Microsoft

Start Here!™

Learn Microsoft® Visual C#® 2010

John Paul Mueller

Ready to learn programming?

Start Here!™

Learn the fundamentals of modern programming with Visual C# 2010—and begin building your first apps for the desktop and web. If you have absolutely no previous experience, no problem—simply start here! This book introduces must-know concepts and techniques through easy-to-follow explanations, examples, and exercises.

Here's where you start learning Visual C#

- Learn how an application performs tasks by tracing its code
- Query and manipulate application data with LINQ
- Access web services with REST and SOAP
- Build simple apps with Windows® Presentation Foundation
- Explore rich Internet apps with Microsoft Silverlight®
- Find and fix errors by debugging your applications
- Put it all together by creating your first programs

GET CODE SAMPLES ONLINE
Ready to download at
http://go.microsoft.com/FWLink/?Linkid=229177

For system requirements, see the ***Introduction.***

PLUS—Download your companion reference, *Start Here! Fundamentals of Microsoft .NET Programming*, using the instruction page at the back of the book.

ISBN: 978-0-7356-5772-4
9 780735 657724 90000

U.S.A. $34.99
Canada $36.99
[*Recommended*]

Programming/Microsoft Visual Studio

Start Here! Learn Microsoft® Visual C#® 2010

DEVELOPER ROADMAP

Start Here
- Beginner-level instruction
- Easy to follow explanations and examples
- Exercises to build your first projects

Step by Step
- For experienced developers learning a new topic
- Focus on fundamental techniques and tools
- Hands-on tutorial with practice files plus eBook

Developer Reference
- Professional developers; intermediate to advanced
- Expertly covers essential topics and techniques
- Features extensive, adaptable code examples

Focused Topics
- For programmers who develop complex or advanced solutions
- Specialized topics; narrow focus; deep coverage
- Features extensive, adaptable code examples

microsoft.com/mspress

About the Author

John Paul Mueller makes his living explaining highly technical topics to others. He's written 88 tech books and more than 300 articles on Windows programming, .NET security, and database management (among other topics), for publishers and magazines including *Visual C++ Developer*, *asp.netPRO*, and *Visual Basic Developer*.

Microsoft®

Microsoft

Start Here!™

Learn Microsoft®

Visual C#® 2010

John Paul Mueller

ISBN: 978-0-7356-5772-4

1 2 3 4 5 6 7 8 9 LSI 6 5 4 3 2 1

Printed and bound in the United States of America.

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Book Support at *mspinput@microsoft.com*. Please tell us what you think of this book at *http://www.microsoft.com/learning/booksurvey*.

Acquisitions and Developmental Editor: Russell Jones

Production Editor: Teresa Elsey

Editorial Production: S4Carlisle Publishing Services

Technical Reviewer: Russ Mullen

Indexer: WordCo Indexing Services, Inc.

Cover Design: Jake Rae

Cover Composition: Karen Montgomery

This book is dedicated to our beagle, Reese—the peanut butter dog. She's the guardian of the orchard, checker of the fire, and warmer of the lap. Her incredibly soft fur amazes and soothes at the same time.

Contents at a Glance

Contents

Introduction

C# IS AN AMAZING C-LIKE language that has almost all of the flexibility of C and C++, without any of the arcane programming rules. You can create applications quickly and easily using C#. The mixture of the Visual Studio Integrated Development Environment (IDE) aids and the natural flow of the language itself makes working with C# possible for even the complete novice. As your skills grow, you'll find that C# grows with you and makes nearly any kind of application possible, even applications that you normally don't associate with higher level languages.

Start Here! Learn Microsoft Visual C# 2010 is your doorway to discovering the joys of programming in C# without the usual exercises and rote learning environment of a college course. Instead of boring regimen, you begin programming immediately in Chapter 1, "Getting to Know C#." In fact, you'll create three completely different applications in Chapter 1 alone, which makes this book different from other novice-level books on the market. Yes, the examples are decidedly simple to begin with, but it won't take you long to begin interacting with web services, creating Silverlight applications, and working at the command line.

What's truly amazing about this book is that every tool it uses is free. You'll discover an amazing array of C# application types and it won't cost you a penny to uncover them. These aren't old school techniques either—you'll use the newest methods of creating applications such as working with Language INtegrated Query (LINQ) to ask the application to supply data to you. Of course, the techniques you learn will transfer easily to the paid versions of Microsoft's products that include a great deal more capability and provide better flexibility.

Who Should Read This Book

The focus of this book is to learn by doing. If you're a hands-on sort of a person and find other texts boring and difficult, this is the book for you. Every example is completely explained and you'll use a special tracing method to discover the inner secrets of each programming technique. You'll at least encounter most basic application types by the time you've completed this book.

Assumptions

This book was conceived and created for the complete novice—someone who has no programming experience at all. It is also appropriate for someone has been exposed to another language, but lacks significant experience in that language. This book uses a hands-on training approach, so you're not going to be doing a lot of reading—you'll be trying everything out as part of the learning process. Therefore, you need to have a system that's capable of running the tools and a desire to use that system during your learning process.

You should be able to work with Windows as an operating system. The book assumes that you know how to work with a mouse and that you've worked with other applications that have basic features such as a File menu. Even though this book is for the complete novice from an application development perspective, it doesn't do a lot of hand-holding when it comes to working with basic Windows functionality.

Who Should Not Read This Book

You're going to be disappointed if you're an advanced programmer and interested in learning C# as a second language. The examples in this book are relatively basic, and the explanations are kept simple. Developers who have a lot of experience will feel that I'm exploring the obvious—but what is obvious to experienced programmers often isn't obvious at all to someone who is just learning to write code.

Organization of This Book

Start Here! Learn Microsoft Visual C# 2010 uses a hands-on approach to learning where readers actually trace through applications and discover how they work by seeing them perform tasks. Because this book is targeted toward complete novices, it should be read sequentially; later chapters require knowledge covered in previous chapters. I strongly suggest starting at the first chapter and working forward through the book. If you do have some experience with another language, you could possibly start at Chapter 3. This book provides the following topics.

- **Chapter 1: Getting to Know C#** You'll create three desktop applications in this chapter that show the sorts of things that C# is capable of doing. Part of this process is learning how to trace through applications so that you can see how they perform the tasks that they do, so you'll learn the tracing technique

used throughout the rest of the book in this chapter. This chapter also helps you download and install the tools you need to work with C#.

- **Chapter 2: Developing a Web Project** In addition to the desktop applications introduced in Chapter 1, it's also possible to create web applications using C#. This chapter shows two completely different web applications that will help you understand the small differences involved in tracing through web applications. You'll also learn how to download and install the tools used to create web applications.
- **Chapter 3: Using Simple Data Manipulation Techniques** The first two chapters help acquaint you with C# on the desktop and the web. This chapter exposes you to the main purpose behind most applications—data manipulation. You'll use a new technique to manipulate data that relies on LINQ. The five examples in this chapter emphasize the fact that data manipulation need not be hard.
- **Chapter 4: Using Collections to Store Data** Although Chapter 3 focuses on simple data, this chapter begins showing you how to work with complex data. You'll discover how to create containers to store similar data together. This chapter contains three examples that emphasize three different types of data storage.
- **Chapter 5: Working with XML** It seems as if just about everything runs on the eXtensible Markup Language (XML) today. The four examples in this chapter show you how to work with XML files so that you can do things like save application settings and work with web services.
- **Chapter 6: Accessing a Web Service** Web services make it possible to obtain data through a remote connection. Often this connection relies on the Internet, but web services are everywhere. In fact, you'll be surprised at how many free web services exist and the impressive range of data you can access through them. The two examples in this chapter show you how to use the two techniques, REpresentational State Transfer (REST) and Simple Object Access Protocol (SOAP), that C# provides to access web services.
- **Chapter 7: Using the Windows Presentation Foundation** Windows Presentation Foundation (WPF) is a new way to create applications with C#. It helps you create applications with impressive interfaces and new features that aren't available using older C# development methods. The four examples in this chapter emphasize techniques that you can use to create great applications using WPF.

- **Chapter 8: Working with Libraries** At some point you'll want to reuse some of the code you create. Libraries provide the means for reusing code easily and in a standardized way. The example in this chapter shows how to create and use a library as part of an application.
- **Chapter 9: Creating Utility Applications** Many people haven't used the command line, but most administrators are at least aware of it. The command line makes it possible to type a single command that performs tasks that would require multiple mouse clicks. The example in this chapter shows how to create applications that have a command-line interface so that you can work with them quickly and automate them in various ways.
- **Chapter 10: Using LINQ in Web Applications** Earlier chapters explored the use of LINQ in desktop applications. Fortunately, it's quite easy to use LINQ in web applications, too. You use LINQ for the same purpose—to ask the application to supply certain types of data. The three examples in this chapter show different ways to use LINQ in a web application.
- **Chapter 11: Working with Silverlight Applications** Silverlight applications can perform amazing tasks. You can create them to work in either a browser or at the desktop. The technology works with multiple browsers and on multiple platforms. In short, you can use Silverlight to transform your C# application into something that works everywhere. The two examples in this chapter help you understand the basics of Silverlight development using C#.
- **Chapter 12: Debugging Applications** Throughout the book you've used tracing techniques to discover how applications work. Debugging is a step further. When you debug an application, you look for errors in it and fix them. The example in this chapter extends what you already know about tracing to make it easier to begin debugging your applications.

Free eBook Reference

When you purchase this title, you also get the companion reference, *Start Here!™ Fundamentals of Microsoft® .NET Programming*, for free. To obtain your copy, please see the instruction page at the back of this book.

The *Fundamentals* book contains information that applies to any programming language, plus some specific material for beginning .NET developers.

As you read through this book, you'll find references to the *Fundamentals* book that look like this:

For more information, see <topic> in the accompanying Start Here! Fundamentals of Microsoft .NET Programming *book.*

When you see a reference like this, if you're not already familiar with the topic, you should read that section in the *Fundamentals* book. In addition, the *Fundamentals* book contains an extensive glossary of key programming terms.

Conventions and Features in This Book

This book presents information using conventions designed to make the information readable and easy to follow:

- This book relies heavily on procedures to help you create applications and then trace through them to see how they work. Each procedure is in a separate section and describes precisely what you'll accomplish by following the steps it contains.
- Boxed elements with labels such as "Note" provide additional information or alternative methods for completing a step successfully. Make sure you pay special attention to warnings because they contain helpful information for avoiding problems and errors.
- Text that you type (apart from code blocks) appears in **bold**.
- 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 the Tab key.
- A vertical bar between two or more menu items (such as File | Close), means that you should select the first menu or menu item, then the next, and so on.

System Requirements

You will need the following hardware and software to work through the examples in this book:

- One of following operating systems: Windows XP with Service Pack 3 (except Starter Edition), Windows Vista with Service Pack 2 (except Starter Edition), Windows 7, Windows Server 2003 with Service Pack 2, Windows Server 2003 R2, Windows Server 2008 with Service Pack 2, or Windows Server 2008 R2
- Visual C# 2010 Express edition
- Visual Web Developer 2010 Express edition
- A computer that has a 1.6 GHz or faster processor (2 GHz recommended)
- 1 GB (32 Bit) or 2 GB (64 Bit) RAM (Add 512 MB if running in a virtual machine or SQL Server Express editions, more for advanced SQL Server editions.)
- 3.5 GB of available hard disk space
- 5400 RPM hard disk drive
- DirectX 9 capable video card running at 1024 x 768 or higher-resolution display
- DVD-ROM drive (if installing Visual Studio from DVD)
- An Internet connection to download software or chapter examples

Depending on your Windows configuration, you might require Local Administrator rights to install or configure Visual C# 2010 Express edition and Visual Web Developer 2010 Express edition products.

Code Samples

Most of the chapters in this book include exercises that let you interactively try out new material learned in the main text. All sample projects, in both their pre-exercise and post-exercise formats, can be downloaded from the following page:

http://www.microsoftpressstore.com/title/9780735657724

Follow the instructions to download the Start_Here_CSharp_Sample_Code.zip file.

Note In addition to the code samples, your system should have Visual Studio 2010 and SQL Server 2008 installed. The instructions below use SQL Server Management Studio 2008 to set up the sample database used with the practice examples. If available, install the latest service packs for each product.

Installing the Code Samples

Follow these steps to install the code samples on your computer so that you can use them with the exercises in this book.

1. Unzip the Start_Here_CSharp_Sample_Code.zip file that you downloaded from the book's website. (Name a specific directory along with directions to create it, if necessary.)
2. If prompted, review the displayed end user license agreement. If you accept the terms, select the accept option, and then click Next.

Note If the license agreement doesn't appear, you can access it from the same webpage from which you downloaded the Start_Here_CSharp_Sample_Code.zip file.

Using the Code Samples

The folder created by the Setup.exe program creates a book folder named "Start Here! Programming in C#" that contains 12 subfolders—one for each of the chapters in the book. To find the examples associated with a particular chapter, access the appropriate chapter folder. You'll find the examples for that chapter in separate subfolders. Access the folder containing the example you want to work with. (These folders have the same names as the examples in the chapter.) For example, you'll find an example called "No-Code Windows Forms" in the "Create a New Windows Forms Application Project" section of Chapter 1 in the \Start Here! Programming in C#\Chapter 01\No Code Windows Forms folder on your hard drive. If your system is configured to display file extensions of the C# project files, use .sln as the file extension.

Acknowledgments

Thanks to my wife, Rebecca, for working with me to get this book completed. I really don't know what I would have done without her help in researching and compiling some of the information that appears here. She also did a fine job of proofreading my rough draft. Rebecca keeps the house running while I'm buried in work.

Russ Mullen deserves thanks for his technical edit of this book. He greatly added to the accuracy and depth of the material you see here. Russ is always providing me with great URLs for new products and ideas. However, it's the testing Russ does that helps most. He's the sanity check for my work. Russ also has different computer equipment from mine, so he's able to point out flaws that I might not otherwise notice.

Matt Wagner, my agent, deserves credit for helping me get the contract in the first place and taking care of all the details that most authors don't really consider. I always appreciate his assistance. It's good to know that someone wants to help.

A number of people read all or part of this book to help me refine the approach, test the coding examples, and generally provide input that all readers wish they could have. These unpaid volunteers helped in ways too numerous to mention here. I especially appreciate the efforts of Eva Beattie and Osvaldo Téllez Almirall, who provided general input, read the entire book, and selflessly devoted themselves to this project. I also appreciated Rod Stephen's input on a number of questions.

Finally, I would like to thank Russell Jones, Dan Fauxsmith, Christian Holdener, Becka McKay, Christie Rears, and the rest of the editorial and production staff at O'Reilly for their assistance in bringing this book to print. It's always nice to work with such a great group of professionals. This is my first book with this group and I hope we get to work together again in the future.

Errata & Book Support

We've made every effort to ensure the accuracy of this book and its companion content. Any errors that have been reported since this book was published are listed on our Microsoft Press site:

http://www.microsoftpressstore.com/title/9780735657724.

If you find an error that is not already listed, you can report it to us through the same page.

If you need additional support, email Microsoft Press Book Support at *mspinput@microsoft.com*.

Please note that product support for Microsoft software is not offered through the addresses above.

We Want to Hear from You

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

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

The survey is short, and we read every one of your comments and ideas. Thanks in advance for your input!

Stay in Touch

Let's keep the conversation going! We're on Twitter: *http://twitter.com/MicrosoftPress*.

Chapter 1

Getting to Know C#

After completing this chapter, you'll be able to:

- Install all of the tools required to use C# to develop applications.
- Start Visual Studio 2010 Express so that you can use it to create applications.
- Create and explore a standard desktop application without using any code.
- Create and explore a Windows Presentation Foundation (WPF) application without using any code.

C# IS AN INCREDIBLE LANGUAGE. You can use it to create just about any kind of application—desktop, web, or mobile—using less code than you're likely to need with just about any other language. However, as shown in this chapter, you may not even need to write much code; the Visual Studio Integrated Development Environment (IDE) provides a graphical interface that also writes code for you in the background. Amazing! You design how you want the program to look, then you inform the IDE about behaviors the application should have—and then the IDE writes the code for you! This chapter walks you through several no-code examples that actually do something useful. With that said, normally you'll write at least *some* code to create most applications.

Of course, before you can create a C# application, you need some sort of tool to create it with. (Technically, you could write an application using Notepad and compile it at the command line, but that's a lot of work, especially when you can obtain a tool free and use it to write useful applications the easy way.) The first section of this chapter shows how to download and install the tools you need for the rest of the examples in the book. If you already have a full version of Visual Studio installed on your system, you can skip the first section of this chapter and move right to the "Starting Visual C# 2010 Express" section.

This chapter doesn't tell you absolutely everything there is to know about the IDE; it does provide some basics to get you started. The second section of the chapter helps you launch Visual C# 2010

Express the first time; you can then look around to see what it provides. Don't worry, you'll learn a great deal more about the features of this IDE before you get through the book.

After the IDE walkthrough, the remainder of the chapter focuses on the three no-code desktop application examples. The IDE does write some code for you, and you'll examine that as part of working through the examples. The best way to learn about coding is to try things out and explore code written by someone else; this book allows you to do both.

Obtaining and Installing Visual Studio 2010 Express

Before you can do anything with C#, you need an environment in which to work. Fortunately, you can obtain a free working environment, Visual Studio 2010 Express, directly from Microsoft. After you install the required products, you'll be able to work with any of the examples in this book and be on your way to a new world of developing applications.

Downloading the Products

Microsoft produces a number of Express products that you can download from *http://www.microsoft.com/express/Downloads/*, but for the purposes of this book you need to download only the following items:

Important You should download and install the packages from the download link in the order listed here.

- **Visual C# 2010 Express** Provides a Visual Studio IDE suitable for developing C# applications.
- **Visual Web Developer 2010 Express** Provides a Visual Studio IDE and other tools that help you develop web applications.
- **Visual Studio 2010 Service Pack 1** Fixes bug in the two Visual Studio Express versions. You should install this last.

The download for Visual C# 2010 Express simply produces a file on your hard drive. The Visual Web Developer 2010 Express download also installs the product for you. As part of the Visual Web Developer 2010 Express installation, you also get the Microsoft Web Platform Installer; because it's part of the package you don't need to perform a separate download to obtain it. But make sure you download and install both the C# and Visual Web Developer Express versions *before* you download and install Visual Studio 2010 Service Pack 1. The next three sections provide detailed instructions for installing all three products, so you can follow along or simply follow the prompts yourself.

Note You must have an Internet connection to install the products described in this chapter. In all cases, the installer will rely on this connection to download product features as part of the installation process.

