא תלמד להשתמש בפקודה Copy Web Site כדי ליצור עותק של כל הפרויקט Planet Wrox. בתרגיל מאוחר יותר תשתמש שוב בעותק, כשתקבע את תצורת IIS עבור אתר הייצור.

נסה זאת שימוש בפקודה Copy Web Site

בתרגיל זה תשתמש בפקודה Copy Web Site יחד עם האפשרות Local File System כדי לקבוע את מיקום היעד של העותק. שלוש אפשרויות ההעתקה האחרות (Local IIS, FTP) ו-Remote IIS פועלות באופן דומה. ההבדל הגדול לעומת אפשרויות אלו הוא בכך שהן שואלות פרטים

לגבי היעד, כמו שם משתמש וסיסמה, או אתר אינטרנט IIS שבו תרצה להשתמש. בעזרת העותק שתיצור בתרגיל זה, תוכל ליצור באופן ידני אתר אינטרנט IIS, כפי שתלמד לעשות מאוחר יותר בפרק.

1. סגור את כל הקבצים הפתוחים ב-VISUAL WEB DEVELOPER, ומהתפריט הראשי בחר Build ← Rebuild Web Site. פעולה זו כופה על VWD להדר את היישום כולו, אפילו אם כבר הודרו כמה מחלקיו. בעיות כלשהן שמתגלות באתר נרשמות על ידי VWD ב-Error List. כדי לאמת שהאתר נקי משגיאות, פתח את Error List (בחר View ← Error List מהתפריט הראשי) וודא שאין כל שגיאות הידור. תקן את כל השגיאות שעלולות לצוץ.
 2. כשרשימת השגיאות ריקה, בחר Website ← Copy Web Site, או לחץ על הסמל Copy Web Site בסרגל הכלים של סייר הפתרון, כפי שמוצג בתרשים 18-3.
 3. בראש תיבת הדו-שיח, לחץ על הלחצן Connect כדי לפתוח תיבת דו-שיח שמאפשרת לבחור את מיקום יעד האתר.
- בצד השמאלי של תיבת הדו-שיח, ודא שנבחרה File System. בצד הימני, אתר את התיקייה C:\BegASPNET, לחץ עליה פעם אחת כדי לבחור בה, ואז לחץ על Create New Folder (הלחצן עם סמל התיקייה הצהובה בפניה הימנית העליונה של תיבת הדו-שיח). הקלד Release והקש Enter כדי להחיל את השם החדש. תרשים 18-4 מראה את תיבת הדו-שיח בצורתה הסופית.
- לבסוף, לחץ Open כדי לבחור C:\BegASPNET\Release כיעד של פעולת ההעתקה.

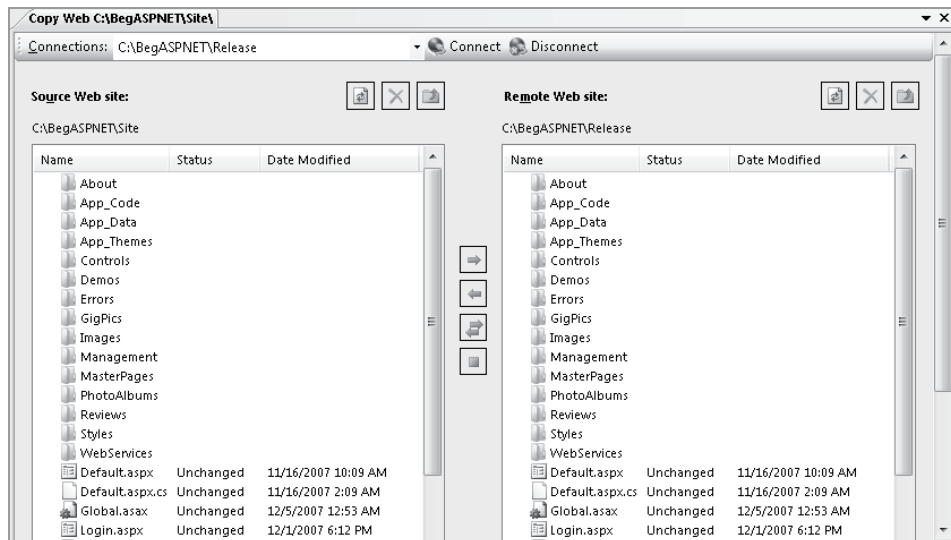


תרשים 18-4

4. בתיבת הדו-שיח Copy Web Site, התמקד ברשימה שמשמאל על ידי לחיצה על קובץ או תיקייה, ואז הקש Ctrl+A כדי לבחור בכל הקבצים ברשימה Source Web Site.
5. לחץ על הלחצן Synchronize Selected Files (הלחצן בעל שני החיצים שמצביעים בכיוונים מנוגדים) שבין שתי הרשימות, שנראה בתרשים 18-5. פעולה זו מתחילה את

תהליך הסנכרון. מכיוון שהתיקיה שמוצגת בצד הימני ריקה, כל הקבצים מהצד השמאלי מועתקים לצד הימני. כשפעולת ההעתקה תסתיים, תיבת הדו-שיח אמורה להיראות כמו תרשים 5-18.

6. פתח חלון של סייר החלונות ודפדף אל C:\BegASPNET\Release. אמת שכל הקבצים הרלוונטיים שמרכיבים את האתר נמצאים כאן.



תרשים 5-18

כיצד זה פועל

הפקודה Copy Web Site יוצרת עותק של כל הקבצים שמרכיבים את האתר. היא יכולה ליצור עותק של האתר במקומות שונים, כמו מערכת הקבצים המקומית, שרת FTP ושרת IIS. בתרגיל זה יצרת עותק בדיסק קשיח מקומי. שני הלחצנים הראשונים המסומנים בחיצים כחולים שבין שתי רשימות הקבצים שבתרשים 5-18 מאפשרים להעתיק קבצים מהמקור אל מיקום מרוחק כלשהו, ולהיפך. הלחצן השלישי, בעל שני החיצים, מאפשר לא רק להעתיק את הקבצים, אלא גם לסנכרן אותם. פירוש הדבר, שבמקרה הפעלה חוזרת של Copy Web Site, מסוגלת לקבוע איזה קבצים השתנו מאז ההפעלה הקודמת ולהעתיק רק אותם.

בתרגיל זה בחרת להעתיק את הקבצים למערכת המקומית. זו דרך מצוינת ליצור עותק מנותק מסביבת הפיתוח, שגם יכול להתבצע במחשב המקומי. כמובן שבאפשרותך להעתיק אותה קבוצת קבצים למחשב אחר באמצעות תוכנית FTP, באמצעות Disk-On-Key, ועוד. בנוסף, אם האתר יופעל במחשב אחר, ספק האירוח עשוי להציע לך ממשק מבוסס-web לטעינת הקבצים אל השרת שלו.

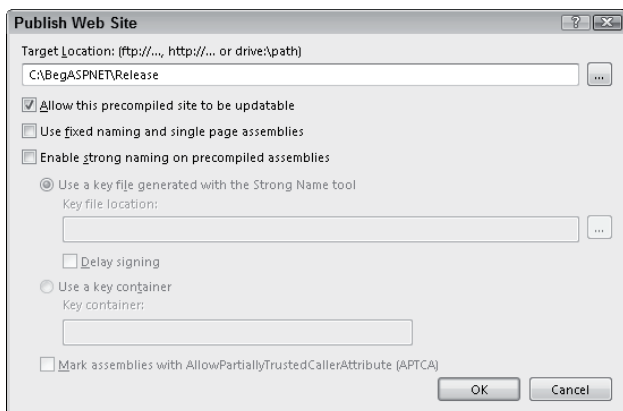
הדבר הטוב באפשרות העתקה זו הוא סנכרון הקבצים. ייתכן שהדבר לא ייראה בעל חשיבות כשמדובר ביצירת עותק במחשב המקומי, אבל כשיוצרים עותק בחיבור FTP איטי, תוכל ליהנות מכך. במקרה זה יועתקו רק קבצים חדשים או כאלה שהשתנו, ולא יועברו קבצים שלא השתנו.

העותק המקומי הנפרד מאפשר להכניס תחילה שינויים למספר קטן של קבצים (כמו web.config) ואז לטעון את הכל למחשב המארז.

הפצת האתר (publishing)

הפקודה **Publish Web Site** עניינה בהפצת האתר. היא זמינה רק בגרסאות המסחריות של Visual Studio ולא ב- Express Editions. היא דומה לפקודה Copy Web Site בכך שהיא יוצרת עותק של אתר האינטרנט שתוכל להשתמש בו לפריסה. היא שונה ממנה בכך שהיא מאפשרת הידור מראש (precompile) של היישום. כלומר, כל הקוד בקוד ברקע של דפי ASPX, פקדים, קבצי קוד בתיקיה App_Code וכד', מהודר לתוך מאספים (assemblies) של .NET, קבצים עם סיומת DLL בתיקיה bin של האתר. התועלות העיקריות של הידור מראש הן הגנת המקור (אחרים עם הרשאה לשרת אינם יכולים לגשת למקור), וביצועים משופרים החל מהפעם הראשונה שבה דף מבוקש. דפים שאינם מהודרים מראש, מהודרים תוך כדי ביצוע כשהם מבוקשים בפעם הראשונה, והדבר גוזל מעט זמן. זכור שמבקרים סדירים באתר לעולם לא יוכלו לראות את קוד המקור של היישום. כל מה שיוכלו לראות הוא את קוד ה-HTML הסופי שנשלח לדפדפן.

הפקודה Publish Web Site זמינה מתפריט **Build**, וגורמת להופעת תיבת הדו-שיח שמוצגת בתרשים 18-6.



תרשים 18-6

לחיצה על הסימן (...) שבקצה מיקום היעד גורמת להופעת תיבת הדו-שיח שמוצגת בתרשים 18-4, שמאפשרת לבחור מיקום יעד. כשנבחרת האפשרות Allow This Precompiled Site to Be Updateable (אפשר עדכון של אתר מהודר-מראש זה), VWD מהדרת את כל הקוד שכתוב ב-VB.NET וב-C# לתוך מאספים של .NET. ומציבה אותם בתיקיה bin שהיא יוצרת בשורש האתר. היא משאירה רק את סימני העריכה בקבצי ASPX ו-ASCX. כשאפשרות זו מכובה, כל קוד סימני העריכה מהודר גם הוא לתוך קבצי DLL. הקבצים הממשיים עדיין צריכים להיפרס לשרת, אבל תוכנם הוחלף בטקסט של שומר המקום הבא: "This is a marker file generated by the precompilation tool, and should not be deleted!". כשהדף מבוקש בדפדפן, רכיב

ASP.NET runtime מוצא את התוכן המתאים במאספים שבתקייה bin ומגיש את התוכן כאילו מדובר היה בדף רגיל. אפשרות אחרונה זו טובה במיוחד כשברצונך למנוע מאנשים אחרים לשנות את האתר. מכיוון שכל הקוד וסימני העריכה מהודרים לתוך קבצי DLL, אין דרך לשנותם עוד בשרת, אלא רק לטעון קבוצה חדשה של קבצים שנוצרו על ידי הפקודה.

העתקת האתר או הפצתו לתקייה חדשה במחשב המקומי היא רק צעד אחד בתהליך הפריסה. הצעד הבא הוא הגדרת התצורה של שרת האינטרנט באופן שיידע היכן לחפש את הקבצים.

הפעלת האתר בשליטת IIS

כדי לנפות ולבדוק את היישום, השתמשת עד כה בשרת האינטרנט (web server) המובנה שכלול בתוכנה VWD (Visual Web Developer). כדי למזער את השלכות האבטחה, מוגבלות הבקשות לשרת וזה לאלו שמגיעות מהמחשב המארח המקומי בלבד, לפיכך תצטרך להשתמש ב-IIS, שמגיע עם רוב הגרסאות העיקריות של Windows. כדי שהאתר יפעל בסביבת IIS, עליך לבצע שלושה צעדים:

1. להתקין ולקבוע את תצורת IIS.
2. להתקין את ASP.NET Framework 3.5 ולקבוע את תצורתה.
3. לקבוע הגדרות אבטחה.

המצב הנוכחי של המערכת קובע מהם הצעדים הדרושים בפועל. בסעיפים הבאים תלמד כיצד לבצע צעדים אלה.

התקנת שרת האינטרנט וקביעת התצורה שלו

השרת IIS מגיע אמנם עם רוב גרסאות Windows, אך הוא אינו מותקן כברירת מחדל, לפיכך עליך להתקינו תחילה. עליך גם לוודא שגרסת Windows שברשותך תומכת ב-IIS. שרת IIS נתמך על ידי Windows XP Professional אך לא על ידי Windows XP Home Edition. הגרסאות Starter ו-Home Basic של Windows Vista מכילות אמנם כמה רכיבים של IIS, אבל לא תוכל להפעיל בסביבתן דפים של ASP.NET, ולכן תזדקק לפחות ל-Windows Vista Home Premium Edition. שרת IIS נתמך במלואו בגרסאות מבוססות השרת של Windows.

כדי להתקין ולקבוע את תצורת IIS במחשב Windows עליך להיות מחובר כמנהל. לכן, אם החשבון שבאמצעותו אתה מתחבר למחשב אינו בעל הרשאות ניהול, עליך לבקש מהמנהל להתקין ולקבוע את תצורת IIS עבורך.

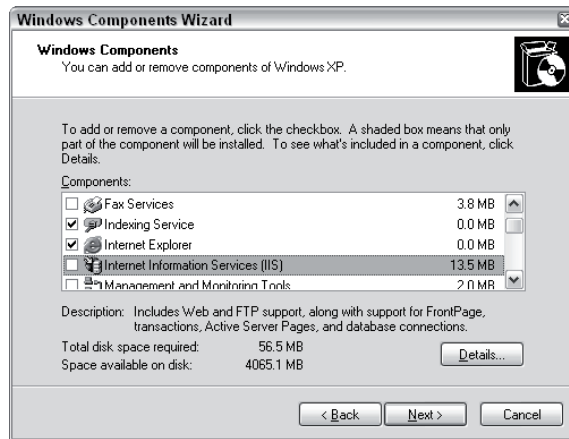
בנוסף להתקנת IIS תלמד גם כיצד ליצור ולהגדיר אתר אינטרנט ב-IIS. בגלל הדרך שבה פועלת האבטחה בסביבת Windows קרוב לוודאי שהאתר לא יפעל מייד לאחר הגדרת IIS, אלא אם כן תשנה כמה מהגדרות האבטחה של Windows. הנושא יידון בשני סעיפים מאוחרים יותר: "האבטחה ב-IIS" ובתרגיל נסה זאת "קביעת תצורת מערכת הקבצים".

כיצד לוודא ש-IIS מותקן

בסעיף זה תלמד כיצד לבדוק אם IIS מותקן בגרסאות החשובות ביותר של Windows: Windows XP, Windows Vista, Windows Server 2003 ו-Windows Server 2008. נוהל בדיקה זו והתקנת IIS שונים עבור כל אחת מגרסאות Windows. הסעיפים הבאים ינחו אותך דרך האפשרויות השונות. עליך לבחור באפשרות שתואמת את מערכת ההפעלה שלך.

Windows XP ו-Windows Server 2003

התקנת IIS מתבצעת מתוך הפריט Add or Remove Programs. מגיעים לפריט זה מלוח הבקרה, או על ידי בחירת Start ← Run ואז הקלדת **appwiz.cpl**. כשתימצא בתיבת הדו-שיח Add or Remove Programs, לחץ על הלחצן Add/Remove Windows Components שבצד השמאלי של התיבה, כדי להעלות את תיבת הדו-שיח Windows Component Wizard שמוצגת בתרשים 18-7.



תרשים 18-7

במערכת Windows XP אתה אמור לראות את הפריט Internet Information Services (IIS) ברשימת הרכיבים. במערכת Windows Server 2003 לחץ תחילה על Application Server ואז לחץ על הלחצן Details.

ב-Windows Server 2003 בדוק את הפריט ASP.NET. הן עבור Windows XP והן עבור Windows Server 2003, אמת שנבחרה האפשרות Internet Information Services (IIS) ואז לחץ על הלחצן Details. סמן לפחות את הפריטים הבאים:

- ⊙ Common Files
- ⊙ Internet Information Services Snap-In
(בשרת 2003 נקרא: Internet Information Services Manager)
- ⊙ World Wide Web Service

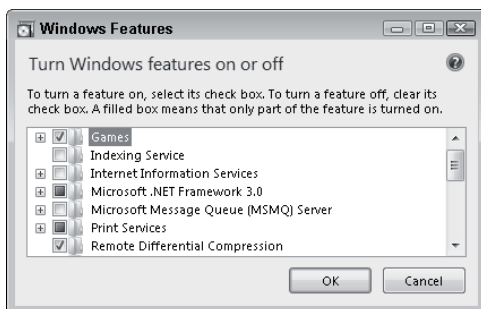
הערה: עבור הדוגמאות שבספר זה אינך זקוק לשרת SMTP (מכיוון שבאפשרותך להשתמש בשרת הדואר החיצוני של ספק האינטרנט, ISP, שלך), לפיכך תוכל לבטל את סימון האפשרות SMTP Service. נוהל אבטחה נכון הוא לא להתקין את שרת SMTP, שמסייע להקטין את סיכוני האבטחה הכרוכים בדרך כלל בהתקנת IIS.

לאחר בחירה באפשרות World Wide Web Service, לחץ על הלחצן Details שלה כדי לוודא שלפחות אפשרות זו נבחרה גם כן. שאר הפריטים הם רשות.

השלם את המשך הבחירה באשף Windows Component על ידי מעקב אחר הנחיות המסך. אתה עשוי להישאל לגבי מדיית ההתקנה של Windows. ברגע שסיימת עם האשף Windows Component, בצע אתחול מחדש של המחשב והתחבר שוב. שרת IIS מוכן כעת לפעולה. תוכל להמשיך עם הגדרת תצורת ASP.NET בסעיף "התקנת ASP.NET וקביעת התצורה".

Windows Vista

התקנת IIS מתבצעת מהפריט Programs and Features שכלוח הבקרה, או על ידי לחיצת Start, הקלדת appwiz.cpl בתיבה Search, והקשת Enter. כשתימצא בתיבת הדו-שיח Programs and Features, לחץ על הקישור Turn Windows features On or Off, כדי להעלות את תיבת הדו-שיח Windows Features, שמוצגת בתרשים 18-8.



תרשים 18-8

לחץ על הפריט Internet Information Services כדי לבחור בו. הפעולה גם בוחרת בכמה מתת-התכונות הנדרשות של IIS. עכשיו הרחב את World Wide Web Services ← Application Development Features ובחר ASP.NET. הפעולה גורמת גם לבחירת כמה תכונות נוספות של Development.

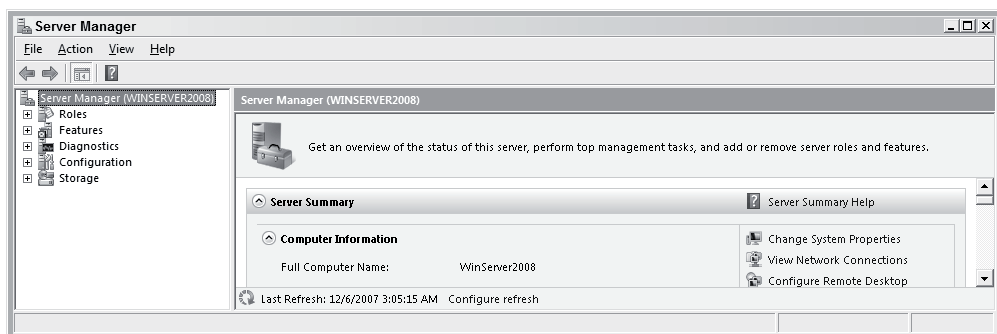
לבסוף, לחץ OK ו-Windows תתקין עבורך את התכונות הנדרשות. תוכל להמשיך עם קביעת תצורת ASP.NET בסעיף "התקנת ASP.NET וקביעת התצורה".

Windows Server 2008

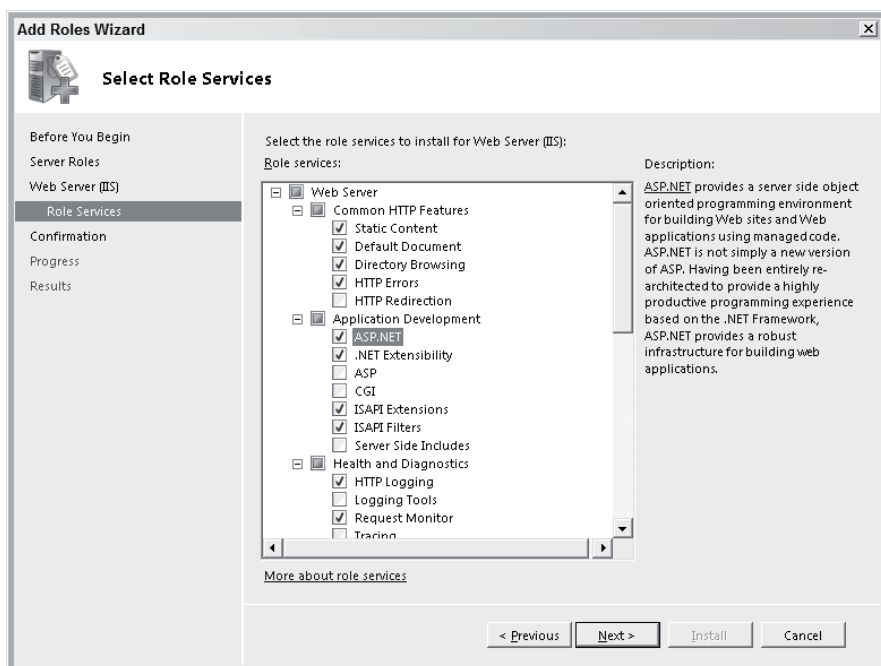
התקנת IIS מתבצעת מהפריט Programs and Features שכלוח הבקרה, או על ידי לחיצה על Start, הקלדת appwiz.cpl בתיבה Search, והקשת Enter. כשתימצא בתיבת הדו-שיח Programs

and Features, לחץ על הקישור Turn Windows features On or Off כדי להעלות את תיבת הדו-שיח Server Manager, שמוצגת בתרשים 9-18.

לחץ לחיצה ימנית על הפריט Roles בעץ שמשמאל, ומתפריט הקיצור בחר Add Roles. עקוב אחר הנחיות המסך והתקן את Web Server (IIS). כשתגיע למסך שבו תוכל לבחור Role Services שונים, ודא שמבין האפשרויות שתחת הצומת Application Development נבחרה לפחות ASP.NET, כפי שמוצג בתרשים 10-18.



תרשים 9-18



תרשים 10-18

פעולה זו מבטיחה את הבחירה של מספר תכונות נוספות שנדרשות על ידי IIS ועל ידי ASP.NET.

המשך עם Roles Wizard על ידי לחיצת Next עד ש-Windows מתחילה בהתקנת הקבצים הדרושים עבור התפקידים שנבחרו. כשהכל יהיה מוכן, ייתכן שתצטרך לאתחל את המחשב מחדש. כשתתחבר שוב, IIS יהיה מוכן לפעולה.

לאחר התקנת IIS בהצלחה, עליך לוודא שגם ASP.NET Framework 3.5 מותקנת.

התקנת ASP.NET וקביעת התצורה

אם התקנת את Visual Web Developer 2008 (אין חשיבות למהדורה) במחשב היעד, הרי ש-ASP.NET Framework 3.5 כבר מותקנת. אחרת, עליך להורידה מאתר מיקרוסופט <http://msdn.microsoft.com/net>. עקוב אחר הקישור Download או השתמש באפשרות החיפוש וחפש "download framework 3.5". ודא שטענת את גרסה 3.5 ולא בחרת בטעות גרסה מוקדמת של סביבת העבודה .NET. לאחר הורדת סביבת העבודה .NET, הפעל את תוכנית ההתקנה ועקוב אחר הוראות המסך.

אם התקנת IIS מתבצעת **לאחר** ש-ASP.NET Framework 3.5 כבר מותקנת במחשב, עליך לדווח ל-IIS שסביבת העבודה כבר קיימת. בדרך כלל הדבר נעשה במשך התקנת סביבת העבודה של .NET Framework. אם תתקין את IIS מאוחר יותר, תצטרך לעשות זאת באופן ידני. כדי לרשום את ASP.NET ב-IIS, בצע את הצעדים הבאים:

1. פתח שורת פקודה (command prompt). ב-Windows XP או ב-Windows Server 2003: בחר Start ← Run, הקלד cmd והקש Enter. ב-Windows Vista או Windows Server 2008: לחץ Start, הקלד cmd בתיבת החיפוש, ואז הקש Ctrl+Shift+Enter כדי להפעיל את שורת הפקודה עם הרשאות גבוהות יותר. כשתאשר את הפעולה תיפתח שורת פקודה כרגיל.

2. נווט לתיקייה NET Framework version 2.0 על ידי הקלדת הפקודה הבאה:

```
cd \Windows\Microsoft.NET\Framework\v2.0.50727
```

הקש Enter.

3. הקלד `aspnet_regiis -i` והקש שוב Enter.

לאחר זמן מה תקבל הודעה שאומרת כי ASP.NET הותקנה בהצלחה.

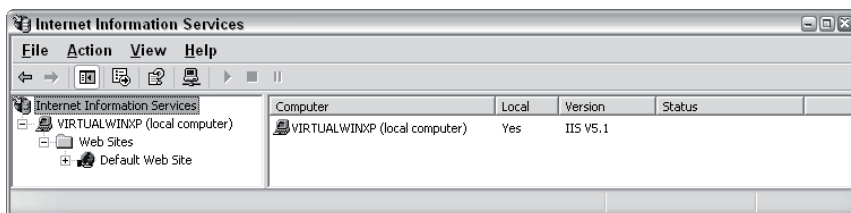
הערה: שים לב ש-2.0 בגרסת סביבת העבודה אינה שגיאה. אתרי אינטרנט שנוצרים עם ASP.NET Framework 3.5 עדיין פועלים במסגרת גרסה 2.0 של ASP.NET runtime שאת תצורתה אנחנו קובעים כאן.

הצעד הבא, לאחר התקנת IIS וסביבת העבודה NET וקביעת התצורה שלהם, הוא יצירת אתר האינטרנט בסביבת IIS. תלמד כיצד לעשות זאת בתרגיל **נסה זאת** שלהלן. גם כעת עליך לבחור בהנחיות שתואמות למערכת ההפעלה שלך. לאחר תרגיל **נסה זאת** הבא, תוסיף ותלמד על הגדרת הרשאות אבטחה עבור המערכת שלך.

נסה זאת קביעת תצורת אתר במערכת Windows XP או Server 2003

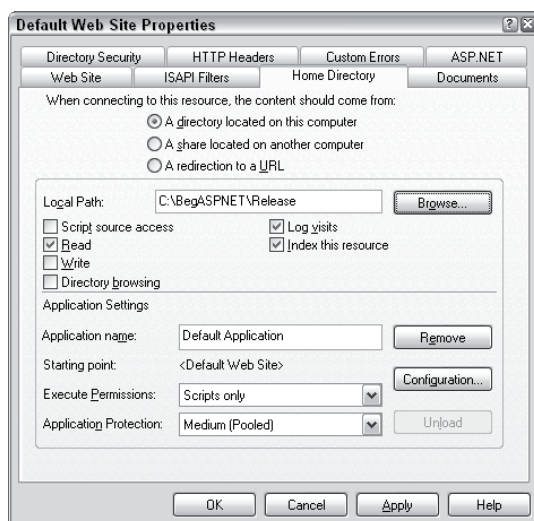
רוב הצעדים ב-Windows XP זהים לאלה של Windows Server 2003. עם זאת, שים לב שהמסכים שתראה בתרגיל זה נלקחו מסביבת Windows XP ועשויים להיות שונים במקצת מ-Windows Server 2003. מכיוון ש-Windows XP מאפשרת להפעיל רק אתר אינטרנט אחד בכל רגע נתון, הרי שבתרגיל זה תשנה את אתר ברירת המחדל. רשום לעצמך את כל השינויים שתעשה, כדי שתוכל להחזיר את המצב לקדמותו כשתרצה.

1. פתח את Internet Information Services Manager. תמצא פריט זה בלוח הבקרה תחת Administrative Tools. לחילופין, לחץ Start ← Run, הקלד **inetmgr** והקש Enter. כעת יופיע Internet Information Services Manager.
2. בעץ הניווט הרחב את השרת שלך, ואז הרחב את הצומת Web Sites עד שתראה את Default Web Site שמוצג בתרשים 11-18.



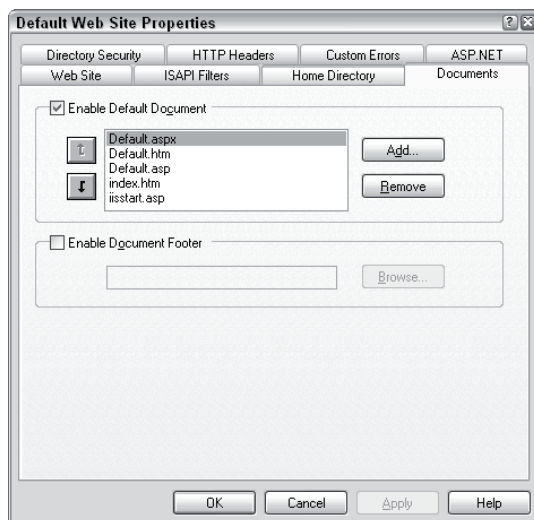
תרשים 11-18

3. לחץ לחיצה ימנית על Default Web Site ובתפריט הקיצור בחר Properties. עבור לכרטיסייה Home Directory, ובאזור Local Path לחץ על הלחצן Browse ודפדף לתיקייה Release שיצרת קודם במיקום C:\BegASPNET\Release. כשתסיים, תיבת הדו-שיח צריכה להיראות כמו תרשים 12-18.



תרשים 12-18

4. ודא שנבחרה הגרסה הנכונה של ASP.NET. כדי לעשות זאת, עבור לכרטיסייה ASP.NET ובדוק שהגרסה שנבחרה מהרשימה הנפתחת היא 2.0.50727. אם אינך רואה אפשרות זו כלל, פנה לסעיף "התקנת ASP.NET וקביעת התצורה" מוקדם יותר בפרק.
5. הדבר הבא שיש לבדוק הוא ששרת IIS משתמש בדף ברירת המחדל הנכון, זה שמוגש כשהבקשה מכילה רק שם תיקייה או את שורש האתר ללא ציון שם דף. האתר Planet Wrox משתמש ב-Default.aspx שהוא שם מסמך ברירות המחדל הנפוץ ביותר. כדי לבדוק זאת, עבור לכרטיסייה Documents. אם Default.aspx אינה רשומה, לחץ על הלחצן Add, הכנס Default.aspx ולחץ OK כדי להוסיף את הפריט לרשימה. לבסוף, השתמש בלחצן שמסומן בו חץ שמצביע מעלה, כדי להעביר את הפריט לראש הרשימה, כפי שמוצג בתרשים 13-18.

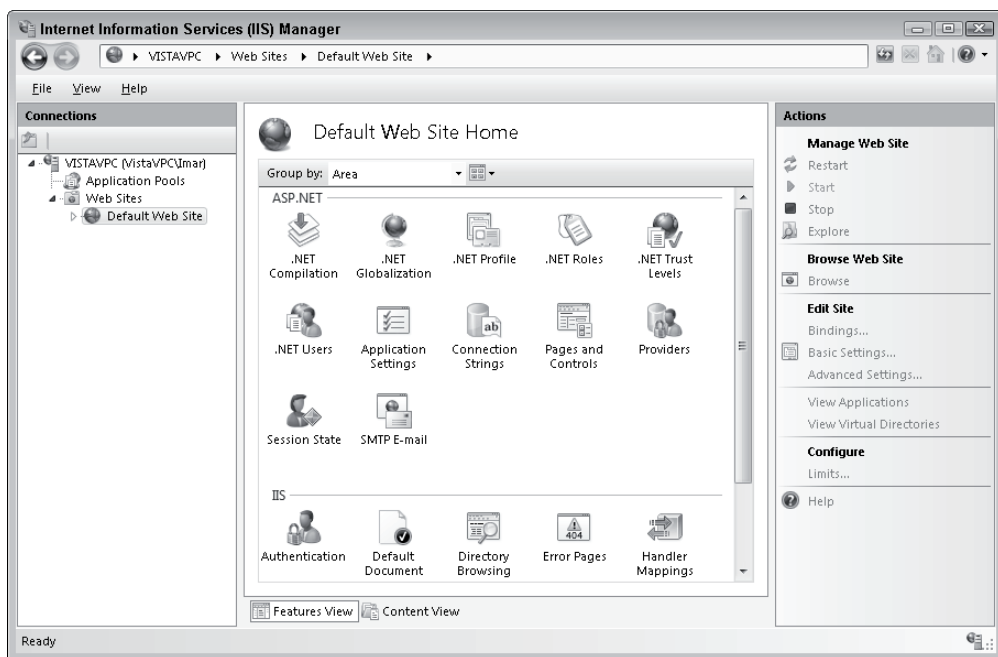


תרשים 13-18

6. לחץ OK כדי להחיל את השינויים. אם תופיע תיבת דו-שיח לגבי דריסת הערכים עבור צומת בן כלשהו, לחץ OK כדי למנוע החלת הגדרה זו על צמתי בן.
7. כשמתמשים ב- Windows Server 2003 נותר לוודא ש-ASP.NET מורשה להתבצע. כדי לבדוק זאת, לחץ על הצומת Web Service Extensions של Internet Information Services שתחת שם השרת שלך. כעת תופיע רשימה עם Web Service Extensions שזמינים כרגע במחשב. ודא כי ASP.NET v2.0.50727 מוגדרת כ-Allowed.
- בנקודה זו, IIS הוגדר באופן נכון. עם זאת, האתר לא יפעל ככל הנראה, מכיוון שעליך להגדיר עדיין את מערכת הקבצים. תלמד כיצד לעשות זאת לאחר תרגיל נסה זאת הבא (שמסביר כיצד להגדיר את IIS בסביבת Windows Vista או Server 2008) וההסבר שבסופו.

בתרגיל זה תלמד כיצד לקבוע את תצורת "אתר ברירת המחדל" התקני שמגיע עם IIS. ב- Windows Vista או Windows Server 2008 ניתן ליצור בסביבת IIS יותר מאתר אחד, אולם אפשרות זו אינה נידונה כאן. כדי לדעת יותר על יצירת כמה אתרי אינטרנט בסביבת IIS פנה למנהלן המערכת או קרא את מידע העזרה שכלול ב-IIS. רוב הצעדים ב- Windows Vista זהים לאלה שב- Windows Server 2008. עם זאת, דע שהמסכים שתראה בתרגיל זה נלקחו מ- Windows Vista ועשויים להיות שונים במקצת ב- Windows Server 2008.

1. פתח את Internet Information Services Manager. תמצא פריט זה בלוח הבקרה תחת Administrative Tools אם זה ב- Classic View. לחץ Start → Run, הקלד **inetmgr**, והקש Enter.
2. הרחב את העץ משמאל עד שתראה את Default Web Site, כפי שמוצג בתרשים 14-18.
3. לחץ לחיצה ימנית על הפריט Default Web Site בעץ שמשמאל, ומתפריט הקיצור בחר Advanced Settings. תוכל לפתוח את אותה תיבת דו-שיח על ידי לחיצת Advanced Settings בחלונית Actions שבימין בתרשים 14-18.

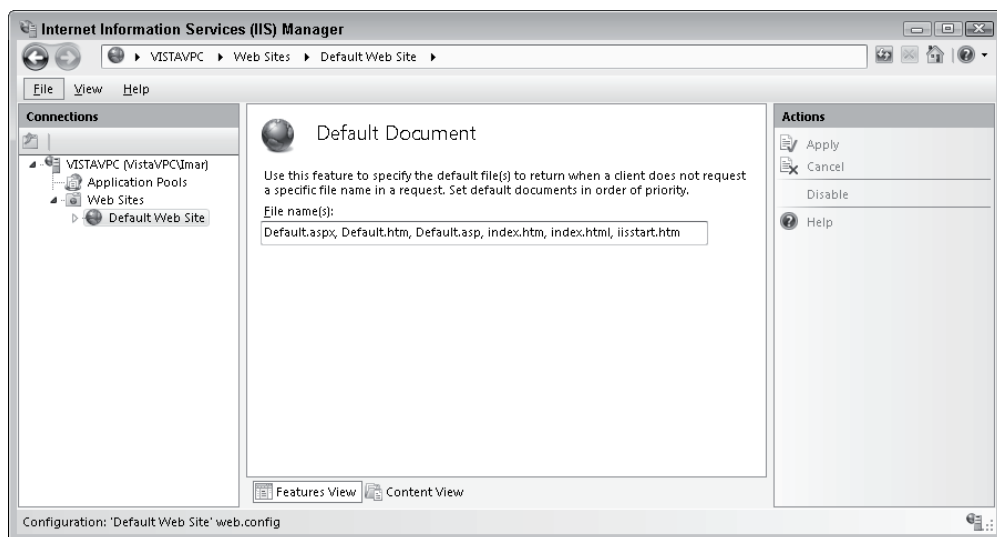


תרשים 14-18

4. בתיבת הדו-שיח Advanced Settings, לחץ על המאפיין Physical Path, ואז לחץ על הסימן (...) כדי לבחור בתיקייה C:\BegASPNET\Release ולחץ OK. לחץ שוב OK כדי לסגור את תיבת הדו-שיח Advanced Settings.

5. ב- Windows Vista וב- Server 2008 מאפשר כלי הניהול גישה למספר רב של אפשרויות ASP.NET Configuration. תרשים 14-18 מראה את הפריטים שאפשר לגשת אליהם מכאן, כולל .NET Profile, .NET Roles, .NET Users, Application Settings ו- Connection Strings. כדי לראות כיצד זה פועל, לחץ לחיצה כפולה על הפריט .NET Users. כעת תראה את חשבונות המשתמשים שיצרת קודם. לחיצה כפולה על משתמש מאפשרת לנהל את התפקידים של משתמש זה. באופן דומה, לחיצה כפולה על הפריט Application Settings מאפשרת לשנות את כתובות הדואר האלקטרוני שהעברת ל- web.config מוקדם יותר בפרק.

