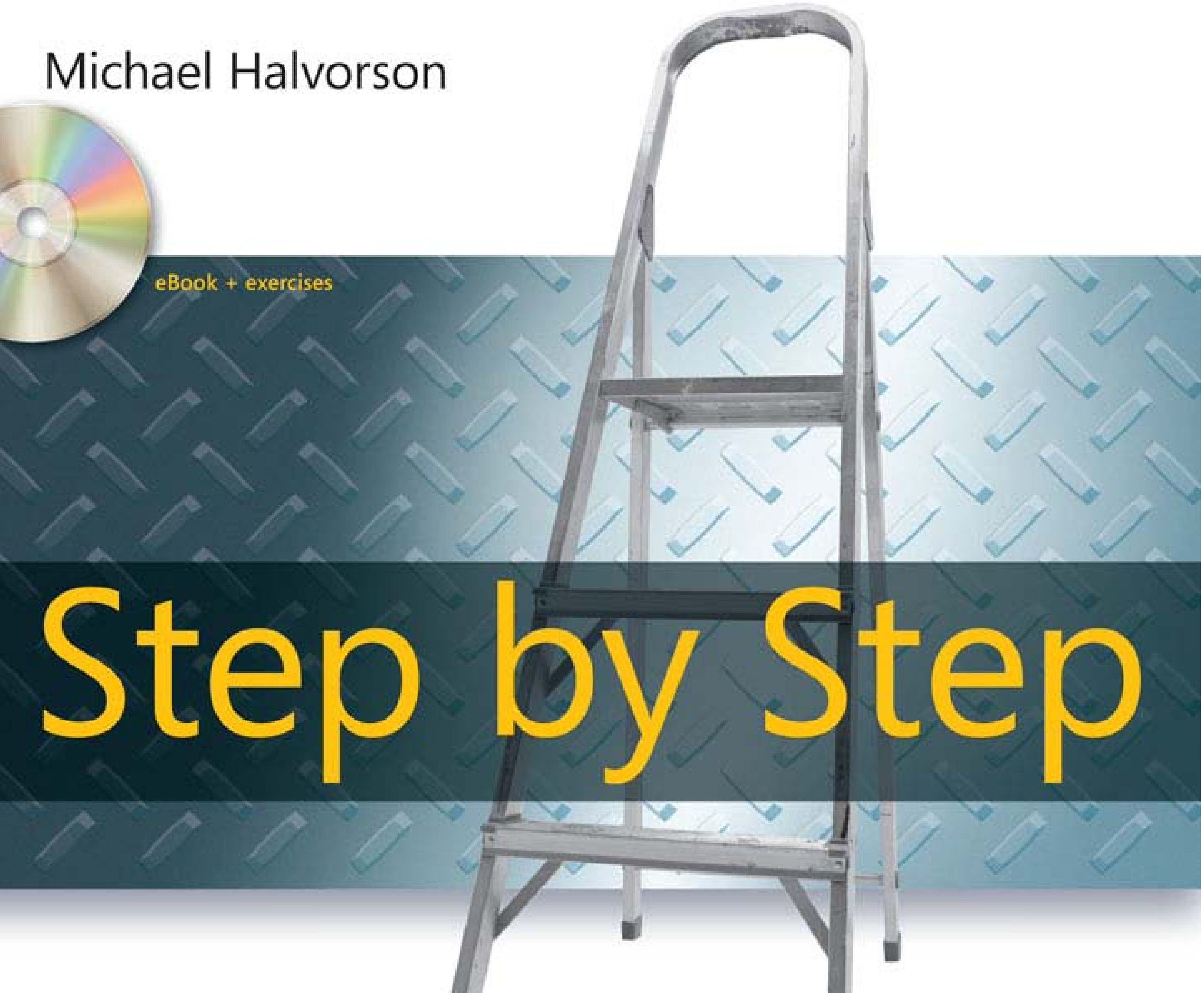
Microsoft

Microsoft®

Visual Basic® 2010

Michael Halvorson



Step by Step

How to access your CD files

The print edition of this book includes a CD. To access the CD files, go to http://aka.ms/626690/files, and look for the Downloads tab.

Note: Use a desktop web browser, as files may not be accessible from all ereader devices.

Questions? Please contact: mspinput@microsoft.com

Microsoft Press

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Control Number: 2010924441

ISBN: 978-0-7356-2669-0

Printed and bound in the United States of America.

4 5 6 7 8 9 10 11 12 QG 7 6 5 4 3 2

Distributed in Canada by H.B. Fenn and Company Ltd.

A CIP catalogue record for this book is available from the British Library.

Acquisitions Editor: Ben Ryan
Developmental Editor: Devon Musgrave
Project Editor: Valerie Woolley
Editorial Production: Christian Holdener, S4Carlisle Publishing Services
Technical Reviewer: Technical Review services provided by Content Master, a member of CM Group, Ltd.
Cover: Tom Draper Design

Body Part No. X16-88509

[2012-03-16]

For Henry

Contents at a Glance

Table of Contents

Acknowledgments

Writing a computer programming book is fascinating because the whole process begins well before the software is actually finished. Authors meet with software developers and computer book publishers, explore product specifications and early releases of the software, review the comments and suggestions that readers of previous editions have offered, develop a writing plan and schedule, and begin testing their ideas with beta versions of the product. This iterative process produces important insights and continues (with mounting fervor) until the software is complete and the final books are shipped to the printer.

Microsoft Press is a fantastic place to write a computer programming book. At each stage in the publishing process, talented team members work together to cultivate valuable technical contacts and resources, build visionary product deployment strategies, explore the hidden benefits of emerging technologies, and pick the right words and images to describe them. *Microsoft Visual Basic 2010 Step by Step*, now in its eighth edition, has benefited significantly from this dynamic and innovative publishing environment over the years.

I gratefully acknowledge the support and assistance of the following people who helped to plan, edit, test, produce, and market our book this time (in the order that I worked with them): Ben Ryan, Devon Musgrave, Valerie Woolley, Susan McClung, and Christian Holdener. In particular, Valerie Woolley enthusiastically kept my writing on schedule and insured that our book would fit well in the *Step by Step* series that Microsoft Press is so well known for. I am also very grateful to the Microsoft Visual Studio 2010 development team for providing me with beta and release candidate software to work with.

As always, I offer my deepest gratitude and affection to my family for their continued support of my writing projects and various academic pursuits.

Introduction

Microsoft Visual Basic 2010 is an important upgrade and enhancement of the popular Visual Basic programming language and compiler, a technology that enjoys an installed base of millions of programmers worldwide. Visual Basic 2010 is not a stand-alone product but a key component of Microsoft Visual Studio 2010—a comprehensive development system that allows you to create powerful applications for Windows, the Web, handheld devices, and a host of other environments. Whether you purchase one of the commercial editions of Visual Studio 2010 (described later in this Introduction) or you download Visual Basic 2010 Express for a free test-drive of the software, you are in for an exciting experience. The latest features of Visual Basic will increase your productivity and programming prowess, especially if you enjoy using and integrating information from databases, entertainment media, Web pages, and Web sites. In addition, an important benefit of learning Visual Basic and the Visual Studio Integrated Development Environment (IDE) is that you can use many of the same tools to write programs for Microsoft Visual C++ 2010, Microsoft Visual C# 2010, Microsoft Visual Web Developer 2010, and other popular products.

Microsoft Visual Basic 2010 Step by Step is a comprehensive introduction to Visual Basic programming using the Visual Basic 2010 software. I've designed this practical, hands-on tutorial with a variety of skill levels in mind and by following the premise that programmers learn by doing. In my opinion, the best way to master a complex technology like Visual Basic is to learn essential programming techniques through carefully prepared tutorials that can be completed on your own schedule and at your own pace. And although I have significant experience with college teaching and corporate project management, this book is not a dry textbook or an "A to Z" programmer's reference. Instead, it is a practical hands-on programming tutorial that puts *you* in charge of your learning, developmental milestones, and achievements. By using this book, programmers who are new to this topic will learn Visual Basic software development fundamentals in the context of useful, real-world applications; and experienced Visual Basic programmers can quickly master the essential tools and techniques offered in the Visual Basic 2010 upgrade.

Complementing this comprehensive approach is the book's structure—4 topically organized parts, 20 chapters, and 56 step-by-step exercises and sample programs. By using this book, you'll quickly learn how to create professional-quality Visual Basic 2010 applications for the Windows operating system and a variety of Web browsers. You'll also have fun!

Visual Basic Versions

So how did we get here, anyway? Between 1991 and 1998, Microsoft released six versions of Visual Basic (versions 1.0 through 6.0), which revolutionized software development for Windows by introducing event-driven programming to a wide audience based on the QuickBasic programming language and an IDE. After a significant period of additional development and innovation, Microsoft released Visual Basic .NET 2002, an object-oriented programming language closely related to Visual Basic but implemented on the Microsoft .NET Framework, a comprehensive library of coded solutions intended to be used by most new applications that run on the Windows platform. As improved versions of Visual Basic came out in 2003, 2005, and 2008, Visual Basic became a component within the Visual Studio suite, and significant support was added to the product for database, Internet, and team development projects, as well as continued improvements to the .NET Framework. Visual Basic 2010 is now so tightly integrated with Visual Studio that it is available only as a component in the Visual Studio 2010 programming suite, which includes Visual C#, Visual C++, Visual Web Developer, and other Microsoft .NET development tools. Since 2005, both Visual Basic and Visual Studio have been marketed without the ".NET" moniker, although they are still based on the .NET Framework technology.

Visual Studio 2010 is distributed in several different product configurations, including Professional, Premium, Ultimate, and Express. I've written this book to be compatible with all editions of Visual Basic 2010 and Visual Studio 2010, but especially with the tools and techniques available in Visual Studio 2010 Professional and Visual Basic 2010 Express. Although Visual Basic 2010 is similar in many ways to Visual Basic 2008, there are many important differences and improvements, so I recommend that you complete the exercises in this book using the Visual Basic 2010 software.

Note The Visual Studio 2010 software is not included with this book. The CD distributed with most versions of this book contains practice files, sample databases, and other useful information that requires the Visual Studio 2010 software (sold separately) for use. If you don't have Visual Studio, you can download Visual Basic 2010 Express for free, and it contains an amazing palette of features, though obviously not all the features of Visual Studio Professional, Premium, or Ultimate. As you complete the exercises in this book, I will note from time to time which features are unavailable to you if you are using Visual Basic 2010 Express. Also note that if you are using Visual Basic 2010 Express and you want to complete Chapter 20, "Creating Web Sites and Web Pages by Using Visual Web Developer and ASP.NET," you will need to download Visual Web Developer 2010 Express to complete the exercises. Visual Web Developer is included in Visual Studio Professional, Premium, and Ultimate, but not Visual Basic Express.

Downloading Visual Basic 2010 Express

As noted previously, if you don't have Visual Studio 2010 Professional, Visual Studio 2010 Premium, or Visual Studio 2010 Ultimate, you can complete most of the exercises in this book by downloading Visual Basic 2010 Express from the Web for free. This will give you an opportunity to learn Visual Basic programming and see for yourself if you want to upgrade to a full release of the Visual Studio software.

To download Visual Basic 2010 Express, complete the following steps:

1. Open a Web browser (such as Internet Explorer), and go to *http://www.microsoft.com/express*.
2. Follow the instructions on the screen to download Visual Basic 2010 Express.

   On the Express Web site, you will also see an Express product feature chart that compares the Express product to the full versions of Visual Studio. Although there are some key differences between the full versions and Visual Basic 2010 Express, many of these differences have no effect on how you learn the essential techniques and features of Visual Basic programming. After you experiment with the Express product, you can decide whether you want to upgrade to one of the full versions of Visual Studio or not. Now, let's get started learning about Visual Basic and how this programming course works!

Finding Your Best Starting Point in This Book

This book is designed to help you build skills in a number of essential areas. You can use it if you're new to programming, switching from another programming language, or upgrading from Visual Basic 2008. Use the following table to find your best starting point in this book.

If you are …	Follow these steps
New to programming	**1.** Install the practice files as described in the section "Installing and Using the Practice Files," later in this Introduction. **2.** Learn basic skills for using Visual Basic 2010 by working sequentially from Chapter 1 through Chapter 17. **3.** Complete Part IV, "Database and Web Programming," as your level of interest or experience dictates.
Upgrading from Visual Basic 2005 or 2008	**1.** Install the practice files as described in "Installing and Using the Practice Files." **2.** Complete Chapters 1 through 4, skim Chapters 5 through 17, and complete Chapters 18 through 20.

If you are …	Follow these steps
Upgrading from Visual Basic 6.0	**1.** Install the practice files as described in the section "Installing and Using the Practice Files." **2.** Read Chapters 1 through 4 carefully to learn the new features of the Visual Studio 2010 development environment. **3.** Skim Chapters 5 through 13 to review the fundamentals of event-driven programming, using variables, and writing decision structures. Give special attention to Chapters 5, 6, 9, and 12. **4.** Work sequentially from Chapters 14 through 20 to learn the new Visual Basic 2010 features related to user interface design, database programming, and Web programming.
Referencing this book after working through the chapters	**1.** Use the index to locate information about specific topics, and use the table of contents to locate information about general topics. **2.** Read the Quick Reference at the end of each chapter for a brief review of the major tasks in the chapter. The Quick Reference topics are listed in the same order as they're presented in the chapter.

Hardware and Software Requirements

You'll need the following hardware and software to complete the exercises in this book:

- Windows 7, Windows Vista, Windows XP, Windows Server 2003, or Windows Server 2008
- Visual Studio 2010 (Professional, Premium, or Ultimate) or Visual Basic 2010 Express
- 1.6 GHz processor
- 1 GB RAM
- 3 GB of available hard drive space
- 5400 RPM hard disk drive
- DirectX 9–capable video card that runs at a display resolution of 1024 × 768 or higher
- DVD drive

You also need to have Administrator-level access to your computer.

Note This book and the practice files were tested using Visual Studio 2010 Professional and Visual Basic 2010 Express on Windows 7. You might notice a few differences if you're using other editions of Visual Studio 2010. In particular, if you're using Visual Basic 2010 Express, a few features will be unavailable to you. In addition, all the screen shots in this book were captured using Windows 7. If you are using another version of Windows or Windows Server, you'll notice a few differences in some of the screen shots.

Prerelease Software

This book was reviewed and tested against the Release Candidate of Visual Studio 2010. The Release Candidate was the last preview before the final release of Visual Studio 2010. This book is expected to be fully compatible with the final release of Visual Studio 2010 and Visual Basic 2010. If there are any changes or corrections for this book, they will be collected and added to an easy-to-access Microsoft Knowledge Base article on the Web. See “Support for This Book,” later in this Introduction.

Digital Content for Digital Book Readers: If you bought a digital-only edition of this book, you can enjoy select content from the print edition’s companion CD.
Visit **http://www.microsoftpressstore.com/title/9780735626690** to get your downloadable content. This content is always up-to-date and available to all readers.

Installing and Using the Practice Files

The CD inside this book contains the practice files that you’ll use as you perform the exercises in the book. For example, when you’re learning how to display database tables on a form by using the *DataGridView* control, you’ll open one of the practice files—an academic database named Faculty2010.accdb—and then use Visual Studio database programming tools to access the database. By using the practice files, you won’t waste time creating files that aren’t relevant to the exercise. Instead, you can concentrate on learning how to master Visual Basic 2010 programming techniques. With the files and the step-by-step instructions in the chapters, you’ll also learn by doing, which is an easy and effective way to acquire and remember new skills.

Important Before you break the seal on the CD, be sure that this book matches your version of the software. This book is designed for use with Visual Studio 2010 and the Visual Basic 2010 programming language. To find out what software you’re running, you can check the product package, or you can start the software, open a project, and then click About Microsoft Visual Studio on the Help menu at the top of the screen.

Installing the Practice Files

Installing the practice files on your hard disk requires approximately 10 megabytes (MB) of disk space. Follow these steps to install the practice files on your computer’s hard disk drive so that you can use them with the exercises in this book.

1. Remove the CD from the package inside this book and insert it into your CD drive.

Note An End-User License Agreement (EULA) should open automatically. If this agreement does not appear, you can double-click StartCD.exe on the CD. If you have Windows 7 or Windows Vista, click Computer on the Start menu, double-click the icon for your CD drive, and then double-click StartCD.exe.

2. Review the EULA. If you accept the terms, select the Accept option, and then click Next.

   A menu appears with options related to the book.

3. Click Practice Files.
4. Follow the on-screen instructions.

> **Note** For best results when using the practice files with this book, accept the preselected installation location, which by default is C:\Vb10sbs. If you change the installation location, you'll need to adjust the paths in several practice files manually to locate essential components, such as artwork and database files, when you use them. Trust me—it is good to use the default installation location.

5. When the files have been installed, remove the CD from your drive and replace it in the package inside the back cover of your book.

   If you accepted the default settings, a folder named C:\Vb10sbs has been created on your hard disk drive, and the practice files have been placed in that folder. You'll find one folder in C:\Vb10sbs for each chapter in the book. (Some of the files represent completed projects, and others will require that you enter some program code.) If you have trouble running any of the practice files, refer to the text in the book that describes those files.

Using the Practice Files

Each chapter in this book explains when and how to use the practice files for that chapter. When it's time to use a practice file, the book includes instructions for opening the file. The chapters are built around scenarios that simulate real programming projects so that you can easily apply the skills you learn to your own work.

> **Note** Visual Basic 2010 features a new file format for its projects and solutions. Accordingly, you won't be able to open the practice files for this book if you're using an older version of the Visual Basic or Visual Studio software. To see what version of Visual Basic or Visual Studio you're using, click the About command on the Help menu.

Visual Studio is extremely customizable and can be configured to open and save projects and solutions in different ways. The instructions in this book generally rely on the default setting for Visual Studio. For more information about how settings within the development environment affect how you write programs and use the practice files, see the section "Customizing IDE Settings to Match Step-by-Step Exercises" in Chapter 1, "Exploring the Visual Studio Integrated Development Environment."

For those of you who like to know all the details, here's a list of the Visual Basic projects included on the CD. Each project is located in its own folder and has several support files. Look at all the things you will be doing!

Project	Description
Chapter 1	
MusicTrivia	A simple trivia program that welcomes you to the programming course and displays a digital photo.
Chapter 2	
Lucky7	Your first program—a game that simulates a Las Vegas Lucky Seven slot machine.
Chapter 3	
Birthday	Uses the *DateTimePicker* control to pick a date.
CheckBox	Demonstrates the *CheckBox* control and its properties.
Hello	A Hello World program that demonstrates the *Label* and *TextBox* controls.
List Box	Demonstrates the *ListBox* control for gathering input.
Radio Button	Demonstrates the *RadioButton* control for gathering input.
WebLink	Demonstrates the *LinkLabel* control that opens a Web browser in your Visual Basic application.
Chapter 4	
Menu	Demonstrates how to use Visual Studio dialog box controls, toolbars, and menus.
Chapter 5	
Advanced Math	Advanced use of operators for integer division, remainder division, exponentiation, and string concatenation.
Basic Math	Basic use of operators for addition, subtraction, multiplication, and division.
Constant Tester	Uses a constant to hold a fixed mathematical entity.
Data Types	Demonstrates Visual Basic fundamental data types and their use with variables.
Framework Math	Demonstrates the .NET Framework classes with mathematical methods.
Input Box	Receives input with the *InputBox* function.
Variable Test	Declares and uses variables to store information.
Chapter 6	
Select Case	Uses a *Select … Case* decision structure and a *ListBox* control to display a welcome message in several languages.
User Validation	Uses the *If … Then … Else* decision structure and a *MaskedTextBox* control to manage a logon process.
Chapter 7	
Celsius Conversion	Converts temperatures from Fahrenheit to Celsius by using a *Do* loop.
Digital Clock	A simple digital clock program that demonstrates the *Timer* control.

Project	Description
For Loop	Demonstrates using a *For … Next* loop to display text in a *TextBox* control, and using the *Chr* function to create a wrap character.
For Loop Icons	Uses a global counter variable in an event procedure as an alternative to loops. This program also displays images by using a *PictureBox* control.
Timed Password	Demonstrates how to use a *Timer* control to create a logon program with a password time-out feature.
Windows Version Snippet	Shows how to use the Insert Snippet command to display the current version of Windows running on a user's computer.
Chapter 8	
Debug Test	A simulated debugging problem, designed to be solved using the Visual Studio debugging tools.
Chapter 9	
Disc Drive Error	Crashes when a CD or DVD drive is used incorrectly. This project is used as the basis of a Visual Basic error handler.
Disc Drive Handler	Completed error handler for loading files that demonstrates the *Try … Catch* syntax.
Chapter 10	
Text Box Sub	A general-purpose Sub procedure that adds items to a list box.
Track Wins	A clean version of the Lucky7 slot machine project from Chapter 2, which you enhance by using public variables and a function that computes the game's win rate.
Chapter 11	
Array Class Sorts	Shows how to create and manipulate large integer arrays. Demonstrates the *Array.Sort* and *Array.Reverse* methods and how to use a *ProgressBar* control to give the user visual feedback during long sorts.
Dynamic Array	Computes the average temperature for any number of days by using a dynamic array.
Fixed Array	Computes the average weekly temperature by using a fixed-length array.
Chapter 12	
Controls Collection	Uses a *For Each … Next* loop and the Visual Studio *Controls* collection to move objects on a form.
URL Collection	Demonstrates a user-defined collection containing a list of Uniform Resource Locators (URLs), or Web addresses, recently visited by the user.
Chapter 13	
Encrypt Text	Demonstrates the *Chr*, *Asc*, *Length*, *Substring*, and *FileExists* methods, as well as a simple encryption scheme to jumble the text in files. Teaches useful text-processing techniques.
Quick Note	A simple note-taking utility that demonstrates the *Clock.LocalTime* property; the *WriteAllText* method; and the *TextBox*, *MenuStrip*, and *SaveFileDialog* controls.

Project	Description
Sort Text	A text file editor with a menu bar that demonstrates how to manage Open, Close, Save As, Insert Date, Sort Text, and Exit commands in a program. Contains a *ShellSort* module for sorting arrays that can be added to other programming projects.
Text Browser	Displays the contents of a text file in a Visual Basic program. Demonstrates menu and dialog box commands, a *Try … Catch* error handler, the *ReadAllText* method, and serves as a foundation for the other programs in this chapter.
Xor Encryption	Explores the *StreamWriter* class and the *OpenTextFileWriter* and *ReadAllText* methods for file management, as well as using the *Xor* operator to encrypt files with a hidden code that is entered by the user.
Chapter 14	
Add Controls	Demonstrates how controls are added to a Windows Form at run time by using program code (not the Designer).
Anchor and Dock	Uses the *Anchor* and *Dock* properties of a form to align objects at run time.
Desktop Bounds	Uses the *StartPosition* and *DesktopBounds* properties to position a Windows Form at run time. Also demonstrates the *FormBorderStyle* property, *Rectangle* structure, and *ShowDialog* method.
Lucky Seven Help	The enhanced Lucky7 program (Track Wins) from Chapter 10, which you enhance again through the addition of a second form to display Help information.
Chapter 15	
Draw Shapes	Demonstrates a few of the useful graphics methods in the *System.Drawing* namespace, including *DrawEllipse*, *FillRectangle*, and *DrawCurve*.
Moving Icon	Animates an icon on the form, moving it from the top of the form to the bottom each time that you click the Move Down button.
Transparent Form	Demonstrates how to change the transparency of a form by using the *Me* object and the *Opacity* property.
Zoom In	Simulates zooming in, or magnifying, a picture box object on a form (in this case, a high-resolution image of the planet Earth).
Chapter 16	
Form Inheritance	Uses the Visual Studio Inheritance Picker to create a form that inherits its characteristics and functionality from another form.
Person Class	Demonstrates how to create new classes, properties, and methods in a Visual Basic project. The new *Person* class is an employee record with first name, last name, and date of birth fields, and it contains a method that computes the current age of an employee.
Chapter 17	
Print Dialogs	Demonstrates how to create Print Preview and Page Setup dialog boxes.
Print File	Handles more sophisticated printing tasks, including printing a multipage text file with wrapping lines. Includes lots of code to use in your own projects.
Print Graphics	Prints graphics from within a Visual Basic program by using an error handler, the *Print* method, and the *DrawImage* method.
Print Text	Demonstrates how simple text is printed in a Visual Basic program.