Installing Visual C# 2010 Express

To download Visual C# Express, click the bullet next to its entry on the download page, *http://www.microsoft.com/express/Downloads*. When you select a language from the drop-down list, the page starts the download automatically. The initial download is only 3.1 MB, so it won't take long. (The installer will download 104 MB more data during the installation process.) Double-click the vcs_web.exe file when the download completes. (Click Yes if you see the User Account Control dialog box.) You'll see a Setup dialog box appear for a few minutes. When you see the Welcome To Setup dialog box, you can start the installation process described in the following steps.

Note The sizes of the file downloads in this chapter are approximate and will probably change with time. The main reason for including them is to give you some idea of how large a download will be and how long it will take.

Performing the Visual C# 2010 Express Installation

1. Click Next. The License Terms dialog box appears.
2. Read the licensing terms, select I Have Read And Accept The License Terms, and click Next. The Destination Folder dialog box appears. Normally, the default destination works fine and that's the assumption this book makes when telling you about Visual C# 2010 Express-specific folders. Therefore, unless you have a good reason to change the default folder, accept the default.
3. Click Install. The installer begins downloading the required files from the Internet. The download is 45 MB, so it may take a few minutes to complete. The actual installation process begins automatically when the download is complete. So get a cup of coffee, grab your favorite magazine, and kick back for a few minutes. At some point, a dialog box appears, indicating that the installation is complete.
4. Click Exit. You're now ready to create desktop applications using Visual C# 2010 Express!

Installing Visual Web Developer 2010 Express

To download Visual Web Developer 2010 Express, click the bullet next to its entry on the download page. Click Install. You'll see a Microsoft web page where you can install the Microsoft Web Platform Installer. Click Install Now to start the download process. After a few minutes, you'll have a file named

Vwd.exe on your system. Double-click this file to open and start the installer. (Click Yes if the User Account Control dialog box appears.) The installer downloads some additional files and installs them automatically, after which you see the Web Platform Installer 3.0 dialog box shown in Figure 1-1.

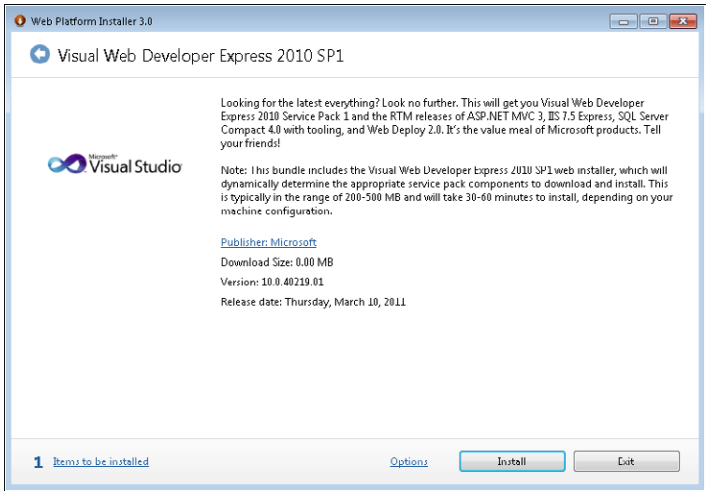
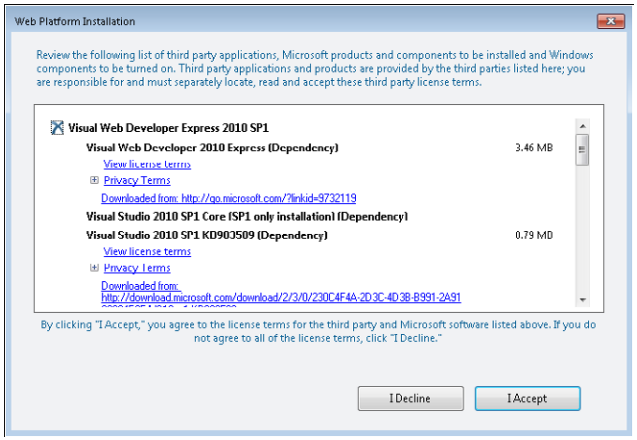


FIGURE 1-1 The Web Platform Installer starts the Visual Web Developer 2010 Express installation.

You're ready to begin installing Visual Web Developer 2010 Express. The following steps take you through the installation process:

Performing the Visual Web Developer 2010 Express Installation

1. Click Install. You'll see the Web Platform Installation dialog box shown here.

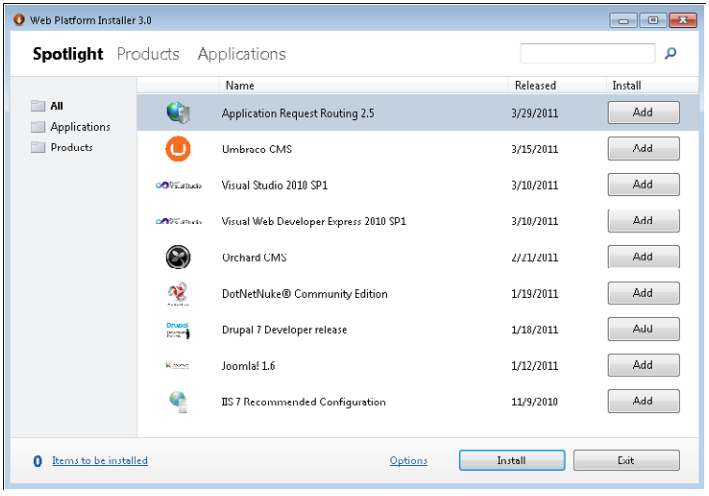


This dialog box contains a list of the applications that the installer will download and install to create a Visual Web Developer 2010 Express installation for you. Many of the items have links

for privacy and licensing terms. You'll need to read the privacy and licensing terms for each product before you proceed so that you know the requirements of using that product.

Note Don't change the default installation selections. For example, you won't need a copy of SQL Server to work through the examples in this book. Configuring these other items can prove difficult in some cases, so this is one situation where the default installation is best.

2. Read the privacy and licensing terms. Click I Accept. The installer will begin downloading and installing each of the products in the list for you automatically. This process will take a while, so you can gaze out the window and contemplate your weekend activities while whistling a merry tune. Eventually, you'll see the Web Platform Installer 3.0 dialog box shown here, from which you can install additional products. At this point, Visual Web Developer 2010 Express is installed and ready.



3. For this book, you don't need to install any additional products, so click Exit.

Installing Visual Studio 2010 Service Pack 1

It's possible that the newly downloaded and installed copy of Visual C# 2010 Express and Visual Web Developer 2010 Express will already have Service Pack 1 (SP1) installed. You can check for this requirement by looking at the About dialog box for each of the applications (click Help | About to see the dialog box). Of course, you might have an older copy of these Express products, or have another Visual Studio product installed on your system. The various IDEs won't start until all your Visual Studio products have SP1 installed, so check for the SP1 compliance and follow the instructions in this section only if you actually need them. In the event of a problem, a dialog box like the one shown in Figure 1-2 appears.

FIGURE 1-2 You'll see this dialog box if the Service Pack 1 installation fails.

To download Visual Studio Service Pack 1, click the bullet next to its entry on the download page. Click Install. You'll see another page load. Click Download on this page to start the download. After the download is complete, double-click the file VS10sp1-KB983509.EXE to begin the installation process. (Click Yes if the User Account Control dialog box appears.) At this point, the installation proceeds automatically. Click Finish when the installation completes.

Starting Visual C# 2010 Express

An Integrated Development Environment (IDE) provides an environment that contains tools to help you create applications. It provides editors (to write code), designers (to lay out graphical elements), a compiler (to create executable code), a debugger (to find mistakes in your code), and other tools that make the development process easier. The Visual C# 2010 Express IDE helps you create desktop applications, which is the focus of this chapter.

Note You need to register both Visual C# 2010 Express and Visual Web Developer 2010 Express. The products you download will only run for 30 days without registration. Registration is free. All you need to do is choose Help | Register Product and follow the instructions to register the applications.

Now that you have a copy of the IDE installed on your computer, it's time to start it to see what it looks like. To start Visual C# 2010 Express, choose Start | All Programs | Microsoft Visual Studio 2010 Express | Microsoft Visual C# 2010 Express. You'll see the IDE start up shown in Figure 1-3.

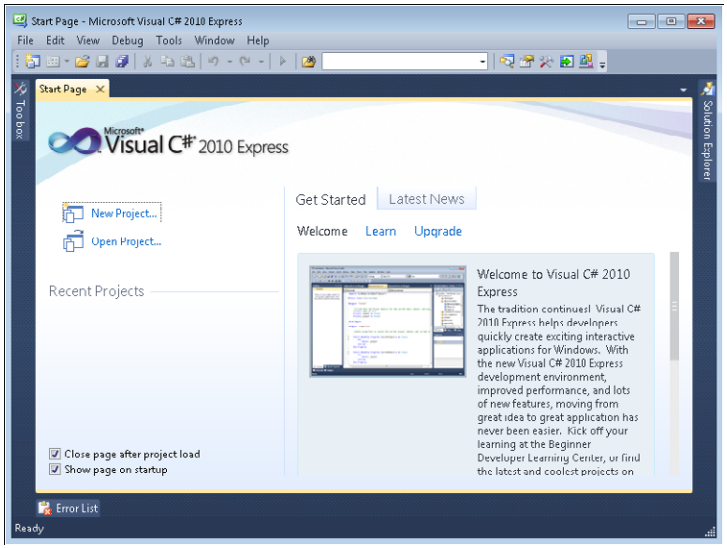


FIGURE 1-3 The Visual Studio IDE opens with the Start Page showing.

This first view of Visual C# 2010 Express is the one that you'll see every time you start the IDE. The left side of the page contains links for creating new projects or opening existing projects. After you have created some applications, you'll also see a list of applications you've recently worked with, which makes it fast and easy to open current projects. On the bottom left are options to close the Start page after you open a project (to reduce clutter) and to display the Start page every time the IDE opens. Generally, you'll leave these options set as shown in the figure to make your work environment efficient.

The right side of the Start page contains helpful information. The first tab contains information you can use to get started using C# more quickly. The second tab provides access to the latest information about C#; however, to see this information, you must click Enable RSS Feed. The page will automatically update with the latest information.

Tip Opening the latest information in the IDE can slow things down at times. A better option is to add the RSS feed to Outlook (or the RSS feed reader of your choice) by following these steps: Make sure Outlook is running. Copy the URL from the RSS Feed field and paste it into your browser's address field. Press Enter, and after a few seconds your browser will ask if you want to add the RSS feed to Outlook. Click Yes.

Creating the No-Code Web Browser

Desktop applications have been around for a long time. Initially, developers had to write all sorts of weird code to make them work, but modern IDEs make it possible to create most applications in significantly less time. This example demonstrates the Windows Forms approach, which is the approach that Windows developers have used for many years to create applications. This particular example shows how to create a fully functional Web browser. You'll actually be able to use it to surf the Internet should you desire to do so.

Understanding the Benefits of Windows Forms

Windows Forms technology has been around for many years, and it's incredibly stable. In addition, most developers have created a Windows Forms application sometime in their career. The combination of long use and familiarity make Windows Forms applications a good starting point for anyone. One of the more important reasons to create a Windows Forms application is that you have access to an astonishing array of controls and tools. If you need to support older platforms, Windows Forms is also the best choice for compatibility reasons. You don't need anything special installed on older systems to use a Windows Forms application except the version of the .NET Framework required by the application. The .NET Framework contains the code that makes C# and other .NET languages run. It is available wherever you need it. In short, even though Windows Forms applications are older technology, they're still relevant for developers today. Microsoft plans to continue supporting Windows Forms applications into the foreseeable future, so you certainly don't need to worry about the practicality of this approach for your next application.

Creating a New Windows Forms Application Project

You always begin a new project by opening the IDE and then clicking the New Project link. The IDE displays the New Project dialog box shown in Figure 1-4.

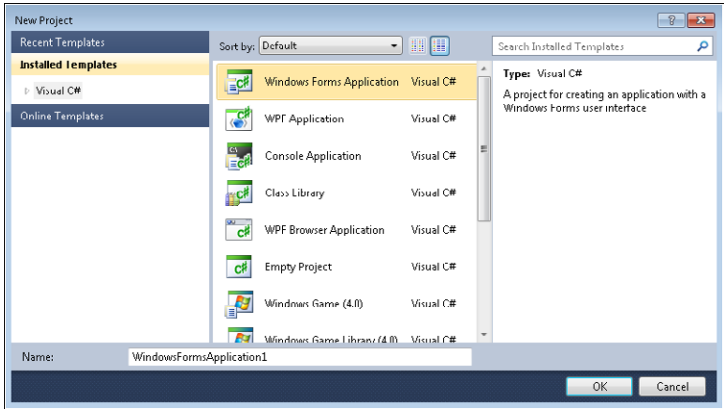


FIGURE 1-4 The New Project dialog box contains the templates you use to create new applications.

The left pane contains a list of template folders. Each folder contains a particular group of templates. In this case, you're interested in the Visual C# folder. The center pane shows the templates contained within the selected template folder. Because this project is about creating a Windows Forms application, highlight the Windows Forms Application template. The right pane contains information about the selected template.

Every project requires a name—preferably something better than the default *WindowsForms Application1*. Always give your projects a descriptive name so that you always know what they contain. In this case, type **No-Code Windows Forms** in the Name field. The name is a little long, but descriptive. Click OK and the IDE creates a new project for you like the one shown in Figure 1-5.

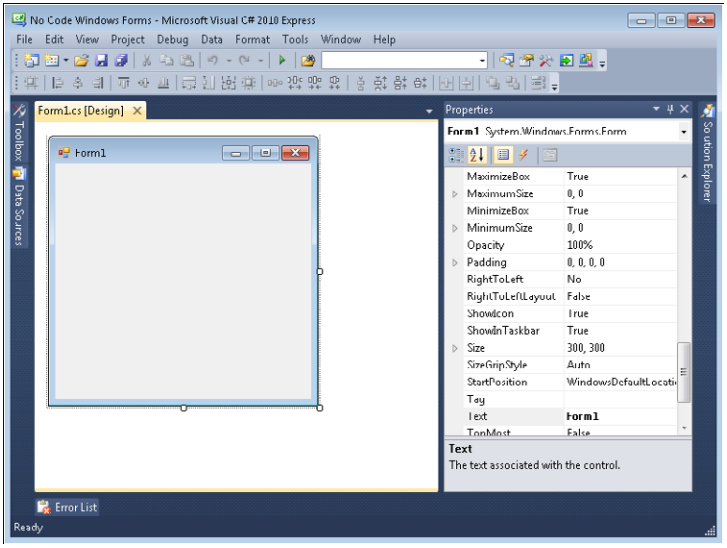


FIGURE 1-5 A Windows Forms Application begins with a designer that displays a blank form.

Note It's perfectly normal to see some small differences between your display and the screenshots in this book. Visual Studio is an incredibly flexible IDE and you can configure it to meet your specific needs. However, if you see large differences (for example, the screenshot doesn't look anything at all like the one in the book), you have probably made an error in following the procedure and will need to retrace your steps. Visual Studio is also incredibly forgiving—nothing bad is going to happen if you have to start over.

Quite a few windows are visible in the figure, but don't get overwhelmed. The book discusses them as needed. For now, all you really need to know is that the form designer appears on the left side of the display and the Properties window appears on the right. You use the designer to create the user interface for your application. The Properties window lets you configure the application elements as described in the "Configuring the Windows Forms Controls" section later in this chapter. You'll get familiar with what controls are and how to use them soon. If you don't currently see the Properties window in your IDE, choose View | Other Windows | Properties Window, or press Ctrl+W,P.

Note The content of the Properties window reflects the object you select. The contents will change when you select a form instead of a specific control. Each control will also display different content in the Properties window. Later, when you use Solution Explorer, you'll find that the Properties window content will change to reflect any entries you choose in Solution Explorer. If your Properties window content doesn't match the screenshot in the book, make sure you've selected the proper form, control, or Solution Explorer entry.

You may not think you can do too much with the application yet, but you can. It's possible to configure the form. Normally, you'll perform some form configuration before you even add any controls. Start by giving your form a better name. Highlight the *(Name)* field in the Properties window, and type **BrowserTest,** as shown in Figure 1-6. (Do not put a space between the words. BrowserTest needs to be all one word for it to work.)

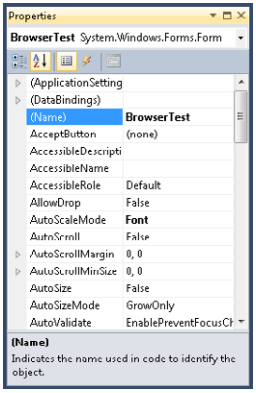


FIGURE 1-6 The Properties window tells you about form and controls settings in your application.

Notice that the Properties window displays a description of the property you've highlighted in a pane at the bottom of the window. If you don't see this pane, you can always display it by dragging the splitter bar that appears near the bottom of the window up to provide more space for the description. The *(Name)* property is a text property, meaning it's made up of characters (letters and/or numbers) so you simply type something to fill it. Other properties will have other ways to provide information, such as a list of acceptable values or even special dialog boxes that help you configure the property. You'll see these other kinds of properties in action as the book progresses.

Tip You can display the properties in two different ways to make them easier to find. The example in this section displays the properties in alphabetical order. You can also display the properties grouped into categories. To switch between views, click either Categorized or Alphabetical at the top of the Properties window.

It's important to give easily understood names to the controls and forms that make up your application so that they are easier to work with. A name can't start with a number, nor can it contain

any spaces. Many developers use an underscore (_) as a substitute for a space. For example, you could give your form the name **Browser_Test**. If you try to give your form an invalid name, the IDE displays an error dialog box informing you that the name is invalid, and returns the name to the previous (valid) name.

Scroll down to the *Text* property. This property determines the text that appears in the form's title bar. Type **Web Browser Test** for this property's value. Notice that the title bar text changes in the Designer after you press Enter.

Saving Your Project

It's a good idea to get into the habit of saving your project regularly. Saving the project reduces the likelihood that you'll lose information. Click Save All on the Standard toolbar, choose File | Save All, or press Ctrl+Shift+S. Save All saves all the files that have been modified; Save saves only the current file. You'll see the Save Project dialog box shown in Figure 1-7.

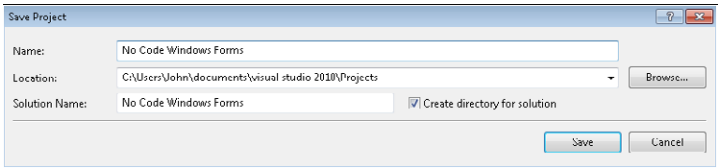


FIGURE 1-7 Save your project often to prevent loss of changes you make to it.