6. הצעד הבא הוא לבדוק ש-IIS עוצב באופן שישתמש במסמך ברירות מחדל הגיוני, כפי שמוגש כשמבקשים שם תיקייה או את שורש האתר. האתר Planet Wrox משתמש ב- Default.aspx שהוא שם מסמך ברירות המחדל הנפוץ ביותר באתרי אינטרנט מבוססי ASP.NET. כדי לבדוק זאת, לחץ לחיצה כפולה על האפשרות Default Document ברשימה של IIS, שנראית בתרשים 14-18. ודא ש- Default.aspx נוכח ונמצא בראש הרשימה. אם הפריט אינו ברשימה, או שאינו נמצא בראשה, הקלד אותו באופן ידני בתיבת הטקסט שבתחילת הרשימה ולחץ על הקישור Apply בחלונית Actions שבצד ימין (ראה תרשים 15-18).



תרשים 15-18

7. סגור את Internet Information Services Manager מכיוון שתצורת האתר הוגדרה כהלכה ככל שהדבר נוגע ל-IIS. עם זאת, האתר עדיין לא יפעל כהלכה מכיוון שעליך להגדיר הרשאות אבטחה עבור מערכת הקבצים, כפי שתראה מאוחר יותר.

כיצד זה פועל

תרגיל נסה זאת זה הופיע בשתי גרסאות, גרסת Windows XP / Server 2003 וגרסת Windows Vista / Server 2008. בין אם בחרת בזו או בזו, הצעדים שעשית דומים מאוד. התחלת בעיצוב אתר אינטרנט ב-IIS. לכל התקנה חדשה של IIS יש Default Web Site, האתר שמאזין ל- <http://localhost> כברירת מחדל. ב-Windows XP אתה "תקוע" עם האתר הזה, אבל מערכות הפעלה אחרות מאפשרות ליצור מספר רב של אתרים.

בצעד הבא גרמת לשורש האתר להצביע על התיקייה Release שמכילה את האתר. כשמיפוי זה נקבע, IIS מסוגל לראות איזה קבצים עליו להגיש כשמבקשים URL כמו <http://localhost>. פירוש הדבר ש-URL כמו <http://localhost/Login.aspx> ממופה לקובץ הפיזי ב- `C:\BegASPNET\Release\Login.aspx`.

בסיום התרגיל עיצבת מסמך ברירת מחדל, הקובץ שמוגש כשמבקשים URL ללא שם קובץ מפורש, דוגמת <http://localhost/Reviews> או <http://localhost/Default.aspx>. קביעת Default.aspx כברירת המחדל תגרום ל-IIS לנסות למצוא וגם להציג קובץ בשם זה במקום השם הלא מפורש.

הדבר האחרון שעליך לעשות כדי לוודא שהאתר יפעל בהתקנת IIS המקומי, הוא לקבוע את הגדרות האבטחה. נושא זה יידון בשני הסעיפים הבאים.

האבטחה ב-IIS

כתוצאה מהשילוב הטוב של שרת האינטרנט המובנה בתוכנה Visual Web Developer 2008, ייתכן שלא תהיה ער לכל מה שקורה בקרביים של המערכת, ואיזה הגדרות אבטחה תקפות. כדי להשתמש במשאבים באתר - כמו קבצי ASPX, קבצי קוד ברקע, בסיס הנתונים שבתיקייה App_Data, והתמונות שבאתר - זקוק שרת האינטרנט להרשאות של Windows כדי לגשת למשאבים אלה. פירוש הדבר, שעליך להגדיר את Windows באופן שתעניק גישה למשאבים אלה לחשבון ששרת האינטרנט משתמש בו. אבל מהו בדיוק החשבון הזה? החשבון המסוים שזקוק להרשאה תלוי בכמה גורמים, ובכלל זה גרסת Windows, סביבת הפעולה של האתר (סביבת IIS או שרת האינטרנט המובנה), והגדרות של IIS.

עם זאת, ברוב המקרים יש לשקול שני תרחישים בלבד: שימוש בשרת האינטרנט המובנה, או שימוש ב-IIS כשרת האינטרנט.

במקרה הראשון, החשבון ששרת האינטרנט המובנה משתמש בו הוא החשבון שמשמש לך להתחברות אל מחשב Windows שלך. חשבון זה דומה בדרך כלל להגדרה כזו: `DomainName\UserName` או `MachineName\UserName`. התחברות ל-Windows עם חשבון זה גורמת להפעלת VWD 2008, אשר בתורה מפעילה את שרת האינטרנט המובנה. פירוש הדבר, ששרת האינטרנט כולו פועל עם ההרשאות שלך. מכיוון שסביר שאתה מנהלן במחשב

Windows המקומי, ולכן גם בעל הרשאות גישה לכל הקבצים שמרכיבים את האתר, הדברים ככל הנראה פעלו היטב עד כה ללא כל שינויים בהגדרות האבטחה.

במקרה השני, כששתמשים ב-IIS, הדברים שונים לגמרי. כברירת מחדל, יישום ASP.NET פועל בסביבת IIS עם חשבון מיוחד שנוצר בזמן התקנת IIS. במערכת Windows XP החשבון נקרא ASP.NET. ובמערכות Windows Vista, Windows Server 2003 ו-Windows Server 2008 החשבון נקרא Network Service. בנוסף לחשבון שמשמש עבור יישומי ASP.NET, עליך גם להגדיר חשבון שימש את שרת האינטרנט עבור משאבים שאינם קשורים ישירות ל-ASP.NET, כמו תמונות, קבצי CSS וכד'. הטבלה הבאה מפרטת את החשבונות שבהם משתמשים שרת האינטרנט המובנה, ASP.NET, ו-IIS עבור קבצים שאינם קשורים ל-ASP.NET.

מערכת ההפעלה	מבנה שם המשתמש	שם ברירת המחדל של ASP.NET	גישה לשרת באמצעות
Windows XP	DomainName\UserName or MachineName\UserName	ASPNET	IUSR_MachineName
Windows Vista		Network Service	IUSR
Windows Server 2003		Network Service	IUSR_MachineName
Windows Server 2008		Network Service	IUSR

שים לב שמערכות Windows XP ו-Windows Server 2003 משתמשות בשם החשבון בסיומת **IUSR_MachineName** (MachineName מציינת את שם המחשב שלך). Windows Vista ו-Windows Server 2008 משתמשות בסיומת IUSR בלבד.

לדוגמה, אם ברצונך להגדיר את IIS ב-Windows Vista, עליך להשתמש ב-Network Service כשם החשבון עבור ASP.NET, וב-IUSR (ללא הסיומת MachineName_) כשם החשבון של שרת האינטרנט.

לאחר גילוי החשבונות שעליך לעצב, הצעד האחרון הוא הגדרת תצורת מערכת הקבצים.

הגדרות NTFS עבור Planet Wrox

מבלי להתחשב בחשבון שאתה משתמש בו, הרי כדי ששרת האינטרנט יוכל לגשת למשאבים שלך, עליך לשנות את מערכת הקבצים של Windows. הדבר נחוץ רק כאשר הדיסק הקשיח שלך מפורמט על ידי מערכת הקבצים NTFS, ולא על ידי FAT או FAT32, מערכות הקבצים הישנות של מיקרוסופט. בתרגיל מאוחר יותר בסעיף זה תלמד כיצד לזהות את סוג הכונן.

כדי להגדיר בהצלחה מערכת קבצים מסוג NTFS עבור אתר האינטרנט Planet Wrox, צריך להעניק את ההרשאות הבאות לחשבונות שרת האינטרנט שקבעת בסעיף הקודם.

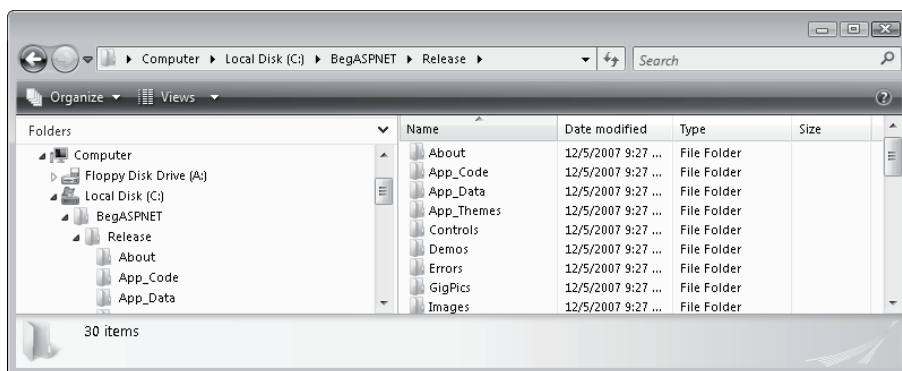
שם התיקייה	הרשאות	הסבר
C:\BegASPNET\Release	List folder contents; Read	שרת האינטרנט וחשבונות ASP.NET צריכים לקרוא את כל הקבצים והתיקיות שמרכיבים את אתר האינטרנט. צריך להגדיר תיקיות בנים, כמו Reviews, באופן שיוכלו לרשת הגדרות אלו.
C:\BegASPNET\App_Data C:\BegASPNET\GigPics	Modify; List folder contents; Read; Write	חשבון ASP.NET צריך לקרוא ולכתוב מ/אל בסיס נתונים SQL Server 2005 בתיקייה App_Data. החשבון גם צריך לשמור את התמונות המועלות בתיקייה GigPics.

בתרגיל הבא תלמד כיצד לקבוע את הגדרות האבטחה עבור תיקיות אלו.

נסה זאת קביעת תצורת מערכת הקבצים

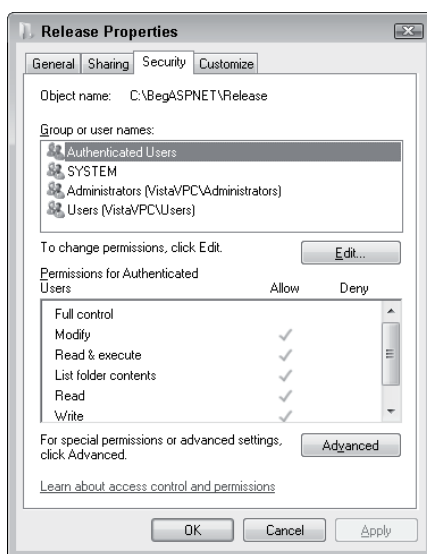
בתרגיל זה תלמד להגדיר את תצורת מערכת הקבצים עבור אתר האינטרנט Planet Wrox. בתרגיל תראה עותקים של תצוגות מסך של Windows Vista. לגרסאות האחרות של Windows יש מסכים דומים. אם תתקשה בכיצוע הצעדים הבאים, פנה ל"עזרה" של Windows וחפש שם את "security settings NTFS", או התקשר עם המנהלן שלך.

1. פתח את סייר החלונות ואתר את כונן C שלך. לחץ עליו לחיצה ימנית ומתפריט הקיצור בחר Properties. ודא שתחת File system מסומן NTFS. אם תראה שם את מערכות הקבצים הישנות FAT או FAT32, דלג על כל התרגיל הזה מכיוון ש-FAT אינה מאפשרת לשנות את הגדרות האבטחה.



תרשים 16-18

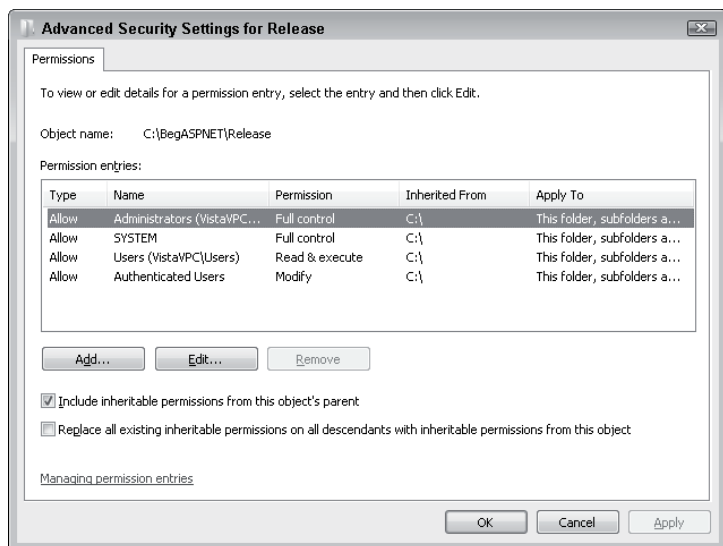
2. באותו חלון של סיר החלונות, דפדף אל התיקייה Release לפי הכתובת C:\BegASPNET\Release, שנראית בתרשים 16-18.
3. לחץ לחיצה ימנית על התיקייה Release, ומתפריט הקיצור בחר Properties, ועבור לכרטיסייה Security. אם אינך רואה את הכרטיסייה Security במערכת Windows XP, סגור את תיבת הדו-שיח ובחר Tools – Folder Options < View וגלול מטה עד סוף הרשימה Advanced settings. ודא שהאפשרות Use simple file sharing (Recommended) אינה מסומנת. פתח שוב את תיבת הדו-שיח Properties של התיקייה Release. כעת תראה את הכרטיסייה Security כפי שהיא מוצגת בתרשים 17-18.



תרשים 17-18

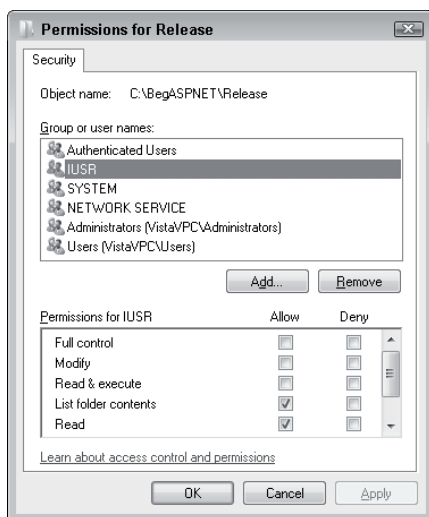
4. כברירת מחדל, התיקייה Release יורשת את ההגדרות שלה מתיקיית האב (C:\BegASPNET), אשר בתורה מקבלת את ההגדרות שלה מהשורש של כונן C. כדי לשבור את שרשרת ההורשה הזו, לחץ על הלחצן Advanced כדי לפתוח את תיבת הדו-שיח Advanced Security Settings. במערכות Windows Vista ו- Windows Server 2008 לחץ על הלחצן Edit שבתוכה כדי להעביר את המסך למצב עריכה. כעת יוצג מסך דומה לתרשים 18-18.
5. נקה את האפשרות Include Inheritable Permissions From This Object's Parent (ב- Windows XP ההגדרה נושאת את התווית "Inherit from parent the permission entries that apply to child objects", וב- Windows Server 2003 התווית היא "Allow inheritable permissions from the parent to propagate to this object and all child objects"). לחץ OK. תופיע תיבת דו-שיח שתשאל אם ברצונך להעתיק (Copy) או להסיר (Remove) את ההגדרות הקיימות. בחר **Copy** וסגור את כל תיבות הדו-שיח עד שתחזור אל תיבת הדו-שיח Release Properties שמוצגת בתרשים 17-18.

6. הצעד הבא הוא הוספת חשבונות. ב- Windows Vista וב- Windows Server 2008 לחץ על הלחצן Edit, שמוצג בתרשים 17-18, ואז לחץ על הלחצן Add. ב- Windows XP וב- Windows Server 2003, תוכל ללחוץ ישירות על הלחצן Add. הקלד את שם החשבון, ASPNET או Network Service, בהתאם לגרסת Windows שבשימושך. במקום להקליד את השם באופן ידני תוכל לחפש אותו. לחץ Advanced ואז לחץ על הלחצן Find Now. ברגע שהחשבון הנכון נמצא בתיבת הטקסט, לחץ על הלחצן OK.



תרשים 18-18

הערה: אם אינך רואה את החשבון ASPNET ב- Windows XP, ודא שההתקנה והרישום של NET Framework נעשו כהלכה. במקרה של ספק, הפעל את הכלי i-regiis_aspnet כפי שהוצג קודם.



תרשים 18-19

כאשר החשבון שנבחר מופיע ברשימה Group or user names, ודא שרק List Folder Read-ו Contents נבחרו (סומנו).

חזור על צעד זה והוסף את החשבונות IUSR_MachineName או IUSR בהתאם לגרסת Windows שלך. בסופו של דבר אמורה תיבת הדו-שיח להיראות בדומה לתרשים 18-19.

7. סגור את כל תיבות הדו-שיח הפתוחות, פרט לתיבת הדו-שיח Release Properties שמוצגת בתרשים 17-18.

8. לחץ על הלחצן Advanced כדי לפתוח שוב את תיבת הדו-שיח Advanced Security Settings. ב- Windows Vista או Windows Server 2008 לחץ על הלחצן Edit בתיבה כדי להעבירה למצב עריכה. בחר בתיבת הסימון השנייה מתרשים 18-18. הדבר כופה על Windows להחיל את אותן הגדרות אבטחה על כל תת-הקבצים ותת-התיקיות, תוך החלפת כל ההגדרות הקיימות. לחץ OK ואז אשר את השינויים שייערכו. לבסוף, לחץ על כל תיבות הדו-שיח הפתוחות שנשארו.

9. חזור לסייר החלונות, ולחץ לחיצה ימנית על App_Data מהתיקייה Release. מתפריט הקיצור פתח את Security settings שלה, ושנה את ההרשאות עבור החשבון ASPNET או Network Service על ידי הוספת הרשאות Read-ו Write-ו Execute-ו (Read-ו Write-ו Execute-ו). עבור Windows Vista ו- Windows Server 2003, עליך ללחוץ תחילה על הלחצן Edit כדי להעלות את תיבת הדו-שיח Properties במצב עריכה. חזור על צעד זה עבור התיקייה GigPics.

10. בשלב זה אתר האינטרנט שלך צריך לפעול. כדי לאמת זאת, פתח דפדפן ועבור אל <http://localhost>. אתר האינטרנט Planet Wrox צריך להופיע. כדי לאמת שהכל כשורה, דפדף באתר על ידי בקשת דפים מהתפריט הראשי, מלא את טופס הקשר, צור אלבום חדש, העלה תמונות ועוד. אם תקבל הודעת שגיאה, פנה לסעיף "איתור וטיפול בשגיאות שרת אינטרנט", להלן.

אם עדיין לא הצלחת לגרום לאתר לפעול, נסה להגדיר את מערכת הקבצים עבור הקבוצה Everyone. למרות שמנקודת הראות של אבטחה זו אינה הקבוצה המומלצת לשימוש בסביבת ייצור, אך פעולה זו תעזור לך לקבוע אם לפניך בעיית אבטחה, או לא. אם האתר יפעל עבור החשבון Everyone, זו אכן בעיית שקשורה לאבטחה, ותצטרך לוודא שהגדרת את החשבונות הנכונים.

כיצד זה פועל

כל הקבצים והתיקיות במערכת Windows מוגנים על ידי מערכת הקבצים NTFS של Windows. כדי להבטיח פעולה נכונה של אתר האינטרנט, עליך להעניק לחשבון שבשימוש שרת האינטרנט את ההרשאות הדרושות לפעולה עם הקבצים והתיקיות של האתר. עבור רוב הקבצים והתיקיות מספיקה הרשאת Read. עם זאת, עליך לשנות את ההרשאות עבור שתי תיקיות. בזמן ביצוע יש כתיבה ל- App_Data ול- GigPics, לפיכך עליך להעניק הרשאות Write-ו Modify לשתי תיקיות אלו.

איתור וטיפול בשגיאות שרת אינטרנט

כשמנסים לקבוע את תצורת שרת האינטרנט, עלולות להיות מספר תקלות. סעיף זה מפרט את התקלות השכיחות ביותר ומספק הסבר כיצד להתגבר עליהן. עם זאת, השגיאות או התקלות האלו עלולות להיגרם כתוצאה ממספר רב של סיבות, אשר לא ניתן לדון בכולן במסגרת ספר זה. בכל תת-סעיף מוצג נוסח הודעת המערכת ולאחריו הסבר והדרכה.

- ⊙ **It is an error to use a section registered as allowDefinition='MachineToApplication' beyond application level**

שגיאה זו מתקבלת כשהאתר אינו נמצא בשורש שרת האינטרנט, או כשלא הגדרת את התיקיה כיישום נפרד. על פי נתוני התצורה הנוכחית של האתר Planet Wrox, שגיאה זו עלולה להופיע כאשר לדוגמה, תמפה את המיקום הפיזי של האתר דרך שרת האינטרנט (IIS) אל הכתובת C:\BegASPNET ואז תדפדף אל <http://localhost/Release>. כדי לתקן שגיאה זו, ודא ששורש האתר בסביבת IIS מצביע על התיקיה שמכילה את קובץ web.config הראשי, במקרה שלנו C:\BegASPNET\Release. אותה שגיאה תופיע גם כשתפתח תיקיה לא נכונה ב-VWD. לדוגמה, אם תפתח את C:\BegASPNET ותדפדף לתיקיה /Site. במקום זאת, פתח ב-VWD את C:\BegASPNET\Site כאתר האינטרנט.

- ⊙ **HTTP Error 401.3–Unauthorized**

שגיאה זו מופיעה כשלחשבון ששרת האינטרנט משתמש בו אין הרשאה לקרוא את הקבצים מהדיסק. כדי לתקן בעיה זו, פנה לתרגיל "קביעת תצורת מערכת הקבצים" לעיל, וקבע את ההרשאות המתאימות.

- ⊙ **Failed to update database**
"C:\BEGASPNET\RELEASE\APP_DATA\ASPNETDB.MDF"
because the database is read-only

שגיאה זו מופיעה כאשר קבצי בסיס הנתונים מסומנים לקריאה-בלבד, או כשלחשבון שבשימוש שרת האינטרנט אין הרשאת כתיבה בקבצי בסיס הנתונים. במקרה הראשון, פתח את Properties של הקובץ בסייר החלונות ובדוק אם תיבת הסימון Read Only נקיייה. במקרה השני, ודא שהחשבון שבשימוש ASP.NET הוא לפחות בעל הרשאת Modify לתיקיה App_Data.

- ⊙ **HTTP Error 403.14–Forbidden**

ממבט ראשון שגיאה זו מצביעה לכאורה על בעיה בהרשאות NTFS, אולם לעתים קרובות היא נגרמת כתוצאה מכך שאין דף ברירת המחדל או מכיוון ששם הדף שגוי. אם שגיאה זו הופיעה, ודא שהאתר או התיקיה שאתה ניגש אליהם מכילים דף בשם Default.aspx, וכי שם מסמך זה הוגדר ב-IIS כשם מסמך ברירות המחדל.

- ⊙ **HTTP Error 404.0–Not Found**

שגיאה זו מופיעה כשמבקשים קובץ או תיקיה שאינם קיימים, כמו <http://localhost/DoesNotExist> או <http://localhost/DoesNotExist.gif>.

- ⊙ **An error has occurred while establishing a connection to the server. When connecting to SQL Server 2005, this failure may be caused by the fact that under the default settings SQL Server does not allow remote connections. (provider: Named Pipes Provider, error: 40–Could not open a connection to SQL Server)**

שגיאה זו מופיעה כתוצאה מכמה סיבות. למרות שהודעת השגיאה מזכירה את SQL Server 2005 במפורש, עלולים לפגוש בשגיאה זו גם עבור SQL Server 2000. השגיאה נובעת, בדרך כלל, מבעיות שקשורות בשרת בסיס הנתונים. השגיאה עלולה להופיע כתוצאה מאיות שגוי של שם השרת במחרוזת חיבור, כשהשרת אינו פעיל, או כשניתן לגשת לשרת רק מהמחשב המקומי אבל לא מהרשת. כדי לוודא ששרת SQL פועל כהלכה, מלוח הבקרה פתח את Administrative Tools, עבור אל הפריט Services, התבונן בשורה SQL Server ובדוק אם SQL Server הופעל. נספח ב' עוסק בנושא האבטחה של שרת SQL בפירוט רב, ומספק פתרונות לבעיות אלו.

במקרה של בעיה או תקלה שאינך מצליח לפתור, פנה לאתר קהילת Wrox בכתובת <http://p2p.wrox.com>. תמצא שם רבים (כולל אני עצמי) שיבינו במה מדובר ונקווה שיוכלו לעזור לך במציאת פתרון.

כשתפרוס את האתר במחשב שמותקנת בו כבר התוכנה SQL Server 2005 Express Edition – סיימת את תהליך הפריסה. כאשר תנסה להפעיל את האתר עם שרת SQL אחר, עליך לוודא שבסיס הנתונים אכן מכיל את כל הנתונים הנדרשים. בסעיף הבא תלמד כיצד לעשות זאת.

העברת נתונים לשרת מרוחק

העברת האתר שלך מהמחשב המקומי לשרת האינטרנט הוא תהליך ישיר למדי. אתה מעתיק את הנתונים למקום חדש, מגדיר את IIS, משנה כמה הגדרות אבטחה, וזהו. מכיוון שהאתר ממשיך להשתמש ב-SQL Server 2005 Express Edition, הוא יפעל כהלכה.

אבל, כאשר אתה צריך להעביר את האתר לשרת או מחשב מארח אחר, הדברים אינם כה קלים. אמנם, העתקת הקבצים שמרכיבים את האתר היא פשוטה ביותר על ידי שימוש ב-FTP, אבל העתקת נתונים מבסיס נתונים SQL Server 2005 אל המחשב המארח היא לעתים קרובות מורכבת יותר. הסיבה היא שרוב המחשבים המארחים ברשת אינם תומכים ב-SQL Server 2005 Express Edition, ולפיכך אינך יכול להעתיק בקלות את קבצי mdw לתקייה App_Data בשרת המרוחק. שרתים מארחים אלה מציעים לעתים קרובות את הגרסאות המלאות של SQL Server, שניתן לגשת אליהן עם כלי ניהול מבוסס-web או עם כלים כמו SQL Server Management Studio (עבור SQL Server 2005/2008), או Enterprise Manager (עבור SQL Server 2000).

כדי להקל על העברת נתונים מבסיס הנתונים המקומי שהינו SQL Server 2005, אל בסיס נתונים מסוג SQL Server אחר, יצרה מיקרוסופט את אשף הפצת בסיס הנתונים (Database Publishing Wizard).

שימוש באשף הפצת בסיס הנתונים

אשף הפצת בסיס הנתונים מאפשר ליצור תסריט SQL שמכיל את כל המידע הדרוש כדי ליצור מחדש את בסיס הנתונים והנתונים שלו בשרת מרוחק. יצירה מחדש זו של הנתונים היא תהליך דו-שלבי:

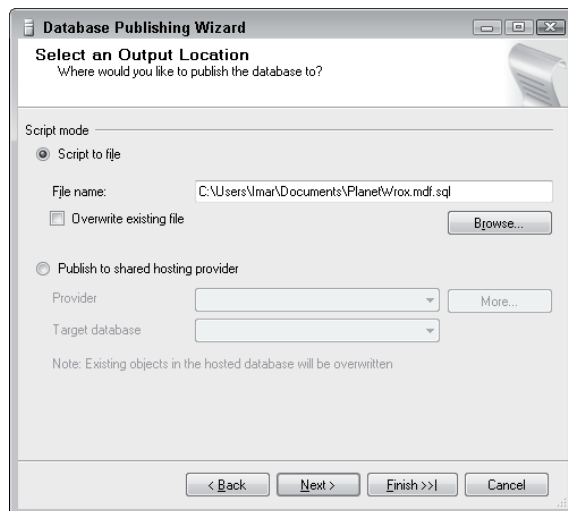
1. צור תסריט SQL מבסיס הנתונים המקומי שלך מסוג SQL Server.
2. שלח תסריט זה למחשבים המארחים ובצע אותו שם.

הצעד הראשון מתואר בפירוט בתרגיל הבא. לא תראה כיצד לבצע את התסריט בשרת המארח, מכיוון שהדבר שונה ממארח אחד לאחר. במקום זאת, תקבל כמה עצות כלליות כדי שתדע מה לחפש במחשב המארח.

נסה זאת יצוא בסיס הנתונים של Planet Wrox

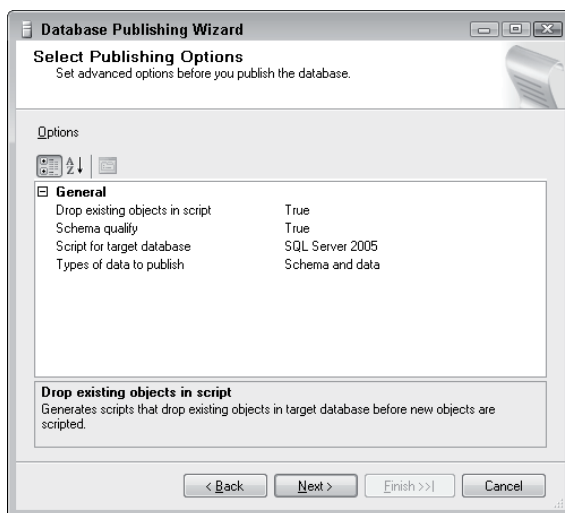
ניתן לגשת לאשף הפצת בסיס הנתונים מחלון סייר בסיס הנתונים (Server Explorer בגרסאות המסחריות של Visual Studio). ברגע שתסיים לבצע את פעולות האשף, יהיה בידך קובץ SQL שמכיל את כל קוד SQL שדרוש ליצירה מחדש של בסיס הנתונים בשרת האחר.

1. כאשר פרויקט Planet Wrox טעון ב-VWD, בחר View ← Database Explorer (או View ← Server Explorer).
2. לחץ לחיצה ימנית על בסיס הנתונים PlanetWrox.mdf ומתפריט הקיצור בחר Publish to Provider לחץ Next. ודא שבסיס הנתונים שלך נבחר, וגם נבחרה האפשרות Script All Objects in the database. לחץ Next. כעת תופיע תיבת הדו-שיח שמוצגת בתרשים 18-20.



תרשים 18-20

3. במסך זה תוכל לבחור בין שתי אפשרויות. הראשונה מאפשרת ליצור קובץ טקסט עם משפטי SQL הדרושים. השנייה מאפשרת להתחבר ישירות לבסיס הנתונים בשרת המארח. אם המחשב המארח תומך בכך, תוכל לקבל את כל המידע הדרוש כדי להגדיר כאן ספק (Provider). בינתיים, בחר Script to File ולחץ על הלחצן Next.
4. בצעד הבא, קבל את כל ברירות המחדל שמוצגות בתרשים 18-21, ולחץ Next פעם נוספת.
5. לחץ Finish. האשף ייצור עבורך את תסריט SQL הדרוש. פתח את הקובץ עם Notepad והתבונן במשפטי SQL שכלולים בו. למרות שרובו יראה לך ככל הנראה כחסר מובן, ניתן להשתמש בו כפי שהוא כדי ליצור מחדש את בסיס הנתונים בבסיס נתונים תואם SQL Server 2005.



תרשים 18-21

כיצד זה פועל

את התוכן של בסיס נתונים ניתן להפריד לשתי קטגוריות: תיאור המבנה של בסיס הנתונים, והנתונים עצמם. כשאשף הפצת בסיס הנתונים מתבצע, הוא חוקר תחילה את מבנה בסיס הנתונים ויוצר משפטי CREATE של SQL עבור כל הפריטים שהוא מוצא בבסיס הנתונים, כמו לדוגמה, הטבלאות שיצרת בפרקים קודמים. אחר כך האשף יוצר משפטי INSERT שיוצרים מחדש את כל הרשומות (כמו סקירות, סגנונות, ואפילו משתמשים) בבסיס הנתונים המיועד. ניקוי תיבת הסימון Script All Objects בתחילת האשף, מאפשר לבחור באופן בררני חלקים מסוימים בלבד מבסיס הנתונים. הדבר מאפשר, לדוגמה, כתיבת תסריט עבור חלק מהטבלאות בלבד.

בסופו של דבר, האשף אוסף את כל משפטי SQL ושומר אותם בקובץ SQL יחיד. קובץ זה מוכן כעת לביצוע במחשב המארח, כדי ליצור מחדש את בסיס הנתונים.

יצירת בסיס הנתונים מחדש

לכל אחד משרתי האירוח יש כללים ונהלים מיוחדים בכל הקשור למתן גישה אל SQL Server שמותקן בהם. למרות זאת ניתן לסווג את המארחים לשלוש קטגוריות.

בקטגוריה הראשונה כלולים מארחים שאינם מאפשרים גישה מרחוק לבסיס הנתונים שלהם, ודורשים ממך לשלוח קובץ SQL שהם יפעילו אותו עבורך. במקרה זה אינך צריך לעשות מאומה, אלא לשלוח את הקובץ ולהמתין שהמארח ייצור עבורך את בסיס הנתונים.

הקטגוריה השנייה כוללת את המארחים שמאפשרים לבצע משפטי SQL על פני ממשק אינטרנט. בדרך כלל תתחבר אל לוח הבקרה המקוון שלך ואז תבצע את משפטי SQL שנוצרו על ידי אשף הפצת בסיס הנתונים, או על ידי העלאת הקובץ, או על ידי הדבקת תוכן הקובץ באזור הטקסט של דף אינטרנט. ללא קשר לשיטה, תסיים עם בסיס נתונים שאליו תוכל לגשת מהיישום שלך. התהליך שונה בכל מארח. כדי לקבל מידע נוסף, פנה אל מערכת ה"עזרה" או אל מערך התמיכה של שירות האירוח. ידוע על כמה בעיות עם כלי ניהול מבוססי-web של בסיסי נתונים של כמה ספקים. כתוצאה מבעיות אלו, יש שגיאות כשמונסים לבצע קובץ SQL שנוצר על ידי האשף. למרות שהקובץ עצמו תקף מבחינה טכנית, משהו עדיין יכול להשתבש. אם זה המקרה, התקשר עם ספק שירות האינטרנט (ISP) שלך לצורך פתרון הבעיה.

הקטגוריה האחרונה כוללת מארחים שמאפשרים להתחבר אל SQL Server שלהם דרך האינטרנט. הדבר מאפשר להשתמש בכלים כמו SQL Server Management Studio כדי להתחבר באופן ישיר משולחן העבודה לבסיס הנתונים במארח, ולבצע תסריטי SQL מרחוק.

ל- SQL Server Management Studio יש גרסת חינם, Express Edition, שתוכל להוריד מאתר מיקרוסופט (<http://msdn.microsoft.com/vstudio/express/sql>). כלי זה פועל בדרך דומה למדי לגרסה המסחרית שכלולה בתשלום SQL Server 2005, וחשיבותה רבה לכל מי שעובד עם בסיסי נתונים מסוג SQL Server.

לאחר יצירת בסיס הנתונים מחדש בשרת היעד, עליך להגדיר מחדש את תצורת היישום מבוסס ASP.NET כדי שישתמש בבסיס הנתונים החדש. הדבר מתבצע על ידי שינוי מחרוזות החיבור באתר האינטרנט. כדי שהדבר יפעל, עליך לשנות שתי מחרוזות חיבור: האחת היא LocalSqlServer1 PlanetWroxConnectionString1 שיצרת בפרק קודם; השנייה היא מחרוזת החיבור LocalSqlServer1 ASP.NET Application Services עבור ASP.NET. כבירת מחל. הדרך הקלה ביותר להשיג זאת היא לנקות את מחרוזות החיבור הישנה, ואז להוסיף חדשה. מחרוזת החיבור שלך צריכה להיראות בהתאם לבסיס הנתונים שבשימוש ובתצורתו. תוכל למצוא דוגמאות רבות של מחרוזות חיבור נכונות באתר www.connectionstrings.com. קטע הקוד הבא מספק דוגמה פשוטה להגדרה מחדש של תצורת היישום כדי שישתמש בשרת בסיס נתונים בשם DatabaseServer. דוגמה זו מחייבת חיבור עם שם משתמש וסיסמה.

```
<connectionStrings>
  <clear />
  <add name="PlanetWroxConnectionString1" connectionString="Data
    Source=DatabaseServer;Initial Catalog=PlanetWrox;User Id=YourUserName;
    Password=YourPassword;" providerName="System.Data.SqlClient"/>
  <add name="LocalSqlServer" connectionString="Data
    Source=DatabaseServer;Initial Catalog=ASPNETDB;User Id=YourUserName;
    Password=YourPassword;" providerName="System.Data.SqlClient"/>
</connectionStrings>
```

קטע קוד זה מציג הן את מחזורות החיבור עבור בסיס הנתונים Planet Wrox (שמכיל את הטבלאות Reviews ו-Albums), והן את זו של בסיס הנתונים ASPNETDB (שמכיל מידע על חברות, תפקידים ופרופילים), אל SQL Server אחר. למידע נוסף על הגדרת יישום מבוסס ASP.NET ו-SQL Server כדי שיפעלו זה עם זה, פנה לנספח ב'. נספח ב' גם מסביר כיצד תוכל לשלב את נתוני בסיס הנתונים ASPNETDB לתוך בסיס הנתונים שלך. יש לכך חשיבות רבה כאשר המארח מחייב תוספת תשלום עבור כל בסיס נתונים נפרד.

בשלב זה סיימת למעשה לעצב את אתר האינטרנט החדש שיצרת. איחולי! עם זאת, לפני שתנוח על זרי הדפנה ותתחיל ליהנות מהאתר החדש, קרא את רשימת התיוג הבאה שתעזור לך לאבטח את האתר ולשפר את ביצועיו.

רשימת תיוג לפריסה

במקום לסיים פרק זה עם עצות פרקטיות כלליות לפריסה, תמצא בסעיף זה רשימה מעשית של דברים שיש לבדוק כשתהיה מוכן להעביר את האתר לייצור.