Project	Description
Chapter 18	
ADO Faculty Form	Demonstrates how ADO.NET is used to establish a connection to a Microsoft Access 2007 database and display information from it.
Chapter 19	
DataGridView Sample	Shows how the *DataGridView* control is used to display multiple tables of data on a form. Also demonstrates how navigation bars, datasets, and table adapters are interconnected and bound to objects on a form.
Chapter 20	
Chap20	Demonstrates using Visual Web Developer and ASP.NET 4 to create a car loan calculator that runs in a Web browser, offers Help information, and displays faculty database records.

Uninstalling the Practice Files

Use the following steps to remove the practice files added to your hard disk drive by the Visual Basic 2010 Step by Step installation program. After uninstalling the practice files, you can delete manually any Visual Basic project files that you have created on your own, should you choose to do so.

If you are running the Windows 7 or Windows Vista operating system:

1. In Control Panel, in the Programs category, click Uninstall A Program.
2. Select VB 2010 SBS in the list of programs, and then click Uninstall.
3. Follow the on-screen instructions to remove the practice files.

If you are running the Windows XP operating system:

1. In Control Panel, open Add Or Remove Programs.
2. In the Currently Installed Programs list, click VB 2010 SBS. Then click Remove.
3. Follow the on-screen instructions to remove the practice files.

Conventions and Features in This Book

Before you start the exercises in this book, you can save time by understanding how I provide instructions and the elements I use to communicate information about Visual Basic programming. The following lists identify stylistic conventions and discuss helpful features of the book.

Conventions

- The names of all program elements—controls, objects, methods, functions, properties, and so on—appear in *italic*.
- Hands-on exercises for you to follow are given in numbered lists of steps (1, 2, and so on). A round bullet (•) indicates an exercise that has only one step.
- Text that you need to type appears in **bold**.
- As you work through steps, you'll occasionally see tables with lists of properties that you'll set in Visual Studio. Text properties appear within quotes, but you don't need to type the quotes.
- A plus sign (+) between two key names means that you must press those keys at the same time. For example, "Press Alt+Tab" means that you hold down the Alt key while you press Tab.
- Readeraids labeled Note, Tip, and Important provide additional information or alternative methods for a step. You should read these before continuing with the exercise.

Other Features

- You can learn special programming techniques, background information, or features related to the information being discussed by reading the sidebars that appear throughout the chapters. These sidebars often highlight difficult terminology or suggest future areas for exploration.
- You can learn about options or techniques that build on what you learned in a chapter by trying the One Step Further exercise at the end of that chapter.
- You can get a quick reminder of how to perform the tasks you learned by reading the Quick Reference table at the end of a chapter. These handy tables are also designed to be used as a topical reference after you complete the book and you need a quick reminder about how to perform a programming task.

Helpful Support Links

You are invited to check out the following links that provide support for the Visual Studio 2010 software and this book's contents.

Visual Studio 2010 Software Support

For questions about the Visual Studio 2010 software, I recommend two Microsoft Web sites:

- *http://msdn.microsoft.com/vbasic/* (the Microsoft Visual Basic Developer Center home page)
- *http://www.microsoft.com/communities/* (the home of technical communities related to Microsoft software products and technologies)

Both Web sites give you access to professional Visual Basic developers, Microsoft employees, Visual Basic blogs, newsgroups, webcasts, technical chats, and interesting user groups. For additional information about these and other electronic and printed resources, see the Appendix, "Where to Go for More Information."

Support for This Book

Every effort has been made to ensure the accuracy of this book and the contents of the companion CD. As corrections or changes are collected, they will be added to a Microsoft Knowledge Base article. Microsoft Press provides support for books and companion CDs at the following Web site:

http://www.microsoft.com/learning/support/books/

If you have comments, questions, or ideas regarding the book or the companion CD, or questions that are not answered by visiting the sites previously mentioned, please send them to Microsoft Press via an e-mail message to *mspinput@microsoft.com*.

Please note that Microsoft software product support is not offered through these addresses, nor does the author of this book offer direct product support.

We Want to Hear from You

We welcome your feedback about this book. Please share your comments and ideas through the following short survey:

http://www.microsoft.com/learning/booksurvey

Your participation helps Microsoft Press create books that better meet your needs and your standards.

Note We hope that you will give us detailed feedback in our survey. If you have questions about our publishing program, upcoming titles, or Microsoft Press in general, we encourage you to interact with us using Twitter at *http://twitter.com/MicrosoftPress*. For support issues, use only the e-mail address shown previously.

Chapter 2
Writing Your First Program

After completing this chapter, you will be able to:

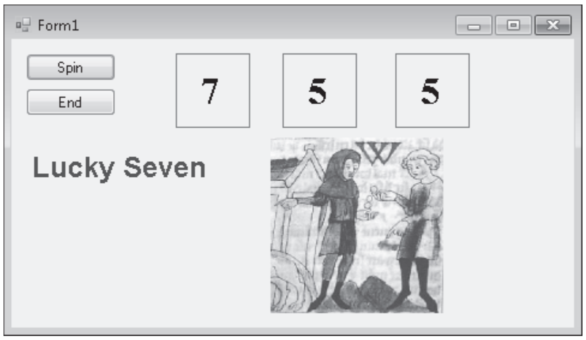
- Create the user interface for a new program.
- Set the properties for each object in your user interface.
- Write program code.
- Save and run the program.
- Build an executable file.

As you learned in Chapter 1, "Exploring the Visual Studio Integrated Development Environment," the Microsoft Visual Studio 2010 Integrated Development Environment (IDE) contains several powerful tools to help you run and manage your programs. Visual Studio also contains everything you need to build your own applications for Windows and the Web from the ground up.

In this chapter, you'll learn how to create a simple but attractive user interface with the controls in the Visual Studio Toolbox. Next you'll learn how to customize the operation of these controls with property settings. Then you'll see how to identify just what your program should do by writing program code. Finally, you'll learn how to save and run your new program (a Las Vegas–style slot machine) and how to compile it as an executable file.

Lucky Seven: Your First Visual Basic Program

The Windows-based application you're going to construct is Lucky Seven, a game program that simulates a lucky number slot machine. Lucky Seven has a simple user interface and can be created and compiled in just a few minutes using Microsoft Visual Basic. Here's what your program will look like when it's finished:



Programming Steps

The Lucky Seven user interface contains two buttons, three lucky number boxes, a digital photo depicting your winnings, and the label "Lucky Seven." I produced these elements by creating seven objects on the Lucky Seven form and then changing several properties for each object. After I designed the interface, I added program code for the Spin and End buttons to process the user's button clicks and produce the random numbers. To re-create Lucky Seven, you'll follow three essential programming steps in Visual Basic: Create the user interface, set the properties, and write the program code. Table 2-1 shows the process for Lucky Seven.

TABLE 2-1 Building the Lucky Seven Program

Programming Step	Number of Items
1. Create the user interface.	7 objects
2. Set the properties.	13 properties
3. Write the program code.	2 objects

Creating the User Interface

In this exercise, you'll start building Lucky Seven by first creating a new project and then using controls in the Toolbox to construct the user interface.

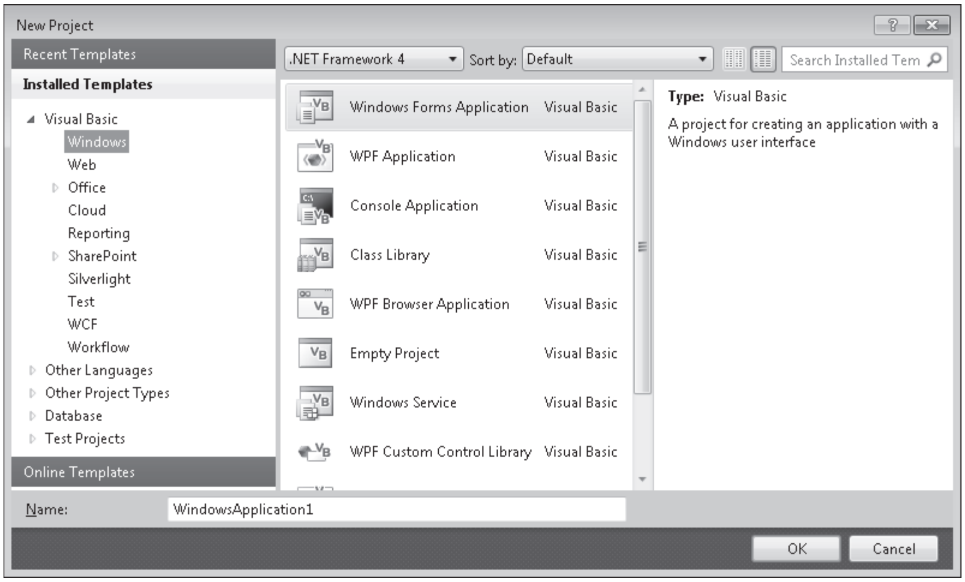
Create a new project

1. Start Visual Studio 2010.
2. On the Visual Studio File menu, click New Project.

Tip You can also start a new programming project by clicking the blue New Project link on the Start Page.

The New Project dialog box opens, as shown on the following page.

The New Project dialog box provides access to the major project types available for writing Windows and Web applications. If you indicated during setup that you are a Visual Basic programmer, Visual Basic is your primary development option (as shown here), but the other languages in Visual Studio (Visual C#, Visual C++, and Visual F#) are always available through this dialog box. Although you will select a basic Windows



Type: Visual Basic

A project for creating an application with a Windows user interface

Name: WindowsApplication1

application project in this exercise, this dialog box is also the gateway to other types of development projects, such as a Web application, console application, Microsoft Office add-in, Windows Azure Cloud Service, Silverlight application, or Visual Studio deployment project.

Near the top of the New Project dialog box, you will notice a drop-down list box. This feature allows you to specify the version of the Microsoft .NET Framework that your application will target. This feature is sometimes called *multi-targeting*, meaning that through it, you can select the target environment that your program will run on. For example, if you retain the default selection of .NET Framework 4, any computer that your application will run on must have .NET Framework 4 installed. (Not to worry—the .NET Framework is usually installed as part of the operating system installation, or when you install a new Visual Basic program that you have written.) Unless you have a specific need, you can just leave this drop-down list at its default setting of .NET Framework 4. Visual Basic 2010 Express does not include this drop-down list. You'll learn more about the .NET Framework in Chapter 5, "Visual Basic Variables and Formulas, and the .NET Framework."

3. Click the Windows Forms Application icon in the central Templates area of the dialog box, if it is not already selected.

   Visual Studio prepares the development environment for Visual Basic Windows application programming.

4. In the Name text box, type **MyLucky7**.

   Visual Studio assigns the name MyLucky7 to your project. (You'll specify a folder location for the project later.) I'm recommending the "My" prefix here so you don't confuse your new application with the Lucky7 project I've created for you on disk.

   Tip If your New Project dialog box contains Location and Solution Name text boxes, you need to specify a folder location and solution name for your new programming project now. The presence of these text boxes is controlled by a check box in the Project And Solutions category of the Options dialog box, but it is not the default setting. (You display this dialog box by clicking the Options command on the Tools menu.) Throughout this book, you will be instructed to save your projects (or discard them) *after* you have completed the programming exercise. For more information about this "delayed saving" feature and default settings, see the section entitled "Customizing IDE Settings to Match Step-by-Step Exercises" in Chapter 1.

5. Click OK to create the new project in Visual Studio.

   Visual Studio cleans the slate for a new programming project and displays the blank Windows form that you will use to build your user interface.

Now you'll enlarge the form and create the two buttons in the interface.

Create the user interface

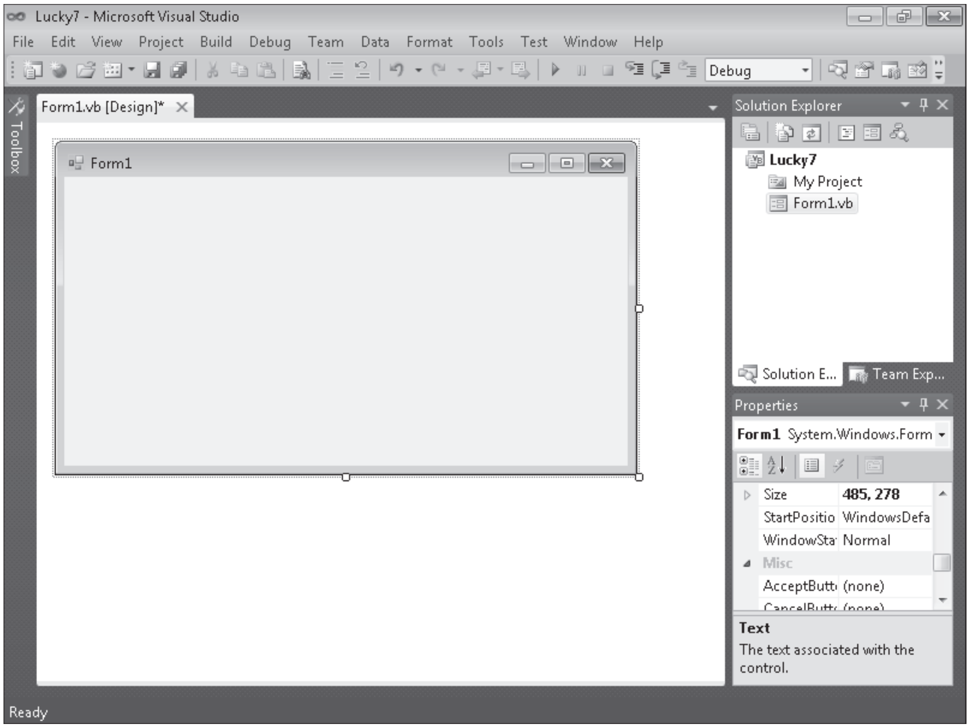
1. Point to the lower-right corner of the form until the mouse pointer changes to a resizing pointer, and then drag to increase the size of the form to make room for the objects in your program.

   As you resize the form, scroll bars might appear in the Designer to give you access to the entire form you're creating. Depending on your screen resolution and the Visual Studio tools you have open, you might not be able to see the entire form at once. Don't worry about this—your form can be small, or it can fill the entire screen because the scroll bars give you access to the entire form.

   Size your form so that it is about the size of the form shown on the following page. If you want to match my example exactly, you can use the width and height dimensions (485 pixels × 278 pixels) shown in the lower-right corner of the screen.

   To see the entire form without obstruction, you can resize or close the other programming tools, as you learned in Chapter 1. (Return to Chapter 1 if you have questions about resizing windows or tools.)

   Now you'll practice adding a button object on the form.

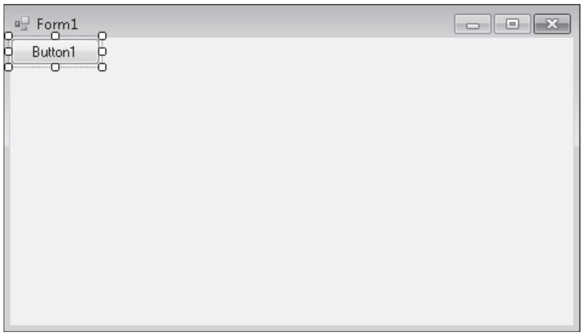


2. Click the Toolbox tab to display the Toolbox window in the IDE.

   The Toolbox contains all the controls that you'll use to build Visual Basic programs in this book. The controls suitable for creating a Windows application are visible now because you selected the Windows Application project type earlier. Controls are organized by type, and by default the Common Controls category is visible. (If the Toolbox is not visible now, click Toolbox on the View menu to display it.)

3. Double-click the *Button* control in the Toolbox, and then move the mouse pointer away from the Toolbox.

   Visual Studio creates a default-sized button object on the form and hides the Toolbox, as shown here:



The button is named *Button1* because it is the first button in the program. (You should make a mental note of this button name—you'll see it again when you write your program code.) The new button object is selected and enclosed by resize handles. When Visual Basic is in *design mode* (that is, whenever the Visual Studio IDE is active), you can move objects on the form by dragging them with the mouse, and you can resize them by using the resize handles. While a program is running, however, the user can't move user interface (UI) elements unless you've changed a property in the program to allow this. You'll practice moving and resizing the button now.

Move and resize a button

1. Point to the button so that the pointer changes to a four-headed arrow, and then drag the button down and to the right.

   The button moves across the surface of the form. If you move the object near the edge of the form or another object (if other objects are present), it automatically aligns itself to a hidden grid when it is an inch or so away. A little blue "snapline" also appears to help you gauge the distance of this object from the edge of the form or the other object. The grid is not displayed on the form by default, but you can use the snapline to judge distances with almost the same effect.

2. Position the mouse pointer on the lower-right corner of the button.

   When the mouse pointer rests on a resize handle of a selected object, it becomes a resizing pointer. You can use the resizing pointer to change the size of an object.

3. Enlarge the button by dragging the pointer down and to the right.

   When you release the mouse button, the button changes size and snaps to the grid.

4. Use the resizing pointer to return the button to its original size.

Now you'll add a second button to the form, below the first button.

Add a second button

1. Click the Toolbox tab to display the Toolbox.

2. Click the *Button* control in the Toolbox (single-click this time), and then move the mouse pointer over the form.

   The mouse pointer changes to crosshairs and a button icon. The crosshairs are designed to help you draw the rectangular shape of the button on the form, and you can use this method as an alternative to double-clicking to create a control of the default size.

3. Click and drag the pointer down and to the right. Release the mouse button to complete the button, and watch it snap to the form.

4. Resize the button object so that it is the same size as the first button, and then move it below the first button on the form. (Use the snapline feature to help you.)

> **Tip** At any time, you can delete an object and start over again by selecting the object on the form and then pressing DELETE. Feel free to create and delete objects to practice creating your user interface.

Now you'll add the labels used to display the numbers in the program. A *label* is a special user interface element designed to display text, numbers, or symbols when a program runs. When the user clicks the Lucky Seven program's Spin button, three random numbers appear in the label boxes. If one of the numbers is a 7, the user wins.

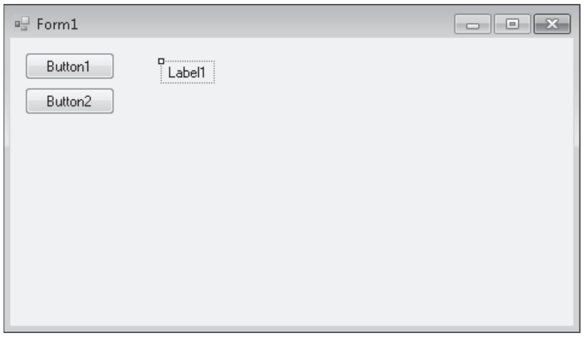
Add the number labels

1. Double-click the *Label* control in the Toolbox.

   Visual Studio creates a label object on the form. The label object is just large enough to hold the text contained in the object (it is rather small now), but it can be resized.

2. Drag the *Label1* object to the right of the two button objects.

   Your form looks something like this:



3. Double-click the *Label* control in the Toolbox to create a second label object.

   This label object will be named *Label2* in the program.

4. Double-click the *Label* control again to create a third label object.

5. Move the second and third label objects to the right of the first one on the form.

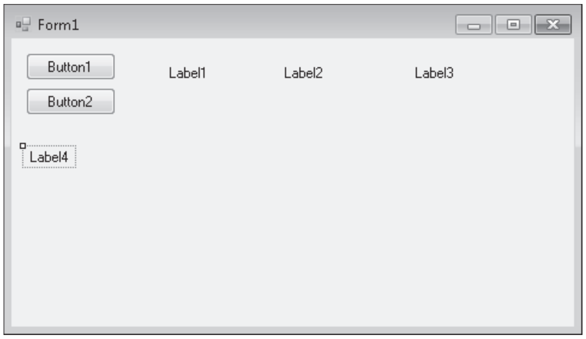
   Allow plenty of space between the three labels because you will use them to display large numbers when the program runs.

   Now you'll use the *Label* control to add a descriptive label to your form. This will be the fourth and final label in the program.

6. Double-click the *Label* control in the Toolbox.

7. Drag the *Label4* object below the two command buttons.

   When you've finished, your four labels should look like those in the following screen shot. (You can move your label objects if they don't look quite right.)

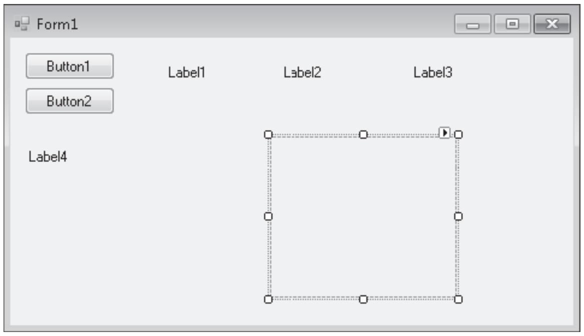


Now you'll add a picture box to the form to graphically display the payout you'll receive when you draw a 7 and hit the jackpot. A *picture box* is designed to display bitmaps, icons, digital photos, and other artwork in a program. One of the best uses for a picture box is to display a JPEG image file.

Add a picture

1. Click the *PictureBox* control in the Toolbox.
2. Using the control's drawing pointer, create a large rectangular box below the second and third labels on the form.

   Leave a little space below the labels for their size to grow as I mentioned earlier. When you've finished, your picture box object looks similar to this:



   This object will be named *PictureBox1* in your program; you'll use this name later in the program code.

Now you're ready to customize your interface by setting a few properties.

Setting the Properties

As you discovered in Chapter 1, you can change properties by selecting objects on the form and changing their settings in the Properties window. You'll start by changing the property settings for the two buttons.

Set the button properties

1. Click the first button (*Button1*) on the form.

   The button is selected and is surrounded by resize handles.

2. Click the Properties window title bar.

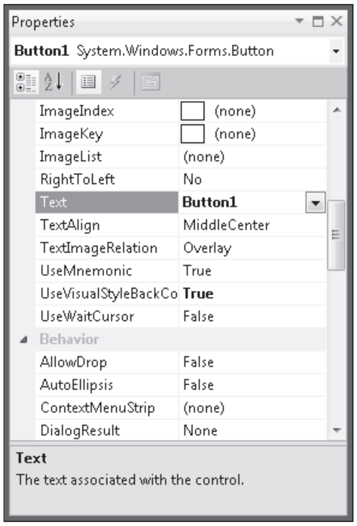
Tip If the Properties window isn't visible, click the Properties Window command on the View menu, or press F4.