The Name field contains the name of this particular project. The IDE suggests a name based on the name you provided when you created the project. The Location field tells where the project is stored. Visual Studio defaults to using the C:\Users\<User Name>\documents\visual studio 2010\Projects folder on your hard drive, but you can store your projects anywhere. The Solution Name field contains the name of the *solution* that holds the project. A solution is a kind of container. You can store multiple projects in a single solution. For example, you might store an application as well as a program to test it in a single solution. A solution will often have a different name than the first project you create—but for now, keep the project and solution names the same.

Adding Windows Forms Controls

The IDE's border area displays some tabs, each of which corresponds to a particular window. Don't worry too much about them now, but one tab of immediate interest is the Toolbox. Clicking a tab displays its associated window. If you want the window visible without clicking it all the time, click Auto Hide (the pushpin icon in the upper-right corner of the window). Try it out now: click Auto Hide on the Properties window to hide it, and then click Auto Hide on the Toolbox to display it. Notice that the thumbtack icon changes to show whether a window will automatically hide. Your IDE will look something like the example shown in Figure 1-8.

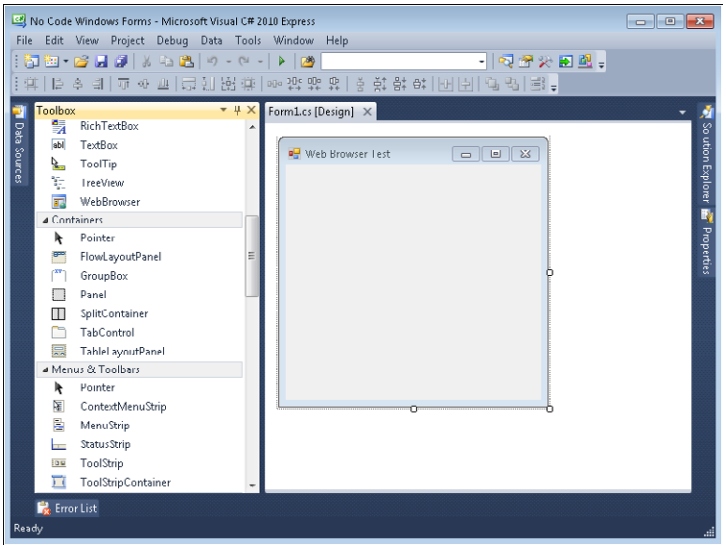


FIGURE 1-8 The Toolbox contains controls you use to create a user interface.

The Toolbox contains a wealth of *controls*. Controls are the building blocks of application development. You can snap them together in various ways to create a basic application design. Take some time to scroll through the list and explore the available controls now. As you can see, the Toolbox groups the controls into categories to make them easier to find. Otherwise, you'd spend your entire day looking for controls rather than creating incredibly useful applications. Most applications rely on the standard set of controls that you can find in the Common Controls category. One of these controls is the *WebBrowser* control used for this example.

Adding a control to your form is easy. You have three convenient ways to add the control:

- Drag the control from the Toolbox and drop it onto the form.
- Click the control within the Toolbox and then click where you want to place it on the form.
- Double-click the control within the Toolbox. This places it in a default position on the form.

Try one of these techniques now with the *WebBrowser* control. You'll see the control added to the form, as shown in Figure 1-9.

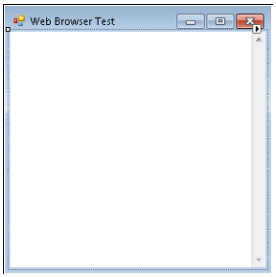


FIGURE 1-9 The *WebBrowser* control doesn't look like much when you first add it, but it contains information later.

As you can see, the control is invisible, but you can tell that the IDE added the control to the form because of the *sizing handles* (the little squares in each corner). In addition, in the upper-right corner you'll see an arrow that you can click to display a shortcut menu containing quick (and common) configuration settings. The control provides a vertical scroll bar that appears on the right side of the control in the figure. Your no-code application is ready for configuration.

Configuring the Windows Forms Controls

After you design the user interface for your application by selecting controls from the Toolbox, you'll normally hide the Toolbox window and display the Properties window again so that you can perform configuration tasks. Use the following steps to configure the *WebBrowser* control for this example.

Creating the No Code Windows Forms Application

1. Click the *WebBrowser* control in the form to select it.
2. Select the *(Name)* property and type **MyBrowser**.
3. Select the *ScriptErrrorsSuppressed* property and choose *True*. This is a *Boolean* property—it can only have one of the values *True* or *False*. Selecting *True* means that the *WebBrowser* control won't display scripting errors that occur when the control displays the URL you select.
4. Select the *Url* property and type **http://www.microsoft.com**. You could change this URL to any value you like. The *Url* property value you provide determines what resource the *WebBrowser* control displays when the application starts. At this point, the control is configured and ready for use.

Testing the Windows Forms Application

Believe it or not, you have a usable application at this point—and you haven't written a single line of code! It's true that the application doesn't do much—but it's a good place to start. To use the application, you need to tell the IDE to *compile* it. Compiling converts human-readable code into something that the computer can understand. The precise manner in which this works isn't important now, but you'll learn more about it as the book progresses. For now, simply choose Debug | Build Solution or press F6. In the lower-left corner of the IDE you'll see a message saying the build succeeded. (If you don't see the build succeeded message, it means that you made a mistake in following the previous sections and that you need to retrace your steps.) What this means is that the compiler was able to create executable code from the design you created and the executable is now ready to test.

To start the application, choose Debug | Start Debugging, or press F5, or click Start Debugging on the Standard toolbar. You'll see the application start. The browser window is going to be small at first, but you can resize it to see more of the page. Figure 1-10 shows some typical results from this application.

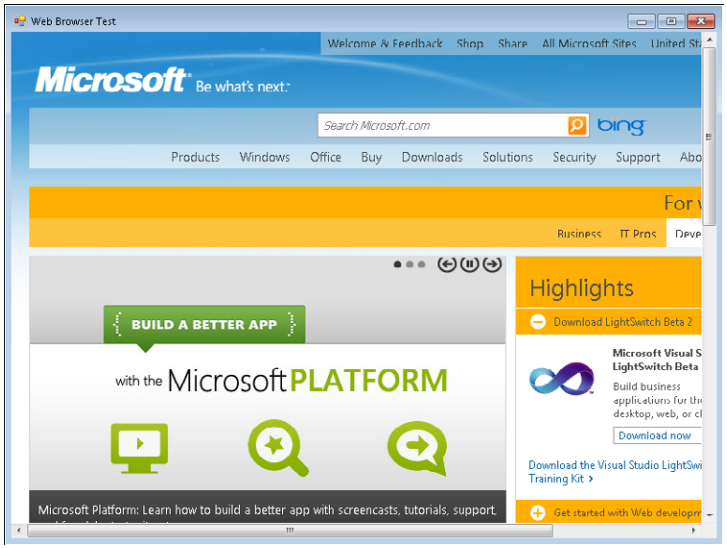


FIGURE 1-10 The example application displays a web page.

The application is fully functional. Click a link and you'll see the next page, just as you would in Internet Explorer. Right-click the application window and you'll see a shortcut menu containing all the usual browser controls. For example, you can move forward and backward through the history list, just as you would in Internet Explorer. Of course, it would be nice to have visible controls to perform these tasks, but you can worry about that later. For now, you've created your first usable application. To stop your application, click the Close box in the upper-right corner of the application window (the red X).

Viewing the Web Browser Code

Although you didn't write any code to make this application work, the IDE has been busy on your behalf. It generated code that matches all the design decisions you made. When you compiled the application earlier, you actually created an executable file based on the code that the IDE generated for you. Even though you won't normally edit this IDE-generated code, it's interesting to look at, because you can learn a great deal from it.

To see the Designer code, you must open a different IDE window. Hide the Properties window and display the Solution Explorer window shown in Figure 1-11.

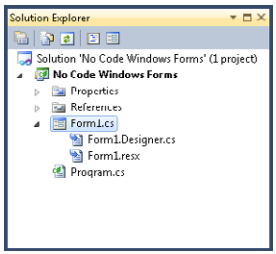


FIGURE 1-11 Solution Explorer provides you with access to the application files.

Solution Explorer presents a view of the files in your project. In this case, the figure shows the Form1 entry opened up to display the files associated with Form1—the form that contains the *WebBrowser* control. Notice the Form1.Designer.cs file. This is the file that contains the code used to create the form. Double-click this entry and you'll see the code you've created during the design process. Hide Solution Explorer so that you can see the code a little better. If you scroll down a bit, you'll see the entries that start to look familiar, like the ones shown in Figure 1-12.

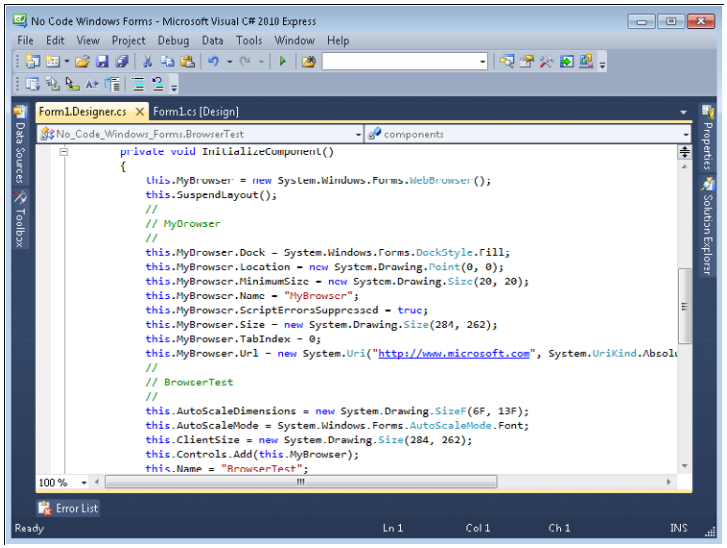


FIGURE 1-12 Even though you haven't written any code, the IDE has performed the task for you.

Note Make sure you open the correct file—you'll only see the information shown in this screenshot if you open Form1.Designer.cs. Also, you'll need to scroll down in the file to see the *InitializeComponent()* method. You may also need to click the plus sign (+) next to Windows Forms Designer generated code to expand the code so that it looks like the code shown here.

Here you can see the results of all of the changes you made. For example, you renamed the *WebBrowser* control as *MyBrowser* and you can see a number of *MyBrowser* code entries. Look a little closer and you'll see the property changes as well. For example, the line *MyBrowser.Name = "MyBrowser"*, simply states that you changed the name of the control to *MyBrowser* using the Properties window. The line of code literally says that the *MyBrowser* control's *Name* property is *"MyBrowser"*. Try browsing through the code to see more of what the IDE has done for you, but be careful not to change any of it.

Tip One of the ways that professional programmers learn new coding techniques is the very technique you just used—trying something out using a tool and then seeing what code the tool produced. You'll use this technique several times in the book because it's so incredibly useful.

Ending Your Session

When you're finished working with an example, it's a good idea to end your session. Choose File | Exit to close the IDE. Starting the IDE fresh for each example ensures that you're working with a clean environment and that there is less of a chance that errors will occur. Make sure that you end your session after each of the examples throughout the book. The book's procedures assume that you're starting with a fresh copy of the IDE each time, so the instructions might not work if you try to use the same session for all of the examples.

Creating the No-Code WPF Web Browser

Windows Presentation Foundation (WPF) is the latest technology for creating applications. In fact, the IDE you're using to create your applications relies on WPF. The site at *http://10rem.net/blog/2010/10/28/the-present-and-future-of-wpf* provides examples of additional real-world applications that rely on WPF. You'll find that WPF has many advantages over Windows Forms applications. Of course, it's hard to compare two technologies unless you perform the same task with each of them. The example in this section does just that. It shows how to create a Web browser application with the same capabilities as the one found in the "Creating the No-Code Web Browser" section, except that in this case, you'll use WPF instead.

Understanding the Benefits of WPF

Windows Forms applications will remain a faithful standby for many years because of the infrastructure in place to support it. However, the technology is getting old and isn't well-suited to today's user needs. Microsoft created WPF to make it easy to combine multiple presentation technologies in one package. When working with WPF, you can use these types of presentations:

- Forms
- Controls
- Complex text (such as found in a PDF)
- Images

- Video
- Audio
- 2D graphics
- 3D graphics

To obtain access to this wealth of presentation technologies, you'd normally need to combine several disparate application development techniques that might not even work well together. In short, you use WPF when you want to create an application that provides all of the experiences that modern users have come to expect. However, to obtain the extra functionality, you need additional skills. For example, even with the best tools, you can't create a 3D presentation without the appropriate skill set.

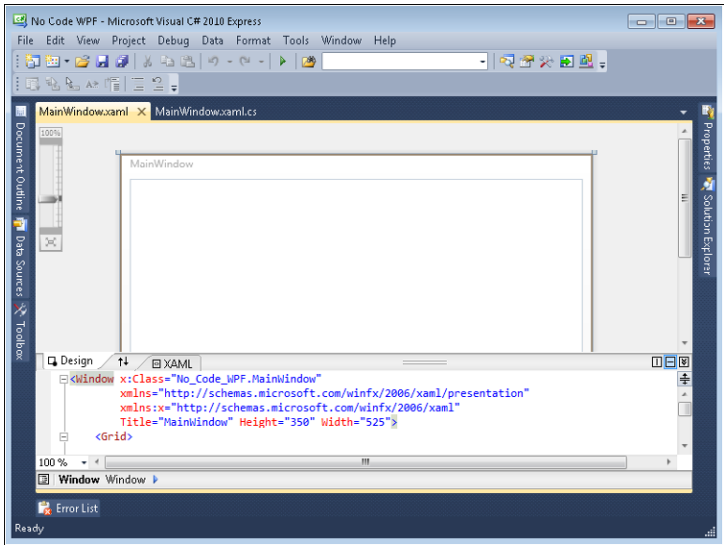
Using WPF has other benefits and this book will tell you about them as it progresses. However, one benefit stands out. WPF relies on a declarative language called Extensible Application Markup Language (XAML, pronounced *zammel*) to create the user interface. This language makes it possible to create an application with less code that is less reliant on precise connections with underlying application layers. As a consequence, you can often change the user interface without changing the underlying application layers—something that causes Windows Forms developers a lot of pain today.

Starting a New WPF Application Project

The example in this section creates a browser application precisely like the one in the section "Creating the No-Code Web Browser" except that this example relies on WPF. The following steps help you create the application project:

Creating the No-Code WPF Application

1. Start the Visual C# 2010 Express IDE if you haven't started it already.
2. Click New Project. The New Project dialog box appears.
3. Select the WPF Application template from the Visual C# folder.
4. Type **No Code WPF** in the Name field.
5. Click OK. The IDE creates the new project for you, as shown here.



You'll notice immediately that the WPF environment is completely different from the Windows Forms environment. For one thing, it looks a lot more complex. The environment really isn't that much more complex and you'll find that it provides a lot more flexibility. The top half of the Designer window shows a graphical interface similar to the one you used to create the Windows Forms example. The bottom half shows the XAML associated with the user interface you create—similar to the Form1.Designer.cs file described in the "Viewing the Web Browser Code" section of the chapter. The only difference is that the WPF environment shows you this information from the outset so that you can create the user interface graphically or by writing XAML code to do it.

Fortunately, you don't have to look at the XAML if you don't want to. Click Collapse Pane in the Designer window and the XAML pane simply disappears, as shown in Figure 1-13.

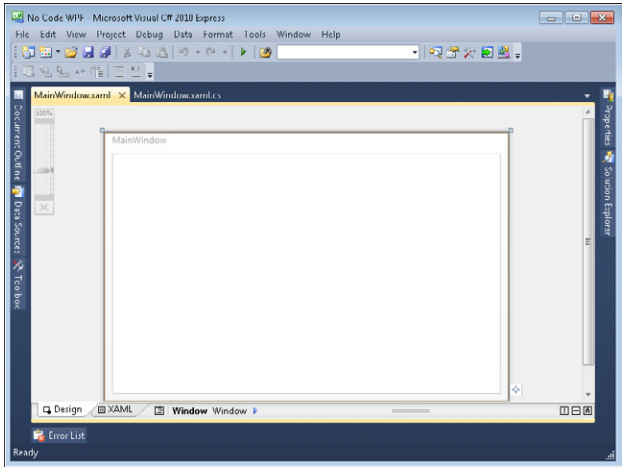


FIGURE 1-13 The WPF designer lets you hide the XAML tags from view.

If you decide later that you really do want to see the graphical environment and the XAML side-by-side, you can click Vertical Split or Horizontal Split in the Designer window. It's also possible to see the XAML by clicking the XAML tab. In this case, you see a full page of XAML instead of just seeing part of the code in a pane. So, there really isn't anything scary about this environment after all.

Before you do anything else, you'll want to give your application better title bar text so that it identifies the purpose of the application. Display the Properties window, select the *Title* property, and type **No Code WPF**. You can hide the Properties window again.

Adding WPF Controls

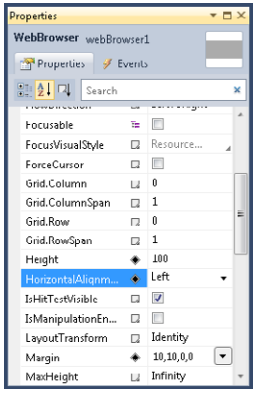
As with any application you develop, WPF applications rely on the Toolbox as a source of controls. To add controls to this example, you need to display the Toolbox by clicking its tab and then clicking the Auto Hide button on the Toolbox window. You can add the *WebBrowser* control (the only control used in this example) using any of the three techniques described in the "Adding Windows Forms Controls" section of the chapter.

Configuring the WPF Controls

When you add the *WebBrowser* control to your WPF application, you'll notice that it appears in the upper-right corner of the *MainWindow*. A WPF application relies on windows, not on forms as a Windows Forms application does. Because of this difference, configuring the *WebBrowser* control is a bit different from configuring it for a Windows Forms application. The following steps tell you how to perform this task:

Modifying the WPF Application Controls

1. Hide the Toolbox and display the Properties window. One thing you'll notice immediately is that the WPF properties window doesn't provide any helpful information about the property you select, as shown here.



This difference means you must know a bit more about the properties you're using when working with WPF. Fortunately, Microsoft provides detailed help for the controls and you can always refer to Help by pressing F1.

Tip If you find that you've set a property incorrectly, you can always return it to its default value by right-clicking the property and choosing Reset Value. This feature makes it possible to experiment safely with your application settings.

