Introduction

The release of Visual Studio 2005 and Visual Studio Team Systems marked a major revision to the .NET development experience. It brought us code snippets, custom project templates, refactoring, data binding wizards, smart tags, modeling tools, automated testing tools, and project and task management—to name just a few features.

Visual Studio 2008 builds on these tools and provides additional core changes and additions to the Visual Studio integrated development environment (IDE). The languages have many new improvements, the Framework has a number of additions, and the tools have been significantly enhanced. For instance, Visual Studio 2008 includes such things as Windows Presentation Foundation (WPF) for building richer client solutions, Windows Communication Foundation (WCF) to help build more dynamic service-oriented solutions, and Windows Workflow Foundation (WF) to enable structured programming around business processes. In addition, there are language enhancements such as the Language Integrated Query (LINQ) and team systems enhancements such as code metrics, performance profiling, and a revised team build system. All of these tools are meant to increase your productivity and success rate. This book is meant to help you unlock the many tools built into Visual Studio so that you can realize these gains.

Who Should Read This Book?

Developers who rely on Visual Studio to get work done will want to read this book. It provides great detail on the many features inside the latest version of the IDE. The book covers all the following key topics:

- Understanding the basics of solutions, projects, editors, and designers
- Writing macros, add-ins, and wizards
- Debugging with the IDE
- Refactoring code
- Sharing code with team members and the larger community
- Writing ASP.NET applications
- Writing and consuming web services and using the Windows Communication Foundation (WCF)
- Coding with Windows forms and with Windows Presentation Foundation (WPF)
- Working with data and databases
- Creating and hosting workflow-based applications using Windows Workflow Foundation (WF)
Using team collaboration and the Visual Studio Team System products

Modeling applications

Testing applications at both the system and unit test level

Managing source code changes and builds

This book is not a language book; it is a tools book. If you are trying to understand Visual Basic or C#, you will want a companion book that focuses on those subjects. If you can write C# or Visual Basic code, this book will radically help you to optimize your productivity with Visual Studio. Again, this book is not a primer on the .NET languages. However, we do cover the new language features (such as LINQ) in both C# and Visual Basic. We also try to provide simple examples that can be read by developers of both languages. By and large, however, this book has one primary focus: detailing and explaining the intricacies of the Visual Studio 2008 IDE to enable developers to be more productive.

How Is This Book Organized?

You can read this book cover to cover, or you can pick the chapters that apply most to your current need. We sometimes reference content across chapters, but for the most part, each chapter can stand by itself. This organization allows you to jump around and read as time (and interest) permits. There are four parts to the book; each part is described next.

Part I: An Introduction to Visual Studio 2008

The chapters in this part provide an overview of what to expect from Visual Studio 2008. Readers who are familiar only with prior versions of Visual Studio will want to review these chapters. In addition, we cover the new language enhancement for the 2008 versions of VB and C#.

Part II: An In-Depth Look at the IDE

This part covers the core development experience relative to Visual Studio. It provides developers with a base understanding of the rich features of their primary tool. The chapters walk through the many menus and windows that define each tool. We cover the base concepts of projects and solutions, and we explore in detail the explorers, editors, and designers.

Part III: Writing and Working with Code

Part III builds on the topics discussed in Part II by digging into the powerful productivity features of Visual Studio 2008. These chapters investigate the developer productivity aids that are present in the IDE, and discuss how to best use Visual Studio for refactoring and debugging your code.
**Part IV: Extending Visual Studio**

For those developers interested in customizing, automating, or extending the Visual Studio IDE, these chapters are for you. We explain the automation model and then document how to use that API to automate the IDE through macros. We also cover how you can extend the IDE’s capabilities by writing your own add-ins.

**Part V: Creating Enterprise Applications**

Part V focuses on how to work with the IDE tools to write your applications. Each chapter provides an in-depth overview of how to use Visual Studio to help you design and develop an application. We cover writing applications using ASP.NET, web services and WCF, Windows forms, WPF, WF, and working with data and databases.

**Part VI: Visual Studio Team System**

Finally, Part VI discusses the special set of Visual Studio versions collectively referred to as Visual Studio Team System (VSTS). We devote an entire chapter to each individual VSTS edition: Development Edition, Architecture Edition, Test Edition, and Database Edition. We also explore, in-depth, the key concepts of team collaboration, work item tracking, and version control using the VSTS client editions in conjunction with the Team Foundation Server product. And lastly, we discuss the concept of automated builds within the context of Visual Studio Team System.

**Conventions Used in This Book**

The following typographic conventions are used in this book:

Code lines, commands, statements, variables, and text you see onscreen appears in a monospace typeface.

Placeholders in syntax descriptions appear in an *italic monospace* typeface. You replace the placeholder with the actual filename, parameter, or whatever element it represents.

*Italics* highlight technical terms when they’re being defined.

A code-continuation icon is used before a line of code that is really a continuation of the preceding line. Sometimes a line of code is too long to fit as a single line on the page. If you see ➥ before a line of code, remember that it’s part of the line immediately above it.

The book also contains Notes, Tips, and Cautions to help you spot important or useful information more quickly.
CHAPTER 1

A Quick Tour of Visual Studio 2008

Windows Vista marked the release of the .NET Framework 3.0. This release included many great new features for the Windows developer. Among them was the initial release of Windows Presentation Foundation (WPF), Windows Communication Foundation (WCF), and Windows Workflow Foundation (WWF). However, Visual Studio remained unchanged. You could use these new features but they were not “built in” to the IDE. Instead, you had to use extensions to build applications on these .NET Framework elements and Visual Studio 2005.

Visual Studio 2008 represents a re-synch of the development tools and the .NET Framework. The .NET Framework evolves from 3.0 (released with Vista) to 3.5, an incremental release that ships with Visual Studio 2008. The IDE now natively supports WPF, WCF, and WWF out of the box. In addition, there are many other new bells and whistles including direct Office application support, CardSpace, LINQ, a large revision to ASP.NET, the CLR add-in framework, and more. In addition to Visual Studio, SQL Server will get an update in 2008.

NOTE

Although Visual Studio 2008 represents a major, milestone release, it is by no means the end for Visual Studio. On the heels of 2008 will be a large revision to Microsoft’s developer collaboration tool, Visual Studio Team Systems and Team Foundation Server.

If you’ve been doing this very long, you’ve come to expect a new release like Visual Studio 2008 to come with new
programming models, unfamiliar terms, fresh dialog boxes, and new ways to view code; it can be hard to find your footing on what seems to be unfamiliar ground. This chapter represents what to expect as a first encounter with Visual Studio 2008. We will first do a run-through of the tool to help you get your bearing. We'll then help you sort through the Visual Studio product line. Let this chapter serve as your map of what’s great in 2008; it will get you moving in the right direction.

NOTE

Part I, “An Introduction to Visual Studio 2008,” is broken into three chapters. This chapter covers what’s new inside the tools and documents the product line. Chapter 2, “A Quick Tour of the IDE,” is an introduction to getting the tool installed, running it, and creating that first project. It also serves to familiarize you with the basics of the IDE. In Chapter 3, “.NET Framework and Language Enhancements in 2008,” we will cover the language and .NET Framework enhancements that are part of the 2008 release.

Some Welcome Enhancements to the IDE

Visual Studio 2008 and the .NET Framework 3.5 introduce hundreds of new features to an already full-featured toolset. This latest version is about increasing developer productivity when writing applications targeted at the new version of the .NET Framework. This includes expanded project types, a reduction in mundane tasks, and ever-evolving aspects of team-oriented software engineering. This section and those that follow highlight these enhancements that promise to make your work life easier. Of course, we will go over each of these items in greater detail throughout the book; think of the content in this chapter as your “executive overview” for the hurried developer.

Use a Single Tool for Many Jobs

Many of us work in environments that include applications built on various versions of the .NET Framework. This becomes even more prevalent as more versions are released. For example, you may have an existing application in production built on .NET 2.0. You may be writing a new application on .NET 3.5. However, if your production application requires occasional maintenance, you do not want to have to keep two versions of Visual Studio on your machine.

Visual Studio 2008 supports the ability to target a specific version of the .NET Framework for an application. This means you can use a single tool to develop against many applications built on various .NET Framework flavors. Setting the .NET Framework version of an application will appropriately set the toolbox, project types, available references, and even IntelliSense inside the IDE to be in synch with the chosen .NET Framework version. Figure 1.1 shows creating a new application with Visual Studio 2008 and selecting the .NET Framework version (upper-right corner).

Notice the Add Reference dialog in Figure 1.2. It shows adding a reference to a Windows application that targets .NET Framework 2.0. In this instance, any component that is part of the 3.0 or 3.5 version of the .NET Framework is disabled (grayed out).
You can also decide to move your application to a different (hopefully newer) version of the .NET Framework. You can do so inside the project properties dialog (right-click your project file and select Properties). Figure 1.3 shows an example. Notice the Target Framework drop-down. You can change this and the IDE will then reset IntelliSense, reference, your toolbox, and more to the newly selected target framework.
Of course, you can use Visual Studio 2008 to open an existing application built on a prior version of the .NET Framework. When doing so, you have the option of upgrading or keeping it tied to the existing .NET Framework version. These features serve to help you upgrade to the advantages of 2008 and continue to work with applications built on an older version of the .NET Framework.

**NOTE**

The Framework setting is per project. Therefore, you can create a single solution that contains multiple projects and each can target a different version of the .NET Framework.

### Cleaner Windows

There have been a number of improvements to the overall management and access of the many Windows inside the IDE. Many user interface elements have a new look and new features.

**The IDE Navigator**

Developers can now navigate open windows in the IDE without touching a mouse. This keeps your fingers on the keyboard and can lead to greater productivity. Visual Studio 2008 provides a couple of options here. The first is a simple Window switching hotkey.
Suppose you have a number of code windows open in the IDE. To navigate forward (left to right) through them, you can use the key combination Ctrl+- (minus sign). This is for the standard development settings in the IDE; your settings may differ. To go backward (right to left), you use Ctrl+Shift+- (minus sign). This provides faster Window switching without your having to scroll with the mouse or search through your solution.

You can get similar results using a new visual aid called the IDE Navigator. This tool is similar to the Alt+Tab feature of Windows that allows for fast application switching. To access it, you use Ctrl+Tab (and Ctrl+Shift+Tab). You use this key combination to open the dialog and navigate open code Windows and active tool Windows. Figure 1.4 shows the result. Notice that active files are cycled through on the right.

![IDE Navigator](image)

FIGURE 1.4 The IDE Navigator in action.

**TIP**

To change the keyboard combinations assigned to the IDE navigator, select the menu option Tools, Options. Under the Environment node, select Keyboard. Here you can set keyboard shortcut keys. You will want to change the settings assigned to Window.NextDocumentWindowNav and Window.PreviousDocumentWindowNav.

Alternatively, you can access the IDE Navigator directly using Alt+F7. This brings up the tool with the active tool windows list selected. You can jump between the lists using the right- and left-arrow keys.

**Improved Docking**

In prior versions, it was often difficult to get your code or tool window to dock correctly in the IDE. In 2008, docking windows is much improved. There are new icons and visualizations that make this process very simple. Figure 1.5 shows an example of docking the
Server Explorer window on top of the Toolbox pane. You can see that there are options to move this window to the left of the toolbox, below it, and so on. Selecting each option shows a visual representation of the results before you release your mouse button.

![Figure 1.5 Improved window docking.](image)

**Standard Dialogs**
Another welcome change to the IDE is the use of Windows standard dialog boxes for doing such tasks as opening a file, saving something, or printing code. In prior version of Visual Studio, the IDE had its own versions of these common tasks. However, this only made things confusing because most Windows users are accustomed to working using specific tools.

As an example, consider Figures 1.6 and 1.7. Figure 1.6 is the Open File dialog in Visual Studio 2005. This dialog was specific to Visual Studio. Figure 1.7 shows the same dialog in Visual Studio 2008. Notice that the dialog is the same dialog you would get in any other Windows-based application.

**Choose Your Font**
There is a new setting called Environment Font inside the Options dialog (Tools menu) under the Environment node, Fonts and Colors. This option allows you to set the font for the entire IDE to the selection of your choice. Figure 1.8 shows selecting this option from the list.
Some Welcome Enhancements to the IDE

Changing this font changes your IDE. For example, suppose you set the Environment Font to Courier New, 8pt. This changes dialogs, menus, and more. Figure 1.9 shows the results of such a change.

Keep Your Settings

Many of you have customized your IDE to fit your exact needs. These settings can be painful to have to re-create. Thankfully, Visual Studio 2008 supports settings migration. If, for example, you already have Visual Studio 2005 installed, Visual Studio 2008 will allow you to migrate these settings on startup.
Alternatively, if you are upgrading to a new computer or want to share your Visual Studio 2008 settings, you can do so using the Import and Export Settings tool. This option is available from the Tools menu in both Visual Studio 2005 and 2008. It is a wizard that allows you to import settings, export them to a file, or reset your settings to one of the IDE defaults. Figure 1.10 shows an example of exporting settings.
Some Welcome Enhancements to the IDE

When you export or import settings, you can select those you want to apply. Figure 1.11 shows the second step in the export process. Notice you can pick and choose those items you want to export.

![Export and Import Settings Wizard](image)

**FIGURE 1.10** Exporting your settings.

**FIGURE 1.11** Selecting the settings you want to export.
The final step when exporting is to save the file as a \texttt{.vssettings} file. This file can then be shared among users. It can also be used to migrate settings from one PC and one IDE version to another. With it, you can rerun the Import and Export Settings Wizard, choosing to import settings from a file.

**Share (and Consume) Code with the Community**

Writing code is often a community thing. The developer community is broad and, for the most part, supportive of one another. Chances are, if you are having a problem, someone else has had the same issue. You can often reach out across the Internet and find solutions, components, sample code, articles, and the like that help to solve your issue. There is some kinship among developers that attracts them to posting sample code, newsgroup answers, and tips. Perhaps it is as simple as knowing that they may be the next in line to need an answer to a critical question, or maybe it’s just the need to show off to one another. Either way, Visual Studio 2008 continues the community support that was built into 2005.

You can still author components targeted at the development community. In 2008, however, you can now indicate whether your components are targeted at both 2005 and 2008 or simply one or the other.

Another change to the community features is the simplification of the community menu items into a couple of menu items under the Help menu. The first menu item links directly to MSDN Forums and replaces Ask a Question and Check Question Status. This allows you to link to the forums and work directly with the features there. The second item is Report a Bug. This menu option replaces Send Feedback.

For more information on the community features of Visual Studio 2008 (and MSDN Forums), see Chapter 7, “The .NET Community: Consuming and Creating Shared Code.”

**Expanded Class Designer Support**

The Class Designer was introduced in Visual Studio 2005. It provides a graphical means for writing and modifying classes. You can use it to define new classes and their relationships, add properties and methods to those classes, and modify elements within a property or method; it even allows for the refactoring of code. A change to a given method name within the Class Designer, for instance, will change the method’s name as well as update all the method’s callers to use the new name.

Visual Studio 2008 now provides Class Designer support for C++ applications. You can use it to visualize classes and their relationships. This includes seeing native C++ inheritance structures, enums, template classes, and more. There are, however, some limitations. If you are a C++ developer, you will want to review these inside of the production documentation.

The Class Designer is an integral part of Visual Studio Professional Edition and above. However, we cover it in detail inside the Visual Studio Team System part of the book, Chapter 26, “Development Edition.”
Develop User Applications

.NET has quickly become pervasive throughout the entire Windows product world. It took only a few short years, but it is now fair to say that .NET is everywhere; Windows programming and .NET programming are now synonymous. Many of the user applications we interact with have some if not all of their base in .NET. This includes web applications, rich clients built on Windows, mobile applications, Office solutions, smart clients that work across the Web, and more. The good news is that the .NET developer is in high demand, and you can leverage your skills to target a wide audience.

Figure 1.12 shows the New Project dialog in Visual Studio 2008; it serves as an example of the myriad of user solutions that are possible with .NET.

Although Figure 1.12 shows many project templates, it does not represent a road map with respect to user applications. The .NET developer has many choices for creating the user experience using both Visual Studio and the new Expressions Studio (discussed later in the chapter). The following list is meant to provide an overview of the presentation technologies available to the .NET developer:

- **ASP.NET**—This allows you to build web-based (and browser-based) solutions using HTML, AJAX, and server-side processing.

- **NetCF**—The .NET Compact Framework 3.5 runs on small devices and allows you to build applications that target these mobile devices.
Sliverlight—This is Microsoft’s new solution for developing highly interactive solutions experiences that combine video and animation, delivered across the Web for both Windows and Mac.

VSTO—Visual Studio Tools for Office allows you to build solutions based on the Office productivity tools (including Outlook and SharePoint).

WinForms—These are Windows forms used to deliver business applications and tools built on the windows platform. WinForms applications can be stand-alone or data-driven. In addition, WinForm applications may connect to web services, leverage resources on the client, and more.

WPF—Windows Presentation Foundation combines WinForms, XAML, Smart Clients, 3D graphics, and more to allow you to create the richest, most fully featured client solutions that run on Windows. WPF applications can be delivered similar to a WinForms application. In addition, they can exist as a browser-hosted solution that runs in a security sandbox.

XNA—This technology allows you to build Xbox games using Visual Studio (a topic for another book).

Each of these technologies is supported by Visual Studio 2008. With them, you have many options for creating user applications. This section highlights a number of development improvements in Visual Studio 2008 with respect to developing the user experience.

NOTE

There are many new client options in Visual Studio 2008. We highlight them all here. However, these and more are covered in greater detail throughout the rest of the book. Chapter 14, “Creating ASP.NET Applications,” covers ASP.NET; Chapter 15, “Building Windows Forms Applications,” is about standard Windows forms; Chapter 16, “Creating Richer, Smarter User Interfaces,” focuses squarely on WPF; Chapter 17, “Creating Rich Browser Applications,” discusses AJAX and XBAP applications; and Chapter 21, “Developing Office Business Applications,” covers creating solutions based on Microsoft Office.

Enhance the Web Developer’s Productivity

The vast majority of applications built these days involve some semblance of a web component—be it a full-blown browser-based, web application; a smart client that works across the Web; a web service; or otherwise. In fact, the line between a traditional rich client and a web application is blurring. Technologies such as AJAX (Asynchronous JavaScript and XML), Web Services, Smart Clients, and XAML (Extensible Application Markup Language) have ensured this. You can now build rich user experiences as your needs dictate. Of course, Microsoft has remained suitably focused on expanding Visual Studio’s capabilities with respect to web development.

Web developers want tools that help them through the common functions of building their application. Let’s face it, the HTML, CSS, and XML standards can sometimes be a pain to follow by memory and by hand. Instead, we want tools that guide us. And of
course, as soon as we want to work with code, we want to be able to access the entire related source and massage it as necessary. Visual Studio 2008 builds on recent (2005) enhancement. It’s a step forward in adding productivity to the web development day. We highlight many of these advancements in the following sections.

NOTE

We will cover many of the items that follow (and more) in greater detail in Chapter 14.

Create a Richer Web Interface

AJAX represents the capability to leverage the ubiquitous support for JavaScript in web browsers to create a more interactive user experience. Client applications built to leverage AJAX still have a client-server paradigm. However, with AJAX the client can update portions of a given page without appearing to have posted back to the server (of course, it typically does). In addition, most AJAX-enabled applications put more processing on the client for things like toggling sections of a page, working with tabs, auto-completing data entry, popping up dialogs, and more. The result is a step forward in interactivity for a user.

AJAX is not a Microsoft-specific technology. It represents more of a programming model. However, Microsoft released the AJAX Extensions for Visual Studio targeting both the 2003 and 2005 versions. These controls allowed developers to more easily create AJAX experiences. These controls have been enhanced and are now incorporated into Visual Studio 2008 and the .NET Framework 3.5. Figure 1.13 shows the controls inside the Visual Studio Toolbox.