⊙ ודא שלא הפעלת את הניפוי בקובץ web.config. הדבר גורם לתקורה בלתי-נחוצה ופוגע בביצועי האתר מכיוון שביצוע הקוד איטי יותר והדפדפן אינו יכול להטמין (cache) קבצים חשובים. כדי להבטיח שהניפוי מושבת, פתח את הקובץ web.config שישמש אותך בסביבת הייצור, ובדוק אם debug מוגדר ל-`false`.

```
<compilation debug="false">
```

⊙ ודא שהפעלת את מנגנון השגיאות המותאמות על ידי הגדרת התכונה `mode` של האלמנט `customErrors` ב-`web.config` ל-`On` או ל-`RemoteOnly`. במקרה הראשון, כל אחד יוכל לראות את דפי השגיאה המותאמים אישית; במקרה השני, רק משתמשים מקומיים של שרת האינטרנט יוכלו לראות את פרטי השגיאה. לעולם אל תשאיר את `mode` מוגדרת ל-`Off`, מכיוון שהדבר עלול לגרום לחשיפת מידע. קטע הקוד הבא מציג תצורה בטוחה של האלמנט `customErrors`:

```
<customErrors mode="On" defaultRedirect="~/Errors/Error500.aspx">
```

```
  Optional <error /> elements go here
```

```
</customErrors>
```

- ⊙ השבת את המעקב, או לפחות הגבל את מידע המעקב למשתמשי המחשב המקומי בלבד. האלמנט <trace> הבא שכלול ב-web.config חוסם את המעקב מפני משתמשים שבאים ממחשבים שאינם שרת האינטרנט עצמו. בנוסף, הקוד מונע את הצגת מידע המעקב בדף:

```
<trace mostRecent="true" enabled="true" requestLimit="1000"
      pageOutput="false" localOnly="true"/>
```

- ⊙ שקול אם לקבוע את הגדרת התכונה retail של אלמנט הפריסה ב-machine.config לערך true:

```
<configuration>
  <system.web>
    <deployment retail="true"/>
  </system.web>
</configuration>
```

- ⊙ קוד זה משמש לציון העובדה כי זו הגרסה המוכנה להפצה של אתר האינטרנט. הוא משנה את כל שלושת הפריטים הקודמים להגדרה מאובטחת: הניפוי והמעקב מושבתים, ואילו הודעות השגיאה נגישות למשתמשים מקומיים בלבד.
- ⊙ כדי לבצע שינוי זה עליך להיות מחובר למערכת כמנהלן. בנוסף, צור תחילה עותק גיבוי של הקובץ. מכיוון שקובץ תצורה זה נמצא בשורש כל אתרי האינטרנט מבוססי ASP.NET שלך, לא תרצה לשבש אותו.
- ⊙ חפש באתר קבצים חשובים שעשויים להכיל מידע רגיש (כמו מסמכי Word או קבצי טקסט אשר מכילים הנחיות והתראות), והסר אותם מהגרסה שתשוחרר לייצור, או העבר אותם לתיקייה App_Data. קבצים בתיקייה זו אינם נגישים באופן ישיר. עם זאת, הקוד שלך עדיין יוכל לגשת לקבצים, כפי שראית בפרק 9.
- ⊙ ודא שהופעל רישום שגיאות. קוד רישום השגיאות שיצרת בפרק הקודם מידע אותך כאשר מתרחשת שגיאה, והדבר מאפשר לנקוט פעולות מנע לתיקון שגיאות לפני שיופיעו שוב.
- ⊙ כשמשתמשים באתר בערכות נושא, הסר את המאפיין theme או styleSheetTheme מהאלמנט <pages> שב-web.config. אתר האינטרנט Planet Wrox משתמש בערכות נושא, אבל המאפיין styleSheetTheme התווסף כדי לאפשר תמיכה בזמן-עישוב ב- Visual Web Developer. כל מה שתצטרך בשרת הייצור שלך הוא זה:

```
<pages theme="Monochrome">
  ...
</pages>
```

בדרך זו הדף לא יכלול אותו גיליון סגנון פעמיים.

מה הלאה?

עכשיו, משסיימת את הקמת אתר האינטרנט מבוסס ASP.NET הראשון שלך, אני משוכנע שאתה מצפה ליצירת האתר הבא. האתר Planet Wrox יכול לשמש בסיס לאתרים חדשים שתבנה. קרוב לוודאי שלא תשתמש באתר שלך באף אחד מהדפים שלו, אבל אני מקווה שספר זה ואתר האינטרנט Planet Wrox נתנו לך מספיק השראה לבניית אתר אינטרנט חדש משלך.

מכיוון שספר זה מיועד למתחילים את דרכם בנושא פיתוח אתרים, לא יכולתי לספק מידע מקיף ומעמיק על כמה נושאים חשובים. כמה נושאים שזכו לפרק משלהם בספר זכאים בהחלט לספר שלם משל עצמם.

רשת האינטרנט עצמה היא מקור טוב למידע נוסף. אתרי האינטרנט הבאים עשויים לסייע לך בחיפוש מידע נוסף אודות ASP.NET והטכנולוגיות הקשורות אליה:

- ⊙ **p2p.wrox.com** - פורום דיון ציבורי של Wrox שתוכל לפנות אליו בכל השאלות הנוגעות לתכנות. לספר זה יש מדור נפרד באתר, והדבר מאפשר לשאול שאלות ממוקדות. אני מבקר בפורומים אלה לעתים קרובות, ואעשה כמיטב יכולתי לענות על כל שאלה שתתעורר לגבי הספר.
- ⊙ **imar.spaanjaars.com** - אתר האינטרנט של מחבר הספר, שבו הוא מעדכן אותך בנושאים שונים הקשורים לתכנות ברשת.
- ⊙ **www.asp.net** - אתר קהילת מיקרוסופט עבור טכנולוגיית ASP.NET. פנה לאתר זה עבור חדשות הקשורות ל-ASP.NET, הורדות נוספות ושיעורי הדרכה.
- ⊙ **msdn.microsoft.com/asp.net** - "הבית" הרשמי של ASP.NET באתר המפתחים של מיקרוסופט. תמצא שפע של מידע אודות ASP.NET.

סיכום

פריסת האתר החדש ברשת האינטרנט היא הפעולה המרכזית לאחר סיום מחזור הפיתוח שלו. עם זאת, לא סביר שתפרוס את האתר שלך פעם אחת בלבד. ברגע שתשחרר את הגרסה הראשונה של האתר, כבר תחשוב על תכונות חדשות ומגניבות שתמצא להוסיף לו, על שיפורים שונים ושיפוצים. הדבר הופך את פיתוח האתר לסיפור ללא סוף. השתמש ברשימת התיוג כדי לעשות את האתר קל לפריסה (deploy).

דרך אחת לעשות זאת היא להעביר הגדרות תצורה לקובץ web.config. פעולה זו תגרום לכך ששינוי פרמטרים של האתר בסביבות הפיתוח והייצור ייעשה במקום מרכזי אחד.

כשתהיה מוכן להציג את האתר בפומבי, מומלץ ליצור עותק שלו ולנקותו מקבצי ניסיון שאינם נחוצים לשימוש האתר, לפני שתשלח את הקבצים לשרת היעד. העתקת האתר והפצתו אחר כך הן פעולות קלות לביצוע עם הפקודות Copy Web Site ו-Publish Web Site.

מכיוון שתפרוס את האתר בסביבת שרת IIS, עליך להכיר כמה מהגדרות החשובות של שרת אינטרנט זה. בפרק זה ראית כיצד לעצב את אתר ברירת המחדל, ולבצע כמה שינויי תצורה.

בגלל הדרך שבה פועלת האבטחה ב-Windows וב-IIS, עליך להגדיר גם את הדיסק הקשיח שלך באופן שחשבונות שבשימוש שרת האינטרנט יוכלו לקרוא את הקבצים באתר, ולכתוב לתיקיות מסוימות כמו App_Data ו-GigPics.

הפרק הסתיים בדיון קצר באשף הפצת בסיס הנתונים, אשר מאפשר ליצור עותק של בסיס הנתונים על פי תסריט. עותק זה יכול לשמש לשחזור בסיס הנתונים בשרת מרוחק. בנספח ב' נדון בהרחבה בעיצוב שרתי בסיס נתונים מרוחקים ובהתחברות אליהם.

תרגילים

בפרק זה אין תרגילים, מכיוון שהאתר Planet Wrox מוכן לחלוטין. עם זאת, האתגר הגדול ביותר שלך מתחיל עכשיו: בניית אתרי אינטרנט על בסיס המידע שרכשת בספר זה. אם אי-פעם תבנה אתר עם המידע מספר זה, ותרצה לחלוק אותו אתי, אנא צור עמי קשר דרך האתר הפרטי שלי (imar.spaanjaars.com). בהצלחה והנאה!

נספח א'

תשובות לתרגילים

פרק 1

פתרון תרגיל 1

סימני העריכה של דף ב-VWD מכילים את המקור הגולמי והבלתי-מעובד של הדף, כולל קוד HTML, פקדי שרת של ASP.NET וקוד תוצר תכנות. שרת האינטרנט מעבד את הדף, מבצע את הקוד שבדף, ושולח את קוד HTML הסופי לדפדפן.

פתרון תרגיל 2

XHTML ו-HTML קשורים זה בזה, מכיוון ש-XHTML זו גרסה של HTML שכתובה מחדש עם כללי XML. שפת HTML הרבה יותר נוחה בעבודה, מכיוון ש-XHTML כופה על המתכנת לכתוב קוד שתואם את כללי XML, כמו אותיות רישיות נכונות, תגיות סגורות, ותכונות שמוקפות במרכאות.

פתרון תרגיל 3

הדרך הקלה ביותר לאחסן קטעי HTML שמשתמשים בהם לעתים קרובות, היא לבחור בהם בחלון הקוד ולגרור אותם למקום פנוי בארגז הכלים. כשהפריט מתווסף, תוכל לשנות את שמו ולקבוע לו שם בעל משמעות. אחר כך, בכל פעם שתזדקק לפריט, תוכל ללחוץ עליו לחיצה כפולה או לגרור אותו מארגז הכלים אל הדף שאתה מעצב.

פתרון תרגיל 4

יש כמה דרכים לאפס חלק מהשינויים המותאמים אישית שעשית, ובכלל זה:

- ⊙ איפוס עיצוב חלון התצוגה, על ידי בחירת `Reset Window Layout ← Window`.
- ⊙ איפוס ארגז הכלים, על ידי לחיצה ימנית עליו ובחירת `Reset Toolbox` מתפריט הקיצור.
- ⊙ איפוס כל הגדרות VWD, על ידי התפריט `Import and Export Settings` שתחת התפריט הראשי `Tools`.

פתרון תרגיל 5

כדי לשנות מאפיין של פקד בדף, לחץ על הפקד ב- Design View או ב- Markup View. כעת השתמש בטבלת המאפיינים (שתוכל להציג על ידי הקשת F4) כדי לשנות את ערך המאפיין. לחילופין, תוכל לשנות את המאפיין ישירות בחלון המסמך.

פרק 2

פתרון תרגיל 1

לקטגוריה של קבצי אינטרנט שייכים כמה סוגי קבצים, כולל קבצי aspx (טפסי אינטרנט שהופכים לדפים בדפדפן), קבצי HTML (שמכילים HTML סטטי עבור האתר), קבצי CSS (שמכילים מידע CSS), וקבצי config שכוללים מידע על תצורת האתר. הרשימה המלאה של הקבצים השייכים לקטגוריה זו נמצאת בטבלה שבפרק 2.

פתרון תרגיל 2

כשברצונך לעצב קטע טקסט עם גופן מודגש (bold) ואותיות נטויות (italics), עליך לסמן את הטקסט וללחוץ על הלחצן Bold בסרגל הכלים Formatting. אחר כך עליך ללחוץ על הלחצן Italic. קוד HTML הסופי בדף אמור להיראות כך:

```
<b><i>Welcome to Planet Wrox</i></b>
```

פתרון תרגיל 3

הדרך הראשונה היא להשתמש בסייר הפתרון. לחץ לחיצה ימנית על הפרויקט. מתפריט הקיצור שמופיע, בחר Add Existing Item ודפדף עד שתגיע לפריט או פריטים שברצונך להוסיף.

כחלופה לכך, ביכולתך לגרור קבצים מסייר החלונות או משולחן העבודה ולשחררם ישירות בתוך פרויקט של VWD.

כחלופה נוספת, תוכל לאחסן את הקבצים ישירות בתיקיית הפרויקט ולהשתמש בסייר החלונות. לדוגמה, קבצים שתוסיף לתיקייה C:\BegASPNET\Site יהפכו באופן אוטומטי לחלק מהאתר. אם הקבצים החדשים אינם מופיעים ישירות ב-VWD, לחץ על הסמל Refresh בסרגל הכלים של סייר הפתרון.

פתרון תרגיל 4

VWD מספקת שלוש תצוגות שונות של הקוד: Design View, Markup View (ידועה גם כ- Source View או Code View) ו-Split View. Design View נותן מושג כיצד דף האינטרנט ייראה בדפדפן, בעוד ש-Markup View מציג את סימני העריכה הגולמיים. Split View מאפשר לראות את שני הקודמים בו-זמנית.

עבור קבצים אחרים יש ל-VWD תצוגות שונות. לדוגמה, בדרך כלל מתייחסים לקוד שנכתב עבור דפי ASPX כאל תצוגת קוד ברקע או בפשטות הקוד ברקע.

פרק 3

פתרון תרגיל 1

התועלת הגדולה ביותר של גיליון סגנון חיצוני היא העובדה שניתן להחילו על האתר כולו. שינוי כלל אחד בקובץ זה יכול להשפיע על כל הדפים באתר שמשתמשים בכלל זה. עם סגנונות פנימיים, עליך לשנות באופן ידני את כל הקבצים באתר. גיליונות סגנון חיצוניים גם מקלים מאוד על שימוש חוזר בסגנון מסוים באלמנטים שונים רבים באתר. צור מחלקה או בורר ID והשתמש בהם היכן שתמצא לנכון.

פתרון תרגיל 2

הכלל יכול להיראות כך:

```
h1
{
  font-family: Arial;
  color: Blue;
  font-size: 18px;
  border-top: 1px solid blue;
  border-left: 1px solid blue;
}
```

שים לב לגרסת קצרנות נוספת של מאפיין הגבול (border property). גרסה זו נראית דומה למאפיין הגבול שכבר ראית בפרק זה, שאיפשר לקבוע את הגודל, הסגנון, והצבע של הגבול בפעולה אחת. משתמשים בכלל זה ב- border-top ו- border-left כדי לשנות את הגבולות השמאלי והעליון בלבד; שני הכיוונים האחרים, הגבול הימני והגבול התחתון, אינם מושפעים על ידי כלל זה.

פתרון תרגיל 3

ההכרזה השנייה עולה על הראשונה מבחינת האפשרות לשימוש חוזר באתר, מכיוון שהיא מציינת Class selector בניגוד לבורר ID. בבורר ID יכול להשתמש פעם אחת בלבד אלמנט בעל תכונה id תואמת, MainContent בדוגמה זו, אבל פעם אחת בלבד בדף יחיד. מצד שני, ב- BoxWithBorders, שהוא Class selector, יכולים להשתמש אלמנטים רבים בדף יחיד, מכיוון שאתה רשאי להחיל מאפיין class זהה על אלמנטים רבים בדף.

פתרון תרגיל 4

VWD מאפשרת לצרף גיליון סגנון בדרכים הבאות:

- ⊙ ב- Markup View, הקלד את האלמנט <link> הדרוש באופן ישיר ב- <head> של הדף.
- ⊙ ב- Markup View, גרור קובץ CSS מסייר הפתרון לתוך הקטע <head> של הדף.
- ⊙ ב- Design View, גרור קובץ CSS מסייר הפתרון ושחרר אותו בדף.
- ⊙ מהתפריט הראשי, בחר Attach Style Sheet ← Format ואז דפדף לקובץ CSS.
- ⊙ לחץ על הקישור Attach Style Sheet בחלונות Manage Styles או Apply Styles.

פרק 4

פתרון תרגיל 1

המנגנון שמאפשר לפקדים לשמר את מצבם נקרא ViewState.

פתרון תרגיל 2

ערכי הפקדים מאוחסנים על ידי ASP.NET runtime בשם _VIEWSTATE. שדה מוסתר זה נשלח לשרת בכל פעולת postback, ואז מתבצעת פעולת פירוק הערכים והשמתם בפקדים.

פתרון תרגיל 3

הפקד DropDownList מאפשר למשתמש לעשות בחירה אחת בלבד, בעוד שהפקד ListBox מאפשר בחירה מרובה. בנוסף, כאשר DropDownList אינו מורחב, הוא מציג פריט אחד בלבד מהרשימה, בעוד ש-ListBox מסוגל להציג מספר רב של פריטים בו-זמנית.

פתרון תרגיל 4

כדי לגרום לפקד CheckBox לבצע קריאה לשרת כשאתה בוחר או מנקה אותו בדפדפן, עליך להגדיר את המאפיין AutoPostBack ל-True.

```
<asp:CheckBox ID="CheckBox1" runat="server" AutoPostBack="True" />
```

פתרון תרגיל 5

רבים מפקדי השרת של ASP.NET מאפשרים לשנות צבעים על ידי שימוש במאפיינים כמו BackColor ו-ForeColor. בנוסף, תוכל להשתמש ב-BorderColor, BorderStyle ו-BorderWidth כדי לשנות את מאפייני הגבול סביב פקד בדפדפן. לבסוף, כדי להשפיע על גודל הפקד, עליך לקבוע את המאפיינים Height ו-Width שלו.

פתרון תרגיל 6

במקום להגדיר מאפייני סגנון פרטיים כפי שמפורט בתרגיל הקודם, עדיף להגדיר מאפיין CssClass שמצביע על גיליון סגנון. בדרך זו קל יותר לנהל את הדפים, מכיוון שהמידע הקשור לסגנון מאוחסן במקום אחד - בגיליון הסגנון. בנוסף, טעינת הדף תהיה מהירה יותר מכיוון שלא כל מידע הסגנון נשלח עבור כל פקד בכל בקשה. במקום זאת, הדפדפן קורא את גיליון הסגנון פעם אחת בלבד ושומר עותק שלו במטמון המקומי.

פרק 5

פתרון תרגיל 1

שני סוגי הנתונים, Byte ו-SByte, מתוכננים להחזיק ערכים מספריים קטנים. שניהם תופסים בדיוק אותה כמות של זיכרון במחשב, לכן עדיף להשתמש בסוג הנתונים Byte. הסוג Byte אינו

מאפשר לאחסן מספרים שליליים, אלא ערכים שבין 0 ל-255 בלבד. עם זאת, מומלץ לאחסן את תאריך הלידה ולא את גילו של אדם. על ידי השוואת תאריך הלידה לתאריך הנוכחי, תוכל תמיד לחלץ את הגיל מתאריך הלידה, כפי שמדגימה הדוגמה הבאה:

VB.NET

```
Dim albertsBirthday As DateTime = New DateTime(1879, 3, 14)
Dim age As Integer = DateTime.Now.Year - albertsBirthday.Year ' מציג 130 בשנת 2009
```

C#

```
DateTime albertsBirthday = new DateTime(1879, 3, 14);
int age = DateTime.Now.Year - albertsBirthday.Year; // מציג 130 בשנת 2009
```

פתרון תרגיל 2

קוד זה נראה מורכב מעט מכיוון שהוא משתמש בביטוי ובהצבה בשורת קוד דחוסה אחת. בגלל סדר הביצוע, מחושב תחילה הביטוי ואחר כך נעשית ההצבה. בקוד המקור משמעות הדבר שבתחילה מחושב ערך הביטוי "ShoppingCart.Items.Count > 0". ביטוי זה בודק אם המאפיין Count של האוסף Items בסל הקניות גדול מאפס. כפי שלמדנו בפרק זה, תוצאת החישוב היא ביטוי בוליאני, בעל הערך True או False. ערך בוליאני זה מוקצה למאפיין Visible של הפקד Button, והדבר גורם ללחצן להיות נראה או נסתר. כשמרחיבים מעט את הקוד, קל יותר להבין את המתרחש:

VB.NET

```
Dim canShowButton As Boolean = False
If ShoppingCart.Items.Count > 0 Then
    canShowButton = True
End If
DeleteButton.Visible = canShowButton
```

C#

```
bool canShowButton = false;
if (ShoppingCart.Items.Count > 0)
{
    canShowButton = true;
}
DeleteButton.Visible = canShowButton;
```

פתרון תרגיל 3

כדי ליצור גרסה מיוחדת של Person עליך ליצור מחלקה נוספת שיורשת מהמחלקה Person ומרחיבה את התנהגותה על ידי הוספת המאפיין PhoneNumber:

VB.NET

```
Public Class PersonWithPhoneNumber
    Inherits Person
    Private _phoneNumber As String
```

```
Public Property PhoneNumber() As String
    Get
        Return _phoneNumber
    End Get
    Set(ByVal value As String)
        _phoneNumber = value
    End Set
End Property
End Class
```

C#

```
public class PersonWithPhoneNumber : Person
{
    public string PhoneNumber { get; set; }
}
```

פרק 6

פתרון תרגיל 1

האלמנט `ContentPlaceHolder` חייב להיות מוצב בדף הראשי. הוא מכיל אזור שניתן למילוי על ידי דפי תוכן. הפקד `Content` חייב להיות מוצב בדף תוכן שמבוסס על הדף הראשי. הוא משמש לאספקת התוכן לאלמנט `ContentPlaceHolder` שהוא מחובר אליו.

פתרון תרגיל 2

כדי לקשר את הפקד `Content` אל `ContentPlaceHolder` בדף הראשי, עליך להגדיר את `:ContentPlaceHolderID`

```
<asp:Content ID="Content1" ContentPlaceHolderID="IdOfContentPlaceHolder" Runat="Server">
</asp:Content>
```

פתרון תרגיל 3

יש כמה דרכים לעשות זאת. ראשית, באותו קובץ `skin` אפשר ליצור `named skin` בעל מחלקת `CSS` שונה. אחר כך תחבר את `named skin` הזה (הכוונה ל-`skin` עם שם) אל הפקד שברצונך לשנות.

```
<asp:Button runat="server" SkinID="RedButton" CssClass="RedButton" />
```

הפקד `Button` בדף `ASPX` חייב להגדיר אז את `SkinID` ל-`RedButton`:

```
<asp:Button ID="Button1" runat="server" Text="Button" SkinID="RedButton" />
```

לחילופין, ביכולתך להשבית את ערכת הנושא בפקד `Button`, ולהקצות לו מחלקת `CSS` שונה ישירות בדף `ASPX`.

```
<asp:Button ID="Button1" runat="server" EnableTheming="False" CssClass="RedButton"
    Text="Button" />
```



```
.RedButton
{
    background-color: Red;
}
```

פתרון תרגיל 4

מוקדם במחזור החיים של דף, מחילים עליו StyleSheetTheme. הדבר נותן לפקדים בדף ASPX את ההזדמנות לדרוס את ההגדרות הראשוניות שקיבלו מ-StyleSheetTheme. פירוש הדבר ש-StyleSheetTheme רק מציג את המראה והתחושה של פקדים, אבל משאיר בידי הפקדים היחידים את היכולת לדרוס מראה זה. לעומת זאת, ערכת הנושא דורסת את כל ההגדרות שמוחלות על ידי הפקדים. הדבר מאפשר למתכנת הדף לכפות את המראה והתחושה על פקדים באתר. אם עדיין מתעורר הצורך לשנות פקדים יחידים, תוכל לנטרל את ערכת הנושא על ידי הגדרת מאפיין EnableTheming לערך False.

פתרון תרגיל 5

יש שלוש דרכים להגדיר את המאפיין Theme באתר אינטרנט שמבוסס ASP.NET 3.5. האפשרות הראשונה היא להגדיר את המאפיין ישירות בהנחיה @ Page באופן שיחול על דף זה בלבד:

```
<%@ Page Language="C#" ..... Theme="Monochrome" %>
```

כדי שערכת נושא תחול על כל הדפים באתר, יש להגדיר את התכונה theme של האלמנט <pages> בקובץ web.config:

```
<pages theme="Monochrome">
```

האפשרות האחרונה היא להגדיר את ערכת הנושא בתכנות. עליך לעשות זאת באירוע PreInit של המחלקה Page שבו תוכל לטפל בדפים יחידים באתר, או במקום מרכזי על ידי שימוש ב-BasePage כפי שעשית בפרק זה.

פתרון תרגיל 6

דף בסיס (base page) מאפשר למרכז את התנהגות כל הדפים באתר. במקום לקודד את אותה פונקציונליות פעם אחר פעם בכל דף, מעבירים קוד זה אל דף הבסיס וכל דפי ASPX יכולים להשתמש בו. כשתיישם את ממתג ערכות הנושא, כל שתצטרך לעשות הוא לכתוב מעט קוד במחלקה המרכזית BasePage. כל הדפים באתר שיורשים מהמחלקה BasePage יקבעו כעת בצורה נכונה את ערכת הנושא שנבחרה, ללא צורך בתוספת קוד כלשהי.

פרק 7

פתרון תרגיל 1

שתי קבוצות עיקריות של מאפייני סגנון זמינות לצורך שינוי הפריטים של הפקד Menu. כל מאפייני הקבוצה הראשונה מתחילים ב-Static ומתייחסים לפריטי התפריט הראשי שמוצגים כשהדף נטען בפעם הראשונה. ה-StaticMenuItemStyle חל על התפריט כולו, בעוד StaticMenuItemStyle, StaticMenuItemStyle

StaticHoverStyle ו-StaticSelectedStyle משמשים לשינוי המראה של פריטי התפריט הממשיים במצבים שונים (מצב רגיל, כשהעכבר מרחף עליהם, וכשהם הפריט הפעיל של התפריט). אותם מאפייני סגנון זמינים גם עבור תת-הפריטים, אבל כולם מתחילים ב-Dynamic ולא ב-Static.

פתרון תרגיל 2

כדי לכוון משתמש מחדש אל דף אחר על ידי תכנות, תוכל להשתמש ב-Response.Redirect וב-Server.Transfer. האפשרות הראשונה שולחת הוראת הכוונה מחדש לדפדפן ולפיכך נחשבת להכוונה מחדש של צד-לקוח. לעומת זאת, Server.Transfer מתרחשת בשרת ומאפשרת להציג דף אחר מבלי להשפיע על סרגל הכתובת של המשתמש.

פתרון תרגיל 3

כדי להציג TreeView שאין לו יכולת להרחיב או לצמצם צמתים, עליך להגדיר את המאפיין ShowExpandCollapse של TreeView ל-False.

פרק 8

פתרון תרגיל 1

מאפיין רגיל משתמש במשתנה גיבוי רגיל כדי לאחסן את ערכו, בעוד המשתנה ViewState משתמש לשם כך באוסף ViewState. מאפיין רגיל מאופס בכל postback, בעוד מאפיין ViewState מסוגל לשמר את ערכו. עם זאת, ליתרון זה של המאפיין ViewState יש מחיר. אחסון הערך ב-ViewState מוסיף לגודל הדף, הן בשעת הבקשה והן בשעת התגובה. מאפיין רגיל אינו סובל מחיסרון זה. כדי להקטין את גודל הדף עליך לשקול בזהירות מה לאחסן ב-ViewState.

פתרון תרגיל 2

כדי לגרום למאפיין לשמר את מצבו על פני postbacks, עליך להפכו למאפיין ViewState. הקוד הנדרש זהה כמעט לזה של NavigateUrl, אבל הוא משתמש בסוג הנתונים Direction ולא במחרוזת. בדוגמת VB.NET, שנה את החלקים המוצללים של Get ו-Set של המאפיין. בדוגמת C#, הסר את המאפיין האוטומטי (ראה פרק 5) והחלף אותו בקוד הבא.

VB.NET

```
Public Property DisplayDirection() As Direction
```

```
Get
```

```
Dim _displayDirection As Object = ViewState("DisplayDirection")
```

```
If _displayDirection IsNot Nothing Then
```

```
Return CType(_displayDirection, Direction)
```

```
Else
```

```
Return Direction.Vertical ' Not found in ViewState; return a default value
```

```
End If
```

```
End Get
```

```
Set(ByVal Value As Direction)
```

```
ViewState("DisplayDirection") = Value
```

```
End Set
```

```
End Property
```

C#

```
public Direction DisplayDirection
{
    get
    {
        object _displayDirection = ViewState["DisplayDirection"];
        if (_displayDirection != null)
        {
            return (Direction)_displayDirection;
        }
        else
        {
            return Direction.Vertical; // Not found in ViewState; return a default value
        }
    }
    set
    {
        ViewState["DisplayDirection"] = value;
    }
}
```

עם מאפיין ViewState זה אינך זקוק עוד לשדה הפרטי `_displayDirection` שהוכרז תחילה מחוץ למאפיין בפקד המשתמש `Banner`.

פתרון תרגיל 3

לשימוש בסוג נתונים מותאם אישית כמו סוג הספירה `Direction`, יש שתי תועלות לעומת השימוש בסוגי הנתונים נומרי או מחרוזת. בגלל הדרך שבה `IntelliSense` עוזרת לבחור בפריט הנכון, אינך צריך לזכור מספרי פלא או מחרוזות כמו 0 או 1. בנוסף, המהדר עוזר לבדוק את האיות. לדוגמה, אם תכתוב `Direction.Vrtical` במקום `Direction.Vertical` תקבל הודעת שגיאה בזמן הפיתוח.

פרק 9

פתרון תרגיל 1

עליך לכתוב תחילה מאפיין בקוד ברקע של פקד המשתמש, בדומה למאפיין `DisplayDirection` שיצרת בפרק הקודם עבור הפקד `Banner`. מאפיין זה יכול להיראות כך:

VB.NET

```
Private _pageDescription As String
Public Property PageDescription() As String
    Get
        Return _pageDescription
    End Get
    Set(ByVal Value As String)
        _pageDescription = Value
    End Set
End Property
```

C#

```
public string PageDescription { get; set; }
```

אחר כך שנה את הכרזת הפקד. לדוגמה, ב-Contact.aspx תוכל לשנות את הפקד כך:

```
<uc1:ContactForm ID="ContactForm1" runat="server" PageDescription="Contact Page"/></p>
```

שים לב שהמאפיין PageDescription מכיל תיאור קצר של הדף המכיל. מובן מאליו שביכולתך להשתמש באיזה טקסט שתמצא לנכון כדי לתאר את הדף במאפיין.

לבסוף, תצטרך להוסיף את PageDescription לנושא או לגוף הודעת הדואר האלקטרוני. בקטע הקוד הבא תראה כיצד להרחיב את הנושא עם הערך של מאפיין חדש זה:

VB.NET

```
myMessage.Subject = "Response from web site. Page " & PageDescription  
myMessage.From = New MailAddress("you@yourprovider.com", "Sender Name Here")
```

C#

```
myMessage.Subject = "Response from web site. Page " + PageDescription;  
myMessage.From = new MailAddress("you@yourprovider.com", "Sender Name Here");
```

מעתה ואילך יתווסף התיאור המותאם אישית הזה של הדף לנושא הודעת הדואר.

פתרון תרגיל 2

אם אינך בוחן את המאפיין IsValid, המערכת חשופה לפגיעת נתונים לא תקפים. משתמשים יכולים לנטרל שימוש ב-JavaScript בדפדפן שלהם ולשלוח נתונים לא תקפים אל הדף שלך. על ידי בדיקת IsValid תוכל לקבוע אם בטוח להמשיך עם הנתונים שנשלחו או לא.

פתרון תרגיל 3

המאפיין From של המחלקה MailMessage הוא מהסוג MailAddress. משמעות הדבר שבאפשרותך להקצות לו באופן ישיר מופע יחיד של מחלקה זו. מכיוון שיש אפשרות שיהיה יותר מנמען אחד, הרי שהמאפיין To הוא אוסף (collection) של אובייקטים MailAddress, ולכן עליך להשתמש בשיטה Add כדי להוסיף מופעים של MailAddress למאפיין To.

פתרון תרגיל 4

כדי לקרוא לפונקציית בדיקת התוקף או האימות (validation) של צד-לקוח, עליך להגדיר את המאפיין ClientValidationFunction של CustomValidator כך:

```
<asp:CustomValidator ID="CustomValidator1" runat="server"  
ClientValidationFunction="ValidatePhoneNumbers" ></asp:CustomValidator>
```

פונקציית הלקוח שעליך להוסיף לסימני העריכה של הדף חייבת להיות בעלת החתימה הבאה:

```
function FunctionName(source, args)  
{}
```

הארגומנט source מכיל פנייה לפקד CustomValidator הממשיך בקוד HTML של צד-לקוח. הארגומנט args מספק מידע תוכן על הנתונים ומאפשר לציין אם הנתונים תקפים או לא. שמות

הארגומנטים אינם חייבים להיות source ו-args; עם זאת, כשמשמשים בשמות אלה, הדמיון בין פונקציית צד-לקוח למקבילה לה בצד-שרת יהיה רב ככל האפשר. סכמת קביעת שמות נפוצה אחרת, שמשמשים בה עבור רבים ממטפלי האירוע האחרים של ASP.NET, היא להשתמש ב-sender ו-1 e בהתאמה.

פתרון תרגיל 5

המאפיין IsValid של הארגומנט args בשיטת בדיקת התוקף המותאמת אישית שלך, הוא זה שאומר למנגנון האימות אם הנתונים שבדקת תקפים או לא. הדבר חל הן על קוד צד-לקוח והן על קוד צד-שרת. בקטע הקוד הבא תראה כיצד זה מתבצע בשיטת האימות של צד-לקוח עבור הפקד ContactForm.ascx:

```
if (txtPhoneHome.value != '' || txtPhoneBusiness.value != '')
{
    args.IsValid = true;
}
else
{
    args.IsValid = false;
}
```

פרק 10

פתרון תרגיל 1

הפקד ScriptManager הוא רכיב אשר דרוש כמעט בכל הפעולות הקשורות ל-Ajax. הוא דואג לרישום קבצי JavaScript של צד-לקוח, מטפל באינטראקציה עם שירותי האינטרנט המוגדרים באתר, ואחראי לעדכון החלקי של דפים. כשאתה סבור שתזדקק ליכולות Ajax רק בדפים ספורים, הרי שבדרך כלל תציב את ScriptManager ישירות בדף תוכן. עם זאת, ביכולתך להציב את ScriptManager גם בדף ראשי כדי שיהיה זמין בכל רחבי האתר.

כאשר ScriptManager נמצא בדף ראשי, תוכל להשתמש ב-ScriptManagerProxy כדי לרשום שירותי אינטרנט יחידים בדף תוכן. מכיוון שבדף יכול להיות רק ScriptManager אחד, לא תוכל להוסיף עוד אחד בדף תוכן שמשמש בדף הראשי שמכיל את ScriptManager. ה-ScriptManagerProxy משמש כגשר בין דף התוכן לבין ScriptManager, והדבר נותן גמישות רבה באשר למקום שבו יירשמו השירותים.

פתרון תרגיל 2

הוספת פקד UpdateProgress תאפשר להודיע למשתמשים שמתקיים עדכון חלקי של דף. תחבר פקד זה אל אל UpdatePanel על ידי AssociatedUpdatePanelID שלו. בתוך <ProgressTemplate> תגדיר את סימני העריכה שתמצא לנכון, כדי להודיע למשתמשים על העדכון שמתבצע. <ProgressTemplate> אופייני מכיל סמל הנפשה, טקסט כלשהו, או את שניהם.

פתרון תרגיל 3

כדי ליצור שירות אינטרנט שניתן לקרוא לו מתסריט, עליך להוסיף קודם לאתר קובץ Web Service על ידי תיבת הדו-שיח Add New Item. ה- Web Service שתיצור יירש מ- System.Web.Services.WebService, המחלקה שמגדירה את התנהגות ברירת המחדל של כל שירותי האינטרנט. כשאתה יוצר Web Service, עליך להוסיף לו את המאפיין ScriptService:

VB.NET

```
<System.Web.Script.Services.ScriptService()> _  
<WebService(Namespace:="http://tempuri.org/")> _  
<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _  
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _  
Public Class NameService  
    Inherits System.Web.Services.WebService
```

C#

```
[System.Web.Script.Services.ScriptService]  
public class NameService : System.Web.Services.WebService  
  
    לבסוף, לכל שיטה ממחלקה זו שברצונך לחשוף כשיטת web, עליך להוסיף את התכונה  
    :WebMethod
```

VB.NET

```
<WebMethod()> _  
Public Function HelloWorld(ByVal yourName As String) As String
```

C#

```
[WebMethod]  
public string HelloWorld(string yourName)
```

פרק 11

פתרון תרגיל 1

המשפט DELETE נכשל מכיוון שיש קשר בין Id של הטבלה Genre לבין GenreId של הטבלה Review. כל זמן שקשר זה בתוקף, אינך יכול למחוק סגנונות שעדיין מצורפות אליהם סקירות. כדי שתוכל למחוק את הסגנונות המבוקשים, עליך למחוק תחילה את הסקירות המצורפות להם, או להקצות אותן לסגנון אחר על ידי המשפט UPDATE.

פתרון תרגיל 2

הקשר בין הטבלאות Genre ו-Review הוא קשר חד-סטרי. הקשר כופה ש-GenreId שמוקצה לסקירה יתקיים כ-Id בטבלה Genre. בו-בזמן הוא מונע מחיקת סגנונות שעדיין מצורפות להם סקירות. עם זאת, הקשר אינו מונע ממך למחוק רשומות מהטבלה Review.

פתרון תרגיל 3

כדי למחוק סקירות בעלות Id שערכו 100 או פחות, יש צורך במשפט SQL הבא:

```
DELETE FROM Review WHERE Id <= 100
```

פתרון תרגיל 4

לפני שתוכל למחוק את הסגנון, תצטרך להקצות את הסקירות הקיימות לסגנון אחר. תוכל לעשות זאת עם משפט UPDATE הבא:

```
UPDATE Review SET GenreId = 11 WHERE GenreId = 4
```

משפט זה מקצה GenreId שערכו 11 לכל הסקירות שקודם לכן היו בעלות GenreId של 4. כתוצאה מכך אין עוד סקירות מצורפות לסגנון בעל ID שערכו 4, וניתן להסירו בעזרת משפט SQL הבא:

```
DELETE FROM Genre WHERE Id = 4
```

פרק 12

פתרון תרגיל 1

הפקד הטוב ביותר עבור תרחיש זה הוא GridView. קל להגדירו ויש לו תמיכה מובנית בדפדוף, עדכון ומחיקת נתונים. בשיתוף עם הפקד DetailsView תוכל להציע למשתמשים את כל ארבע פעולות CRUD.