2. Type **Auto** in the *Height* property. This value ensures that the control automatically adjusts to its container size in the y axis.
3. Change the *HorizontalAlignment* property value to *Stretch*. This change lets the *WebBrowser* control extend the length of the window, no matter what size the window is.
4. Type **http://www.microsoft.com** in the *Source* property. This change sets the starting URL for the *WebBrowser* control.
5. Change the *VerticalAlignment* property value to *Stretch*. This change lets the *WebBrowser* control extend the height of the window no matter what size the window is.
6. Type **Auto** in the *Width* property. This value ensures that the control automatically adjusts to its container size in the x axis. At this point, the control is configured for use.

Trying the WPF Application

It's time to try the WPF application. Like the Windows Forms application, you must compile the WPF application by choosing Debug | Build Solution or by pressing F6. You'll see a Build Succeeded message in the lower-left corner of the IDE, as before. To start the application, choose Debug | Start Debugging, press F5, or click Start Debugging on the Standard toolbar. You'll see an application that looks similar to the Windows Forms application, as shown in Figure 1-14.

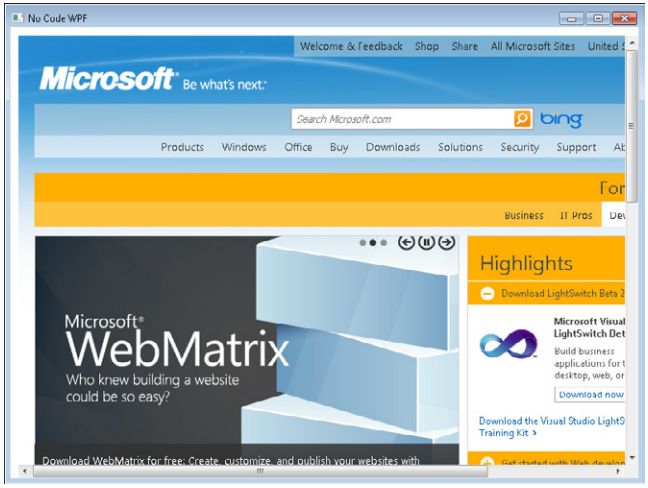


FIGURE 1-14 The WPF application produces about the same output as the Windows Forms application.

The two applications aren't precisely the same in appearance, but they're very close. They do work precisely the same way. Click any link in the window and you'll go to that page. You can access all of the browser controls by right-clicking the window and choosing an option from the shortcut menu. In short, you've created a WPF version of the Windows Forms application you created earlier—all without any coding! When you're done with the application, click the Close box as usual.

Viewing the WPF Code

As with the Windows Forms example, every design decision you make when working with WPF creates code. The IDE creates this code for you in the background. You can see this code by clicking the XAML tab in the IDE. Remember that XAML is actually a form of XML, so it looks like code that you may have seen in other situations. Figure 1-15 shows what the XAML looks like for this example. (I've reformatted it for the book—the code you'll see will appear on a single line, but it's the same code.)

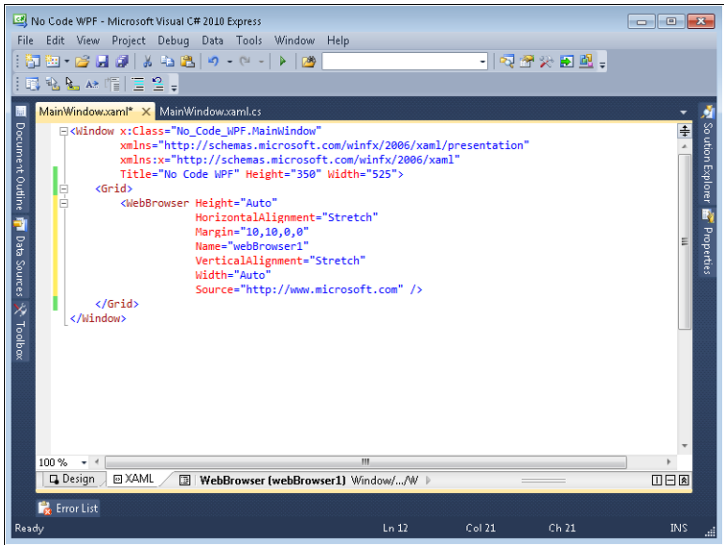


FIGURE 1-15 The XAML code for the example application is simpler than the Windows Forms alternative.

If anything, this code is a little clearer than the Windows Forms example code. All of the changes you made appear as part of the <*WebBrowser*> tag. Each attribute/value pair describes a single change.

You might wonder why this example didn't change the name of the form and the control as the Windows Forms example did. It turns out that these properties don't appear in the Properties window. If you want to make this particular change, you need to work with the XAML directly. For example, if you want to change the name of the *WebBrowser* control, you'd type **Name="MyBrowser"**.

Creating the No Code WPF Browser Application

Both of the applications presented so far in the chapter have one thing in common—they create a separate application that appears like any other application on your hard drive. The application starts just like any other application you've seen before. The WPF Browser Application example in this section is different. It starts up in your browser. That's right—this is a special kind of application that appears in your browser, even though you aren't accessing it from the Internet. The benefit of this kind of application is that it lets you start the user on the local hard drive and move onto the Internet or a local server without any change in appearance. The user only knows that the application appears in a browser, not where the application or its associated data resides.

Understanding the Benefits of a Mixed Application

Don't get the idea that Windows Forms and WPF are mutually exclusive—that you must choose between one technology and the other. In fact, Microsoft has purposely made it possible for each technology to host the other. It's possible to create an application that mixes the two together, so that you can get the best of each. You could potentially update an existing application with WPF elements to give users the kind of experience they demand without reworking the entire application.

The best way to use this potential is to build application programming skills a little at a time. You can start with Windows Forms applications and add WPF elements gradually until you know both technologies well. The mixed environment also makes it possible to gradually move users to the new environment so that they require less training time.

Setting Internet Explorer as the Default

Before you can use this application type successfully, you need to set Internet Explorer as your default browser. Follow these instructions to ensure that you have the correct setup:

Configuring Internet Explorer as the Default Browser

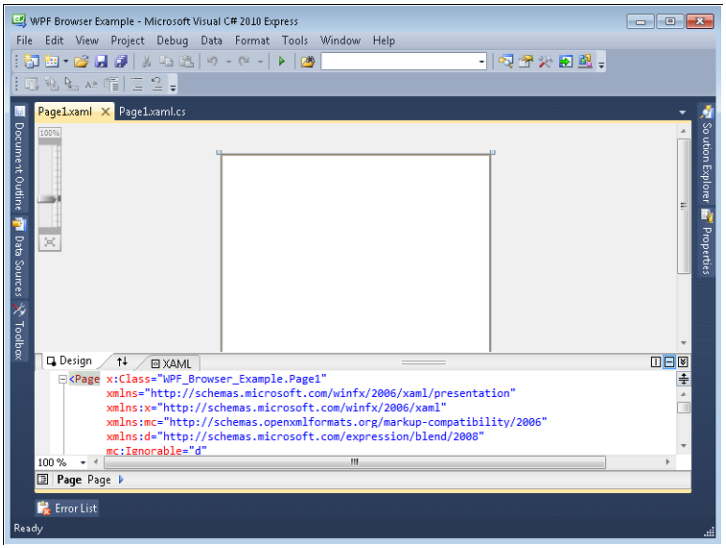
1. Choose Start | Control Panel. The Control Panel opens.
2. Click Network And Internet. The Network and Internet options appear.
3. Click Internet Options. The Internet Properties dialog box appears.
4. Click the Programs tab. This tab contains a number of options, including the default browser.
5. Click Make Default. Internet Explorer becomes the default browser (if it isn't the default already).

Starting a WPF Browser Application Project

Now that you have Internet Explorer configured, it's time to create the WPF project. The following stps show how to create a basic WPF project that won't require any coding.

Creating the WPF Browser Application

1. Start the Visual C# 2010 Express IDE if you haven't started it already.
2. Click New Project. The New Project dialog box appears.
3. Select the WPF Browser Application template from the Visual C# folder.
4. Type **WPF Browser Example** in the Name field.
5. Click OK. The IDE creates the new project for you, as shown here.



As you can see, this is another WPF application. However, notice that this application doesn't have a MainWindow—instead it has a page. That's because the application is hosted in Internet Explorer and isn't created as a-standalone application.

Adding WPF Browser Controls

This example doesn't rely on the *WebBrowser* control used for the other two examples in the chapter. If you try to use the *WebBrowser* control in your WPF Browser application, the application will likely crash. That's because you're attempting to host a copy of Internet Explorer within itself (at least, that seems to be the theory). So this example relies on a different control for demonstration purposes. Begin by displaying the Toolbox by clicking its tab and then clicking the Auto Hide button on the

Toolbox window. Add the *Image* control (the only control used in this example) using any of the three techniques described in the "Adding Windows Forms Controls" section of the chapter.

Configuring the WPF Browser Controls

When you add the *Image* control to your WPF application, you'll notice that it appears in the upper-right corner of the *Page1*. Working with an *Image* control is similar to working with the *WebBrowser*, but there are some differences. The following steps tell you how to configure the *Image* control for use:

Modifying the WPF Browser Application Controls

1. Hide the Toolbox and display the Properties window.
2. Set the *Height* property to *Auto*.
3. Change the *HorizontalAlignment* property value to *Stretch*.
4. Type **http://apod.nasa.gov/apod/image/1104/m74_baixauli_900.jpg** in the *Source* property. This change sets the picture that the *Image* control displays. If you have some other favorite picture you'd like to see, you can provide its location as a source instead.

Tip If you set the *Source* property successfully, you'll see the picture appear immediately in the IDE, unlike the *WebBrowser* control where you must try the application out to see whether the *Source* property is correct. A number of controls provide instant feedback, which makes them easier to use.

5. Change the *VerticalAlignment* property value to *Stretch*.
6. Set the *Width* property to *Auto*.

Trying the WPF Browser Application

The IDE does provide certain shortcuts when working with applications. Normally, you want to compile your application first to determine whether there are any errors, and then run it. However, this time try something different. Choose Debug | Start Debugging, press F5, or click Start Debugging on the Standard toolbar to start the application without first compiling it. What you'll see is that the IDE automatically performs three tasks:

1. Saves your project to disk.
2. Compiles the application for you and displays the success message in the lower-left corner of the IDE (you need to look quickly).
3. Starts the application for you.

Even though the IDE will perform these tasks for you, it's still better to do them yourself. It's a good idea to get into the habit of saving your project often and looking for errors when you compile it. Still, it's nice to know that the IDE performs these steps for you when you forget. Figure 1-16 shows what the example application looks like.

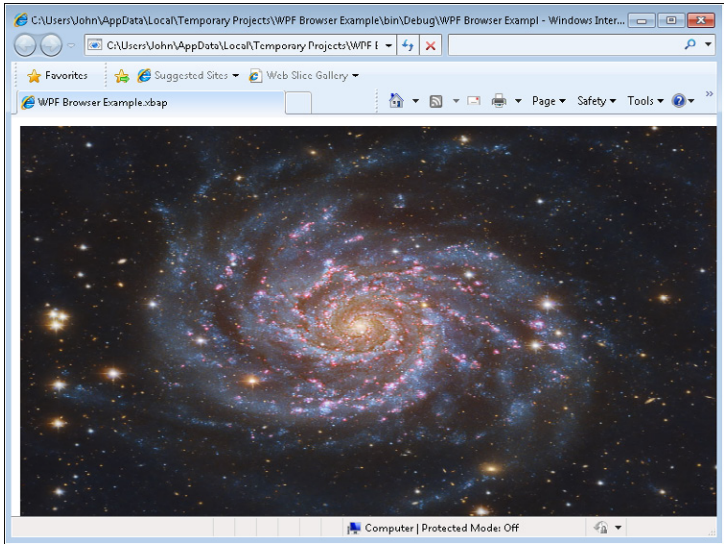


FIGURE 1-16 The WPF Browser Application displays within a browser, rather than as a desktop application.

The example shows a stunning picture of the universe (M74, a spiral galaxy). As you can see, the page exists in Internet Explorer and it could just as easily be an application that relies on both local and remote resources. Closing Internet Explorer stops the application and returns the IDE to development mode.

Viewing the WPF Browser Code

As with the previous WPF example, you click the XAML tab to see the code produced for you by the IDE. Instead of a *WebBrowser* control, you'll see the code for an *Image* control this time. Figure 1-17 shows the code you'll see (with the code reformatted for presentation in the book—your code will appear on a single line).

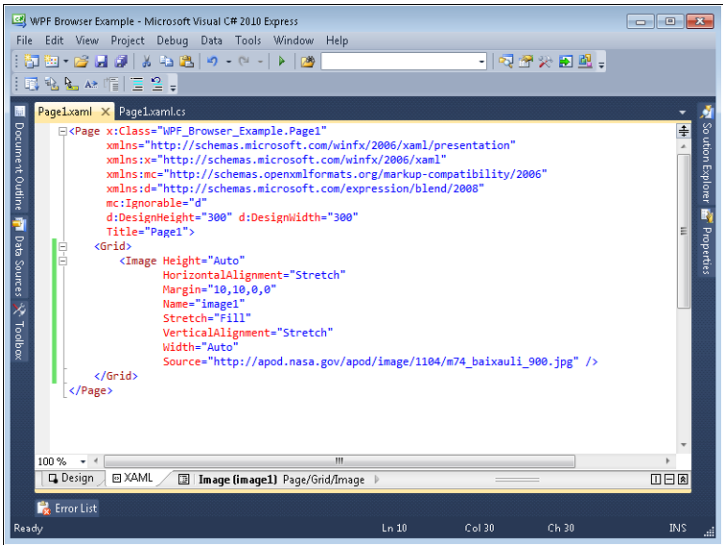


FIGURE 1-17 The XAML for this application shows the use of the Image control to display content.

Get Going with C#

This chapter gets you started with Visual C#. You install products that permit both desktop and web development. In addition, you create three desktop application examples that require no coding on your part. Of course, you now know that all three examples do have code in them and that the IDE creates this code for you. The biggest lesson you can learn from these examples is to let the IDE help you create your applications whenever possible. Using IDE features to speed development efforts means that you spend less time coding and more time enjoying some time out on the town.

You discovered some new techniques for creating an application in this chapter. Although most applications do require that you add code to make them functional, you can play around with many of the controls and develop an application that's at least partially functional. Take some time now to play around with some of the more interesting controls to see what they do. Of course, we'll cover many controls as the book progresses, but it's important to realize that working with applications can be fun and that play time (time spent seeing what happens when you do something) is a big part of application development—at least it is for the best developers.

Chapter 2, "Developing a Web Project," adds to the information you've already learned in this chapter. However, instead of working with desktop applications, you'll work with web applications. In this chapter, you opened the Visual C# 2010 Express IDE and learned some basics about it; Chapter 2 goes through the same process for Visual Web Developer 2010 Express. By the time you finish Chapter 2, you'll have created some additional no-code web examples and will understand how they differ from desktop applications.

Chapter 2

Developing a Web Project

After completing this chapter, you'll be able to:

- Start Visual Web Developer 2010 Express so you can build web applications with it
- Create a standard project without writing any code
- Create a standard website without writing any code

DESKTOP APPLICATIONS ARE STILL THE primary way that businesses interact with data—but a vast array of other options are available. One increasingly common choice relies on the Internet (or an intranet) to host various kinds of applications. This book won't show you every kind of application you can create in Visual Studio, but it does provide an overview of how to build the more popular types.

Most applications begin with the need to access some type of data from a client application. The client-server paradigm has been around for many years in a number of forms. These Internet applications are just another form.

For more information, see "client-server" in the accompanying Start Here! Fundamentals of Microsoft .NET Programming *book. To obtain your copy, see the section titled "Free Companion eBook" in the introduction to this book, or turn to the instruction page at the back of the book.*

This chapter begins by exploring the tool you use to create web applications of various types: Visual Web Developer 2010 Express. The applications you will focus on first are intended for the client. Knowing how to create a user interface for any sort of data is helpful, even data hosted by someone else. In fact, with the incredible stores of data available online, it's a wonder that people still find something new to store—but they do. Visual Web Developer 2010 Express can help you create most of the client application types that the .NET Framework supports.

After you get to know Visual Web Developer 2010 Express a little better, you'll begin working with some actual applications, creating a simple project, and using it to define a simple web application.

The second project shows you how to create a simple website and access it using a browser. These two application types go a long way toward getting you started programming the Internet, but of course, they're just the beginning. Other chapters in this book explore web applications in considerably more detail.

Note This chapter assumes that you've installed Visual Web Developer 2010 Express on your system. If you haven't performed this task, look at the instructions found in the "Obtaining and Installing Visual Studio 2010 Express" section of Chapter 1. This section shows how to install both Visual C# 2010 Express and Visual Web Developer 2010 Express. It also contains instructions for updating your installation to use Service Pack 1 (SP1), which contains important fixes that affect the examples in this book.

Starting Visual Web Developer 2010 Express

After you have Visual Web Developer 2010 Express installed on your system, follow these steps to start the Integrated Development Environment (IDE) (which is different from the Visual C# 2010 Express product used in Chapter 1): choose Start | All Programs | Microsoft Visual Studio 2010 Express | Microsoft Visual Web Developer 2010 Express. You'll see the IDE start up, as shown in Figure 2-1.

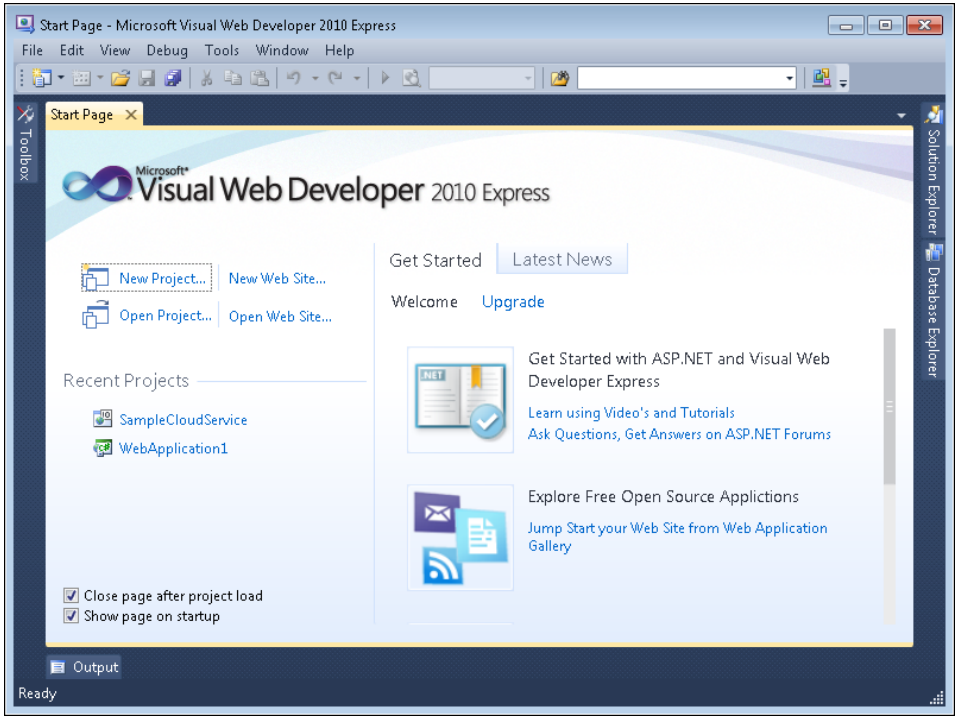


FIGURE 2-1 The Visual Web Developer IDE opens with the Start Page.