3. At the top of the Properties window, click the Categorized button.

   For information about categorized properties, see the section entitled "The Properties Window" in Chapter 1.

4. Resize the Properties window (if necessary) so that there is plenty of room to see the property names and their current settings.

   Once you get used to setting properties, you will probably use the Properties window without enlarging it, but making it bigger helps when you first try to use it. The Properties window in the following screen shot is a good size for setting properties:



ImageIndex	(none)
ImageKey	(none)
ImageList	(none)
RightToLeft	No
Text	Button1
TextAlign	MiddleCenter
TextImageRelation	Overlay
UseMnemonic	True
UseVisualStyleBackCo	True
UseWaitCursor	False
Behavior	
AllowDrop	False
AutoEllipsis	False
ContextMenuStrip	(none)
DialogResult	None

Text
The text associated with the control.

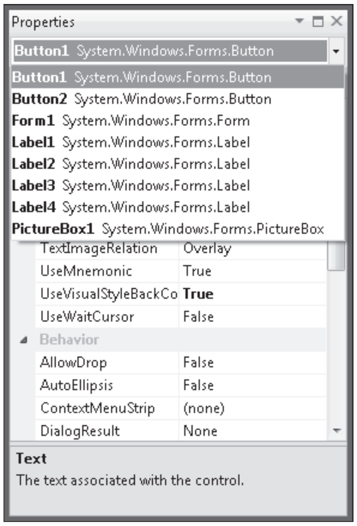
The Properties window lists the settings for the first button. These include settings for the background color, text, font height, and width of the button. Because there are so many properties, Visual Studio organizes them into categories and displays them in outline view. If you want to see the properties in a category, click the arrow sign (>) next to the category title.

5. If it is not already visible, scroll in the Properties window until you see the *Text* property located in the Appearance category.
6. Double-click the *Text* property in the first column of the Properties window.

   The current *Text* setting ("Button1") is highlighted in the Properties window.
7. Type **Spin**, and then press ENTER.

   The *Text* property changes to "Spin" in the Properties window and on the button on the form. Now you'll change the *Text* property of the second button to "End." (You'll select the second button in a new way this time.)
8. Open the Object list at the top of the Properties window.

   A list of the interface objects in your program appears as follows:



```
Button1  System.Windows.Forms.Button

Button1  System.Windows.Forms.Button
Button2  System.Windows.Forms.Button
Form1  System.Windows.Forms.Form
Label1  System.Windows.Forms.Label
Label2  System.Windows.Forms.Label
Label3  System.Windows.Forms.Label
Label4  System.Windows.Forms.Label
PictureBox1  System.Windows.Forms.PictureBox
```

TextImageRelation	Overlay
UseMnemonic	True
UseVisualStyleBackCo	**True**
UseWaitCursor	False
Behavior	
AllowDrop	False
AutoEllipsis	False
ContextMenuStrip	(none)
DialogResult	None

Text
The text associated with the control.

9. Click Button2 System.Windows.Forms.Button (the second button) in the list box.

   The property settings for the second button appear in the Properties window, and Visual Studio highlights Button2 on the form.
10. Double-click the current *Text* property ("Button2"), type **End**, and then press ENTER.

   The text of the second button changes to "End."

Tip Using the Object list is a handy way to switch between objects in your program. You can also switch between objects on the form by clicking each object.

Now you'll set the properties for the labels in the program. The first three labels will hold the random numbers generated by the program and will have identical property settings. (You'll set most of them as a group.) The descriptive label settings will be slightly different.

Set the number label properties

1. Click the first number label (*Label1*), hold down the SHIFT key, click the second and third number labels, and then release the SHIFT key. (If the Properties window is in the way, move it to a new place.)

   A selection rectangle and resize handles appear around each label you click. You'll change the *TextAlign*, *BorderStyle*, and *Font* properties now so that the numbers that will appear in the labels will be centered, boxed, and identical in font and font size. (All these properties are located in the Appearance category of the Properties window.) You'll also set the *AutoSize* property to False so that you can change the size of the labels according to your precise specifications. (The *AutoSize* property is located in the Layout category.)

Tip When more than one object is selected, only those properties that can be changed for the group are displayed in the Properties window.

2. Click the *AutoSize* property in the Properties window, and then click the arrow that appears in the second column.
3. Set the *AutoSize* property to False so that you can size the labels manually.
4. Click the *TextAlign* property, and then click the arrow that appears in the second column.

   A graphical assortment of alignment options appears in the list box; you can use these settings to align text anywhere within the borders of the label object.
5. Click the center option (MiddleCenter).

   The *TextAlign* property for each of the selected labels changes to MiddleCenter.
6. Click the *BorderStyle* property, and then click the arrow that appears in the second column.

   The valid property settings (None, FixedSingle, and Fixed3D) appear in the list box.
7. Click FixedSingle in the list box to add a thin border around each label.

8. Click the *Font* property, and then click the ellipsis button (the button with three dots that's located next to the current font setting).

   The Font dialog box opens.

9. Change the font to Times New Roman, the font style to Bold, and the font size to 24, and then click OK.

   The label text appears in the font, style, and size you specified.

   Now you'll set the text for the three labels to the number 0—a good "placeholder" for the numbers that will eventually fill these boxes in your game. (Because the program produces the actual numbers, you could also delete the text, but putting a placeholder here gives you something to base the size of the labels on.)

10. Click a blank area on the form to remove the selection from the three labels, and then click the first label.

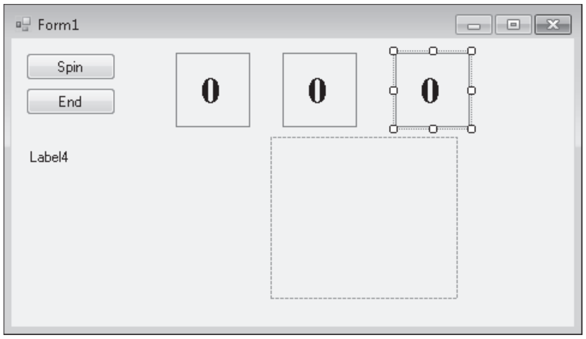
11. Double-click the *Text* property, type **0**, and then press ENTER.

    The text of the *Label1* object is set to 0. You'll use program code to set this property to a random "slot machine" number later in this chapter.

12. Change the text in the second and third labels on the form to **0** also.

13. Move and resize the labels now so that they are appropriately spaced.

    Your form looks something like this:



Now you'll change the *Text*, *Font*, and *ForeColor* properties of the fourth label.

Set the descriptive label properties

1. Click the fourth label object (*Label4*) on the form.

2. Change the *Text* property in the Properties window to **Lucky Seven**.

3. Click the *Font* property, and then click the ellipsis button.

4. Use the Font dialog box to change the font to Arial, the font style to Bold, and the font size to 18. Then click OK.

   The font in the *Label4* object is updated, and the label is resized automatically to hold the larger font size because the object's *AutoSize* property is set to True.

5. Click the *ForeColor* property in the Properties window, and then click the arrow in the second column.

   Visual Studio displays a list box with Custom, Web, and System tabs for setting the foreground colors (the color of text) of the label object. The Custom tab offers many of the colors available in your system. The Web tab sets colors for Web pages and lets you pick colors using their common names. The System tab displays the current colors used for user interface elements in your system.

6. Click the purple color on the Custom tab.

   The text in the label box changes to purple.

Now you're ready to set the properties for the last object.

The Picture Box Properties

When the person playing your game hits the jackpot (that is, when at least one 7 appears in the number labels on the form), the picture box object will contain a picture in JPEG format of a person dispensing money. (I am supplying you with this digitized image, but you can substitute your own if you like.) You need to set the *SizeMode* property to accurately size the picture and set the *Image* property to specify the name of the JPEG file that you will load into the picture box. You also need to set the *Visible* property, which specifies the picture state at the beginning of the program.

Set the picture box properties

1. Click the picture box object on the form.
2. Click the *SizeMode* property in the Properties window (listed in the Behavior category), click the arrow in the second column, and then click StretchImage.

   Setting *SizeMode* to StretchImage before you open a graphic causes Visual Studio to resize the graphic to the exact dimensions of the picture box. (Typically, you set this property before you set the *Image* property.)

3. Click the *Image* property in the Properties window, and then click the ellipsis button in the second column.

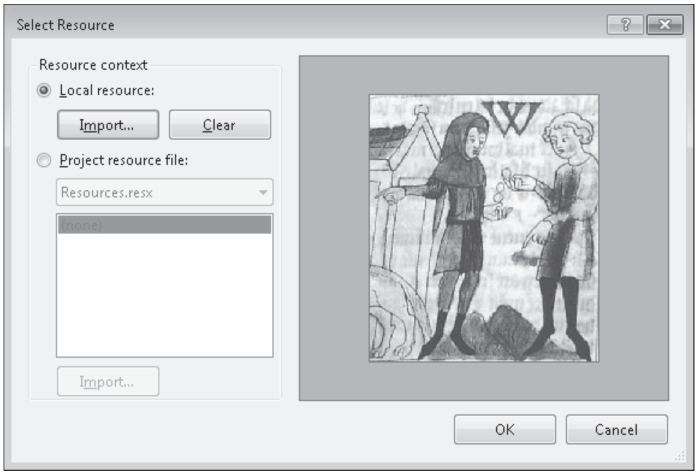
   The Select Resource dialog box opens.

4. Click the Local Resource radio button, and then click the Import button.
5. In the Open dialog box, navigate to the C:\Vb10sbs\Chap02 folder.

   This folder contains the digital photo PayCoins.jpg.

6. Select PayCoins.jpg, and then click Open.

   An screen shot of one person paying another appears in the Select Resource dialog box. (The letter "W" represents winning.)



7. Click OK.

   The PayCoins photo is loaded into the picture box. Because the photo is relatively small (24 KB), it opens quickly on the form.

8. Resize the picture box object now to fix any distortion problems that you see in the image.

   I sized my picture box object to be 144 pixels wide by 146 pixels high. You can match this size by using the width and height dimensions located on the lower-right side of the Visual Studio IDE. (The dimensions of the selected object are given on the lower-right side, and the location on the form of the object's upper-left corner is given to the left of the dimensions.)

   This particular image displays best when the picture box object retains a square shape.

Note As you look at the picture box object, you might notice a tiny shortcut arrow called a *smart tag* near its upper-right corner. This smart tag is a button that you can click to quickly change a few common picture box settings and open the Select Resource dialog box. (You'll see the smart tag again in Chapter 4, "Working with Menus, Toolbars, and Dialog Boxes," when you use the *ToolStrip* control.)

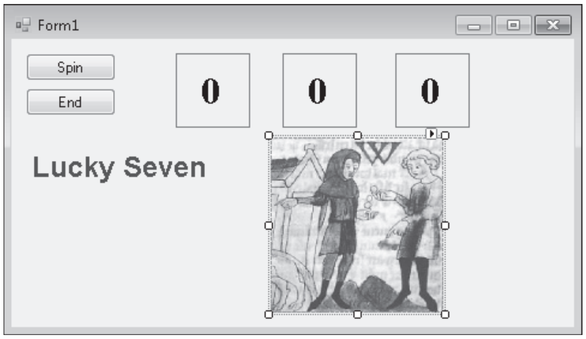
Now you'll change the *Visible* property to False so that the image will be invisible when the program starts.

9. Click the *Visible* property in the Behavior category of the Properties window, and then click the arrow in the second column.

   The valid settings for the *Visible* property appear in a list box.

10. Click False to make the picture invisible when the program starts.

Setting the *Visible* property to False affects the picture box when the program runs, but not now, while you're designing it. Your completed form looks similar to this:



Tip You can also double-click property names that have True and False settings (so-called Boolean properties), to toggle back and forth between True and False. Default Boolean properties are shown in regular type, and changed settings appear in bold.

11. You are finished setting properties for now, so if your Properties window is floating, hold down the CTRL key and double-click its title bar to return it to the docked position.

Reading Properties in Tables

In this chapter, you've set the properties for the Lucky Seven program step by step. In future chapters, the instructions to set properties will be presented in table format unless a setting is especially tricky. Table 2-2 lists the properties you've set so far in the Lucky Seven program, as they'd look later in the book. Settings you need to type in are shown in quotation marks. You shouldn't type the quotation marks.

TABLE 2-2 Lucky Seven Properties

Object	Property	Setting
Button1	*Text*	"Spin"
Button2	*Text*	"End"
Label1, Label2, Label3	*AutoSize* *BorderStyle* *Font* *Text* *TextAlign*	False FixedSingle Times New Roman, Bold, 24-point "0" MiddleCenter
Label4	*Text* *Font* *ForeColor*	"Lucky Seven" Arial, Bold, 18-point Purple
PictureBox1	*Image* *SizeMode* *Visible*	"C:\Vb10sbs\Chap02\Paycoins.jpg" StretchImage False

Writing the Code

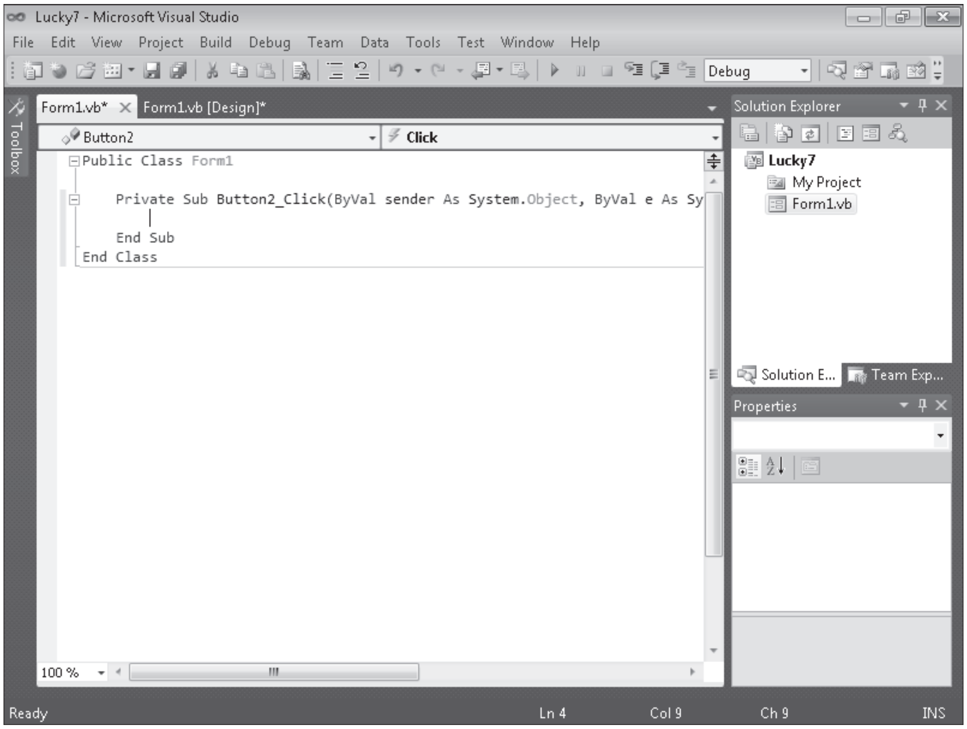
Now you're ready to write the code for the Lucky Seven program. Because most of the objects you've created already "know" how to work when the program runs, they're ready to receive input from the user and process it. The inherent functionality of objects is one of the great strengths of Visual Studio and Visual Basic—after objects are placed on a form and their properties are set, they're ready to run without any additional programming. However, the "meat" of the Lucky Seven game—the code that actually calculates random numbers, displays them in boxes, and detects a jackpot—is still missing from the program. This computing logic can be built into the application only by using program statements—code that clearly spells out what the program should do at each step of the way. Because the Spin and End buttons drive the program, you'll associate the code for the game with those buttons. You enter and edit Visual Basic program statements in the Code Editor.

In the following steps, you'll enter the program code for Lucky Seven in the Code Editor.

Use the Code Editor

1. Double-click the End button on the form.

   The Code Editor appears as a tabbed document window in the center of the Visual Studio IDE, as shown here:



```
Public Class Form1

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As Sy

    End Sub
End Class
```

Inside the Code Editor are program statements associated with the current form. Program statements that are used together to perform some action are typically grouped in a programming construct called a *procedure*. A common type of procedure is a Sub procedure, sometimes called a *subroutine*. Sub procedures include a *Sub* keyword in the first line and end with *End Sub*. (I'll talk about the Public and Private keywords later.) Procedures are typically executed when certain events occur, such as when a button is clicked. When a procedure is associated with a particular object and an event, it is called an *event handler* or an *event procedure*.

When you double-clicked the End button (*Button2*), Visual Studio automatically added the first and last lines of the *Button2_Click* event procedure, as the following code shows. (The first line was wrapped to stay within the book margins.) You may notice other bits of code in the Code Editor (words like *Public* and *Class*), which Visual Studio has added to define important characteristics of the form, but I won't emphasize them here.

```
Private Sub Button2_Click(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles Button2.Click
End Sub
```

The body of a procedure fits between these lines and is executed whenever a user activates the interface element associated with the procedure. In this case, the event is a mouse click, but as you'll see later in the book, it could also be a different type of event.

2. Type **End**, and then press the ENTER key.

   When you type the statement, Visual Studio recognizes *End* as a unique reserved word or *keyword* and displays it in a list box with Common and All tabs. Microsoft calls this auto-extend feature IntelliSense because it tries to intelligently help you write code, and you can browse through various Visual Basic keywords and objects alphabetically. (In this way, the language is partially discoverable through the IDE itself.)

   After you press the ENTER key, the letters in *End* turn blue and are indented, indicating that Visual Basic recognizes *End* as one of several hundred unique keywords within the Visual Basic language. You use the *End* keyword to stop your program and remove it from the screen. In this case, *End* is also a complete *program statement*, a self-contained instruction executed by the *Visual Basic compiler*, the part of Visual Studio that processes or *parses* each line of Visual Basic *source code*, combining the result with other resources to create an executable file. Program statements are a little like complete sentences in a human language—statements can be of varying lengths but must follow the grammatical "rules" of the compiler. In Visual Studio, program statements can be composed of keywords, properties, object names, variables, numbers, special symbols, and other values. You'll learn more about how program statements are constructed in Chapter 5.

   As you enter program statements and make other edits, the Code Editor handles many of the formatting details for you, including adjusting indentation and spacing and adding any necessary parentheses. The exact spelling, order, and spacing of items within program statements is referred to as *statement syntax*. In the early days of compilers, programmers were almost totally responsible for getting the precise syntax for each program statement correct on their own, but now sophisticated development tools such as Visual Studio help immensely with the construction of accurate program statements.

When you pressed the ENTER key, the *End* statement was indented to set it apart from the *Private Sub* and *End Sub* statements. This indenting scheme is one of the programming conventions you'll see throughout this book to keep your programs clear and readable. The group of conventions regarding how code is organized in a program is often referred to as *program style*.

Now that you've written the code associated with the End button, you'll write code for the Spin button. These program statements will be a little more extensive and will give you a chance to learn more about statement syntax and program style. You'll study many of the program statements later in this book, so you don't need to know everything about them now. Just focus on the general structure of the code and on typing the program statements exactly as they are printed.

Write code for the Spin button

1. At the top of the Solution Explorer window, click the View Designer button in the Solution Explorer window to display your form again.

Note When the Code Editor is visible, you won't be able to see the form you're working on. The View Designer button is one mechanism you can use to display it again. (If more than one form is loaded in Solution Explorer, click the form that you want to display first.) You can also click the Form1.vb [Design] tab at the top edge of the Code Editor. To display the Code Editor again, click the View Code button in Solution Explorer.

2. Double-click the Spin button.

   After a few moments, the Code Editor appears, and an event procedure associated with the *Button1* button appears near the *Button2* event procedure.

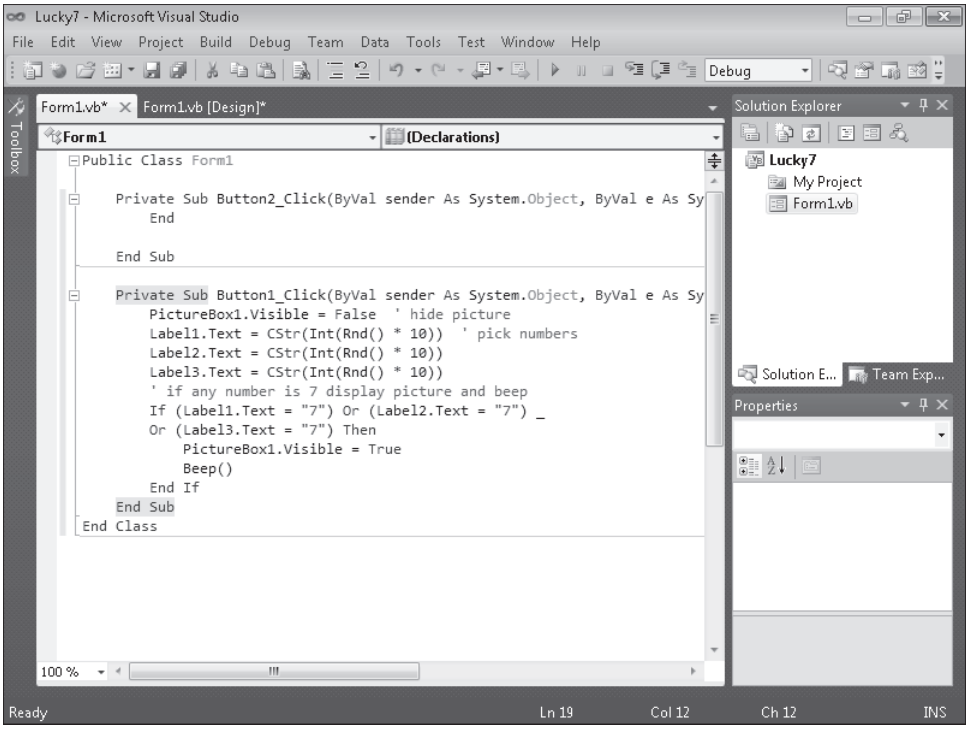
   Although you changed the text of this button to "Spin," its name in the program is still *Button1*. (The name and the text of an interface element can be different to suit the needs of the programmer.) Each object can have several procedures associated with it, one for each event it recognizes. The click event is the one you're interested in now because users will click the Spin and End buttons when they run the program.

3. Type the following program lines between the *Private Sub* and *End Sub* statements. Press ENTER after each line, press TAB to indent, and take care to type the program statements exactly as they appear here. (The Code Editor will scroll to the left as you enter the longer lines.) If you make a mistake (usually identified by a jagged underline), delete the incorrect statements and try again.