פתרון תרגיל 2

עבור רשימה לא-סדורה פשוטה עדיף להשתמש בפקד Repeater. התועלת הגדולה ביותר מהפקד Repeater היא שהוא אינו פולט קוד HTML משל עצמו, ומאפשר לשלוט בסימני העריכה הסופיים. החיסרון של הפקד בכך שאינו תומך בעריכה או מחיקה של נתונים. הדבר אינו מהווה בעיה אם כל מה שעליך לעשות הוא רק להציג את הנתונים שברשימה. בפרק 13 יש הסבר כיצד להשתמש בפקד Repeater.

פתרון תרגיל 3

BoundField קשור ישירות לעמודה במקור הנתונים, ומציע מספר מוגבל של דרכים להתאמה אישית של המראה שלו. עם זאת, TemplateField נותן שליטה מלאה בדרך שבה השדה מרונדר. כשלעצמו, זהו שדה אידיאלי לתרחישים מורכבים יותר – כמו לדוגמה, כשברצונך להוסיף לדף פקדי אימות, או כשברצונך לאפשר למשתמש לפעול עם פקד אחר, למשל DropDownList ולא בפקד ברירת המחדל TextBox.

פתרון תרגיל 4

תמיד אחסן את מחרוזות החיבור בקובץ web.config. לקובץ זה יש אלמנט <connectionStrings> שתוכנן במיוחד לאחסון מחרוזות חיבור. האחסון ב-web.config מקל מאוד על מציאת מחרוזות

החיבור ועל הכנסת שינויים בהן. אם תאחסן אותן ברמת הדרג, תצטרך לחפש בכל הפרויקט את מחרוזות החיבור הרלוונטיות.

ניתן לגשת למחרוזות החיבור על ידי תחביר קשירת ביטוי (expression binding syntax). לדוגמה, כדי להגדיר את מחרוזות החיבור ב-`SqlDataSource` תוכל להשתמש בקוד כמו זה:

```
ConnectionString="<%$ ConnectionStrings:PlanetWroxConnectionString1 %>"
```

פרק 13

פתרון תרגיל 1

כדי לקבל מהמערכת את 10 הסקירות הטריות ביותר, זקוקה השאילתה לשני מבנים חשובים של LINQ. תחילה היא זקוקה לפסקת `Order By` (ב-`C#`) כדי לסדר את הרשימה בסדר יורד. אחר כך השאילתה צריכה להשתמש בשיטה `Take` כדי לקחת מקבוצת התוצאה את 10 הסקירות הראשונות.

VB.NET

```
Dim recentReviews = (From myReview In myDataContext.Reviews _  
    Order By myReview.CreateDateTime Descending _  
    Select New With {myReview.Title, myReview.Genre.Name}).Take(10)
```

```
GridView1.DataSource = recentReviews  
GridView1.DataBind()
```

C#

```
var recentReviews = (from myReview in myDataContext.Reviews  
    orderby myReview.CreateDateTime descending  
    select new  
    {  
        myReview.Title,  
        myReview.Genre.Name  
    }).Take(10);
```

```
GridView1.DataSource = recentReviews;  
GridView1.DataBind();
```

הקוד משתמש גם במילת המפתח `New` (ב-`C#`) כדי ליצור סוג אנונימי חדש, שמכיל רק את כותרות הסקירות ושמות הסגנונות.

פתרון תרגיל 2

התועלת הגדולה ביותר מהפקד `ListView` היא בכך שהוא משלב את העוצמות של פקדי הנתונים האחרים. בדיוק כמו הפקד `GridView`, גם הפקד `ListView` מקל על הצגת נתונים בטבלה שמשתמשים יכולים לערוך. בנוסף, הפקד `ListView` מאפשר להכניס רשומות חדשות, התנהגות שקיימת בפקדים כמו `DetailsView` ו-`FormView`, אבל לא ב-`GridView`.

לבסוף, הפקד `ListView` מאפשר שליטה מלאה על סימני העריכה שנשלחים לדפדפן. תכונה חשובה זו קיימת מהמוכן רק בפקד `Repeater`.

פתרון תרגיל 3

תחילה תצטרך לשנות את הדף Default.aspx באופן שיקשר כל תמונה ממוזערת לדף פירוט ויעביר את ה-ID של התמונה לדף חדש זה. ב-`<ItemTemplate>` של הפקד `ListView` ב-Default.aspx הוסף את הפקד `HyperLink` הבא סביב הפקד `Image` שכבר נמצא שם:

```
<asp:HyperLink ID="HyperLink1" runat="server"
    NavigateUrl='<%# "PictureDetails.aspx?Id=" + Eval("Id").ToString() %>'>
<asp:Image ID="Image1" runat="server" ImageUrl='<%# Eval("ImageUrl") %>'
    ToolTip='<%# Eval("ToolTip") %>' />
</asp:HyperLink>
```

שים לב ש-`NavigateUrl` נבנה מהטקסט הסטטי `PictureDetail.aspx?Id=` ומה-ID של התמונה בבסיס הנתונים. כעת צור דף חדש בשם `PictureDetails.aspx` והוסף פקד `Image` בסימני העריכה:

```
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" Runat="Server">
    <asp:Image ID="Image1" runat="server" />
</asp:Content>
```

לבסוף, כדי להגדיר את `ImageUrl`, עליך לבצע את שאילתת LINQ הבאה באירוע `Load` של הדף שנמצא בקוד ברקע:

VB.NET

```
Protected Sub Image1_Load(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Image1.Load
    Dim pictureId As Integer = Convert.ToInt32(Request.QueryString.Get("Id"))

    Using myDataContext As New PlanetWroxDataContext()
        Dim imageUrl As String = (From picture In myDataContext.Pictures _
                                   Where picture.Id = pictureId _
                                   Select picture).Single().ImageUrl
        Image1.ImageUrl = imageUrl
    End Using
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
{
    int pictureId = Convert.ToInt32(Request.QueryString.Get("Id"));
    using (PlanetWroxDataContext myDataContext = new PlanetWroxDataContext())
    {
        string imageUrl = (from picture in myDataContext.Pictures
                            where picture.Id == pictureId
                            select picture).Single().ImageUrl;
        Image1.ImageUrl = imageUrl;
    }
}
```

קטע קוד זה מקבל תחילה את ה-ID של התמונה ממחרוזת השאילתה, ואז מזין אותו לשאילתת LINQ. השיטה `Single` משמשת לאחזור תמונה יחידה מהטבלה `Picture` אשר `ImageUrl` שלה משמש להצגת התמונה בדפדפן.

פרק 14

פתרון תרגיל 1

האירוע Load של Page מתעורר תמיד לפני אירועים שמעורר המשתמש, כמו למשל האירוע Click של הפקד Button.

פתרון תרגיל 2

כדי שהשורות הזוגיות והאי-זוגיות תופענה לסירוגין, יש להגדיר RowStyle ו-AlternatingRowStyle. אחר כך תצטרך להגדיר את המאפיינים BackColor ו-ForeColor שלהם בהתאמה:

```
<asp:GridView ID="GridView1" runat="server" ... >
  <AlternatingRowStyle BackColor="White" ForeColor="Black" />
  <RowStyle BackColor="Black" ForeColor="White" />
  <Columns>
    ... Column definition here
  </Columns>
</asp:GridView>
```

זכור ששיבוץ מידע סגנון כמו זה, עלול להוביל לניפוח הדף שתוצאתו היא דפים שקשה לתחזקם. פתרון טוב בהרבה הוא העברה של הגדרת הסגנון לקובץ CSS נפרד ושימוש ב-CssClass של הסגנונות. קובץ CSS שלך אמור להכיל את ה-Selectors הבאים:

```
.AlternatingRowStyle
{
    background-color: white;
    color: black;
}

.RowStyle
{
    background-color: black;
    color: white;
}
```

עם הגדרות CSS אלה עליך לשנות את סגנונות GridView בדרך זו:

```
<AlternatingRowStyle CssClass="AlternatingRowStyle" />
<RowStyle CssClass="RowStyle" />
```

פתרון תרגיל 3

הפקדים קשורי-הנתונים השונים עלולים לעורר חריגים שתוכל לטפל בהם על ידי מטפלי האירוע שלהם. ברגע שטיפלת בחריג כהלכה, עליך לקבוע את המאפיין ExceptionHandled של הארגומנט e למצב True. בקטע הקוד הבא תראה כיצד לעדכן פקד Label עם הודעת שגיאה. ExceptionHandled מוגדר באופן שיעצור את מעבר ה-Exception אל ממשק המשתמש, שכן שם הוא עלול להסתיים ב"מסך המוות הצהוב".

VB.NET

```
Protected Sub SqlDataSource1_Deleted(ByVal sender As Object, _  
    ByVal e As System.Web.UI.WebControls.SqlDataSourceStatusEventArgs) _  
    Handles SqlDataSource1.Deleted  
    If e.Exception IsNot Nothing Then  
        lblErrorMessage.Text = "We're sorry, but something went terribly wrong while " & _  
            "deleting your genre."  
        e.ExceptionHandled = True  
    End If  
End Sub
```

C#

```
protected void SqlDataSource1_Deleted(object sender, SqlDataSourceStatusEventArgs e)  
{  
    if (e.Exception != null)  
    {  
        lblErrorMessage.Text = @"We're sorry, but something went terribly wrong while  
            deleting your genre.";  
        e.ExceptionHandled = true;  
    }  
}
```

פרק 15

פתרון תרגיל 1

כל עניינו של האימות (authentication) הוא הוכחת זהותך למערכת דוגמת אתר אינטרנט. לאחר האימות, ההרשאות שלך הן שקובעות מה תוכל, או לא תוכל, לעשות במערכת.

פתרון תרגיל 2

כדי להרחיב את הגישה לתיקייה Management גם עבור John וכל המשתמשים בעלי תפקיד Editors, עליך להרחיב את המאפיין roles כדי שיכלול את Editors. גם צריך להוסיף אלמנט allow כשהמאפיין users שלו מוגדרת ל-John:

```
<system.web>  
  <authorization>  
    <allow roles="Managers, Editors"/>  
    <allow users="John"/>  
    <deny users="*/>  
  </authorization>  
</system.web>
```

המאפיין roles מאפשר לציין מספר רב של תפקידים, מופרדים בפסיק זה מזה. כדי להעניק לחשבון John את ההרשאה המבוקשת, צריך להוסיף אלמנט allow ולמלא את המאפיין users שלו בשמו של John.

מנקודת המבט של תחזוקה, עדיף בהרבה להוסיף את John לתפקידים Managers או Editors, אם אפשר. עם זאת, אתה עלול לגרום לכך של-John תהיינה יותר הרשאות מאלו שהתכוונת להעניק לו (הוא יוכל אז לגשת לכל מה ש-Managers או Editors יכולים לגשת אליו). ככלל, עדיף לנהל משתמשים באמצעות תפקידים ככל שרק ניתן, אבל טוב לדעת שניתן להעניק לחשבונות יחידים את הזכויות הדרושות בלבד (או לשלול מהם זכויות כלשהן במפורש באמצעות האלמנט deny).

פתרון תרגיל 3

כשתרצה לכוון מחדש את כל המשתמשים לאותו דף, עליך רק להגדיר את DestinationPageUrl:

```
<asp:Login ID="Login1" runat="server" DestinationPageUrl="~/MyProfile.aspx">
```

כששתמש יתחבר בהצלחה, הוא יועבר באופן אוטומטי אל MyProfile.aspx.

פתרון תרגיל 4

LoginStatus מציג טקסט פשוט שמציין אם המשתמש מחובר או לא. ברירת המחדל היא הצגת הטקסט "Login" כשהמשתמש אינו מחובר, והצגת הטקסט "Logout" כשהמשתמש כבר מחובר. לחיצה על הקישור שולחת את המשתמש לדף ברירת המחדל של Login, או מנתקת את חיבור המשתמש.

LoginView דומה במקצת, בכך שהוא מציג תוכן שונה שתלוי בשאלה אם המשתמש מחובר כרגע. עם זאת, מכיוון שהפקד הוא מונע-תבנית לחלוטין, יש לך שליטה מלאה בתוכן שיוצג. תוכל להשתמש באלמנט RoleGroups כדי לקבוע מהן התבניות שתוצגנה רק למשתמשים בעלי תפקיד מסוים, והדבר מאפשר, למשל, הבחנה בין תפקידים שונים של משתמשים.

פרק 16

פתרון תרגיל 1

את ערכת הנושא המועדפת ישם כמאפיין String וקרא לה FavoriteTheme. קבע לה ערך ברירת מחדל כדי להבטיח שתמיד תהיה בידך ערכת נושא תקפה. לבסוף, גרום לכך שהמאפיין יהיה נגיש למשתמשים אנונימיים. בצורתו הסופית יכול המאפיין Profile להיראות כך:

```
<add name="FavoriteTheme" defaultValue="Monochrome" allowAnonymous="true" />
```

פתרון תרגיל 2

על פי התחביר שראית בשאלה, תוכל לגשת כעת למאפיין החדש ולהשתמש בו כדי לשנות את ערכת הנושא הנוכחית ב-BasePage:

VB.NET

```
Private Sub Page_PreInit(ByVal sender As Object, ByVal e As System.EventArgs) _  
    Handles Me.PreInit  
    Dim myProfile As ProfileCommon = CType(HttpContext.Current.Profile, ProfileCommon)  
    If Not String.IsNullOrEmpty(myProfile.FavoriteTheme) Then
```

```

    Page.Theme = myProfile.FavoriteTheme
End If
End Sub

```

C#

```

private void BasePage_PreInit(object sender, EventArgs e)
{
    ProfileCommon myProfile = (ProfileCommon) HttpContext.Current.Profile;
    if (!string.IsNullOrEmpty(myProfile.FavoriteTheme))
    {
        Page.Theme = myProfile.FavoriteTheme;
    }
}

```

פתרון תרגיל 3

כדי להשלים את בורר ערכת הנושא על ידי שימוש ב-Profile, יש צורך גם לשנות את הקוד בדף הראשי. במקום לאחסן את ערכת הנושא שנבחרה על ידי המשתמש בעוגייה, עליך לאחסן אותה כעת ב-Profile. שנה את הקוד ב-Page_Load באופן הבא:

VB.NET

```

If Not Page.IsPostBack Then
    Dim selectedTheme As String = Page.Theme

    If Not String.IsNullOrEmpty(Profile.FavoriteTheme) Then
        selectedTheme = Profile.FavoriteTheme
    End If
    If lstPreferredTheme.Items.FindByValue(selectedTheme) IsNot Nothing Then
        lstPreferredTheme.Items.FindByValue(selectedTheme).Selected = True
    End If
End If

Select Case Page.Theme.ToLower()

```

C#

```

if (!Page.IsPostBack)
{
    string selectedTheme = Page.Theme;
    if (!string.IsNullOrEmpty(Profile.FavoriteTheme))
    {
        selectedTheme = Profile.FavoriteTheme;
    }
    if (lstPreferredTheme.Items.FindByValue(selectedTheme) != null)
    {
        lstPreferredTheme.Items.FindByValue(selectedTheme).Selected = true;
    }
}

switch (Page.Theme.ToLower())

```

את הקוד ב-lstPreferredTheme_SelectedIndexChanged שבדף הראשי ניתן לפשט ולכתוב אותו כך:

VB.NET

```
Protected Sub lstPreferredTheme_SelectedIndexChanged(ByVal sender As Object, _  
    ByVal e As System.EventArgs) Handles lstPreferredTheme.SelectedIndexChanged  
    Profile.FavoriteTheme = lstPreferredTheme.SelectedValue  
    Response.Redirect(Request.Url.ToString())  
End Sub
```

C#

```
protected void lstPreferredTheme_SelectedIndexChanged(object sender, EventArgs e)  
{  
    Profile.FavoriteTheme = lstPreferredTheme.SelectedValue;  
    Response.Redirect(Request.Url.ToString());  
}
```

כדי לאפשר Profile עבור משתמשים אנונימיים, עליך לאפשר זיהוי אנונימי. תעשה זאת עם האלמנט הבא ב-web.config, ישירות תחת <system.web>:

```
<anonymousIdentification enabled="true" cookieName="PlanetWroxAnonymous" />
```

פרק 17

פתרון תרגיל 1

ניפוי (debugging) הוא התהליך של צפייה בביצוע הקוד בסביבת הפיתוח, חקירת משתנים, ובדיקת אובייקטים כדי להבין את נתיב הביצוע של הקוד, תוך חיפוש באגים במטרה לתקנם. הניפוי מתבצע בדרך כלל בזמן הפיתוח ב- Visual Web Developer IDE.

מעקב (tracing) לעומת זאת, מספק מידע על ביצוע הקוד ב-runtime. כפי שדנו בפרק זה, תוכל להשתמש ב-trace כדי לקבל מידע על אירועים שמתעוררים ועל סדר התרחשותם. בנוסף, תוכל להוסיף מידע משלך לצורך מעקב. מכיוון שנטרול המעקב דרך התצורה מקטינה מאוד את תקורת הביצועים שכרוכה במעקב, תוכל להשאיר בקוד את הקריאות לפעולות המעקב, ולהפעיל או לנטרל את המעקב בהתאם לצרכיך.

פתרון תרגיל 2

הדרך הטובה ביותר למנוע חריג אפשרי מלהופיע בממשק המשתמש, היא לעטוף את הקוד בבלוק Try/Catch. בדרך זו תוכל להציג למשתמש הודעת שגיאה מותאמת וברורה למקרה שמשוהו ישתבש. הקוד יראה בסופו של דבר כך:

VB.NET

```
Try  
    ' Execute code here to send an e-mail or write to a file.  
Catch ex As Exception  
    lblErrorMessage.Text = "Something went wrong while executing the requested operation"  
End Try
```

C#

```
try
{
    // Execute code here to send an e-mail or write to a file.
}
catch (Exception ex)
{
    lblErrorMessage.Text = "Something went wrong while executing the requested operation";
}
```

פתרון תרגיל 3

כדי לדעת איזה חריגים קורים באתר ולמצוא היכן הם קורים (כלומר, איזה דפים או קבצי קוד גורמים להם), ביכולתך לרשום את כל החריגים על ידי הקוד שנמצא במטפל האירוע `Application_Error`. פרטי השגיאה שתיירט בשיטה זו אמורים לעזור לך להבין את הגורם לשגיאה, שללא ספק יעזור במציאת תיקון לשגיאה.

כדי למנוע מהמשתמשים לראות את הודעות השגיאה בנוסח של "מסך המוות הצהוב", עליך להשתמש בדפי שגיאה מותאמים. עליך ליצור טופס אינטרנט פשוט שיאמר למשתמשים שמהווה השתבש. כדי לומר ל- ASP.NET runtime להציג את התוכן של קובץ זה במקום את הודעת השגיאה המקורית, עליך להוסיף את האלמנט הבא אל `web.config`:

```
<customErrors mode="On" defaultRedirect="~/Errors/AllOtherErrors.aspx">
  <error statusCode="500" redirect="~/Errors/Error500.aspx" />
</customErrors>
```

אלמנט זה מגדיר דף מיוחד עבור קוד שגיאה 500, שמתרחשת כאשר הקוד נכשל באופן בלתי-צפוי. כשמתרחשים חריגים אחרים, כמו השגיאה "Page Not Found", נשלחים המשתמשים לדף `AllOtherErrors.aspx` שהוא יותר כללי.

נספח ב'

קביעת התצורה של SQL Server 2005/2008

הערה: בזמן כתיבת שורות אלו מיקרוסופט שחררה את **SQL Server 2008 Express Edition**, אשר בכל הנוגע לעבודתנו בספר זה מבחינת היכולות וההגדרות ל-2005, כך שניתן לעבוד מול 2008 באותה דרך בה אנו עובדים עם 2005.

בספר זה אנו מניחים שבסיס הנתונים עבור פרויקט Planet Wrox הוא SQL Server 2005 Express Edition של מיקרוסופט. SQL Server 2005 Express Edition הינו כלי עדיף לפיתוח, מכיוון שהוא ניתן חינם, אינו מעמיס את המחשב, וגם קל לשימוש. עם זאת, לא תמיד הוא מתאים לסביבת ייצור, מכיוון שאינו מתוכנן לטפל במספר רב של משתמשים בו-זמנית. במקרים שבהם המוצר Express Edition אינו מספיק, עליך לחשוב על אחיז הגדולים יותר, הגרסאות המסחריות של SQL Server 2005 דוגמת Standard Edition או Enterprise Edition. בנספח זה תלמד בהרחבה:

- ⊙ אודות אבטחה ב- SQL Server 2005.
- ⊙ איך לאפשר לבסיס נתונים SQL Server 2005 וליישום ASP.NET 3.5 לפעול ביחד.
- ⊙ איך להשיג ולהשתמש ב- SQL Server Management Studio Express, כלי חינוכי של מיקרוסופט שמסייע בניהול בסיס הנתונים.

למרות שהנספח אינו דן ב- SQL Server 2000, תמצא שרוב המושגים חלים גם על גרסה ישנה יותר זו של SQL Server. למעשה, ביכולתך אפילו להשתמש ב- SQL Server Management Studio Express כדי לנהל בסיס נתונים SQL Server 2000.

בנוסף לקביעת התצורה של SQL Server ושל יישום ASP.NET 3.5 באופן שיוכלו לפעול ביחד, תלמד בנספח זה גם כיצד לקבוע את תצורת בסיס נתונים SQL Server באופן שיתמוך בשירותי היישום של ASP.NET 3.5. דבר זה נחוץ כאשר דרוש שהיישום ישתמש בבסיס נתונים מותאם אישית ולא בבסיס הנתונים aspnetdb.mdf שנוצר באופן אוטומטי.

קביעת התצורה של SQL Server 2005

לפני שתקבע את תצורת בסיס הנתונים, עליך להיות מודע לעקרונות האבטחה השונים הקשורים באופן בלתי נמנע עם בסיסי נתונים ויישומי אינטרנט. בפרק 18 למדת על התפקיד החשוב שממלאים חשבונות המחשב השונים שבשימוש שרת האינטרנט בעת קביעת הגדרות האבטחה עבור מערכת הקבצים. הדבר אינו שונה כשמתחברים ל- SQL Server. בסעיף הבא תקבל הדרכה מהירה על דרכי ההתחברות השונות ל- SQL Server. בסעיף שאחריו תלמד כיצד

לצרף את קבצי בסיס הנתונים mdf ל- SQL Server. אחר כך נדון בהגדרת היישום ובסיס הנתונים באופן שיוכלו להתקשר זה עם זה.

מינוח ומושגים

כשאתה מתחבר לבסיס נתונים SQL Server, הוא צריך לדעת מי אתה, כדי שיוכל לכפות על אובייקטים כמו טבלאות בסיס הנתונים את מדיניות הגישה הנכונה. SQL Server 2005 תומך בשני מנגנוני אימות שונים: SQL Server Authentication ו- Windows Authentication (שנקרא לעתים קרובות Integrated Security). לכל אחד מהמנגנונים יתרונות משלו, והם דורשים כתיבת מחרוזות חיבור שונות כדי להתחבר אל SQL Server. בסעיף הבא תכיר את שתי מחרוזות החיבור הנפוצות ביותר, אבל אני ממליץ על ביקור באתר www.connectionstrings.com שבו תמצא רשימה נרחבת של מחרוזות חיבור.

אימות SQL Server (SQL Server Authentication)

באימות SQL Server, בסיס הנתונים הוא זה שמנהל את המשתמשים. ניהול משתמשי בסיס נתונים SQL Server מתבצע על ידי SQL Server Management Studio Express Edition (מאוחר יותר בנספח זה תלמד כיצד להשיגו ולהשתמש בו), או על ידי הגרסאות המלאות שמגיעות עם הגרסאות המסחריות של SQL Server 2005.

כדי לחבר יישום אינטרנט אל SQL Server 2005 שמשתמש באימות SQL Server, עליך להעביר שם משתמש וסיסמה במחרוזת החיבור של היישום. מחרוזת חיבור אופיינית נראית כך:

```
Data Source=ServerName;Initial Catalog=DatabaseName;User Id=UserName;Password=Password;
```

במקרה זה פונה Data Source אל שרת SQL, על ידי ציון שמו, Initial Catalog יכיל את שם בסיס הנתונים, User Id ו- Password יכילו את שם המשתמש והסיסמה בהתאמה.

אימות Windows

באימות Windows (המקביל לאימות של SQL Server), ניהול המשתמשים הוא באחריות מערכת ההפעלה Windows. כדי שבסיס הנתונים ידע מי ניגש למערכת, מתבצעת כל הפעילות האינטראקטיבית מול בסיס הנתונים בהקשר של המשתמש הקורא. עדיין מוטל עליך למפות חשבון Windows לחשבון SQL Server, כדי ש- SQL Server ידע אם החשבון הוא בעל הרשאות מספיקות. מאוחר יותר תראה כיצד לעשות זאת. כשמשמשים באימות Windows תראה מחרוזת חיבור אופיינית כזו:

```
Data Source=ServerName;Initial Catalog=DatabaseName;Trusted_Connection=True
```

במקום לציין שם משתמש וסיסמה, תוסיף למחרוזת החיבור Trusted_Connection=True כדי לציין שברצונך להתחבר לשרת בהקשר של המשתמש הקורא.

מכיוון ששתי השיטות מבצעות בסופו של דבר את אותה הפעולה (הן מאפשרות להתחבר אל SQL Server), מתעוררת השאלה במי מהשתיים להשתמש.

בחירה בין אימות Windows לבין אימות שרת

ככלל, מומלץ להשתמש באימות Windows. העובדה שאינך צריך להשתמש בסיסמה במחרוזת החיבור, פירושה שהיישום יהיה מעט יותר בטוח. אינך צריך לשלוח את הסיסמה בשרת, והיא אינה מאוחסנת בקובץ תצורה של היישום.

עם זאת, אימות SQL Server קל יותר להגדרה. מכיוון שמציינים במפורש את שם המשתמש והסיסמה, אין צורך בידיעת החשבון הסופי של המשתמש שהיישום מתבצע תחתיו.

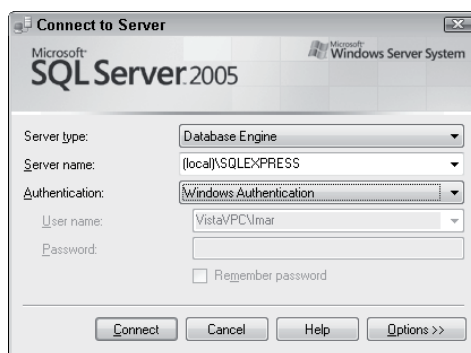
מאוחר יותר בנספח זה תראה כיצד להשתמש באחד משני מנגנוני אימות אלה כדי להתחבר לבסיס הנתונים. עם זאת, עליך להתבונן תחילה במשהו אחר, הכלים שמשמשים לניהול SQL Server.

שימוש ב- SQL Server Management Studio

משתמשים ב- SQL Server Management Studio כדי לנהל בסיסי נתונים. הכלי מאפשר לצרף בסיסי נתונים אל SQL Server או לנתקם ממנו; ליצור אובייקטים חדשים של בסיס נתונים, כמו טבלאות, בבסיסי נתונים קיימים; לבחור, ליצור, לערוך ולמחוק נתונים; והרבה יותר מזה. כשמשתמשים בגרסה מסחרית של SQL Server 2005, הגישה אל SQL Server Management Studio כבר קיימת, מכיוון שהגרסה שמספקים כוללת את מנוע בסיס הנתונים. כשמשתמשים בגרסת החינם Express Edition של SQL Server צריך להוריד מהרשת תחילה את SQL Server Management Studio Express. מהדורת Express זו של כלי ניהול בסיס הנתונים דומה למקבילה המסחרית, ומאפשרת לבצע את רוב משימות ניהול בסיס הנתונים, ואולי אף את כולן.

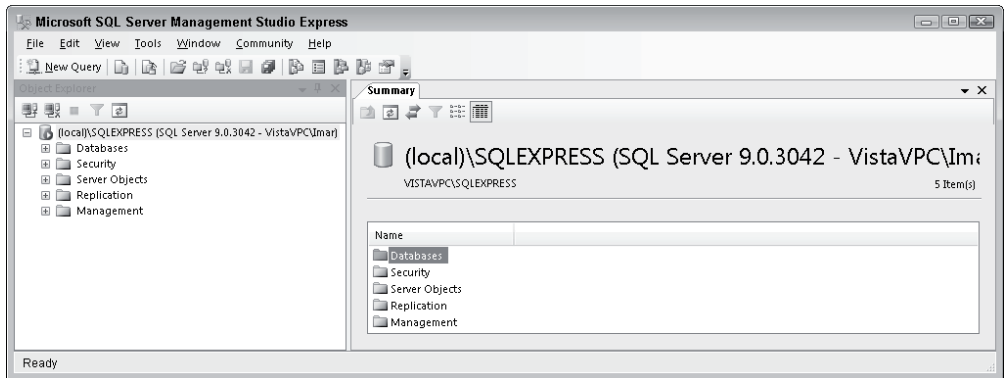
השגת והתקנת SQL Server Management Studio Express

תוכל להוריד את SQL Server Management Studio Express מדף ההורדות הראשי ב- www.microsoft.com/downloads, ולחפש את "SQL Server Management Studio Express". לאחר הורדת קובץ הגדרת Management Studio, לחץ עליו לחיצה כפולה כדי להתחיל בתהליך ההתקנה, ועקוב אחר הוראות המסך. בסיום ההתקנה תוכל להפעיל את SQL Server Management Studio מתפריט Start. יוצג בפניך מסך דומה לזה שבתרשים ב-1 שמאפשר לבחור במופע SQL Server שברצונך להתחבר אליו.



תרשים ב-1

אפשרות ההתחברות תלויה בשרת בסיס הנתונים שאליו מתחברים. באפשרותך להתחבר עם אימות Windows ולהשתמש בחשבון Windows הנוכחי, או שברשימה הנפתחת Authentication תבחר SQL Server Authentication ותזין שם משתמש וסיסמה. לאחר ההתחברות תקבל מסך דומה לזה שבתרשים ב-2 שמאפשר לנהל את מופע SQL Server.



תרשים ב-2

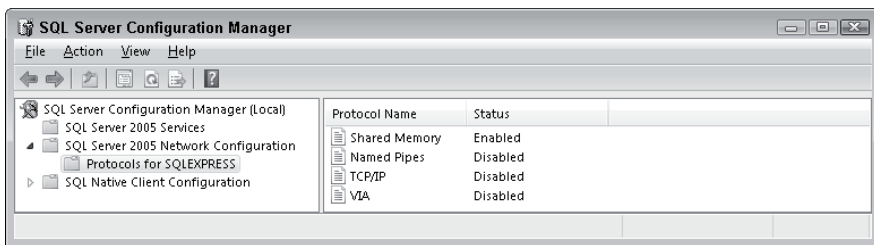
כשיש קושי להתחבר אל SQL Server מרוחק (לדוגמה, מופע SQL Server שאינו נמצא בשרת הפיזי שבו פועל Management Studio), ייתכן שתצטרך להפעיל תחילה את SQL Server עבור חיבורים מרוחק. נדון בכך מיד.

הפעלת חיבורים מרוחק אל שרת SQL Server

כשפועלים עם SQL Server 2005 עלולים לקבל את השגיאה הבאה: An error has occurred while establishing a connection to the server. כשמתחברים ל- SQL Server 2005 יכולה שגיאה זו להיגרם כתוצאה מהעובדה שהגדרות ברירת המחדל של שרת SQL אינן מאפשרות חיבורים מרוחק.

שגיאה זו יכולה להופיע גם כאשר שרת בסיס הנתונים אינו תקין או אינו מגיב, וגם כששרת SQL אינו מוגדר לתמיכה בחיבורים מרוחק. בהתקנה שמתבססת על ברירת המחדל, SQL Server מאפשר רק ליישומים מקומיים להתחבר, וחוסם חיבורים מרוחק באופן אוטומטי. כדי לפתור זאת, ולהעניק גישה לבסיס הנתונים גם למערכות מרוחקות, עקוב אחר הצעדים הבאים:

1. בתפריט Start של SQL Server 2005, פתח את הפריט SQL Server Configuration Manager. פריט זה עשוי להימצא תחת תת-התפריט Configuration Tools, הדבר תלוי בגרסת SQL Server שבשימוש.
2. בחלון שנפתח, אתר את מופע SQL Server תחת הפריט SQL Server 2005 Network Configuration, ולחץ עליו כדי להציג את רשימת הפרוטוקולים הזמינים. תרשים ב-3 מציג את הרשימה עבור מופע SQL Server בשם SQLEXPRESS.



תרשים ב-3

3. ברשימת הפרוטוקולים שמימין, לחץ לחיצה ימנית על Named Pipes, ומתפריט הקיצור בחר Enable (אם הסטטוס הנוכחי מוגדר כ-Disabled).

4. חזור על הצעד הקודם, אבל הפעם בחר TCP/IP.

5. אתחל מחדש את SQL Server. תוכל לעשות זאת כך: לחץ לחיצה ימנית על השרת בסייר האובייקט (Object Explorer) של SQL Server Management Studio (מוצג בתרשים ב-2), ומתפריט הקיצור בחר Restart. אם תקבל שגיאה שנוגעת להרשאות אבטחה, ייתכן שתצטרך לאתחל את המחשב מחדש.

SQL Server מאפשר כעת חיבורים שמקורם ממחשבים מרוחקים. עם זאת, לפני שתוכל להשתמש בבסיסי הנתונים שלך בפועל, עליך לצרף אותם ל- SQL Server. הדבר מתואר בסעיף הבא.

הערה: אם תתקל בקשיים בעת ההתחברות אל SQL Server, ודא ש- SQL Server פועל. כדי לאמת זאת, פתח את לוח הבקרה, ואז פתח את הפריט Administrative Tools (כדי לראות פריט זה ב- Windows Vista, חייב לוח הבקרה להיות בתצוגת Classic View). כעת, פתח את הפריט Services ואמת שמופע SQL Server שאתה מתחבר אליו אכן פועל. אם התקנת SQL Server Express במחשב המקומי, השירות נקרא SQL Server (Express).

צירוף בסיסי נתונים אל SQL Server

SQL Server Express מאפשר לפעול עם קבצי בסיס נתונים בשתי דרכים: ביכולתך לצרף אותם בזמן ביצוע, על ידי שימוש בתכונה מיוחדת של המחרוזת המתחברת, או לצרפם לפני תחילת השימוש בבסיס הנתונים, על ידי שימוש בכלים כמו Management Studio. כשמשתמשים בגרסאות המלאות של SQL Server, חייבים תמיד לצרף את בסיסי הנתונים כפי שמתואר בסעיף זה.

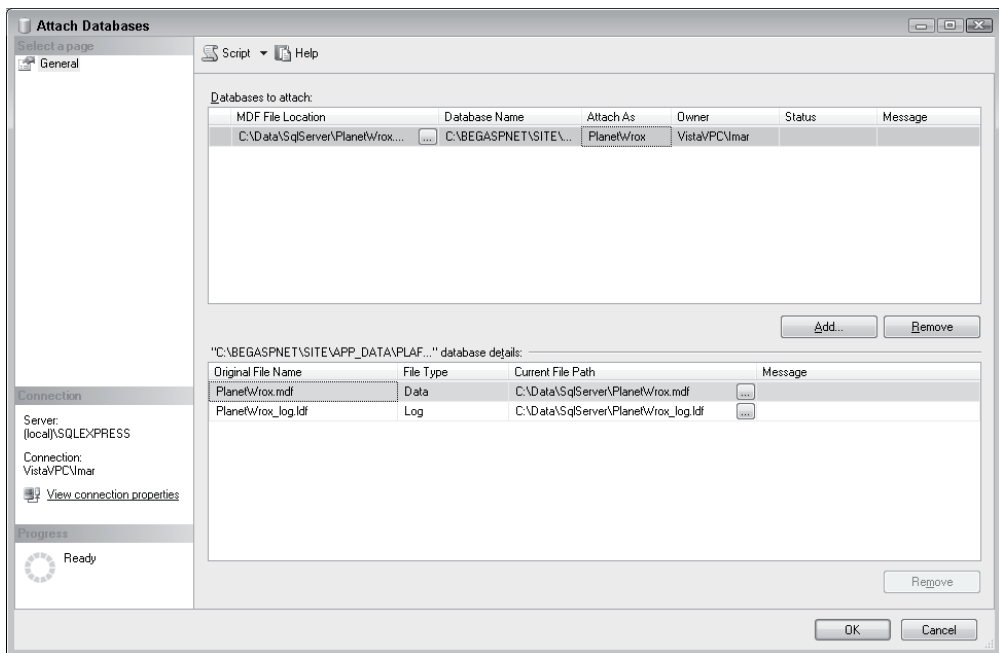
עד כה השתמשת באפשרות הראשונה בכל הפרקים שעסקו בבסיס נתונים. מחרוזת החיבור של Planet Wrox שבה השתמשת נראית כך:

```
<add name="PlanetWroxConnectionString1" connectionString="Data Source=.\SQLEXPRESS;
AttachDbFilename=|DataDirectory|\PlanetWrox.mdf;Trusted_Connection=True;
User Instance=True" providerName="System.Data.SqlClient" />
```

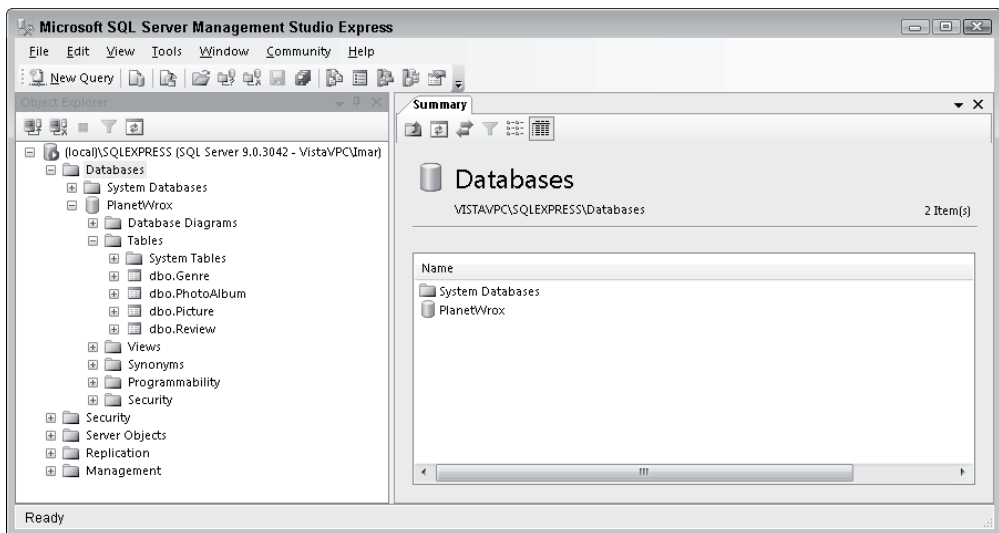
מחרוזת חיבור זו מצביעה על בסיס נתונים בשם PlanetWrox.mdf שנמצא בתיקייה App_Data של אתר האינטרנט (כפי שנקבע על ידי |DataDirectory|). מחרוזת החיבור מורה ל- SQL Server לצרף קובץ בסיס נתונים זה אל SQL Server תוך כדי ביצוע, כשמשתמשים בו. כשבסיס הנתונים אינו נחוץ עוד, הוא ניתן שוב.