The IDE begins by displaying the Start Page. You can turn this feature off by clearing the Show Page On Startup option in the lower-left corner. If you later decide you want to see the Start Page, choose View | Start Page and select the Show Page On Startup option again. The Close Page After Project Load option works for both projects and websites. It frees up screen real estate by closing the Start Page when it's no longer needed after you create or open a project or website.

The left side of the Start Page also contains links for creating or opening a project or website. The "Understanding the Difference Between Websites and Projects" section of this chapter describes the differences between a project and website, so don't worry about it for now.

Anything you've worked on recently (both projects and websites) appears in the Recent Projects list. Click the entry for the project or website you want to open. If you're using Windows 7, remember that you also have access to the Jump Lists feature by right-clicking the Microsoft Visual Web Developer 2010 Express entry in the Start menu, and choosing the project or website you want to open.

On the right side of the display, the Get Started tab contains a number of interesting entries. These entries are all devoted to helping you become more productive with Visual Web Developer 2010 Express quickly. They're also different from the Visual C# 2010 Express offerings. Here are the four Get Started topics and why you should look at them:

- **Get Started with ASP.NET and Visual Web Developer Express** This option doesn't display help information—you get help by pressing F1. Instead, the first link for this entry provides access to videos and tutorials you can use to learn more about Visual Web Developer. The second link provides access to the Active Server Page (ASP).NET forums where you can ask questions of other developers and various experts that roam the forums.
- **Explore Free Open Source Applications** Click the link for this option to see open source applications at *http://www.microsoft.com/web/gallery/*. When you get to the site, you'll see a number of free applications. You can select an application and click Install to download and automatically install the application to your hard drive so that you can use it. For example, you'll find a number of interesting Content Management Systems (CMSs), such as Joomla and DotNetNuke. It pays to spend some time browsing this site even if you don't end up downloading anything, because looking at the range of available applications can provide useful ideas for your own applications.
- **Find Affordable Web Hosting** Click this link to find a number of affordable web hosting companies at *http://www.microsoft.com/web/hosting/home*. Each company offers different features at different rates, so you're likely to find a solution that meets your needs.

Note You don't need a web hosting company for development. You need one only when you're planning to publish your applications online—usually for public consumption.

- **Get More Software at No Cost** This section contains a number of links for free software. For example, if you click the Microsoft DreamSpark for Students link, you'll go to *http://www.microsoft.com/web/hosting/home,* where you can find out more about this product. DreamSpark is more than a single application; the site actually provides access to a number of applications, including Visual Studio 2010 Professional and Microsoft Certification exams.

The Latest News tab provides information in Really Simple Syndication (RSS) form about Visual Web Developer updates and changes. To use this feature, click the Enable RSS Feed option. However, you should know that obtaining the latest information in the IDE can slow things down at times. A better option is to add the site's RSS feed to Outlook. To do that, first make sure Outlook is running. Copy the Uniform Resource Locator (URL) from the RSS Feed field and paste it into your browser's address field. Press Enter, and after a few seconds your browser will ask if you want to add the RSS feed to Outlook.

Note The link provided for Visual Web Developer 2010 Express is different from the one for Visual C# 2010 Express, so you'll want to add them both to Outlook.

Creating the No-Code Project

Web development is substantially different from desktop development. For one thing, when creating a web application you're always interacting with a web server, even if that server is installed on your own system. A desktop application has no such intermediary—the operating system executes the application directly on the local system. In addition, web applications normally rely on a browser to host them on the client computer. You'll encounter a number of these differences as the book progresses, but this chapter will introduce you to a few of the desktop/web application differences.

Note Visual Web Developer 2010 Express supports multiple languages—Visual Basic .NET and Visual C#—and a wealth of project types. This book won't discuss the Visual Basic .NET features of Visual Web Developer—you can find those features discussed in *Start Here! Programming in Visual Basic .NET*—however, you'll explore all the C# project types as you progress through this book.

The example in this section is a simple project. You'll create an ASP.NET application with a basic interface. As with the desktop applications presented in Chapter 1, you'll let the IDE create the required source code for you.

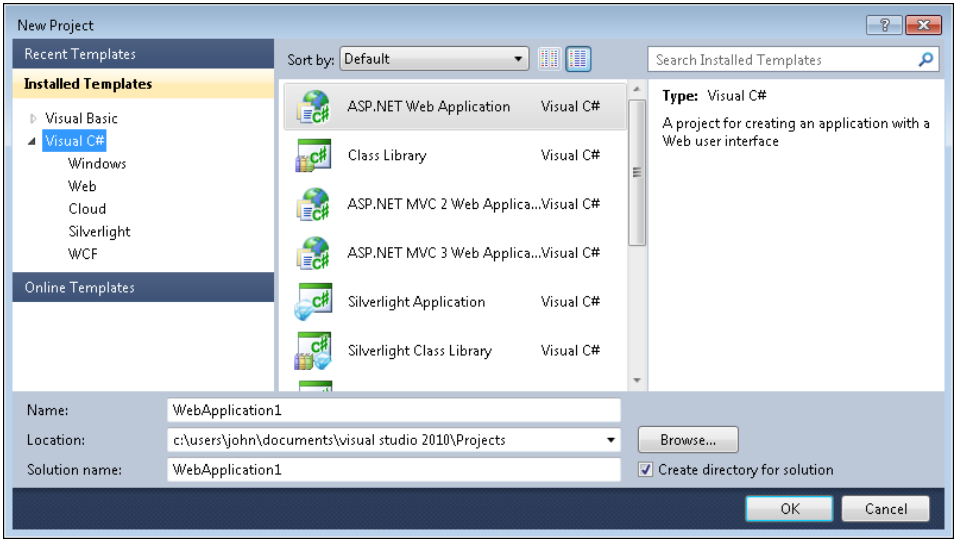
Starting the New Project

This section of the chapter shows how to build a project. This process is typical for every kind of project, even if you're using a different template than the one discussed in this section. Of course, each template produces a different kind of application, so what you see after you complete the process will differ depending on which template you're using. Carefully follow these steps to get started.

Create a New Web Project

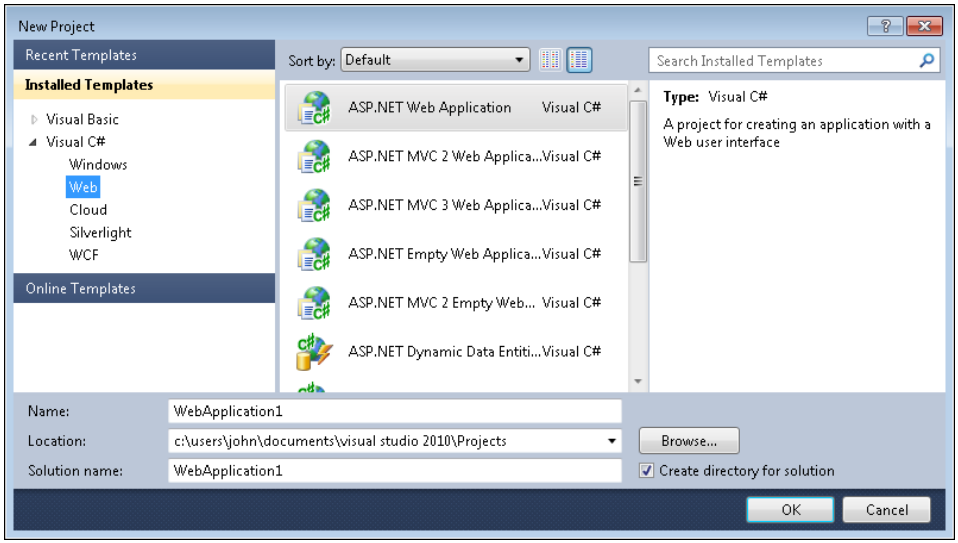
1. Choose Start | All Programs | Microsoft Visual Studio 2010 Express | Microsoft Visual Web Developer 2010 Express. You'll see the IDE start up.

2. Click New Project. You'll see the New Project dialog box shown here.



   Notice that Visual Web Developer 2010 Express supports both Visual Basic .NET and Visual C#. Make sure you always select the Visual C# folder to work with the C# templates. Otherwise, you'll create a Visual Basic .NET application.

3. Highlight the Visual C# folder. You'll see a number of subfolders that help you locate application templates by type. For example, if you click the web folder, you'll see only those templates associated with web projects.



Choosing a specific type can save time when you know the type of application you want to create. The center pane of the New Project dialog box contains the list of templates within a particular folder. The right pane describes the template you select. Notice that the left pane confirms that you've selected a Visual C# template.

The New Project dialog box also contains controls to change the appearance of the center pane. You can choose small or larger icons. In addition, you can sort the templates in a specific order.

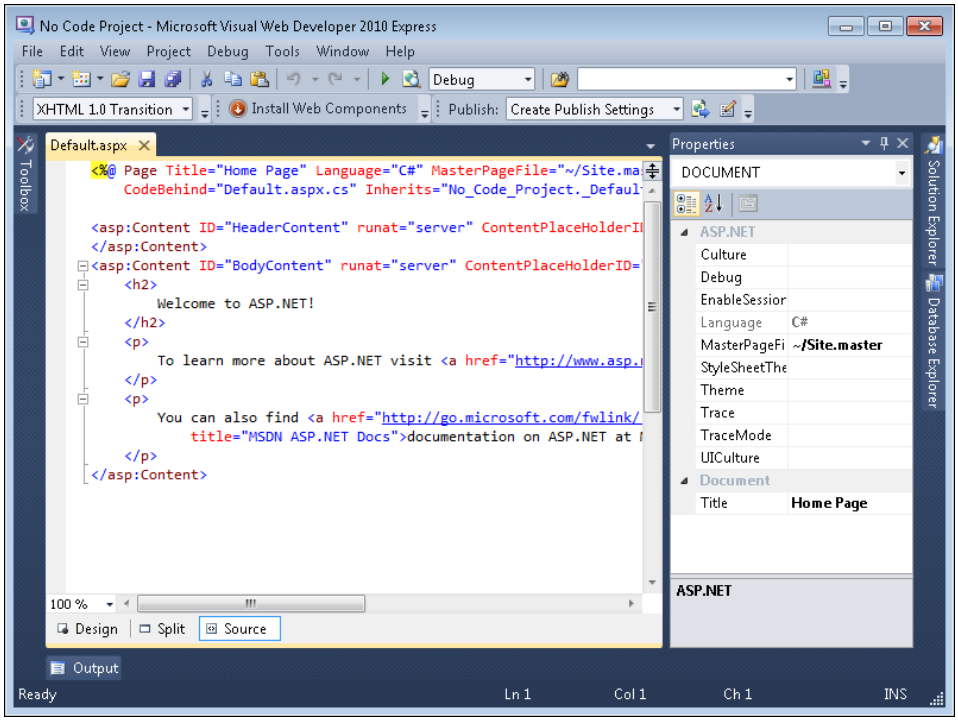
4. Select a project type. The example application uses the ASP.NET Web Application template.

5. Type the name **No Code Project** in the Name field. Notice that the Solution Name field automatically changes to reflect the name you just typed in the Name field. The Solution Name field can contain a different value. A solution is a kind of container. You can store multiple projects in a single solution. For example, you might store an application and its test program in a single solution. Thus, the Solution Name field can be different from the project name because it reflects the name for a multi-project solution.

6. Choose a location where you want to store the project files. (Click Browse to display the Project Location dialog box to choose the folder you want to use.) The default location is c:\users\<User Name>\documents\visual studio 2010\Projects; however, you can choose any location on your hard drive to store the project. Unlike the desktop applications created in Chapter 1, the simple act of creating a project stores files on disk, which is why you must choose a storage location in the New Project dialog box.

7. Select the Create Directory For Solution option if you want the solution file to appear in its own folder. This feature is useful primarily when you're creating a multiple-project solution, because each project will appear in its own subfolder. However, keeping the option selected for a single project solution doesn't cause any problems, so normally you keep this option selected.

8. Click OK. The IDE will create the new project for you based on the template you select. Some templates provide default content; others are completely blank. The template used for the example project provides the default content shown here.



The default display takes you to the code immediately, which isn't what you want in this case. You can click Design to see the graphical interface or click Split to see a combination of the graphical interface and code. Click Design and you'll see the graphical view of the default site, as shown in Figure 2-2.

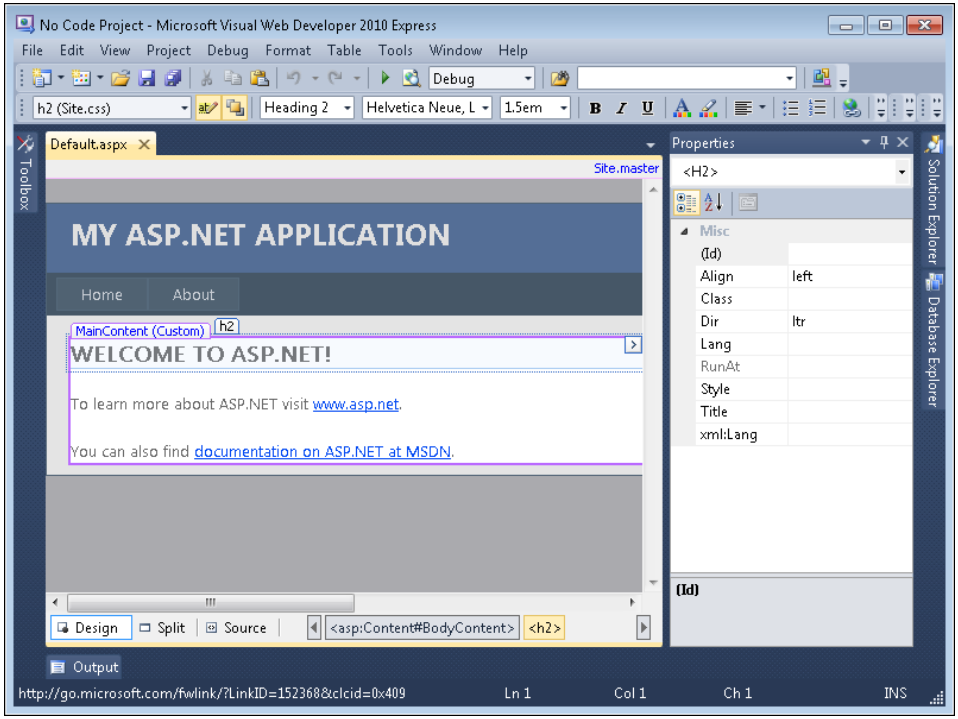


FIGURE 2-2 The sample application includes a number of interesting elements.

That's quite a bit of content. The "Understanding the Default Site" section explains all this content in a little more detail.

Understanding the Default Site

The default site that the ASP.NET Web Application template creates contains a number of individual elements. Each element contributes toward the whole site. In many cases, you'll want to keep all these elements as a starting point for your project. But because they can prove confusing, this section explains the most important elements—the ones you need to know about now to create a program without coding anything. Later, this book describes more of the template elements so you can begin coding your website.

Looking at the Elements

Before going any further, it's important to understand how these default site elements appear in the IDE. If you can see the Properties window, click the Auto Hide button in the upper-right corner. Click Solution Explorer, and then click the Auto Hide button so the window remains fixed in position. You'll see a list of the default site elements like the one shown in Figure 2-3.

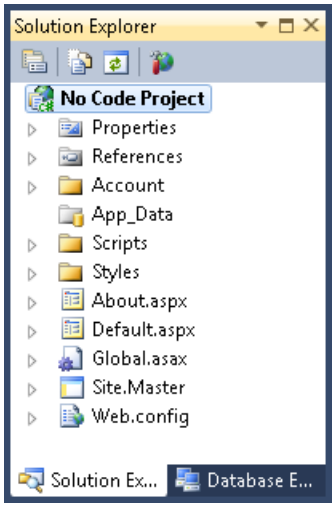


FIGURE 2-3 Solution Explorer makes it possible to see all of the files for your application.

Solution Explorer provides access to all the files that make up the default site, even those you won't use for this example. The entries you need to know about for this project are:

- **Site.Master** Provides a template that gives the entire site the same look and feel. This file is the master page—a page that controls all the other pages. Using a master page makes it possible to create complex sites with far less code. The master page contains the overall site design, so you need to make changes to the master page only when you want to change your entire site to have a different look and feel.
- **Site.css** Describes the formatting used for the entire site. For example, if you want all headings to use a bold font, you'd place that information in this file.
- **Default.aspx** Contains the content for the first page that anyone who visits your site sees when they enter your site using just the domain URL. (As with any other site, someone can enter a page-specific URL to access another content page directly.) This default page normally contains an overview of your site as well as links to other information on your site.
- **About.aspx** Holds information about your site, the application, or your organization. The default site provides this simply as a placeholder page; you won't find any actual content on this page.

The default site contains a number of features that you may not require at all. For example, the master page contains a link to a login page that users can use to log on to your site. Unless you need this security feature, you probably won't keep it in place. However, for now you won't need to worry about whether these features are in place. The example in this section doesn't use them, and you don't need to worry about them.

Working with the Master Page

The master page, Site.Master, contains the overall design for your site. When you open a content page that uses the master page, you see an entry for it in the upper-right corner of the page in Design view.

Note The master page file may not *always* be named Site.Master, but it is when you're working with the default site.

Begin by looking at the Def ault.aspx file that you see when Visual Web Developer 2010 Express first opens the project for you. If you place the cursor in any location controlled by the master page, you'll see a red circle with a line through it, as shown in Figure 2-4.