Tip As you enter the program code, Visual Basic formats the text and displays different parts of the program in color to help you identify the various elements. When you begin to type a property, Visual Basic also displays the available properties for the object you're using in a list box, so you can double-click the property or keep typing to enter it yourself. If Visual Basic displays an error message, you might have misspelled a program statement. Check the line against the text in this book, make the necessary correction, and continue typing. (You can also delete a line and type it from scratch.) In addition, Visual Basic might add necessary code automatically. For example, when you type the following code, Visual Basic automatically adds the *End If* line. Readers of previous editions of this book have found this first typing exercise to be the toughest part of this chapter—"But Mr. Halvorson, I know I typed it just as you wrote it!"—so please give this program code your closest attention. I promise you, it works!

```
PictureBox1.Visible = False  ' hide picture
Label1.Text = CStr(Int(Rnd() * 10))  ' pick numbers
Label2.Text = CStr(Int(Rnd() * 10))
Label3.Text = CStr(Int(Rnd() * 10))
' if any number is 7 display picture and beep
If (Label1.Text = "7") Or (Label2.Text = "7") _
Or (Label3.Text = "7") Then
   PictureBox1.Visible = True
   Beep()
End If
```

When you've finished, the Code Editor looks as shown in the following screen shot:


Lucky7 - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Test Window Help
Debug
Form1.vb* Form1.vb [Design]*
Form1
(Declarations)
Solution Explorer
Lucky7
My Project
Form1.vb
Solution E... Team Exp...
Properties
Toolbox
100 %
Ready Ln 19 Col 12 Ch 12 INS

```
Public Class Form1

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As Sy
        End

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As Sy
        PictureBox1.Visible = False  ' hide picture
        Label1.Text = CStr(Int(Rnd() * 10))  ' pick numbers
        Label2.Text = CStr(Int(Rnd() * 10))
        Label3.Text = CStr(Int(Rnd() * 10))
        ' if any number is 7 display picture and beep
        If (Label1.Text = "7") Or (Label2.Text = "7") _
        Or (Label3.Text = "7") Then
            PictureBox1.Visible = True
            Beep()
        End If
    End Sub
End Class
```

4. Click the Save All command on the File menu to save your additions to the program.

   The Save All command saves everything in your project—the project file, the form file, any code modules, and other related components in your application. Since this is the first time that you have saved your project, the Save Project dialog box opens, prompting you for the name and location of the project. (If your copy of Visual Studio is configured to prompt you for a location when you first create your project, you won't see the Save Project dialog box now—Visual Studio just saves your changes.)

5. Browse and select a location for your files.

   I recommend that you use the C:\Vb10sbs\Chap02 folder (the location of the book's sample files), but the location is up to you. Since you used the "My" prefix when you originally opened your project, this version won't overwrite the Lucky7 practice file that I built for you on disk.

6. Clear the Create Directory For Solution check box.

   When this check box is selected, it creates a second folder for your program's solution files, which is not necessary for solutions that contain only one project (the situation for most programs in this book).

7. Click Save to save your files.

Note If you want to save just the item you are currently working on (the form, the code module, or something else), you can use the Save command on the File menu. If you want to save the current item with a different name, you can use the Save As command.

A Look at the *Button1_Click* Procedure

The *Button1_Click* procedure is executed when the user clicks the Spin button on the form. The procedure uses some pretty complicated statements, and because I haven't formally introduced them yet, it might look a little confusing. However, if you take a closer look, you'll probably see a few things that look familiar. Taking a peek at the contents of these procedures will give you a feel for the type of program code you'll be creating later in this book. (If you'd rather not stop for this preview, feel free to skip to the next section, "Running Visual Basic Applications.")

The *Button1_Click* procedure performs three tasks:

- It hides the digital photo.
- It creates three random numbers for the number labels.
- It displays the photo when the number 7 appears.

Let's look at each of these steps individually.

Hiding the photo is accomplished with the following line:

```
PictureBox1.Visible = False  ' hide picture
```

This line is made up of two parts: a program statement and a comment.

The *PictureBox1.Visible = False* program statement sets the *Visible* property of the picture box object (*PictureBox1*) to False (one of two possible settings). You might remember that you set this property to False once before by using the Properties window. You're doing it again now in the program code because the first task is a spin and you need to clear away a photo that might have been displayed in a previous game. Because the property will be changed at run time and not at design time, you must set the property by using program code. This is a handy feature of Visual Basic, and I'll talk about it more in Chapter 3, "Working with Toolbox Controls."

The second part of the first line (the part displayed in green type on your screen) is called a *comment*. Comments are explanatory notes included in program code following a single quotation mark ('). Programmers use comments to describe how important statements work in a program. These notes aren't processed by Visual Basic when the program runs; they exist only to document what the program does. You'll want to use comments often when you write Visual Basic programs to leave an easy-to-understand record of what you're doing.

The next three lines handle the random number computations. Does this concept sound strange? You can actually make Visual Basic generate unpredictable numbers within specific guidelines—in other words, you can create random numbers for lottery contests, dice games, or other statistical patterns. The *Rnd* function in each line creates a random number between 0 and 1 (a number with a decimal point and several decimal places), and the *Int* function returns the integer portion of the result of multiplying the random number by 10. This computation creates random numbers between 0 and 9 in the program—just what you need for this particular slot machine application.

```
Label1.Text = CStr(Int(Rnd() * 10))  ' pick numbers
```

You then need to jump through a little hoop in your code. You need to copy these random numbers into the three label boxes on the form, but first the numbers need to be converted to text with the *CStr* (convert to string) function. Notice how *CStr*, *Int*, and *Rnd* are all connected in the program statement—they work collectively to produce a result like a mathematical formula. After the computation and conversion, the values are assigned to the *Text* properties of the first three labels on the form, and the assignment causes the numbers to be displayed in bold, 24-point, Times New Roman font in the three number labels.

The last group of statements in the program checks whether any of the random numbers is 7. If one or more of them is, the program displays the graphical depiction of a payout, and a beep announces the winnings.

```
' if any number is 7 display picture and beep
If (Label1.Text = "7") Or (Label2.Text = "7") _
Or (Label3.Text = "7") Then
    PictureBox1.Visible = True
    Beep()
End If
```

Each time the user clicks the Spin button, the *Button1_Click* procedure is executed, or *called*, and the program statements in the procedure are run again.

Running Visual Basic Applications

Congratulations! You're ready to run your first real program. To run a Visual Basic program from the development environment, you can do any of the following:

- Click Start Debugging on the Debug menu.
- Click the Start Debugging button on the Standard toolbar.
- Press F5.

Try running your Lucky Seven program now. If Visual Basic displays an error message, you might have a typing mistake or two in your program code. Try to fix it by comparing the printed version in this book with the one you typed, or load Lucky7 from your hard disk and run it.

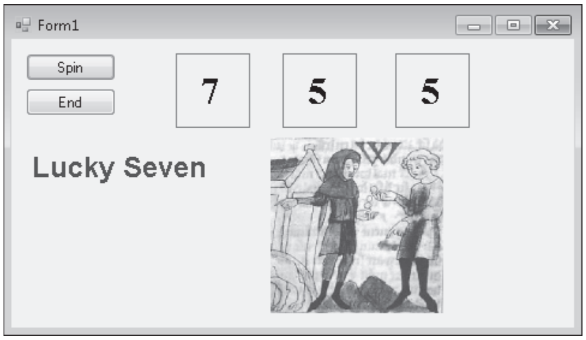
Run the Lucky Seven program

1. Click the Start Debugging button on the Standard toolbar.

   The Lucky Seven program compiles and runs in the IDE. After a few seconds, the user interface appears, just as you designed it.

2. Click the Spin button.

   The program picks three random numbers and displays them in the labels on the form, as follows:



Because a 7 appears in the first label box, the digital photo depicting the payoff appears, and the computer beeps. You win! (The sound you hear depends on your Default Beep setting in the Sound Control Panel. To make this game sound really cool, change the Default Beep sound to something more dynamic.)

3. Click the Spin button 15 or 16 more times, watching the results of the spins in the number boxes.

   About half the time you spin, you hit the jackpot—pretty easy odds. (The actual odds are about 2.8 times out of 10; you're just lucky at first.) Later on, you might want to make the game tougher by displaying the photo only when two or three 7s appear, or by creating a running total of winnings.

4. When you've finished experimenting with your new creation, click the End button.

   The program stops, and the development environment reappears on your screen.

Tip If you run this program again, you might notice that Lucky Seven displays exactly the same sequence of random numbers. There is nothing wrong here—the Visual Basic *Rnd* function was designed to display a *repeating* sequence of numbers at first so that you can properly test your code using output that can be reproduced again and again. To create truly "random" numbers, use the *Randomize* function in your code, as shown in the exercise at the end of this chapter. The .NET Framework, which you'll learn to use later, also supplies random number functions.

Sample Projects on Disk

If you didn't build the MyLucky7 project from scratch (or if you did build the project and want to compare what you created to what I built for you as I wrote the chapter), take a moment to open and run the completed Lucky7 project, which is located in the C:\Vb10sbs\Chap02\Lucky7 folder on your hard disk (the default location for the practice files for this chapter). If you need a refresher course on opening projects, see the detailed instructions in Chapter 1. If you are asked if you want to save changes to the MyLucky7 project, be sure to click Save.

This book is a step-by-step tutorial, so you will benefit most from building the projects on your own and experimenting with them. But after you have completed the projects, it is often a good idea to compare what you have with the practice file "solution" that I provide, especially if you run into trouble. To make this easy, I will give you the name of the solution files on disk before you run the completed program in most of the step-by-step exercises.

After you have compared the MyLucky7 project to the Lucky7 solution files on disk, reopen MyLucky7 and prepare to compile it as an executable file. If you didn't create MyLucky7, use my solution file to complete the exercise.

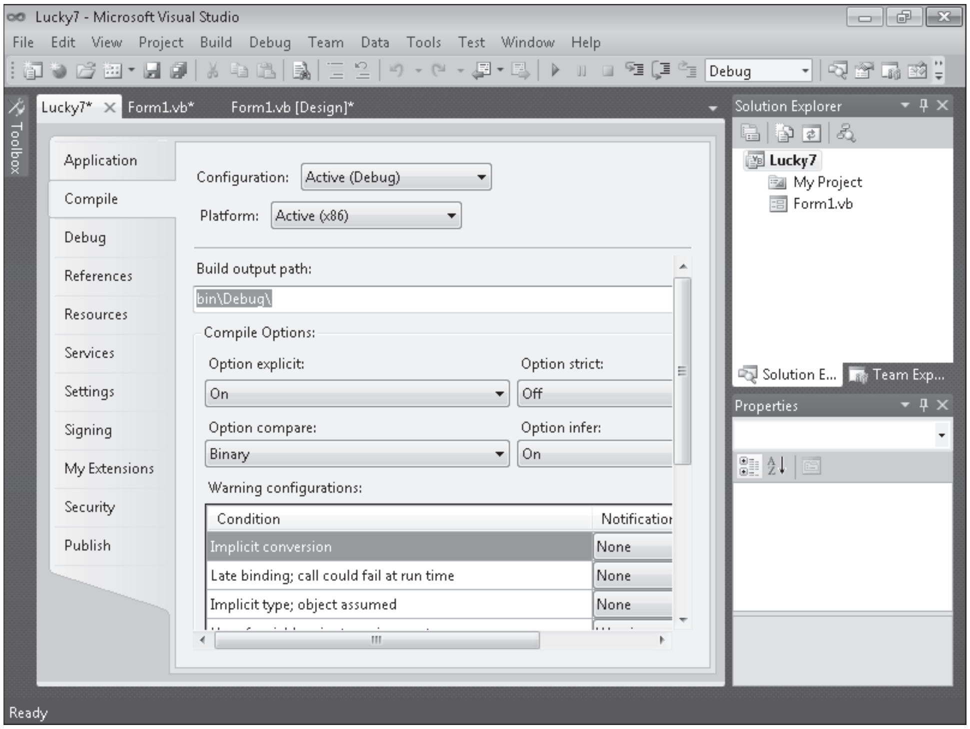
Building an Executable File

Your last task in this chapter is to complete the development process and create an application for Windows, or an *executable file*. (Had you created a different project type, of course, such as a Web application, the result of your development efforts would have been a different type of file—but we'll discuss this later.) Windows applications created with Visual Studio have the file name extension .exe and can be run on any system that contains Windows and the necessary support files. (Visual Basic installs these support files—including the .NET Framework files—automatically.) If you plan to distribute your applications, see the section entitled "Deploying Your Application" later in the chapter.

At this point, you need to know that Visual Studio can create two types of executable files for your Windows application project: a debug build and a release build.

Debug builds are created automatically by Visual Studio when you create and test your program. They are stored in a folder called Bin\Debug within your project folder. The debug executable file contains debugging information that makes the program run slightly slower.

Release builds are optimized executable files stored in the Bin\Release folder within your project. To customize the settings for your release build, you click the [*ProjectName*] Properties command on the Project menu, and then click the Compile tab, where you see a list of compilation options that looks like this:



Configuration: Active (Debug)
Platform: Active (x86)
Build output path: bin\Debug\
Compile Options:
Option explicit: On
Option strict: Off
Option compare: Binary
Option infer: On
Warning configurations:

Condition	Notification
Implicit conversion	None
Late binding; call could fail at run time	None
Implicit type; object assumed	None

Try creating a release build named MyLucky7.exe now.

Create an executable file

1. On the Build menu, click the Build MyLucky7 command.

   The Build command creates a Bin\Release folder in which to store your project (if the folder doesn't already exist) and compiles the source code in your project. The result is an executable file of the Application type named MyLucky7.exe. To save you time, Visual Studio often creates temporary executable files while you develop your application; however, it's always a good idea to recompile your application manually with the Build or Rebuild command when you reach an important milestone.

   Try running this program outside the Visual Studio IDE now from the Windows Start menu.

2. On the Windows taskbar, click Start.

   The next command depends on the version of Windows you're using.

3. If you have Windows 7 or Windows Vista, type **run** in the Search text box and press ENTER to open the Run dialog box. If you have Windows XP or earlier, click the Run command to open the Run dialog box.
4. Click Browse and then navigate to the C:\Vb10sbs\Chap02\Mylucky7\Bin\Release folder.
5. Click the MyLucky7.exe application icon, click Open, and then click OK.

   The Lucky Seven program loads and runs in Windows. Because this is a simple test application and it does not possess a formal publisher certificate that emphasizes its reliability or authenticity, you may see the following message: "The publisher could not be verified. Are you sure you want to run this software?" If this happens to you, click Yes to run the program anyway. (Creating such certificates is beyond the scope of this chapter, but this program is quite safe.)

6. Click Spin a few times to verify the operation of the game, and then click End.

Tip You can also run Windows applications, including compiled Visual Basic programs, by opening Windows Explorer and double-clicking the executable file. To create a shortcut icon for MyLucky7.exe on the Windows desktop, right-click the Windows desktop, point to New, and then click Shortcut. When you're prompted for the location of your application file, click Browse, and select the MyLucky7.exe executable file. Click the OK, Next, and Finish buttons. Windows places an icon on the desktop that you can double-click to run your program.

7. On the File menu, click Exit to close Visual Studio and the MyLucky7 project.

   The Visual Studio development environment closes.

Deploying Your Application

Visual Studio helps you distribute your Visual Basic applications by providing several options for *deployment*—that is, for installing the application on one or more computer systems. Since the release of Visual Studio in 2002, Visual Basic applications have been compiled as assemblies—deployment units consisting of one or more files necessary for the program to run. *Assemblies* contain four elements: Microsoft intermediate language (MSIL) code, metadata, a manifest, and supporting files and resources. Visual Studio 2010 continues to offer this same basic deployment architecture, with some noteworthy improvements for different platforms and application types.

How do assemblies actually work? First, assemblies are so comprehensive and self-describing that Visual Studio applications don't actually need to be formally registered with the operating system to run. This means that theoretically a Visual Basic 2010 application can be installed by simply copying the assembly for the program to a second computer that has the correct version of the .NET Framework installed—a process called *XCOPY installation*, after the MS-DOS XCOPY command that copies a complete directory (folder) structure from one location to another. In practice, however, it isn't practical to deploy Visual Basic applications by using a copy procedure such as XCOPY (via the command prompt) or Windows Explorer. For commercial applications, an installation program with a graphical user interface is usually preferred, and it's often desirable to register the program with the operating system so that it can be uninstalled later by using Control Panel. In addition, it is often useful to take advantage of the Web for an application's initial deployment and to have an application check the Web periodically for updates.

Although the advanced options related to deployment and security go beyond the scope of this book, you should be familiar with your deployment options. To manage the deployment process, Visual Studio 2010 supports two deployment technologies, *ClickOnce* and *Windows Installer*.

Essentially, ClickOnce is a robust Web-based publishing technology that allows you to control how applications are made available to users via the Internet, although ClickOnce installations can also be distributed via CD-ROM. With ClickOnce, you can create an installation service for Windows applications, Office solutions, or console applications that users can access on their own with minimal interaction. With ClickOnce, you can specify prerequisites, such as a particular version of the .NET Framework, and you can easily publish updates on a Web page or a network file share to make improvements to your program. You can get started with ClickOnce at any time by using the Publish command on the Build menu. And you can control how ClickOnce works by setting properties using the Properties command on the Project menu. (Click the Publish tab in the Project Designer for specific features.)

Windows Installer is a more classic installation process. In Visual Studio, you add a setup or a Windows Installer project to your solution, which automatically creates a setup program for the application. The installer package is distributed to your users, and individual users run the setup file and work through a wizard to install the application. The setup project can be customized to allow for different methods of installation, such as from CD-ROMs or Web servers. You can get started with Windows Installer by using the New Project command on the File menu to create a custom setup project. (Select the Setup And Deployment\Visual Studio Installer option under Other Project Types to see the list of available setup projects.)

Whether you choose ClickOnce or Windows Installer, you'll find that Visual Studio 2010 has brought many improvements to the installation process, and these technologies will directly benefit you and your customers. For additional information, see the online Help documentation related to the installation option that you want to use.

One Step Further: Adding to a Program

You can restart Visual Studio at any time and work on a programming project you've stored on disk. You'll restart Visual Studio now and add a *Randomize* statement to the Lucky Seven program.

Reload Lucky Seven

1. On the Windows taskbar, click Start, click All Programs, click Microsoft Visual Studio 2010, and then click the Microsoft Visual Studio 2010 program icon (or the Microsoft Visual Basic 2010 Express program icon, if you're using Visual Basic 2010 Express).

   A list of the projects that you've most recently worked on appears on the Visual Studio Start Page in the Recent Project pane. Because you just finished working with Lucky Seven, the MyLucky7 project should be first on the list.

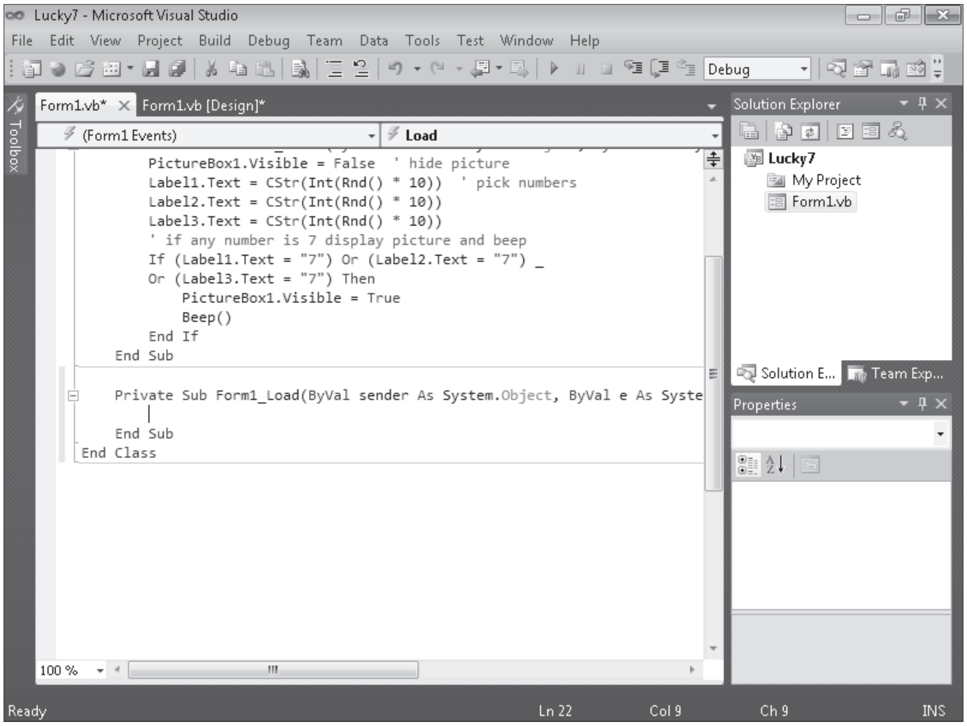
2. Click the MyLucky7 link to open the Lucky Seven project.

   The Lucky Seven program opens, and the MyLucky7 form appears. (If you don't see the form, click Form1.vb in Solution Explorer, and then click the View Designer button.)

   Now you'll add the *Randomize* statement to the *Form_Load* procedure, a special procedure that is associated with the form and that is executed each time the program is started.

3. Double-click the form (not one of the objects) to display the *Form_Load* procedure.

   The *Form_Load* procedure appears in the Code Editor, as shown here:


Lucky7 - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Test Window Help
Debug
Form1.vb* Form1.vb [Design]*
(Form1 Events)
Load
Solution Explorer
Lucky7
My Project
Form1.vb
Solution E... Team Exp...
Properties
Toolbox
100 %
Ready Ln 22 Col 9 Ch 9 INS

```
        PictureBox1.Visible = False  ' hide picture
        Label1.Text = CStr(Int(Rnd() * 10))  ' pick numbers
        Label2.Text = CStr(Int(Rnd() * 10))
        Label3.Text = CStr(Int(Rnd() * 10))
        ' if any number is 7 display picture and beep
        If (Label1.Text = "7") Or (Label2.Text = "7") _
        Or (Label3.Text = "7") Then
            PictureBox1.Visible = True
            Beep()
        End If
    End Sub

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As Syste

    End Sub
End Class
```

4. Type **Randomize**, and then press ENTER.

   The *Randomize* statement is added to the program and will be executed each time the program starts. *Randomize* uses the system clock to create a truly random starting point, or *seed*, for the *Rnd* statement used in the *Button1_Click* procedure. As I mentioned earlier, without the *Randomize* statement, the Lucky Seven program produces the same string of random spins every time you restart the program. With *Randomize* in place, the program spins randomly every time it runs, and the numbers don't follow a recognizable pattern.

5. Run the new version of Lucky Seven, and then save the project. If you plan to use the new version a lot, you might want to create a new .exe file, too.

6. When you're finished, click Close Project on the File menu.

   The files associated with the Lucky Seven program are closed.

Chapter 2 Quick Reference

To	Do This
Create a user interface	Use Toolbox controls to place objects on your form, and then set the necessary properties. Resize the form and the objects as appropriate.
Move an object	Point to the object, and when a four-headed arrow appears, drag the object.

To	Do This
Resize an object	Click the object to select it, and then drag the resize handle attached to the part of the object you want to resize.
Delete an object	Click the object, and then press DELETE.
Open the Code Editor	Double-click an object on the form (or the form itself). *or* Select a form or a module in Solution Explorer, and then click the View Code button.
Write program code	Type Visual Basic program statements associated with objects in the Code Editor.
Save a program	On the File menu, click the Save All command. *or* Click the Save All button on the Standard toolbar.
Save a form file	Make sure the form is open, and then, on the File menu, click the Save command. *or* Click the Save button on the Standard toolbar.
Create an .exe file	On the Build menu, click the Build or Rebuild command.
Deploy an application by using ClickOnce technology	Click the Publish command on the Build menu, and then use the Publish wizard to specify the location and settings for the application.
Reload a project	On the File menu, click the Open Project command. *or* On the File menu, point to Recent Projects and Solutions, and then click the desired project. *or* Click the project in the recent projects list on the Visual Studio Start Page.