זהו פתרון מעולה לפיתוח מקומי, מכיוון שהוא מאפשר להשתמש בקלות בבסיסי נתונים של SQL Server, ליצור בסיסי נתונים חדשים ולהשתמש בהם, ולהעביר בסיסי נתונים מפרויקט לפרויקט. עם זאת, אפשרות זו אינה טובה מספיק כשמדובר בבסיס נתונים שמשמש בייצור, ועליך לצרף תחילה את בסיס הנתונים אל SQL Server. הצעדים הבאים מסבירים כיצד לצרף את בסיס הנתונים PlanetWrox.mdf למופע של SQL Server, במקרה שיתעורר הצורך. תוכל להשתמש באותם צעדים עצמם כדי לצרף בסיס נתונים למופע של SQL Server Express Edition. הדבר שימושי שכברצונך להשתמש ב-SQL Server Management Studio כדי לבצע ב-SQL Server משימות תחזוקה שאינך יכול לבצע עם Visual Studio (לדוגמה, ניהול משתמשים ותפקידים).

1. צור תיקייה שתכיל את בסיס הנתונים החדש, למשל C:\Data\SqlServer.
2. העבר את הקבצים PlanetWrox.mdf ו-aspnetdb.mdf, ואת קבצי Idf הקשורים אליהם, מהתיקייה App_Data של אתר האינטרנט ב-C:\BegASPNET, אל התיקייה החדשה.
3. קבע לחשבון שבשימוש SQL Server (ברירת המחדל, גם ב-Windows XP, היא החשבון Network Service) ולחשבון שלך עצמך, את הרשאות Modify לתיקייה שבה נמצא בסיס הנתונים (C:\Data\SqlServer). פרק 18 מסביר כיצד להעניק הרשאות אלו.
4. פתח SQL Server Management Studio (בגרסה כלשהי) והתחבר למופע של SQL Server שברצונך לצרף לו את בסיס הנתונים. ב-Windows Vista תוכל לבצע תהליך זה רק כמנהלן. כדי לעשות זאת: ב-Windows Start Menu, לחץ לחיצה ימנית על פריט התפריט Management Studio, ומתפריט הקיצור בחר Run as Administrator. אתה אמור לראות מסך דומה לתרשים ב-2, מבלי להתחשב כיצד הפעלת את SQL Server Management Studio.
5. לחץ לחיצה ימנית על הצומת Database, שמוצג בתרשים ב-2, ומתפריט הקיצור בחר Attach.
6. בתיבת הדו-שיח שנפתחת, לחץ על הלחצן Add ובחר בקובץ mdف שהעברת בצעד 2 אל C:\Data\SqlServer.
7. לחץ על הערך בעמודה Attach As והקלד PlanetWrox עבור השם החדש שיינתן לבסיס הנתונים. ברגע שתסיים, אמורה תיבת הדו-שיח להיראות כמו תרשים ב-4.
8. לחץ OK כדי לצרף את בסיס הנתונים אל SQL Server. אם תקבל שגיאה, ודא שהחשבון שלך (או הקבוצה Users שאתה חלק ממנה) והחשבון ASPNET או החשבון Network Service, הם בעלי הרשאות Modify לתיקייה C:\Data\SqlServer ולקבצי mdف שתיקייה זו מכילה.
9. בסיס הנתונים נגיש כעת תחת הצומת Databases של סיר האובייקט של Management Studio. אם תרחיב את האלמנט Databases ותתבונן בבסיס הנתונים, תראה פריטים מוכרים כמו Tables, שכבר ראית בספר זה בעת הדיון בסייר בסיס הנתונים שב-Visual Web Developer. תרשים ב-5 מציג את בסיס הנתונים המצורף והטבלאות שלו.



תרשים ב-4



תרשים ב-5

10. חזור על הצעדים 5 עד 8, אבל הפעם צרף את בסיס הנתונים aspnetdb.mdf בצעד 7 הקלד aspnetdb כשם החדש של בסיס הנתונים.

בשלב זה, רק לחשבונות מנהליים (מנהלני Windows או החשבון המנהלי המובנה של SQL Server שנקרא SA) יש גישה לבסיס הנתונים. כדי שאתר האינטרנט Planet Wrox יפעל עם שני בסיסי נתונים אלה, עליך להגדיר הן את אבטחת SQL Server והן את האתר. מייד נראה כיצד לעשות זאת.

חיבור היישום אל SQL Server 2005

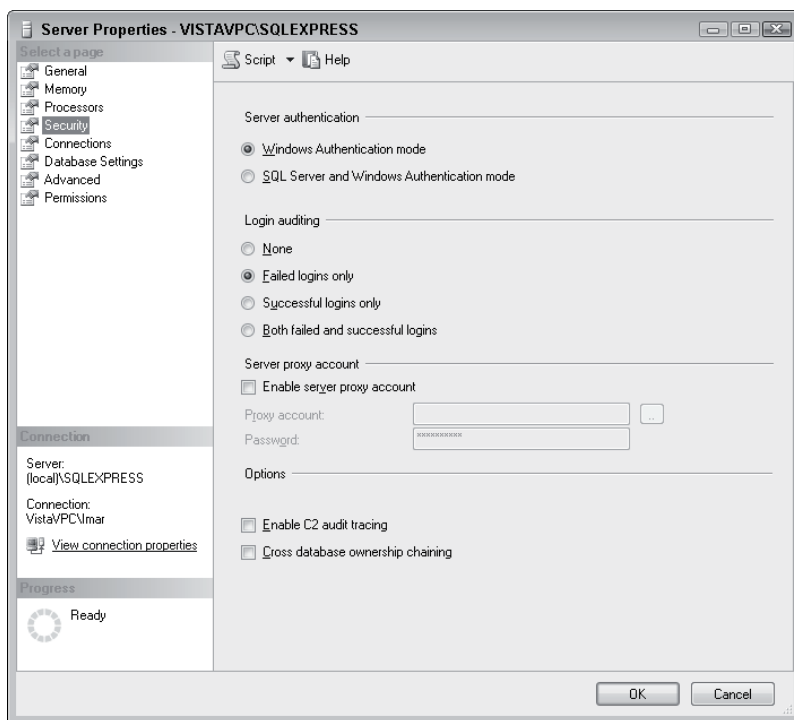
בסעיף הבא תראה כיצד להתחבר אל SQL Server משני תרחישים שונים אך נפוצים: שימוש באימות SQL Server ושימוש באימות Windows כאשר IIS ו-SQL Server נמצאים באותו שרת. בשני התרחישים תלמד כיצד להגדיר את תצורת SQL Server, אתר האינטרנט Planet Wrox, ובמידת הצורך, חשבונות Windows.

כשתפעל עם חברת אירוח חיצונית שמארחת את האתר, סביר שתשתמש בתרחיש הראשון. מארחים ברשת שמציעים SQL Server, משתמשים לעתים קרובות באימות SQL Server, ולפיכך דורשים ממך להעביר לשרת בסיס הנתונים שם משתמש וסיסמה.

התרחיש השני שימושי כשמארחים את האתר במחשב שבו פועלים גם SQL Server 2005 וגם IIS. תרחישים מתקדמים יותר, כמו שימוש באימות Windows כאשר SQL Server ו-IIS נמצאים במחשבים שונים, הם מעבר לתחום הדיון בנספח זה.

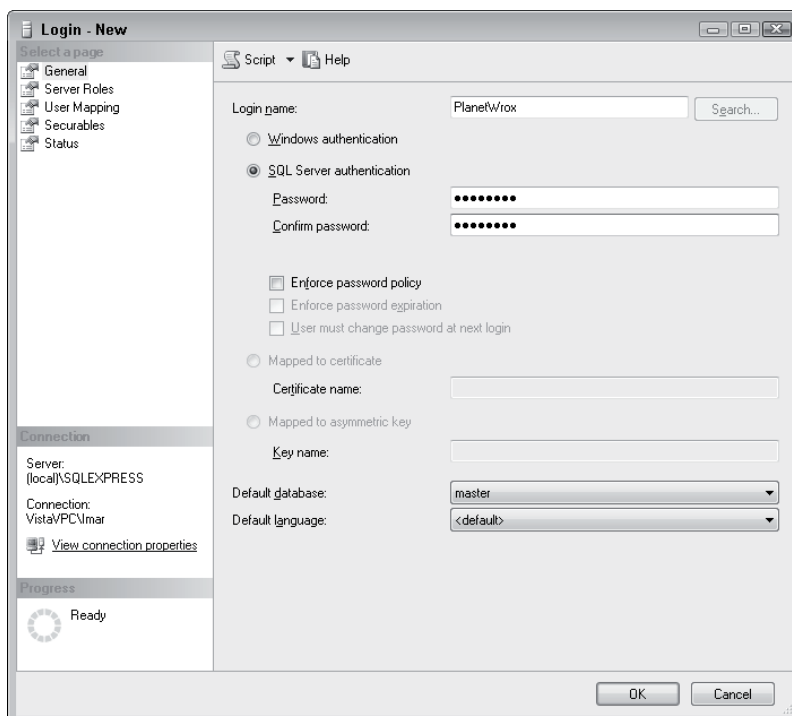
תרחיש 1 - שימוש באימות SQL Server

מנקודת המבט של תצורה, זהו התרחיש הקל ביותר להגדרה. עליך לוודא שמערכת SQL Server שלך תומכת באימות SQL Server, ליצור משתמש ב-SQL Server, ולהשתמש בחשבון זה במחרוזת החיבור של אתר האינטרנט Planet Wrox. כדי לעשות זאת עקוב אחר הצעדים הבאים:



תרשים ב-6

1. ב- SQL Server Management Studio לחץ לחיצה ימנית על שם השרת בסייר האובייקט (מוצג בתרשים ב-2), ומתפריט הקיצור בחר Properties. עבור לקטגוריה Security. תיפתח תיבת הדו-שיח שמוצגת בתרשים ב-6.
2. בחר בפרויט SQL Server and Windows Authentication Mode שבראש המסך (אם טרם נבחר). לפני שתלחץ OK, לחץ על הקישור Help שבראש המסך, קרא מעט על SQL Server Authentication ועל Windows Authentication, והחלט אם אמנם יש צורך באימות SQL Server. אימות Windows בטוח יותר מאשר אימות SQL Server, ולכן אני ממליץ להשתמש באפשרות זו כשרק אפשר.
3. הפעל מחדש את SQL Server. לחץ לחיצה ימנית על השרת בסייר האובייקט, ומתפריט הקיצור בחר Restart. אם תקבל שגיאה שנוגעת להרשאות אבטחה, ייתכן שתצטרך לאתחל את המחשב.
4. חזור לסייר האובייקט של SQL Server Management Studio, והרחב את הצומת Security שמוצג בתרשים ב-2. ודא שבחרת בצומת שנמצא מייד מתחת לשם השרת, ולא בזה ששייך לבסיס נתונים מסוים. לחץ לחיצה ימנית על Logins ומתפריט הקיצור בחר New Login.
5. הקלד שם כניסה למערכת (Login name), בחר באפשרות אימות SQL Server, והקלד סיסמה. בדוגמה זו, ובדוגמאות הבאות, אשתמש בשם PlanetWroxUser כשם המשתמש, וב-Pa\$\$w0rd (אפס" ולא האות O) כסיסמה.



תרשים ב-7

6. נקה את Enforce Password Expiration. פעולה זו גם מבטלת את בחירת User Must Change Password At Next Login. בסופו של דבר אמורה תיבת הדו-שיח להיראות כמו זו שבתרשים ב-7. אם ברצונך לדעת יותר על ההגדרות השונות במסך זה, לחץ על הלחצן Help שבראש המסך.

7. לחץ OK כדי ליצור את החשבון החדש.

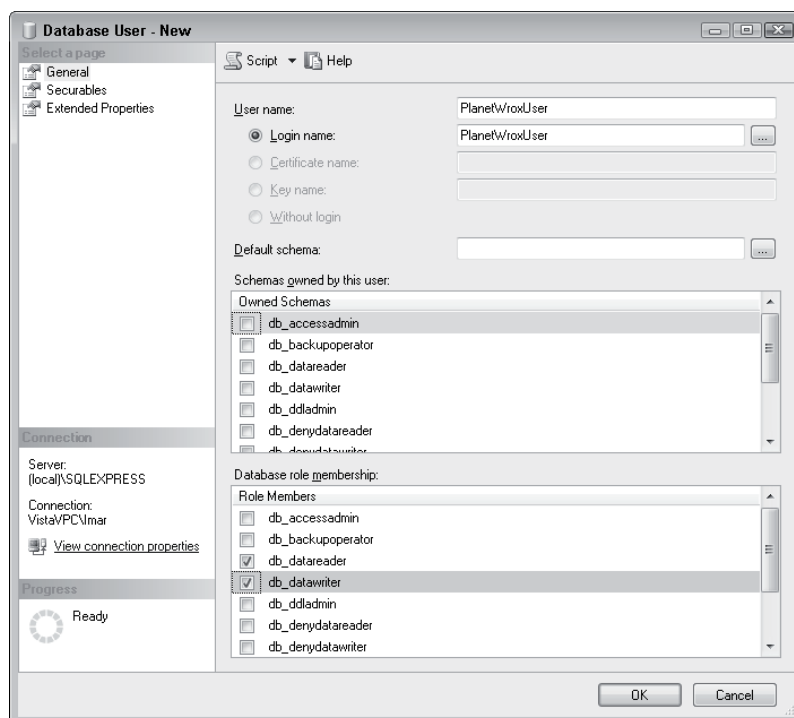
השלב הבא לאחר יצירת החשבון הוא להעניק לחשבון החדש את ההרשאות המתאימות לבסיס הנתונים.

1. בסייר האובייקט, הרחב את Databases, אחר כך הרחב את בסיס הנתונים PlanetWrox, ואחריי הרחב את הצומת Security. לבסוף, הרחב את הצומת Users, לחץ עליו לחיצה ימנית, ומתפריט הקיצור בחר New User.

2. בתיבת הטקסט User name הקלד PlanetWroxUser.

3. בתיבת הטקסט Login name הקלד שוב PlanetWroxUser. לחילופין, לחץ על הלחצן Browse, ואז לחץ Browse ובחר משתמש מהרשימה שנפתחה.

4. בתחתית המסך תראה תיבה שכותרתה Database Role Membership. בתיבה זו תוכל לבחור במספר תפקידים עבור המשתמש החדש. הכלל כאן הוא: קבע למשתמש מספר קטן ככל האפשר של הרשאות. בחירות טובות הן db_datareader ו-db_datawriter. שמאפשרות לחשבון לקרוא מטבלאות מבסיס הנתונים ולכתוב בהן. לכן, סמן שתי אפשרויות אלו, כפי שמוצג בתרשים ב-8.



תרשים ב-8

5. כדי להגדיר אפשרויות אבטחה מפורטות ומגוונות עבור האובייקטים של בסיס הנתונים לחץ על האפשרויות Securables שבחלק השמאלי של תרשים ב-8. תיבת דו-שיח זו מאפשרת לקבוע הרשאות לחשבון המשתמש עבור אובייקטים בבסיס הנתונים כמו טבלאות, תצוגות (views) ופרוצדורות מאוחסנות. אתר האינטרנט Planet Wrox אינו מחייב שינויים כלשהם בתיבת דו-שיח זו.

6. לבסוף, לחץ OK כדי להקצות את החשבון PlanetWroxUser לתפקידים db_datareader ו-db_datawriter.

7. חזור על חמשת הצעדים הקודמים, אבל הפעם הגדר את בסיס הנתונים aspnetdb שצירפת קודם. כשתגיע לצעד 5, בחר בכל התפקידים שמתחילים ב-aspnet_. אלה הם התפקידים ששירותי היישום של ASP.NET משתמשים בהם. קבע לחשבון המשתמש החדש מספיק הרשאות כדי שיתמוך בשירותי היישום השונים כמו חברות ותפקידים, אבל לא יותר מכך. אין צורך לבחור באף אחד מהתפקידים האחרים.

8. סגור את SQL Server Management Studio. בשלב זה אינך זקוק לו עוד.

כעת, כאשר SQL Server וחשבונות המשתמש מוגדרים בצורה נכונה, השלב הסופי הוא שינוי הגדרת אתר האינטרנט כדי שישתמש בחשבון המשתמש החדש הזה.

1. מהתיקייה C:\BegASPNET\Release, פתח את הקובץ web.config של היישום PlanetWrox.

2. שנה את האלמנט <connectionStrings> כך:

```
<connectionStrings>
  <clear />
  <add name="PlanetWroxConnectionString1" connectionString="Data Source=ServerName;
    Initial Catalog=PlanetWrox;User ID=PlanetWroxUser;password=Pa$$w0rD"
    providerName="System.Data.SqlClient"
  />
  <add name="LocalSqlServer" connectionString="Data Source=ServerName;
    Initial Catalog=aspnetdb;User ID=PlanetWroxUser;password=Pa$$w0rD"
    providerName="System.Data.SqlClient"
  />
</connectionStrings>
```

אל תשכח להחליף את הערך ב- Data Source בשם השרת שלך. שם זה תלוי בשרת ובתצורה. הוא יכול להיות משהו פשוט, כמו "(local)" שמצביע על SQL Server במחשב המקומי, או "DatabaseServer" שמצביע על שרת בשם DatabaseServer ברשת, או משהו כמו "DatabaseServer\Sql2005" שמצביע על מופע בעל שם בשם Sql2005 במחשב בשם DatabaseServer.

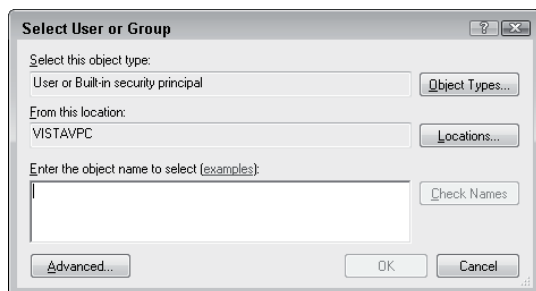
3. שמור את השינויים ופתח את האתר על ידי הפעלת הדפדפן ופנייה לכתובת <http://localhost> (או לכתובת שהגדרת בפרק 18). הכל אמור לפעול עדיין, אבל האתר אינו משתמש עוד בבסיס הנתונים שבתיקייה App_Data אלא ב- SQL Server שהגדרת במחרוזת החיבור.

אם תקבל שגיאה בזמן הדפדוף לאתר, ייתכן שכדי לראות את הודעת השגיאה הממשית תצטרך לכבות את הגדרת השגיאות המותאמות ב-web.config (הגדר אותה RemoteOnly במקום Off). סיבות אפשריות לשגיאה הן שם משתמש או סיסמה שגויים, שם שגוי של שרת במחרוזת החיבור, או שגיאה בהגדרת החברות של החשבון PlanetWroxUser בתפקיד בסיס הנתונים.

תרחיש 2 - שימוש באימות Windows כאשר IIS ובסיס הנתונים באותו מחשב

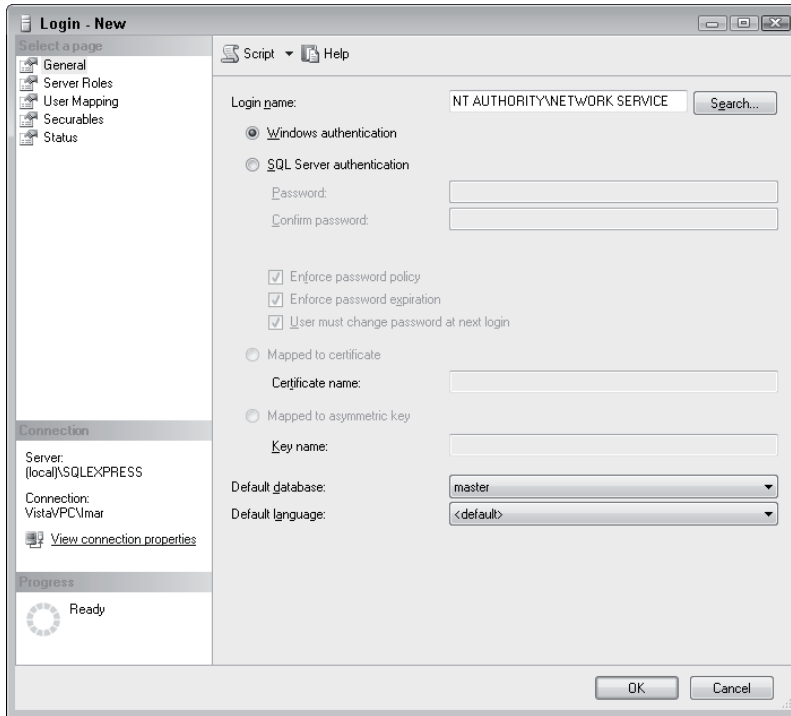
תרחיש זה נפוץ בעיקר במקרה של הפעלת האתר במחשב מקומי שבשליטתך. הן שרת האינטרנט (IIS או שרת האינטרנט המובנה) והן SQL Server מתבצעים באותו מחשב פיזי. תרחיש זה מקל על שימוש באימות Windows, מכיוון שהן שרת האינטרנט והן SQL Server יכולים להשתמש באותו חשבון Windows. כדי להגדיר את השרת עבור תרחיש זה עקוב אחר הצעדים הבאים:

1. מציאת החשבון ששרת האינטרנט משתמש בו. למידע מדויק כיצד לעשות זאת פנה לפרק 18. ככלל, כשהאתר מתבצע בסביבת שרת האינטרנט המובנה, החשבון הוא החשבון שלך. כשמשתמשים ב-IIS בסביבת Windows XP, יש צורך בחשבון של ASP.NET. כשמשתמשים ב-IIS בסביבת Vista או בסביבת גרסת שרת של Windows יש צורך בחשבון Network Service. בהמשך סעיף זה נשתמש בחשבון Network Service.
2. הצעד הבא הוא מיפוי חשבון Windows זה לחשבון SQL Server. פתח SQL Server Management Studio והתחבר למופע SQL Server שלך. הרחב את הצומת Security של השרת (ולא את הצומת של בסיס נתונים מסוים), כפי שמוצג בתרשים ב-2. לחץ לחיצה ימנית על Logins ומתפריט הקיצור בחר New Login.
3. לחץ על הלחצן Search כדי להעלות את תיבת הדו-שיח Select User or Group, שמוצגת בתרשים ב-9.
4. לחץ על הלחצן Advanced ואחריו לחץ על הלחצן Find Now כדי להעלות רשימה עם שמות משתמשים.



תרשים ב-9

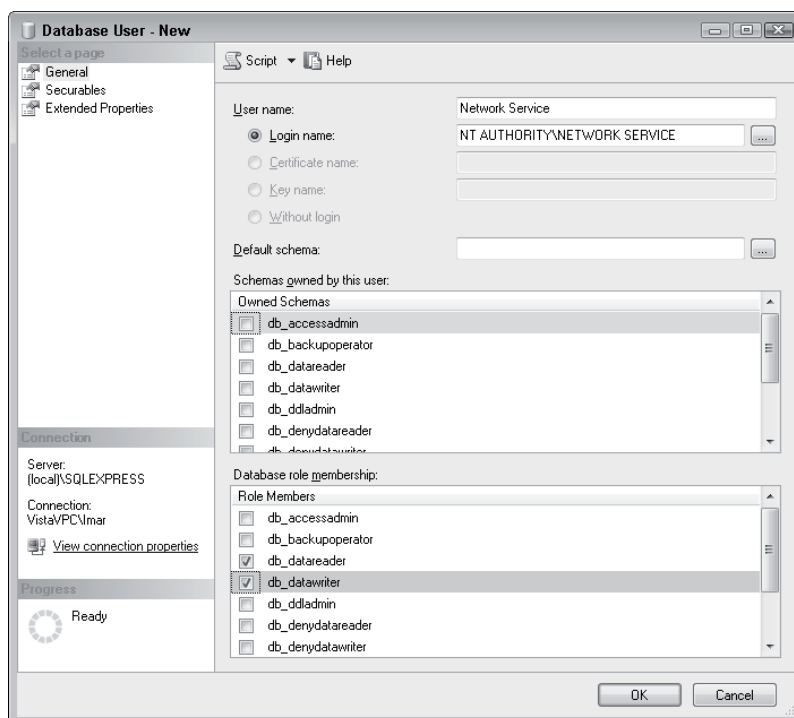
5. בחר ברשימה את חשבון המשתמש שעליך להגדיר, ולחץ פעמיים OK כדי לחזור לתיבת הדו-שיח New Login. התיבה אמורה להכיל כעת את login name הנכון שנבחר, כפי שמוצג בתרשים ב-10.
6. לחץ OK כדי ליצור את חשבון המשתמש החדש.



תרשים ב-10

לאחר יצירת החשבון הצעד הבא הוא להעניק לחשבון חדש זה את הרשאות הגישה המתאימות אל בסיס הנתונים.

1. פתח את הצומת Security של בסיס הנתונים PlanetWrox. הרחב את הצומת Users, לחץ עליו לחיצה ימנית, ומתפריט הקיצור בחר New User.
2. בתיבת הטקסט User name, הקלד Network Service (או את שם החשבון שקבעת קודם).
3. בתיבת הטקסט Login name, לחץ על הלחצן (...) ואז לחץ Browse כדי שתוכל לבחור בשם משתמש. בחר בחשבון NT Authority\Network Service (או את שם החשבון שהגדרת קודם).
4. בתחתית המסך רואים תיבה בשם Database role membership שמיועדת לבחירת תפקידים עבור המשתמש החדש. הכלל הוא: הענק למשתמשים הרשאות מועטות ככל האפשר. בחירות טובות הן db_datareader ו-db_datawriter שמאפשרות לחשבון לקרוא מטבלאות מבסיס הנתונים ולכתוב בהן. לשם כך סמן שתי אפשרויות אלו, כפי שמוצג בתרשים ב-11.



תרשים ב-11

5. כדי להגדיר אפשרויות אבטחה מפורטות ומגוונות עבור האובייקטים של בסיס הנתונים, לחץ על האפשרויות Securables שנראית בתרשים ב-11. תיבת דו-שיח זו מאפשרת לקבוע הרשאות לחשבון המשתמש עבור אובייקטים בבסיס הנתונים כמו טבלאות, תצוגות (views), ופרוצדורות מאוחסנות. אתר האינטרנט Planet Wrox אינו מחייב שינויים כלשהם בתיבת דו-שיח זו.

6. לבסוף, לחץ OK כדי להקצות את החשבון Network Service לתפקידים db_datareader ו-db_datawriter.

7. חזור על הצעדים 1-6, אבל הפעם הגדר את בסיס הנתונים aspnetdb שצירפת קודם. כשתגיע לצעד 5, בחר בכל התפקידים שמתחילים ב-aspnet_. אלה הם התפקידים ששירותי היישום של ASP.NET משתמשים בהם. קבע לחשבון המשתמש החדש מספיק הרשאות כדי שיתמוך בשירותי היישום השונים כמו חברות ותפקידים, אבל לא יותר מכך.

כעת, כאשר SQL Server וחשבונות המשתמש מוגדרים בצורה נכונה, הצעד הבא הוא הגדרת אתר האינטרנט כדי שישתמש בחשבון המשתמש החדש הזה.

1. מהתיקייה C:\BegASPNET\Release, פתח את הקובץ web.config של היישום PlanetWrox.

2. שנה את האלמנט <connectionStrings> כדי שיראה כך:

```
<connectionStrings>
```

```

<clear />
<add name="PlanetWroxConnectionString1" connectionString="Data Source=ServerName;
    Initial Catalog=PlanetWrox;Trusted_Connection=True"
    providerName="System.Data.SqlClient"
/>
<add name="LocalSqlServer" connectionString="Data Source=ServerName;
    Initial Catalog=aspnetdb;Trusted_Connection=True"
    providerName="System.Data.SqlClient"
/>

```

</connectionStrings>

אל תשכח להחליף את הערך ב- Data Source בשם של השרת שלך. הוא יכול להיות משהו פשוט כמו "(local)" שמצביע על SQL Server במחשב המקומי, "DatabaseServer" שמצביע על שרת בשם DatabaseServer ברשת, או משהו כמו "DatabaseServer\Sq12005" שמצביע על מופע בעל שם בשם Sq12005 במחשב בשם DatabaseServer.

3. שמור את השינויים ופתח את האתר על ידי הפעלת הדפדפן ופנייה לכתובת <http://localhost> (או לכתובת שהגדרת בפרק 18). הכל אמור לפעול כמצופה, אבל האתר אינו משתמש עוד בבסיס הנתונים שבתיקייה App_Data אלא ב- SQL Server שהגדרת במחרוזת החיבור באמצעות אימות Windows. עובדה זו מזוהה על ידי התכונה Trusted_Connection=True שבמחרוזת החיבור.

אם תקבל שגיאה בזמן הדפדוף לאתר, ייתכן שכדי לראות את הודעת השגיאה הממשית תצטרך לכבות את הגדרת השגיאות המותאמות ב-web.config (או להגדירה RemoteOnly). סיבות אפשריות לשגיאה הן שם שגוי של שרת במחרוזת החיבור, או שגיאה בהגדרת החברות של החשבון Network Service בתפקיד בסיס הנתונים.

ברגע שמצאת את החשבון הנכון והגדרת נכון את SQL Server, הופך השימוש באימות Windows לקל. למעשה, מחרוזת החיבור נעשית כעת מעט קלה יותר ובטוחה יותר, מכיוון שאינך צריך עוד לאחסן בה שם משתמש וסיסמה.

קביעת התצורה של שירותי היישום

כבר למדת בספר זה ששירותי היישום של ASP.NET משתמשים בבסיס נתונים SQL Server 2005 שניתן לגשת אליו על ידי שימוש במחרוזת חיבור בשם LocalSqlServer. שירותי היישום יוצרים בסיס נתונים חדש בשם aspnetdb.mdf בתיקייה App_Data של האתר בפעם הראשונה שמשתמשים באחד משירותי היישום, כמו Membership או Profile.

מה יקרה כשתרצה להשתמש בבסיס נתונים משלך ולא בבסיס הנתונים aspnetdb.mdf שהוא ברירת המחדל? ומה אם אינך סומך על כך שיייוצר עבורך בסיס נתונים מקומי SQL Server Express, ובמקום זאת ברצונך לקבל את נתוני החברות, התפקיד והפרופיל מהשרת שלך?

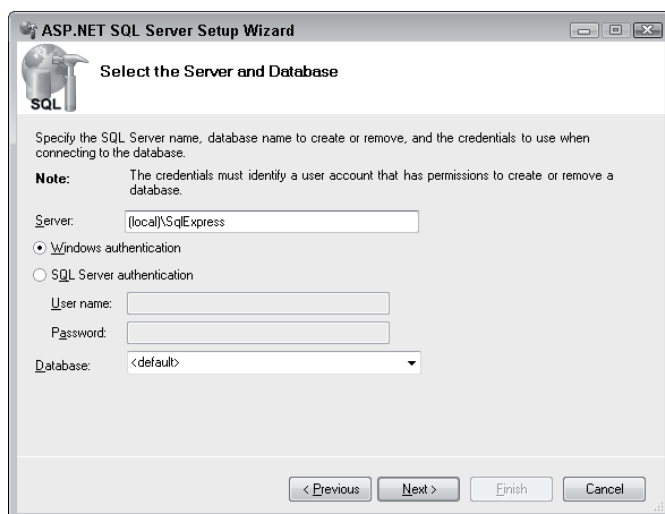
במקרים כאלה עליך לומר ליישום, באמצעות הקובץ web.config, היכן נמצא בסיס הנתונים. יש שתי דרכים לעשות זאת: לדרוס את מחרוזת החיבור LocalSqlServer, או לדרוס את הגדרות שירותי היישום השונים. לפני שנתבונן בשתי אפשרויות אלו, עליך לדעת כיצד להכין בסיס נתונים מותאם אישית עבור שירותי היישום של ASP.NET.

קביעת התצורה של בסיס הנתונים עבור שירותי היישום

כשבסיס הנתונים aspnetdb נוצר בתיקייה App_Data, הוא כבר מכיל כמה אובייקטים כמו טבלאות ופרוצדורות מאוחסנות (קטעי קוד SQL שניתן לבצעם בשרת). כשברצונך להשתמש בבסיס נתונים משלך במקום בזה של ברירת המחדל, עליך להכין אותו לשימוש שירותי היישום. סביבת העבודה NET של מיקרוסופט כוללת כלי שימושי שתוכנן במיוחד למטרה זו. את הכלי, ששמו aspnet_regsql, תמצא בכתובת

```
%windir%\Microsoft.NET\Framework\v2.0.50727
```

בסרגל הכתובת של Windows Explorer עליך להקליד %windir% בדיוק כפי שמוצג כאן, והדבר יורחב באופן אוטומטי לתיקיית Windows. כדי להפעיל את הכלי לחץ לחיצה כפולה על regsql.exe שבתיקייה. פעולה זו מפעילה את ASP.NET SQL Server Setup Wizard. לחץ פעמיים על Next כדי להגיע למסך שמאפשר לבחור ב-SQL Server ובבסיס נתונים (ראה תרשים ב-12).



תרשים ב-12

הקלד את שם שרת בסיס הנתונים (למשל, (local)\SqlExpress) עבור SQL Server Express Edition המקומי, ומהרשימה הנפתחת שבתחתית המסך בחר בסיס נתונים. אם בסיס הנתונים שלך אינו נמצא ברשימה, עליך לצרפו תחילה. מהלך צירוף בסיסי נתונים הוסבר כבר בנספח זה.

לאחר בחירת בסיס נתונים, לחץ פעמיים על Next והכלי ייצור את האובייקטים הדרושים בבסיס הנתונים. לחץ Finish כדי לסגור את האשף, אינך זקוק לו עוד.

גרסת GUI של כלי שירות זה, אשר ראית זה עתה, כוללת תמיכה בכל שירותי היישום הדרושים. אם ברצונך לשלוט שליטה מלאה במה שנוצר, פתח חלון פקודה, נווט אל %windir%\Microsoft.NET\Framework\v2.0.50727, הקלד `aspnet_regsql /?` והקש Enter. כעת תקבל רשימה (ארוכה) של אפשרויות. גלול מעלה ברשימת האפשרויות כדי לראות את כל

האפשרויות הזמינות. כדי להוסיף תמיכה בשירות יישום, עליך להגדיר, לפחות את האפשרויות הבאות:

- ⊙ **-S** : קביעת שם עבור SQL Server.
- ⊙ **-E** : ציון שברצונך להשתמש באימות Windows כדי להתחבר אל SQL Server.
- ⊙ **-A** : לאחר אפשרת זו חייבות לבוא אותיות, אחת או יותר, שמציינות תמיכה בשירותי היישום השונים, m עבור חברות (Membership), r עבור מנהל התפקיד (Role), ו-p עבור Profiles.
- ⊙ **-d** : קביעת שם בסיס הנתונים שאת תצורתו יש להגדיר. אם לא מופיע כאן שם, הכלי משתמש כברירת המחדל של בסיס הנתונים - aspnetdb.

הפקודה הבאה יוצרת תמיכה ב-Membership, ב-Role manager, וב-Profiles כנגד מופע בשם SqlExpress, של SQL Server מקומי, בבסיס הנתונים PlanetWrox:

```
aspnet_regsql -S (local)\SqlExpress -E -A mrp -d PlanetWrox
```

לאחר זמן מה תקבל הודעה שמציינת שהשירותים הנדרשים הוגדרו בהצלחה.

בין אם השתמשת באשף או בממשק שורת הפקודה, מעתה ואילך יוכלו שירותי היישום של ASP.NET להשתמש בבסיס הנתונים שהגדרת.

דריסת מחרוזת החיבור של LocalSqlServer

הדריסה של המחרוזת היא הדרך הקלה והמהירה ביותר לשימוש בבסיס נתונים שונה. עליך רק לדרוס את מחרוזת החיבור LocalSqlServer ולגרום לה להצביע על בסיס הנתונים החדש. כבר ראית בנספח זה מהי מחרוזת חיבור זו, אבל היא מובאת כאן שנית כדי שתוכל לראות כיצד היא פועלת.

```
<connectionStrings>
  <clear />
  ...
  <add name="LocalSqlServer" connectionString="Data Source=.\SQLEXPRESS;
    Initial Catalog=aspnetdb;Trusted_Connection=True"
    providerName="System.Data.SqlClient"
  />
</connectionStrings>
```

שים לב שהאלמנט <connectionStrings> מכיל אלמנט <clear />. האחרון משמש לניקוי כל מחרוזות החיבור שהוגדרו קודם, כולל מחרוזת החיבור LocalSqlServer שהוגדרה ב-machine.config. קובץ התצורה מגדיר עכשיו מחרוזת חיבור חדשה, בעלת אותו שם, אבל עם מחרוזת חיבור שונה. בדוגמה זו מחרוזת החיבור מצביעה על בסיס נתונים מצורף בשם aspnetdb של שרת מקומי בשם SqlExpress. כששירותי היישום צריכים להתחבר אל SQL Server, הם מוצאים את מחרוזת החיבור LocalSqlServer שמאפשרת להם להתחבר ל-SQL Server שהגדרת.

דריסת מחרזות החיבור LocalSqlServer היא דרך מהירה לגרום ליישום אינטרנט בסביבת ASP.NET להשתמש בשרת אחר. עם זאת, לפתרון זה שני חסרונות. ראשית, זה מוזר שמחרזות חיבור בשם LocalSqlServer מצביעה על שרת מרוחק. שנית, אפשרות זו אינה מאפשרת לדרוס את הגדרות הספק לחלוטין. תוכל להימנע משתי בעיות אלו על ידי הגדרה מחדש של תצורת שירותי היישום ב-web.config. תלמד לעשות זאת בסעיף הבא.