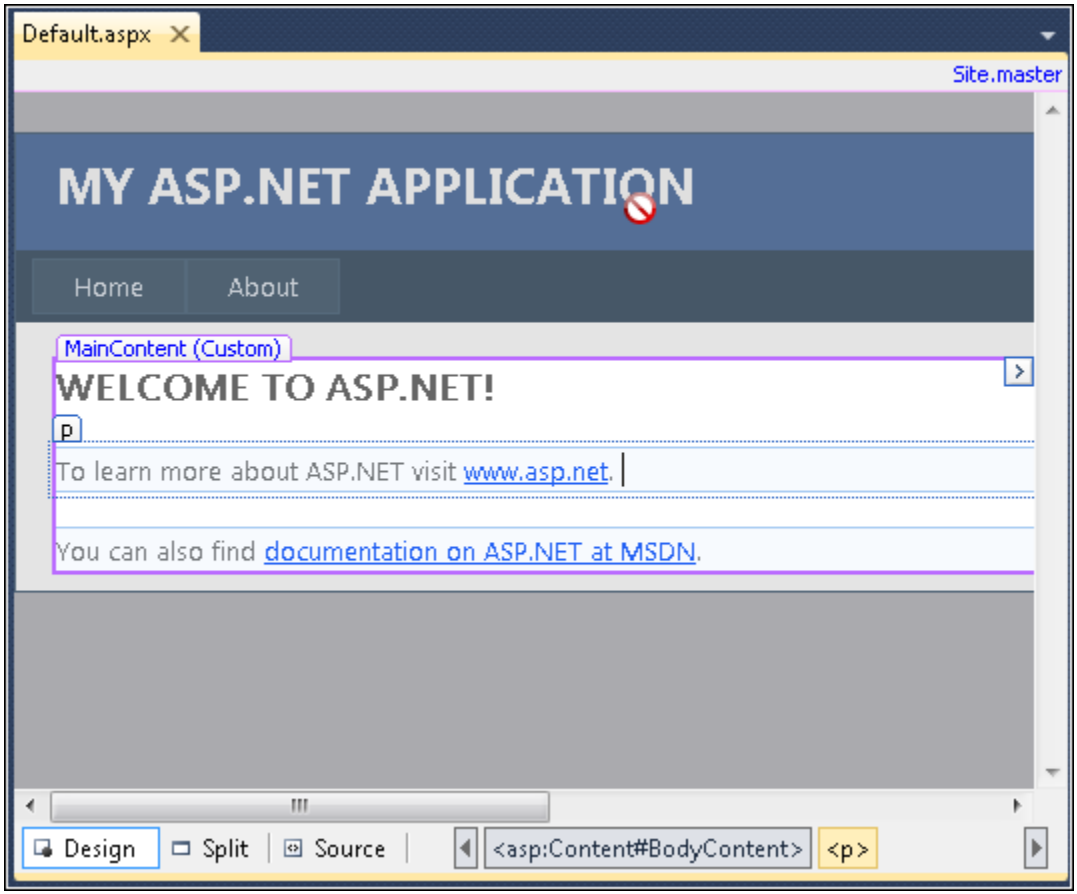


FIGURE 2-4 The master page contains all of the elements that are common to all pages on a website.

To change the site name, open the master page by clicking the Site.Master link in the upper-right corner. Figure 2-5 shows what you see when you click this link and choose the Design tab.

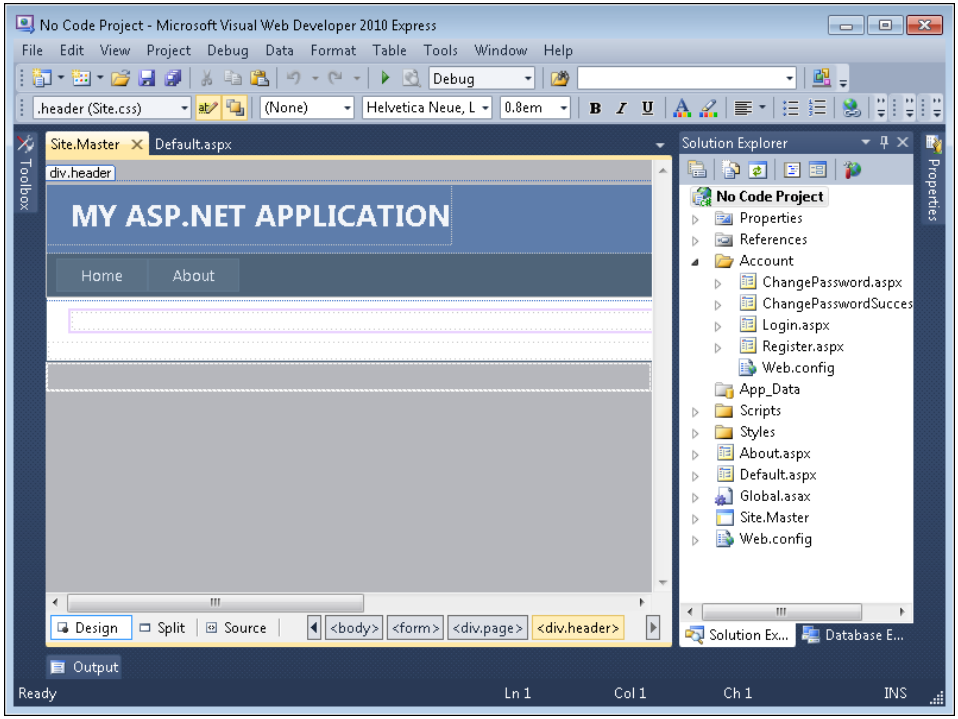
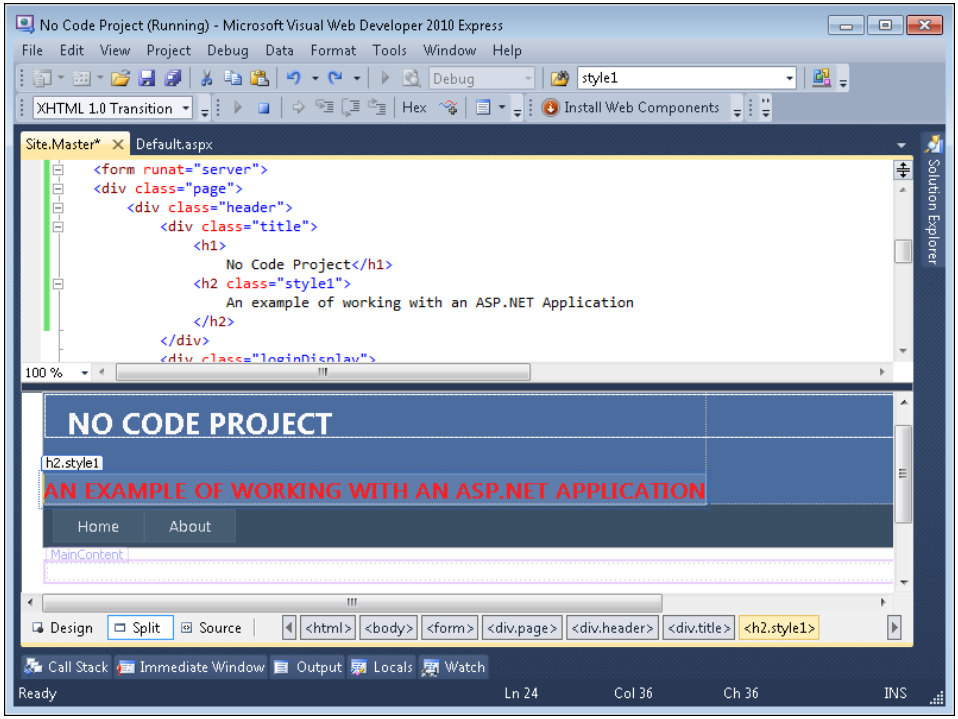


FIGURE 2-5 In order to change master page content, you must open the Site.Master file.

All the elements that were previously inaccessible are now ready to edit. Making a change here affects every page that uses this master page. Now that you can access the master page, you can make changes to it.

Edit the Master Page

1. Type **No Code Project** for the heading.
2. Press Enter to create another line.
3. Change the Block Format to Heading 2 and type **An Example of Working with an ASP.NET Application.** Notice that the color of the text is unreadable against the background.
4. Highlight the entire line, click Foreground Color, and choose Red as the new color.
5. Scroll to the right side of the page. Highlight and delete the login entries because this example doesn't use them. At this point, your Site.Master file should look like the one shown on the next page.



This shows the Split view of the file. As you can see at the top, the code reflects the changes made in the various steps. Notice that changing the color of the second heading creates a new style entry. This change appears only in the Site.Master file, not in the Site.css file used to control the styles for the entire site.

6. Save and close the Site.Master file.

Changing the Default.aspx Content

The Default.aspx file contains content. The master page controls the overall layout of the page and the Style.css file controls the appearance of the page. So when you work with this page, you'll typically want to focus on the actual content, using the other two resources only when you want to change the layout or appearance of all the pages on your site.

This part of the example displays a custom heading and an image as content. Use these steps to make the changes.

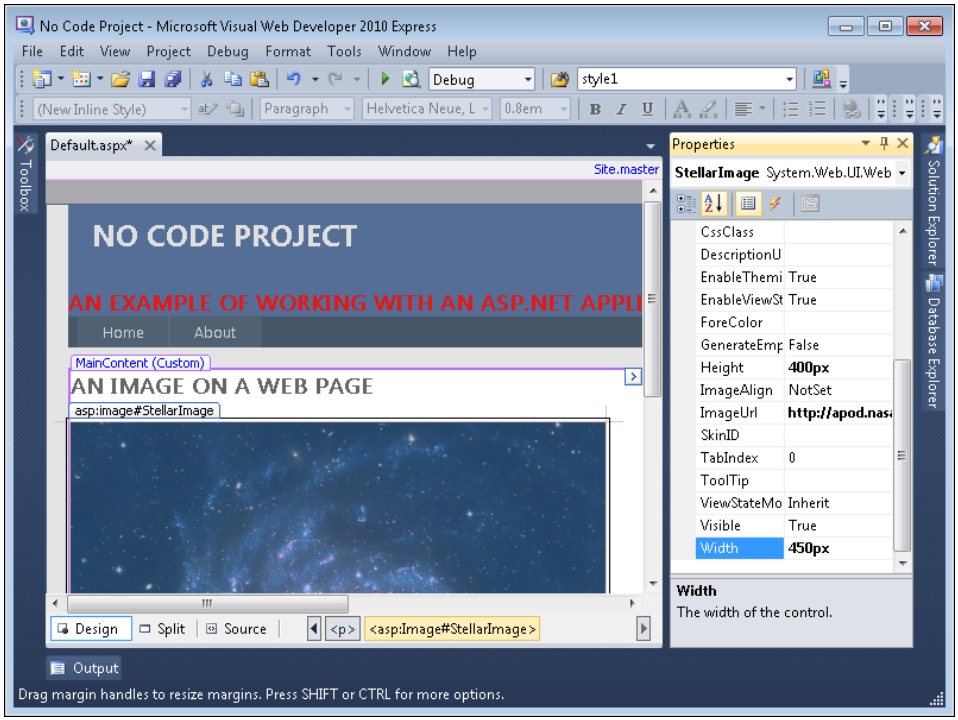
Add Content to Default.aspx

1. Highlight the existing heading text and type **An Image on a Web Page**. The next step is to display an actual image.
2. Highlight the existing text under the heading and delete it.
3. Click the Toolbox tab, and then click Auto Hide to keep it displayed. As with Windows Forms applications, you can use one of three techniques to add controls to a webpage:
   - Drag the control from the Toolbox and drop it onto the page.
   - Single-click a control within the Toolbox and then click the page where you want the control to appear.
   - Double-click the control within the Toolbox, placing it in a default location on the page.
4. Use one of the preceding three techniques to add an *Image* control to the webpage.
5. Close the Toolbox by clicking Auto Hide.
6. Display the Properties window by clicking its tab and then clicking Auto Hide.
7. Be sure that the *Image* control you added is selected, and then type **StellarImage** into the *(ID)* property field. The *(ID)* property serves the same purpose as the *(Name)* property for Windows Forms applications—it identifies the control so that you can access it easier later.
8. Type **400** in the *Height* property. This property sets the height of the image in pixels. If you don't set the image height, the page displays the image at the same size as the image source.

Tip To maintain an image's *aspect ratio* (the relationship between its height and width), you can set either the *Height* or *Width* property. The image automatically resizes the image in both dimensions to maintain the aspect ratio. For example, when the source image is 800 pixels wide by 600 pixels high, setting the *Height* property to 300 automatically changes the *Width* property to 400. Use the property that matters most to your site's layout.

9. Type **http://apod.nasa.gov/apod/image/1104/m74_baixauli_900.jpg** in the *ImageUrl* property. The image will display on the page automatically.

10. Type **450** in the *Width* property. This property sets the image width in pixels. If you don't set the image width, the page will display it at the original size (839 x 746), which is too large. Your Default.aspx page should now look like this.



At this point, it's helpful to close the Properties window and click Source. You'll see the source code used to create Default.aspx—there isn't much, as shown in Figure 2-6.

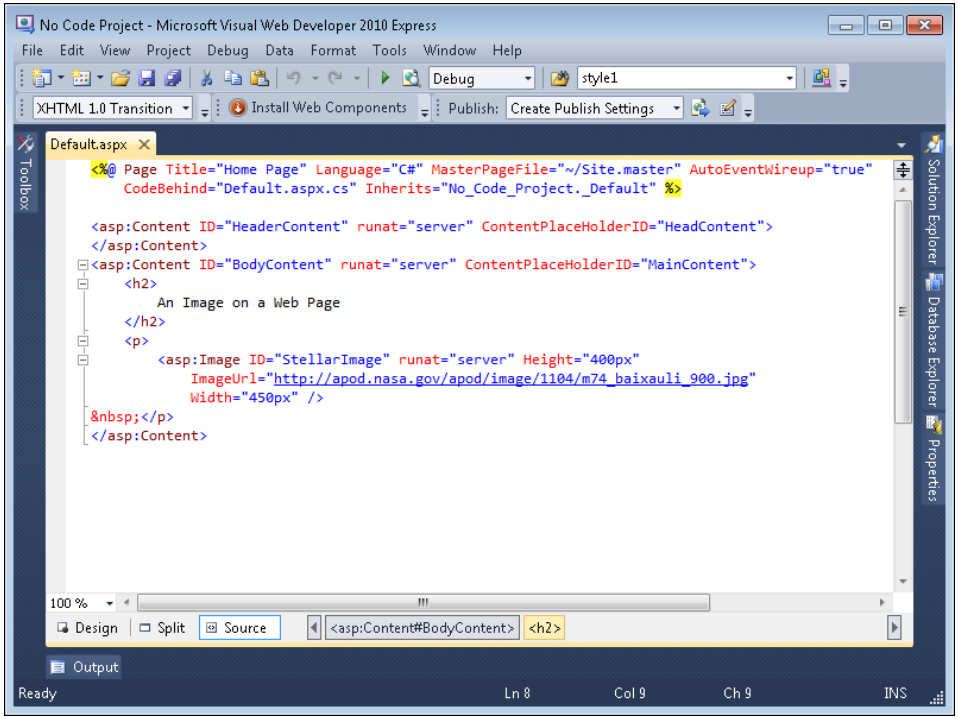


FIGURE 2-6 Even though the application output looks complex, it doesn't require much code.

The source code begins with some ASP script code. Any code you see that appears between the delimiters <% and %> is ASP script. This script defines programming-related features of Default.aspx, including the programming language (C#), the name of the master page file, and the name of the file used to hold the C# code for the page (the code behind file). Setting *AutoEventWireup* to *"true"* simply means that any events that the user generates on the page (such as clicking a button) will automatically get passed to the C# code that supports the page. The *Inherits* entry tells which class within the code behind file to use with this page. You'll discover more about ASP script later in this book; for now, all you really need to know is that entry defines some aspect of the page.

After the ASP script code, you see an *<asp:Content>* tag. This is also an ASP.NET entry that refers to a kind of control used on webpages. In this case, the control is described in the Master.Site file. The *ContentPlaceHolderID="HeadContent"* entry tells you that this is the header content from the Master.Site file. You can place header-specific information for Default.aspx here, such as *<meta>* tags that describe the page content. Meta-information is information about something else—in this case, *<meta>* tags describe the content of the page.

A second *<asp:Content>* tag appears next. This one uses the *ContentPlaceHolderID="MainContent"* entry from the Master.Site file. The content appears within this placeholder. There's a level 2 heading (the *<h2>* tag) that contains the content title you defined and a paragraph (*<p>* tag) that contains the *Image* control, which is actually an *<asp:Image>* tag. Each property you defined earlier appears as a separate attribute in the file. You'll see more examples of how this kind of content works as the book progresses.

Viewing the Master.Site File Code

The "Changing the Default.aspx Content" section earlier in this chapter explored the code used to define the default page. That code relies heavily on the master page code that resides in the Master. Site file. Reopen this file by clicking the Site.Master link in the Default.aspx file Design view. Click Source when the Master.Site opens. You'll see the code shown in Figure 2-7.

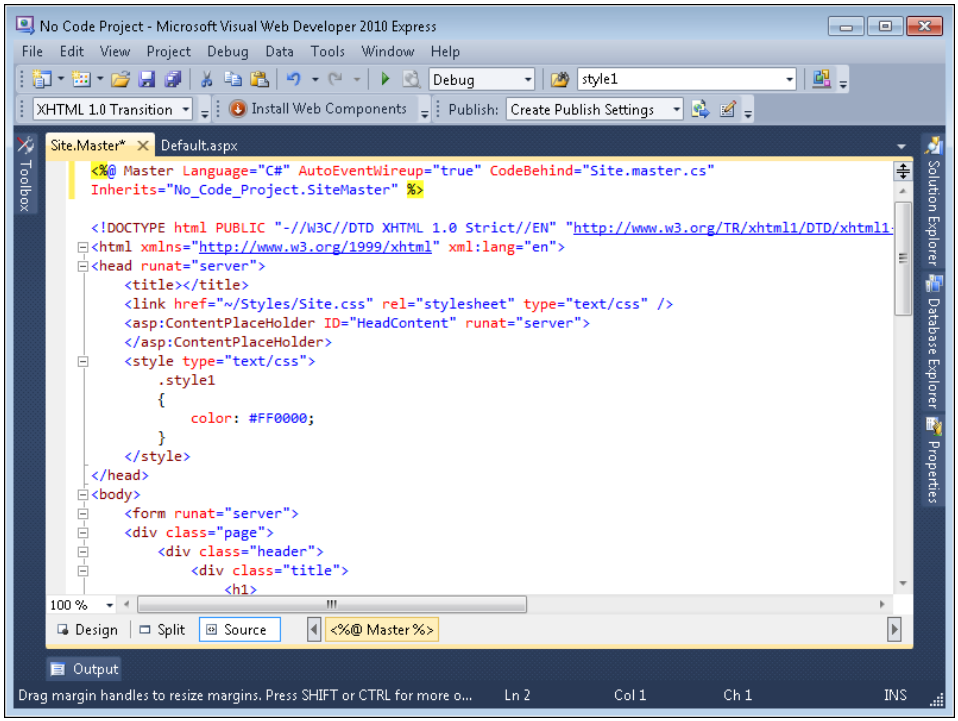


FIGURE 2-7 The Site.Master file contains a lot of code that applies to all pages that use it.