These controls allow you to create a page that can receive partial, asynchronous updates (using UpdatePanel) and show update progress (using UpdateProgress). They also allow you to create your own controls and features that implement AJAX without having to write the client-side JavaScript.

In addition to these controls, Visual Studio 2008 supports IntelliSense, code comment documentation, and client-side debugging for JavaScript. It also contains the Microsoft AJAX Library, which is a JavaScript common library that supports object-oriented development for JavaScript. For a detailed overview of these items, AJAX, and more, see Chapter 17.

Microsoft Silverlight

Microsoft’s Silverlight is another exciting client technology for the Web. Silverlight allows for an even greater user experience delivered through the browser. You use it to create media-rich, highly interactive experiences. Silverlight requires a browser add-on (or plug-in). It works with Windows, Mac, and Linux in a wide variety of browsers. Silverlight does not ship with Visual Studio 2008; however, the Silverlight extensions for Visual Studio are available as a plug-in to the tool.

Work with an Expanded Toolbox

Visual Studio 2008 provides a rich set of tools and controls for the web developer. It still supports the standard controls for ASP.NET and HTML. This includes labels, text boxes, buttons, and the like. In addition, the validation, data, WebParts, and login controls are all still present. There are, of course, enhancements to many of these controls. However, here we focus on introducing the new controls for the ASP.NET developer.

There are three ASP.NET user interface controls that should be highlighted to the web developer: ListView, DataPager, and LinqDataSource. The first of these, ListView, is a control that simplifies the display of repeating data. The ListView control is driven based on user templates. In this way, you can easily configure how you want your UI to behave during operations such as view, edit, add, and delete. In addition, the ListView supports sorting, paging, and, of course, data binding. Figure 1.14 shows the Configure ListView screen. Here you can choose a default Runtime View layout, visual styles, and more.

FIGURE 1.14 Configuring a ListView control.
The ListView control is template driven. You, of course, have complete access to the layout and control of these templates. In addition, the control can assist in setting up the appropriate template. Figure 1.15 shows the ListView Tasks. Notice that you can change the Current View shown in the WebForm designer to another template view.

Figure 1.16 shows the template view for the EditItemTemplate. Again, the template layout source is available for you to customize. In fact, in this example, we put the form labels and controls into a table for a cleaner look. Notice too that the record navigation is available from within these templates. This allows your users to cycle through sections of their data and make appropriate updates.

Another new control in 2008 that we’d like to highlight is the DataPager control. This control allows you to manage the paging of data and the UI associated with that paging. You can use this control by itself or embed it as part of another control you create. In fact, the ListView control mentioned previously already uses the DataPager control. You can associate other, data-bound controls to a DataPager by using the DataPager’s PagedControlID property (the given control must implement the IPageableItemContainer interface).

You have full control over the customization, layout, and behavior of the DataPager. Figure 1.17 shows the DataPager Fields editor (accessed from the control’s Tasks window). Notice that you can set appearance and behavior for all items associated with a given DataPager layout.
The final new ASP.NET control we will highlight here is the LinqDataSourceControl. LINQ (Language Integrated Query) is a new programming model introduced in Visual Studio 2008. It combines database query and .NET language. In this way, you can write strongly typed code (and not just simple strings) to query your data. This includes writing queries...
Develop User Applications

with full IntelliSense support that is based on your data. In addition, LINQ can work with
data from various data sources including SQL Server, XML, and many more. For a richer
overview of this new technology, check out Chapter 18, “Working with Databases.”

The LinqDataSourceControl gives you the ability to bind controls using the LINQ technol-
ogy in a familiar construct as other ASP.NET data sources. You can use the
LinqDataSourceControl to gain access to database data, in-memory collections of data,
data-source classes, and more. When you connect with these data sources, the
LinqDataSourceControl allows you to write your data interaction using the power of
LINQ. Your database query code for selecting, grouping, ordering, filtering, updating,
inserting, and deleting will all be strongly typed and based on LINQ. In fact, the
LinqDataSourceControl writes this code for you. Figure 1.18 shows an example of config-
uring a LinqDataSourceControl to connect and work with a LINQ data class.

Develop and Design at the Same Time

Most web form developers switch between the Source and Design view of a web form
many times during its development. The Source view allows you full access to editing the
XHTML of the page. Design view lets you see the page develop and gives access to the
many shortcuts attached to controls in the designer. Visual Studio 2008 makes this switch-
ing much easier. It provides a Split view. With it, you can see both the XHTML and the
Designer. Figure 1.19 shows an example.
Split view tries to keep both the source and the design in synch. This works when you drag items from the toolbox to either the Source or the Design view panes. However, the Design view can get out of synch when you are doing a lot of edits to your source. In these cases, the Design view indicates that it is out of synch (see the middle of Figure 1.19). A click on the Designer and everything is back in synch. This new view can make developing your web forms more intuitive and more productive.

Create and Manage Your Look

Styles and style sheets have been part of the web development world for years now. They help ensure a consistent look and feel that can be managed centrally. The CSS (cascading style sheet) specification also continues to evolve; it allow you to do more and more with the visual display of our applications. Of course, this introduces additional elements, attributes, and complexity. Thankfully, Visual Studio 2008 introduces better tools for creating and managing both inline styles (styles defined directly in your XHTML source) and style sheets (.css files).

One such tool is the Manage Styles pane available to the web form developer. Figure 1.20 shows the pane in action. From here you can attach a style sheet to a web form, create a new style, preview styles, and more. Notice the Options button and related menu. These options help you control which styles you want to see and how you want to see them.

Also revised for 2008 is the New Style/Modify Style dialog. Here you can create or modify a style based on the appropriate CSS spec (through 2.1). This dialog has been revamped to give you access to new features and make things a bit more intuitive. You can access this
dialog from the Build Style button on the Style Sheet toolbar, from within a style sheet, or from within the Manage Styles pane using the New Style option. Figure 1.21 shows an example of the New Style dialog as accessed from the Manage Styles pane.

You can quickly apply styles to the elements on your page using the Apply Styles pane. Here you can see all the styles define on your form or the attached style sheet. Each style is shown with a visual representation for quick access. To apply the style, you simply select the form element and click the style. Figure 1.22 shows an example of this feature. Notice too that the ToolTip text for a style shows its CSS definition.

The final, new style management item we will cover here is the CSS Properties pane. Here you can quickly see the properties of a given style and manage them as you would the properties of a control. Figure 1.23 shows the use of this pane. Notice that you can work with styles defined within your web form. Each style is also grouped or available alphabetically. This can make things a bit easier for those who do not like to hunt through IntelliSense when defining styles.

**Centrally Manage Navigation and Design**
Visual Studio 2005 introduced the capability to create master pages. These pages centralized the management of a site’s design and navigation elements. In addition, master pages are supported by the designer, which allows for a richer design experience (and replaced the need for include files that were only seen in a browser). A developer is able to see the page in the context of the site’s central design while in design mode.
FIGURE 1.21  Creating or modifying a style.

FIGURE 1.22  Applying styles to your source.
One drawback, however, was that you could have only one level of master page. You could not nest master pages inside of each other and view the results in the designer. For example, a common scenario is to define a main master page to contain your site's frame, navigation, footer information, and the like. You might then create separate, sub–master pages for various content types. These sub–master pages would have their own, central design elements in addition to those defined by the main master page. Again, this scenario is not supported in 2005.

Thankfully, Visual Studio 2008 does support this scenario. With it, you can select an existing master page when creating a new master page. Figure 1.24 shows doing just that. Previously, during creation of a master page, this option was disabled.

With the new, nested master page, you can define master content inside the content placeholder defined by the main master page. You then add a new content placeholder inside your nested page. Pages based on this nested master page will put their content inside this area. Of course, you can go multiple levels deep with nested master pages. The designer will then show your various master pages during design. Figure 1.25 shows a simple example of an .aspx page created from a nested master page.

**Smarter Clients**

Rich clients are having a bit of a renaissance in 2008. The line between a web-based and Windows-like user experience continues to blur thanks to many UI advancements in terms of technology and tools. We have already discussed AJAX. Here we look at how...
2008 allows you to create richer, smarter clients with Microsoft’s new Windows Presentation Foundation (WPF).

WPF is both an addition to the .NET Framework and a set of tools with which developers can create richer solutions for the Web. The presentation code itself is based on XAML (Extensible Application Markup Language). The UI XAML is created using the Visual Studio WPF designer (or a similar tool called Expression Blend). The XAML is then run by the .NET CLR. Because it is processed on the client and not bound by HTML, it runs as a vector-based, hardware-accelerated user experience. The result is an extremely rich and interactive experience that supports both 2D and 3D graphics.
Visual Studio provides a familiar experience for creating WPF solutions. You first define a WPF project and add WPF forms to the project. When creating your solution, you select a project type based on whether the application will run as a browser add-in or as an installed desktop client. Figure 1.26 shows the WPF project templates. Selecting WPF Application will create a basic WPF application that is pushed to or installed on a client. It may have access to local resources on the client.

![FIGURE 1.26 Creating a new WPF project with Visual Studio 2008.](image)

The WPF Browser Application, on the other hand, is meant to be deployed through a URL and run as a browser extension. The application, called an XBAP (XAML browser application), runs inside a sandbox. It does not have rights to the client machine and is cleaned up as part of the browser’s cache. It does not require a download provided that the user has the .NET Framework 3.5 on their machine. It can work with the browser’s cookies and is supported by both IE and Firefox.

Making the wrong choice here is not too problematic. You can move the WPF forms between application types. Note that the other two application types highlighted in Figure 1.26 are WPF User Controls and WPF Custom Control Library. Both are for creating reusable controls for WPF applications.

The next step in building your WPF form is to simply open it and drag and drop UI controls onto a design surface. One big difference for Windows developers, however, is that you now have control over the form layout code (or XAML). This is more akin to designing a web form with Visual Studio. Figure 1.27 shows the XAML designer in action.

![FIGURE 1.27 XAML designer in action.](image)

Notice that the XAML controls are listed in the toolbox on the left. While they are similar to Windows and web controls, they are a new set of controls just for WPF. Also, notice how the designer has a split view between the design surface and the XAML. These stay in
synch as you develop your code. Finally, the properties window shown on the left provides a similar experience when you’re editing the many properties of a XAML control.

**FIGURE 1.27** Designing a WPF form.

We cover the designer in greater detail in Chapter 16. Here we focus on the forms engine, the controls, event programming, debugging support, deployment, IntelliSense, configuration, and more inside of WPF.

**Designers and Developers**

When discussing WPF, it’s important to note the work that went into the technology and tools to support a strong designer-developer workflow. It’s understood that traditionally developers have been left to try to “design” the UI. It goes without saying that developers have not exactly shined as UI designers.

However, even in scenarios where designers were employed on projects, the design would often fall short or the implementation would be an arduous process. Designers had their own tools that did not talk with those of the developers. A design was often provided to the development team as a picture or some basic HTML. Developers were often left to try to realize the intricacies of the design while having to concern themselves with coding the solution. In the end, nobody was happy. The design was never exactly what was envisioned and developers would spend too much time trying to get the look right.

WPF tries to right this wrong. It keeps the UI markup (XAML) as totally separate from the implementation code (C# or VB). In addition, Microsoft has provided design tools that let
Develop User Applications

Designers create real user interfaces that can be leveraged by the development team. There are no more “lost in translation” issues. Instead, a designer can create or open a WPF UI element, edit it using the power of Expression Blend (they do not have to learn Visual Studio), and save it back to the solution or send it to a developer. The developer can then open the same item inside of Visual Studio and begin responding to key UI events with their code. This back-and-forth can continue as the UI builds up over time. Nobody’s code gets stepped on; everyone focuses on their strength.

Figure 1.28 shows a sample of the Expression Blend tool. Notice that the same .xaml file (WFP form) is open here. Designers can use a tool with which they are more familiar to lay out the UI and apply visual techniques such as 3D and animation. Also notice that the C# project file is being used to maintain continuity between the solutions. After the designers are finished, their save goes right back to the development team. They can even test their UI in the context of the application by building and running the project.

![Designing a WPF form inside of Expression Blend.](image)

**FIGURE 1.28** Designing a WPF form inside of Expression Blend.

**Expression Blend**

Microsoft’s Expression Blend is a sister tool to Visual Studio 2008. It gives designers the ability to create rich WPF (XAML) forms, animations, and more—all without code. You can also use it to build Silverlight applications.

Expression Blend is entirely built on WPF. It provides a great example of exactly what is possible with this technology.
Although the Expression suite of products is outside the scope of this book, you should be aware of these tools. We will provide an overview of them in the later section “The Visual Studio Product Line.”

Making a Choice
The many options for developing rich user experiences on Windows invariably leads to the question: “What is the right UI technology for my next Windows application?” Of course, the answer depends on your specific scenario. When making your decision, you should keep the intent of each technology in mind.

WPF is a Windows technology that requires the .NET Framework on the client. It can run in a browser sandbox or as a fully featured Windows application. You should leverage WPF when you need to create a rich user experience with high design aesthetics, want a low cost of installation and update, and require distributed connectivity.

Silverlight, on the other hand, requires only a browser plug-in. The browser plug-in runs a mini version of the CLR on the client. In this way, you still can write your code in C# or VB and have access to bits of the .NET Framework. Silverlight is a very lightweight, cross-platform version of WPF. Its core strength is video, animation, and sound (multimedia). Think of Silverlight as an alternative to Adobe’s Flash product.

Lastly, Windows forms still has a place in creating installed, business applications that connect with the operating system, leverage resources on the client, connect to databases, and more. They offer standard, workman-like user experiences for data entry, file manipulation, configuration, and related task-oriented solutions. They are easy to develop and do not typically involve much work from a design perspective.

Office-Based Solutions
Developers have been able to customize Office for a long time now; some of us still remember writing Excel macros on Windows 3.1 or automating Word with Word Basic. Visual Studio 2008 is another step forward for Office development. The tools are now are built into the IDE. With them, you can create Office-based projects and solutions that leverage Word, Excel, Project, Visio, PowerPoint, Outlook, SharePoint, and InfoPath.

There are project templates for all the Office products. In fact, you can create solutions for both Office 2003 and 2007. Figure 1.29 shows the New Project dialog for Office solutions. These are Visual Basic templates but the same exist for C#.

NOTE
Visual Studio Tools for Office (VSTO) is not part of Visual Studio Professional Edition. It is no longer a separate product. Visual Studio 2008 ships with the new VSTO 3.0. This should increase the awareness of these tools and their usage.

There are a few scenarios that would lead you to create an application based on Office. The most common is when you need to extend a line-of-business (LOB) application to
provide functionality inside the common, information-worker productivity tools of Office. A LOB typically has a lot of rich data and some workflow.

For example, you may work with a financial, manufacturing, or payroll application. Each of these fills a specific need. However, users may need to work with the data that is housed inside the application and make key decisions that feed back into these systems. This work is often done through cut-and-paste and is often not captured by the systems. Users lose productivity switching back and forth between the Office tools and the LOB. This is where you should consider creating an Office Business Application (OBA) to help bridge this gap.

**Develop Documents, Templates, and Add-Ins**

Notice the many templates in Figure 1.29. There are three separate templates for Excel 2007, for example. Each of these templates provides a specific purpose. Office application templates allow you to create solutions built on a single document, a document template, or as an add-in to the given Office application. The following list provides a brief overview of these three project subtypes:

- **Document**—Document projects allow you to build a solution based on a specific document. There are typically not multiple instances of the document. As an example, suppose you have an Excel document that needs to read and write project resource billing information from and to an ERP system. This document might be updated weekly as part of a resource meeting. The data should be up-to-date and changes should feed the billing system. In this instance, you would create a solution based on this single document.

![Image of VSTO templates](image.png)

**FIGURE 1.29** The many project templates inside VSTO.
Template—An Office Template project is one that is based on an Office template file (an Excel .xltx, for example). Creating a solution based on an Office template file gives you the flexibility to provide users with assistance when creating a new instance of a given template. You might push common document templates out to your users. When a user creates a new instance, the template may reach into data housed in other systems to help the user fill out the details of the document. You might then, in turn, capture the results in a database after routing the template through a basic SharePoint workflow.

Add-in—An Add-in project allow you to extend the features and functionality of a given Office application. You create Add-ins to offer additional productivity and solutions inside a given application. You might, for example, write an Outlook Add-in that allows users to more easily file and categorize their email.

Whichever template you choose, Visual Studio 2008 provides a rich, design-time experience for building your Office solution. As an example, Figure 1.30 shows the Visual Studio 2008 design experience building a solution for a Word 2007 Template. In this example, a user is creating a quote for training. The fields in the document pull from a LOB database that includes customer information, resource data, and common pricing.

VSTO 3.0 also provides design support for working with the Office 2007 Ribbon. In this way, your application can behave like the rest of Office. You can embed productivity features and aids on your own tab of the Office Ribbon. Figure 1.31 shows a simple...
Create SharePoint Solutions
SharePoint has become nearly as ubiquitous as Office. Companies are leveraging it for knowledge management, collaboration, and business process automation. Of course, this inevitably means customization and extension by developers. If you customized SharePoint using the existing tools for Visual Studio prior to 2008, you know that this presented a big challenge. The design-time support was not there, configuration was difficult, and debugging was very challenging.

Visual Studio 2008 presents a much richer toolset for SharePoint developers. With it, you can create SharePoint workflows (see the project templates back in Figure 1.29) and build WebParts based on ASP.NET. In addition, the debug experience has been reduced from approximately 15 to 5 steps in the IDE. SharePoint development is now a first-class consideration inside the IDE. This should streamline the process for extending SharePoint and help meet the business demand this collaboration product has generated.

This section highlighted many of the new features in VSTO 3.0 for Visual Studio 2008. There are many more advancements, however. These include access to open XML formats, the capability to add features to existing Outlook forms, custom task panes in Word and Excel, data binding, and improved deployment and security. We cover all of these and more in Chapter 21.
Target Mobile Devices

Visual Studio 2008 continues to extend the power to write applications that target mobile devices. This latest version allows you to build applications on a wide variety of technologies and devices. You can build solutions based on Windows Mobile 2003, Windows CE 5.0 and 6.0, and Windows Mobile 5.0 and 6.0 for both Smartphone and Pocket PC. In addition, you can target many devices and device types that leverage these operating systems and related versions of the .NET Compact Framework.

The IDE provides the forms designers, code editors, and debugging support for building mobile applications. You start with a Smart Device project. You can then select your Target Platform, the target version of the .NET Compact Framework (NetCF), and the project template type (forms application, class library, control, and so on). Figure 1.32 shows an example. Here the application is targeting Windows Mobile 5.0 and the latest version of NetCF.

Designing Windows Mobile forms is a familiar process. There is a set of device controls in the toolbox. You select a control and drop it on a Mobile form. The form looks and acts like the device you are targeting. Figure 1.33 is an example of the design experience. This example is a simple customer search form. Notice the menus at the bottom of the screen. You use these menus to allow the user to work with the form from the major keys on the phone (of course, a Smartphone has no stylus or mouse).

As you develop your application, you can deploy and test it against device emulators. This allows you to debug your application and see how it behaves on a simulated device. Visual Studio 2008 ships with the basic emulators for Smartphones and Pocket PC. However, there are more than 20 device emulators available for download. Figure 1.34 shows the
application deployed and running on the basic Smartphone emulator. From here, you can interact with the application and debug your code.

FIGURE 1.33  Designing a Smartphone form.

FIGURE 1.34  Running a mobile application in a device emulator.
A big addition for 2008 is the capability to write unit tests for mobile device applications. You can now create these tests using the standard unit test framework built into Visual Studio as of 2005. These unit tests behave the same way. In addition, they can integrate with Team System products (like Team Test) and publish test data to Team Foundation Server.

The Compact Framework 3.5 provides some welcome enhancements. These include support for LINQ data binding, Windows Communication Foundation, better compression, advancements in sound, better security management, and more.