Chapter 20

Creating Web Sites and Web Pages by Using Visual Web Developer and ASP.NET

After completing this chapter, you will be able to:

- Start Visual Web Developer and create a new Web site.
- Use Visual Web Developer tools and windows, including the Web Page Designer.
- Use the Visual Web Developer Toolbox to add server controls to Web pages.
- Add text, formatting effects, and Visual Basic code to a Web page that calculates loan payments for a car loan.
- Create a Web page that displays Help information.
- Use the *HyperLink* control to link one Web page to another on a Web site.
- Use the *GridView* control to display a table of database information on a Web page.
- Set the *Title* for a Web page and edit the master page.

In this chapter, you'll learn how to build Web sites and Web pages by using the Visual Web Developer tool included with Microsoft Visual Studio 2010. Visual Web Developer has the look and feel of the Visual Studio Integrated Development Environment (IDE), but it is customized for Web programming and Microsoft ASP.NET 4, the Microsoft .NET Framework component designed to provide state-of-the-art Internet functionality. Although a complete description of Web programming and ASP.NET isn't possible here, there's enough in common between Web programming and Windows programming to allow you to do some useful experimentation—even if you have little or no experience with Hypertext Markup Language (HTML). Invest a few hours in this chapter, and you'll see how quickly you can build a Web site that calculates loan payments for car loans, create a Web page with Help information, and display loan prospects from a Microsoft Access database by using the *GridView* control.

Inside ASP.NET

ASP.NET 4, Microsoft's Web development platform, has been enhanced in this release. Some of the improvements include how Web pages are created in the Web Page Designer; various feature enhancements to ASP.NET Web pages and ASP.NET MVC; support for recently introduced browsers and handheld devices; a new ASP.NET *Chart* server control; enhancements to the *FormView*, *ListView*, and *QueryExtender* controls; new dynamic data controls and enhancements; and improvements to the AJAX (Asynchronous JavaScript and XML) programming model. Although ASP.NET has some similarities with an earlier Web programming technology named Active Server Pages (ASP), ASP.NET has been significantly enhanced since its first release in Visual Studio .NET 2002, and continues to evolve as new features are added to the .NET Framework and Visual Studio software. Visual Web Developer is the tool that you use to create and manage ASP.NET user interfaces, commonly called *Web pages* or (in a more comprehensive sense) *Web sites*.

Tip In programming books about ASP.NET, you'll sometimes see Web pages referred to as *Web Forms* and Web sites referred to as *Web applications* or *ASP.NET applications*.

By using Visual Web Developer, you can create a Web site that displays a user interface, processes data, and provides many of the commands and features that a standard application for Windows might offer. However, the Web site you create is viewed in a Web browser, such as Internet Explorer, Mozilla Firefox, Apple Safari, or even one of the new mobile device types, including Google Chrome, the Research in Motion BlackBerry smart phone, and the Apple iPhone. These Web sites are typically stored on one or more *Web servers*, which use Microsoft Internet Information Services (IIS) to display the correct Web pages and handle most of the computing tasks required by your Web site. (In Visual Studio 2010, Web sites can also be located and run on a local computer that does not require IIS, giving you more options for development and deployment.) This distributed strategy allows your Web sites to potentially run on a wide range of Internet-based or stand-alone computers—wherever your users and their rich data sources are located.

To create a Web site in Visual Studio 2010, you click the New Web Site command on the File menu, and then use the Visual Web Developer to build one or more Web pages that will collectively represent your Web site. Each Web page consists of two pieces:

- A Web Forms page, which contains HTML, ASP.NET markup, and controls to create the user interface.
- A code-behind file, which is a code module that contains program code that "stands behind" the Web Forms page.

This division is conceptually much like the Windows Forms you've been creating in Microsoft Visual Basic—there's a UI component and a code module component. The code for both of these components can be stored in a single .aspx file, but typically the Web Forms page code is stored in an .aspx file, and the code-behind file is stored in an .aspx.vb file.

In addition to Web pages, Web sites can contain code modules (.vb files), HTML pages (.htm files), configuration information (Web.config files), global Web application information (Global.asax files), cascading style sheet (CSS) information, scripting files (JavaScript), master pages, and other components. You can use the Web Page Designer and Solution Explorer to switch back and forth between these components quickly and efficiently.

Web Pages vs. Windows Forms

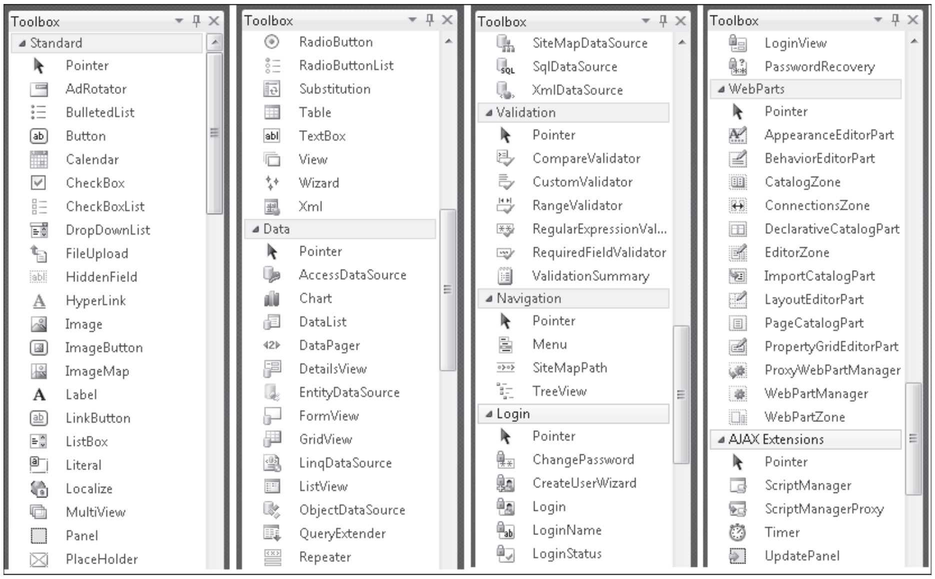
What are the important differences between Web pages and Windows Forms? To begin with, Web pages offer a slightly different programming paradigm than Windows Forms. Whereas Windows Forms use a Windows application window as the primary user interface for a program, a Web site presents information to the user through one or more Web pages with supporting program code. These pages are viewed through a Web browser, and you can create them by using the Web Page Designer.

Like a Windows Form, a Web page can include text, graphic images, buttons, list boxes, and other objects that are used to provide information, process input, or display output. However, the basic set of controls you use to create a Web page is not the set on the Common Controls tab of the Toolbox. Instead, ASP.NET Web sites must use controls on one of the tabs in the Visual Web Developer Toolbox, including Standard, Data, HTML, and many others. Each of the Visual Web Developer controls has its own unique methods, properties, and events, and although there are many similarities between these controls and Windows Forms controls, there are also several important differences. For example, the Visual Studio *DataGridView* control is called *GridView* in Visual Web Developer and has different properties and methods.

Many Web page controls are *server controls*, meaning that they run on the Web server. Server controls have an “asp” prefix in their tag. HTML controls (located on the HTML tab of the Visual Web Developer Toolbox) are *client controls* by default, meaning that they run only within the user’s browser. For now, however, you simply need to know that you can use server controls, HTML controls, or a combination of both in your Web site projects. As you gain experience in Web programming, you may want to investigate AJAX programming in Visual Studio, which can enhance the efficiency of your Web applications and add advanced user-interface elements for users.

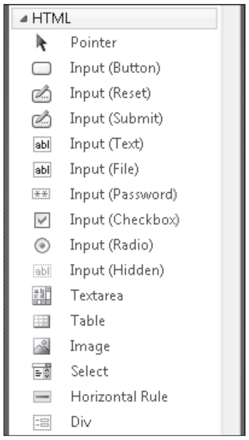
Server Controls

Server controls are more capable than HTML controls and function in many ways like the Windows Forms controls. Indeed, many of the server controls have the same names as the Windows Forms controls and offer many of the same properties, methods, and events. In addition to simple controls such as *Button*, *TextBox*, and *Label*, more sophisticated controls such as *Chart*, *FileUpload*, *LoginView*, and *RequiredFieldValidator* are provided on a number of tabs in the Toolbox; Visual Studio 2010 has added a number of controls to the list. The screen shot on the following page shows a sample of the server controls in the Visual Web Developer Toolbox. (Dynamic Data and Reporting controls are not shown.)



HTML Controls

The HTML controls are a set of older user interface (UI) controls that are supported by all Web browsers and conform closely to the early HTML standards developed for managing UI elements on a typical Web page. They include *Button*, *Text*, and *Checkbox*—useful basic controls for managing information on a Web page that can be represented entirely with HTML code. Indeed, you might recognize these controls if you've coded in HTML before. However, although they're easy to use and have the advantage of being a "common denominator" for Web browsers, they're limited by the fact that they have no ability to maintain their own state. (In other words, the data that they contain will be lost between views of a Web page.) The following screen shot shows the HTML controls offered on the HTML tab of the Toolbox in Visual Web Developer:



Building a Web Site by Using Visual Web Developer

The best way to learn about Visual Web Developer and ASP.NET is to get some hands-on practice. In the exercises in this chapter, you'll create a simple car loan calculator that determines monthly payments and contains an About tab that explains how the program works. Later in the chapter, you'll use the *GridView* control to display a table of data on a Web page in the same Web site. You'll begin by verifying that Visual Studio is properly configured for ASP.NET programming, and then you'll create a new Web site project. Next, you'll use the Web Page Designer to create a Web page with text and links on it, and you'll use controls in the Visual Web Developer Toolbox to add controls to the Web page.

Considering Software Requirements for ASP.NET Programming

Before you can create your first ASP.NET Web site, you need to make sure your computer is set up properly. To perform ASP.NET programming, you need to have Visual Web Developer installed. Visual Web Developer is a component of Visual Studio 2010 Professional, Premium, and more advanced editions. You can also download Visual Web Developer 2010 Express at *http://www.microsoft.com/express/Web/,* and it contains almost all the features described in this chapter (I'll point out any differences as we go). If you are using Visual Web Developer 2010 Express, be sure to set the settings to Expert by clicking the Tools menu, clicking Settings, and then clicking Expert Settings. This will ensure that the steps in this chapter more closely match your software.

Visual Studio 2010 and Visual Web Developer include their own local Web server, so setting up and configuring a Web server with Microsoft Internet Information Services (IIS) and the .NET Framework is not required. Having a local Web server makes it easy to create and test your ASP.NET Web sites, and you'll see it described below as the ASP.NET Development Server.

In Visual Studio 2010, you can create and run your Web site in one of three locations:

- Your own computer (via the ASP.NET Development Server)
- An HTTP server that contains IIS and related components
- An FTP site (a remote file server)

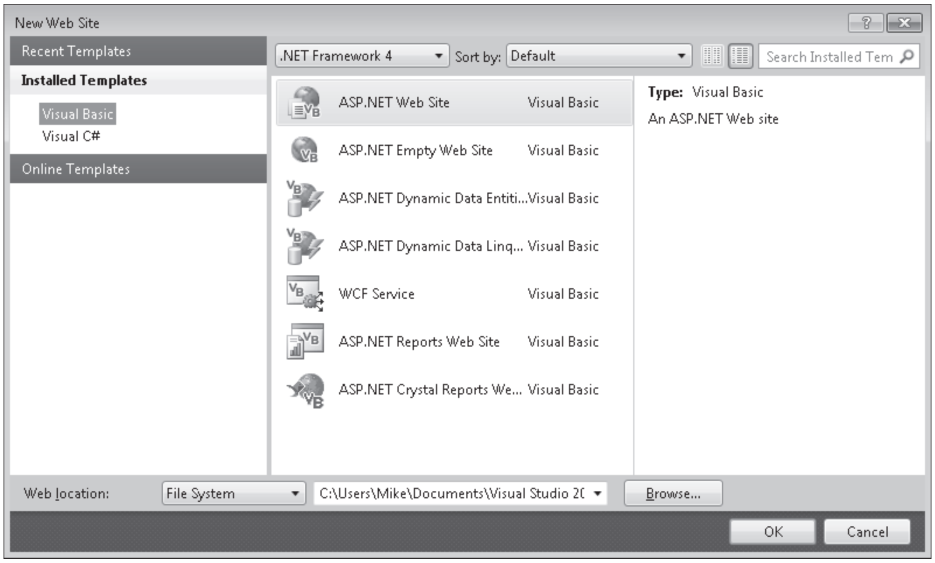
The first location is the option we'll use in this book because it requires no additional hardware or software. In addition, when you develop your Web site on the local file system, all the Web site files are stored in one location. When you're finished testing the application, you can deploy the files to a Web server of your choosing.

Create a new Web site

1. Start Visual Studio, and then click the New Web Site command on the File menu.

> **Note** If you don't see the New Web Site command on the File menu, then you don't have Visual Web Developer installed. To download Visual Web Developer Express, visit *http://www.microsoft.com/express/Web/* and follow the installation instructions.

Although you might have seen the New Web Site command before, we haven't used it yet in this book. This command starts Visual Web Developer and prepares Visual Studio to build a Web site. You see a New Web Site dialog box similar to the following:



ASP.NET Web Site	Visual Basic
ASP.NET Empty Web Site	Visual Basic
ASP.NET Dynamic Data Entiti...	Visual Basic
ASP.NET Dynamic Data Linq...	Visual Basic
WCF Service	Visual Basic
ASP.NET Reports Web Site	Visual Basic
ASP.NET Crystal Reports We...	Visual Basic

Type: Visual Basic
An ASP.NET Web site

Web location: File System C:\Users\Mike\Documents\Visual Studio 2(

In this dialog box, you can select the Web site or application template, the location for the Web site (local file system, HTTP server, or FTP site), and the programming language that you want to use (Visual Basic or Microsoft Visual C#). You can also identify the version of the .NET Framework that you want to target with your Web application. (Version 4 offers the most features, but there are times that you may need to design specifically for platforms with an earlier version of the .NET Framework. However, Visual Web Developer 2010 Express does not provide the option of targeting a specific version of the .NET Framework.)

2. In the New Web Site dialog box, verify that Visual Basic is the selected language and that ASP.NET Web Site is the selected template.
3. In the Web Location list, make sure that File System is selected.
4. Type **C:\Vb10sbs\MyChap20** in the File Name text box.

   Although you have been specifying the folder location for projects *after* you have built the projects in this book, in Visual Web Developer, projects are saved up front.

The “my” prefix in the path will avoid a conflict with the solution Web site in the practice files (C:\Vb10sbs\Chap20) that I’ve built for you.

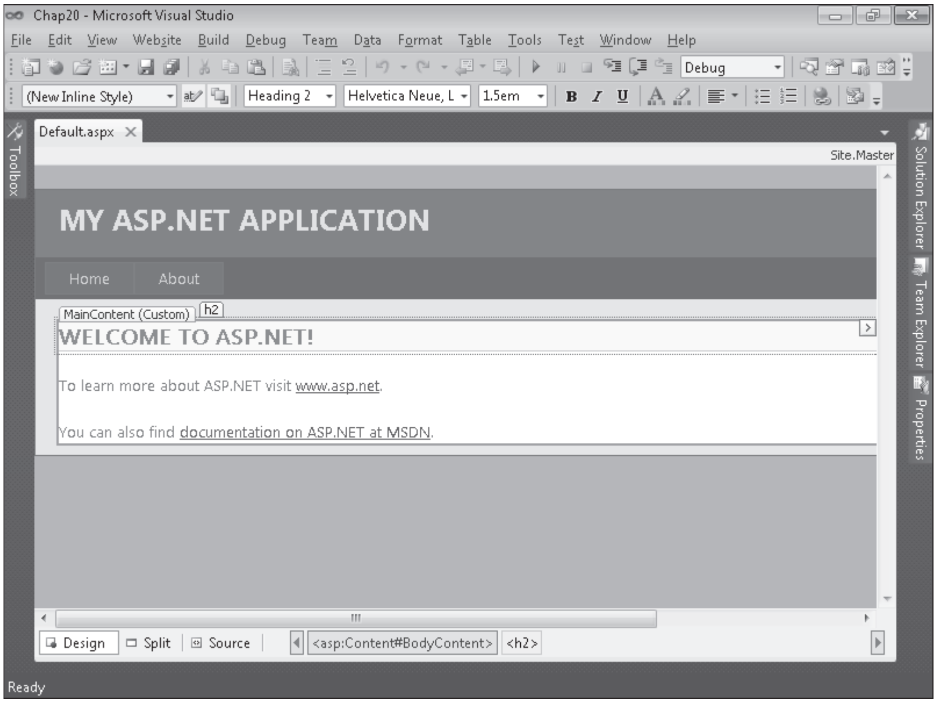
5. Click OK to accept your selections.

   Visual Studio loads Visual Web Developer and creates a Web page (Default.aspx) to contain the user interface and a code-behind file (Default.aspx.vb) that will store the code for your Web page.

6. If you don’t see Default.aspx open in the Web Page Designer, double-click Default.aspx in Solution Explorer now to open it.

7. At the bottom of the Web Page Designer, click the Design tab.

   Your screen looks something like the one shown in the following screen shot:



MY ASP.NET APPLICATION

Home About

MainContent (Custom) h2

WELCOME TO ASP.NET!

To learn more about ASP.NET visit www.asp.net.

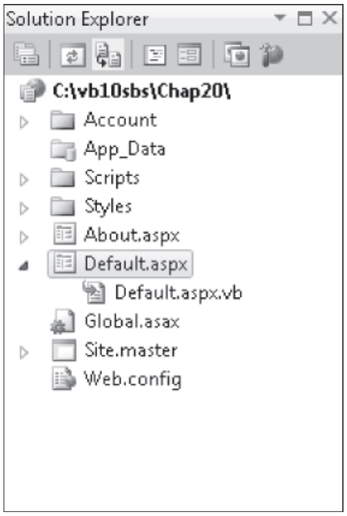
You can also find documentation on ASP.NET at MSDN.

Unlike the Windows Forms Designer, the Web Page Designer displays the Web page in three possible views in the IDE, and three tabs at the bottom of the Designer (Design, Split, and Source) allow you to change your view of the Web page.

The Design tab shows you approximately how your Web page will look when a Web browser displays it. When the Design tab is selected, a basic template page (“My ASP.NET Application”) appears in the Designer with the result of source-code formatting, and you can add controls to your Web page and adjust how objects on the page are arranged.

On the Source tab, you can view and edit the HTML and ASP.NET markup that’s used to display the Web page in a Web browser. If you’ve used Microsoft Expression Web, you’ll be familiar with these two ways of displaying a Web page and perhaps also with some of the HTML tags that control how Web pages are actually displayed. The Split tab offers a composite view of the Design and Source tabs.

A few additional differences between the Windows Forms Designer and the Web Page Designer are worth noting at this point. The Toolbox now contains several collections of controls used exclusively for Web programming. Solution Explorer also contains a different list of project files for the Web site you're building, as shown in the following screen shot. In particular, notice the Default.aspx file in Solution Explorer; this file contains the UI code for the active Web page. Nested under the Default.aspx file, you'll find a file named Default.aspx.vb. A configuration file named Web.config and a master page file named Site.master are also listed.



Note When you close your new Web site and exit Visual Web Developer, note that you open the Web site again by clicking the Visual Studio File menu and then clicking the Open Web Site command. Web sites are not opened by using the Open Project command on the File menu.

Now you're ready to add some text to the Web page by using the Web Page Designer.

Using the Web Page Designer

Unlike a Windows Form, a Web page can have text added directly to it when it is in the Web Page Designer. In Source view, the text appears within HTML and ASP.NET tags somewhat as it does in the Visual Studio Code Editor. In Design view, the text appears in top-to-bottom fashion within a grid as it does in a word processor such as Microsoft Word, and you'll see no HTML. In the next exercises, you'll type text in Design view, edit it, and then make formatting changes by using buttons on the Formatting toolbar. Manipulating text in this way is usually much faster than adding a *Label* control to the Web page to contain the text. You'll practice entering the text for your car loan calculator in the following exercise.

Add text in Design view

1. Click the Design tab, if it is not currently selected, to view the Web Page Designer in Design view.

   A faint rectangle appears at the top of the Web page, near the template text "WELCOME TO ASP.NET." The template text is there to show you how text appears on a Web Form, and where you can go to get additional information about ASP.NET. You'll also notice that your Web page has Home and About tabs, which are provided for you as part of your default page.

2. Position your insertion point at the end of the text "WELCOME TO ASP.NET."

   A blinking I-beam appears at the end of the line.

3. Press the BACKSPACE key to remove "WELCOME TO ASP.NET," and then type **Car Loan Calculator**.

   Visual Studio displays the title of your Web page exactly as it will appear when you open the Web site in your browser.

4. Delete the line beginning with "To learn more about ASP.NET…," and in its place, type the following sentence:

   Enter the required information and click Calculate!

5. Delete the sentence in the template beginning with "You can also find documentation…"

   Now you'll use the Formatting toolbar to format the title with italic formatting and a different color.

6. Right-click the Standard toolbar in Visual Web Developer to display the list of toolbars available in the IDE.

7. If you do not see a check mark next to Formatting in this list, click Formatting to add the Formatting toolbar.

   The Formatting toolbar now appears in the IDE if it was not already visible. Notice that it contains a few features not usually found on a text formatting toolbar.

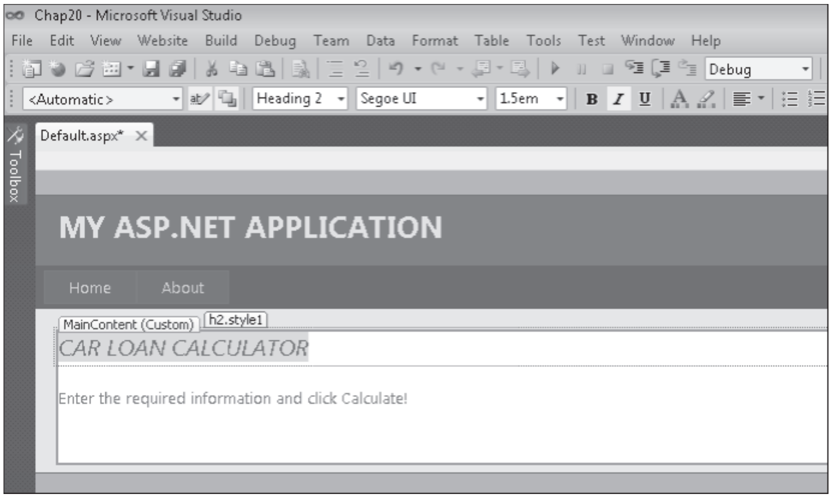
8. Select the text "Car Loan Calculator."

   Before you can format text in Visual Web Developer, you must select it.