The first line is an ASP script similar to the one you saw in Default.aspx, and serves the same purpose. Of course, Master.Site doesn't contain any *MasterPageFile* entry—because it's the master page!

Immediately below the ASP script, you'll see some entries that you'd find in any webpage, such as the *<!DOCTYPE>*, *<html>*, and *<head>* tags. These are all standard for a webpage. However, look inside the *<head>* tag and you'll see some ASP.NET entries. The *<asp:ContentPlaceHolder ID="HeadContent" runat="server">* tag is a placeholder tag that defines the position of header content that will be added later by the various pages that rely on this master page. You'll remember

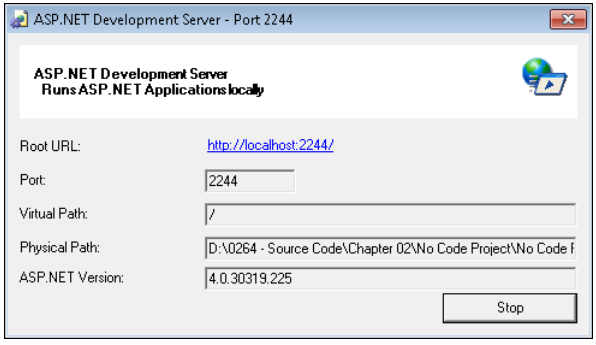
seeing the *HeadContent* identifier from the Default.aspx file—this is where that identifier comes from. The <*head*> tag also contains a <*link*> tag that points to the Site.css file, which defines all the styles for the site.

The "Working with the Master Page" section already discussed the <*body*> tag content briefly. One of the tags you want to pay attention to in the <*body*> tag is the <*asp:ContentPlaceHolder ID="MainContent" runat="server"/*> tag. This tag describes the other content placement tag you saw in Default.aspx. Those <*asp:Content*> tags are where you'll add page-specific content in the pages that rely on this master page. The other tags in the <*body*> tag describe the layout and content features common to all pages. Don't worry about getting too deeply into this information now; just view it, start becoming familiar with the tag names, and start thinking about how the various pieces interact with each other.

Viewing the Site in a Browser

You've looked at the master page, Master.Site, and a content page that relies on the master page, Default.aspx. It's time to see the application in action. Press F5, choose Debug | Start Debugging, or click Start Debugging on the Standard toolbar. The IDE starts the ASP.NET Development Server. This server appears as an icon in the Notification Area. Right-click the icon and you'll see three options on the shortcut menu:

- **Open in Web Browser** Opens a copy of the default page in the default browser. The server and the browser run independently. You can close the browser and reopen the page by choosing this option.
- **Stop** Stops the ASP.NET Development Server and shuts it down. This isn't the same as shutting down a web server installed on your system. You can restart the server at any time by pressing F5 again.
- **Show Details** Displays information about this particular ASP.NET Development server, as shown here (clicking the link opens a copy of the default page in your browser).



After the ASP.NET Development Server starts, it opens a copy of your default browser and displays the Default.aspx page, as shown in Figure 2-8.

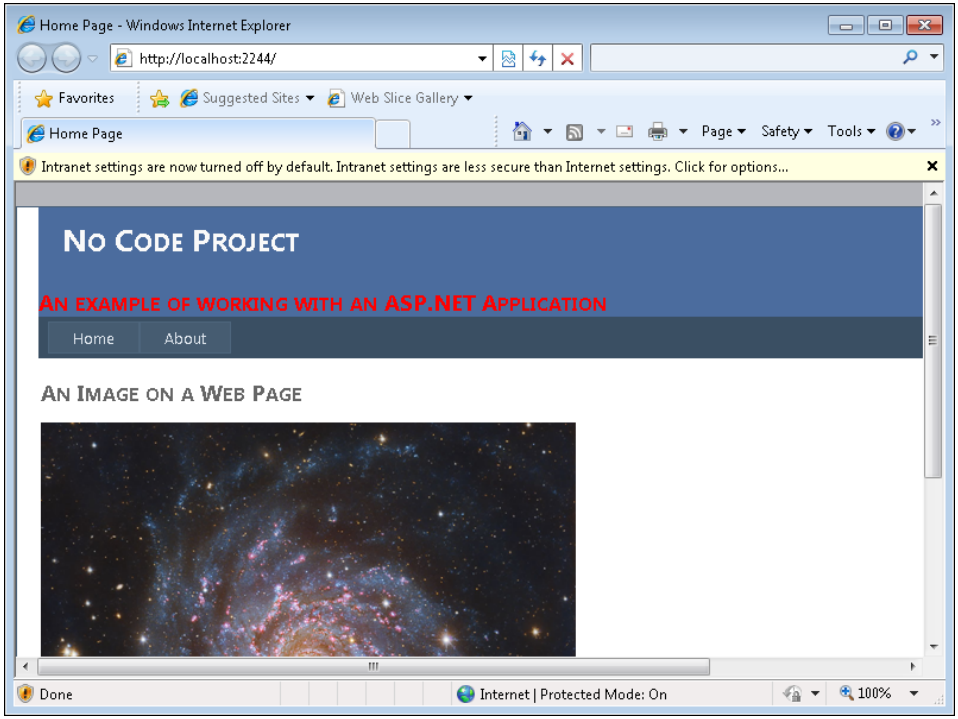


FIGURE 2-8 The example application displays a picture within a browser, and also provides access to other site features.

Notice that the URL contains a port setting (the *2244* after the *localhost* domain in the Address field). The IDE configures each of your applications to use a different, non-standard, port as a security feature. Using a non-standard port makes it less likely that someone will attempt to gain access to your system through the ASP.NET Development Server.

If you're using a default Internet Explorer setup, you'll likely see the warning note displayed at the top of the client window in this screenshot. Click the warning message and you'll see a shortcut menu. Choose the Enable Intranet Settings option. At this point, you'll see a message box warning you that intranet settings are less secure than Internet settings. Click Yes to enable the intranet settings so that you can easily debug your ASP.NET applications. The page will redisplay with all the features in a usable state.

Notice the two tabs on the page: Home and About. If you click About, you'll see the About.aspx page content. It doesn't look like the pages have changed, but the page content has. The Address field does change to show the change in pages, but the overall effect is that only the content changes, not the layout. ASP.NET provides a host of very cool effects that you'll try out as you go through the examples in the book. When you finish working with the example, right-click the ASP.NET Development Server icon in the Notification Area and choose Stop from the shortcut menu.

Creating the No Code Website

Visual Web Developer 2010 Express gives you a choice between creating a *project* and a *website*. There are situations when you will use a project instead of a website—each type has advantages and disadvantages. The purpose of this section is to explore the difference between projects and websites.

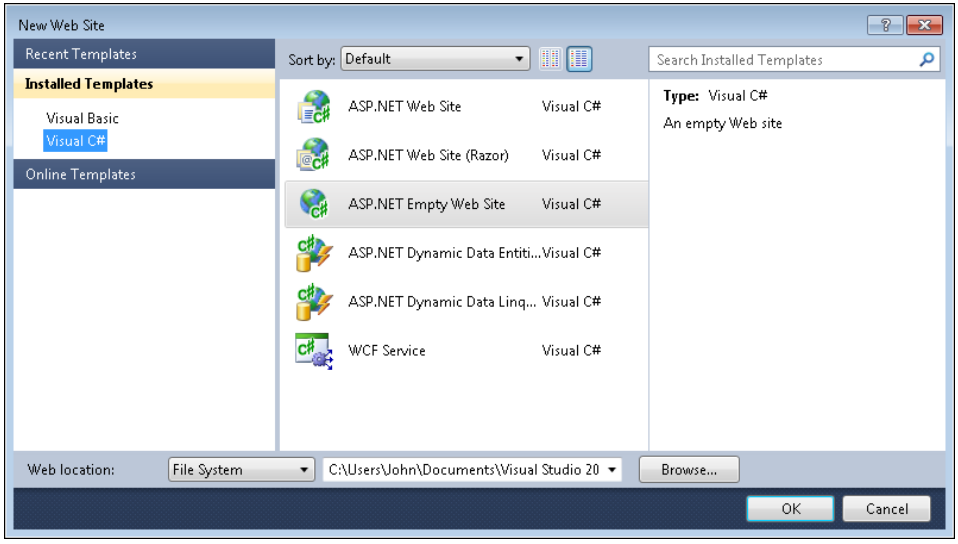
Defining a Website Location

A project always appears on your hard drive. You create the project as described in the "Starting the New Project" section of this chapter. Websites can begin on the hard drive, just like projects—but you can also create them on either a website, using the Hypertext Transfer Protocol (HTTP), or on a File Transfer Protocol (FTP) site, using FTP. The following steps help you get a new website started.

Create a New Website

1. Choose Start | All Programs | Microsoft Visual Studio 2010 Express | Microsoft Visual Web Developer 2010 Express. You'll see the IDE start up.

2. Click New Web Site. You'll see the New Web Site dialog box shown here.



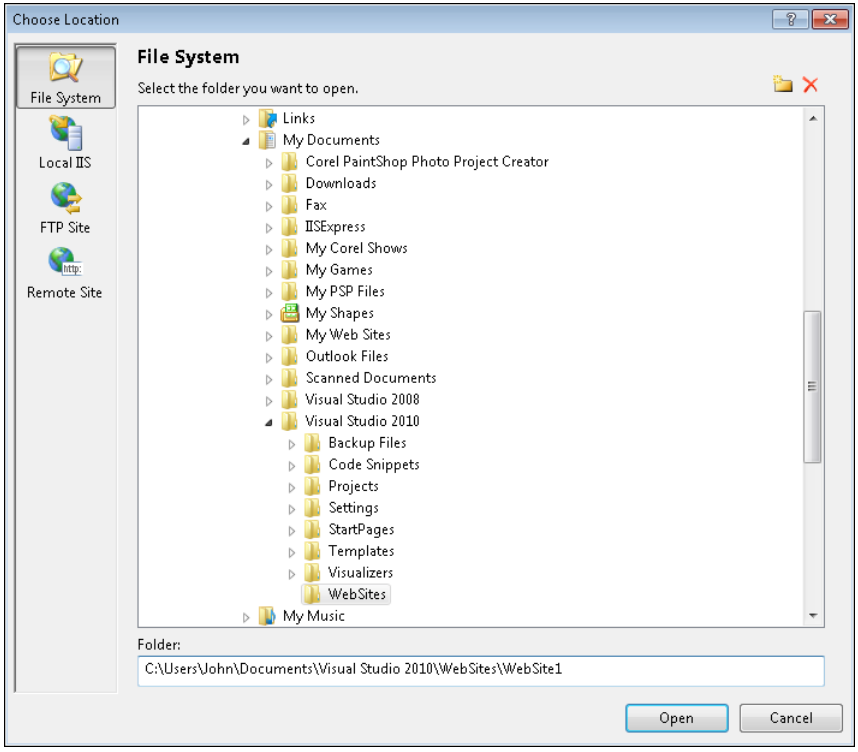
   One of the first things you should notice is that fewer projects are available when working with a new website. For example, no Silverlight projects are available when using this option, nor will you find an entry for using Azure. Even though a website offers more location flexibility, you lose the option of using certain types of templates. Of course, if you need the location flexibility, using a new website project will still likely be your best choice.

3. Select a project type. For this example application, select the ASP.NET Web Site template.

4. Select an option from the Web Location drop-down list. Use File System for this example, as shown in the preceding figure.

5. Provide a location (path) and name in the location field. When working with a website, you don't have the option of using a solution to group projects together. This example uses a File System connection in the default directory, with No Code Site as its location. You need to provide one of three kinds of information in this field, depending on the option you selected from the Web Location drop-down list:

   - **File System** Provide a path and website name. The default path is C:\Users\<User Name>\Documents\Visual Studio 2010\WebSites\, but you can use any location on a local hard drive or on a network drive that you can access. As with projects, the simple act of creating a project stores files on disk, which is why you must choose a storage location in the New Project dialog box. Click Browse to display a Choose Location dialog box like the one shown here where you can choose a file system location anywhere your system can access.



   - **HTTP** Supply a fully qualified URL for the website you want to use. The URL must include the *http://* protocol. Click Browse to display the Choose Location dialog box. In this case, you can choose between Local IIS and Remote Site options. In both cases, you end up with

a fully qualified URL pointing to the website. When working with a Local IIS site, you can also select the Use Secure Sockets Layer option to create a secure connection to the site (when the site supports the SSL).

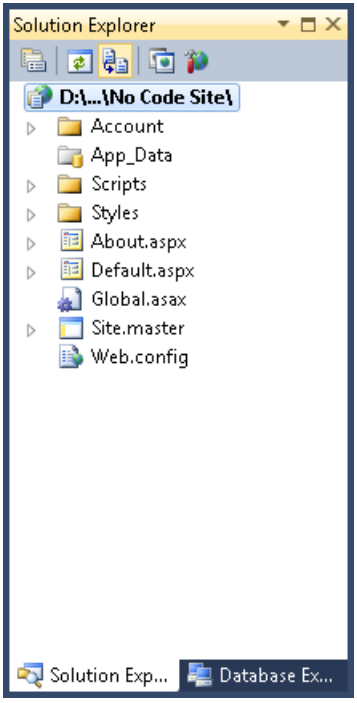
- **FTP** Supply a fully qualified URL and accompanying information to access an FTP site. Unless your site allows anonymous access, you must click Browse in this case to display the FTP information. This information includes the server domain, port number, initial server directory, whether to use passive mode, and the login information (name and password).

6. Click OK. The IDE creates a new website for you. The basic site features look precisely the same as the project features described earlier.

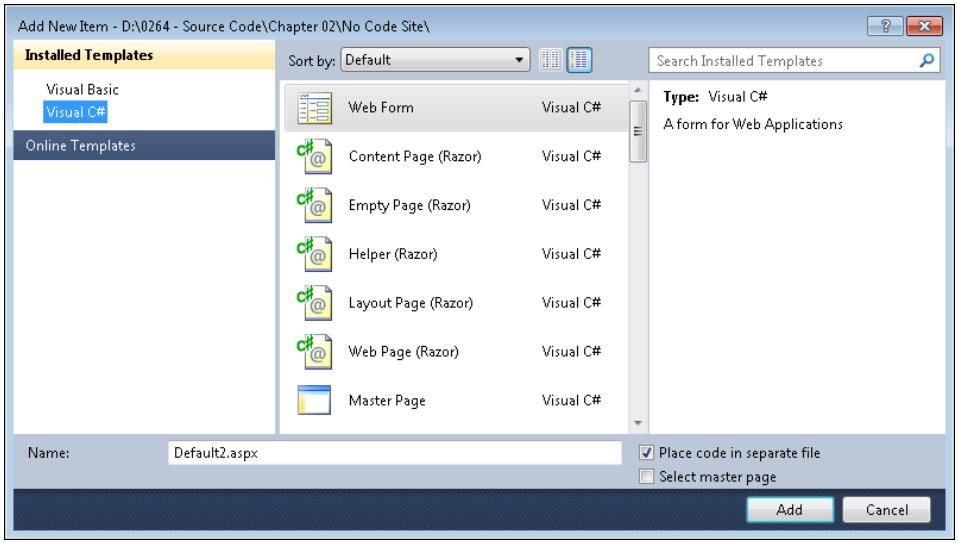
Adding a New Page

In the project example earlier in the chapter you modified Default.aspx. You could perform precisely the same changes in this site, but you can make other changes. In this case, you'll add another page to the site using the following steps.

1. Click the Solution Explorer tab and then click Auto Hide to keep the window open. You'll see a list of folders and files contained within the site, as shown here.



2. Right-click the topmost (site) entry in the list and choose Add New Item from the shortcut menu. You'll see the Add New Item dialog box, as shown on the next page.



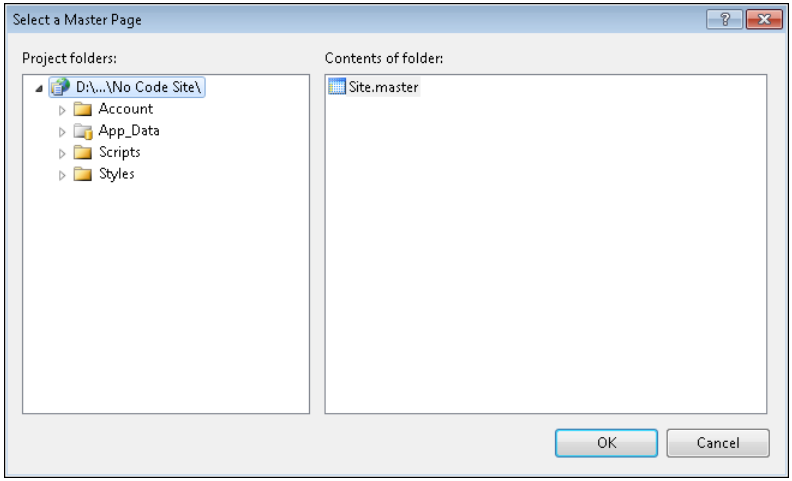
3. Highlight the Web Form entry, as shown in the figure. (As you can see from the figure, you can add quite a few items using this dialog box, some of which are discussed later in this book.)

4. Type **Image.aspx** in the Name field. This is the name of the file as it appears in Solution Explorer later.

5. Select the Select Master Page option. This selection will create a page that uses the existing master page, rather than a stand-alone page that uses its own layout and formatting.

