THE EXPERT'S VOICE® IN .NET

Visual Studio Condensed

For Visual Studio 2013 Express, Professional, Premium, and Ultimate Editions

YOUR QUICK-REFERENCE GUIDE TO THE FEATURES THAT MATTER MOST

Patrick Desjardins

Apress®

Contents at a Glance

Introduction

Visual Studio is a broad subject, but this book covers it in a concise way. In order to give you enough information to use Visual Studio effortlessly, this book covers a little bit of everything. Some features covered are limited to specific editions of Visual Studio; in those cases, the book marks the edition required with a banner at the beginning of the section. To avoid repetition, however, all features that are limited to the Express edition are specified, as are Premium and Ultimate edition features. For features available in several editions, only the lowest required edition is specified. For example, if a description is applicable to Premium and Ultimate editions, we will mark that feature as Premium.

This book does not cover Visual studio features for specific platforms or technologies in depth. Chapter 8 is dedicated to platform-specific Visual Studio features, but it discusses each one only briefly. Also, because the focus here is on the development environment itself, there are few examples of code, all of them written in C#. Even if you are a Visual Basic or C++ developer, this should not affect you because the focus is on Visual Studio.

This book contains information about how to use Visual Studio by explaining all its visual aspects and possible configurations. Then, it demonstrates Visual Studio's capabilities for development and debugging. Several tools to improve your code are discussed, along with testing tools. You will learn about collaborative tools that Visual Studio offers with Team Foundation Server. Finally we take a peek at framework-specific features and how to extend Visual Studio.

CHAPTER 1

Introducing Visual Studio

Visual Studio (VS) has been the single Microsoft Integrated Development Environment (IDE) since its release in 1995. Previously, Microsoft Visual Basic, Visual C++ and Visual FoxPro were three different software formats. Between 1995 and 1998, Microsoft released three versions of Visual Studio. In particular, Visual Studio 6.0 was one of the most popular development tools of its time, for four years until Microsoft announced its new orientation with the .NET Framework in 2002. VS 6.0 is the first version that has more than one edition. Further versions rapidly appeared, with a version in 2003, followed by 2005 with .NET Framework 2.0. Three years later the 2008 version came out, then 2010, 2012, and finally 2013. The latest editions offer more tools and features to improve developer capabilities.

In this chapter, you will learn the differences between all Visual Studio 2013 editions and get an overview of every visual component of this latest version of the Microsoft IDE.

Visual Studio Editions

Visual Studio 2013 became available on October 17, 2013. It is available in five editions that fulfill different needs.

- The **Express** edition cannot have any extensions and does not offer any satellite tools like profiling, testing, or architecting, and it has limited project templates.
- **Professional** is the entry-level version if you want more than just basic development. It can do unit testing but none of the other types of tests. You have no limitation on project templates.
- **Premium** lets you read architecture and modeling graphic files.
- To use all the architectural tools, you'll need the **Ultimate** version.
- For testers and QA, Visual Studio has a **test** edition that is very limited but allows for doing some test impacts and using the lab manager. This version is not discussed further in the following summary.

Different editions meet different needs, and I will explain the features in more depth throughout the book.

Visual Studio Express Edition

The Visual Studio Express edition is the free version of Visual Studio. It comes in different flavors. One is for the Web, one for Windows, one for Windows Desktop, and one for phone development. The Express edition allows you to start developing with Microsoft .NET technology and has limited features compared to any paid version. However, Express projects can be opened with all Visual Studio editions, and Visual Studio Express can be upgraded to any other edition at any time.

The difference between the Windows and Windows Desktop versions is that one is for Windows Store app development (supported by Windows 8 and later versions) and the other is for WPF, WinForm, and Win32 applications. The Web edition creates web applications, web APIs, or Azure applications.

Visual Studio Express projects can be used for commercial purposes. One major drawback of the Express edition that you need to consider is that you cannot use any add-ins or extensions. Popular tools like ReSharper or Web Essentials cannot be installed. Nevertheless, the NuGet tool (part of Express since 2012) can be used, and we will talk about it in Chapter 3.

A second detail to consider is that the type(s) of application you can create will be defined by the installed edition. For example, with the Visual Studio Web edition, you will not be able to create a Windows Phone application or a Windows WinForm application; Visual Studio will not offer the option in its Create menu to create a new project of a different type.

A third feature that does not come with the Express edition but does with all other editions is the Exception Assistant tool. The Exception Assistant is a message box that appears when an unhandled exception occurs while debugging. It displays the type of exception, the stack trace, the inner exception, a help link for the error, an error message, and additional information to help you identify why the code is in an exception state. This scenario occurs in situations where you might not have considered an exception scenario and having additional information about it is precious. Visual Studio Express, since version 2010, does not provide this dialog box.

Visual Studio Express has its own application ID. This means it can be installed in parallel with other Visual Studio editions. This is not the case with any other edition of Visual Studio. For example, if you have the Professional edition, you won't be able to install the Premium edition side by side. However, you will be able to have Visual Studio Express Web edition in parallel with Visual Studio Express Desktop edition or Ultimate edition.

■ **Note** You can install Visual Studio 2013 in parallel with Visual Studio 2012, 2010, 2005, and so on because each of these versions has a different application ID.

Finally, keep in mind that this version is primarily intended for trying out the base IDE tools, or for the hobbyist developer. Visual Studio Express edition cannot use Microsoft Unit Testing Framework. It has limited project templates, and its integration with MSDN is limited.

Visual Studio Professional Edition

Microsoft Visual Studio Professional Edition is the entry point of the full Visual Studio IDE. With this edition and above, it is possible to create every type of application within the same IDE. This mean you can create Windows WinForms, Web Form applications, ASP.NET MVC, Windows services, WCF services, phone applications, Windows 8 applications, and so on.