דריסת ההגדרות של שירותי היישום

דריסת הגדרות הספקים השונים בקובץ web.config נותנת את הגמישות הרבה ביותר. לא רק שהיא מאפשרת להפנות את שירותי היישום השונים לחיבור של בסיס נתונים חדש, אלא שהיא גם מאפשרת לשנות את ההגדרות האחרות שכל ספק תומך בהן. כדי לדרוס את הגדרות היישום Planet Wrox, עליך לקבוע לכל ספק (Membership, Role Manager ו-Profile) הגדרת <provider> נפרדת.

הקובץ machine.config שבכתובת %windir%\Microsoft.NET\Framework\v2.0.50727\CONFIG מכיל דוגמה כיצד צריכים להיראות הספקים השונים.

אם ברצונך לדרוס את כל שלושת הספקים של היישום Planet Wrox באופן שישתמשו ב-PlanetWroxConnectionString1 והגליל ולא בברירת המחדל LocalSqlServer, תצטרך להוסיף את מידע התצורה הבא לקובץ web.config. כשתוסיף הגדרות אלו, ודא שדרסת את כל ההגדרות הנוכחיות שכבר קיימות בקובץ. כדי שהדבר יתבצע, עליך לקבוע את תצורת בסיס הנתונים PlanetWrox עבור שירותי היישום על ידי שימוש ב-aspnet_regsql, כפי שהוסבר לעיל.

```
<profile>
  <providers>
    <clear />
    <add
      type="System.Web.Profile.SqlProfileProvider, System.Web, Version=2.0.0.0,
        Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
      name="AspNetSqlProfileProvider"
      connectionStringName="PlanetWroxConnectionString1"
      applicationName="/"
    />
  </providers>
  <properties>
    <add name="FirstName" />
    <add name="LastName" />
    <add name="DateOfBirth" type="System.DateTime"/>
    <add name="Bio" />
    <add name="FavoriteGenres" type="System.Collections.Generic.List`1[System.Int32]"
      allowAnonymous="true"
    />
  </properties>
</profile>
```

```

<roleManager enabled="true">
  <providers>
    <clear />
    <add
      type="System.Web.Security.SqlRoleProvider, System.Web, Version=2.0.0.0,
        Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
      name="AspNetSqlRoleProvider"
      connectionStringName="PlanetWroxConnectionString1"
      applicationName="/"
    />
  </providers>
</roleManager>

<membership>
  <providers>
    <clear />
    <add
      type="System.Web.Security.SqlMembershipProvider, System.Web, Version=2.0.0.0,
        Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
      name="AspNetSqlMembershipProvider"
      connectionStringName="PlanetWroxConnectionString1"
      enablePasswordRetrieval="false"
      enablePasswordReset="true"
      requiresQuestionAndAnswer="false"
      applicationName="/"
      requiresUniqueEmail="false"
      passwordFormat="Hashed"
      maxInvalidPasswordAttempts="5"
      minRequiredPasswordLength="6"
      minRequiredNonalphanumericCharacters="1"
      passwordAttemptWindow="10"
      passwordStrengthRegularExpression=""
    />
  </providers>
</membership>

```

לכל שירות יישום יש חלק <providers> שמאפשר להגדיר את ספקי השירות (אחד, או יותר). האלמנט <clear /> משמש להסיר מהיררכיית התצורה את כל הספקים שהוגדרו קודם. אחר כך מוסיפים כל ספק על ידי האלמנט <add />. שים לב שכל קטעי התצורה משתמשים ב-connectionStringName="PlanetWroxConnectionString1" כדי להצביע על מחזורת החיבור המותאמת אישית שמוגדרת באותו קובץ web.config. בנוסף, שים לב שביכולתך לשנות כל אחת מההגדרות האחרות שנתמכות על ידי הספק.

כל הגדרות התצורה של הספק הן בעלות המאפיין applicationName שמוגדרת כ-"/". זהו השם שכל יישום משתמש בו כדי לחפש בבסיס הנתונים משתמשים מבוקשים, תפקידים ועוד. תוכל

ליצור שם אחר, אם תרצה, אבל חשוב להקפיד שכל הספקים ישתמשו באותו applicationName, אחרת ימצאו עצמם מחפשים מידע לא נכון. ללא applicationName תואם, השירותים לא יהיו מסוגלים לקשר רשומות של שירות אחד לאחר. דבר זה בתורו עלול להוביל לחוסר התאמה בבסיס הנתונים בין משתמשים, תפקידים והפרופילים שלהם.

לאחר שתצורת שירותי היישום הוגדרה מחדש, אינך זקוק עוד למחרוזת החיבור LocalSqlServer שייתכן והוספת ל-web.config מוקדם יותר בנספח זה.

לאחר שמירת web.config עם כל השינויים שנעשו בו, האתר אמור להמשיך להתבצע כרגיל. עם זאת, שירותי היישום ישתמשו כעת בבסיס הנתונים PlanetWrox ולא בבסיס נתונים aspnetdb נפרד.

נספח ג'

תכונות חדשות (SP1)

(נכתב על ידי מאיר קרודו)

ASP.NET 3.5 Service Pack 1

מי בכלל צריך לדעת משהו על Service Pack? קרוב לוודאי שאתה סבור שחבילות שירות נועדו רק לתיקון באגים במוצר, ואינן מספקות באמת שום ערך חדש. בדרך כלל, זה נכון. עם זאת, מיקרוסופט בחרה לפרסם קבוצה של תכונות חדשות במסגרת העדכון NET 3.5 Service Pack 1 (להלן נשתמש גם בכינוי המקוצר SP1). קבוצת תכונות זו מתפרסת על פני סביבת העבודה NET, ובכלל זה גם ASP.NET. נספח זה עוסק בעיקר בתכונות החדשות של ASP.NET.

בסעיפים הבאים תמצא תיאור של התוספות ב-SP1.

מידע נוסף תוכל למצוא בספרון: **New Feature in ASP.NET 3.5 Service Pack 1** של הוצאת הספרים WROX. תוכל למצוא אותו באתר של WROX בלינק:

<http://www.wrox.com/WileyCDA/WroxTitle/New-Features-in-ASP-NET-3-5-Service-Pack-1.productCd-0470457341.html>

היסטוריית Ajax

כמה פעמים קרה שאתה או מי מהמשתמשים שלך פניתם ליישום Ajax, ביצעתם כמה פעולות, ולחצתם על הלחצן Back בדפדפן? האם ציפית לחזור למצב הקודם של היישום, ומצאת עצמך לגמרי מחוץ ליישום? קרוב לוודאי שנלקחת אל מחוץ ליישום, אל הדף שבו היית לפני שנכנסת ליישום. מדוע זה קרה? דפדפן האינטרנט אינו יודע שהדף עצמו השתנה. אם תבדוק את כתובת האינטרנט (ה-URL), תמצא שהיא נשארה ללא שינוי, וגם לא היו posts כלשהם, ולכן הדפדפן לא יכול היה לדעת שמהו השתנה.

כיום, עם שחרור ASP.NET 3.5 Service Pack 1, יכולה ASP.NET AJAX לתמוך בלחצנים Back ו-Forward בדפדפן. לרוע המזל, התקנת SP1 בלבד לא תעזור. יש צורך להוסיף קוד כדי להפיק תועלת מהתכונה.

jQuery

מבחינה טכנית jQuery אינה תכונה של .NET 3.5 Service Pack 1. זמן קצר לאחר שחרור SP1 להפצה הכריזה מיקרוסופט כי ספריית המקור הפתוח jQuery JavaScript תיכלל בגרסאות עתידיות של Visual Studio. בינתיים שחררה מיקרוסופט גרסת IntelliSense-encoded (מקודד) של jQuery, שנתרמה לארגון jQuery. ניתן להשתמש בקובץ זה כדי לקבל תמיכה מלאה של IntelliSense ב-JQuery במסגרת Visual Studio 2008.

לא ניתן לתת סקירה מלאה של jQuery במגבלה של עמודים אחדים, אבל אשתדל לתת כאן סקירה קצרה. אחד הדברים המגניבים שהתוכנה מספקת זו היכולת לקרוא לשירות אינטרנט בצורה ישירה, מבלי להזדקק ל-ScriptManager של ASP.NET. עם jQuery, הגרסה הנוכחית לעיל מטפלת בתקשורת עם שירות האינטרנט, מקבלת את הנתונים חזרה, ופועלת בצורה אינטראקטיבית עם התוצאה מבלי להשתמש ב-ASP.NET AJAX Script Manager.

השילוב של ספריית jQuery עם שירותי האינטרנט של ASP.NET AJAX, שאינו גורם לאף אחת מהתקורות של ASP.NET AJAX Script Manager, הוא שמקנה ל-JQuery את יתרונה הגדול. אולי זה לא נשמע כמשהו מיוחד, עם זאת, כל חיסכון בהורדת נפח הנתונים משפר את ביצועי המערכת (חשוב על רשת בעלת מאות או אלפי מסופים שפועלת עם יישום אחד!).

שאלה שאני בטוח שמתכנתים שואלים היא, "מדוע להשתמש ב-JQuery במקום לקרוא באמצעות ASP.NET AJAX?" ובכן, התשובה היא ש-ASP.NET AJAX מספקת תמיכה פשוטה וקלה לקריאת שירותי אינטרנט. לתמיכה זו יש מחיר - העברת יותר מידע מהשירות אל דפדפן הלקוח. jQuery מספקת גם היא תמיכה לקריאת שירותי אינטרנט. חווית הפיתוח איתה אמנם אינה כה טובה, אבל התוצאה היא שניתן להבחין בשיפור קל בביצועים. הדבר נובע מכך שעם jQuery פחות חומר מורד מהשרת לדפדפן הלקוח.

Entity Framework

במסגרת חבילת התוכנה SP1 שחררה מיקרוסופט לשוק את ADO.NET Entity Framework. מבחינה טכנית, ADO.NET Entity Framework אינה חלק מ-ASP.NET. עם זאת, יחד עם AJAX ונתונים דינמיים של ASP.NET, היא מהווה חלק מהבסיס של מודל הנתונים של ADO.NET.

טכנולוגיה זו מפשטת את צורת העבודה עם מבנה הנתונים, אשר מאוחסנים בדרך כלל בבסיס נתונים, למודל נתונים ידידותי יותר מבחינת האפליקציה.

לדוגמה, בבסיס הנתונים ניתן לשמור מידע על לקוחות בטבלה בשם Customers ואת המידע על ההזמנות של הלקוחות ניתן לשמור בטבלת Orders, ונתונים לגבי אנשי קשר ניתן לשמור בטבלת Contacts וכו'.

כאשר מערכת מסוימת אמורה לעבוד עם הנתונים הללו, היא צריכה לדעת איזה נתון נמצא באיזו טבלה, דבר זה מצביע על קשר בין המערכת לבין מבנה הנתונים.

החיסרון בגישה זו היא, שכאשר מבנה הנתונים משתנה, המערכת לא חסינה מפני שינוי זה והיא חייבת כעת להתאים את עצמה למבנה החדש.

בנוסף, המערכת חייבת לבצע קשרים בין הטבלאות כדי להביא נתונים. לדוגמה, כדי לאחזר את רשימת ההזמנות של לקוח מסוים, יש לקשור את טבלת Customer עם טבלת Orders, ואז לשלוף את הנתונים הנכונים לפי הלקוח.

במהלך העבודה השוטף, המתכנת רגיל לעבוד עם אובייקטים. כך שכאשר ירצה לקבל את ההזמנות של לקוח מסוים, יהיה עליו לטעון אובייקט לקוח ולגשת להזמנות שלו באמצעות מאפיינים (Properties).

נתונים דינמיים (Dynamic Data)

Dynamic Data מספקת תשתית המאפשרת לך ליצור במהירות ובקלות אפליקציות אינטרנט מבוססות נתונים. התשתית הזו משתמשת בטכנולוגיית LINQ to SQL או במודל הנתונים של Entity Framework. טכנולוגיה זאת מנצלת יכולות ופונקציונליות של פקדים כמו: **ListView**, **GridView**, **FormView**, **DetailsView** כדי לייצג נתונים טבלאיים בתבניות מותאמות ולאפשר בדיקת תקפות ואימות נתונים.

שירותי הנתונים של ADO.NET

הרעיון מאחורי טכנולוגיה זו, היא לאפשר למערכות לחשוף מידע (נתונים) כשירות ברשת אשר ניתן יהיה לצרוך אותם על ידי אפליקציות web ברשת משותפת או ברחבי האינטרנט.

נגשים לשירותים אלו על ידי בקשות http לצורך ביצוע פעולות CRUD.

הפשטות שבשימוש בטכנולוגיות אינטרנט גורמת לכך, שטכנולוגיה כמו ADO.NET Data Services הינה אידיאלית לשימוש כמקור נתונים למערכות נוסח AJAX בעלות ממשק משתמש עשיר.

נספח ד'

Silverlight

(נכתב על ידי אלכס גולש)

נספח זה עוסק בנושאים הבאים:

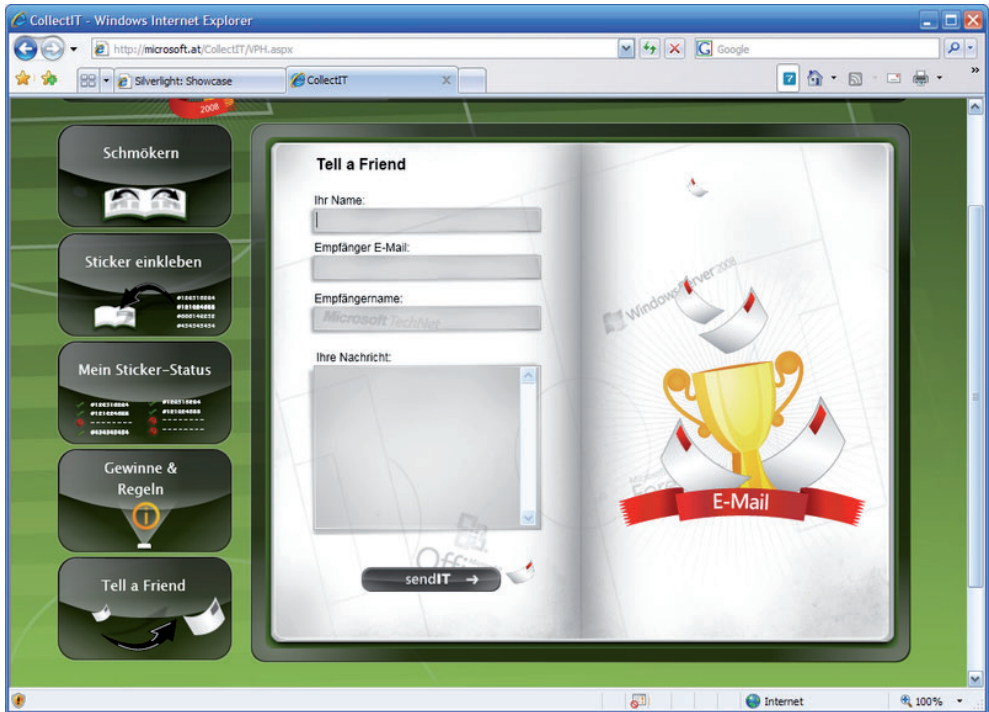
- ⊙ מהו Silverlight?
- ⊙ XAML
- ⊙ סגנונות ותבניות
- ⊙ קישור נתונים

מהו Silverlight?

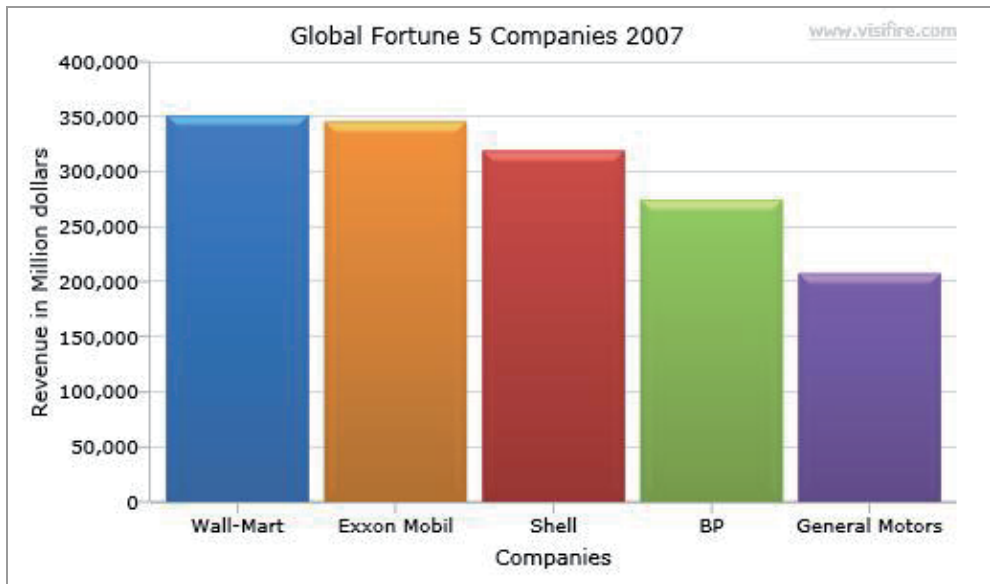
Silverlight הוא יישום חדש של סביבת העבודה NET שמשמש לבניית הדור הבא מערכות אינטרנט, המאפשרות חוויות מדיה וממשק משתמש עשיר. היישום ניתן להפעלה עם דפדפנים שונים ובפלטפורמות שונות. היישום מתבצע בכל הדפדפנים הנפוצים כולל Internet Explorer של מיקרוסופט, Firefox של Mozilla, Safari של Apple ו-Google Chrome, באמצעות plug-in המותקן בדפדפן שהינו קטן מאוד (כ-4.5MB) ולפיכך התקנתו מהירה מאוד.

Silverlight הוא שילוב של טכנולוגיות שונות לפלטפורמת פיתוח אחת, שמאפשרת לבחור את הכלים ושפת התכנות שברצונך להשתמש בהם. Silverlight משתלב באופן חלק עם JavaScript וקוד AJAX קיימים, כדי להשלים פונקציונליות שכבר יצרת.

XAML (eXtensible Application Markup Language) היא הבסיס ליכולות התצוגה של Silverlight, ועקבית עם **WPF** (Windows Presentation Foundation), טכנולוגיית התצוגה בסביבת העבודה .NET 3.0. XAML מאפשרת ליצור ממשק משתמש יוצא דופן – ממשק משתמש שמתבסס על גרפיקת וקטורים, שכל חלק ממנה יכול לארח כמעט כל חלק אחר באופן טבעי.



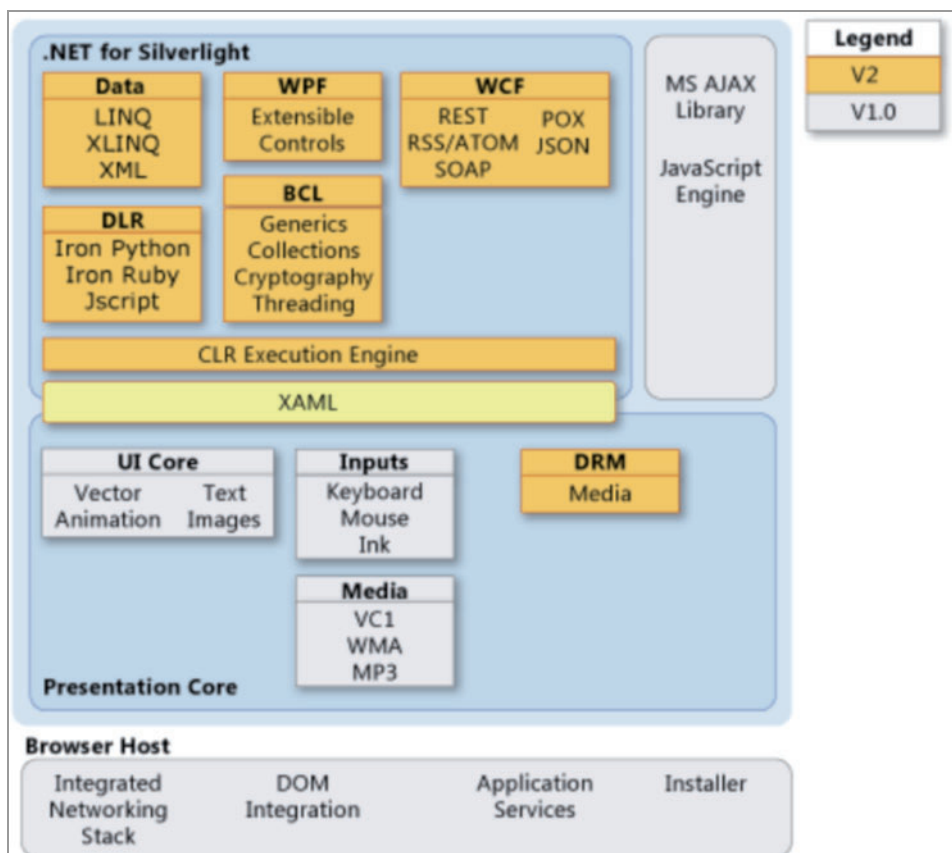




להלן כמה מאפיינים בולטים של Silverlight:

1. **חוויית וידאו מהאיכות הגבוהה ביותר.** הייה נכון לצפות בכמה סרטי וידאו מהאיכות הטובה ביותר שראית אי-פעם, כולם מוטבעים באתרי אינטרנט גרפיים באיכות גבוהה. טכנולוגיות ה- streaming media שבהן משתמשת מיקרוסופט כיום, הן אותן טכנולוגיות ששימשו עבור VC-1, ה-codec שהניע את BluRay ו-HD DVD.
2. **מגוון פלטפורמות (מערכות הפעלה ודפדפנים).** סוף-סוף קיימת היכולת לבנות יישומי אינטרנט שיפעלו בכל דפדפן ובסביבת כל מערכת הפעלה. בעת שחרורו מבטיח Silverlight עבודה הן עם Mac והן עם Windows! ופרויקט Mono כבר הבטיח תמיכה ב-Linux!
3. **מפתחים ומעצבים גרפיים יכולים לשחק יחד!** מפתחים שמכירים את Visual Studio וסביבת NET, יוכלו לפתח במהירות יישומי Silverlight מדהימים. יישומים אלה יפעלו הן בסביבת Mac והן בסביבת Windows. מפתחים יוכלו, סוף-סוף, להתמקד ב-back-end של ליבת היישום, ולהשאיר את הפן החזותי לצוות העיצוב הגרפי שישתמש בעוצמת XAML.
4. **זול יותר.** Silverlight הינו הדרך הזולה ביותר שקיימת כיום להזרמת קבצי וידאו באינטרנט, ובאיכות הגבוהה ביותר האפשרית. הרישוי פשוט ביותר. כל מה שאתה צריך הוא IIS בשרת Windows.
5. **תמיכה בשפות צד שלישי.** מפתחים יוכלו מעתה להשתמש ב-Ruby, Python ו-EcmaScript כשהם משתמשים בעוצמה של Dynamic Language Runtime. פירוש הדבר שמפתח Ruby יוכל לפתח יישומי Silverlight כשהוא נהנה מיתרונות סביבת העבודה .NET.

6. ניפוי מרחוק ללא תלות בפלטפורמה או בדפדפן. כשמתעורר צורך לנפות יישום שמתבצע ב-Mac - אין שום קושי. תוכל כעת להשתמש בכל עזרי הניפוי הטובים שמספקת Visual Studio, כגון קביעת נקודות עצירה (breakpoints), ואפשרות "להסתובב" בתוך הקוד, ולנתחו באמצעות הכלים שהמפתח מכיר.
 7. סביבת הפיתוח הטובה ביותר שקיימת. Visual Studio הינה סביבת פיתוח עטורת פרסים! כפי שהיא ממשיכה להתפתח בקביעות, כך ימשיך גם Silverlight.
 8. **Silverlight מאפשר שמירת עותק.** האם שמת לב כמה קל להוריד סרטי וידאו מ-YouTube למחשב שלך, ולשמור אותם לצפייה מאוחר יותר? ל-Silverlight יהיו לבסוף התכונות שתאפשרנה לספקי תוכן שליטה מלאה על תכני המולטימדיה האינטראקטיביים שלהם (rich media content). streaming television, תחנות שידור עצמאיות חדשות, הכל יהיה אפשרי מעתה!
 9. מהירות גדולה ביותר. אתרי אינטרנט שמאפשרים AJAX ומתחילים להשתמש ב-Silverlight חשים שיפור דרמטי במהירות, תוך מינוף סביבת העבודה .NET.
- בכוונת Silverlight להתחרות ב- Adobe Flash וברכיבי התצוגה של Ajax. הוא מתחרה גם ב-JavaFX של Sun Microsystems ששוחרר כמה ימים לאחר Silverlight.



ארכיטקטורת Silverlight

ה-plug-in המותקן בדפדפן כולל runtime שמבוסס על תת-מערכת של .NET Framework 3.5. הוא מספק את תת-המערכת הפונקציונלית הרלוונטית. ה-plug-in מתאים לכל סביבת הפעלה ולכל דפדפן. כל יישומי Silverlight מתארחים בדף HTML (כל צורה שלו), והם בעלי גישה מלאה ל-HTML DOM. בנוסף, הם מאפשרים גישה מהסביבה המארחת (JavaScript, HTML DOM) לתוכן שלהם (managed code).

הקוד שנכתב עבור יישומי Silverlight תואם במידה רבה את הקוד שנכתב עבור סביבת העבודה NET המלאה, אבל התאימות היא בקוד המקור בלבד.

יישומי Silverlight נוצרים מתוך Visual Studio 2008 עם SP1 (כל גרסאות Visual Studio, כולל המהדורות החופשיות של "Express") וכלי Silverlight עבור Visual Studio 2008 מותקנים. מפתחים מקבלים חווית פיתוח מוכרת (IntelliSense, צביעת קוד, בדיקת סוג מובהק, וכו') ויכולים להשתמש בידע קודם שלהם מסביבת NET כדי ליצור את הדור הבא של יישומי אינטרנט.

XAML

Silverlight משתמש ב-XAML (eXtensible Application Markup Language) כדי להגדיר ממשק משתמש. XAML הינה שפת סימני עריכה שמבוססת על XML. XAML היא שפה הצהרתית שמאפשרת לבטא כל אובייקט CLR (מתוך עץ המחלקות של NET) תוך שימוש בקבוצת מאפיינים.

מדוע XAML?

- ⊙ הפרדה בין העיצוב לקוד
- ⊙ יצירה קלה של כלים (אשר ייצרו/יבינו XML)
- ⊙ בניה היררכית
- ⊙ ניתנת בקלות ל-parsing ול-serialize / deserialize
- ⊙ יותר קריאה מקוד
- ⊙ יותר קצרה מקוד

קבצי XAML הם קבצי XML בעלי סיומת .xaml.

לקבצי XAML יש הכרזות מרחב שמות שממופות לסכמה או לאוספים:

```
<UserControl x:Class="SelaPlayer.Page"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Width="800" Height="600">
  <Button Content="Hello Silverlight!"/>
</UserControl>
```

כל אלמנט שנוצר ב-XAML יוצר ב-runtime אובייקט NET ממשי. כל מה שמבוטא ב-XAML יכול להיות מבוטא כאובייקטים בקוד ברקע.

XAML:

```
<TextBlock Text="Hello Silverlight" FontSize="25"/>
```

Code Behind:

```
TextBlock textBlock = new TextBlock();
textBlock.FontSize = 25;
textBlock.Text = "Hello Silverlight";
```

מאפיינים ב-XAML מבוטאים כתכונות של XML.

מאפייני XAML משמשים להוספת תכונות לאלמנטים של אובייקט כמו Proerties ו-Events. בקטע סימני העריכה שלהלן משמשים המאפיינים Text, Height ו-Width של XAML כדי להגדיר את אותם המאפיינים בפקד TextBlock:

```
<TextBlock Title= "Silverlight" Height="100" Width="200" />
```

ערכים מורכבים ניתנים להגדרה תוך שימוש בתחביר property element כמו:

```
<Element.Property>
<Button Width="149" Height="102" Content="Click Me">
  <Button.Background>
    <LinearGradientBrush EndPoint="0,1" StartPoint="0,0">
      <GradientStop Color="#FFF2E8E8" Offset="0"/>
      <GradientStop Color="#FFE8E8E8" Offset="0.5"/>
      <GradientStop Color="#FFD29F9F" Offset="0.5"/>
      <GradientStop Color="#FFF2E8E8" Offset="1"/>
    </LinearGradientBrush>
  </Button.Background>
</Button>
```

אלמנטים של XAML יורשים את המאפיין x:Name.

ניתן להשתמש ב-x:Name המוצהר של אלמנט כדי לפנות לאלמנט מקוד ברקע.

XAML:

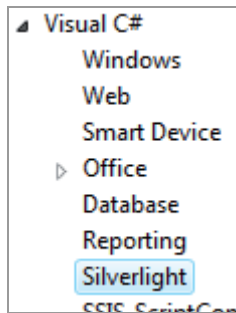
```
</Button x:Name="buttonClickMe".../>
```

Code Behind:

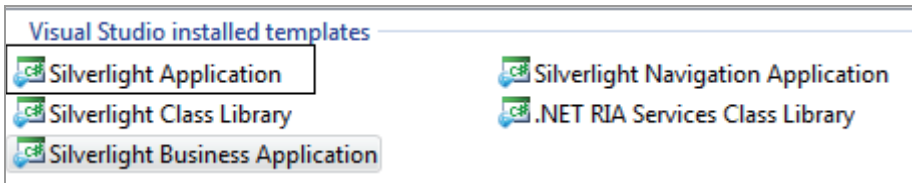
```
buttonClickMe.Content = "Click Me!";
buttonClickMe.Click += buttonClickMe_Click;
```

ניצור עכשיו יישום ראשון פשוט:

1. פתח את Visual Studio 2008, ומ- Project Types בחר ב-Silverlight.



2. בחר ב- Silverlight Application.



3. במקרה שלפנינו נבנה יישום שייראה כמו התרשים הבא:

4. ה-XAML עבור מסך כזה יהיה:

```
<Grid Margin="5" ShowGridLines="False" >
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="*" />
    <ColumnDefinition Width="2*" />
  </Grid.ColumnDefinitions>
  <Grid.RowDefinitions>
    <RowDefinition />
    <RowDefinition />
    <RowDefinition />
  </Grid.RowDefinitions>
  <TextBlock Text="User Name" Grid.Column="0" Grid.Row="0" />
  <TextBlock Text="Password" Grid.Column="0" Grid.Row="1" />
  <TextBox Text="Jhon Doe" Grid.Column="1" Grid.Row="0" Margin="5" />
  <PasswordBox Password="Secret" Grid.Column="1" Grid.Row="1" Margin="5" />
  <Button Content="Login" Grid.Column="1" Grid.Row="2"
    HorizontalAlignment="Right" Width="100" Height="25"
    Margin="5" />
</Grid>
```

User Name	Jhon Doe
Password	
Login	

סגנונות ותבניות

ליישום יש מראה ותחושה רגילים של יישום Windows. זה משעמם למדי, בפרט עבור RIA. Silverlight משתמש בשתי תכונות כדי לטפל בכך: עיצוב סגנון ותבניות.

סגנון – מנגנון שמאפשר להשתמש בערכי מאפיינים של פקד כמשאבים לשימוש חוזר. מבחינה מושגית הדבר דומה ל-CSS עם HTML. הגדרת הסגנון כוללת את סוג הפקד, כלומר, הסגנון חל רק על הפקד הספציפי.

סגנון יכול להיגזר מסגנון אחר של אותו פקד על ידי שימוש במאפיין BasedOn.

הבה נגדיר סגנון עבור הפקד Border:

```
<Style x:Key="borderStyle" TargetType="Border">
  < Setter Property="CornerRadius" Value="10"/>
  < Setter Property="Background" Value="#FFDEDEDE"/>
  < Setter Property="Margin" Value="10"/>
</Style>
```

הצבת סגנון בתוך דף XAML ספציפי מאפשרת להשתמש בו עבור התוכן הספציפי הזה, בעוד שהצבתו בקובץ App.xaml מאפשרת שימוש חוזר בסגנון בכל הדפים והפקדים של היישום.

כדי להשתמש בסגנון מוגדר:

```
< Border Style="{StaticResource borderStyle}">
  <!--Some contents for the border -->
</Border >
```

ניתן להשתמש באותו סגנון פעמים רבות ברחבי היישום (תלוי, כמובן, בטווח ההגדרה). ברגע שתשתנה הגדרת הסגנון, ישתנו, באופן מיידי, כל האובייקטים שעליהם חל הסגנון.

תכונה רבת עוצמה נוספת של מודל התכנות של Silverlight, היא היכולת להתאים אישית באופן מלא את המראה והתחושה של הפקדים שבשימוש. הדבר מאפשר למפתחים ולמעצבים לעצב את ממשק המשתמש של פקדים בדרכים מתוחכמות ודרמטיות, ונותן גמישות עצומה שמאפשרת ליצור בדיוק את חווית המשתמש המבוקשת.

תבנית מאפשרת התאמה אישית מלאה של ממשק המשתמש של פקד – היא מאפשרת למפתחים/מעצבים להגדיר את ממשק המשתמש של הפקד ללא כל שינוי בקוד.

כשאנו יוצרים Control Template, באפשרותנו ליצור Style שיכיל את התבנית הזו, או ליצור Control Template כמשאב עצמאי.

כדי לאפשר נגישות גלובלית יש ליצור את הסגנון/תבנית בקובץ App.XAML (אבל ניתן ליצור אותם גם כמשאב מקומי (ברמת קובץ ה-XAML)).

הגישה המומלצת היא להשתמש ב-Control Templates כחלק מסגנון.

הקוד הבא יוצר לחצן אליפטי (תבנית היא חלק מהסגנון של לחצן):

```
<Style x:Key="roundButtonStyle" TargetType="Button">
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate TargetType="Button">
        <Grid>
          <Ellipse Width="200" Height="200">
            <Ellipse.Fill>
              <RadialGradientBrush GradientOrigin="0.2,0.2">
                <GradientStop Offset="0.2" Color="White"/>
                <GradientStop Offset="1" Color="Green"/>
              </RadialGradientBrush>
            </Ellipse.Fill>
          </Ellipse>
          <ContentPresenter Content="{TemplateBinding Content}"
            FontSize="{TemplateBinding FontSize}"/>
        </Grid>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>
```



כשמחילים זאת על הלחצן:

```
<Button Content="Exit" Width="100" Height="50"
  FontSize="20" Click="Button_Click" Style="{StaticResource roundButtonStyle}"/>
```

גם כשמשתמשים ב-Control Templates, הפקד עדיין שומר את כל הפונקציונליות שהייתה לפקד המקורי (במקרה של לחצן - click, ריחוף עם העכבר וכו').

קשירת נתונים

יכולות קישור הנתונים של Silverlight הן רבות עוצמה. הן מאפשרת חיבור כל מאפיין Dependency לכל מאפיין אחר. אם החיבור הוא בין מאפיינים שקשורים לממשק המשתמש, השינויים יעודכנו מיידית על המסך.

נניח שיש לנו תיבת רשימה שיש לאכלס אותה בנתונים. כיצד עשינו זאת בדרך הישנה? על ידי ביצוע לולאה על פני אוסף/מערך של נתונים והוספת כל אחד מהם כפריט לתיבת הרשימה.

הבה נעשה זאת בדרך של Silverlight!

הוספת תיבת רשימה לדף XAML שלנו:

```
<ListBox x:Name="myList"/>
```

בהנחה שיש לנו רשימה/מערך של אובייקטים נוכל לקשור אותה כ-ItemsSource (בדוגמה זו הדבר מתבצע בקוד ברקע, אבל ניתן לבצעו גם על ידי XAML):

```
public class MyData
{
    public string Name { get; set; }
    public int Age { get; set; }
    public bool IsOk { get; set; }
}
```

הגדר את אוסף המופעים עם הנתונים:

```
IEnumerable<MyData> res = ...;
myList.ItemsSource = res;
```

כעת תיבת הרשימה שלנו תהיה קשורה (מחוברת) למקור החי של הנתונים (בניגוד ל-ADO.NET היכן שקישור הנתונים מתרחש בשרת והלקוח מקבל פלט מרונדר בלבד).

SilverlightApplication4.myData
SilverlightApplication4.myData
SilverlightApplication4.myData
SilverlightApplication4.myData
SilverlightApplication4.myData

כדי לקבוע איזה נתונים מאוסף המחלקות שלנו יוצגו כפריטים בתיבת הרשימה, עלינו להגדיר DataTemplate עבור ListBoxItem (כל שורה בתיבת הרשימה).

```
<ListBox.ItemTemplate>
<DataTemplate>
    <StackPanel Orientation="Horizontal">
        <TextBlock Text="{Binding Name}" FontSize="15" Margin="5,0,10,0"/>
        <TextBlock Text="{Binding Age}" FontSize="15" Foreground="Red"
            Margin="5,0,0,0"/>
    </StackPanel>
</DataTemplate>
</ListBox.ItemTemplate>
```

תיבת הרשימה שלנו תיראה כעת כך:

Alex	34
Josh	33
Anna	38
Bill	45
Nicole	44

סיכום

יישום cross-browser, cross-platform של .NET Framework. לבניית הדור הבא של מערכות אינטרנט בעלות ממשק משתמש עשיר. הטכנולוגיה רצה על כל הדפדפנים הנפוצים כולל IE של מיקרוסופט, Safari של mac, Chrome של google וכמובן Mozilla Firefox.

הרכיב (plugIn), שיש להוריד לדפדפן כדי להריץ Silverlight, הינו בגודל כ-4.5MB וניתן להריצו במהירות ובקלות.

זהו צירוף של מספר טכנולוגיות לתוך תשתית פיתוח, המאפשרת למפתח לבחור ולהשתמש בשפת הפיתוח המועדפת עליו.

ניתן לשלב שימוש ב-Silverlight עם מערכות ASP.NET, תוך מיזוג קוד Javascript ו-Ajax להשלמת התפקודיות ולהעשרת מערכות אינטרנט קיימות.