9. Click the Italic button on the Formatting toolbar.

10. On the Format menu, click the Font command, click Red in the Color list box, and then click OK.

Your screen looks like this:

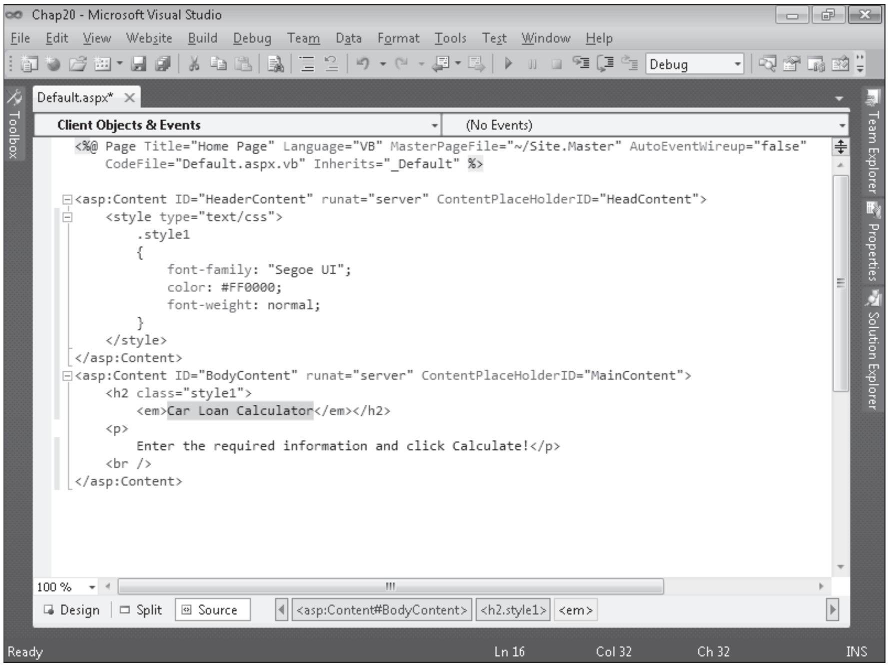

Chap20 - Microsoft Visual Studio
File Edit View Website Build Debug Team Data Format Table Tools Test Window Help
Debug
<Automatic> Heading 2 Segoe UI 1.5em
Default.aspx*
Toolbox
MY ASP.NET APPLICATION
Home About
MainContent (Custom) h2.style1
CAR LOAN CALCULATOR
Enter the required information and click Calculate!

Now, you'll examine the HTML and ASP.NET markup for the text and formatting you entered.

View the HTML and ASP.NET markup for a Web page

1. Click the Source tab at the bottom of the Designer.

The Source tab displays the actual HTML and ASP.NET markup for your Web page. To see more of the markup, you might want to resize a few programming tools temporarily and use the document scroll bars. The markup looks like the following screen shot. Your markup might have some differences.


Chap20 - Microsoft Visual Studio
File Edit View Website Build Debug Team Data Format Tools Test Window Help
Debug
Default.aspx*
Toolbox
Client Objects & Events
(No Events)
Team Explorer
Properties
Solution Explorer
100 %
Design Split Source
<asp:Content#BodyContent> <h2.style1> <em>
Ready Ln 16 Col 32 Ch 32 INS

```
<%@ Page Title="Home Page" Language="VB" MasterPageFile="~/Site.Master" AutoEventWireup="false"
    CodeFile="Default.aspx.vb" Inherits="_Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
    <style type="text/css">
        .style1
        {
            font-family: "Segoe UI";
            color: #FF0000;
            font-weight: normal;
        }
    </style>
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <h2 class="style1">
        <em>Car Loan Calculator</em></h2>
    <p>
        Enter the required information and click Calculate!</p>
    <br />
</asp:Content>
```

A Web page is made up of page information, scripting code, cascading style sheet (CSS) information, HTML tags, ASP.NET tags, image references, objects, and text. The *@ Page* directive contains information about the language you selected when creating the Web application, the name of any code-behind file, and any inherited forms.

HTML and ASP.NET tags typically appear in pairs so that you can see clearly where a section begins and ends. For example, the *<style>* tag identifies the beginning of text formatting, and the *</style>* tag identifies the end. Notice that the "Car Loan Calculator" text appears within *<em></em>* tags to make the text italic. Below the "Car Loan Calculator" text, the second line of text you entered is displayed.

Tip Remember that the Source tab is an actual editor, so you can change the text that you entered by using standard text editing techniques. If you know something about HTML and ASP.NET, you can add other tags and content as well.

2. Click the Design tab to display your Web page in Design view, and open the Toolbox if it is not visible.

Adding Server Controls to a Web Site

Now you'll add *TextBox*, *Label*, and *Button* controls to the car loan calculator. Although these controls are located in the Visual Web Developer Toolbox, they're very similar to the Windows Forms controls of the same name that you've used throughout this book. (I'll cover a few of the important differences as they come up.) The most important thing to remember is that in the Web Page Designer, controls are inserted at the insertion point if you double-click the control name in the Toolbox. After you add the controls to the Web page, you'll set property settings for the controls.

Use *TextBox*, *Label*, and *Button* controls

1. Display the Standard tab of the Toolbox, if it isn't already visible.
2. Position the insertion point below the last line of text on the Web page, and then press ENTER to create a little blank space below the text for the controls.

   Because controls are placed at the insertion point, you need to use the text editing keys to position the insertion point appropriately before double-clicking a control in the Toolbox.

Note By default, the Web Page Designer positions controls relative to other controls. This is an important difference between the Web Page Designer and the Windows Forms Designer. The Windows Forms Designer allows you to position controls wherever you like on a form. You can change the Web Page Designer so that you can position controls wherever you like on a Web page (called *absolute positioning*); however, you might get different behavior in different Web browsers.

3. Double-click the *TextBox* control on the Standard tab of the Toolbox to create a text box object at the insertion point on the Web page.

   Notice the *asp:textbox#TextBox1* text that appears above the text box object. The "asp" prefix indicates that this object is an ASP.NET server control. (This text disappears when you run the program.)

4. Click the right side of the text box object to place the insertion point at the outside edge, and then press ENTER.

5. Double-click the *TextBox* control again to add a second text box object to the Web page.

6. Repeat Steps 4 and 5 to create a third text box object below the second text box.

   Now you'll use the *Label* control to insert labels that identify the purpose of the text boxes.

7. Click to the right of the first text box object to place the insertion point at the right edge of the text box.

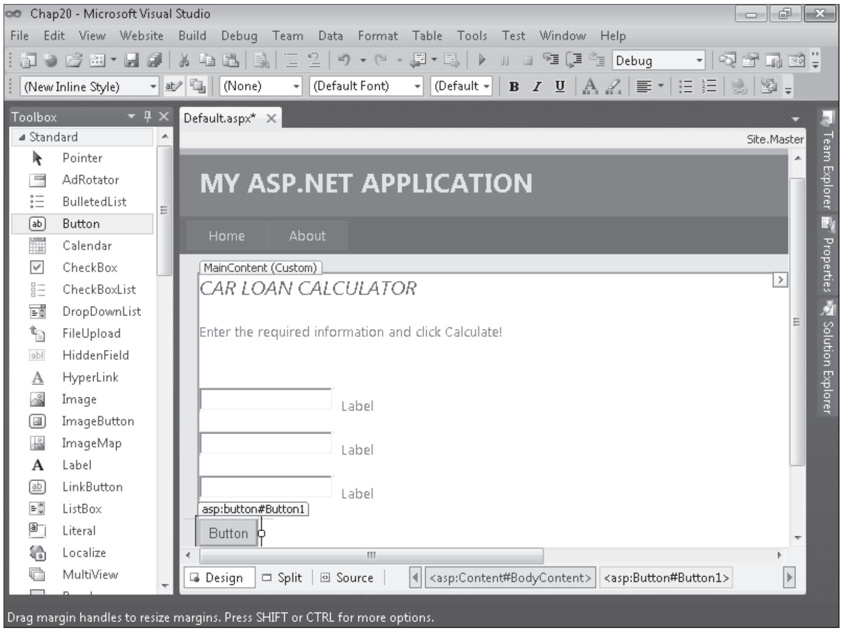
8. Press the SPACEBAR key twice to add two blank spaces, and then double-click the *Label* control in the Toolbox to add a label object to the Web page.

9. Repeat Steps 7 and 8 to add label objects to the right of the second and third text boxes.

10. Click to the right of the third label object to place the insertion point to the right of the label, and then press ENTER.

11. Double-click the *Button* control to create a button object at the bottom of the Web page.

    The *Button* control, like the *TextBox* and *Label* controls, is very similar to its Windows Forms counterpart. Your screen looks like this:



Chap20 - Microsoft Visual Studio

MY ASP.NET APPLICATION

MainContent (Custom)

CAR LOAN CALCULATOR

Enter the required information and click Calculate!

Label

Label

Label

asp:button#Button1

Button

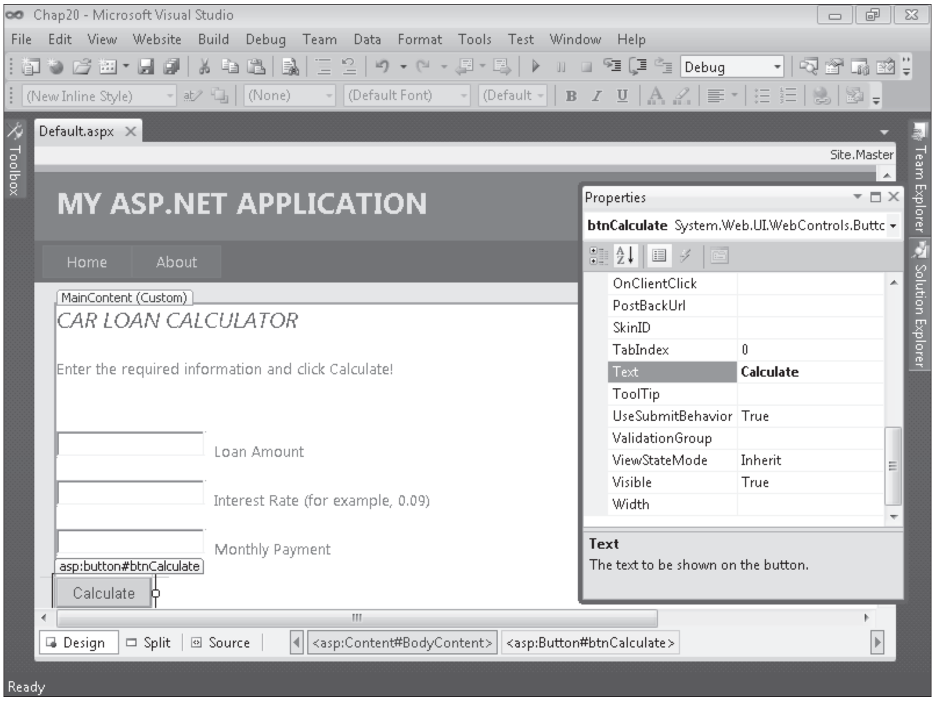
Drag margin handles to resize margins. Press SHIFT or CTRL for more options.

Now you'll set a few properties for the seven new controls you have created on the Web page. If it is not already visible, open the Properties window by pressing F4. As you set the properties, you'll notice one important difference between Web pages and Windows Forms—the familiar *Name* property has been changed to *ID* in Visual Web Developer. Despite their different names, the two properties perform the same function.

12. Set the following properties for the objects on the form:

Object	Property	Setting
TextBox1	*ID*	txtAmount
TextBox2	*ID*	txtInterest
TextBox3	*ID*	txtPayment
Label1	*ID*	lblAmount
	Text	"Loan Amount"
Label2	*ID*	lblInterest
	Text	"Interest Rate (for example, 0.09)"
Label3	*ID*	lblPayment
	Text	"Monthly Payment"
Button1	*ID*	btnCalculate
	Text	"Calculate"

Your Web page looks like this:


Chap20 - Microsoft Visual Studio
File Edit View Website Build Debug Team Data Format Tools Test Window Help
Debug
(New Inline Style) (None) (Default Font) (Default
Toolbox
Default.aspx
Site.Master
MY ASP.NET APPLICATION
Home About
MainContent (Custom)
CAR LOAN CALCULATOR
Enter the required information and click Calculate!
Loan Amount
Interest Rate (for example, 0.09)
Monthly Payment
asp:button#btnCalculate
Calculate
Design Split Source
<asp:Content#BodyContent> <asp:Button#btnCalculate>
Ready
Team Explorer
Solution Explorer

Properties
btnCalculate System.Web.UI.WebControls.Buttc

OnClientClick	
PostBackUrl	
SkinID	
TabIndex	0
Text	Calculate
ToolTip	
UseSubmitBehavior	True
ValidationGroup	
ViewStateMode	Inherit
Visible	True
Width	

Text
The text to be shown on the button.

Writing Event Procedures for Web Page Controls

You write default event procedures (or event handlers) for controls on a Web page by double-clicking the objects on the Web page and typing the necessary program code in the Code Editor. Although the user will see the controls on the Web page in his or her own Web browser, the actual code that's executed will be located on the local test computer or a Web server, depending on how you configured your project for development and how it is eventually deployed. For example, when the user clicks a button on a Web page that is hosted by a Web server, the browser sends the button click event back to the server, which processes the event and sends a new Web page back to the browser. Although the process seems similar to that of Windows Forms, there's actually a lot going on behind the scenes when a control is used on an ASP.NET Web page!

In the following exercise, you'll practice creating the default event procedure for the *btnCalculate* object on the Web page.

Create the *btnCalculate_Click* event procedure

1. Double-click the Calculate button on the Web page.

   The code-behind file (Default.aspx.vb) opens in the Code Editor, and the *btnCalculate_Click* event procedure appears.

2. Type the following program code:

```
Dim LoanPayment As Double
'Use Pmt function to determine payment for 36 month loan
LoanPayment = Pmt(CDbl(txtInterest.Text) / 12, 36, CDbl(txtAmount.Text))
txtPayment.Text = Format(Abs(LoanPayment), "$0.00")
```

   This event procedure uses the *Pmt* function, a financial function that's part of the Visual Basic language, to determine what the monthly payment for a car loan would be by using the specified interest rate (*txtInterest.Text*), a three-year (36-month) loan period, and the specified principal amount (*txtAmount.Text*). The result is stored in the *LoanPayment* double-precision variable, and then it is formatted with appropriate monetary formatting and displayed by using the *txtPayment* text box object on the Web page.

   The two *Text* properties are converted from string format to double-precision format by using the *CDbl* function. The *Abs* (absolute value) function is used to make the loan payment a positive number. (*Abs* currently has a jagged underline in the Code Editor because it relies on the *System.Math* class, which you'll specify next.) Why make the loan payment appear as a positive number? The *Pmt* function returns a negative number by default (reflecting money that's owed), but I think negative formatting looks strange when it isn't part of a balance sheet, so I'm converting it to positive.

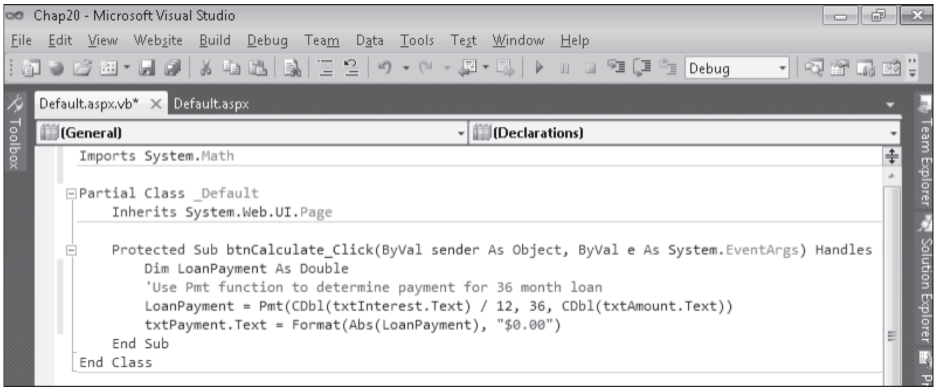
Notice that the program statements in the code-behind file are just regular Visual Basic code—the same stuff you've been using throughout this book. Basically, the process feels similar to creating a Windows application.

3. Scroll to the top of the Code Editor, and then enter the following program statement as the first line of the file:

   ```
   Imports System.Math
   ```

   As you learned in Chapter 5, "Visual Basic Variables and Formulas, and the .NET Framework," the *Abs* function isn't included in Visual Basic by default, but it is part of the *System.Math* class in the .NET Framework and can be more easily referenced in your project by the *Imports* statement. Web applications can make use of the .NET Framework class libraries just as Windows applications can.

   The Code Editor looks like this:


Chap20 - Microsoft Visual Studio
File Edit View Website Build Debug Team Data Tools Test Window Help
Debug
Default.aspx.vb* Default.aspx
(General)
(Declarations)
Toolbox
Team Explorer
Solution Explorer

```
Imports System.Math

Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub btnCalculate_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
        Dim LoanPayment As Double
        'Use Pmt function to determine payment for 36 month loan
        LoanPayment = Pmt(CDbl(txtInterest.Text) / 12, 36, CDbl(txtAmount.Text))
        txtPayment.Text = Format(Abs(LoanPayment), "$0.00")
    End Sub
End Class
```

4. Click the Save All button on the Standard toolbar.

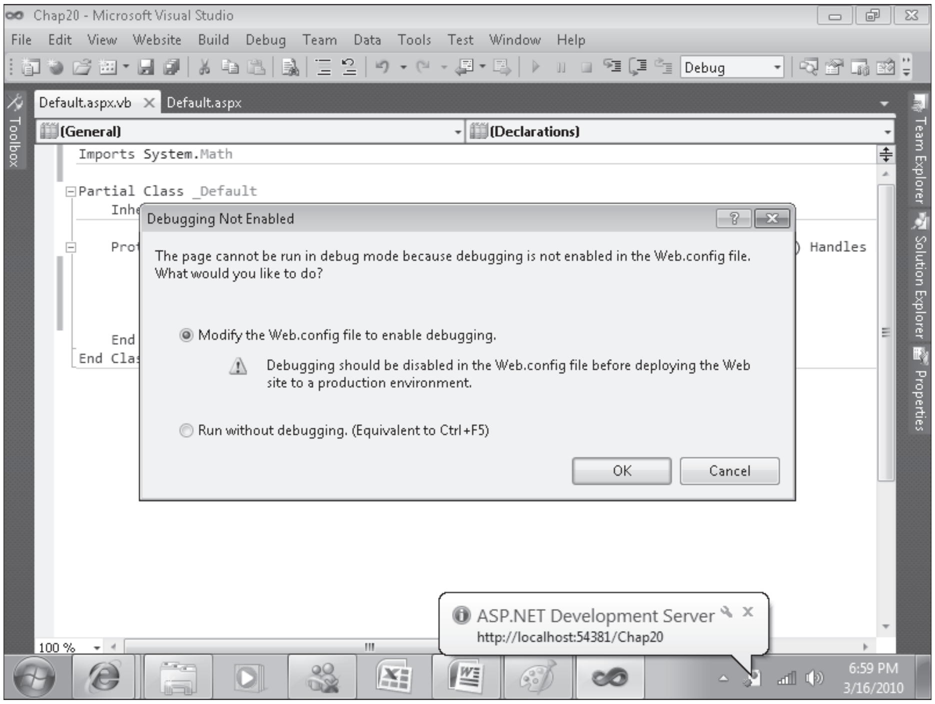
That's it! You've entered the program code necessary to run the car loan calculator and make your Web page interactive. Now you'll build and run the project and see how it works. You'll also learn a little bit about security settings within Internet Explorer, a topic closely related to Web development.

Build and view the Web site

1. Click the Start Debugging button on the Standard toolbar.

   Visual Studio starts the ASP.NET Development Server, which runs ASP.NET applications locally (on your own computer) so that you can test this application. A status balloon appears at the bottom of your screen and lets you know the local Uniform Resource

Locator (URL) on your computer that has been established, as shown in the following screen shot. You'll also see a message about debugging:



```
Imports System.Math

Partial Class _Default
    Inhe
    Prot                                                    ) Handles
    End
End Clas
```

Debugging Not Enabled

The page cannot be run in debug mode because debugging is not enabled in the Web.config file. What would you like to do?

[✓ Modify the Web.config file to enable debugging.]

Debugging should be disabled in the Web.config file before deploying the Web site to a production environment.

Run without debugging. (Equivalent to Ctrl+F5)

ASP.NET Development Server
http://localhost:54381/Chap20

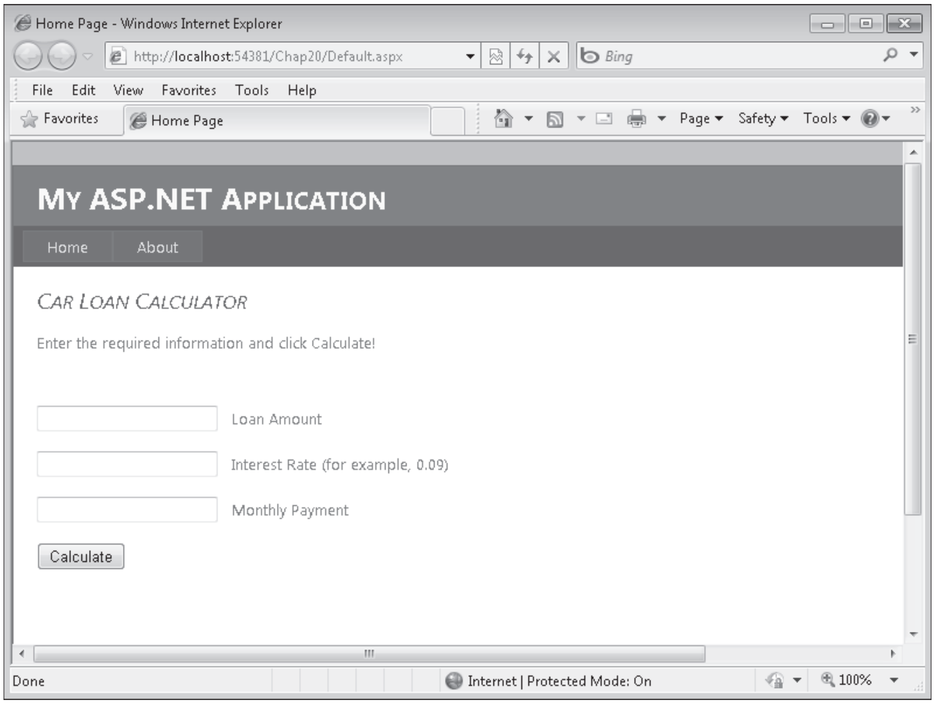
The potentially confusing Debugging Not Enabled dialog box is not a major concern. Visual Web Developer is just indicating that the Web.config file in your project does not currently allow debugging (a standard security feature). Although you can bypass this dialog box each time that you test the application within Visual Web Developer by clicking the Run Without Debugging button, I recommend that you modify the Web.config file now.

Security Tip Before you widely distribute or deploy a real Web site, be sure to disable debugging in Web.config to keep your application safe from unauthorized tampering.

2. Click OK to modify the Web.config file.

   Visual Studio modifies the file, builds your Web site, and displays the opening Web page in Internet Explorer.

   The car loan calculator looks like the screen shot on the following page. If Internet Explorer does not appear, you might need to select it on the Windows taskbar.


Home Page - Windows Internet Explorer
http://localhost:54381/Chap20/Default.aspx
Bing
File Edit View Favorites Tools Help
Favorites
Home Page
Page Safety Tools
MY ASP.NET APPLICATION
Home
About
CAR LOAN CALCULATOR
Enter the required information and click Calculate!
Loan Amount
Interest Rate (for example, 0.09)
Monthly Payment
Calculate
Done
Internet | Protected Mode: On
100%

Security Tip You might see the Information Bar at the top of Internet Explorer indicating that intranet settings are turned off by default. An intranet warning is again related to Internet Explorer's design to protect you from rogue programs or unauthorized access. An intranet is a local network (typically a home network or small workgroup network), and because Visual Studio uses intranet-style addressing when you test Web sites built on your own computer, you're likely to see this warning message. To suppress the warning temporarily, click the Information Bar and then click Don't Show Me This Again. To remove intranet warnings more permanently, click the Internet Options command on the Tools menu of Internet Explorer, click the Security tab, and then click Local Intranet. Click the Sites button, and clear the check mark from Automatically Detect Intranet Network in the Local Intranet dialog box. However, exercise caution whenever you disable security warnings, as they are meant to protect you.