**Write Connected, Service-Oriented Solutions**

Many business applications involve specific processes, or workflows, around documents, records, and related data. These business processes typically involve staged review and approval by business personnel; they might also require communication between various systems. A business process is also typically long-running—meaning the process is not a simple, single execution but rather a multistep process that has to “wait” before moving to the next step.

Building these processes into business application was typically a lot of custom development work with little guidance, or it meant tying your application to third-party tools. Web services helped but developers lacked an easy way to build them with support for multiple protocols, different transports, strong security, and transactional support.

Visual Studio 2008 now provides in-the-box (and in the .NET Framework) support for building business processes as workflows and reliably integrating them with other applications, systems, and partners. This section takes a look at Windows Workflow (WF) for defining reusable business process and Windows Communication Foundation (WCF) for unlocking that business process across system boundaries.

**NOTE**

We will highlight .NET Framework and language enhancements (as opposed to IDE enhancements) to support communication and workflow in Chapter 3.

**Develop an Application/Business Process**

A workflow represents a set of actions (called activities) that are taken based on one another, business rules and conditions, and user actions. Workflows typically model business processes and often involve user interaction. Windows Workflow provides a framework, tools, and an engine for enabling workflows in your application. With it, you can create a workflow and know that it will run wherever the .NET Framework is installed.
You interact with workflows from within your application. This could be any type of application (Windows, ASP.NET, console, service, and so on). You typically submit data to the workflow and you might get a response, or you might use the workflow to kick off certain tasks or activities in your business process. The workflow can also persist itself during long-running transactions and then rehydrate on demand.

Two principal workflow types are built into Visual Studio 2008: sequential workflows and state-machine workflows. A sequential workflow follows a series of steps from top to bottom (as you design). Each step is gated based on the prior step and perhaps a set of conditions. A state-machine workflow, on the other hand, responds to different states (think status) being passed into the workflow. The states provide the gates between other states and may also trigger certain events.

You choose your workflow type at design time. There are both sequential and state-machine templates. Figure 1.35 shows an example of the workflow templates built into Visual Studio 2008. Notice also the SharePoint workflow templates. These provide the same workflows with extra hooks for SharePoint developers.

Building a workflow is a very visual process. The thought behind this is that workflows themselves are traditionally illustrated as a set of steps (boxes) and connections (arrows). You still have access to code, but the designer provides a strong visual tool for building both sequential and state-machine workflows. With it, you can indicate a start and end point, add activities to the diagram, and configure each activity in terms of its task.
Figure 1.36 shows a sequential workflow inside Visual Studio. On the left are the workflow activity shapes. You use these to visually represent your workflow. Each shape has a set of properties that require developer configuration. This simple workflow example allows for the approval of a sales quote. There is an if/then condition set to determine whether the sales quote requires approval. If so, the quote waits (or listens) for approval. The quote is then either approved or rejected by a user. If the user does not respond in a timely manner, a timeout message is sent. You can follow this workflow through the visual representation in the diagram.

The workflow toolset also includes a basic rules and conditions editor. This editor allows you to set and manage workflow rules. These rules are then output as XML to be configured at runtime should the need arise. Figure 1.37 shows an example of the rules editor. In this dialog, the sales quote value condition from the previous example is being set. This condition is then attached to the if/then activity in the diagram.

NOTE

Create and Consume Services

Most organizations have multiple systems, each designed for a specific purpose. They may have a financial system, HR, order management, inventory, customer service, and more. These applications each house specific business processes. However, most organizations need to unlock these business processes from their application and reuse them as part of an integrated solution. This is where service-oriented solutions have helped. By exposing an application’s business process as a service, multiple clients can take advantage of that process.

The promise of code reuse has been with us a long time. However, service-oriented code reuse became very popular with the advent of web services. The ubiquitous nature of HTTP and port 80 coupled with XML-based interfaces allowed for a new level of communication between application boundaries. Developers began wrapping key business functions as services and calling them from multiple clients.

Visual Studio 2008 and the .NET Framework 3.5 represent another step forward in this service-oriented paradigm. With these tools, you can create services based on Microsoft’s Windows Communication Foundation. WCF is a framework which recognizes that developers need multiple layers of communication (not just the SOAP protocol carried over an HTTP transport), require strong security, often need to support transactions, and do not want to write all the plumbing code to do so.

You create WCF services as a code library, as a workflow, or as a web service application. Figure 1.38 shows the new project templates listed under the WCF project types. From here you can indicate that your web service contains a workflow (think business process) or simply create a service library that will call custom code you write.
You can also still create .asmx web services through the web project templates. This same template area also provides access to the new, WCF Service Application template. With it, you can create a WCF service that is configured similarly to a web service.

WCF is all about configuration. It frees you from having to write all the service plumbing code. Instead, you can focus on the functionality of your service. For example, you can add service endpoints to your service depending on which communication stack you intend to support (HTTP, TCP/IP, MSMQ, Named Pipes, and so on). Figure 1.39 shows the WCF configuration editor. Notice how the binding support for an endpoint is a configuration (and not coding) task.

NOTE

Work with Data

Data is the domain of the business developer. It makes sense then that the number one tool and framework for business development continues to provide new and better ways of accessing and exposing that data. Data access is everywhere inside Visual Studio and the .NET Framework. Here we highlight some of the new things you will encounter in the 2008 version.

NOTE
Data access is covered in more detail in Chapter 18.
Design Your Data

A typed dataset allows you to create a .NET class that is based on your database table semantics but works as a dataset behind the scenes. Typed datasets have been part of Visual Studio for a while now. You can auto-generate them based on table schemas and edit the code used to query, update, insert, and delete data.

Visual Studio 2008 provides additional design-time support for typed datasets. You can create a typed dataset file (.xsd) and use the Toolbox and Server Explorer to build the dataset. Figure 1.40 shows an example of this. Tables from the Server Explorer were added to the design surface to build the dataset.

In addition, the typed dataset in 2008 now supports hierarchical updates. This feature allows you to save data in multiple, related tables through the typed dataset. In Figure 1.40, this would mean you could edit both the employee details and the related territories and update as a single process.

Map Objects to Relational Data

Most business applications rely on data that is stored in relational databases. Databases have a well-known structure of tables, columns, and relationships. This structure makes for a nice storage and reporting mechanism. However, there is often a mismatch between the needs of an object-oriented, .NET developer and the database technology. The object developer thinks in terms of objects, properties, methods, relationships, encapsulation, and the like. A lot of time is spent converting data from a database into an object-oriented structure and back again. In addition, developers are asked to write code in Structured Query Language (SQL) and .NET.
LINQ, which we introduce in Chapter 3, is a technology that helps bridge this gap. It provides a means to work with database records as objects. It also allows you to write your queries using .NET (C# and VB). On top of LINQ is an object relational (O/R) designer for SQL Server called LINQ to SQL. With it, you can design and build .NET classes based on database schemas. In this way, you can quickly take database concepts and convert them to object-oriented ones.

You use a Visual Studio template to create a LINQ to SQL class. You can add such a class to any project type (web, Windows, class library, and so on). Figure 1.41 shows the Add New Item dialog with the LINQ to SQL Classes template selected.

You can then use the O/R Designer to build your LINQ class. You first use Server Explorer to open a connection to the database with which you are working. You can then drag and drop tables from the database onto the design surface. The designer understands (and enforces) the table relationships. It maps tables to classes and columns to properties. Figure 1.42 shows an example of the designer. Here there is a customer class with an association link to the customer’s orders. Notice also the methods listed on the right side of the diagram. These were created by dragging stored procedures from Server Explorer to the designer. A developer may now work with these classes and methods in a similar manner as they would with other .NET objects.
FIGURE 1.41 Creating a new LINQ to SQL class.

FIGURE 1.42 Mapping database relationships to objects.
Build Occasionally Connected Applications

Many applications written these days require access to data both on- and offline. For example, you might need report data or customer service records when traveling or at a customer site. You cannot always rely on being connected. Developers have been dealing with this issue in various ways. The latest version of Visual Studio provides data synchronization services to provide a common, easy solution to this difficult problem.

Data synchronization allows you to synch and cache data between a central database and a user’s client system. The cache is a version of SQL Server Compact Edition (CE). Microsoft is already taking advantage of these services in Vista, Office, and even their Zune software. You too can take advantage of these services in your applications.

Looking back at Figure 1.41, you will notice a template called Local Database.... This template is actually called Local Database Cache. With it, you create a .sync file to configure how synchronization should happen between a local data store and a server. Opening the local database cache file in Visual Studio opens the Configure Data Synchronization dialog. Here you can set up tables that should be cached on a client, set up the server database, and configure the client database information. Figure 1.43 shows an example of this dialog.

![Configure Data Synchronization Dialog](image)

FIGURE 1.43 Configuring data synchronization.

The Cached Tables area of the Configure Data Synchronization dialog allows you to define which tables will be synchronized. Figure 1.44 shows an example of adding tables to this...
list. Here you configure how synchronization happens for each table. Synchronization services are discussed further in Chapter 18.

![Configure Tables for Offline Use](image)

**FIGURE 1.44** Adding and configuring table synchronization.

### The Visual Studio Product Line

Like the 2005 version, Visual Studio 2008 comes in many flavors; each is baked for a different appetite. There is a recipe targeted at the hobbyist to the enterprise architect, the beta tester to the operations guy—and, oh yeah, there are morsels for the developer too! Sorting through all the Visual Studio products and editions can be confusing. We hope the following aids you in your Visual Studio flavor selection.

#### Express Editions

Microsoft offers the Visual Studio Express Editions on a per-language basis (VB, C#, C++, Web, SQL). These editions are free, downloadable, low-barrier to entry versions targeted directly at the novice, hobbyist, student, or anyone else looking to write some code without breaking the bank. There is even an edition targeted at video game developers. Along with the editions are tutorials, videos, sites, fun projects, and more.

The Express Edition can be seen as Microsoft’s answer to all the “freeware” tools available to today’s developers. After all, if you are a college student looking to put up a buddy’s website, you are more likely to look for the low-cost solution. Of course, five years down
the road when you're making decisions for your company, Microsoft wants to be sure you’ve had a chance to work with its products. The Expression Edition fits this niche nicely.

These editions purposely do not have all the power of their professional patriarch (a Class Designer, unit testing, enterprise templates, XSLT support, source code control, 64-bit support, and so on). However, they do have full language support (like LINQ) and access to the .NET Framework items like WPF.

The Express Editions also have a more streamlined user experience that does not expose the full complexity (or power) of the professional editions. However, developers will be able to create client/server form-based applications, websites, and even web services using these tools.

**NOTE**

For more information regarding the Visual Studio Express Editions or to download one of these editions, you can visit Microsoft’s site at the following URL: http://www.microsoft.com/express/default.aspx.

**Standard Edition**

The Standard version of Visual Studio is the base-level entry point for professional developers. This edition is similar in nature to the Express Editions. However, it contains all the .NET languages in a single package. In addition, it gives developers an expanded feature set over the Express versions. These additional capabilities include the following:

- Multiproject solution support
- Multitargeting support for working with .NET 2.0, 3.0, and 3.5
- Design support for creating Web, Windows, WPF, and Ajax User Experiences
- Database design tools for working with databases beyond SQL Express Edition
- Windows Communication Foundation (WCF) and Windows Workflow (WF) support
- Visual modeling via the visual Class Designer
- Support for XML editing including XSLT
- Click-once deployment tool
- Capability to write, record, and run macros
- Support for creating and using Visual Studio add-ins
- Conversion Wizard for converting legacy projects to 2008
- SQL Server 2005 Express included
- Compatibility with Visual Source Safe (VSS)
**Professional Edition**

Most corporate developers and consultants will find a home within one of the professional editions of Visual Studio. In fact, you may already have a version of Visual Studio depending on what license you bought and when. Visual Studio Professional gives you all the language support (including VB, C#, C++), everything in the Standard Edition, and, of course, the whole host of new enhancements we’ve discussed in this chapter.

The primary differences between standard and professional (besides the MSDN packaging) is a set of features that you do not get in the standard edition. The following features ship only with the professional version (and above):

- Visual Studio Tools for Office (VSTO)
- SQL Server 2005 Developer Edition
- Capability to build software for mobile devices
- Class Designer and Object Test Bench
- Crystal Reports
- Unit Testing (no longer just a Team Edition feature)
- Server Explorer

Outside of these additional features, the Professional Edition is packaged as part of an MSDN subscription. Visual Studio Professional is packaged with various MSDN flavors. This change started with the 2005 version. You can now buy Visual Studio Professional as follows (listed from the fewest features/applications to the most):

- **Visual Studio Professional only (without MSDN)**—Includes just Visual Studio Professional as discussed.

- **Visual Studio Professional with MSDN Professional**—Includes Visual Studio as discussed. In addition, you get access to older versions of Visual Studio. You also get Visual Source Safe. The MSDN subscription provides developer licenses for Windows and Window Server, Virtual PC, SDKs, driver development kits, and more. You also get two technical support incidents and access to managed newsgroups.

- **Visual Studio Professional with MSDN Premium**—Includes all the preceding with the following extras: includes Expression Web and Blend; adds developer licenses for SQL Server and other application platforms (BizTalk, Commerce Server, and many more); provides licenses for working with Microsoft Dynamics (GP, CRM, POS, and so on); includes licenses for Microsoft Office Systems 2007 (including Visio, Groove, Word, Excel, and many more).

**NOTE**

For more information regarding the professional editions of Visual Studio, you can visit Microsoft’s site at the following URL: http://msdn2.microsoft.com/en-us/subscriptions/aa718657.aspx.
Team Systems

Team Systems is a set of integrated, software development life-cycle tools. Microsoft released the first versions of its Team Systems products in 2005. These included versions of Visual Studio targeted at various members of the software development life cycle. In the middle is a centralized management and reporting server called Team Foundation Server. By all accounts, these tools were a great success. Microsoft has built on this success with the release of Visual Studio Team Systems 2008.

The Client Tools

The Visual Studio Team System editions are targeted to roles within the software development life cycle. These roles include developer, architect, tester, and database developer. In addition, there are client access licenses available for project managers and other interested parties. However, the aforementioned roles each represent versions of Team Systems that can be purchased.

It’s important to note that each role-based version of Team Systems comes with Visual Studio 2008 Professional. It also comes with MSDN Premium (as defined previously). The Team System client tools also give you access to the central Team Foundation Server (purchased separately). What makes each of these products unique and significant, however, is what additional goodies they do and do not contain. The intent is a targeted set of tools to different roles on the project. The following list outlines the features that drive these products to their target users:

- **Visual Studio Team System Development Edition**—Targeted at most developers, this version includes static code analysis, code profiling, dynamic code analysis, code metrics, code analysis check-in policies, unit testing, and code coverage analysis. These tools help developers with verifying, testing, and checking their code against common issues.

- **Visual Studio Team System Database Edition**—Targeted at developers who need to work closely with and manage database development, this edition enables you to create database projects, generate sample data, compare schemas, compare data, do database-level unit testing, and more.

- **Visual Studio Team System Architect Edition**—Designed for the software architect, this product improves design and design validation of distributed systems. Features include the System Designer, Application Designer, Logical Datacenter Designer, Deployment Designer, and a Settings and Constraints Editor.

- **Visual Studio Team System Test Edition**—Targeted at the software tester, this edition includes the capability to create unit tests and see code coverage analysis. Its real strength for testers, however, is its capability to manage and create web, load, manual, generic, and ordered tests. In addition, there is the Team System 2008 Test Load Agent. This is a separate product that works with Team Test to generate massive load for various load-testing scenarios.

- **Visual Studio Team Suite**—For those who must have it all (and have unlimited budgets), this product is the everything-but-the-kitchen-sink option. It includes all
the features of Team Architect, Developer, Database Developer, and Test in a single package. Microsoft also understands that there are those among our ranks who can’t stand to not have it all. For them, Microsoft has created Team Suite—the full IDE experience that transcends all team roles.

NOTE
We cover Team Systems features in great depth in Part VI of this book, “Visual Studio Team System.”

The Server
Team Foundation Server (TFS) is at the center of Team Systems. While the client tools enable great functionality, the server allows you to assign work, report statistics, and track the overall health of your project. Project information is synchronized among architects, developers, testers, project managers, and operations.

The functionality behind Team Foundation Server revolves around project management and source control. Project management and tracking are accomplished through work items. A work item can be a task on the project, an issue or bug, a software requirement, a feature, or a test scenario. In general, a work item represents a generic unit of work on the project. Of course, work items are customizable and can have states, new fields, and business rules associated with them. Work items can also be driven by a methodology. Finally, work items play a central part in ensuring project team communication and reporting.

The source control features in Team Foundation Server include enterprise-class features such as change sets, shelving, automatic build rules, the capability to associate work items to changed source, parallel development, a source control policy engine, branching, checkpoints, and more.

Surrounding these project management and source control features are a build management engine, a reporting infrastructure, and a project portal. The build tools allow for both automatic, scheduled builds and on-demand builds. Builds are reported against, documented, automatically tested, and analyzed for code coverage and churn (as an example). The reporting engine and project portal combine to further enhance the view into the project by team members. Built on Windows SharePoint Services (WSS), it delivers the latest test and build reports, documentation, announcements, and quality analysis.

NOTE
You purchase TFS as you would any other Microsoft Server product (such as Microsoft SQL Server or Microsoft BizTalk Server). For more information on licensing TFS, see http://msdn2.microsoft.com/en-us/vsts2008/products/bb933758.aspx.

The Expression Tools
A sister product line to Visual Studio is the new Microsoft Expression tools. These will, undoubtedly, get a lot of attention in upcoming releases. The tools are targeted at designers’ build applications on the Microsoft platform. They offer rich design experiences for
building Web, Windows, and Silverlight applications. The tools are also built to enable workflow between designers and developers.

It is important that you are aware of these tools so that you know where you might use them and because they work with Visual Studio projects and project files and offer some similar capabilities (but in different ways). A high-level overview of these tools is listed here:

- **Expression Blend**—Used to create WPF interfaces based on XAML. You can also use it to create Silverlight applications.
- **Expression Design**—Allows a designer to create vector-based illustrations that include drawing, text, and more.
- **Expression Web**—A design tool for creating ASP.NET Web Forms and websites.
- **Expression Encoder**—A tool for encoding video and audio and publishing the same to Silverlight applications.
- **Expression Media**—A tool for organizing and managing design assets (files) into visual catalogs.
- **Expression Studio**—Includes the full set of Expression tools. For those designers who need it all.

**NOTE**

For more information on the Expression products (not covered further in this book), see http://www.microsoft.com/expression/default.aspx.