Index

API	Application Programming Interface
ASP	Active Server Pages
CSS	Cascading Style Sheets
CTS	Common Type System
enum	enumeration
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IIS	Internet Information Services
LINQ	language-integrated query
SQL	Structured Query Language
URL	Uniform Resource Locator
VWD	Visual Web Developer
WSAT	Web Site Administration Tool
XHTML	Extended HTML
XML	Extended Markup Language

A

- <a> element, navigation, 249, 273
- absolute URLs
 - functions of, 253, 246
 - problems with, 253
- abstract classes, 195
- access modifiers, 196–197
 - internal, 197
 - private, 197
 - protected, 197
 - public, 196
- Access Rules, Web Site Administration Tool (WSAT), 572, 587
- AccessDataSource, 408
- accounts
 - ASP.NET accounts, 689
 - IIS account, 689
 - new, creating, 547–554
- Active Server Pages (ASP), 1
- ActiveStepIndex, 126
- Add method, 175
- Add New Item, 41–42
 - base page, 216
 - master pages, 205–207
 - user controls, 284
 - Web Services, 359
- Add Style Rule
 - functions of, 86
 - use, example of, 88–91
- Add Tab, 24–25
- addition, 158
- addition operator, 154–155
- Administrator, AndAlso operator, 161
- AdRotator server control, 115, 129
- advertisements, AdRotator server control, 129
- Ajax. *See* ASP.NET AJAX
- alignment, styles, 504
- allowAnonymous, 590
- AllowCustomErrorsRedirect, 346
- AlternateText, 286
- AlternatingItemTemplate, 476
- AlternatingRowStyle, 505

- AND, SQL, 380
- And operator, 158–159
- AndAlso operator, 160, 168
- anonymous types
 - creating, 463–468
 - functions of, 463
 - genres/reviews, creating list, 465–468
- anonymous users
 - allowAnonymous, 590
 - anonymous identification, enabling, 615–617
 - anonymous profile, deleting, 616–617
- AnonymousTemplate, 556, 558, 564
- API (Application Programming Interface),
 - Cache API, use of, 540–542
- App_Code folder, 176–180
 - AppCconfiguration in, 670–674
 - calculator, optimizing, example, 176–180
 - class files, adding to, 176–177
 - enums in, 295
 - icon for, 176
 - operation of code in, 179–180
- App_Data folder
 - creating, 6
 - Modify permissions, 376
 - new web site, 38–39
- App_Themes folder, 240
 - creating, 227, 229
- AppConfiguration
 - application settings, moving to web.config, 670–674
 - operation of, 673–674
- AppendAllText, 332
- application folders, 251
- Application Mapping, 9–10
- application services, 547–548
 - configure database for, 739–740
 - LocalSqlServer connection, 698, 738
 - providers, 548
 - settings, overriding, 741–743
 - types of, 547
- application settings, moving to web.config, 670–674
- Application_Error, 636, 641
- applicationName, 568
- Apply Styles
 - functions of, 86
 - use, example of, 99–103
- arithmetic operators, 154–156
- array(s), 148–150
 - C#, 149–150
 - and collections, 151–152
 - defining, 148
 - generics, 150
 - instantiating, 149
 - role names, adding, 148–150
 - size, 148
 - upper bound, 149
 - Visual Basic, 149–150
- ArrayList, 151–152
- ASC, SQL, 380
- ASP.NET
 - development of, 1–2
 - IIS, registering in, 683
 - .NET Framework, downloading installing, 683
- ASP.NET AJAX, 340–372
 - AJAX Extensions, 131, 342
 - AJAX Master Page, 42
 - AJAX Web Form, 42
 - benefits of, 369
 - C#, 343
 - flicker-free pages, creating, 347–351
 - functions of, 340, 341
 - ScriptManager, 346–347
 - Timer control, 352–357
 - tips for use, 370
 - UpdatePanel, adding to page, 343–345
 - UpdateProgress, 347–351
 - Visual Basic, 343
 - Web Services, 362–370
- ASP.NET Development Server, icon for, 7–8
- ASP.NET Server Controls, 107–140
 - AdRotator, 129
 - Ajax Extensions, 131
 - ASP.NET state engine, 131–138
 - C#, 119–123, 127
 - Calendar, 128
 - container, 121
 - data, 130

- defining in page, 112
- FileUpload, 128
- functions of, 107–109
- HiddenField, 129
- HTML, 129
- Image, 128
- ImageButton, 128
- ImageMap, 128
- LinkButton, 128
- list, adding to page, 116–121
- Literal, 128
- Localize, 128
- login, 131
- markup, 14
- navigation, 130
- page processing, 10
- panel, 122–123
- properties of, 113–114
- runtime operation of, 111, 120–121
- Smart Tasks menu, 117, 125
- Standard, 115–130
- Standard versus HTML, choosing, 129
- Substitute, 128
- Table, 129
- text box, adding to page, 109–112, 113–114
- tips for use, 138
- in Toolbox, 115–116
- use, example of, 109–111
- validation, 130
- visibility on server, 123
- Visual Basic, 119, 121, 123, 126
- Wizard, 123–128
- XML, 129
- ASP.NET state engine, 131–138
 - functions of, 133
 - operation of, 133–137
- ASP.NET Web Service template, 37
- ASP.NET web site
 - ASP.NET AJAX, 340–372
 - ASP.NET Server Controls, 107–140
 - Cascading Style Sheets (CSS), 66–106
 - database, 375–402
 - debugging, 641–656
 - deploying, 666–700
 - exception handling, 624–641
 - existing files, adding to site, 45–46
 - file types, 41–49
 - first site, building, 5–9
 - markup, 56–62
 - new site, creating, 38–39
 - page processing at runtime, 9–10
 - pages, consistency among. *See* web page(s); specific types of pages
 - personalizing, Profile, 589–590
 - programming web pages, 141–201
 - security, 545–586
 - styles, 502–511
 - template for, 36–37
 - tracing, 656–663
 - user controls, 282–307
 - Web Forms, 48–55
 - See also* Planet Wrox
- ASP.NET Web Site option, 38
- .aspx files. *See* Web Forms
- ASPX page
 - markup in, 8
 - time/date in, 8–9
 - See also* web page(s)
- assignment operators, 154
- AssociatedControlID, 601
- AsyncPostBack, 354
- AsyncPostBackErrorMessage, 346
- AsyncPostBackTrigger, 346
- AttachDbFileName, 420
- attachment, e-mail, 326
- attributes
 - ASP.NET, 14
 - functions of, 358
 - HTML, 12
 - Web Services, 359
 - XHTML, 13
- authentication
 - forms-based, 554
 - purpose of, 546
 - setting, 547
 - SQL Server, 724–726
 - timeout, 554
 - Windows, 724–726
- authorization, purpose of, 546–547
- Auto mode, Style Application, 59
- AutoEventWireup, 286

automatic properties, 189

AutoPostBack, 117, 122

Average operator, 460

B

background color

Cascading Style Sheets (CSS), 76, 79, 80–81

choosing, 76, 80–81

server control for, 112

styles BackColor, 504

backing variables, 187–188

banner(s)

enums for user controls, 295–299

properties, adding to, 293–299

user control, adding to master page, 287–290

user control, creating, 284–287

user control, registering in web.config, 290–291

base page

C#, 216–220

creating, 216–220

functions of, 213

inheritance, 214, 219–220

inheritance from, 214

Visual Basic, 216–220

BETWEEN, SQL, 380

Block Format, 56–57

Block Selection, 95

BodyFileName, 561–563

boldface, adding, 58–60

bookmark window, contents of, 22

Boolean, Convert.ToBoolean, 674

border(s)

images, 290

Cascading Style Sheets (CSS), 76–77

color, BorderColor, 504

padding, 93–95

server controls for, 112

styles, BorderStyles, 504

width, BorderWidth, 504

BoundField, 427

breadcrumbs

creating, 272–274

functions of, 22

SiteMapPath server control, 130

breakpoints

Breakpoints window, 648

debugging, example of, 641–644

browser(s)

alternate, opening page in, 92–93

ASPX page processing, 9–10

default documents, 253–254

DOCTYPE and page rendering, 14

finding unlisted browsers, 92–93

history, deleting, 92–93

HTTP, 9

server controls, actions of, 112

web page, opening, 7–8

Build Style, 80, 90

built-in web servers, accounts/permissions, 688–689

bulleted lists

BulletedList server control, 116, 464

bullets, adding to list, 61–62

ordered and unordered, 61–62

button(s)

ButtonField, 427

ButtonType, 560

CreateUserWizard properties, 560

send button, on contact form, 312–313

server controls for, 116, 119, 128

server controls, use of, 109–112

skins, 240–244

Timer control, 353–354

By Genre, 267

ByRef keyword, 175

ByVal keyword, 175

C

caching, 530–543

C#, 533, 535–541

cache dependency, 531

with data source controls, 538–539

events of, 530–531

file contents, inserting in cache, 541–543

functions of, 530

invalidating cache, 531

operation of, 537–538

output caching, 532–536

pitfalls related to, 531–537

programmatic caching, 540–541

- retrieval from cache, example of, 531–532
- Substitute server control, 128
- Visual Basic, 531, 535–541
- calculator
 - App_Code folder to optimize, 176–180
 - creating, programming example, 165–169
 - debugging, 642–645, 649–652
 - namespaces, use of, 181–182
 - object-oriented programming, 186, 191
- calendar
 - calendar server control, 116, 128
 - Set Date, 134–135
 - state engine, use of, 131–134
- Call Stack window, 648
- CancelButtonText, 560
- Cascading Style Sheets (CSS), 66–106
 - ASPX page, writing, 68–72
 - cached copy, use of, 68–69
 - cascading, meaning of, 86
 - classes, 72
 - color, 76, 78, 97
 - control adapters, 279
 - Css Class property, 259, 264–266, 268, 504, 508
 - declarations, 72, 73, 81, 89–90
 - embedded style sheets, 85–87
 - external style sheets, benefits of, 85
 - file extension (.css), 42
 - formatting, benefits over HTML, 68
 - Friendly Control Adapters, 260
 - and HTML attributes, 12
 - ID selectors, 292
 - images, referring to, 231–233
 - inline style sheets, 86
 - IntelliSense, use with, 79
 - operation at run-time, 71–72, 83–84
 - page, viewing in browser, 71
 - pixel size and screen, 84
 - Planet Wrox exercise, 69–72, 79–83
 - properties, 72, 76–77
 - Properties Grid. *See* Cascading Style Sheets (CSS) Properties Grid
 - rules, 71–72, 81–82, 89, 104
 - saving, 83, 92
 - selectors, 72, 73–75, 87–90
 - server controls for, 112, 115
 - shorthand version, 78–79
 - style sheets, 73–84
 - styles, adding to page, 85–87
 - syntax, 69–71
 - for themes, 225
 - values, 72, 77–78
 - versions of, 69
 - Visual Web Developer (VWD), use of. *See* Cascading Style Sheets (CSS) in VWD
- Cascading Style Sheets (CSS) in VWD, 87–105
 - embedded style sheets, 93–99
 - inline style sheets, 93–99
 - managing styles, 101–103
 - new styles in existing style sheet, 88–91
 - Planet Wrox exercise, 88–103
 - style sheet, attaching to document, 91–93
 - styles, applying to page, 99–101
 - tips for use, 104
 - tools, 87
- Cascading Style Sheets (CSS) Properties Grid
 - categorized/alphabetical mode, switching, 98
 - clear styles, 100
 - color, choosing, 97–98
 - functions of, 87, 104
 - use, example of, 97–98
- case sensitivity, passwords, 550
- case statements, 168
- casting, data type conversion, 146–147
- Catch block, 627–629, 633–634
- CausesValidation, 487
- ChangePassword, 564–566
- CheckBox, server controls, 122, 127, 137
- CheckBoxField, 427
- CheckBoxList
 - Profiles, 597–606
 - server controls, 116, 119, 120
- ChildrenAsTriggers, 345

- class(es)
 - abstract, 195
 - Cascading Style Sheets (CSS), 72
 - inheritance, 193–196
 - in object-oriented programming, 186–187
- class files
 - file extension (.cs/.vb), 44
 - functions of, 44
 - warning when adding, 48
- Class selector, Cascading Style Sheets (CSS), 75
- ClearStyles, 100
- Click handler, 354
- client ID
 - master page problems, 212–213
 - user controls and modification of, 291–292
- client-side script, debugging, 652–656
- client-side validation, 311
 - creating, 320–325
- code
 - App_Code folder, 176–180
 - bookmark, adding to files, 22
 - breadcrumb in, 22
 - Code Behind files, 10, 51–55
 - comments, 182–185
 - formatting symbols, 7
 - functions, 173–176
 - inline code, 51–55
 - IntelliSense, 52–53
 - namespaces, 180–182
 - object oriented. *See* object-oriented programming
 - placeholders, 166
 - runtime, 10
 - special characters, automatic replacement of, 59
 - subroutines, 173–176
 - Visual Basic compared to C#, 54
 - See also* programming web pages
- Code Behind files, 51–55
 - calculator example, 168, 176–180
 - file extension (.aspx), 44, 51
 - compared to inline code, 51–55
 - master pages, 203, 205–207
 - namespaces, viewing, 180
 - NavigateUrl property, creating, 299–302
 - new web site, 38–39
 - templates, creating, 221–224
 - TreeView, 270–271
 - types of, 38
 - user controls, adding logic to, 293, 295–299
 - Web Services, 359–361
- code snippets, dragging to Toolbox, 23–24
- Code View. *See* Markup View
- Code Window, 53
- CollapseImageToolTip, 269
- CollapseImageUrl, 269
- collections, 151–152
 - C#, 151–152
 - functions of, 151–152
 - limitations of, 151–152
 - Visual Basic, 151–152
- color
 - borders, 504
 - Cascading Style Sheets (CSS), 76, 78, 96–97
 - choosing, 97–98
 - styles, 504
 - See also* background color; font color; foreground color
- columns
 - GridView columns, customizing, 421–427
 - identity columns, 392, 394–397
- Command(s), 27
- CommandField, 414, 427
- comments, 182–185
 - inline comments, 182–184
 - syntax for, 182–184
 - XML comments, 184–185
- Common Type System (CTS), data types, listing of, 144–146
- CompareValidator, 316, 318, 321, 599, 601, 635
- comparison operators, 156–158
- comparison operators, of WHERE clause, 379

- compile errors, 625–626
 - display of, 626
- compiler(s), functions of, 625
- concatenation operators, 158–159
- configuration file
 - file extension (.config), 42
 - functions of, 42
 - machine.config, 569–571
 - new web site, 38–39
 - See also* web.config
- configuration settings, 567–571
 - attributes, 567–568
 - membership member, modifying, 569–571
 - operation of, 569–571
 - System.Configuration, 669–670
- ConfigurationManager, values, retrieving from, 669–670
- ConfigureDataSource, 440–442
- confirmation e-mail, 561–563
- connection strings
 - ConnectionString, 413
 - ConnectionStringName, 567
 - default, 567
 - functions of, 378
 - LocalSqlServer, overriding, 741–743
 - new account, creating, 552–553
 - SQL Server as target, 553
 - storing, 418–420
 - web site resource on, 698, 725
- constructors
 - default, 191–192
 - object-oriented programming, 191–193
 - overloaded, 192
- contact forms
 - client/server-side validation, creating, 321–325
 - creating, 312–313
 - extending operations on, 317–320
 - UpdateProgress, 347–351
 - See also* validation controls
- container(s)
 - naming container, 292
 - server controls for, 121
- content pages, 208–210
 - adding to web site, 208–210
 - contents of, 208
 - placeholders to master page, 203–207, 211–212
 - user controls, adding to, 287
- ContentPlaceHolders, master pages, 203–207, 211–213
- ContentTemplate, 345
- ContextTypeName, 469
- ContinueDestinationPageUrl, 560
- control(s)
 - data source controls, 408
 - data-bound controls, 406–408
 - event handler, writing for, 518–521
 - ID, setting, 166
 - login controls, 548–563
 - properties, binding to resources, 638–639
 - server. *See* ASP.NET Server Controls
 - validation controls, 310–325
 - See also* user controls
- control adapters
 - Cascading Style Sheets (CSS), 279
 - Friendly Control Adapters, 260
- ControlParameter, 420
- ControlToCompare, 316
- ControlToValidate, 314, 439
- Convert, data type conversion, 146
- Convert to Hyperlink, 63, 91
- Convert.ToBoolean, 674
- cookies
 - CookieParameter, 420
 - expiration, 237
 - safety factors, 233–234
 - themes, storing in, 233–234, 237–240
 - Value property, 237
- Copy property, 332
- copying web site, 674–677
 - Copy Web Site, use of, 675–676
 - operation of, 678
- Count operator, 460
- Create Custom Content, 212
- Create User, 523, 544
- CreateUserButtonImageUrl, 560
- CreateUserText, 549, 555
- CreateUserUrl, 549, 555
- CreateUserWizard, 560–561

- confirmation e-mail, sending with, 561–563
- new account, creating, 549–552
- operation of, 564
- properties of, 560
- CssClass property, 259, 264–266, 268
- styles option, 504, 508
- CType, 147
- Current Page, 103
- custom error pages, 635–636, 638
- Customize, 27
- CustomValidator, client/server-side validation, creating, 321–325

D

- data controls
 - data server controls, 130
 - data source controls, 408
 - data-bound controls, 406–408
- data source controls
 - caching with, 538–539
 - error handling, 525–530
 - types of, 408
- data store, web applications, 547–548
- data tips, 645
- data types, 142
 - Common Type System (CTS), 144–146
 - converting, 146–148
 - enumeration, creating with, 293–299
 - and variables, 142–146
- database(s), 373–404, 410–415
 - accessing within VWD, 376–378
 - application services, configure for, 739–740
 - with ASP.NET. *See* SQL Server
 - cache dependency, 531–532
 - connecting to, 376–378
 - connection strings, 378
 - connection strings, storing, 418–420
 - data, creating, 389
 - data server controls, 130
 - data source controls, 408
 - Database Explorer, 17, 376–378
 - Database Publishing Wizard, 695–697
 - data-bound controls, 406–408
 - deleting data, 389

- DetailsView, creating/customizing, 434–447
- DetailsView, inserting data, 415–418
- exceptions related to, 633
- exporting, 695–697
- fields, types of, 427
- filtering data, 379–380, 420–426
- functions of, 374
- GridView, data-driven web page, creating, 409–415
- GridView columns, customizing, 427–433
- identity columns, 394–397
- joining data, 385–389
- LINQ to SQL. *See* LINQ
- ordering data, 381
- parameters, 412–413, 420
- primary keys, 394
- queries, creating, 381–385
- Query Designer, 381–385
- reading data, 378–379
- records, creating/using, 390–392
- recreating, 698–699
- registration of, 377–378
- relational, 374–375
- SQL Server, attaching to, 728–730
- SqlDataSource controls, 409–415
- Table Designer, use of, 394–398
- tables, creating, 394–398
- tables, relationship between, 399–402
- tips for use, 402, 448
- updating data, 389, 433–447
- DataBind, 468
- data-bound controls, 406–408
 - columns/fields, configuring, 427–433
 - list controls, 406–407
 - paging controls, 408
 - single item controls, 407
 - styles, 504–512
- DataContext, 456–457
- DataList, styles for, 504–505
- DataNavigateUrlFields, 429, 432
- DataNavigateUrlFormatString, 429, 432
- DataPager control, 492–498
 - functions of, 469, 492
 - ListView, hook up with, 492

- photo album, front-end page, creating, 492–498
- DataSource, 465, 468
- DataSourceID, 259, 414, 417, 477
- DataTextField, 429, 465
- date. *See* time/date
- date-binding syntax, 418
- debugging, 641–655
 - Autos window, 647
 - breakpoints, 641–644
 - Breakpoints window, 648
 - calculator, example of, 642–645
 - Call Stack window, 648
 - Debug menu, 15
 - disabling in web.config, 699
 - extensive, example of, 649–652
 - Immediate window, 648
 - JavaScript in Internet Explorer, 653–655
 - keyboard shortcuts, 644–646
 - Locals window, 648
 - operation of, 644–645, 652–653
 - tips for, 663–664
 - Watch window, 646–647
- decision making, logical constructions, 161–164
- declarations, Cascading Style Sheets (CSS), 72, 73, 81, 89
- default constructors, 191–192
- default documents, 253–254
- Default.aspx
 - code-behind file, 38–39
 - creating, 5–6
 - default document, 253–254
 - link to Login.aspx, 249–250
 - opening, 24–25, 83
- defaultValue, 591
- delete
 - Delete property, 332
 - on GridView, 522–523
 - rows, error handling, 525–530
 - users, 574
- DELETE, SQL, 389
- deploying web site, 666–697
 - C#, 667, 669–673
 - copying site, 674–677

- database, recreating, 698–699
- deployment targets, options, 674
- hardcoded settings, moving to web.config, 667–674
- IIS, 679–688
- NTFS settings, 689–695
- publishing web site, 678–679
- remote server, transfer data to, 696–698
- retail attribute, true setting, 700
- server errors, types of, 694–697
- tips for, 699–700
- Visual Basic, 667, 669–673
- DESC, SQL, 381
- Design View
 - Cascading Style Sheets (CSS), 70–71, 82–83, 91
 - content pages, 209–210
 - functions of, 49–50
 - markup, adding to page, 56–59
 - master pages in, 206–207
 - menu creation, 260–262
 - opening, 18, 19–20
 - updating, 19–20
 - user controls, creating, 287–290
 - viewing with Markup View, 50
- DestinationPageUrl, 555, 556
- DetailsView
 - creating, 434–438
 - customizing, 438–445
 - inserting data with, 415–418
 - operation of, 445–447
 - settings, 415
- DirectCast, 147
- DisableCreatedUser, 561
- DisappearAfter, 259
- Display property
 - Cascading Style Sheets (CSS), 76
 - validation control, 314
- DisplayAfter, 347
- DisplayDirection, 295–303
- DisplayMode, 465
- DisplayRememberMe, 555
- DivideByZeroException, 634
- division operator, 154–156
- Do While loop, 172

- docking windows
 - changing to floating, 23
 - process of, 23
- DOCTYPE, functions of, 14
- Document Window
 - documents, switching between, 18
 - functions of, 18
 - modifying, 26
 - tab size, changing, 26
- Documents folder, 5-6
- DropDownList server control, 116-117, 120-121, 125, 137
 - filtering data, 420-426
- dynamic files, 621
- DynamicEnableDefaultPopOutImage, 259
- DynamicHoverStyle, 266
- DynamicLayout, 347
- DynamicMenuItemStyle, 266

E

- EditItemTemplate, 476
- EditRowStyle, 505
- elements
 - ElementID, 74
 - HTML, 10-11
 - ID, 88-89
 - XHTML, 13
- Elseif statement, 162
- e-mail, 325-339
 - C#, 327-336
 - confirmation e-mail, sending, 561-563
 - contact form validation, 317-320
 - CreateUserWizard, use of, 561-562
 - exceptions related to, 631
 - hardcoded settings, 667
 - MailAddress, 326, 328-331
 - MailDefinition, 560, 564
 - MailMessage, 326-331
 - operation of, 329-331, 335-336
 - RequireMail, 560
 - RequiresUniqueEmail, 568
 - sending, code for, 328-331
 - sending from contact form, code for, 333-337
 - SendMailOnError, 670-673
 - text files, working with, 332-333
 - Visual Basic, 327-337

- web site, configuring for, 326-327
- embedded style sheets
 - creating, 93-99
 - elements of, 85-86
 - moving between files, 103
- empty web site template, 37
- EmptyDataRowStyle, 505
- EmptyDataTemplate, 476
- EmptyItemTemplate, 476
- Enable
 - Timer control, 352
 - trace, 623
- Enable AutoPostBack, 235
- Enable Intranet Settings, 7
- EnableClientScript, 314
- EnableDelete, 469
- EnableInsert, 469
- EnablePageMethods, 346
- EnablePasswordReset, 567
- EnablePasswordRetrieval, 567
- EnableTheming, 226
- EnableUpdate, 469
- EnableViewState, 137, 352
- encapsulation, object-oriented
 - programming, 196
- encrypted format, passwords, 567
- EndRequest, 597
- enumeration (enum)
 - C#, 293-299
 - example of, 293-294
 - smart user controls with, 293-299
 - Visual Basic, 293-299
- equivalence, operators for, 156-158
- error(s)
 - catching. *See* error handling; exception handling
 - error messages, 627
 - logic errors, 626-627
 - server errors, 694-695
 - syntax errors, 625-626
- error handling, 525-530
 - C#, 527-530
 - error logging, enabling, 700
 - GridView, deleting rows, 525-530
 - operation of, 529-530
 - Visual Basic, 527-530

- Error List, 626, 675
 - contents of, 22
 - docking, 24–25
- ErrorCallback, 368
- ErrorMessage property, compared to Text property, 314–315
- event(s)
 - argument class, 197
 - object-oriented programming, 197–198
 - page life cycle, events of, 215–216, 512–513
- event handling, 518–530
 - C#, 519–525
 - delete, enable on GridView, 522–524
 - error handling, 525–530
 - object-oriented programming, 197–198
 - operation of, 519–524
 - RowDataBound, writing for, 518–521
 - with strongly typed objects, 521–524
 - Visual Basic, 519–525
- Exception, 523, 529
- exception handling, 625–640
 - C#, 626, 628–632, 634, 637–638
 - catching exceptions, 627–628
 - custom error pages, 635–636
 - global, 636–641
 - programming errors, types of, 625–627
 - Try Catch, use of, 630–635
 - Visual Basic, 625–626, 628–634, 637–638
- ExceptionHandled, 525, 527
- Exists property, 332
- ExpandDepth, 269
- ExpandImageToolTip, 269
- ExpandImageUrl, 269
- Expired, 237
- exporting
 - Import and Export Settings, functions of, 28
 - templates, 222, 224
- expression binding syntax
 - Expression Editor, 668–669
 - functions of, 418, 699
 - use of, 699–700
- external style sheets
 - benefits of, 85

- moving between files, 103

F

- favorites
 - FavoriteGenres, Profile, using, 597, 598, 602–605, 610
 - genre preferences, storing, 597–606
- fields, types of, 427
- file(s)
 - automatic copy, 46
 - class, 44
 - data files, 44–45
 - existing, adding to site, 45
 - Global Application Class, 44
 - new, adding to site, 41–43
 - organizing, 46–48
 - project file, 35
 - Solution Explorer, 17
 - storing in folders, 17–18
 - Web files, 41–44
 - Web Forms, 48–55
 - Web Services, 44
- File class, properties of, 332–333
- file extensions
 - file types/extensions, listing of, 42–44
 - hidden, exposing, 45
- File System, 5, 38–40
 - deployment option, 674
- FileUpload server control, 116, 128
- filtering data, 379–380, 420–426
 - applying filter, 424–426
 - filter, setting up, 420–424
 - operation of, 424–426
 - values, listing of, 420
 - WHERE clause, 379–380
- Finally, 594–595
- Find and replace, base page creation, 218
- Find in Files, starting, 22
- Find Results, window, 22
- FindByValue, 238
- FinishButtonClick, 127
- First query operator, 462
- FirstOrDefault query operator, 462
- flicker-free pages. *See* ASP.NET AJAX
- float
 - Cascading Style Sheets (CSS), 81–84
 - restoring window, 24

- sidebar, 84
- windows, creating, 22–24
- folders
 - App_Code folder, 176–180
 - App_Data folder, 38–39
 - App_Themes folder, 225, 227, 229, 240
 - application folders, 251
 - creating, 38
 - Documents folder, 5
 - Management folder, 249–250, 572–580
 - New Folder, 51
 - organizing site files, 46–48
 - settings folder, location of, 29
 - special files, warning, 48
 - for themes, 225, 227, 229
- font(s)
 - Cascading Style Sheets (CSS), 70, 73–74, 77, 79, 81
 - choosing, 56, 74, 77, 81
 - Font Name, 56
 - Font Size, 56
 - server controls for, 112
 - styles, 96, 504
- font color
 - Cascading Style Sheets (CSS), 77
 - Cascading Style Sheets (CSS) in VWD, 89–90
 - choosing, 77, 80, 89–90
- footers
 - floating, 84
 - styles, FooterStyle, 505
- For Each/foreach loops, 120–121, 171
- For loop, 169–170
- foreground color
 - choosing, 57–58
 - server control for, 112
 - styles, ForeColor, 504
- foreign keys, 399–400
- form(s). *See* contact forms; Web Forms
- formatting
 - Block Format, 56–57
 - controls, with styles, 502–512
 - HTML, problems of, 67
 - tags, adding, 59
 - text, 59
 - See also* Cascading Style Sheets (CSS)

- Formatting toolbar
 - Cascading Style Sheets (CSS), 70, 91
 - Convert to Hyperlink, 91
- FormParameter, 420
- FROM, SQL, 378
- From query operator, 458–460
- Front Page Extensions, 38–39
- FTP
 - deployment option, 674
 - open site from, 38–39
- functions, 173–176
 - parameterize, 173–176
 - uses of, 173, 180
- G**
- GeneralFailureText, 533
- generics, 150, 152–153
 - C#, 152–153
 - functions of, 152–153
 - use, example of, 152–153
 - Visual Basic, 152–153
- genre(s)
 - database table, 378, 380–385, 399
 - FavoriteGenres
 - Genre property, 457
 - preferences, storing in Profile, 597–606
 - type interference, 457
- GET
 - data acquisition by, 309, 669
 - forms, submitting, 135
- GetProfile, 588
- Global Application Class
 - file extension (.aspx), 44
 - functions of, 44
- global error handling, 636–642
- Go to Homepage, 63
- greater than, operators for, 156–158
- GridView, 409–415
 - columns, customizing, 421–427
 - control event handler, writing for, 517–521
 - controls, styles applied to, 504–507
 - data-driven web page, creating, 409–415
 - delete on, 522–525
 - page life cycle, events of, 512–513

- RowDataBound, event handler, 517–521
- styles combined with skins/themes, 508–512
- grouping, selectors, 76
- GroupSeparatorTemplate, 476
- GroupTemplate, 476
- H**
- Handler Mapping, 9–10
- hardcoded settings, moving to web.config, 667–674
- hashed format, passwords, 567, 568
- headers
 - choosing headings, 57–58
 - HeaderText, 429
 - styles, HeaderStyle, 505
- height of elements
 - Cascading Style Sheets (CSS), 77, 81–82
 - server controls for, 113
 - styles, 504
- help
 - help files, installing, 4
 - Help menus, 15
 - Start Page topics, 18
- HiddenField server control, 116, 129
- history, deleting, 92–93
- homepage
 - go to, 63
 - styling, 79–84
- HorizontalAlign, 504
- hotspots, on images, 128
- hovers
 - Cascading Style Sheets (CSS), 89
 - creating, 89
 - DynamicHoverStyle, 266
 - StaticHoverStyle, 264–266, 269
 - tooltips, displaying, 269
- HTML, 10–14
 - adding to site. *See* markup
 - ASPX page markup, 9–10
 - attributes, 12
 - code equivalents, auto insert, 59
 - elements, 10–11
 - file extension (.htm/.html), 42
 - formatting, limitations of, 67

- HTML page, 42
- server controls, 130
- with server-side code, avoiding, 108
- tags, 10–12
- viewing with View Source, 21, 135
- compared to XHTML, 13–14
- HTTP
 - functions of, 9
 - page moved, status code, 277
- Hungarian notation, avoiding, 144
- Hyperlink
 - server control, 116
 - server-side, 249
 - See also* linking pages
- HyperText Markup Language (HTML).
 - See* HTML
- HyperText Transfer Protocol (HTTP).
 - See* HTTP
- I**
- ID
 - client ID, 212–213, 291–292
 - controls, setting, 166
 - DataSourceID, 259, 414, 417, 477
 - ElementID, 74
 - ItemPlaceholderID, 476
 - selector, Cascading Style Sheets (CSS), 73–75
 - SkinID, 243
- identity
 - elements of, 546
 - security aspects, 546–547
- identity columns, SQL Server, 3, 392, 394–397
- If Else statement, 162–164
- If statement, 161–164, 168
- IIS (Internet Information Services), 679–688
 - ASP.NET registering, 683
 - deployment option, 674
 - functions of, 37–38
 - installing/configuring, 680–682
 - permissions to account, 689
 - release site to, 694
 - Windows Server 2008, 681–682
 - Windows Server 2008 configuration, 683–688

- Windows Vista configuration, 683–688
 - Windows Vista install, 681
 - Windows XP/Server 2003 configuration, 683–686
 - Windows XP/Windows Server 2003 install, 680–681
 - image(s)
 - adding to themes, 232–233
 - AlternateText, 286
 - banners, user controls, 284–299
 - borders around, 290
 - Cascading Style Sheets (CSS), 231–233
 - ImageButton, 116, 128
 - ImageField, 427
 - ImageMap, 116, 128
 - server controls for, 116, 128
 - Immediate window, 648
 - importing
 - Import and Export Settings, functions of, 28
 - namespaces, 182
 - Indent Size, 26
 - Index.aspx, default document, 254
 - inheritance, 193–196
 - from base page, 214, 219–220
 - creating, 196
 - hierarchy of classes with, 194–195
 - Inherits keyword, 180–182, 195
 - initialization, web pages, 215
 - inline code, 51–55
 - compared to code behind files, 51–55
 - file extension (.aspx), 51
 - inline comments, 182–184
 - operation at run-time, 54–55
 - inline style sheets
 - creating, 93–99
 - elements of, 87
 - Inline Style, 102–103
 - moving, 103
 - INSERT, SQL, 379, 389
 - InsertItemPosition, 477
 - InsertItemTemplate, 439–445, 476
 - InsertTable, 60–61
 - instance
 - of calculator, 179–180
 - constructors, 191–192
 - instance name, 419
 - named and unnamed, SQL Server, 725
 - instantiation, arrays, 149
 - integrated development environment (IDE)
 - functions of, 15
 - See also* Visual Web Developer (VWD)
 - IntelliSense
 - with enums, 293, 297
 - functions of, 52
 - with UpdatePanel, 345
 - use, example of, 52–53, 79, 112
 - and Web Services, 367–368
 - internal access modifiers, 196
 - Internet Explorer, JavaScript, debugging in, 653–655
 - Internet Information Services (IIS). *See* IIS (Internet Information Services)
 - Interval, Timer control, 352
 - Intranet settings, enabling, 7
 - invalidating cache, 532
 - Is keyword, 158
 - IsUserInRole, Roles, checking with, 583–585
 - IsValid property, 315, 324–325
 - italics, adding, 56, 59, 61, 96
 - Item Template, 222
 - ItemDeleted, 434
 - ItemDeleting, 434
 - ItemInserted, 434
 - ItemInserting, 434
 - ItemPlaceholderID, 476
 - ItemSeparatorTemplate, 476
 - ItemTemplate, 476
 - ItemUpdated, 434
 - iteration
 - collections, 151–152
 - See also* loops
- ## J
- JavaScript
 - client-side validation, 310, 322–324
 - debugging in Internet Explorer, 653–655
 - file extension (.js), 43
 - JScript File, 43
 - menu items, visibility of, 263

- and Web Services, 359, 363–364, 367–368
- joining
 - database data, 385–389
 - JOIN, SQL, 385–389

K

- key(s)
 - access key, setting, 113
 - primary keys, database, 394
- keyboard shortcuts
 - customizing, 27
 - debugging, 644, 645–646
 - displaying, 27
- keywords
 - ByRef, 175
 - ByVal, 175
 - inherits, 180–182, 195
 - New, 191
 - Public, 174, 196
 - Return, 175
 - With, 193

L

- Label server controls, 109–112, 116
- Language, 5, 39
- language translation, Localize server control, 128
- language-integrated query. *See* LINQ
- Last query operator, 462
- LastName, 597
- LastOrDefault query operator, 462
- LayoutTemplate, 475
- less than, operators for, 156–158
- life cycle of page, events of, 215–216, 512–517
- LIKE, SQL, 380
- LINQ to SQL
 - anonymous types, creating, 463–468
 - C#, 461–466
 - genres/reviews, creating list, 465–468
 - query operators, 458–460
 - Visual Basic, 458–464, 487–492
- line continuation character, 54
- LinkButton server control, 116, 128, 560
- linking pages, 62–64
 - Convert to Hyperlink, 63, 91

- Hyperlink, 249
- Hyperlink server control, 116
- HyperLinkField, 427, 429, 432
- link to external pages, 63
- link within site, 63
- links, adding to page, 63
- methods for, 62
- visited links, styling, 92–93

LINQ

- functions of, 450–451
- LINQ to Objects, 452
- LINQ to XML, 452
- query, syntax/structure of, 451

LINQ to SQL, 452–500

- data source control. *See* LinqDataSource control
- DataPager control, 492–497
- dragging and dropping, 452–453
- ListView control, 475–492
- operation of, 456–458
- query, execution of, 456–457
- to SQL Classes. *See* LINQ to SQL Classes
- tips for use, 498

LINQ to SQL Classes

- adding, 453–458
- file extension (.dbml), 44
- functions of, 44

LinqDataSource control, 469–475

- functions of, 469
- operation of, 474
- photo album, creating, 470–473
- in Profiles, 602–604
- properties of, 470

list(s)

- bulleted. *See* bulleted lists
- creating in LINQ to SQL, 465–468
- list controls, 406–407
- ListBox, 116–117
- ListItems, 120–121
- server controls for, 116–121

ListView control, 475–492

- DataPager, use of, 493–498
- functions of, 469, 475
- operation of, 458–512

- photo album, inserting/deleting items, 477–486
- photo album, uploading images by user, 486–492
- properties of, 476–477
- templates, 475–476
- templates, customizing, 486–492
- Literal server control, 116, 128, 638
- loading page
 - events of, 215
 - Page_Load, 328–329, 600
- Localize server control, 116, 128
- LocalOnly, 658
- LocalSqlServer, 567, 698
 - connection string, overriding, 740–741
 - connection to application services, 698, 738
- Location, 5-6, 38–39
- logic errors, 625–626
- logical operators, 159–161
 - If/If Else/Elseif, 161–164
 - SQL, 380
- login control(s), 548–561
 - confirmation e-mail, 560–561
 - CreateUserWizard, 561–562
 - first time log in, 551–552
 - Login control, 548–551, 555–556
 - and Master Page, modifying, 557–560
 - operation of, 552–554
 - password recovery, 563–566
 - Remember Me Next Time, 554
 - server controls, 130
 - sign up pages, 548–550, 552–553
 - styles, 555–556
- Login control, 548–551, 555–556
 - adding to page, 548–551
 - functions of, 555–556
 - LoginCreatedUser, 551, 561
- LoggedIn, 556
 - LoginName, 557–560
 - LoginStatus, 548, 557, 557
- LoggedInTemplate, 527, 558
 - LoginText, 556
 - LoginView, 556–557
 - operation of, 560
 - properties of, 555