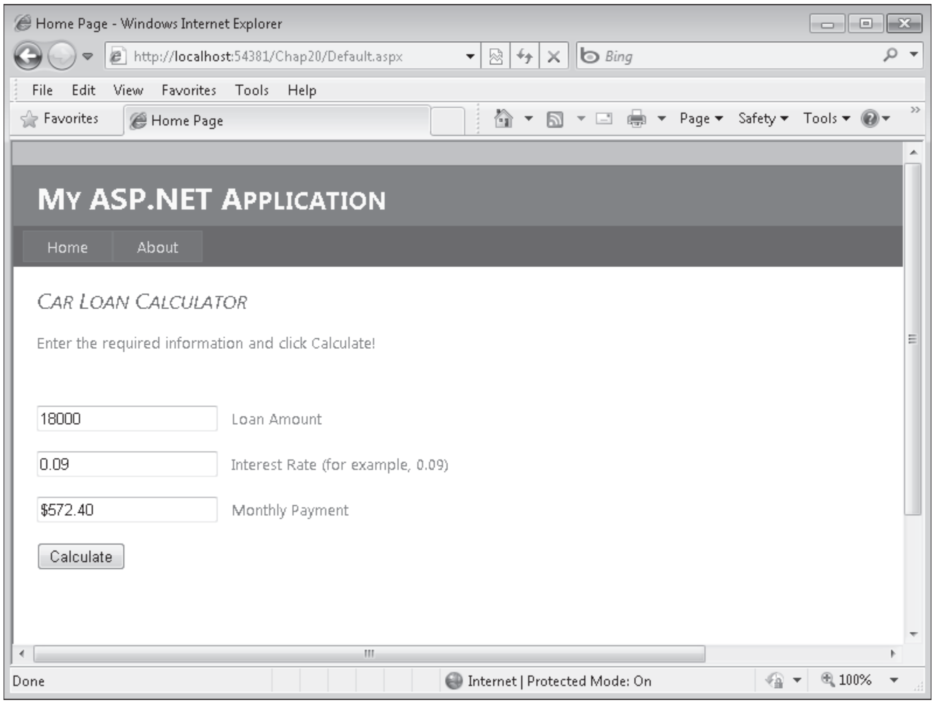
Now, let's get back to testing our Web page.

3. Type **18000** in the Loan Amount text box, and then type **0.09** in the Interest Rate text box.

   You'll compute the monthly loan payment for an $18,000 loan at 9 percent interest for 36 months.

4. Click the Calculate button.

   Visual Basic calculates the payment amount and displays $572.40 in the Monthly Payment text box. Your screen looks like this:


Home Page - Windows Internet Explorer
http://localhost:54381/Chap20/Default.aspx
Bing
File Edit View Favorites Tools Help
Favorites
Home Page
Page Safety Tools
MY ASP.NET APPLICATION
Home
About
Done
Internet | Protected Mode: On
100%

CAR LOAN CALCULATOR

Enter the required information and click Calculate!

18000	Loan Amount
0.09	Interest Rate (for example, 0.09)
$572.40	Monthly Payment

Calculate

5. Close Internet Explorer.

   You're finished testing your Web site for now. When Internet Explorer closes, your program is effectively ended. As you can see, building and viewing a Web site is basically the same as building and running a Windows application, except that the Web site is executed in the browser. You can even set break points and debug your application just as you can in a Windows application.

Curious about installing a Web site like this on an actual Web server? The basic procedure for deploying Web sites is to copy the .aspx files and any necessary support files for the project to a properly configured virtual directory on a Web server running IIS and .NET Framework 4. There are a couple of ways to perform deployment in Visual Web Developer. To get started, click Copy Web Site on the Website menu, or click Publish Web Site on the Build menu. (Visual Web Developer 2010 Express does not include the Publish Web Site command.) For more information about your options, see "ASP.NET Deployment Content Map" in the Visual Studio Help documentation. To find a hosting company that can host ASP.NET Web applications, you can check out *http://www.asp.net*.

Validating Input Fields on a Web Page

Although this Web page is useful, it runs into problems if the user forgets to enter a principal amount or an interest rate or specifies data in the wrong format. To make Web sites like this more robust, I usually add one or more *validator controls* that force users to enter input in the proper format. The validator controls are located on the Validation tab of the Visual Web Developer Toolbox and include controls that require data entry in a field (*RequiredFieldValidator*), require entry in the proper range (*RangeValidator*), and so on. For information on the validator controls, search the Visual Studio Help documentation. They are straightforward to use.

Customizing the Web Site Template

Now the fun begins! Only very simple Web sites consist of just one Web page. Using Visual Web Developer, you can expand your Web site quickly to include additional information and resources, including HTML pages, XML pages, text files, database records, Web services, login sessions, site maps, and more. If you want to add a Web page, you have three options:

- You can create a new Web page by using the HTML Page template or the Web Form template. You select these templates by using the Add New Item command on the Website menu. After you create the page, you add text and objects to the page by using the Web Page Designer.
- You can add a Web page that you have already created by using the Add Existing Item command on the Web site menu, and then customize the page in the Web Page Designer. You use this method if you want to include one or more Web pages that you have already created in a tool such as Expression Web. (If possible, add pages that don't rely on external style sheets and resources, or you'll need to add those items to the project as well.)
- You can use an existing Web page that is part of the Web site template that you are using. For example, in the Web site template that you have open now, there is an About Web page and various Login Web pages that you can customize and use quickly.

In the following exercise, you'll display the About Web page supplied by the template that you are using, and you will customize it with some information about how the car loan calculator application works.

Customize the About.aspx Web page

1. Display Solution Explorer, click the About.aspx file, and click the View Designer button.

   Visual Web Designer displays About.aspx in the Designer, and it displays a line of placeholder text ("Put content here.").

2. Delete the placeholder text, and then type the following information:

   Car Loan Calculator

   The Car Loan Calculator Web site was developed for the *book Microsoft Visual Basic 2010 Step by Step*, by Michael Halvorson (Microsoft Press, 2010). The Web site is best viewed using Microsoft Internet Explorer version 6.0 or later. To learn more about how this ADO.NET application was created, read Chapter 20 in the book.

   Operating Instructions:

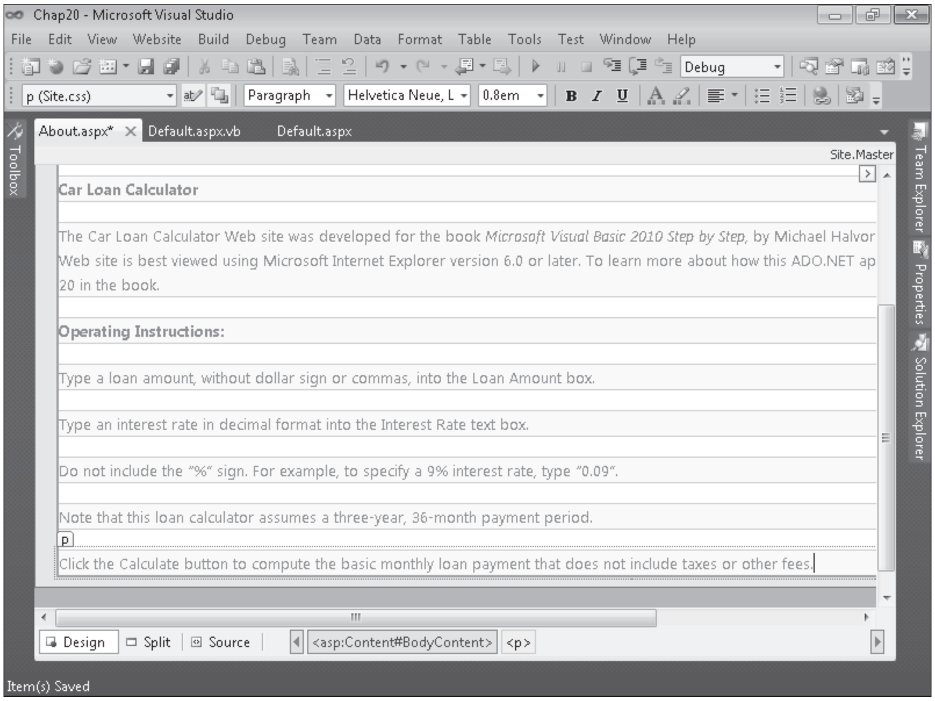
   Type a loan amount, without dollar sign or commas, into the Loan Amount box.

   Type an interest rate in decimal format into the Interest Rate text box. Do not include the "%" sign. For example, to specify a 9% interest rate, type "0.09."

   Note that this loan calculator assumes a three-year, 36-month payment period.

   Click the Calculate button to compute the basic monthly loan payment that does not include taxes or other fees.

3. Using buttons on the Formatting toolbar, add bold formatting for the headings and italic for the book title, as shown here:



Car Loan Calculator

The Car Loan Calculator Web site was developed for the book *Microsoft Visual Basic 2010 Step by Step*, by Michael Halvor
Web site is best viewed using Microsoft Internet Explorer version 6.0 or later. To learn more about how this ADO.NET ap
20 in the book.

Operating Instructions:

Type a loan amount, without dollar sign or commas, into the Loan Amount box.

Type an interest rate in decimal format into the Interest Rate text box.

Do not include the "%" sign. For example, to specify a 9% interest rate, type "0.09".

Note that this loan calculator assumes a three-year, 36-month payment period.

Click the Calculate button to compute the basic monthly loan payment that does not include taxes or other fees.

4. Click the Save All button on the Standard toolbar to save your changes.

5. Click the Start Debugging button.

   Visual Studio builds the Web site and displays it in Internet Explorer.

6. Click the Home tab on the Web page.

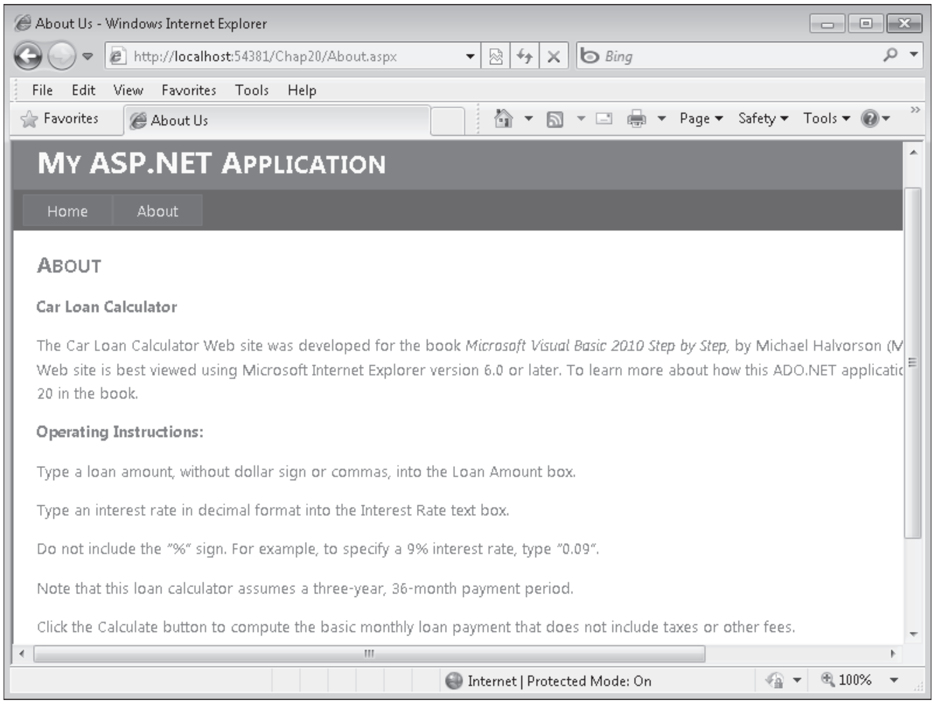
   Visual Studio displays the Home page for your Web site, the car loan calculator.

7. Compute another loan payment to experiment further with the loan calculator.

   If you want to test another set of numbers, try entering **20000** for the loan amount and **0.075** for the interest rate. The result should be $622.12.

8. Now click the About tab to view the About Web page with instructions for your program.

   Internet Explorer displays the About page on the screen. Your browser looks something like this:


About Us - Windows Internet Explorer
http://localhost:54381/Chap20/About.aspx
Bing
File Edit View Favorites Tools Help
Favorites
About Us
Page Safety Tools
MY ASP.NET APPLICATION
Home
About
Internet | Protected Mode: On
100%

ABOUT

Car Loan Calculator

The Car Loan Calculator Web site was developed for the book *Microsoft Visual Basic 2010 Step by Step*, by Michael Halvorson (M
Web site is best viewed using Microsoft Internet Explorer version 6.0 or later. To learn more about how this ADO.NET applicatic
20 in the book.

Operating Instructions:

Type a loan amount, without dollar sign or commas, into the Loan Amount box.

Type an interest rate in decimal format into the Interest Rate text box.

Do not include the "%" sign. For example, to specify a 9% interest rate, type "0.09".

Note that this loan calculator assumes a three-year, 36-month payment period.

Click the Calculate button to compute the basic monthly loan payment that does not include taxes or other fees.

9. Read the text, and then click the Back button in Internet Explorer.

   Just like any Web site, this one lets you click the Back and Forward buttons to jump from one Web page to the next.

10. Close Internet Explorer to close the Web site.

    You've added a simple About page to your Web site, and you have experimented with moving from one page to the next. Pretty cool so far. Now, try something more sophisticated that shows how far you can take your Web site if you choose to include information from a database.

Displaying Database Records on a Web Page

For many users, one of the most exciting aspects of the World Wide Web is the ability to access large amounts of information rapidly through a Web browser. Often, of course, the quantity of information that needs to be displayed on a commercial Web site far exceeds what a developer can realistically prepare using simple text documents. In these cases, Web programmers add database objects to their Web sites to display tables, fields, and records of database information on Web pages, and they connect the objects to a secure database residing on the Web server or another location.

Visual Studio 2010 makes it easy to display simple database tables on a Web site, so as your computing needs grow, you can use Visual Studio to process orders, handle security, manage complex customer information profiles, and create new database records—all from the Web. Importantly, Visual Web Developer delivers this power very effectively. For example, by using the *GridView* control, you can display a database table containing dozens or thousands of records on a Web page without any program code. You'll see how this works by completing the following exercise, which adds a Web page containing loan contact data to the Car Loan Calculator project. If you completed the database programming exercises in Chapter 18, "Getting Started with ADO.NET," and Chapter 19, "Data Presentation Using the *DataGridView* Control," be sure to notice the similarities (and a few differences) between database programming in a Windows environment and database programming on the Web.

Add a new Web page for database information

1. Click the Add New Item command on the Website menu.

   Visual Web Developer displays a list of components that you can add to your Web site.

2. Click the Web Form template, type **FacultyLoanLeads.aspx** in the Name text box, and then click Add.

   Visual Web Developer adds a new Web page to your Web site. You'll customize it with some text and server controls.

3. Click the Design tab to switch to Design view.
4. Enter the following text at the top of the Web page:

   The following grid shows instructors who want loans and their contact phone numbers:

5. Press ENTER twice to add two blank lines below the text.

   Remember that Web page controls are added to Web pages at the insertion point, so it is always important to create a few blank lines when you are preparing to add a control.

Next, you'll display two fields from the *Faculty* table of the Faculty2010.accdb database by adding a *GridView* control to the Web page. *GridView* is similar to the *DataGridView* control you used in Chapter 19, but *GridView* has been optimized for use on the Web. (There are also a few other differences, which you can explore by using the Properties window and Visual Studio Help documentation.) Note that I'm using the same Access database table I used in Chapters 18 and 19, so you can see how similar database programming is in Visual Web Developer. Many programmers also use SQL databases on their Web sites, and Visual Web Developer also handles that format very well.

Add a *GridView* control

1. With the new Web page open and the insertion point in the desired location, double-click the *GridView* control on the Data tab of the Visual Web Developer Toolbox.

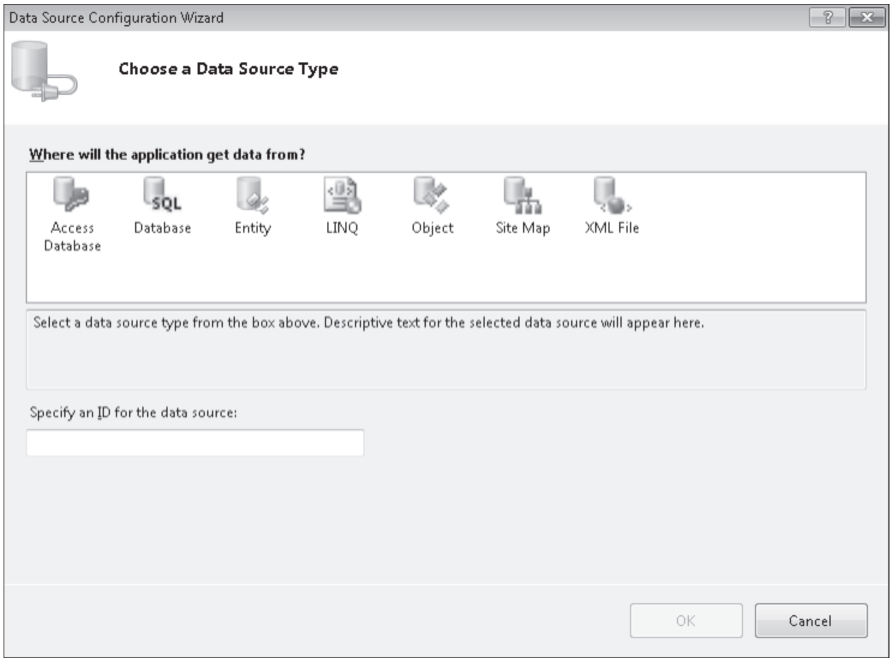
   Visual Web Developer adds a grid view object named *GridView1* to the Web page. The grid view object currently contains placeholder information.

2. If the GridView Tasks list is not already displayed, click the *GridView1* object's smart tag to display the list.

3. Click the Choose Data Source arrow, and then click the <New Data Source> option.

4. Visual Web Developer displays the Data Source Configuration Wizard, a tool that you used in Chapters 18 and 19 to establish a connection to a database and select the tables and fields that will make up a dataset.

   Your screen looks like this:


Data Source Configuration Wizard
Choose a Data Source Type
Where will the application get data from?
Access Database
Database
Entity
LINQ
Object
Site Map
XML File
OK
Cancel

Select a data source type from the box above. Descriptive text for the selected data source will appear here.

Specify an ID for the data source:

5. Click the Access Database icon, type **Faculty2010** in the Specify An ID For The Data Source box, and then click OK.

   You are now prompted to specify the location of the Access database on your system. (This dialog box is slightly different than the one you used in Chapter 18.)

6. Type **C:\Vb10sbs\Chap18\Faculty2010.accdb,** and then click Next.

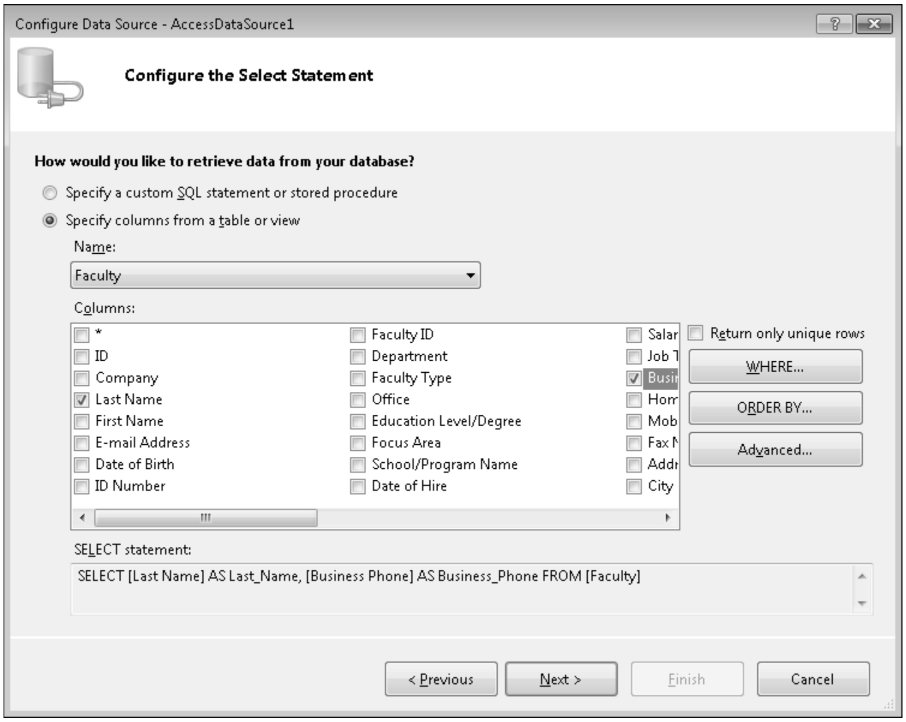
> **Note** If you get a message that says "The Microsoft.ACE.OLEDB.12.0 provider is not registered on the local machine," you might not have Access 2007 or later installed. If you don't have Access 2007 or later installed, you will need to download and install the 2007 Office System Driver: Data Connectivity Components from Microsoft.com.

You are now asked to configure your data source; that is, to select the table and fields that you want to display on your Web page. Here, you'll use two fields from the *Faculty* table. (Remember that in Visual Studio, database fields are often referred to as *columns,* so you'll see the word *columns* used in the IDE and the following instructions.)

7. Click the Name list box arrow, and then click Faculty. (There is probably only one or two database tables here, but if there are several, click the Name arrow to view them.)

8. Select the Last Name and Business Phone check boxes in the Columns list box.

   Your screen looks like this:



SELECT [Last Name] AS Last_Name, [Business Phone] AS Business_Phone FROM [Faculty]

Through your actions here, you are creating an SQL SELECT statement that configures a dataset representing a portion of the Faculty2010.accdb database. You can see the SELECT statement at the bottom of this dialog box.

9. Click Next to see the Test Query screen.
10. Click the Test Query button to see a preview of your data.

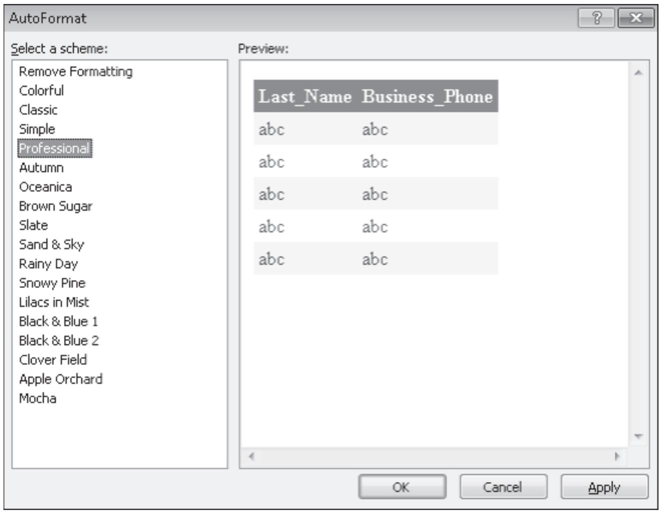
    You'll see a preview of actual *Last Name* and *Business Phone* fields from the database. This data looks as expected, although if we were preparing this Web site for wider distribution, we would take the extra step of formatting the Business Phone column so that it contains standard spacing and phone number formatting.