**Summary**

A new release of Visual Studio means a lot to all the various development camps out there. Visual Studio touches developers who write code in C++, C#, Visual Basic, and many other languages. Literally millions of developers boot up and launch their favorite tool every day. They spend the vast majority of their working hours, days, weeks, and months architecting and building solutions with the tool. This chapter oriented you to some of the many new possibilities available with this latest version.
A

About Box dialog box, 445
absolute positioning, 201, 528-532
accessibility
  options for website projects, 516
  standards compliance (HTML designer), 205
Accessibility Validation dialog box, 205
Accessibility Validation section (build options for website projects), 515
accessing
  cache, 873-874
  code within projects, 374-375
  windows, 377-378
Accordion control, 663
actions panes, 855
  creating, 870-871
  stacking behavior, 871-872
active hyperlinking (productivity tool), 255
activities (workflows), 809-811
  CallExternalMethod activities, 816-818
  CodeActivity activity, 818
  HandleExternalEvent activities, 821-823
  IfElse activities, 819-820
  InvokeWebService activity, 820-821
adaptive rendering, 562
Add Connection dialog box, 698
Add-in Manager, 460
Add-in projects (Office), 34
Add-in Wizard, 442-443
  About Box dialog box text, 445
  application host selection, 443
  code generation, 446-452
  language selection, 443
  name and description of add-in, 444
  options, setting, 444
add-ins
Add-in Wizard, 442-443
   About Box dialog box text, 445
   application host selection, 443
   code generation, 446-452
   language selection, 443
   name and description of add-in, 444
   options, setting, 444
   color palette example, 461
   code generation properties, 463
   code listing for, 473-486
   color selection event handler, 466
   creating Options page, 469-473
   creating user control, 462-466
   displaying tool window, 467-468
   helper routines, 464-466
   inserting generated code, 469
   integrating user control, 466-469
   mouse movement event handlers, 462-463
   user control event handling, 468-469
   defined, 442
   IDTExtensibility2 interface methods, 452-458
   loading, 461
   managing, 460
   .NET Framework 3.5 enhancements, 98
   Office add-ins, creating
      Outlook Form Regions, 865-867
      overview, 859
   Ribbon customization, 860-863
   task pane customization, 863-864
   purpose of, 441
   reacting to commands, 458-459
   registered add-ins, 461
   shared add-ins, 444
   as shared content, 229
   structure of, 452
Add Link dialog box, 963
Add New Project dialog box, 720
Add New Test dialog box, 1137
Add New User Control dialog box, 206
Add Reference dialog box, 513
Add Service Reference dialog box, 773, 796
Add Style Rule dialog box, 187, 536
Add to Source Control dialog box, 931
Add Workspace dialog box, 928
.addin files, 461
adding
   applications to application diagram, 1055
   comments to text windows, 408
   items to class diagrams, 998
   servers to zones, 1065-1066
addresses, 781
adds (source control), 931
administrator group, 923
ADO.NET, Sync Services for ADO.NET, 100
Advanced Properties property page, 1023
AdventureWorks database, 708
agents
   build agents, 1155-1156, 1162
   defined, 890
agile development. See MSF Agile
AJAX
   Accordian control, 663
   adding controls to pages, 662, 666-668
   AJAX Control Toolkit, 660-662
   ASP.NET AJAX controls, 656
   AutoCompleteExtender control, 664
   CalendarExtender control, 664
   CollapsiblePanelExtender control, 663
   ConfirmButtonExtender control, 665
   FilteredTextBoxExtender control, 664
   HoverMenuExtender control, 664
   ListSearchExtender control, 664
   MaskedTextBoxExtender control, 664
   ModalPopupExtender control, 664
  overview, 653-654
   PasswordStrength control, 665
   ReorderList control, 665
   ScriptManager control, 655
   ScriptManagerProxy control, 655
SliderExtender control, 665
TabContainer control, 665
Timer control, 656
UpdatePanel control, 655
UpdateProgress control, 656, 659-660
AJAX Control Toolkit, 660-662. See also AJAX
AJAX Extensions for Visual Studio, 19-20
alerts
managing in MSDN Community Forums, 221-223
in Team Foundation Server, 917
alias, 268
alignment of form controls, 586-590
Allocation subreport, 1032
alpha releases, 1079
analyzing builds, 1167-1171
Anchor property (form controls), 587-588
anchoring controls, 587-588
anonymous types, 88-89
App_Browsers folder (website projects), 510
App_Code folder (website projects), 510
App_Data folder (website projects), 510
App_GlobalResources folder (website projects), 510
App_LocalResources folder (website projects), 510
App_Themes folder (website projects), 510
App_WebReferences folder (website projects), 510
appearance properties
controls, 594-595
forms, 583
application architects, 1048
Application Designer, 1050-1056
application implementation, 1074-1077
application settings and constraints, 1057-1058
Diagram menu, 1051-1053
Toolbox, 1053-1054
application diagrams, 1047-1056
Application Designer Toolbox, 1053-1054
application settings and constraints, 1057-1058
Diagram menu, 1051-1053
generating models, 1051
item templates for, 1049
application hosts, selecting for add-ins, 443
Application property page, 129
application tier (Team Foundation Server), 896-898, 1154-1156
Application.EnableVisualStyles method (form controls), 594
applications. See also projects
adding to application diagram, 1055
connecting in application diagram, 1055-1056
connecting to systems, 1061-1062
debugging. See debugging design, 1050-1051
implementation, 1074-1077
Office. See Office applications settings, 131
Apply Styles dialog box, 25, 539
applying
styles, 539
themes, 547-550
architect role
application architects, 1048
infrastructure architects, 1048
Visual Studio Team Architect, 887
application design steps, 1050-1051
application diagrams, 1051-1058
application implementation, 1074-1077
deployment diagrams, 1071-1074
item templates, 1049-1050
logical datacenter diagrams, 1062-1071
project templates, 1048-1049
system diagrams, 1059-1062
architecture of Team Foundation Build, 1154-1157
areas
team projects, 910-912
work items, 956-958
Areas and Iterations dialog box, 910-912
arguments, removing from event handlers, 97
.asax file extension, 511
.ascx file extension, 511
.asmx file extension, 511, 755-756
ASP.NET, 17
controls. See controls
designers, 518
Development Server, 503
event model, 518, 523-526
Futures, 681
Membership, 567
.NET Framework 3.5 enhancements, 98
Reports Web Site template, 502
user interfaces
  consistency, maintaining, 532-550
  creating user-configurable UIs, 550-560
  master pages, 541-544
  page layout, 528-531
  planning, 526-528
  style sheets, 533-541
  themes, 544-550
Web Service template, 501
web services, 754
Web Site template, 501
website projects
  accessibility options, 516
  build options, 513-515
  creating, 500
  creating web pages, 518-526
  file types for, 510-511
  folders for, 508-510
  MSBuild options, 518
  Property Pages dialog box, 512-518
  references, 512-513
  selecting default programming language, 507
  selecting template, 501-502
  selecting website location, 502-507
  start options, 517
.aspx file extension, 511
assemblies, 95, 802
assertion methods, 1007-1008
associating
  code with work items, 981-983
  icons with wizards, 490
  work items with check-ins, 938-939
association, 1001-1002
at-rest state (Debug menu), 328-329, 598
attaching
  files to work items, 965
to processes, 323-324
to WCF services, 358
attribute classes, 1007
Auto Format dialog box, 554
auto-generating
  bound controls, 725-731
  SQL scripts, 718-719
auto scaling controls, 589-590
AutoCompleteExtender control, 664
automation object model, 365-366
categories, list of, 368
Command object
  executing commands, 413-414
  mapping key bindings, 414
  properties/methods, 412-413
CommandBar object, 394-397
Debugger object, 415
Document object, 398
  properties/methods, 399
text documents, 400-403, 406-408
DTE/DTE2 root object, 369
EnvDTE/EnvDTE80 assemblies, 366
event handling, 416
Project object
  code access, 374-375
  properties/methods, 373
  solution/project hierarchy, 370-372
Solution object
  properties/methods, 372
  solution/project hierarchy, 370-372
Window object
  command window, 389
  linked windows, 392-393
  output window, 390-391
  properties/methods, 378-380
  querying Windows collection, 380-381
  referencing windows, 377-378
  task list window, 384
text windows, 381-382
tool windows, 383-384
Toolbox, 386-387
Autos window (debugger), 351
AutoScaleDimensions property (form controls), 590
AutoScaleMode property (form controls), 590
AutoSize property (form design), 579

B

BackColor property (forms), 583
behaviors, 781
best practices, 1006-1007
beta releases, 1079
Bin folder (website projects), 510
Binary Properties property page, 1019
binding. See also data binding
defined, 782
extracted values to web test requests, 1097
query strings to data, 1094-1096
tests to data, 1092-1094
bindings. See key bindings
bookmarks, 166-170
Bookmarks window, 169-170
brace matching (productivity tool), 277-279
Branch dialog box, 946
branching source code tree, 944-946
Break All command, 337
breaking execution for exceptions, 320-321
Breakpoint Condition dialog box, 344-345
Breakpoint Filter dialog box, 346
Breakpoint Hit Count dialog box, 347
breakpoints, 340. See also Run To Cursor command; tracepoints
Breakpoints window, 341
clearing, 327
conditions, 343-345
configuring, 180, 325
controlling flow of code, 180-181
defined, 178
on file locations, 346
filters, 346
hit count, 347
icons for, 341
indicator margin, 166
setting, 179-180, 324, 340
Breakpoints window, 341
browsers
defined, 135
Object Browser. See Object Browser output for web form applications, 204-205
selecting for load tests, 1104
targeting, 204
bubbling events (WPF), 637
bug reports, sending via Help menu, 223-224
bug work items
MSF for Agile, 953
MSF for CMMI, 955
build agents, 1155-1156, 1162
build and deploy options (database projects), 1121-1122
Build Configuration property page, 116
Build Events property page, 130
Build menu, 62
Build property page, 130
build servers, 1155-1156
Build Solution Action section (build options for website projects), 515
build triggers, 1163-1164
building database projects, 1149-1150
builds, 1153. See also Team Foundation Build
build reports, 1170-1171
build repository, 1155-1157
automation, 893
configuration, 116, 513-515, 1160
creating
configuring build agents, 1162
defining build project file, 1159, 1162
defining build retention policy, 1162
naming builds, 1158
scheduling the build, 1163-1164
selecting project files, 1158
definitions, 1158
editing
build project file, 1165
MSBuild, role of, 1166
events, 130
extending with custom tasks, 1166

How can we make this index more useful? Email us at indexes@samspublishing.com
monitoring/analyzing, 1167-1171
naming, 1158
quality states, 1168-1169
running, 1166
scheduling, 1163-1164
tests, 1161
business analysts, 979-980
business logic, 91-93
business process development
  service-oriented development, 41-42
  workflows, 38-40
Button control, 564
Button Editor, 437
buttons. See toolbar buttons

C

C#
  code editor, 70-71
  Code Snippet Inserter, 268
  creating stored procedures, 720-723
  IDE enhancements, 81-82
  language enhancements, 82-83
    anonymous types, 88-89
    auto-implemented properties, 97
    business logic in partial methods, 91-93
    embedded XML, 96-97
    extension methods, 87-88
    friend assemblies, 95
    implicit typing, 83-85
    lambda expressions, 89-91
    LINQ queries, 94-95
    object initializers, 85-87
    refactoring tools. See refactoring tools
C++ applications, Class Designer support for, 16
  cache
    accessing, 873-874
    Office applications, 856
    storing data in, 872-873
Calendar control, 565
CalendarExtender control, 664
Call Stack (debugger), 322
Call Tree subreport, 1032
Caller/Callee subreport, 1032
CallExternalMethod activities, 810, 816-818
calling web services, 777-778
Canvas control (WPF), 624-626
Capability Maturity Model for Integration (CMMI), 883
cascading style sheets. See CSS editor
CatalogZone control, 553
cells, DataGridView control, 606
change request work items (MSF for CMMI), 954-955
change scripts, 719
change tracking (productivity tool), 253-254
ChangePassword control, 569
changesets in source control, 939-940
channels, 782, 933
check constraints, 701
Check In dialog box, 933
check-in notes, 937-938
check-in policies, 934-936
  configuring rules for, 1036
    in Team Foundation Server, 892
Check Out command (source control), 932
CheckBox control, 565
checking in files to source control, 933-939
checkpoints
  MSF Agile, 883
  MSF for CMMI, 884
Choose Columns dialog box, 974
Choose Location dialog box, 503
Choose Related Work Item dialog box, 963
Choose Toolbox Items dialog box, 585
choosing. See selecting
chords, 294
class coupling, 889
Class Designer, 16
class diagrams
  adding items, 998
  creating, 996-997
  defining
    methods/properties/fields/events, 1002-1004
  displaying members, 997
  relationships between classes, 999-1002
  refactoring with, 290
class diagrams, 1048
  adding items, 998
  creating, 996-997, 1049
  defining methods/properties/fields/events, 1002-1004
  displaying members, 997
  relationships between classes, 999-1002
class library (WPF), 615-617
Class View
  members pane, 146
  objects pane, 144-145
  Rename refactoring tool, 293
  search bar, 144
  toolbar buttons, 142
classes
  class diagrams, 1048
  adding items, 998
  creating, 996-997, 1049
  defining methods/properties/fields/events, 1002-1004
  displaying members, 997
  relationships between classes, 999-1002
  Class View
    members pane, 146
    objects pane, 144-145
    Rename refactoring tool, 293
    search bar, 144
    toolbar buttons, 142
coupling, 889
message classes, 865
SmartTagBase, 874
templates, 492
clearing breakpoints, 327
clickable hyperlinks. See active hyperlinking (productivity tool)
ClickOnce application deployment, 99
client-side scripts, debugging, 361
clients
  client event model, 562
  Team Build client, 1154-1155
  Team Systems products, 50-51
workflows, 803
  defining for sequential workflows, 824-832
  defining in state machine workflows, 844-849
WPF (Windows Presentation Foundation), 27-32
CMMI (Capability Maturity Model for Integration), 883. See also MSF for CMMI
code. See also code editor
  accessing within projects, 374-375
  associating with work items, 981-983
  code lines, counting, 889
  Code Metrics, 1041
    measurements in, 1042
    running, 1042
    viewing results, 1042-1043
  in Visual Studio Team Developer, 889
code stepping. See stepping through code conditions, 819
debugging. See debugging
  formatting in C#, 81
  profiling in Visual Studio Team Developer, 888
  sharing. See shared content
  validating with Visual Studio Team Developer, 888-890
  writing in Windows Forms designer, 193-196
Code Analysis Policy, 934
Code Analyzer, 1034
  configuring rules to enforce, 1035-1036
  prescriptive remediation, 1041
  rule violations, handling, 1036-1037
  running, 1039
  suppressing rules, 1037-1038
  viewing results, 1039
code coverage in Visual Studio Team Developer, 889
code coverage analysis, 1014-1015
code definition window, 183
code editor, 70-71. See also text editor
  code definition window, 183
  customizing, 72

How can we make this index more useful? Email us at indexes@samspublishing.com
debugging in, 178
  configuring breakpoints, 180
  controlling flow of code, 180-181
  setting breakpoints, 179-180
navigation tools
  bookmarks, 168-170
  line numbering, 168
opening, 163
printing from, 181-183
productivity tools
  code outlining, 257-259
  HTML navigation, 259-260
  smart tags, 262
  selection margin, 167
  smart tags, refactoring with, 287-289
  toolbar, 168
windows
  components of, 165-167
  selecting, 163
  writing code in, 164-165

Code Metrics, 1041
  measurements in, 1042
  running, 1042
  viewing results, 1042-1043
  in Visual Studio Team Developer, 889
Code Metrics Results window, 1043
code outlining (productivity tool), 257-259
code pane (code editor), 166
Code Snipper Inserter, 268-270
code snippets (productivity tool), 229, 268-277
Code Snippets Manager, 276
code stepping. See stepping through code
code window, search-and-replace modes, 170
  Find In Files mode, 173-176
  Find Symbol mode, 176-177
  Incremental Search mode, 177-178
  Quick Find mode, 171-172
  Quick Replace mode, 172-173
  Replace In Files mode, 176
CodeActivity activity, 810, 818
CodeElements collection, 374-375
CodePlex, 231

Codezone community, 227
coding problem indicators (productivity tool), 254-255
collaboration. See VSTS (Visual Studio Team System)
collaboration features (XAML), 618-619
CollapsiblePanelExtender control, 663
color palette example (add-ins), 461
  code generation properties, 463
  code listing for, 473-486
  color selection event handler, 466
  creating Options page, 469-473
  creating user control, 462-466
  displaying tool window, 467-468
  helper routines, 464-466
  inserting generated code, 469
  integrating user control, 466-469
  mouse movement event handlers, 462-463
  user control event handling, 468-469
colors
  in code coverage analysis, 1015
  color palette example (add-ins), 461
    code generation properties, 463
    code listing for, 473-486
    color selection event handler, 466
    creating Options page, 469-473
    creating user control, 462-466
    displaying tool window, 467-468
    helper routines, 464-466
    inserting generated code, 469
    integrating user control, 466-469
    mouse movement event handlers, 462-463
    user control event handling, 468-469
  printer settings, 181-182
  syntax coloring (productivity tool), 255-256
Column Details window, 1146
Column Options dialog box, 968
columns
  in performance reports, 1024-1031
  selecting text, 160
Columns property (indexes), 701
command bars, 394-397
command keys
  Code Snippet Inserter, 270
  Complete Word, 264
  Encapsulate Field refactoring tool, 313
  Extract Interface refactoring tool, 305
  Extract Method refactoring tool, 296
  Incremental Search mode, 178
  jumping to code lines, 168
  mapping key bindings, 414
  Promote Local Variable to Parameter refactoring tool, 309
  Remove Parameters refactoring tool, 308
  Rename refactoring tool, 294
  Reorder Parameters refactoring tool, 311
  running macros from, 437-438
  smart tags, 289
  Toolbox (Windows Forms designer), 190
Command object
  executing commands, 413-414
  mapping key bindings, 414
  properties/methods, 412-413
command window, 389, 439-440
CommandBar object, 394-397
commands, 412-413. See also command keys
  add-ins reacting to, 458-459
  executing, 413-414
  mapping key bindings, 414
CommandWindow object, 389
comments
  in add-in code, 457
  comment tasks, 281-282
  inserting into text windows, 408
community features (Visual Studio 2008)
  code sharing features, 16
Help menu, 217
  Customer Feedback Options, 223-224
  MSDN Forums button, 217-223
  Search option, 224-225, 228
Start Page, 212-213
  Getting Started section, 213-214
  news channel, 214-217
  Recent Projects section, 213
  Visual Studio Headlines section, 214
community forums. See MSDN Community Forums
Community menu. See Help menu
Compact Framework 3.5, 38
CompareValidator control, 567
collection
  with Data Comparison tool, 1131-1132
  data update script, 1134
  viewing record-level details, 1132
  in joins, 709
  performance reports, 1033
  with Schema Comparison tool, 1126-1128
  schema update script, 1129-1130
  setting options, 1130
  viewing object definitions, 1128
Compile property page, 130
compiler, 254-255
Complete Word (productivity tool), 264
complex data binding, 725
component designer
  creating components, 206-207
  viewing, 525
  writing code, 207-208
component tray, 190, 585
components
  creating, 206-207
  defined, 190, 206, 584
  writing code for, 207-208
composite controls, 607. See user controls
compressing files, 244
conditions for breakpoints, 343-345
.config file extension, 511
configurable UIs, 550-560
configuration file for testing, 1086
configuring
  breakpoints, 180
  build agents, 1162
  build and deploy options (database projects), 1121-1122
  build configurations, 1160
  CallExternalMethod activities, 816-818
  code coverage analysis, 1014
  CodeActivity activity, 818

How can we make this index more useful? Email us at indexes@samspublishing.com
data generators, 1146
HandleExternalEvent activities, 821-823
IDE, 54-56
IfElse activities, 819-820
InvokeWebService activity, 820-821
load tests, 1106
performance sessions, 1018-1023
rules for Code Analyzer, 1035-1036
Test Project template, 1080
Visual Studio for source control, 925
WCF services, 790-796
zones and servers, 1068-1071
confirmation of settings, 904
ConfirmButtonExtender control, 665
Connect class, 466-469
connecting
applications in application diagram, 1055-1056
applications to systems, 1061-1062
database connections, 698-700
servers to zones, 1067-1068
to Team Foundation Server, 912-913
consistency for user interfaces, 532-550
constant load pattern (load tests), 1101
consuming
WCF services, 796-798
web services, 772
calling web services, 777-778
defining service references, 773-775
updating web references, 776
viewing web references, 776
containers, 590-594
Content Installer. See Visual Studio Content Installer
content pages, creating from master pages, 542-543
Content Type searches, 229
Continue command, 339
continuing executing after breakpoints, 325
contracts, 781
contributors
source control security, 923
Team Foundation Server security, 900