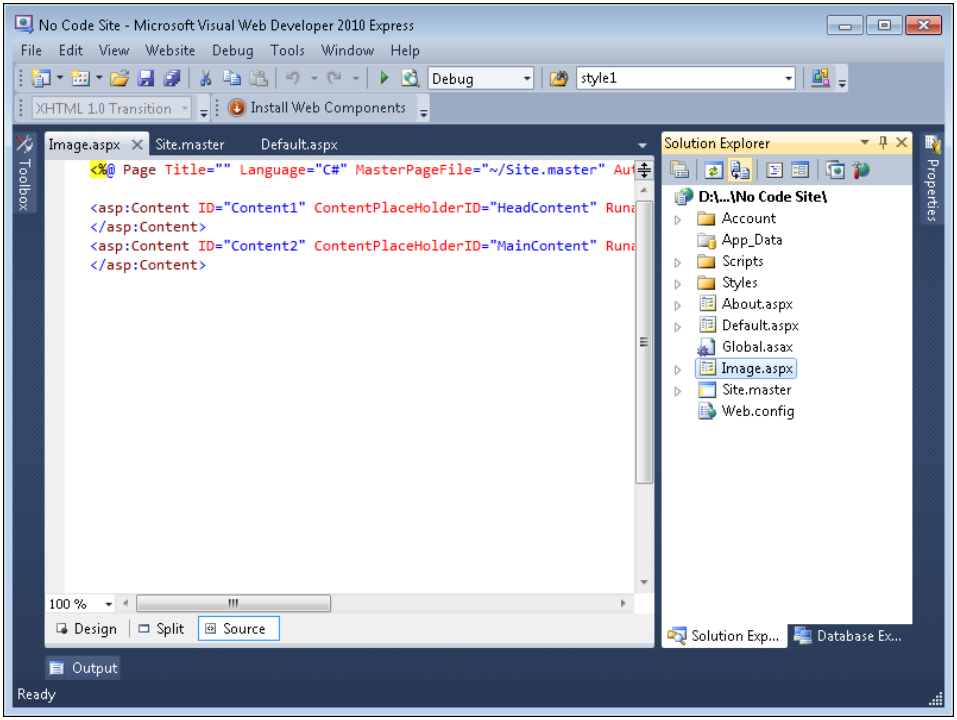
Note If you don't select this option, the resulting page won't look the same as the others on the site.

6. Click Add. You'll see the Select a Master Page dialog box shown here.



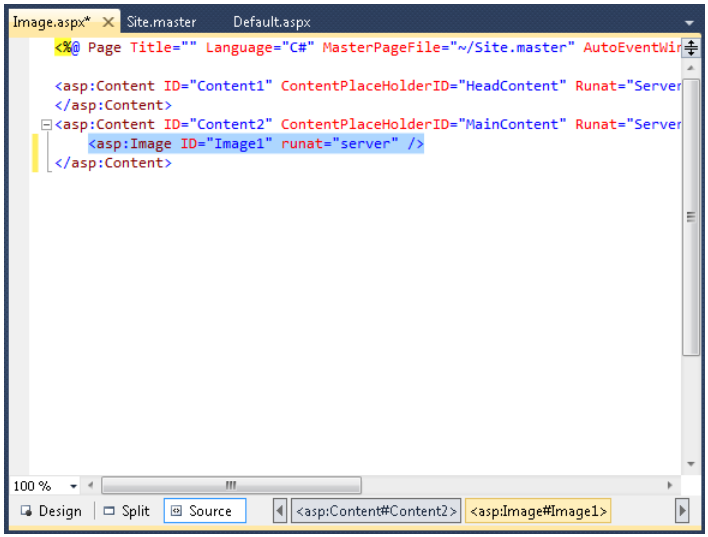
Because only one master page is associated with this site, you see only one entry in the list in the right pane. However, your site can use as many master pages as needed to fully define the characteristics of your site. If your site places the master pages in a special folder, you can navigate to that folder using the entries in the left pane.

7. Highlight Site.master and click OK. You'll see a new page added to your project as shown in Solution Explorer. The page contains only the ASP script and the two placeholder entries for the header and main content, as shown here.



8. Click Auto Hide in Solution Explorer to hide the window. Display the Toolbox by clicking its tab and then clicking Auto Hide.

9. Drag an Image control onto the Source window so that it appears like the one shown here.



Note When working with a web project or site, you can drag and drop controls into the Design or Source windows with equal ease. You can choose whichever solution works best for you.

10. Close the Toolbox by clicking Auto Hide.

11. Display the Properties window by clicking its tab and then clicking Auto Hide.

12. Type **StellarImage** in the *(ID)* property. Notice that you can see each of the changes you're making in the Source window. This is one advantage of using the Source window over using the Design window. Of course, you can't see what's actually happening to the control—all you can see is the code that your change is generating.

13. Type **400** in the *Height* property. This example won't set the *Width* property; the page automatically maintains the aspect ratio when you set just one of the *Width* or *Height* property values.

14. Type **http://apod.nasa.gov/apod/image/1104/m74_baixauli_900.jpg** in the *ImageUrl* property. Because you're working in the Source window, you won't see the image, but the image will appear if you click Design.

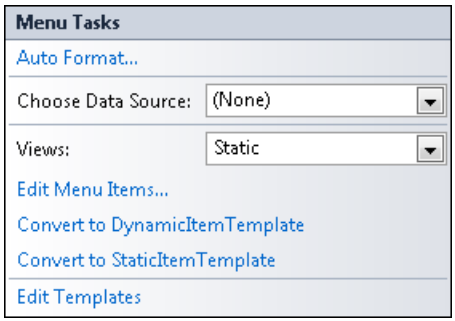
15. Close the Properties window by clicking Auto Hide.

Adding the Page to the Site Menu

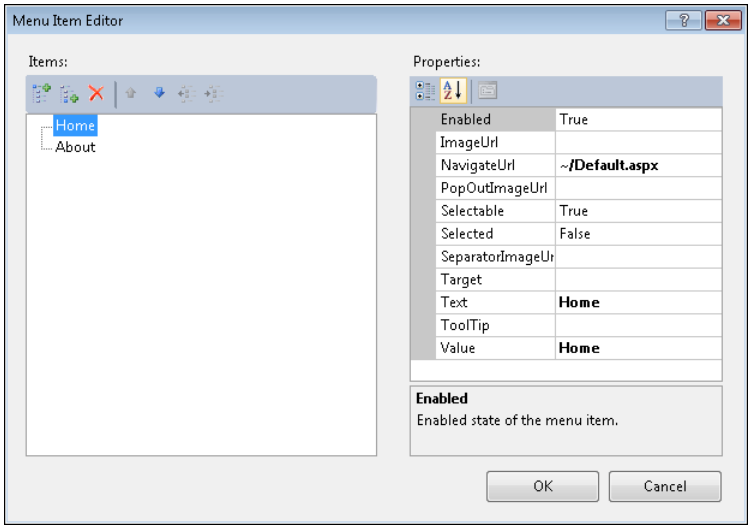
You have a shiny new page—but no way to access it. At this point, you need to add this new page to the master page so that you can select it in the browser.

1. Click Design on the new Image.aspx page. Click the Site.Master link in the upper-right corner. The Site.master file opens.

2. Select the square that contains the words Home and About. Notice the odd arrow that appears when you do this. Many controls provide a similar arrow. When you click the arrow, you see a Menu Tasks dialog box like the one shown here.



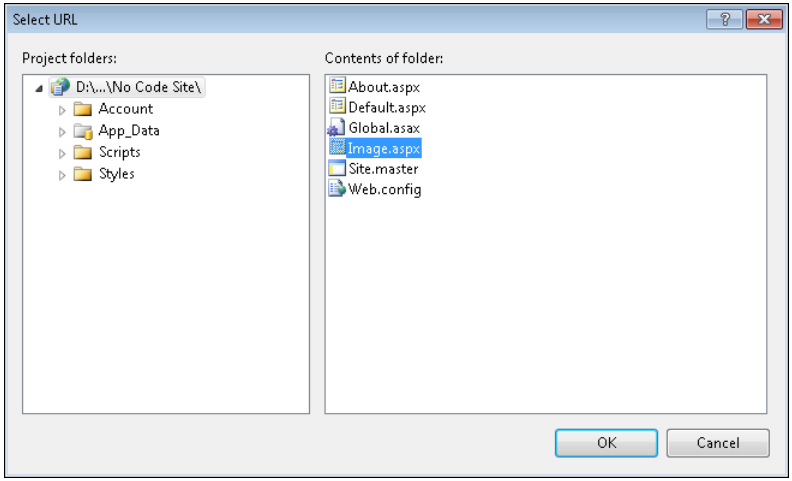
3. Click Edit Menu Items. You'll see the Menu Item Editor window shown here.



This editor lets you change the characteristics of this control without writing any code. As with many other tasks, the IDE writes the code for you in the background based on the input you provide. Writing code this way is less error prone and considerably easier, so always look for these handy control-specific editors whenever possible.

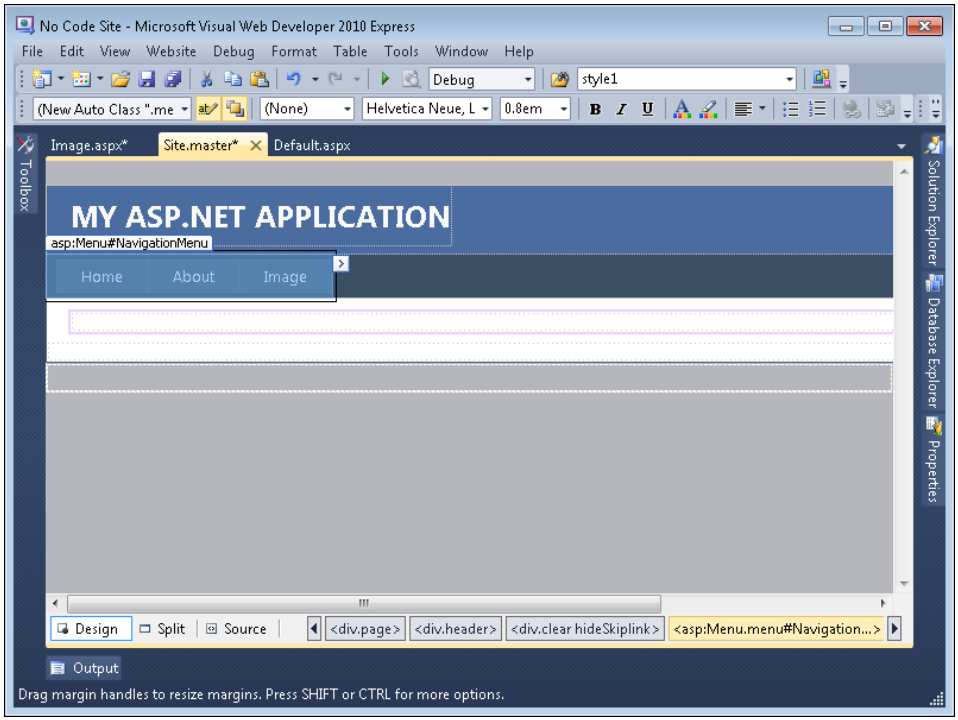
4. Click Add A Root Item. You'll see a new root item added to the list in the left pane.

5. Select the *NavigateUrl* property and then click the ellipsis button (...) that appears on the right side. You'll see the Select URL window shown here.



6. Highlight the Image.aspx entry in the right pane and click OK. The IDE automatically adds the correct entry to the *NavigateUrl* property for you.

7. Type **Image** in the *Text* property. Notice that the IDE automatically adds *Image* to the *Value* property for you. Click OK. The control now has a new entry, *Image*, as shown here.



You're ready to begin using the new page. When the application runs, you'll be able to select the new page you've added simply by clicking its tab.

Trying the Site in a Browser

It's time to try out the changes you've made to the site you created. Begin by choosing File | Save All, pressing Ctrl+Shift+S, or clicking Save All on the Standard toolbar to save your application changes. Now press F5, choose Debug | Start Debugging, or click Start Debugging on the Standard toolbar to see the website in your browser. At this point you see the message shown in Figure 2-9.

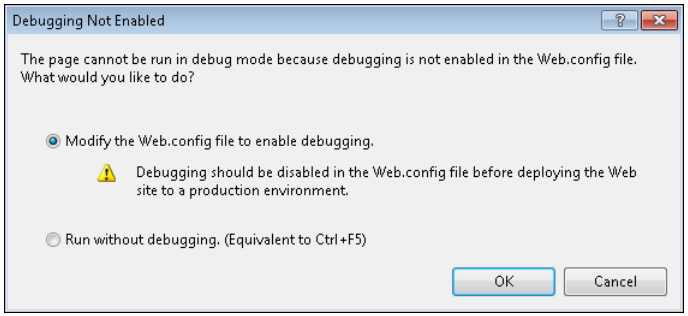


FIGURE 2-9 You must enable debugging in order to see what your website is doing.

A project is configured for a developer to work through issues from the outset and then create a production environment later. On the other hand, a site starts as a production environment, so you must specifically enable debugging. Select the Run Without Debugging option and click OK. The site opens in your browser. Click the Image tab and you'll see the new page you added, as shown in Figure 2-10.

Warning If you allow the IDE to modify the Web.config file, you'll need to compile the site code again before you can run it. Otherwise, the change won't appear when you run the site and you'll wonder why the change didn't take effect.

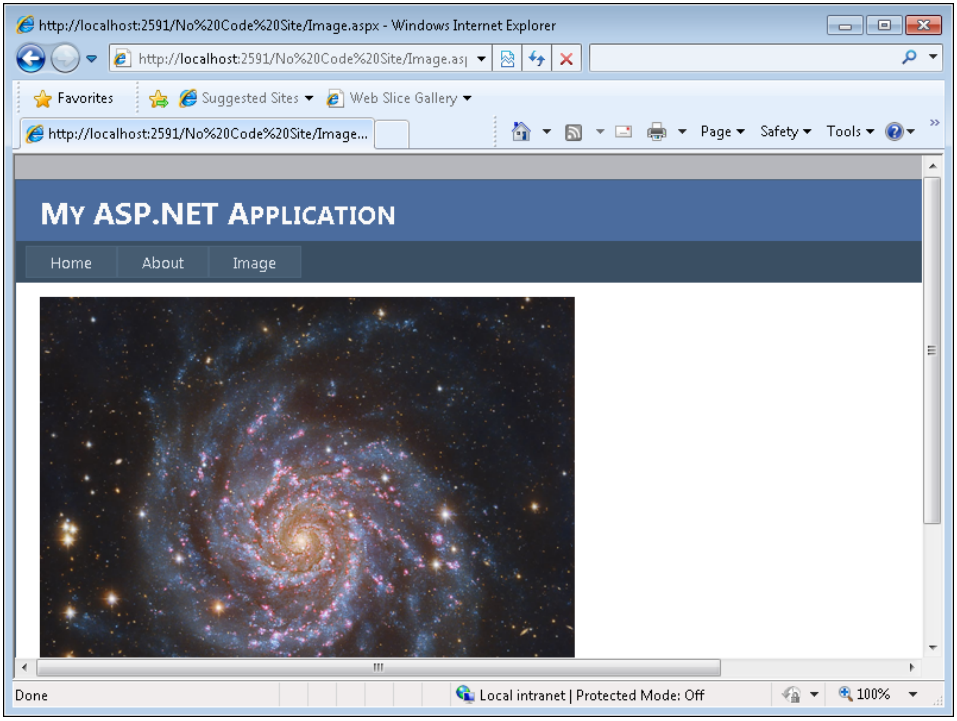


FIGURE 2-10 The new page contains an interesting image.

Feel free to explore the application. When you're finished, right-click the ASP.NET Development Server icon in the Notification Area and choose Stop. The server will stop, and you'll be able to make additional changes to your project.

Get Going with C#

This chapter introduced you to Visual Web Developer 2010 Express. As with the Visual C# 2010 Express introduction in Chapter 1, this chapter has just barely scratched the surface of creating an application, much less what you can do once you start adding code. However, it's amazing to see what the IDE can do for you without any coding on your part. Visual Web Developer helps you start the application, design the user interface, and even writes some of the code for you in the background. As you saw in this chapter, it's possible to create a small but usable application without writing any code at all. You can depend on the IDE to perform quite a lot of work for you.

You can follow many tracks just by using the information in this chapter. For example, you might want to try to create a Silverlight application using the same techniques you used in this chapter to create a project. Check out the other kinds of projects you can create as well. The right pane of the New Project dialog box describes these other project types when you select them.

Make sure you spend some time examining the Toolbox controls as well. Try playing with some of these controls in a test application to see what they do. Remember that playing with the programming environment is an extremely good way to learn. Don't be afraid to experiment. Try listing a few of the controls that you think you might be able to configure and use to create another application without writing any code. All this experimentation will build your knowledge of C# and Visual Web Developer.

Note Any project you create and modify without saving is temporary. When you try to close the project, the IDE will ask if you want to save the project. Click No and the project is placed in the Recycle Bin. If you later decide that you really did want to save that project, you can drag it from the Recycle Bin to a location on your hard drive.

The next chapter introduces you to some coding techniques. However, this book takes a different approach from many other texts in that it leaps right into something truly useful, Language Integrated Query (LINQ). Using LINQ is an interesting experience because it doesn't treat programming as an obscure, abstract task that only those with vast knowledge can perform. Instead, it treats applications as a source for answering questions. That's right, the basis of LINQ is to provide you with a way to ask questions about data and obtain a result. You'll find that Chapter 3, "Basic Data Manipulation Techniques," is a real eye opener when it comes to programming.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

About the Author

JOHN PAUL MUELLER is a freelance author and technical editor. He has writing in his blood, having produced 88 books and over 300 articles to date. The topics range from networking to artificial intelligence and from database management to heads-down programming. Some of his current books include a Windows command-line reference, books on VBA and Visio 2007, a C# design and development manual, and an IronPython programmer's guide. His technical editing skills have helped more than 60 authors refine the content of their manuscripts. John has provided technical editing services to both *Data Based Advisor* and *Coast Compute* magazines. He's also contributed articles to magazines such as *Software Quality Connection, DevSource, InformIT, SQL Server Professional, Visual C++ Developer, Hard Core Visual Basic, asp.netPRO, Software Test and Performance*, and *Visual Basic Developer.* Be sure to read John's blog at *http://blog.johnmuellerbooks.com/.*

When John isn't working at the computer, you can find him outside in the garden, cutting wood, or generally enjoying nature. John also likes making wine and knitting. When not occupied with anything else, he makes glycerin soap and candles, which come in handy for gift baskets. You can reach John on the Internet at *John@JohnMuellerBooks.com*. John is also setting up a website at *http://www.johnmuellerbooks.com/.* Feel free to take a look and make suggestions on how he can improve it.

Microsoft

Your Free eBook Reference

When you purchase this title, you also get the companion volume, *Start Here!™ Fundamentals of Microsoft® .NET Programming*, for free.

To download your eBook, go to
http://go.microsoft.com/FWLink/?Linkid=230718
and follow the instructions.

Need help? Please contact:
mspinput@microsoft.com

What do you think of this book?

We want to hear from you!

To participate in a brief online survey, please visit:

microsoft.com/learning/booksurvey

Tell us how well this book meets your needs—what works effectively, and what we can do better. Your feedback will help us continually improve our books and learning resources for you.

Thank you in advance for your input!

Microsoft® Press