- Login.aspx
 - controls. *See* login control(s)
 - creating page, 209–210
 - link to Default.aspx, 249–250
- LoginView
 - login page, modifying, 557–560
 - templates, 556, 558
- logout
 - Logout link, 560
 - LogoutAction, 557
- LoggedOut, 557
- LoggingOut, 557
 - LogoutText, 557
- "look around" link, 92–93
- loops, 169–173
 - Do While, 172
 - For Each/foreach, 171
 - For, 169–170
 - Until, 171–173
 - While, 171–173

M

- machine.config
 - membership element, modifying, 569–571
 - membership services, settings, 567
- mail. *See* e-mail
- main menu, Visual Web Developer (VWD), 15-16
- Manage Styles
 - functions of, 86, 92–93
 - use, example of, 91–93, 101–103
- Management folder
 - allowing access, Managers role, 572–576
 - blocking access, 546–550
 - Login.aspx, link to Default.aspx, 249–250
 - root-based relative URLs, 250
- Manager(s)
 - photo album, management of, 611–615
 - role, creating/assigning, 572–576
- Manual mode, Style Application, 59
- margins, Cascading Style Sheets (CSS), 77, 79
- markup, 56–62
 - ASP.NET Server Controls, 14

- in ASPX page, 8
- Design View, adding in, 56–59
- formatting text, 58–59
- inserting text, 56–57
- Markup View, adding in. *See* Markup View
- tables, adding, 59–62
- Markup View
 - Cascading Style Sheets (CSS), 69, 79–84, 91
 - markup, adding to page, 56
 - markup, updating, 21
 - master pages in, 211–212
 - menu creation, 260–261
 - opening, 18, 49
 - Properties Grid in, 113
 - user controls, creating, 288
 - utility of, 49
 - viewing with Design View, 49–50
- master pages, 203–207
 - adding to site, 206
 - C#, 203–205
 - Code Behind files, 203–204, 205–206
 - content of, 203
 - with content placeholder, 203–207, 211–212
 - creating, 205–206
 - with default content, 211–212
 - file extension (.master), 42
 - functions of, 42, 203
 - Login control, modifying, 557–560
 - in Markup View, 211–212
 - nesting, 212
 - operation at run-time, 203–204
 - user controls, adding to, 287–290
 - Visual Basic, 203–205
 - warning about, 212–213
- Max operator, 460
- MaximumDynamicDisplayLevels, 259
- MaximumValue, 316
- maxInvalidPasswordAttemptst, 568
- membership
 - Database Role Membership, 733
 - element, modifying, 569–571
 - functions of, 548
 - provider, SQL Server, 548
 - services, settings for, 567
- menu(s)
 - creating, 260–263
 - MenuWrapper, 84, 260–261
 - operation of, 263
 - server controls for, 130
 - style properties, applying, 264–268
 - submenu visibility, 263
- Menu control, 259–268
 - functions of, 255
 - properties of, 259
 - shortcomings of, 260
 - Static properties, 260
 - styling, 264–268
- Messages, error list, 22
- methods
 - functions, 173–176
 - method signature, 174
 - object-oriented programming, 146, 190–191
 - subroutines, 173–176
- Min operator, 460
- MinimumValue, 316
- MinRequiredPasswordLength, 568, 569–570
- Modify permissions, 376
- Modify Style, 80–83, 90
- modulus operator, 156
- mostRecent, 659
- Move property, 332
- MSDN Library, installing, 4
- multiplication operator, 154–156
- MultiView server control, 116, 121
- My Templates, 36
- MyBasePage.zip, 222
- N**
 - name attribute, 590
 - name collisions, avoiding, 180
 - named instance, 725
 - named skins, 243–244
 - NameService, 360
 - namespaces, 180–182
 - creating, 180–182
 - functions of, 180–182
 - importing, 182
 - viewing, 180

- naming container, 292
- NavigateUrl, 249
 - creating, 299–305
- navigation
 - <a> element, 249, 273
 - CSS Control Adapters, use of, 279
 - Menu control, 255, 259–268
 - redirection to other page, 274–277
 - server controls for, 130
 - server-side redirects, 277–279
 - site map, role in, 255–258
 - SiteMapPath, 255, 272–274
 - tips for, 279
 - TreeView, 255, 268–271
 - Uniform Resource Locator (URLs), 249–254
- .NET Framework, downloading/installing, 683
- New Folder, 51
- New Keyword, 191
- New Project, 5-6, 36–37
- New Style, 95–96, 102
- New Web Site, 5-6, 35, 38
- Not operator, 160
- NTFS file system, configuring for site deployment, 689–693
- number(s), storing, data types for, 144–145
- numbering, adding to list, 61

O

- object(s)
 - constructors, 191–193
 - methods, 190–191
 - object initializers, 192
 - in object-oriented programming, 185–186
- Object Browser, 729
- Object Relational Designer Toolbox, 454
- ObjectDataSource, 408
- object-oriented programming, 185–199
 - access modifiers, 196–197
 - classes, 186–187
 - constructors, 191–193
 - encapsulation, 196
 - events, 197–198
 - inheritance, 193–196

- methods, 146, 190–191
- objects in, 185
- properties, 187–190, 192
- one-way data binding expression, 432
- opening web sites, existing sites, 38, 40–41
- Operator property, 316–317
 - validation control, 315
- operators, 153–161
 - arithmetic, 154–156
 - assignment, 154
 - C#, 154–161
 - comparison, 156–158
 - concatenation, 158–159
 - functions of, 153
 - logical, 159–161
 - Visual Basic, 154–161
- OR, SQL, 380
- Or operator, 158–159
- Order By query operator, 459
- ORDER BY, SQL, 380, 381, 392, 440
- ordering data, database, 380–381
- OrElse operator, 160–161
- organization
 - folders, use of, 46–48
 - reorganizing site, 48
- orientation, Menu control, 259, 262
- output caching, 534–538
- output window, contents of, 22
- overloaded constructors, 192

P

- padding
 - Cascading Style Sheets (CSS), 77, 79, 93–95
 - settings, 95
- page(s), Web. *See* Web page(s)
- Page Directive, 138
- Page_Load, 328–329, 600
- Page.IsPostBack, 302–303
- pageOutput, 658
- PagerStyle, 505
- PageWrapper, 84, 103
- paging controls
 - data-bound, 408
 - DataPager control, 468
 - PagerStyle, 505

- Panel server control, 116, 121
 - use, example of, 122–123
- paragraphs, new, 57
- parameters
 - database, 412–414, 420
 - functions/subroutines, 173–176
- Passport, 554
- password(s), 563–567
 - case sensitivity, 550
 - change password, 564–566
 - changing, , 564–566
 - characters in, 550
 - controls, adding to page, 564–567
 - encrypted format, 567
 - hashed format, 566, 568
 - login server controls, 131
 - password recovery, 563–564
 - passwordAttemptWindow, 568
 - passwordFormat, 568
 - PasswordRecovery, 563–566
 - PasswordRecoveryText, 554
 - PasswordRecoveryUrl, 554
 - passwordStrengthRegularExpression, 568
 - placeholders, 561–562
 - provider configuration attributes, 566–568
- PathDirection, 272
- PathSeparator, 272, 274
- permissions
 - elements of, 546
 - IIS/built-in web server, 688
 - ListView template customization, 486–507
 - Modify permissions, 376
 - NTFS file system configuration, 689–693
- personalization. *See* Profile(s)
- photo album
 - creating, 470–475
 - deleting photos, 477–486, 584
 - front-end page, creating, 493–498
 - inserting photos, 477–486
 - management by users, 610–615
 - uploading images by user, 486–492
- Pick URL, 63
- pixels, CSS screen size, 84
- Place Code in Separate File, 42
- placeholders
 - auto placement, 166
 - content, on master pages, 203–207, 211–212
 - mail message, 336
 - menu creation, 259
 - Placeholder server control, 116, 121
 - Repeater, 610
 - username/passwords, 561
- PlainPostBack, 134–137
- Planet Wrox
 - anonymous identification, enabling, 615–617
 - anonymous profile, deleting, 616–617
 - application, connecting to SQL Server, 731–738
 - base page, creating, 216–220
 - breadcrumb, creating, 272–274
 - buttons, 109–112
 - caching, 534–538, 539–542
 - Cascading Style Sheets (CSS), 68–72, 79–84
 - Cascading Style Sheets (CSS) in VWD, 88–103
 - configuration settings, changing, 569–571
 - confirmation e-mail, sending, 561–563
 - contact forms, 312–313, 317–320
 - content pages, adding to site, 208–210
 - controls, enhancing with styles, 505–510
 - copying web site, 675–676
 - database, attaching to SQL Server, 728–730
 - database, exporting, 696–697
 - data-driven web page, GridView, 409–415
 - delete, enable on GridView, 521–525
 - delete rows, error handling, 525–530
 - e-mail, 328–331, 333–337
 - embedded/inline styles, creating, 93–99
 - exception handling, 630–635, 636–642
 - features of, 30

- files, adding to site, 43
- flicker-free pages, creating, 347–351
- folder for, creating, 38–39
- folders, organizing files, 46–48
- formatted text, adding, 56–58
- forms, breaking into pages, 124–128
- functionality of, 30
- genre preferences, storing in Profile, 597–606
- genres/reviews list with LINQ to SQL, 465–468
- homepage, styling, 79–84
- images, adding to theme, 232–233
- labels, 109–112
- LINQ to SQL Classes, 453–458
- links, adding to page, 63
- list server controls, adding to page, 116–120
- login controls, 548–553
- login page, 209–210, 548–549
- management folder, blocking access, 578–582
- Managers role, creating/assigning, 573–577
- master pages, creating, 205–207
- membership element, modifying, 569–571
- menu control, styling, 264–268
- menus, adding to site, 260–263
- My Profile page, 31
- NavigateUrl property, creating, 299–305
- navigation, TreeView, 269–271
- new account, creating, 549–554
- as new site, creating, 38–39
- NTFS file system settings, 689–693
- online example, web site for, 32
- opening, 43
- password controls, adding to page, 564–567
- photo album, creating, 470–475
- photo album, delete photos, 584
- photo album, front-end page, creating, 493–497
- photo album, inserting/deleting items, 477–486
- photo album, management by users, 611–615
- photo album, uploading images by user, 486–492
- Profile, filling in text box, 597–598
- reviews, limit list of, 606–607
- Roles, checking, 584–586
- server controls, use of, 109–112
- sign up pages, 550–554
- site wide exception handling, 636–641
- skins, creating for button control, 240–244
- state engine, use of, 133–137
- style sheets, attaching to document, 91–93
- styles, managing, 101–103
- tables, 59–62
- tables, database, 394–397
- tables, relationship between, 399–402
- templates, creating, 221–224
- text box, adding to page, 109–112
- themes, creating, 227–230
- Timer control, 352–357
- tracing, 659–663
- UpdatePanel, adding to page, 343–345
- user controls, adding to master page, 287–290
- user controls, creating, 284–287
- user controls, enums for, 295–299
- user controls, registering in web.config, 290–291
- users' profiles, viewing, 618–621
- user-selected theme, 233–240
- Web Forms, adding to site, 51–55
- Web Service, calling from client code, 364–370
- Web Service, creating, 363–364
- Web.sitemap file, creating, 257–258
- Welcome Page, 30
- Wizard server control, use of, 124–128
- Pointer server control, 116
- Position
 - Cascading Style Sheets (CSS), 77, 81–83
 - height/width of element, 77, 81–82

POST

- data acquisition by, 309
- forms, submitting, 135
- PostalAddress, 591, 595
- postback
 - AutoPostBack, 117, 122
 - avoiding. *See* ASP.NET AJAX
 - defined, 21
 - page life cycle, events of, 215, 512–513
 - Page.IsPostBack, 302–303
 - Plain PostBack, 134–137
 - PostBackTrigger, 345
- power operator, 154–156
- prefix, tagname, 287
- PreRenderComplete, 515
- primary keys, database, 394
- private access modifiers, 197
- Profile(s)
 - anonymous identification, enabling, 615–617
 - anonymous profile, deleting, 616–617
 - attributes, functions of, 590
 - C#, 592–594, 596–597, 600–601, 603–605, 607, 609, 612–615, 617–620
 - changes, saving, 597
 - configuring, 589
 - creating, 593–595
 - data, storage in, 598–602
 - enabling, 589
 - functions of, 546, 589
 - genre preferences, storing in, 602–606
 - non-standard data, use of, 591–592
 - operation of, 597–598, 601–602, 605–606, 609–610
 - of other users, viewing, 618–620
 - profile groups, 590–591
 - ProfileBase, 596
 - ProfileCommon, 598, 618, 621
 - ProfileManager, 618
 - ProfileParameter, 420
 - reviews, limit list of, 606–610
 - simple properties, 590
 - text box, filling in name from, 596–597
 - tips for use, 622

type attribute, 590

Visual Basic, 592–594, 596–597, 600–601, 603–605, 607, 609, 612–615, 617–620

programmatically caching, 539–540

programming languages

ASP.NET support, 144

Common Type System (CTS), 144–146

interoperability, 144

Language option, 5, 39

primitive types, 144

programming web pages, 141–201

App_Code folder, 176–180

arrays, 148–150

C#, 143–199

calculator, programming example, 165–169

collections, 151–152

comments, 182–185

data type conversion, 146–148

data types, 142

debugging, 641–656

errors, types of, 625–627

exception handling, 625–641

functions, 173–176

generics, 152–153

hardcoded settings, avoiding, 667–674

logical constructions, 161–164

loops, 169–173

namespaces, 180–182

object oriented. *See* object-oriented programming

operators, 153–161

statements, 153

subroutines, 173–176

switches, 164–165

tips for, 199–200

tracing, 656–662

variables, 142–146

Visual Basic, 143–199

property(ies)

adding to user controls, 293–299

automatic, 189

Cascading Style Sheets (CSS), 71–72, 76–77

- object-oriented programming, 187–190, 192
- read-only/write-only, 189–190
- Properties Grid
 - banners, user controls, 286
 - buttons, 109–111
 - functions of, 18
 - in Markup View, 113
 - menu creation, 262
 - Text property, changing, 21
- Property Pages, 55
- PropertyValuesBinary, 597
- PropertyValuesString, 597
- protected access modifier, 197
- providers
 - application services, 548
 - ProviderName, 413
- public access modifiers, 196
- Public keyword, 174, 196
- publishing web site, 678–679

Q

- query(ies), database
 - creating, 381–385
 - Query Designer, 381–385
 - query string, 275
 - QueryStringParameter, 420
 - See also* LINQ
- query operators, 461–462
 - First, 461–462
 - FirstOrDefault, 461–462
 - From, 458–460
 - Last, 461–462
 - LastOrDefault, 461–462
 - LINQ to SQL, 458–462
 - Order By query, 459
 - Select, 458
 - Single, 461
 - SingleOrDefault, 461
 - Skip, 461
 - SkipWhile, 461
 - Take, 461
 - TakeWhile, 461
 - Where, 460

R

- RadioButton server control, 116, 137

- RadioButtonList server control, 116
- RangeValidator, 315
- ReadAllText, 333
- read-only
 - properties, 189–190
 - ReadOnly, 591
- Redirect
 - C#, 275–276
 - operation of, 277
 - to other page, 274–277
 - Visual Basic, 275–276
- ref keyword, 176
- Refactoring, 285
- registration
 - ASP.NET in IIS, 683
 - database, 377–378
 - Register directive attributes, 287
 - of user controls, 287–288, 290–291
 - Web Services, 364, 368
- RegularExpressionValidator, 316, 318, 320
- relational databases
 - functions/uses of, 374–375
 - types of, 375
 - See also* SQL Server 2005 Express Edition
- relative URLs, 249–253
 - benefits of, 250
 - root-based, 250
 - in server-side controls, 251
- RememberMeSet, 525, 555
- remote server
 - deployment option, 674
 - transfer data to, 696–697
- Rename Item, 23
- rendering
 - page life cycle, events of, 215, 512–513
 - PreRenderComplete, 488
 - RenderCurrentNodeAsLink, 272
 - RenderMode, 345
- Repeater control, 465, 467–468, 607, 609–610
- Replace in Files, starting, 22
- Replace property, 336
- RequestLimit, 624
- RequiredFieldValidator, 318, 439, 599, 600

- contact form, creating, 312–313, 318–320
- RequireMail, 561
- RequiresQuestionAndAnswer, 568
- RequiresUniqueEmail, 568
- Reset, Visual Web Developer (VWD)
 - settings, 28–29
- Restore Down, 45
- Return keyword, 175
- Reviews, 456–457
 - limit list of, Profile, 606–607
- Role(s)
 - assigning, 571–572
 - checking programmatically, 582–585
 - Database Role Membership, 733
 - functions of, 546
 - Web Site Administration Tool (WSAT), 573–576
- Role Manager, 571–572
 - assignments, methods for, 571–572
 - configuring, 571
 - functions of, 571
- RoleGroup, 556, 578, 581
- row(s)
 - adding to tables, 61
 - deleting, error handling, 525–530
 - GridView, creating, 516–517
 - RowDataBound, event handler, writing for, 518–522
 - RowStyle, 505, 508–509
 - SelectedRowStyle, 505
- rules, Cascading Style Sheets (CSS), 71–72, 81–83, 89, 104
- runtime
 - page processing, 9–10
 - run-time files, 655

S

- saving
 - to all open documents, 53, 92
 - Cascading Style Sheets (CSS), 82–83, 92
 - Save All, 53
 - Visual Web Developer (VWD), 21
- screen, pixel size, Cascading Style Sheets, 84

- ScriptManager, 345–346
 - functions of, 345
 - placement of, 345–346
 - properties of, 346
 - ScriptManagerProxy, and Web Services, 363–365
 - UpdatePanel, use with, 344
 - and Web Services, 363, 367
- Scripts property, 347
- ScriptService, 364–366
- security, 545–585
 - application services for, 546–547
 - authentication, 546
 - authorization, 546–547
 - configuration settings, 567–571
 - cookies, 234, 237–240
 - identity, 546
 - IIS/built-in web server permissions, 689
 - login controls, 548–562
 - login server controls, 131
 - management folder, blocking access, 577–582
 - NTFS file system configuration, 690–693
 - passwords, 550–567
 - permissions, 546
 - questions, 550
 - Role Manager, 571–572
 - Roles, checking programmatically, 582–585
 - tips for, 586
 - and tracing, 663
 - user input, validating. *See* validation controls
 - Web Site Administration Tool (WSAT), 572–577
- SELECT, SQL, 378, 381–385, 386, 391–392
- Select Case/switch statements, 164–165
- Select Item References, 222–223
- Select Master Page, 208–210, 212
- Select query operator, 458
- Select Template Options, 222
- SelectCommand, 431
- SelectedIndex server control, 117
- SelectedIndexChanged, 235, 237

- SelectedItem server control, 117
- SelectedItemTemplate, 476
- SelectedRowStyle, 505
- SelectedValue server control, 117
- selector(s)
 - Cascading Style Sheets (CSS), 71–72, 73–76, 87–90
 - Class selector, 75
 - grouping/combining selectors, 75–76
 - ID selector, 73–75
 - pseudo class selectors, 90
 - syntax, 95
 - syntax for, 73–75
 - Type selector, 74
 - Universal selector, 74
- SendMailOnError, 670–674
- server(s)
 - accounts/permissions, 689
 - configure for deployment, common errors, 694–695
 - controls. *See* ASP.NET Server Controls
 - IIS, 679–689
 - remote, transfer data to, 696–697
 - remote server deployment option, 674
 - Server Explorer, functions of, 17
 - server-side redirects, 277–279
 - SQL Server. *See* SQL Server 2005 Express Edition
 - validation, creating, 321–325
- Services property, 347
- SessionParameter, 420
- Set As Start Page, 55
- Set Date, 134–135
- SetFocusOnError, 314
- settings
 - folder, location of, 29
 - resetting Visual Web Developer (VWD), 28–29
- shortcut keys. *See* keyboard shortcuts
- ShowExpandCollapse, 269
- ShowLines, 269
- ShowStartingNode, 262
- ShowToolTips, 272
- sidebars, floating, 84
- sign up pages, creating, 550–554
- SignUpConfirmation, 561
- simple properties, Profile, 590
- single item controls, 407
- Single query operator, 462
- SingleOrDefault query operator, 462
- site map, 255–258
 - elements of file, 255–257
 - file extension (.sitemap), 43
 - functions of, 43
 - SiteMapDataSource, 258, 262, 263, 408
 - SiteMapDataSource1, 269
 - Web.sitemap file, creating, 257–258
- SiteMapPath, 272–274
 - breadcrumb, creating, 272–274
 - functions of, 130, 255
 - properties of, 272
 - server controls for, 130
- skins
 - combined with styles/themes, 508–512
 - creating for button control, 242–244
 - defined, 240
 - disabling, 244
 - file extension (.skin), 43, 240
 - functions of, 43
 - named skins, 243–244
 - SkinID, 243
- Skip query operator, 461
- SkipWhile query operator, 461
- Smart Tasks menu, 117–118, 125, 211–212, 410
- SmtpClient, 326–331, 335, 632
- SmtpException, 630
- Solution Explorer
 - accessing, 17
 - Database Explorer, 376–378
 - functions of, 17, 19–20
 - new web site, 39
 - start page, assigning, 54–55
- sorting
 - database, 380, 384
 - Sort Items Alphabetically, 23, 24
 - SortByCategory, 657
 - SORTORDER, 380–381, 384, 392
- Source View. *See* Markup View
- Split View
 - Cascading Style Sheets (CSS), 69
 - functions of, 49–50

- SQL (Structured Query Language)
 - functions of, 375
 - logical operators, 380
 - See also* SQL Server 2005 Express Edition
- SQL Server 2005 Express Edition
 - access, methods of, 698
 - application, connecting to, 731–739
 - authentication, 725–726
 - Configuration Manager, 727
 - connection error, 727
 - connection strings, 418–420
 - data types in, 393–394
 - database, attaching to, 728–730
 - file extension (.mdf), 44
 - functions of, 44, 375–376
 - identity columns, 392, 394–397
 - installing, 3–4
 - and LINQ. *See* LINQ to SQL
 - Management Studio, 698, 726–727
 - named and unnamed instance, installing by, 725
 - Object Browser, 729
 - Object Explorer, 728, 730–733
 - primary keys, 394
 - remote connections, enabling, 727–728
 - restarting, 728
 - Setup Wizard, 739–740
 - SqlMembershipProvider, 548
 - SqlRoleProvider, 548
 - See also* database(s)
- SqlDataSource, data-driven web page, creating, 408–416
- SqlException, 530
- src, 287, 291
- stale data, and caching, 532
- Standard server controls, 115–130
 - See also* ASP.NET Server Controls
- Start Options, 55
- start page
 - assigning for site, 54–55
 - events of, 215
 - functions of, 18, 19–20
- Start Without Debugging, 7
- statements, operators, 153

- state/stateless
 - ASP.NET state engine, 131–138
 - meaning of, 131–132
- static text, page processing, 10
- StaticEnableDefaultPopOutImage, 259, 262
- StaticHoverStyle, 264–266, 269
- StaticMenuItemStyle, 262, 264–266, 265–266, 269
- StaticSelectedStyle, 266
- StepType, 125, 128
- Street, 596
- strings
 - data types converted to, 144–145
 - DataNavigateUrlFormatString, 429, 432
 - PropertyValuesString, 596
 - query strings, 275, 420
 - String.Format, 361–362
 - See also* connection strings
- strongly typed objects
 - defined, 151–152
 - event handling with, 521–525
 - LinqDataSource control, 475
- style(s), 508
 - combined with skins/themes, 508–512
 - GridView controls, enhancing with styles, 506–512
 - login controls, 555–556
 - operation of, 507, 511–512
 - properties of, 503–504
 - Style Application, 59
 - Style Builder, 84, 87
 - types of, 505
 - VWD Auto Format, 504, 506–507
 - See also* style sheet(s)
- Style Application
 - Auto mode, 59
 - functions of, 59
 - Manual mode, 59
 - toolbar, accessing, 56, 59
- Style Builder
 - Cascading Style Sheets (CSS), 79–84
 - functions of, 84, 87
- style sheet(s)
 - Cascading Style Sheets (CSS), 66–106
 - StyleSheetTheme, 226, 288
 - toolbar, 87–88

- Web files, 42
- Style Sheet toolbar
 - functions of, 87
 - use, example of, 87–88
- submenus, visibility of, 263
- subroutines, 173–176
 - functions of, 174, 180
- Substitute server control, 116, 128
- subtraction operator, 154–156
- SuccessText, 564
- Sum operator, 460
- switches, 164–165
- Synchronize Selected Files, 676
- syntax errors, 625–626
- System.Configuration, 669–670
- System.Net.Mail, 325–326

T

tab

- key navigation, server controls for, 113
- setting tab, 26
- Tab Size, 26

table(s), 59–62

- bullets/numbers, adding to lines, 61–62
- calculator, creating, 166
- creating, 59–62
- database. *See* database(s)
- inserting in site, 60–62
- for page layout, avoiding, 60
- in Profiles, 598–600
- rows/columns, adding, 61
- server controls, 116
- Table Designer, 394–398
- Table server control, 129
- TableName, 470

tag(s)

- ASP.NET, 14
- formatting, adding, 59
- HTML, 10–12
- nesting, 13–14
- Tag Selector, 95, 99
- tagname, 288
- tagprefix, 288
- XHTML, 13–14

- Take query operator, 461

- TakeWhile query operator, 461

- Task List, 24

templates

- ListView control, 475, 486–492
- LoginView, 556
- TemplateField, 427
- See also* web site templates

text

- e-mail, 332–333
- File class, properties of, 332
- foreground color, adding, 57–58
- formatting, 59
- inserting text, 56–58
- text-decoration, 89
- themes, 233–240

text box

- ASP.NET state engine maintenance of, 131–137
- C#, 110–111
- in HTML, 108
- profile, filling in name from, 597–598
- in Profiles, 598–601
- server controls for, 116, 137
- server controls, use of, 108–112, 113–114
- Visual Basic, 110

Text property

- compared to ErrorMessage, 314–315
- Login control, 555

themes, 225–240

- applying to site, 226–227
- combined with styles/skins, 508–512
- creating, 227–230
- defined, 225
- disabling, 226
- folders for, 225, 227, 229–230
- images added to, 232–233
- skins, 240–244
- StyleSheetTheme, 226
- switching, 233–240
- Theme property, 226
- user selection of, 233–240
- Visual Basic, 233–240

Tick, 352

time/date

- in ASPX page, 8–9
- calendar server control, 128, 133–134
- columns, database, 431

- Set Date, 134–135
- tables, database, 397
- XML date file, 44
- Timer control, 352–357
 - in ASPX page, using, 352–357
 - C#, 354, 356
 - operation of, 356–357
 - Visual Basic, 354, 356
- toolbars
 - creating, 27
 - editing, 26–27
 - enabling/disabling, 16, 27
 - Formatting, CSS, 70, 91
 - Style Application, 56, 59
 - Style Sheet, 87–88
 - Visual Web Developer (VWD), 15–16, 26–27
- Toolbox
 - code snippets, adding to, 23–24
 - functions of, 16
 - modifying, 23–24
 - Object Relational Designer Toolbox, 454
 - opening, 16, 19–20, 115
 - pinning to keep open, 16, 19–20
 - resetting, 23, 28
 - server controls in, 115–116
 - Visual Web Developer (VWD), 15–16, 23–24
- tooltips
 - displaying, 269
 - server controls for, 113–114
- tracing
 - attributes of trace element, 658–659
 - C#, 656, 663–664
 - disabling, 662, 663, 699
 - functions of, 657
 - operation of, 660–661, 663
 - page-level, 657–658
 - and performance, 663
 - security issue, 663
 - site-wide, 658–661
 - trace data, adding to page, 660–661
 - traceMode, 657, 659
 - Visual Basic, 656, 663
- TreeView, 269–271
 - C#, 270
 - functions of, 255, 269
 - navigation system, building, 269–271
 - operation of, 271
 - properties of, 268–269
 - server controls for, 130
 - submenus, location of, 259
 - Visual Basic, 270
- Triggers, 345
 - Timer control, 352, 354–355
- troubleshooting
 - and error list, 22
 - and Output window, 22
 - See also* debugging; error(s); error handling
- Try Catch
 - exception handling, 625–635
 - Try Catch Finally, 627, 633
- TryCast, 147
- Type property
 - compared to ErrorMessage property, 315
 - validation controls, 315, 317
- Type selector, selectors, 74
- typeof operator, 158
- U**
 - underline, 56, 89
 - Uniform Resource Locator (URLs), 249–254
 - absolute URLs, 253
 - default documents, 253–254
 - functions of, 249
 - relative URLs, 249–253
 - server-side redirects, 277–279
 - Virtual Path property, 251–253
 - Universal selector, selectors, 74
 - unload, events of, 216
 - unnamed instance, 725
 - Until loop, 171–173
 - UPDATE, SQL, 379, 389, 392
 - UpdateMode, 345
 - UpdatePanel, 343–345
 - adding to page, 343–345
 - functions of, 342
 - operation of, 344

- properties of, 345
- ScriptManager, use with, 345
- and Timer control, 352–357
- UpdateProgress, 347–351
 - flicker-free pages, creating, 347–351
 - operation of, 351
 - properties of, 347
- updating, database data, 389, 434–447
- uploading
 - exceptions related to, 633
 - FileUpload server control, 128
- upper bound, arrays, 149
- URLs. *See* Uniform Resource Locator (URLs)
- user(s)
 - edit/delete, 574
 - enable/disable, 574
 - user input, validating. *See* validation controls
- User(s), Web Site Administration Tool (WSAT), 573–575
- user controls, 282–307
 - adding to master page, 287–290
 - C#, 292–305
 - creating, 284–287
 - enumeration, creating for, 293–299
 - file extension (.ascx), 283
 - formatting of, 503–601
 - functions of, 42
 - NavigateUrl, creating, 299–305
 - opening, automatic, 283
 - operation at run-time, 289–290, 291
 - properties, adding to, 293–299
 - registration of, 287–288, 290–291
 - tips for use, 306
 - utility of, 283
 - Visual Basic, 292–305
- user feedback
 - contact form, creating, 312–313
 - UpdateProgress, 347–351
- UserName, placeholders, 561–562

V

- validation
 - client-side, warning about, 311
 - page, events of, 215
 - server controls, 130

- validation controls, 310–325
 - C#, 322, 324–325
 - client/server-side validation, creating, 321–325
 - CompareValidator, 316–317
 - contact form, extending operations on, 317–320
 - ControlToValidate, 314
 - CssClass property, 314
 - CustomValidator, 321–325
 - Display property, 314
 - EnableClientScript, 314
 - ErrorMessage, 314
 - functions of, 310
 - IsValid property, 315
 - operation at run-time, 314, 320
 - Operator, 316
 - RangeValidator, 315
 - RegularExpressionValidator, 316
 - RequiredFieldValidator, use of, 311–313
 - SetFocusOnError, 314
 - Text property, 314
 - Text versus ErrorMessage, 315
 - Type, 316
 - ValidationExpression, 319
 - ValidationGroup, 314
 - ValidationSummary, 314–315, 321–325
 - ValueToCompare, 317
 - Visual Basic, 321, 324–325
- values
 - assigning to variables, 143, 154
 - Cascading Style Sheets (CSS), 71, 77–78
 - ValueToCompare, 317
- VALUES, SQL, 390
- variables
 - assignment operators, 154
 - backing variables, 187–188
 - declaring, 142–146
 - naming, 144
 - values, assigning to, 143, 154
- VerticalAlign, 504
- View server control, 116
- View Source
 - content pages, 209–210

- HTML, viewing, 21, 135
- ViewState
 - cautions in use of, 305
 - control independence from, 137
 - disabling, 137
 - NavigateUrl property, creating, 299–305
 - and performance, 137
 - values assigned, storage of, 304, 306
- virtual path
 - behavior, example of, 251–253
 - mail message, 336
 - setting for, 251, 253
- visibility, Cascading Style Sheets (CSS), 77
- visible server control, 113, 122–123
- VisibleWhenLoggedIn, 555, 559
- VisitAddress, 592
- visited selector, 90
- Visual Studio 2008, 2–3
 - settings folder, 29
 - trial version, 3
- Visual Web Developer (VWD), 15–32
 - Auto Format, 504, 506–507
 - bookmark window, 22
 - customizing, 22–28
 - databases, working with. *See* database(s)
 - Document Window, 18, 26
 - downloading, 3
 - error list, 22
 - find and replace, 22
 - full screen, shrinking, 45
 - functions of, 2
 - help files, 4
 - installing, 3–4
 - as integrated development environment (IDE), 15
 - IntelliSense, 52–53
 - interface, parts of, 15
 - keyboard shortcuts, 27
 - main menu, 15
 - opening existing sites, 38, 40–41
 - output window, 22
 - Properties Grid, 18
 - resetting, 28–29
 - saving, 20

- Solution Explorer, 17
- SQL Server with, 3–4
- Start Page, 18
- tips for use, 32
- toolbar area, 16
- toolbars, modifying, 26–27
- Toolbox, 16, 23–24
- using, building first site, 5–9
- using, developing single web page, 19–20
- versions of, 2
- Web Site Projects, 35
- web site templates, 36–37
- windows, rearranging, 23
- void method, 175
- W**
- Warn, 660
- Warning, error list, 22
- WCF Service template, 37
- Web Application Projects
 - creating project, 36
 - functions of, 35–36
- Web Development Settings, 5
- Web files, 41–44
 - functions of, 42
 - HTML page, 42
 - JScript File, 43
 - Master Page, 42
 - Site Map, 43
 - skin file, 43
 - style sheets, 42
 - user controls, 42
 - web configuration file, 42
- Web Forms, 42
- Web Forms, 48–55
 - adding to site, 51–55
 - AJAX Web Form, 42
 - breaking into pages, Wizard server control for, 124–128
 - C#, 51–54
 - Code Behind files, 51–55
 - creating, 51–55
 - Design View, 49–50
 - file extension (.aspx), 42
 - inline code, 51–55
 - Markup View (Source View), 49

- new web site, 38–39
- POST for submitting, 135
- Split View, 49–50
- tips for use, 64
- user input, validating. *See* validation controls
- views, changing, 50–52
- Visual Basic, 52, 54
- web page(s)
 - base page, 213–214, 216–220
 - consistency, tips for, 245
 - content pages, 208–210
 - initialization, 215
 - life cycle of, 214–216, 511–517
 - linking pages, 62–64
 - master page, 203–207
 - opening in browser, 7–8
 - Page Directive, functions of, 138
 - page request, events of, 215
 - Page_Load, 328–329, 600
 - page-level tracing, 656–657
 - PageWrapper, 84, 103
 - postback, 134–137, 215, 512–513
 - processing, steps in, 9–10
 - programming. *See* programming web pages
 - skins, 240–244
 - start page, assigning, 54–55
 - templates, reusable, 221–224
 - themes, 225–240
- web servers. *See* server(s)
- Web Services, 359–370
 - in Ajax web site, 362–370
 - C#, 360–365
 - calling from client code, 364–370
 - configuring, 362
 - creating, 359–362
 - file extension (.asmx), 44, 359
 - functions of, 37, 44, 357
 - operation of, 361–362, 367–369
 - registration of, 364, 368
 - and ScriptManager, 363, 367
 - and ScriptManagerProxy, 363–365
 - ScriptManagerProxy, 363–366
 - ScriptService in, 364–366
 - service class, adding attribute, 362–363
 - Services property, 347
 - Visual Basic, 358, 360–365
 - WebMethod attribute, 358, 361
- Web site(s)
 - use of term, 35
 - See also* ASP.NET web site; Planet Wrox
- Web Site Administration Tool (WSAT), 573–577
 - functions of, 572
 - Manager's role, creating/assigning, 572–576
 - operation of, 576
- Web Site Projects
 - creating, 35
 - functions of, 35
- web site templates, 36–37, 221–224
 - ASP.NET Web Service template, 37–38
 - ASP.NET web site, 37–38
 - C#, 222–223
 - code, generics for, 152–153
 - creating, 221–224
 - empty web site template, 38
 - exporting, 222, 224
 - My Templates, 36
 - operation of, 224
 - Templates box, 5
 - Visual Basic, 221
 - WCF Service template, 37–38
 - zip files, 223–224
- Web User Control, 284
 - file extension (.ascx), 42
 - functions of, 42
- web.config
 - application settings, moving to, 670–674
 - connection strings, storing, 418–420
 - creating, 5
 - debugging, disabling, 699
 - hardcoded settings, moving to, 667–674
 - management folder, blocking access, 576–580
 - user controls, registering in, 290–291
- WebMethod attribute, 358, 361
- WebResource.axd, 263

- WHERE, filtering data, 379–380, 389
- Where query operator, 460
- While loop, 171–173
- white space, 94, 504
- width of elements
 - border width, 504
 - Cascading Style Sheets (CSS), 77, 81–83
 - server controls for, 113
 - styles, 504
- windows
 - docking, 23
 - dragging and dropping, 23–24
 - floating, 23–24
- Windows, authentication, 725–726
- Windows 2003, IIS install, 680–681
- Windows Server 2003, IIS configuration, 684–686
- Windows Server 2008
 - IIS configuration, 686–689
 - IIS install, 681–682
- Windows Vista
 - IIS configuration, 686–688
 - IIS install, 681
- Windows XP
 - IIS configuration, 684–685
 - IIS install, 680–681
- With keyword, 192
- Wizard server control, 116, 121
 - use, example of, 124–128
- wrap, styles, 504
- Write, 626
- WriteAllText, 332
- write-only properties, 189–190

X

- XHTML, compared to HTML, 13–14
- XML
 - comments, 184–185
 - file extension (.xml), 44
 - LINQ to XML, 477
 - XML File, functions of, 44
 - XML server control, 116, 129
 - XmlDataSource, 408

Z

- zip files, templates, 223–224