Control_Specific_Event(s) event, 524
controls, 561. See also names of specific controls
adding, 200-201
to forms, 189-190
to pages, 520-521, 662, 666-668
appearance properties, 594-595
arranging, 191-193, 201
ASP.NET AJAX controls, 656
bound controls
auto-generating, 725-731
manual binding, 732-735
changing type of, 601
composite controls, 607
containers, 590-594
creating, 206-207, 607-611
custom controls, 575, 610-611
data binding, 635-636
data controls, 573-574, 736-740
data presentation, 603-606
data source controls, 740
defined, 206, 584
event handlers, 526
features of, 562-563
host controls, 868-870
HTML controls, 562
layout panel controls (WPF)
Canvas control, 624-626
DockPanel control, 626-627
Grid control, 627-630
image viewer application example, 639-641
StackPanel control, 630-631
login controls, 567-569
non-visual controls, hiding, 552
positioning, 528-531, 586-590
resizing, 193, 580
as server controls, 561
as shared content, 229
site navigation controls, 571-572
skins, 546-547
standard controls, 563-565
subclassing, 607
data sources, 1183

CustomValidator control, 567
cycles (MSF Agile), 883
cyclomatic complexity, 889

data access, 42
  mapping objects to relational data, 43-44
  synchronization, 46-47
  typed datasets, 43

data binding, 724-725
  ASP.NET controls, 562
  editing typed DataSets, 731-732
  form controls
    auto-generating bound controls, 725-731
    manually bound controls, 732-735
  web controls, 736-740
  in WPF, 635-636, 641-645
data cache
  accessing, 873-874
  Office applications, 856
  storing data in, 872-873
data changes in Visual Studio Team System
  Database Edition, 892
Data Comparison tool, 1131-1132
  data update script, 1134
  viewing record-level details, 1132
Data Connections node, 148
data controls, 573-574, 736-740. See also names of specific controls
data entry, 563
Data Explorer. See Server Explorer
data generation, 1142-1149
  creating data generation plan, 1142-1144
  previewing generated data, 1144-1148
data generators, configuring, 1146
Data menu, 62
data presentation controls, 603-606
data sources
  data source controls, 574, 740
  DataGridView control, 606
  mapping to controls, 728-731

How can we make this index more useful? Email us at indexes@samspublishing.com
seeding web tests, 1089-1096
selecting, 726-728
data tier (Team Foundation Server), 898-899, 1155-1157
data types, 83-85
data update script, 1134
database administrator role, 890-892
Database Diagram Designer, 701-703, 705-706
database diagrams
creating, 702
table relationships, 703-706
Database Explorer. See Server Explorer
database objects, creating in managed code, 719-723
database projects, 716
building and deploying, 1149-1150
creating, 716-718, 1117-1122
Data Comparison tool, 1131-1134
generating data, 1142-1149
creating data generation plan, 1142-1144
previewing generated data, 1144-1148
in Visual Studio Team System Database Edition, 891
organizing, 1118
rename refactoring, 1135-1136
Schema Comparison tool, 1126-1130
Schema View window, 1124
SQL scripts, 718-719
unit tests, 1137-1140
workflow of, 1116-1117
schemas
deploying, 1149-1150
importing, 1119-1120
SQL statements
query types, 709-710
query writing, 706-709
stored procedures, 710-713
triggers, creating, 714-715
user-defined functions, creating, 715-716
views, creating, 710
tables
adding to database diagram, 702
defined, 697
defining, 700-701
editing definitions, 703
relationships, building, 703-706
showing contents of, 709
viewing, 702
TfsWarehouse relational database, 918
DataGridView control, 605-606, 732-734
DataList control, 574, 736
DataPager control, 21, 574
DataSet Designer, 731-732
DataSets, 731-732
DataTips window (debugger), 352-353
.dbml file extension, 511
de facto standards (form design), 579
Debug Location toolbar, 360
Debug menu, 62, 328-333, 598
debug mode, 317-320, 330-333
Debug Options dialog box, 333-334
Debug property page, 131
Debug Source Files property page, 115-116
Debug toolbar, 330
debugger, 131. See also debugging
breakpoints
  Breakpoints window, 341
clearing, 327
  conditions, setting, 343-345
  continuing execution after, 325
  filters, setting, 346
  hit count, setting, 347
  icons for, 341
  on file locations, 346
  setting, 324, 340
Debug menu, 328-333, 598
Debug Options dialog box, 333-334
Debug toolbar, 330
Edit and Continue feature, 355-356
pointing at source files, 115-116
stepping through code, 325-327
  Break All command, 337
  Continue command, 339
  ending session, 339
  Run To Cursor command, 335-336
  Start Debugging command, 336
  Step Into command, 334-338
  Step Out command, 338-339
  Step Over command, 338
tracepoints, 340, 348-349
viewing data
  Autos window, 351
  DataTips window, 352-353
  Locals window, 350
  QuickWatch window, 352
  visualizers, 354-355
  Watch windows, 351-352
windows in, 322
Debugger class, 180, 415
Debuggers section (start options of website projects), 517
debugging, 316. See also debugger; testing
  attaching to web service process, 323-324
  breaking execution for exceptions, 320-321
  client-side scripts, 361
  in code editor, 178
    configuring breakpoints, 180
    controlling flow of code, 180-181
    setting breakpoints, 179-180
debug mode, starting, 317-320
disabling for websites, 319
enabling for websites, 318-319
event object initialization, 436
macros, 430
multithreaded applications, 358-361
phases of, 316-317
with productivity tools. See productivity tools
remote debugging, 356-357
sequential workflows, 833
stored procedures, 713
WCF services, 357-358
windows in debugger, 322
declarative rule conditions, 819
default database reference, 719
default IDE settings, 213
default programming language, 507
Define Deployment dialog box, 1072
defining
  build project file, 1159, 1162
  build retention policy, 1162
Delay activity, 810
delimiters, 277-279
dependencies (projects), 115
deploying
  database projects, 1149-1150
  WPF (Windows Presentation Foundation) applications, 676-679
Deployment Designer, 1071-1073
deployment reports, 1074
validating deployment, 1073
deployment diagrams, 1048, 1071-1073
creating, 1050
deployment reports, 1074
validating deployment, 1073
deployment options (database projects), 1121-1122
deployment reports, 1074
depth of inheritance, 889
derived classes, 998
description pane (Object Browser), 155
design pane (WPF designer), 620-621
design tools (WPF), 30-31
design view, 520
designers. See also names of specific designers
ASP.NET, 518
defined, 159, 162
developers versus, 578
designing. See also planning
applications, 1050-1051
reports, 918-919
sequential workflows, 811-814
state machine workflows, 834-836
user interface, 577-580
web form applications
browser output, 204-205
controls, 200-201
editing markup, 202
styles, 204
web form applications projects, 200
desktop themes. See visual styles
Detach All command, 339
DetailsView control, 574, 736
developer role
Visual Studio Team Developer, 888-890
Visual Studio Team System Database, 890-892
developer testing. See unit tests
developers
designers versus, 578
role with work items, 980-983
Development Tools Environment. See DTE/DTE2 root object
device emulators, 36
diagnostics. .NET Framework 3.5 enhancements, 99
Diagram menu (Application Designer), 1051-1053
diagrams. See database diagrams
direct events (WPF), 637
directories. See folders
disabling website debugging, 319
DISCO (Discovery Document), 753
Disco.exe, 765
Dock property (form controls), 589
docking
controls, 589, 600
IDE windows, 76-78
windows, 11
DockPanel control (WPF), 626-627
document extensions (Office), creating
actions panes, 870-872
data cache, 872-874
host controls, 868-870
overview, 867
smart tags, 874-877
Windows Forms controls, 868-869
Document object, 398
properties/methods, 399
text documents, 400-408
Document Outline, 155-157, 259-260
Document projects (Office), 33
document windows, 377
documents, 398-399
document extensions (Office), creating
actions panes, 870-872
data cache, 872-874
host controls, 868-870
overview, 867
smart tags, 874-877
Windows Forms controls, 868-869
Document object, 398
properties/methods, 399
text documents, 400-408
Document Outline, 155-157, 259-260
text documents, 400-402
distinguishing from nontext documents, 401
editing, 402-403, 406-408
downloading process templates, 987-988
drag-and-drop operations, 151-152
DropDownList control, 565
DTE.AddIns collection, 460
DTE/DTE2 root object, 369
Duplex Contract, 357
dynamic code analysis, 888
Endpoints
- connecting applications in application diagram, 1055-1056
- connecting servers to zones, 1067-1068
- defined, 781
- defining, 790-796
- proxy endpoints, 1061

Enhancements. See new features of Visual Studio 2008

EnvDTE/EnvDTE80 assemblies, 366

Environment settings
- changing, 902
- configuring IDE, 54-56

EnvironmentEvents module, 431

Errors. See also exceptions
- productivity tools. See productivity tools publishing errors (work items), 975

Evaluating code coverage analysis, 1014-1015

Event handlers, 416. See also events
- color palette example (add-ins), 462-463, 466-469
- creating, 584
- for image viewer application example (WPF), 645
- in macros
  - adding event declarations, 434
  - event definitions, 431
  - initializing event object, 435-436
  - writing event handler, 433-434
  - removing arguments from, 97

Event Logs node (Server Explorer), 149

Event model
- ASP.NET, 518, 523-526, 562
- workflows, 826-829

EventDriven activity, 810

Events. See also event handlers
- defining in Class Designer, 1002-1004
- event declarations, adding to macros, 431, 434
- event model
  - ASP.NET, 518, 523-526, 562
  - workflows, 826-829
- event objects, initializing, 435-436
forms, 583-584
Ribbon events, 863
routed events (WPF), 637-638
trace providers, 1021
Excel
managing work items with, 915-916
work items, 972-977
Exception Assistant, 322
exceptions
breaking execution for, 320-321
Exception Assistant, 322
web service exceptions, 779-780
exclusive access (source control), 932
Execute method, 491-495
executing. See running
execution, continuing after breakpoints, 325
explicit naming, 262
explorers
Class View. See Class View
defined, 135
Server Explorer. See Server Explorer
Solution Explorer. See Solution Explorer
Export Report dialog box, 1023
Export Template Wizard, 234-236
exporting
macros, 425-426
process templates, 987-988
project templates, 235-236
settings, 13-16, 55-56
Express Editions (Visual Studio 2008), 47-48
Expression Blend, 31, 52, 681
expression builder, 172
Expression Design, 52
Expression Encoder, 52
Expression Media, 52
Expression products (Visual Studio 2008), 51-52
Expression Studio, 52
Expression Web, 52
extender providers, 591
extending builds with custom tasks, 1166
Extensible Application Markup Language. See XAML
Extensible Markup Language. See XML
extension features (Office)
action panes, 855
data cache, 856
Ribbons, 856
smart tags, 857
table of, 855
task panes, 855
extension methods, 87-88
extensions, Office document extensions
actions panes, 870-872
data cache, 872-874
host controls, 868-870
overview, 867
smart tags, 874-877
Windows Forms controls, 868-869
Extract Interface dialog box, 306
Extract Interface refactoring tool, 303-306
Extract Method refactoring tool, 295-301
accessing, 296
method stubs, generating, 303
single line of code, extracting, 301-302
extracting values from web tests, 1096-1097
extreme programming, 285

F
Favorites folder (Team Explorer), 907
feature sets. See areas (work items)
feedback, Customer Feedback Options (Help menu), 223-224
fields
defining in Class Designer, 1002-1004
Encapsulate Field refactoring tool, 312-314
File Breakpoint Location dialog box, 346
File Comparison tool (source control), 943
File menu, 60
file systems as website locations, 503
files
adding to source control, 930-931
attaching to work items, 965
checking into source control, 933-939
file locations, breakpoints on, 346
file types
  for ASP.NET website projects, 510-511
  for solution items, 112
project definition files, 121
merging changes, 942-944
project definition files, 121-127
retrieving from source control, 932
shelving in source control, 940-941
unshelving in source control, 941
files pane (source control), 927
FilteredTextBoxExtender control, 664
filters
  for breakpoints, 346
  for work items, 967-970
Find and Replace window, 170
  Find In Files mode, 173-176
  Find Symbol mode, 176-177
  Quick Find mode, 171-172
  Quick Replace mode, 172-173
  Replace In Files mode, 176
Find In Files mode, 173-176
Find Results window, 174-176
Find Symbol mode, 176-177
finding, See searching
flagging threads, 358-359
floating IDE windows, 78
flow layout, 528
flow of code, controlling, 180-181
FlowLayoutPanel class, 591
folders
  for ASP.NET website projects, 508-510
  for bookmarks, 170
  solution folders, 113
  team project folders, 906
folders pane (source control), 927
fonts
  printer settings, 181-182
  selecting, 12-13
Fonts and Colors Options dialog box, 181-182
ForeColor property (forms), 583
Foreign Key Relationships dialog box, 703
foreign keys
  creating, 701
  referential integrity, 704
Form.BackgroundImage property (forms), 583
Format menu, 62
formatting
  code in #, 81
  HTML, 202-204
forms. See also web pages;
  Windows Forms Designer
  components, 584
  controls
    adding, 189-190
    appearance properties, 594-595
    arranging, 191-193
    auto-generating bound controls, 725-731
    changing type of, 601
    composite controls, 607
    containers, 590-594
    creating, 607-611
    custom controls, 610-611
    data presentation, 603-606
    defined, 584
    manually bound controls, 732-735
    positioning, 586-590
    resizing, 193
    subclassing, 607
    tab controls, 609
    Tool Strip controls, 596-602
    user controls, 607-610
    z-order, 599
creating, 581-584
customizing appearance of, 189
Document Outline, 155-157
events, 583-584
inheriting from, 582
menus, 596-598
properties, 582-583
resizing, 586-590
startup forms, 581-582
status bars, 600-602
toolbars, 598-600
user interface, 577-580

How can we make this index more useful? Email us at indexes@samspublishing.com
Forms Designer, 585
FormStartPosition enumeration value (forms), 582
FormView control, 574, 736
forums. See MSDN Community Forums
friend assemblies, 95
FTP sites as website locations, 507
function breakpoints, 340
functions
  function breakpoints, 340
  lambda expressions, 89-91
  user-defined functions, 715-716
Functions subreport, 1032

G
General Properties property page, 1018-1019
generating data, 1149
  creating data generation plan, 1142-1144
  previewing generated data, 1144-1148
generic tests, 890, 1110
Get Latest command (source control), 930-932
Getting Started section (Start Page), 213-214
global security groups, 899
Go To Line dialog box, 168
Grid control (WPF), 627-630
grid positioning (Windows Forms designer), 191-192
GridView control, 574, 736-740
groups
  adding to team projects, 909
  mapping to process roles, 900-901
GUI. See user interfaces

H
HandleExternalEvent activities, 810, 821-823
handling events. See event handlers
help favorites, moving, 56
Help menu, 64
  Customer Feedback Options, 223-224
  MSDN Forums button, 217-223
  Search option, 224-225, 228
hiding non-visual controls, 552
hierarchical data (form controls), 603-604
history (work items), tracking, 961
hit count for breakpoints, 347
Hit Count command, 347
host controls, 868-870
hosting WCF services, 798-799
hosts, 781, 802
  defining for sequential workflows, 824-832
  defining in state machine workflows, 844-849
HoverMenuExtender control, 664
HTML controls, 562
HTML designer
  controls
    adding, 200-201
    arranging, 201
  designing web form applications
    browser output, 204-205
    controls, 200-201
    editing markup, 202
    styles, 204
  HTML source editor and, 202-203
  productivity tools, 261
  validation options, 204-205
HTML navigation, 259-260
HTML source editor, 202-203
HTTP (Hypertext Transfer Protocol), 753
HyperLink control, 564
hyperlinks
  active hyperlinking, 255
  HyperLink control, 564
Hypertext Transfer Protocol (HTTP), 753
.ico files, 490
icons
- associating with wizards, 490
- for breakpoints, 341
members pane (Class View), 146
objects pane (Class View), 144
reusing, 862
Solution Explorer, 136-139
IDE
C# enhancements, 81-82
configuring, 54-56
default settings, 213
IDE Navigator, 10-11
menu bar, 59-64
multi-targeting, 79
projects, creating, 59
Properties window, 74-75
Solution Explorer, 73-74
Start Page, 57
startup options, 57-58
text editors
- code editors, 70-71
- customizing, 72
toolbars, 64-66
Toolbox, 67
VB enhancements, 80-81
Visual Designers, 69
windows
- docking, 76-78
- floating, 78
- pinning, 75-76
IDE Navigator, 10-11
IDTCommandTarget interface, 452
IDTExtensibility2 interface, 452-458
IDTToolsOptionsPage interface, 470
IDTWizard interface, 487-488
IfElse activities, 819-820
IfElse activity, 810
IIS
- hosting WCF services, 799
- local IIS version as website location, 504-505
- remote IIS servers as website location, 507
Image control, 565
image effects, 646
image storage, 641-643
image viewer application example (WPF), 638-639
- binding to images, 643-645
- button event handlers, 645
- image effects, 646
- image storage, 641-643
- layout for, 639-641
- path selection dialog box, 646-652
ImageList controls, 604
Images Collection Editor dialog box, 604
Immediate window (debugger), 322
implementing applications, 1074-1077
implicit naming, 262
implicit typing, 83-85
Import and Export Settings Wizard, 56
importing
- database schemas, 1119-1120
- macros, 425-426
- process templates, 990-991
- server settings, 1071
- settings, 13-16, 55-56
incremental searches, 177-178, 382
Incremental Search mode, 177-178
Index Columns dialog box, 701
indexes, 701
Indexes/Keys dialog box, 701
indicator margin (code editor), 166
infrastructure design
- infrastructure architects, 1048
- logical datacenter diagrams, 1062-1063
- adding servers to zones, 1065-1066
- configuring zones and servers, 1068-1071
- connecting servers to zones, 1067-1068
Logical Datacenter Designer Toolbox, 1064
zones, 1064-1065

inheritance, 756, 1000
controls, 607
depth of, 889
from forms, 582
Inheritance Picker dialog box, 582
initializing
objects, 85-87, 435-436
states, 837-844
inline functions, 716
inline styles, 533
Insert Table dialog box, 202
inserting
applications into application diagrams, 1055
comments into text windows, 408
items into class diagrams, 998
servers into zones, 1065-1066
inspectors, 865
installing
macro packaging example content, 244-247
project templates, 237-238
shared content, 229-231
Team Foundation Server (TFS), 901
Visual Studio 2008, 53-56
Instrumentation Properties property page, 1019-1020
IntelliSense, 252, 263-264
brace matching, 277-279
in C#, 81
code snippets, 268-277
Complete Word, 264
customizing, 279
List Members, 265-266
Organize Usings, 268
Parameter Info, 267
Quick Info, 265
in VB, 80
interface (visual relationship), 1000
Interop Forms Toolkit, 80
InvokeWebService activity, 810, 820-821
InvokeWorkflow activity, 810
invoking web methods, 772
IsCached method, 873
issue work items (MSF for CMMI), 955
item templates
creating, 238
as shared content, 229
in Visual Studio Team Architect, 1049-1050
item types (Solution Explorer), 138
Items Collection Editor dialog box, 602
iterations
team projects, 910-912
work items, 956-959

J–K

Join dialog box, 709
joins, creating, 708-709
key bindings, mapping, 414
Key keyword (VB), 89
keyboard shortcuts. See command keys

L

Label control, 564
lambda expressions, 89-91
languages. See also C#; VB
enhancements, 82-83
anonymous types, 88-89
auto-implemented properties, 97
business logic in partial methods, 91-93
embedded XML, 96-97
extension methods, 87-88
friend assemblies, 95
implicit typing, 83-85
lambda expressions, 89-91
LINQ queries, 94-95
object initializers, 85-87
removing arguments from event handlers, 97
selecting, 53, 443

target source language for website projects, 120

Language Integrated Query. See LINQ

Launch Properties property page
performance sessions, 1019
session targets, 1023

layout. See page layout

layout grid (Windows Forms designer), 191-192

layout panel controls (WPF)
Canvas control, 624-626
DockPanel control, 626-627
Grid control, 627-630
image viewer application example, 639-641
StackPanel control, 630-631