11. Click Finish.

    Visual Web Developer closes the wizard and adjusts the number of columns and column headers in the grid view object to match the selections that you have made. However, it continues to display placeholder information ("abc") in the grid view cells.

12. With the GridView Tasks list still open, click the Auto Format command.
13. Click the Professional scheme.

    The AutoFormat dialog box looks like this:

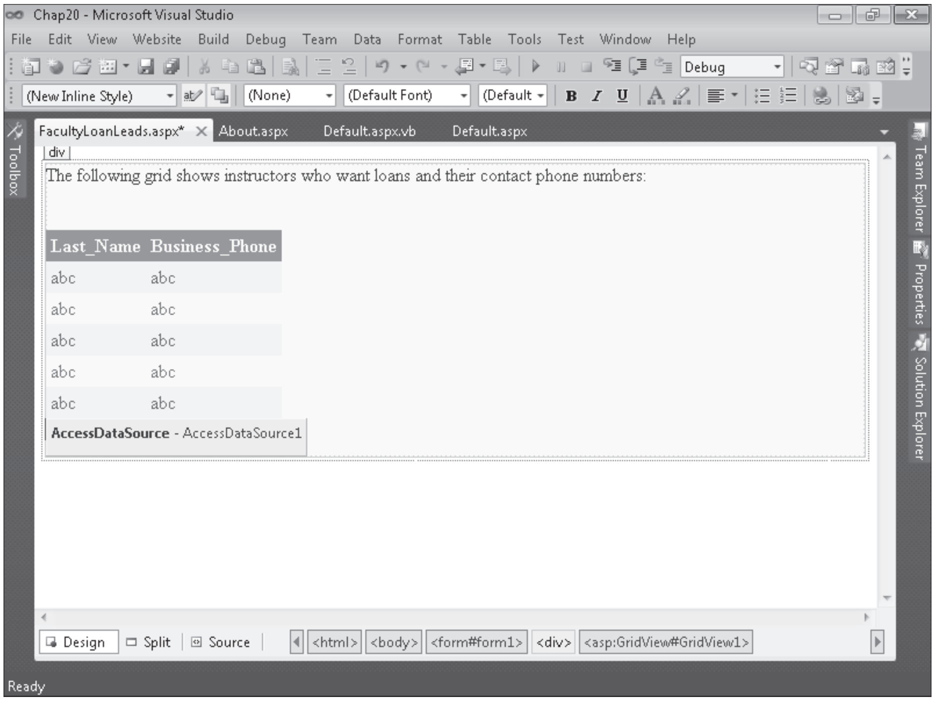


Last_Name	Business_Phone
abc	abc
abc	abc
abc	abc
abc	abc
abc	abc

The ability to format, adjust, and preview formatting options quickly is a great feature of the *GridView* control.

14. Click OK, and then close the GridView Tasks list.

    The FacultyLoanLeads.aspx Web page is complete now, and looks like the screen shot on the following page. (My *GridView* control is within a <*div*> tag, but yours might be within a <*p*> tag.)


Chap20 - Microsoft Visual Studio
File Edit View Website Build Debug Team Data Format Table Tools Test Window Help
Debug
(New Inline Style) (None) (Default Font) (Default
FacultyLoanLeads.aspx* About.aspx Default.aspx.vb Default.aspx
Toolbox
Team Explorer
Properties
Solution Explorer
div
Design Split Source
<html> <body> <form#form1> <div> <asp:GridView#GridView1>
Ready

The following grid shows instructors who want loans and their contact phone numbers:

Last_Name	Business_Phone
abc	abc
abc	abc
abc	abc
abc	abc
abc	abc

AccessDataSource - AccessDataSource1

Now, you'll add a hyperlink on the first Web page (or home page) that will display this Web page when the user wants to see the database table. You'll create the hyperlink with the *HyperLink* control, which has been designed to allow users to jump from the current Web page to a new one with a simple mouse click.

How does the *HyperLink* control work? The *HyperLink* control is located in the Standard Toolbox. When you add a *HyperLink* control to your Web page, you set the text that will be displayed on the page by using the *Text* property, and then you specify the desired Web page or resource to jump to (either a URL or a local path) by using the *NavigateUrl* property. That's all there is to it.

Add a hyperlink to the home page

1. Click the Default.aspx tab at the top of the Designer.

   The home page for your Web site opens in the Designer.

2. Click to the right of the Calculate button object to place the insertion point after that object.

3. Press ENTER to create space for the hyperlink object.
4. Double-click the *HyperLink* control on the Standard tab of the Toolbox to create a hyperlink object at the insertion point.
5. Select the hyperlink object, and then set the *Text* property of the object to "Display Loan Prospects."

   We'll pretend that your users are bank loan officers (or well-informed car salespeople) looking to sell auto loans to university professors. Display Loan Prospects will be the link that they click to view the selected database records.
6. Set the *ID* property of the hyperlink object to "lnkProspects."
7. Click the *NavigateUrl* property, and then click the ellipsis button in the second column.

   The Select URL dialog box opens.
8. Click the FacultyLoanLeads.aspx file in the Contents Of Folder list box, and then click OK.
9. Click Save All to save your changes.

Your link is finished, and you're ready to test the Web site and *GridView* control in your browser.

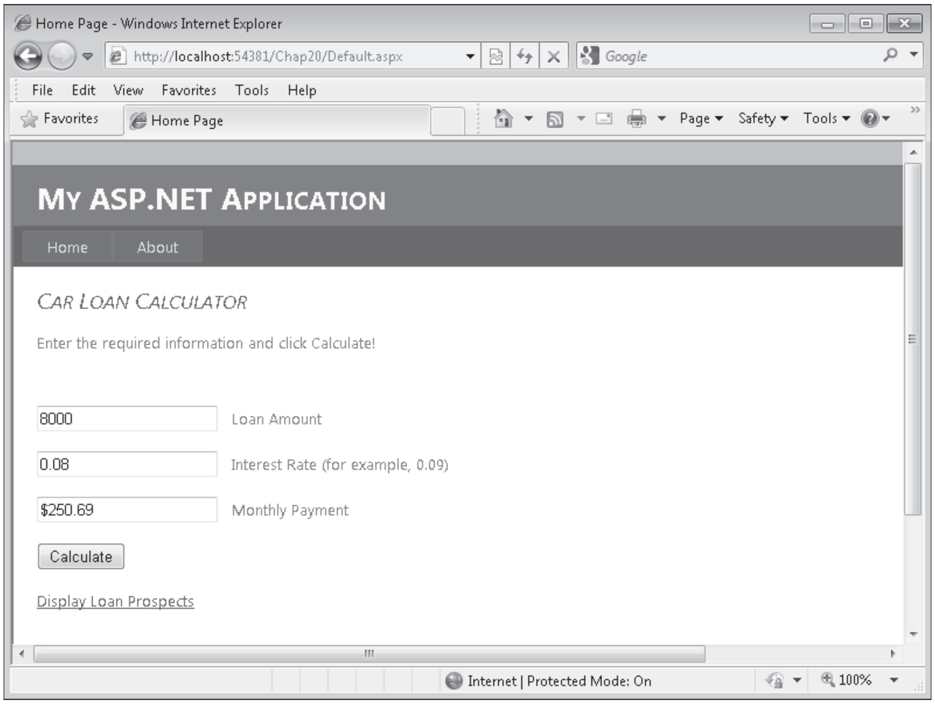
Test the final Car Loan Calculator Web site

Tip The complete Car Loan Calculator Web site is located in the C:\Vb10sbs\Chap20\Chap20 folder. Use the Open Web Site command on the File menu to open an existing Web site.

1. Click the Start Debugging button.

   Visual Studio builds the Web site and displays it in Internet Explorer.
2. Enter **8000** for the loan amount and **0.08** for the interest rate, and then click Calculate.

   The result is $250.69. Whenever you add to a project, it is always good to go back and test the original features to verify that they have not been modified inadvertently. Your screen looks like the screen shot on the following page.


Home Page - Windows Internet Explorer
http://localhost:54381/Chap20/Default.aspx
File Edit View Favorites Tools Help
Favorites
Home Page
Page Safety Tools
MY ASP.NET APPLICATION
Home About

CAR LOAN CALCULATOR

Enter the required information and click Calculate!

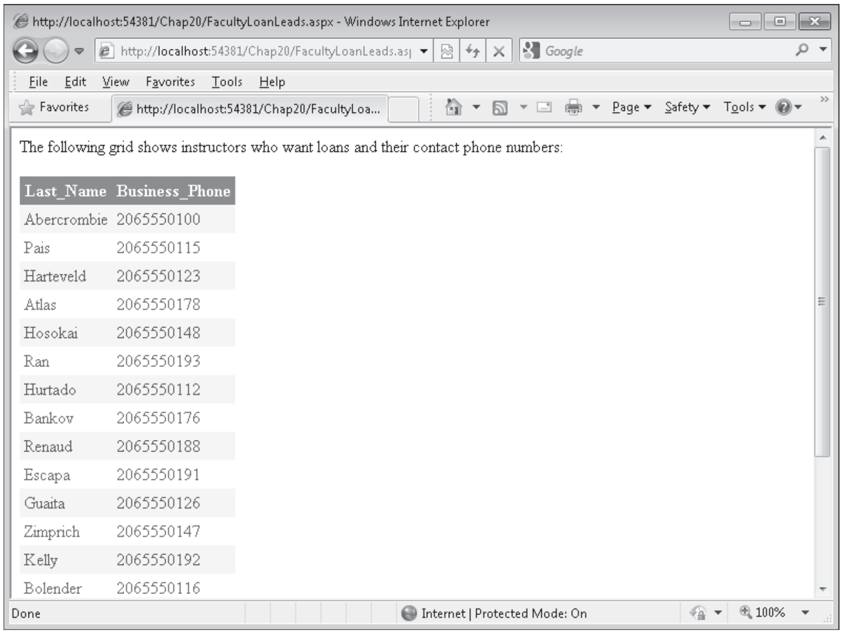
8000	Loan Amount
0.08	Interest Rate (for example, 0.09)
$250.69	Monthly Payment

Calculate
Display Loan Prospects
Internet | Protected Mode: On
100%

The new hyperlink (Display Loan Prospects) is visible at the bottom of the Web page.

3. Click Display Loan Prospects to load the database table.

   Internet Explorer loads the *Last Name* and *Business Phone* fields from the Faculty2010.accdb database into the grid view object. Your Web page looks something like this:


http://localhost:54381/Chap20/FacultyLoanLeads.aspx - Windows Internet Explorer
http://localhost:54381/Chap20/FacultyLoanLeads.as
File Edit View Favorites Tools Help
Favorites
http://localhost:54381/Chap20/FacultyLoa...
Page Safety Tools

The following grid shows instructors who want loans and their contact phone numbers:

Last_Name	Business_Phone
Abercrombie	2065550100
Pais	2065550115
Harteveld	2065550123
Atlas	2065550178
Hosokai	2065550148
Ran	2065550193
Hurtado	2065550112
Bankov	2065550176
Renaud	2065550188
Escapa	2065550191
Guaita	2065550126
Zimprich	2065550147
Kelly	2065550192
Bolender	2065550116

The information is nicely formatted and appears useful. By default, you'll find that the data in this table cannot be sorted, but you can change this option by selecting the Enable Sorting check box in GridView Tasks. If your database contains many rows (records) of information, you can select the Enable Paging check box in GridView Tasks to display a list of page numbers at the bottom of the Web page (like a list that you might see in a search engine that displays many pages of "hits" for your search).

4. Click the Back and Forward buttons in Internet Explorer.

   As you learned earlier, you can jump back and forth between Web pages in your Web site, just as you would in any professional Web site.

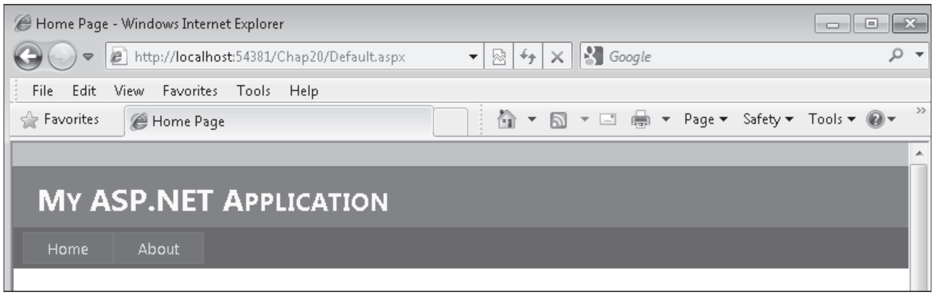
5. When you're finished experimenting, close Internet Explorer to close the Web site.

You've added a table of custom database information without adding any program code!

One Step Further: Setting Web Site Titles in Internet Explorer

Haven't had enough yet? Here are two last Web programming tips to enhance your Web site and send you off on your own explorations.

You might have noticed while testing the Car Loan Calculator Web site that Internet Explorer displayed "Home Page" in the title bar and window tab when displaying your Web site. Your program also displays the very large template title "MY ASP.NET APPLICATION" at the top of the window. In other words, your screen looked like this:


Home Page - Windows Internet Explorer
http://localhost:54381/Chap20/Default.aspx
Google
File Edit View Favorites Tools Help
Favorites
Home Page
Page Safety Tools
MY ASP.NET APPLICATION
Home
About

You can customize what Internet Explorer and other browsers display in the title bar by setting the *Title* property of the *DOCUMENT* object for your Web page; and you can modify the "MY ASP.NET APPLICATION" string by editing the site master page. Give editing both values a try now.

Set the *Title* property

1. With the Default.aspx Web page open in Design view, click the *DOCUMENT* object in the Object list box at the top of the Properties window.

Each Web page in a Web site contains a *DOCUMENT* object that holds important general settings for the Web page. However, the *DOCUMENT* object is not selected by default in the Designer, so you might not have noticed it. One of the important properties for the *DOCUMENT* object is *Title*, which sets the title of the current Web page in the browser.

2. Set the *Title* property to "Car Loan Calculator."

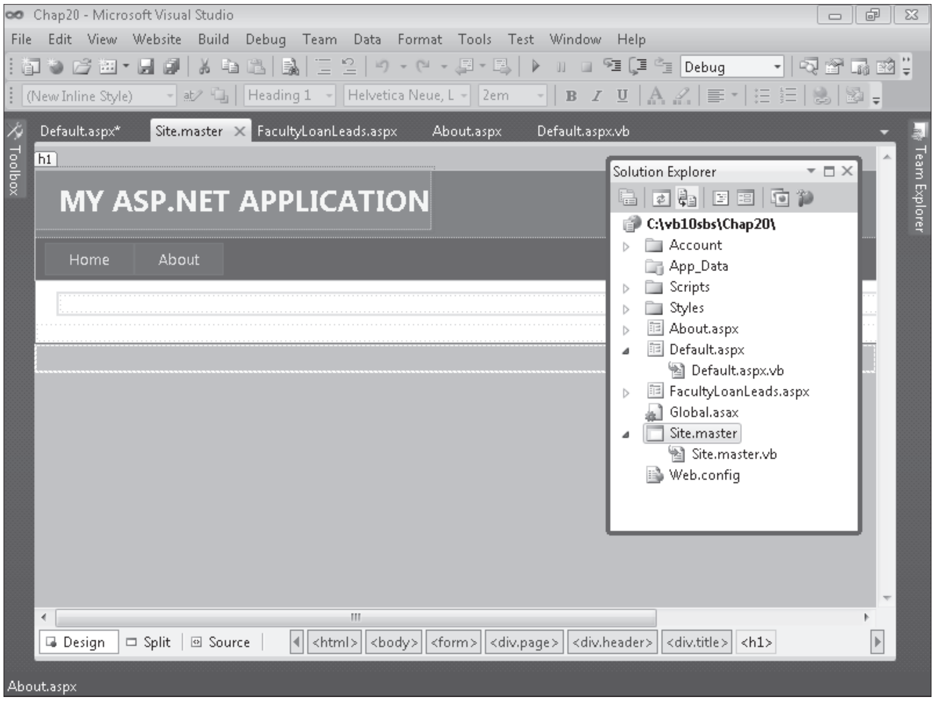
The change does not appear on the screen, but Visual Web Developer records it internally. Now, change the title of your application in the site master page.

Edit the master page title

1. Click the Site.Master file in Solution Explorer, and then click the View Designer button.

   Visual Studio displays the master page in the Designer. The master page is a template that provides default settings for your Web site and lets you adjust characteristics such as appearance, banner titles, menus, and links. For example, you can click smart tags associated with the Web site's menu items and adjust them much as you customized menus in Chapter 4, "Working with Menus, Toolbars, and Dialog Boxes."

   Your screen looks like this:

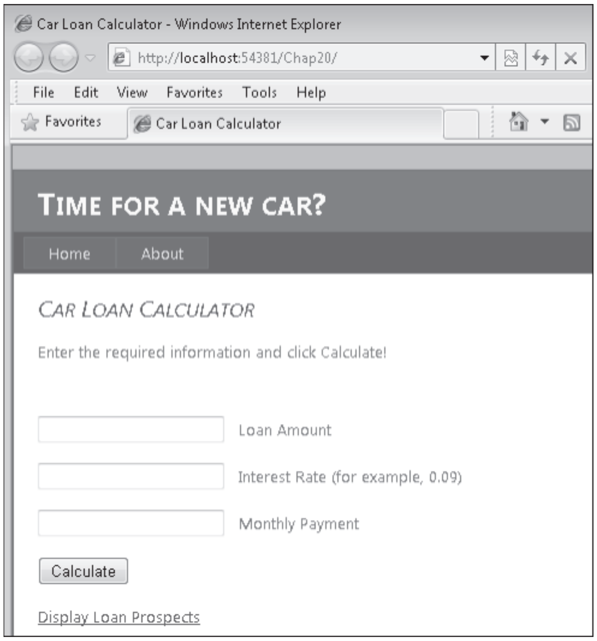

Chap20 - Microsoft Visual Studio
File Edit View Website Build Debug Team Data Format Tools Test Window Help
Debug
(New Inline Style) Heading 1 Helvetica Neue, L 2em
Default.aspx* Site.master FacultyLoanLeads.aspx About.aspx Default.aspx.vb
Toolbox
Team Explorer
h1
MY ASP.NET APPLICATION
Home About
Solution Explorer
C:\vb10sbs\Chap20\
Account
App_Data
Scripts
Styles
About.aspx
Default.aspx
Default.aspx.vb
FacultyLoanLeads.aspx
Global.asax
Site.master
Site.master.vb
Web.config
Design Split Source
<html> <body> <form> <div.page> <div.header> <div.title> <h1>
About.aspx

2. Delete the title "MY ASP.NET APPLICATION" and type **TIME FOR A NEW CAR?**

   Visual Web Designer enters your new title. Now run the Web site again.

3. Click the Start Debugging button.

   Visual Studio opens Internet Explorer and loads the Web site. Now a more useful title bar and banner message appears, as shown in the following screen shot:


Car Loan Calculator - Windows Internet Explorer
http://localhost:54381/Chap20/
File Edit View Favorites Tools Help
Favorites
Car Loan Calculator
TIME FOR A NEW CAR?
Home
About
CAR LOAN CALCULATOR
Enter the required information and click Calculate!
Loan Amount
Interest Rate (for example, 0.09)
Monthly Payment
Calculate
Display Loan Prospects

   Now that looks better.

4. Close Internet Explorer, and then update the *Title* properties for the other Web pages on your Web site.

5. When you're finished experimenting with the Car Loan Calculator, save your changes and close Visual Studio.

Congratulations on completing the entire *Microsoft Visual Basic 2010 Step by Step* programming course! Take a few moments to flip back through this book and see all that you have learned. Now you're ready for more sophisticated Visual Basic challenges and programming techniques. Check out the resource list in the Appendix, "Where to Go for More Information," for a few ideas about continuing your learning. But take a break first—you've earned it!

Chapter 20 Quick Reference

To	Do This
Create a new ASP.NET Web site	Click the New Web Site command on the File menu, click the ASP.NET Web Site template, specify a folder location in the Web Location list box, and then click OK.
Switch between Design view and Source view in the Web Page Designer	Click the Source or Design tabs in the Web Page Designer. For a mixed view, click the Split tab.
Enter text on a Web page	Click the Design tab, and then type the text you want to add.
Format text on a Web page	On the page, select the text that you want to format, and then click a button or control on the Formatting toolbar. Additional formatting options are available on the Format menu.
View the HTML and ASP.NET markup in your Web page	Click the Source tab in the Web Page Designer.
Add controls to a Web page	Display the Web page in Design view, open the Toolbox (which automatically contains Visual Web Developer controls), position the insertion point where you want to place the control on the page, and then double-click the control in the Toolbox.
Change the name of an object on a Web page	Use the Properties window to change the object's *ID* property to a new name.
Write the default event procedure for an object on a Web page	Double-click the object to display the code-behind file, and then write the event procedure code for the object in the Code Editor.
Verify the format of the data entered by the user into a control on a Web page	Use one or more validator controls from the Validation tab of the Toolbox to test the data entered in an input control.
Run and test a Web site in Visual Studio	Click the Start Debugging button on the Standard toolbar. Visual Studio builds the project, starts the ASP.NET Development Server, and loads the Web site in Internet Explorer.
Create a Web page for a project	Click the Add New Item command on the Website menu, and then add a new Web Form or an HTML Page template to the project. Create and format the page by using the Web Page Designer.
Create a link to other Web pages on your Web site	Add a *HyperLink* control to your Web page, and then set the control's *NavigateUrl* property to the address of the linked Web page.
Display database records on a Web page	Add a *GridView* control to a Web page in the Web Page Designer. Establish a connection to the database and format the data by using commands in the GridView Tasks list. (The Choose Data Source command starts the Data Source Configuration Wizard.)
Set the title displayed for Web pages on the Internet Explorer title bar	For each Web page, use the Properties window to set the *DOCUMENT* object's *Title* property.
Adjust the banner title, menus, and other default values in the master page	Select the Site.Master file in Solution Explorer, and then click View Designer. Adjust the master page's default values in the Designer.

Index

Michael Halvorson

Michael Halvorson is the author or co-author of more than 35 books, including *Microsoft Visual Basic 2008 Step by Step*, *Microsoft Office XP Inside Out*, and *Microsoft Visual Basic 6.0 Professional Step By Step*. Halvorson has been the recipient of numerous non-fiction writing awards, including the Computer Press Best How-to Book Award (Software category) and the Society for Technical Communication Excellence Award (Writing category). Halvorson earned a bachelor's degree in Computer Science from Pacific Lutheran University in Tacoma, Washington, and master's and doctoral degrees in History from the University of Washington in Seattle. He was employed at Microsoft Corporation from 1985 to 1993, and he has been an advocate for Visual Basic programming since the product's original debut at Windows World in 1991. Halvorson is currently an associate professor at Pacific Lutheran University.