Level property (validation rules), 1098

life cycle (ASP.NET), 518, 523-526, 562

line numbering, 168

line wrapping. See word wrapping

line-of-business (LOB) applications, 32

lines of code, counting, 889

linked windows, 392-393

linking

code to work items, 983

tool windows, 392-393

work items, 961-964

links. See hyperlinks

LINQ (Language Integrated Query), 22

default database to object mappings, 742
LINQ to SQL classes, 44

.NET Framework 3.5 enhancements, 98
O/R Designer, 742-745
objects, 747-748
overview, 741-742
queries, 94-95
anonymous types, 88-89
lambda expressions, 89-91
sample code, 745-747

LINQ to SQL classes, 44

LinqDataSourceControl, 22-23

List Members (productivity tool), 265-266

listings

adding fields to forms, 993

auto-generated unit test sample, 1010

BusinessEntities.Customer class, 757
calling the GetCustomerProfile web method, 778
code generated by the Add-in Wizard, 446-451

component designer–generated code, 207

Connect, PaletteControl, and PaletteControlOptionPage classes, 473-486

creating SOAP exceptions, 780

Customer Links Web Part, 556
customer profile web service, 761-762

CustomerProfileService implementation code, 785

EntityServices.CustomerProfile class, 759

host/client application code (sequential workflow), 829

host/client application code (state machine workflow), 847

ICustomerProfileService interface definition, 785

image viewer code behind C#, 649-652

image viewer XAML code, 648-649

implementing dialog box with Grid panel, 629-630

marking the customer class as DataContract, 787

new field inside Bug.xml, 992

new tasks to be seeded, 990

replacing button templates, 634-635

sample unit test, 1006

.sitemap file, 572

skin file example, 546

SOAP 1.2 request, 770

SOAP 1.2 response, 771

.VSContent file structure, 240

.VSContent file for smart client example, 242

.vstemplate file sample, 236

Windows Forms designer–generated code, 194-196

Lists of Tests node (Test List Editor), 1084
ListSearchExtender control, 664
ListView control, 20-21, 574
load tests
  creating, 1100-1106
  editing, 1106
  running, 1107-1108
  in Visual Studio Team Test, 890
loading
  add-ins, 461
  macros with events, 433
LOB (line-of-business) applications, 32
local IIS version as website location, 504-505
local type inference, 83-85
local variables
  Promote Local Variable to Parameter refactoring tool, 309-311
  refactoring and, 309
Locals window (debugger), 322, 350
locations
  form design and, 578-579
  locations in files, breakpoints on, 346
  for website projects, 502-507
Logical Datacenter Designer, 1062-1063
  Toolbox, 1064
  zones, 1064-1065
    adding servers to, 1065-1066
    configuring, 1068-1071
    connecting servers to, 1067-1068
logical datacenter diagrams, 1048, 1062-1063
  adding servers to zones, 1065-1066
  configuring zones and servers, 1068-1071
  connecting servers to zones, 1067-1068
  item templates for, 1049
  Logical Datacenter Designer Toolbox, 1064
  zones, 1064-1065
Login control, 569
login controls, 567-569
LoginName control, 569
LoginStatus control, 569
LoginView control, 569
look and feel. See consistency

M

Macro Explorer, 421-422
Macro Recorder toolbar, 420
macros
  event handling
    adding event declarations, 434
    event definitions, 431
    initializing event object, 435-436
    writing event handler, 433-434
  limitations of, 441
  Macro Explorer, 422
  Macros IDE, 423
    adding macro projects, 427
    debugging macros, 430
    editing recorded macros, 427-430
    project files, 424-425, 428
    sharing macros, 425-426
  packaging, 242-247
    with parameters, 439-440
  purpose of, 419-420
  recording, 420
  running, 421
    from command window, 439-440
    from keyboard shortcuts, 437-438
    from toolbars/menus, 436-437
  as shared content, 229
Macros IDE, 423
  adding macro projects, 427
  debugging macros, 430
  editing recorded macros, 427-430
  project files, 424-425, 428
  sharing macros, 425-426
maintainability index, 889
Manage Styles dialog box, 24, 537
Manage Workspaces dialog box, 928
managed code, 719-723
Management Classes node (Server Explorer), 150
management commands (Solution Explorer)
  for projects, 141-142
  for solutions, 141
Management Events node (Server Explorer), 150
managing
tests, 1083-1084
threads, 360-361
web test requests, 1088
manual tests
creating, 1109
running, 1109-1110
in Visual Studio Team Test, 890
manually binding controls, 732-735
many-to-many relationships, 705
mapping
data sources to controls, 728-731
key bindings, 414
object relational mapping
LINQ (Language Integrated Query), 741-748
overview, 740-741
objects to relational data, 43-44
process roles to TFS security groups, 900-901
markup (HTML). See HTML source editor
MaskedEditExtender control, 664
.master file extension, 511
master pages, 25-27
creating, 541-542
for user interfaces, 541-544
nesting, 544
measurements in Code Metrics, 1042
Media Integration Layer, 614
members
displaying with Class Designer, 997
explicit naming, 262
implicit naming, 262
List Members productivity tool, 265-266
members pane
Class View, 146
Object Browser, 153
membership (ASP.NET), 567
menu bar, 59-64, 397. See also command bars
Menu control, 571
menus
creating, 596-598
Menu control, 571
running macros from, 436-437
MenuStrip controls, 596-598
merge tool (source control), 944
merging source code tree, 944-947
message classes, 865
Message Queues node (Server Explorer), 151
method stubs, generating, 303
methodologies, 987-991. See also MSF for Agile; MSF for CMMI
methods
Command object, 412-413
CommandBar object, 397
defining in Class Designer, 1002-1004
Document object, 399
DTE2 root object, 369
EditPoint object, 403
extension methods, 87-88
Extract Method refactoring tool, 295-301
accessing, 296
method stubs, generating, 303
single line of code, extracting, 301-302
in IDTExtensibility2 interface, 452-458
IDTToolsOptionsPage interface, 470
IsCached, 873
OutputWindowPane object, 391
partial methods, business logic in, 91-93
Project object, 373
Promote Local Variable to Parameter refactoring tool, 309-311
Remove Parameters refactoring tool, 307-308
Reorder Parameters refactoring tool, 311-312
Solution object, 372
StartCaching, 873
TaskItem object, 384
TextDocument object, 400
Window object, 378-380
metrics in Visual Studio Team Developer, 889
Microsoft AJAX Library, 20

How can we make this index more useful? Email us at indexes@samspublishing.com
Microsoft ASP.NET Futures, 681
Microsoft Excel. See Excel
Microsoft Office
  Office applications. See Office applications
  Team Foundation Server and, 915-916
Microsoft Office Project Server, 916
Microsoft Project. See Project
Microsoft Silverlight. See Silverlight applications
Microsoft Solutions Framework. See MSF
Microsoft Windows User Experience, 579
mobile device application development, 36-38
ModalPopupExtender control, 664
models. See Class Designer; Visual Studio
  Team Architect
Modify Style dialog box, 536
modules. See areas (work items)
monitoring builds, 1167-1171
mouse movement event handlers, 462-463
moving help favorites, 56
MSBuild, 518, 1166
MSDN
  Community Center, 228
  Community Forums, 218
    alerts, 221-223
    navigating, 220-221
    new threads, starting, 220
  Visual Studio Professional editions, 49
MSDN Forums button (Help menu), 217-223
MSF (Microsoft Solutions Framework), 882
MSF Agile, 883
  roles, mapping to TFS security groups, 900-901
  work items, 951
    bugs, 953
    QoS requirements, 952
    risk, 953
    scenarios, 952
    tasks, 953
MSF for CMMI, 883-885
  roles, mapping to TFS security groups, 900-901
  work items, 953
    bugs, 955
    change requests, 954-955
  issues, 955
  requirements, 954
  reviews, 955
  risk, 955
  tasks, 954
  multi-targeting, 79
  multithreaded applications, debugging, 358-361
  MultiView control, 565

N
named commands, add-ins reacting to, 458-459
named regions, creating, 257
named skins, 546
naming. See also
  renaming
  builds, 1158
  project portals, 902-904
  team projects, 902
  threads, 359
navigating
  HTML navigation, 259-260
  MSDN Community Forums, 220-221
navigation buttons on Standard toolbar, 65
navigation controls, 571-572
navigation tools in code editor, 168-170
nesting master pages, 544
.NET
  languages. See C#; VB
    web services in, 754
.NET Compact Framework, 100
.NET Framework, 79
  enhancements to version 3.5, 98-100
  version of, selecting, 8-10
  Visual Studio 2008 integration, 7
NetCF, 17
network types, selecting for load tests, 1104
New Breakpoint dialog box, 340
new features of Visual Studio 2008, 8
  business process development, 38
    service-oriented development, 41-42
  workflows, 38-40
C# IDE enhancements, 81-82
Class Designer, 16
code sharing, 16
data access, 42
  mapping objects to relational data, 43-44
synchronization, 46-47
typed datasets, 43
docking windows, 11
fonts, selecting, 12-13
IDE Navigator, 10-11
importing/exporting settings, 13-16
language enhancements, 82-83
  anonymous types, 88-89
  auto-implemented properties, 97
  business logic in partial methods, 91-93
  embedded XML, 96-97
  extension methods, 87-88
  friend assemblies, 95
  implicit typing, 83-85
  lambda expressions, 89-91
  LINQ queries, 94-95
  object initializers, 85-87
  removing arguments from event handlers, 97
.NET Framework 3.5 enhancements, 98-100
.NET Framework version, selecting, 8-10
standard dialog boxes, 12
for user application development, 17-18
  mobile device applications, 36-38
VSTO (Visual Studio Tools for Office), 32-35
web development productivity features, 18-27
WPF (Windows Presentation Foundation), 27-32
VB IDE enhancements, 80-81
New File dialog box, 163
New List dialog box, 973
New Load Test Wizard, 1100
New Project dialog box, 106, 117, 119, 188, 581, 716
New Schema Comparison dialog box, 1126
New Style dialog box, 537
New Style/Modify Style dialog box, 24
New Web Site dialog box, 119-120, 500
news channel (Start Page), 214-217
non-visual controls, hiding, 552
nontext documents, distinguishing from text documents, 401
notes, check-in notes, 937-938

O
O/R Designer, 44, 742-745
OBA (Office Business Application), 33
Object Browser, 152-155
  custom component set, editing, 153
  description pane, 155
  Rename refactoring tool, accessing, 293
  scoping options, 153
object definitions, viewing, 1128
object initializers, 85-87
Object Property dialog box, 1023
objects
  LINQ objects, 747-748
  mapping to relational data, 43-44
  relational mapping
    default database to object mappings, 742
    LINQ (Language Integrated Query), 741-748
    O/R Designer, 742-745
  overview, 740-742
Objects Lifetime subreport, 1032
objects pane, 142
  Class View, 144-145
  Object Browser, 153
OCA (occasionally connected applications), 100
Office applications
  add-ins, creating
    Outlook Form Regions, 865-867
  overview, 859
  Ribbon customization, 860-863
  task pane customization, 863-864
document extensions, creating
actions panes, 870-872
data cache, 872-874
host controls, 868-870
overview, 867
smart tags, 874-877
Windows Forms controls, 868-869
extension features
action panes, 855
data cache, 856
Ribbons, 856
smart tags, 857
table of, 855
task panes, 855
icons, reusing, 862
overview, 853-854
Visual Studio Office project types, 858
Office Business Application (OBA), 33
Office development, VSTO (Visual Studio Tools
for Office), 32-35
Office Project Server, 916
OnAddInsUpdate method, 453-454
OnBeginShutdown method, 454
OnConnection method, 454-457
OnDisconnection method, 457
one-to-one relationships, 705
OnMacrosRuntimeReset event, 435
OnPaint event, 611
OnStartupComplete event, 435
OnStartupComplete method, 458
Opacity property (forms), 583
opening code editor, 163
Options dialog box, 279, 437
Options page (color palette add-in example),
creating, 469-473
ordered tests, 890, 1112
Organize Usings (productivity tool), 268
organizing database projects, 1118
outlining
code outlining (productivity tool), 257-259
Edit, Outlining menu, 257-259
HTML. See HTML navigation
Outlook Form Regions, 865-867
output window, 390-391
OutputWindow object, 390-391
OutputWindowPane object, 390-391
packaging
macros, 242-247
shared content, 239-247
page event handlers, adding, 525
page layout
of form controls, 586-594
of forms, 191-193
for user interfaces, 528-531
page-level styles, 533
Page_Init event, 524
Page_Load event, 524
Page_PreInit event, 524
Page_Pre_Render event, 525
Page_Unload event, 525
paging, 724
Panel class, 590
panels, layout panel controls (WPF), 624
Canvas control, 624-626
DockPanel control, 626-627
Grid control, 627-630
image viewer application example, 639-641
StackPanel control, 630-631
Parallel activity, 810
Parameter Info (productivity tool), 267
parameters. See also project parameters;
variables
debugging stored procedures, 713
macros with, 439-440
Parameter Info productivity tool, 267
passing to workflows, 825-826
Promote Local Variable to Parameter
refactoring tool, 309-311
Remove Parameters refactoring tool,
307-308
Reorder Parameters refactoring tool,
311-312
partial classes, 519
partial methods, business logic in, 91-93
partial-page update, 656-658
passing parameters to workflows, 825-826
PasswordRecovery control, 569
PasswordStrength control, 665
path selection dialog box in image viewer application example (WPF), 646-652
patterns (for load tests), defining, 1101
peer-to-peer networking support, 100
Pending Changes window, 934
Performance Counters node (Server Explorer), 151
Performance Explorer, 1016
  performance reports
    columns in, 1024-1031
    comparing, 1033
    subreports, 1031-1032
  performance sessions
    configuring, 1018-1023
    creating, 1016-1018
    targets, 1023
  Reports node, 1023
  toolbar buttons, 1016
performance reports
  Allocation subreport, 1032
  Call Tree subreport, 1032
  Caller/Callee subreport, 1032
    columns in, 1024-1031
    comparing, 1033
  Functions subreport, 1032
  Objects Lifetime subreport, 1032
  Summary subreport, 1031
performance sessions
  configuring, 1018-1023
  creating, 1016-1018
  targets, 1023
permissions
  source control system, 923-924
  team projects, 909
Personal Web Starter Kit template, 502
pinning IDE windows, 75-76
pipe streams, 99
placeholder values (code snippets), 270
planning
  form design, 579-580
  user interfaces, 526-528
policies, check-in policies, 934-936
Policy Failure dialog box, 936
port numbers for ASP.NET Development Server, 504
portal. See project portal
positioning controls for user interfaces, 528-531. See also page layout
post-build events, 130
post-deployment scripts, 1122
Power Pack controls (VB), 80
pre-build events, 130
predeployment scripts, 1122
prescriptive remediation (Code Analyzer), 1041
Preview Changes dialog box, 291, 1136
previewing
  generated data, 1144-1148
  refactoring changes, 291
primary keys
  creating, 701
  referential integrity, 704
Print dialog box, 181, 183
printing from code editor, 181-183
procedures. See stored procedures
process guidance, 915
process model
  MSF Agile, 883
  MSF for CMMI, 884-885
Process Template Manager, 987
process templates
  exporting, 987-988
  importing, 990-991
  security, 900
  selecting, 902
processes, attaching to, 323-324
productivity tools, 251-253
  in code editor
    code outlining, 257-259
    HTML navigation, 259-260
    smart tags, 262
  in HTML designer, smart tags, 261

How can we make this index more useful? Email us at indexes@samspublishing.com
IntelliSense, 252, 263-264  
brace matching, 277-279  
code snippets, 268-277  
Complete Word, 264  
customizing, 279  
List Members, 265-266  
Organize Usings, 268  
Parameter Info, 267  
Quick Info, 265  
Task List, 280-281  
comment tasks, 281-282  
shortcut tasks, 283  
user tasks, 283  
in text editor  
active hyperlinking, 255  
change tracking, 253-254  
coding problem indicators, 254-255  
syntax coloring, 255-256  
in Windows Forms designer, smart tags, 261  
Professional editions (Visual Studio 2008), 49  
profiling, 1015. See also Performance Explorer; performance sessions  
programming with Server Explorer, 151-152  
programming language for website projects, selecting, 507  
project administrators (Team Foundation Server security), 900  
project alerts. See alerts  
project definition files, 121-127  
Project Dependencies property page, 115  
Project Designer, 128  
project files  
build project files  
defining, 1159, 1162  
editing, 1165  
in Macros IDE, 424-425  
saving, 428  
selecting for builds, 1158  
Project Group Membership dialog box, 907, 909  
project items, 127  
project manager role  
Team Foundation Server, 892-895  
with work items, 972-978  
Project object  
code access, 374-375  
properties/methods, 373  
solution/project hierarchy, 370-372  
project parameters, 233-235  
project portal, 914-915  
naming, 902-904  
in Team Foundation Server, 893  
project properties, 128-132  
Project Properties dialog box, 581, 1035  
project reports. See reports  
Project Security dialog box, 958  
project security groups in Team Foundation Server, 900, 907, 909  
project teams. See team projects  
project templates  
creating, 232-238  
for database projects, 1117-1122  
in Visual Studio Team Architect, 1048-1049  
as shared content, 228  
Project work items  
managing, 915-916  
project manager role and, 978  
projects. See also database projects; rich browser applications; solutions; team projects; website projects  
accessing code within, 374-375  
creating, 59, 106, 108, 117, 119, 188  
defined, 105, 117  
dependencies, 115  
management commands in Solution Explorer, 141-142  
project definition files, 121-127  
project items, 127  
project properties, 128-132  
project work, creating, 1080-1082  
usage, 127  
Windows Presentation Foundation projects, creating, 197-198  
Promote Local Variable to Parameter refactoring tool, 309-311
properties
   auto-implemented properties, 97
   Command object, 412-413
   CommandBar object, 397
   defining in Class Designer, 1002-1004
   Document object, 399
   DTE2 root object, 369
   EditPoint object, 403
   Encapsulate Field refactoring tool, 312-314
   forms, 582-583
   OutputWindowPane object, 391
   Project object, 373
   project properties, 128-132
   Solution object, 372
   solution properties, 114-116
   TaskItem object, 384
   TextDocument object, 400
   Window object, 378-380
Properties window, 74-75, 621, 805, 1018
Property Pages dialog box, 512-518
proxy endpoints, 1061
proxy implementation in web services, 763
proxy servers
   Search option (Help menu) and, 227
   source control and, 923
Publish property page, 131
publishing errors (work items), 975
publishing shared content, 247

Q
QoS (quality of service) requirement work items
   (MSF for Agile), 952
quality states (builds), setting, 1168-1169
queries
   LINQ (Language Integrated Query)
   default database to object
      mappings, 742
   O/R Designer, 742-745
   objects, 747-748
   overview, 741-742
   sample code, 745-747
   types of, 709-710
   on work items, 967-970
   writing, 706-709
query strings, binding to data, 1094-1096
Query/View Designer, 706-713
querying Windows collection, 380-381
Quick Find mode, 171-172
Quick Info (productivity tool), 265
Quick Replace mode, 172-173
QuickWatch window (debugger), 352
R
RadioButton control, 565
rafting, 594
RangeValidator control, 567
.rd file extension, 511
readers (Team Foundation Server security), 900
ReceiveActivity activity, 811
Recent Projects section (Start Page), 213
record-level details, viewing, 1132
recorded macros, editing, 427-430
recording
   macros, 420
   web tests, 1086-1087
Refactor menu, 61, 287
Refactor! tool, 80
refactoring
   defined, 285
   rename refactoring, 1135-1136
refactoring tools
   in C#, 81
   Encapsulate Field, 312-314
   Extract Interface, 303-306
   Extract Method, 295-301
      accessing, 296
      method stubs, generating, 303
      single line of code, extracting, 301-302
      invoking, 287
      with Class Designer, 290
      with Refactor menu, 287
      with smart tags, 287-289

How can we make this index more useful? Email us at indexes@samspublishing.com
list of, 286
local variables and, 309
previewing changes, 291
Promote Local Variable to Parameter, 309-311
Remove Parameters, 307-308
Rename, 292-295
Reorder Parameters, 311-312
for Visual Basic, 80, 286
Reference Paths property page, 131
references
service references, 772-775
in website projects, 512-513
References property page, 131
referencing windows, 377-378
referential integrity, 704
reflexive relationships, 706
region keyword, 257
registered add-ins, 461
registering Options page (color palette add-in example), 472-473
regular expressions in Find and Replace window, 172
RegularExpressionValidator control, 567
relational data, mapping objects to, 43-44
relational mapping (objects)
  LINQ (Language Integrated Query)
    default database to object mappings, 742
  O/R Designer, 742-745
  objects, 747-748
  overview, 741-742
  sample code, 745-747
  overview, 740-741
relationships
  between classes, 999-1002
  of tables
    building, 703-706
    in data generation plan, 1143-1144
relative positioning, 201, 528
remote debugging, 356-357
Remote Debugging Monitor, 356
remote IIS servers as website location, 507
Remove Parameters refactoring tool, 307-308
removing arguments from event handlers, 97
Rename dialog box, 294-295, 1135
rename refactoring, 1135-1136
Rename refactoring tool, 292
  accessing, 292-294
  Rename dialog box, 294-295
renaming
  process templates, 991
  solutions, 108
  zipped files, 244
Reorder Parameters refactoring tools, 311-312
ReorderList control, 665
Repeater control, 574, 736
replace. See search-and-replace modes
Replace In Files mode, 176
reports. See also performance reports
  build reports, 1170-1171
  deployment reports, 1074
  in Team Foundation Server, 893, 917-919
Reports node (Performance Explorer), 1023
repositioning EditPoint objects, 407
requests, managing web test requests, 1088
RequiredFieldValidator control, 567
requirement work items (MSF for CMMI), 954
reserved template parameters, 233-235
resizing
  controls, 193, 580
  forms, 586-590
Resolve Conflicts window, 942
Resolve Version Conflict window, 942
resources
  defined, 131
  WPF, embedding, 634
Resources property page, 131
results, viewing
  Code Analyzer, 1039
  Code Metrics, 1042-1043
retention policy (builds), 1162
retrieving files from source control, 932
review work items (MSF for CMMI), 955
Ribbons
adding items to, 862-863
customizing, 860-863
definition, 856

rich browser applications
AJAX
Accordian control, 663
adding controls to pages, 662, 666, 668
AJAX Control Toolkit, 660-662
ASP.NET AJAX controls, 656
AutoCompleteExtender control, 664
CalendarExtender control, 664
CollapsiblePanelExtender control, 663
ConfirmButtonExtender control, 665
FilteredTextBoxExtender control, 664
HoverMenuExtender control, 664
ListSearchExtender control, 664
MaskedEditExtender control, 664
ModalPopupExtender control, 664
overview, 653-654
partial-page update, 656-658
PasswordStrength control, 665
ReorderList control, 665
ScriptManager control, 655
ScriptManagerProxy control, 655
SliderExtender control, 665
TabContainer control, 665
Timer control, 656
UpdatePanel control, 655
UpdateProgress control, 656, 659-660
overview, 653

Silverlight applications
adding to web pages, 690-694
creating, 681-690
development tools, 681
overview, 679-680

WPF (Windows Presentation Foundation) applications
compared to XBAP, 669-670
creating, 670-671
definition, 668
deploying, 676-679
security, 671-676

XBAP (XAML browser applications)
compared to WPF applications, 669-670
definition, 668
security, 671-676
rights, source control system, 923-924
RightToLeft property (form design), 578
RightToLeftLayout property (form design), 578
risks, 890
risk work items
MSF for Agile, 953
MSF for CMMI, 955
routed events (WPF), 637-638
RSS feeds, 214, 216
rules (Code Analyzer)
configuring, 1035-1036
handling violations, 1036-1037
suppressing, 1037-1038
Run To Cursor command, 335-336
running
builds, 1166
Code Analyzer, 1039
Code Metrics, 1042
commands, 413-414
database unit tests, 1140
load tests, 1107-1108
manual tests, 1109-1110
schema update script, 1130
sequential workflows, 832-833
SQL scripts, 719
state machine workflows, 850
unit tests, 1010-1012
WCF services, 788
web tests, 1089

sample image viewer application (WPF), 638-639
binding to images, 643-645
button event handlers, 645
image effects, 646
image storage, 641-643

How can we make this index more useful? Email us at indexes@samspublishing.com
samples as shared content, 229
Sampling Properties property page, 1019
saving
  macro project files, 428
  temporary macros, 421
scalar-valued functions, 716
scaling controls, 589-590
SCE (Source Control Explorer) window, 926-927
scenario work items (MSF for Agile), 952
scenarios (for load tests), defining, 1100-1101
scheduling builds, 1163-1164
Schema Comparison tool, 1126-1128
  schema update script, 1129-1130
setting options, 1130
  viewing object definitions, 1128
schema update script, 1129-1130
Schema View window, 1124
schemas
  creating, 185
  database schemas
    deploying, 1149-1150
    importing, 1119-1120
    in Visual Studio Team System Database Edition, 892
scoping options (Object Browser), 153
ScriptManager control, 655
ScriptManagerProxy control, 655
scripts
  client-side scripts, debugging, 361
  SQL scripts
    auto-generating, 718-719
    defined, 716
    executing, 719
SDLC (Software Development Life Cycle), 881-882
  MSF Agile, 883
    roles, mapping to TFS security groups, 900-901
    work items, 951-953
  MSF for CMMI, 883-885
    roles, mapping to TFS security groups, 900-901
    work items, 953-955
team roles, 882
VSTS (Visual Studio Team System), 885-886. See also source control;
  work items
    Team Foundation Server, 892-895
    Visual Studio Team Architect, 887
    Visual Studio Team Developer, 888-890
    Visual Studio Team System Database, 890, 892
    Visual Studio Team Test, 890
SDM (system definition model), 1050
search bar (Class View), 144
search features for shared content, 229
Search option (Help menu), 224-225, 228
search-and-replace modes, 170
  Find In Files mode, 173-176
  Find Symbol mode, 176-177
  Incremental Search mode, 177-178
  Quick Find mode, 171-172
  Quick Replace mode, 172-173
  Replace In Files mode, 176
searching for work items, 967-970
security
  areas (work items), 958
  macros with events, 433
  source control system, 923-924
  in Team Foundation Server, 899-901, 907, 909
  WPF (Windows Presentation Foundation) applications, 671-676
  XBAP (XAML browser applications), 671-676
Security property page, 131
seeding
  methodology with work items, 987-991
  web tests with data, 1089-1096
selecting
  add-in application host, 443
  add-in language, 443
  browsers for load tests, 1104
  build configurations, 1160
  build tests, 1161
default programming language for website projects, 507
  fonts, 12-13
languages, 53
locations for website projects, 502-507
.NET Framework version, 8-10
network types for load tests, 1104
process templates, 902
project files for builds, 1158
servers for load tests, 1104-1105
templates for website projects, 501-502
test mixes for load tests, 1102-1103
text, 160
windows in code editor, 163
selection margin (code editor), 167
self-checking. See debugging
self-hosted WCF services, 798
SendActivity activity, 811
sequential workflows, 39, 811
   CallExternalMethod activities, configuring, 816-818
   CodeActivity activity, configuring, 818
debugging, 833
designing, 811-814
   HandleExternalEvent activities, configuring, 821-823
host and client, defining, 824-832
IfElse activities, configuring, 819-820
InvokeWebService activity, configuring, 820-821
running, 832-833
server controls, 561
Server Explorer, 147, 697
   Data Connections node, 148
database connections, creating, 698-700
programming with, 151-152
Servers node, 149
   Event Logs node, 149
   Management Classes node, 150
   Management Events node, 150
   Message Queues node, 151
   Performance Counters node, 151
   Services node, 151
Server section (start options of website projects), 517
servers
   adding to zones, 1065-1066
   build servers, 1155-1156
configuring, 1068-1071
correlating to zones, 1067-1068
selecting for load tests, 1104-1105
TFS (Team Foundation Server), 51
Servers node (Server Explorer), 149-151
Service Configuration Editor, 790
Service Reference Settings dialog box, 773
service references, 772-775
service-oriented development, 41-42
services. See WCF; web services
Services node (Server Explorer), 151
sessions. See performance sessions
SetState activity, 810
settings, importing/exporting, 13, 15-16, 55-56
Settings and Constraints window (Application Designer), 1057
Settings property page, 131
shared access (source control), 932
shared add-ins, Visual Studio add-ins
versus, 444
shared content
   installing, 229-231
   item templates, creating, 238
   packaging, 239-247
   project templates, creating, 232-238
   publishing, 247
searching for, 229
signing, 246
types of, 228
shared development servers as website location, 507
SharePoint development, 35
SharePoint sites in Team Foundation Server, 893
sharing
   code, 16
   macros, 425-426
Shelve dialog box, 941
shelvesets, 941
How can we make this index more useful? Email us at indexes@samspublishing.com
shelving
files (source control), 940-941
in Team Foundation Server, 892
shortcut keys. See command keys
shortcut tasks, 283
.sitemap files, 511, 571
signal icons
objects pane (Class View), 145
Solution Explorer, 139
Signing property page, 132
signing .vsi files, 246
Silverlight, 18, 20, 32
Silverlight 1.1 Alpha September Refresh, 681
Silverlight 1.1 Software Development Kit Alpha September Refresh, 681
Silverlight 1.1 Tools Alpha for Visual Studio 2008, 681
Silverlight applications
adding to web pages, 690-694
creating, 681-690
development tools, 681
overview, 679-680
simple data binding, 725
Simple Object Access Protocol. See SOAP
single line of code, extracting, 301-302
site maps, 571
site navigation controls, 571-572
SiteMapPath control, 571-572
skin definitions. See skins
.skin file extension, 511
skins, 546-547
ASP.NET controls, 563
creating, 546-547
SliderExtender control, 665
smart tags, 260
in code editor, 262
creating, 874-877
definition, 857
in HTML designer, 261
refactoring with, 287-289
in Windows Forms designer, 261
SmartTagBase class, 874

snap lines (Windows Forms designer), 192-193
snippets, 229, 268-277
SOAP (Simple Object Access Protocol)
defined, 753
exceptions, throwing, 779-780
requests, 770
responses, 771
Software Development Life Cycle. See SDLC
software models. See Class Designer; Visual Studio Team Architect
solution definition file, 109-111
Solution Explorer, 73-74, 111, 135-136
adding files to source control, 931
checking in source control files, 933
icons, 136-138
projects, management commands, 141-142
Rename refactoring tool, accessing, 293
signal icons, 139
solutions, management commands, 141
toolbar buttons, 139
solution folders, 113
solution items, 112-113
solution properties, 114-116
Solution Property Pages dialog box, 114-116
solution user options file, 109
Solution/Solution2 objects, 370, 372
solutions. See also projects
creating, 106-108
defined, 105-106
management commands in Solution Explorer, 141
renaming, 108
startup projects, 142
Sort options (Class View), 145
sorting tasks, 281
source code tree, branching and merging,
944-947
source control, 921-922
architecture of, 922-923
associating code with work items, 981-983
changesets, 939-940
configuring Visual Studio for, 925
design goals, 922
files
adding, 930-931
checking in, 933-939
merging changes, 942-944
retrieving, 932
shelving, 940-941
unshelving, 941
proxy servers and, 923
security, 923-924
source code tree, branching and merging, 944-947
Source Control Explorer (SCE) window, 926-927
in Team Foundation Server, 51, 892
workspaces, 927-930
Source Control Explorer
adding files to source control, 931
checking in source control files, 933
Source Control Explorer (SCE) window, 926-927
Source Control Merge Wizard, 947
source control settings, 904
Source Control Settings dialog box, 935
source files, pointing debugger at, 115-116
source view (web pages), 520
split panes, creating Windows Presentation Foundation projects, 198
split view
Web Form Designer, 23-24
web pages, 521
SplitContainer class, 591-592
SQL Editor, 711-713
SQL scripts
auto-generating, 718-719
defined, 716
executing, 719
SQL Server
data tier (Team Foundation Server) and, 898-899
projects, creating, 720
required for database projects, 1118
SQL statements
queries
types of, 709-710
writing, 706-709
stored procedures, 710-713
triggers, creating, 714-715
user-defined functions, creating, 715-716
views, creating, 710
squiggles, 254-255
stacking behavior of actions panes, 871-872
StackOrder property (actions panes), 871-872
StackPanel control (WPF), 630-631
standard controls, 563-565
standard dialog boxes, 12
Standard edition (Visual Studio 2008), 48
Standard toolbar, 64-65
standards
compliance (HTML designer), 205
form design, 579
Start Action section
build options for website projects, 514-515
start options of website projects, 517
Start Debugging command, 336
start options for website projects, 517
Start Page, 57, 212-217
StartCaching method, 873
starter kits, 228, 502
starting
debg mode, 317-320
New Load Test Wizard, 1100
threads in MSDN Community Forums, 220
StartPosition property (forms), 582
startup forms, setting, 581-582
startup options (IDE), 57-58
Startup Project property page, 114-115
startup projects, 114-115, 142
state machine workflows, 834
designing, 834-836
host and client, defining, 844-849
initializing and transitioning states, 837-844
running, 850

How can we make this index more useful? Email us at indexes@samspublishing.com
state maintenance, ASP.NET controls, 562
state-machine workflows, 39
StateFinalization activity, 810, 837-844
StateInitialization activity, 810, 837-844
statements (SQL). See SQL statements
states (work items), 960
static code analysis in Visual Studio Team Developer, 888
status bars, creating, 600-602
StatusStrip controls, 600-602
Step Into command, 334-338
Step Out command, 338-339
Step Over command, 338
stepped load pattern (load tests), 1101
stepping into WCF services, 357
stepping through code, 320, 325, 327, 334
  Break All command, 337
  Continue command, 339
  ending session, 339
  Run To Cursor command, 335-336
  Start Debugging command, 336
  Step Into command, 334-338
  Step Out command, 338-339
  Step Over command, 338
Stop statement, 180
stored procedures, 710-713
  creating in C#, 720-723
storing images in image viewer application example (WPF), 641-643
Style Builder dialog box, 187
style rules, 534
  adding to CSS, 187
  creating, 536
style sheets
  adding to master pages, 542
  creating, 535
  CSS editor, 186-187
  themes versus, 544
  for user interfaces, 533-541
  XSLT style sheets, running against XML, 186
styles, 24-25
  applying, 539
  ASP.NET controls, 563
  building, 536
  editing, 539-541
  inline styles, 533
  managing, 537
  page-level styles, 533
  web form applications, 204
  in WPF, 631-632
stylesheets, 24-25
subclassing controls, 607
subreports, 1024, 1031-1032. See also views, creating
suggestions, sending via Help menu, 223-224
Summary subreport, 1031
Surround With snippets, 271
.svc file extension, 511
switching
  between windows, 10-11
  threads, 361
Sync Services for ADO.NET, 100
synchronization, 46-47
syntax coloring (productivity tool), 255-256
system definition model (SDM), 1050
System Designer, 1059-1062
system diagrams, 1048, 1059-1060
  connecting applications to systems, 1061-1062
  item templates for, 1049
  proxy endpoints, 1061
systems, connecting applications to, 1061-1062

T
T-SQL Editor in Visual Studio Team System Database Edition, 891
tab controls, 609
tab order, 595
TabContainer control, 665
TabControl class, 609
Table control, 565
Table Designer, 700-701
Table menu, 63
Table ratios in data generation plan, 1143-1144
TableLayoutPanel class, 590-591
tables, 697. See also databases
adding to database diagram, 702
defining, 700-701
editing definitions, 703
HTML tables, 202
joins, creating, 708-709
relationships, building, 703-706
showing contents of, 709
viewing, 702
views, creating, 710
Tables and Columns dialog box, 703
tabs, customizing Toolbox with, 67
Tabular data (form controls), 605-606
tag navigator (productivity tool), 259
Target Framework section (build options for website projects), 515
target source language for website projects, 120
targets (performance sessions), 1023
Task List, 280-283
Task List window, 384
task panes, 855
customizing, 863-864
task scheduler service, 898
task work items
MSF for Agile, 953
MSF for CMMI, 954
TaskItem object, 384
TaskList object, 384
tasks, 280-281
comment tasks, 281-282
custom tasks, extending builds, 1166
shortcut tasks, 283
sorting, 281
user tasks, 283
TCP port sharing, enabling, 796
Team Architect. See Visual Studio Team Architect
Team Build client, 1154-1155
Team Build Explorer, 1168-1169
Team Build Service, 898
team collaboration. See VSTS (Visual Studio Team System)
Team Explorer, 965-970
Team Explorer window, 905-907, 913, 1155
Team Foundation Administrators group, 899
Team Foundation Build, 893, 1153-1154
architecture of, 1154-1157
builds
creating, 1158-1159, 1162-1164
editing, 1165-1166
monitoring/analyzing, 1167-1171
running, 1166
Team Build Service, 898
Team Foundation Server (TFS), 51, 892-895
alerts, 917
application tier, 896-898
role in Team Foundation Build architecture, 1154-1156
connecting to, 912-913
data tier, 898-899
role in Team Foundation Build architecture, 1155-1157
installing, 901
Microsoft Office and, 915-916
project portal, 914-915
reports, 917-919
security, 899-901, 907, 909
source control. See source control
Team Explorer window, 913
team projects
adding custom groups, 909
adding users to, 905-908
controlling structure and iterations, 910-912
creating, 902-904
permissions, 909
Team Foundation Source Control (TFSC). See source control
Team Foundation Valid Users group, 899
team model
MSF Agile, 883
MSF for CMMI, 885

How can we make this index more useful? Email us at indexes@samspublishing.com
team project folders, 906

team projects. See also projects
adding custom groups, 909
adding users to, 905-908
builds. See builds
controlling structure and iterations, 910-912
creating, 902-904
naming, 902
permissions, 909

Team System. See VSTS

Team Systems products (Visual Studio 2008), 50-51
template code, 229, 268-269
Template projects (Office), 34
templates. See also item templates; project templates
ASP.NET controls, 563
class templates, 492
WCF services, 782-784
web service project templates, 754-756
for website projects, selecting, 501-502
workflow item templates, 806-809
workflow project templates, 803-804
in WPF, 632-635
temporary breakpoints. See Run To Cursor command
temporary macros, 420-421
Terminate All command, 339
test configuration file, 1086
Test Generation Settings dialog box, 1009
test items, creating, 1082-1083
Test Load Agent, 1113
Test menu, 63
test mixes for load tests, selecting, 1102-1103
Test Project template, 1080
test projects, creating, 1080-1082
test results, viewing, 1012
tester role
Visual Studio Team Test, 890
with work items, 983-985
testing, 890, 1079. See also debugging
build tests, selecting, 1161
generic tests, 1110
load tests, 1100
creating, 1100-1106
editing, 1106
running, 1107-1108
manual tests, 1109-1110
mobile device applications, 36-38
ordered tests, 1112
unit tests, 1004-1005
assertion methods, 1007-1008
attribute classes, 1007
best practices, 1006-1007
code coverage analysis, 1014-1015
creating, 1008-1009
for database projects, 1137-1140
running, 1010-1012
sample unit test, 1006
in Visual Studio Team System Database Edition, 891
WCF services, 788
web tests, 1086
extracting values from, 1096-1097
managing test requests, 1088
recording, 1086-1087
running, 1089
seeding with data, 1089-1096
validation rules, 1097-1098

Testing Policy, 935
text
adding to text documents, 405
selecting, 160
text documents, 400-402
adding text to, 405
distinguishing from nontext documents, 401
editing, 402-403, 406-408
text editor, 70. See also code editor
customizing, 72
defined, 159
in Macros IDE, saving project files, 428
productivity tools
active hyperlinking, 255
change tracking, 253-254
coding problem indicators, 254-255
IntelliSense. See IntelliSense
syntax coloring, 255-256
selecting text, 160
virtual space, 162
word wrapping, 160-162
Text Editor Options dialog box, 161
text windows, 381-382, 408
TextBox control, 564
TextDocument object, 400-402
TextPane object, 381-382
TextPoint object, 402, 405
TextSelection object, 403
TextWindow object, 381-382
TFB. See Team Foundation Build
TFS. See Team Foundation Server
TFSBuild.proj file, editing, 1165
TFSC (Team Foundation Source Control). See source control
TfsWarehouse relational database, 918
themes
applying, 547-550
creating, 544-545
for user interfaces, 544-550
think times in load tests, 1100
threads
flagging, 358-359
managing, 360-361
managing alerts in MSDN Community Forums, 221-223
multithreaded applications, debugging, 358-361
naming, 359
.NET Framework 3.5 enhancements, 99
starting in MSDN Community Forums, 220
switching, 361
viewing, 358-359
Threads window, 360
throwing web service exceptions, 779-780
time zones, 100
timer control, 656
tokens, custom comment tokens, 282
tool windows, 377, 383-384
displaying for color palette example (add-ins), 467-468
linking/unlinking, 392-393
toolbar buttons
Class View, 142
Document Outline, 157
Performance Explorer, 1016
Solution Explorer, 139
toolbars, 64-65. See also command bars
code editor, 168
creating, 598-600
customizing, 65-66
determining with CommandBar object, 397
Object Browser, 153
running macros from, 436-437
source control, 926-927
Workflow toolbar, 805-806
Toolbox, 67, 585
Application Designer, 1053-1054
code snippets in, 277
customizing, 67, 585
Logical Datacenter Designer, 1064
Windows Forms designer, 190
ToolBox object, 386-387
Tools menu, 63
ToolStrip controls, 596-602
ToolStripContainer class, 592-594, 600
ToolStripItem class, 598
ToolTip class, 595
ToolTip property, 595
ToolTips, 595
tracepoints, 340, 348-349. See also breakpoints
tracing, 99
tracking work items in Team Foundation Server, 893
tracks
MSF Agile, 883
MSF for CMMI, 884
transforms of images, 646
transitioning states, 837-844

How can we make this index more useful? Email us at indexes@samspublishing.com
transitions (work items), 960

tree view
   Class View, 144-145
   Object Browser, 153
   Performance Explorer, 1017
   Solution Explorer, 136

TreeNode Editor, 603-604

TreeView control, 571, 603-604

triggers
   build triggers, setting, 1163-1164
   creating, 714-715

troubleshooting. See debugging; testing
tunneling events (WPF), 637

type inference, 83-85
typed datasets, 43

U

UDDI (Universal Description, Discovery, and Integration), 753

UI. See user interfaces

Uniform Resource Identifier (URI), 753

unit tests, 1004-1005
   assertion methods, 1007-1008
   attribute classes, 1007
   best practices, 1006-1007
   code coverage analysis, 1014-1015
   creating, 1008-1009
   for database projects, 1137-1140
   running, 1010-1012
   sample unit test, 1006
   in Visual Studio Team Developer, 889
   in Visual Studio Team System Database Edition, 891
   in Visual Studio Team Test, 890

Universal Description, Discovery, and Integration (UDDI), 753

unlinking tool windows, 392-393

unnamed skins, 546

unshelving files (source control), 941

update script

data, 1134
schemas, 1129-1130

UpdatePanel control, 655

UpdateProgress control, 656, 659-660

updating
   GridView control data, 738-740
   web references, 776

uploading process templates, 990-991

URI (Uniform Resource Identifier), 753

URLs for RSS feeds, 216

usability, form design and, 578-579

user application development, new features, 17-18
   mobile device applications, 36-38
   VSTO (Visual Studio Tools for Office), 32-35
   web development productivity features, 18-27
   WPF (Windows Presentation Foundation), 27-32

User Control Designer, 607-610

user controls, 206, 573, 575
   creating for color palette example (add-ins), 462-466
   designing, 607-610
   event handling for color palette example (add-ins), 468-469
   integrating for color palette example (add-ins), 466-469

user interface controls (ASP.NET), 20-23

user interfaces
   consistency, maintaining, 532-550
   creating user-configurable UIs, 550-560
   designing, 577-580
   master pages, 541-544
   page layout, 528-531
   planning, 526-528
   style sheets, 533-541
   themes, 544-550

user tasks, 283

user-configurable UIs, creating, 550-560

user-defined functions, creating, 715-716

users, adding to team projects, 905-908

using declarations (C#), 81
validation
  ASP.NET controls, 563
controls, 565-567
  of deployment, 1073
  options (HTML designer), 204-205
  rules on web tests, 1097-1098
  with Visual Studio Team Developer, 888-890
ValidationSummary control, 567
variables. See also parameters
  implicit typing, 83-85
  local variables, 309-311
  watching, 350-352
VB (Visual Basic)
  code editor, 71
  IDE enhancements, 80-81
  language enhancements, 82-83
    anonymous types, 88-89
    business logic in partial methods, 91-93
    embedded XML, 96-97
    extension methods, 87-88
    friend assemblies, 95
    implicit typing, 83-85
    lambda expressions, 89-91
    LINQ queries, 94-95
    object initializers, 85-87
    removing arguments from event handlers, 97
  refactoring tools, 286
.vb file extension, 511
VBA (Visual Basic for Applications), 853
version control. See source control
version of .NET Framework, selecting, 8-10
View menu, 61
view windows. See browsers; explorers
viewing
  Code Metrics results, 1042-1043
  Component Designer, 525
debugging data, 350
  Autos window, 351
  DataTips window, 352-353
  Locals window, 350
QuickWatch window, 352
  visualizers, 354-355
  Watch windows, 351-352
derived classes, 998
  object definitions, 1128
  record-level details, 1132
  results (Code Analyzer), 1039
test results, 1012
  threads, 358-359
  web methods, 770-771
  web references, 776
  work items, 971
  WSDL, 765-766
views, creating, 710. See also subreports
Virtual Point object, 403
virtual space, 162
Visual Basic. See VB
Visual Basic for Applications (VBA), 853
Visual C# project definition files, 121-123
Visual C++ project definition files, 123-127
Visual Database tools, 697. See also names of specific tools (e.g. Solution Explorer, Server Explorer)
Visual Designers, 69. See also Class Designer
Visual Source Safe (VSS), 922
Visual Studio 2008
  community features
    Help menu, 217-225, 228
    Start Page, 212-217
  configuring for source control, 925
  environment settings, changing, 902
  IDE. See IDE
  installing, 53-56
.NET Framework integration, 7
new features, 8
  business process development, 38-42
  C# IDE enhancements, 81-82
  Class Designer, 16
  code sharing, 16
data access, 42-47
  docking windows, 11
  fonts, selecting, 12-13
  IDE Navigator, 10-11
  importing/exporting settings, 13-16

How can we make this index more useful? Email us at indexes@samspublishing.com
language enhancements, 82-97
.NET Framework 3.5 enhancements, 98-100
.NET Framework version, selecting, 8-10
standard dialog boxes, 12
for user application development, 17-38
VB IDE enhancements, 80-81
versions of
Express Editions, 47-48
Expression products, 51-52
Professional editions, 49
Standard edition, 48
Team Systems products, 50-51
Visual Studio add-ins, shared add-ins versus, 444
Visual Studio automation object model. See automation object model
Visual Studio Content Installer, 229-231
macro package example installation, 244-247
packaging shared content, 239-247
Visual Studio Database Edition projects, 1115-1116
building and deploying, 1149-1150
creating, 1117-1122
Data Comparison tool, 1131-1134
generating data, 1142-1149
rename refactoring, 1135-1136
Schema Comparison tool, 1126-1130
Schema View window, 1124
unit tests, 1137-1140
workflow of, 1116-1117
Visual Studio debugger. See debugger
Visual Studio Headlines section (Start Page), 214
Visual Studio Options dialog box, 279, 437
Visual Studio Professional with MSDN Premium, 49
Visual Studio Professional with MSDN Professional, 49
Visual Studio Team Architect, 887
application design steps, 1050-1051
application diagrams, 1051, 1054-1056
Application Designer Toolbox, 1053-1054
application settings and constraints, 1057-1058
Diagram menu, 1051-1053
application implementation, 1074-1077
deployment implementation, 1071-1074
item templates, 1049-1050
logical datacenter diagrams, 1062-1063
adding servers to zones, 1065-1066
configuring zones and servers, 1068-1071
connecting servers to zones, 1067-1068
Logical Datacenter Designer Toolbox, 1064
zones, 1064-1065
project templates, 1048-1049
system diagrams, 1059-1062
Visual Studio Team Developer, 888-890
Visual Studio Team Suite, 50
Visual Studio Team System. See VSTS
Visual Studio Team Test. See testing
Visual Studio Tools for Office (VSTO), 18, 32-35
visual styles (form controls), 594
visual trees in WPF, 632
visualizers (debugger), 354-355
visually developing code with Class Designer, 996-1004
vsCEMEElement enumeration values, 375
VSCContent files
creating for macro packaging example, 242-243
elements of, 240-242
.vsi files versus, 244
.vsd files, 489-490, 495
.vsi files
publishing, 247
renaming zipped files to, 244
signing, 246
.VSCContent files versus, 244
VSTO command-line tool, 1023
.vsmacros file, 424
VSS (Visual Source Safe), 922
.vssettings files, 16
.vstemplate XML file, editing, 236-237
VSTO (Visual Studio Tools for Office), 18, 32-35
VSTS (Visual Studio Team System), 881, 885-886
  source control. See source control
  Team Foundation Build. See Team Foundation Build
  Team Foundation Server, 892, 894-895
Visual Studio Team System Database Edition, 50, 890, 892, 1115
Visual Studio Team System Test Edition, 50, 890
work items, 950-951, 956
  areas, 956-958
  associating code with, 981-983
  attaching files, 965
  business analyst role, 979-980
  creating, 965
  customizing, 986, 991-993
  developer role, 980-983
  history tracking, 961
  iterations, 956-959
  linking, 961-964
  MSF for Agile work items, 951-953
  MSF for CMMI work items, 953-955
  project manager role, 972-978
  queries on, 967-970
  seeding methodology with, 987-991
  states and transitions, 960
  Team Explorer tools, 965-970
  tester role, 983-985
  viewing, 971
  workstreams, 972
.vsz files, 489-490, 495

W
WAS (Windows Process Activation Service), 799
Watch windows (debugger), 322, 351-352
watching variables
  Autos window, 351
  Locals window, 350
  QuickWatch window, 352
  Watch windows, 322, 351-352
WCF (Windows Communication Foundation), 41-42, 751-752
  clients, 781
  .NET Framework 3.5 enhancements, 99
  project templates, 782-784
  services
    configuring, 790-796
    consuming, 796-798
    creating, 784-788
    debugging, 357-358
    defined, 781
    hosting, 798-799
    running and testing, 788
    terminology, 780-782
WCF Configuration Editor, 790
WCF Service Library project, 783-784
WCF Service template, 501
web controls, data binding, 736-740
Web designer. See HTML designer
web development productivity features, 18-27
  AJAX Extensions for Visual Studio, 19-20
  ASP.NET user interface controls, 20-23
  master pages, 25-27
  Split view, 23-24
  styles/style sheets, 24-25
web form applications, designing, 200-201. See also HTML designer
  browser output, 204-205
  controls, 200-201
  editing markup, 202
  styles, 204
Web Form Designer, Split view, 23-24
web form developer panes, 24-25
web methods
  defined, 752
  invoking, 772
  viewing, 770-771
web pages
  adding controls to, 520-521, 662, 666, 668
  AJAX
    Accordian control, 663
    AJAX Control Toolkit, 660-662
    ASP.NET AJAX controls, 656
    AutoCompleteExtender control, 664
    CalendarExtender control, 664
    CollapsiblePanelExtender control, 663
    ConfirmButtonExtender control, 665
    FilteredTextBoxExtender control, 664
    HoverMenuExtender control, 664
    ListSearchExtender control, 664
    MaskedEditExtender control, 664
    ModalPopupExtender control, 664
    overview, 653-654
    partial-page update, 656-658
    PasswordStrength control, 665
    ReorderList control, 665
    ScriptManager control, 655
    ScriptManagerProxy control, 655
    SliderExtender control, 665
    TabContainer control, 665
    Timer control, 656
    UpdatePanel control, 655
    UpdateProgress control, 656, 659-660
  creating for website projects, 518-526
Web Part controls, 550-560
Web Part pages, creating, 552-560
web references, 776
Web Service Description Language. See WSDL
Web Service Details window (Application Designer), 1058
Web Service Enhancements (WSE), 753
web service methods. See web methods
web service process, attaching to, 323-324
web services, 749-752. See also WCF (Windows Communication Foundation)
  accessing, 764
  invoking web method, 772
  viewing web method, 770-771
  viewing WSDL, 765-766
  adding to workflows, 820-821
  in application tier (Team Foundation Server), 896-898
  calling, 777-778
  consuming, 772
  calling web services, 777-778
  defining service references, 773-775
  updating web references, 776
  viewing web references, 776
  creating, 757-764
  defined, 752
  exceptions, throwing, 779-780
  in .NET, 754
  project templates, 754-756
  settings, 1058
  terminology, 752-754
Web Site Administration Tool (WSAT), 568-569
web tests, 1086
  extracting values from, 1096-1097
  managing test requests, 1088
  recording, 1086-1087
  running, 1089
  seeding with data, 1089-1096
  validation rules, 1097-1098
  in Visual Studio Team Test, 890
WebMethod attribute class, 764
WebPartManager control, 550, 553
WebPartZone control, 551-553
WebService class, 763
Website menu, 61
website projects, 500
  accessibility options, 516
  build options, 513-515
  creating, 119-120, 500
  creating web pages, 518-526
  file types for, 510-511
  folders for, 508-510
MSBuild options, 518
Property Pages dialog box, 512-518
references, 512-513
selecting default programming language, 507
selecting template, 501-502
selecting website location, 502-507
start options, 517
websites, enabling/disabling debugging, 318-319
WF (Windows Workflow Foundation), 99, 802
When Breakpoint Is Hit dialog box, 348
While activity, 810
wildcards in Find and Replace window, 172
Window menu, 63
Window object, 377
  command window, 389
  linked windows, 392-393
  output window, 390-391
  properties/methods, 378-380
querying Windows collection, 380-381
referencing windows, 377-378
task list window, 384
text windows, 381-382
tool windows, 383-384
  Toolbox, 386-387
window panes, 381-382
windows. See also browsers; explorers
  in code editor, components of, 165-167
  in debugger, 322
docking, 11, 76-78
document windows, 377
floating, 78
linked windows, 392-393
pinning, 75-76
referencing, 377-378
selecting in code editor, 163
switching between, 10-11
task list window, 384
text windows, 381-382, 408
tool windows, 377, 383-384, 392-393
Windows Application project template, creating forms, 581-584
Windows collection, querying, 380-381
Windows Communication Foundation. See WCF
Windows Counters property page, 1021
Windows Events property page, 1021
Windows Forms (WinForms), 18, 32, 187-188
Windows Forms controls, 868-869
Windows Forms designer, 188-189, 585
  controls
    adding, 189-190
    arranging, 191-193
    resizing, 193
    customizing form appearance, 189
    productivity tools, smart tags, 261
    writing code in, 193-196
Windows Management Instrumentation (WMI) classes, 150
Windows Presentation Foundation. See WPF
Windows Process Activation Service (WAS), 799
Windows service applications, hosting WCF services, 799
Windows Vista support, 100
Windows Workflow Foundation (WF), 99, 802
WinForms (Windows Forms), 18, 32, 187-188. See also WPF designer
Wizard control, 565
wizards
  Add New Item wizard example, 490
    creating .vsz and .vsdir files, 495
    implementing Execute method, 491-495
    associating icons with, 490
    IDTWizard interface, 487-488
    purpose of, 441
    structure of, 487-490
    .vsdir files, 489-490
    .vsz files, 489-490
WMI (Windows Management Instrumentation) classes, 150
word wrapping, 160-162
Work Item Publishing Errors dialog box, 975
work items, 950-951, 956
  areas, 956-958
  associating code with, 981-983
  associating with check-ins, 938-939

How can we make this index more useful? Email us at indexes@samspublishing.com
attaching files, 965
business analyst role, 979-980
creating, 965
customizing, 986, 991-993
developer role, 980-983
history tracking, 961
iterations, 956-959
linking, 961-964
managing with Project and Excel, 915-916
MSF for Agile work items, 951-953
MSF for CMMI work items, 953-955
project manager role, 972
with Excel, 972-977
with Project, 978
queries on, 967-970
seeding methodology with, 987-991
states and transitions, 960
Team Explorer tools, 965-970
tester role, 983-985
in TFS (Team Foundation Server), 51, 893
viewing, 971
workstreams, 972
Work Items Policy, 935
work products (MSF Agile), 883
Workflow designer, 804-806
Workflow toolbar, 805-806
WorkflowRuntime object, 802
workflows, 38-40
activities, 809-811
components of, 802-803
for database projects, 1116-1117
defined, 801-802
item templates, 806-809
project templates, 803-804
sequential workflows, 811
CallExternalMethod activities, configuring, 816-818
CodeActivity activity, configuring, 818
debugging, 833
designing, 811-814
HandleExternalEvent activities, configuring, 821-823
host and client, defining, 824-832
IfElse activities, configuring, 819-820
InvokeWebService activity, configuring, 820-821
running, 832-833
state machine workflows, 834
designing, 834-836
host and client, defining, 844-849
initializing and transitioning states, 837-844
running, 850
Workflow designer, 804-806
workspaces, 927-930
workstreams, 972
WPF (Windows Presentation Foundation), 18, 27-32, 187, 613
applications
compared to XBAP 669-670
creating, 670-671
definition, 668
deploying, 676-679
security, 671-676
architecture of, 613-615
class library, 615-617
data binding in, 635-636
design tools, 30-31
image viewer application example, 638-639
binding to images, 643-645
button event handlers, 645
image effects, 646
image storage, 641-643
layout for, 639-641
path selection dialog box, 646-652
layout panel controls, 624
Canvas control, 624-626
DockPanel control, 626-627
Grid control, 627-630
StackPanel control, 630-631
.NET Framework 3.5 enhancements, 99
resources, embedding, 634
routed events, 637-638
styles, 631-632
templates, 632-635
visual trees, 632
XAML (Extensible Application Markup Language), 617-619

WPF Browser Application, 29

WPF designer, 619-620
creating Windows Presentation Foundation projects, 197-198
design pane, 620-621
Properties window, 621
XAML pane, 620-621
zoom control, 623
WrapPanel control (WPF), 631
writing code
  in code editor, 164-165
  component code, 207-208
  in Windows Forms designer, 193-196

WS-*, defined, 753
WSAT (Web Site Administration Tool), 568-569
WSDL (Web Service Description Language)
defined, 753
viewing, 765-766
WSE (Web Service Enhancements), 753

XAML (Extensible Application Markup Language), 28, 187, 617
collaboration features, 618-619
syntax simplicity of, 618
XAML pane (WPF designer), 620-621
XBAP (XAML browser applications), 29
  compared to WPF applications, 669-670
definition, 668
security, 671-676
XHTML compliance, ASP.NET controls, 562

XML (Extensible Markup Language)
  code snippets, creating, 271-276
defined, 752
documents, editing, 184-186
embedding in code, 96-97
XML editor, 183
  schemas, creating, 185
  XSLT style sheets, running, 186
XNA, 18
XSD (XML schema document)
  creating, 185
defined, 753
XSLT style sheets, 185-186

Y–Z

z-index, 531
z-order, 599
.zip files, renaming, 244
zipping files for macro packaging example, 244
zones, 551-553
  adding servers to, 1065-1066
  adding Web Parts to, 554-556
  configuring, 1068-1071
  connecting servers to, 1067-1068
defining, 553-554, 1064-1065
zoom control (WPF designer), 623

How can we make this index more useful? Email us at indexes@samspublishing.com