This edition is perfect for the developer who wants to code applications without needing too many features. Basic features like project templates, debugging tools, page inspector, static code analysis, and Windows 8/phone simulator are all available. Almost all IDE features are available except the CodeLens and the Code Clone feature.

Visual Studio has a lot of architecture and modeling tools available. Visual Studio Professional does have some of them. It has the ability to show code dependencies with dependency graphs. It can also create a code map (in read-only format).

Visual Studio Professional's feature limitations start with the testing tools that we will discuss later in Chapter 6. The Professional edition allows only unit testing, while other versions allow for more advanced testing. Its online service also has limitations. It has almost every feature that the Visual Studio Online Edition's Basic and Pro versions have, but not as many as the Advanced version. For example, it cannot handle agile portfolios or work with team rooms. But almost all other features are available, including the use of work items, sprint planning, repositories like TFS or Git, and Team Explorer, which can be used within Visual Studio or with a plug-in for Explorer. However, it cannot use the team review features.

Most of the missing features should not be a showstopper for most developers, and usually this is the edition most developers should be using. It costs a fraction of the price of the Premium and Ultimate editions, and it still offers many tools.

Visual Studio Premium Edition

Visual Studio Premium edition sits between the Professional edition and the Ultimate edition. It has everything the Professional edition has with additional features. This version is for the advanced developer or for the architect of your team.

This is the first edition to have Code Metrics. This is an essential tool to improve your application quality. It also has the Code Coverage tool, Microsoft Fakes for unit testing, and UI testing tools. It is also the first edition to have the Code Clone feature but still doesn't have the CodeLens tool.

For architecture and modeling, the Premium edition adds the architecture validation feature and has UML compliance diagrams such as the activity diagram, use case, sequence, and component. However, the UML diagrams are in read-only format, while all others can be read/write. This is a step better than the Professional edition for code mapping and the dependency graph.

The team foundation is enhanced for the Premium version and has everything the Ultimate version offers. It is possible to manage development tasks and features like agile portfolios, team rooms, work item charting, release and backlog management, sprint planning, Kanban or agile task boards, TFS or Git source control, work item tracking, and build automation.

Visual Studio Ultimate Edition

Visual Studio Ultimate edition is the most complete of all Visual Studio editions. It is the version in which everything is possible, all features are available, and no restrictions apply. Everything is read/write, and it has additional features that we have not yet covered. This version is often used by the team lead or the head architect.

It is the first edition to have IntelliTrace for debugging, which is also available in production. It is also the only version that allows .NET memory dump analysis and coding map debugger interaction. Since 2013, CodeLens is also a new feature available only in this version.

For code architecture and modeling, Ultimate provides the Architecture Explorer and the architecture and layer diagrams. These two features are only available for the Ultimate version in write mode. Premium can read files generated by Ultimate but no other edition.

The Ultimate edition has one feature that no other Visual Studio version has, which is a cloud-based web load and performance test. No version of Visual Studio Online has this feature or any other Visual Studio version. This requires having an Azure account, and it is a feature that we do not use often.

■ **Note** Throughout this book, I will indicate which edition at a minimum is required for the feature discussed. If nothing is specified, you can assume it is available in all editions beginning with the least: the Express edition.

Visual Studio Online

A brand-new addition in 2013 is Visual Studio Online. It is a light version of Visual Studio that was formerly named Team Foundation Services. It is important to know that Visual Studio Online does not replace the more traditional editions of Visual Studio. It allows having a source control with Team Foundation Services (TFS) but also offers work items tracking, agile planning, and build and load testing services. So far, unlike Visual Studio itself, Online does not let you write code for every project type. Nevertheless, Visual Studio Online is the first edition to offer a coding environment for the cloud (if only for Microsoft Azure). It is very lightweight at this moment. The name of the online Visual Studio code editor is Monaco. The online edition allows for the creation of the desktop application as a web application or Azure application.

Visual Studio Online is available in three subscription plans: **Basic**, **Professional**, and **Advanced**. Online Basic is free until you need more than five developers in the project. If you need more than five users, a fee by user is charged. Online Basic works with Visual Studio Express Edition for Web, Windows, or Windows Desktop. Online Professional is distinct from the others, with a limit of ten developers. It also comes with a monthly subscription to the Visual Studio

Professional IDE, which is a major plus. The Advanced plan is aimed at a larger project with several teams. It provides a real-time update on all your teams and allows you to be more efficient with the communication and management of all your developers. It is integrated with Visual Studio and also with Eclipse and Xcode. Unlike Basic and Professional, the Advanced plan allows an unlimited number of developers.

The benefit of choosing an online plan is that you can simply manage your team without having to configure TFS and Visual Studio yourself. As you will see soon, it is possible to work online with a traditional version of Visual Studio. In fact, all other editions of Visual Studio share the online part of the online plan. The main difference is the payment; it is monthly for the online plans and a onetime payment for the full edition. However, all Microsoft Visual Studio versions allow you to try them free of charge.

Table 1-1 compares the features of the three Visual Studio Online plans.

Table 1-1. *Comparison of Visual Studio Online Plans*

Features\Division	Advanced	Professional	Basic
Maximum number of users with this plan on an account	Unlimited	10	Unlimited
Unlimited team projects and private, hosted code repos using TFVC or Git	X	X	X
Project planning and bug tracking tools, including Kanban boards and team velocity forecasts	X	X	X
Integration with popular development tools, including Visual Studio, Eclipse, and Xcode	X	X	X
Work in one IDE to create solutions for the web, desktop, cloud, server, and phone	-	X	-
Support for Office 365 business apps	-	X	-
Track complex projects with hierarchical portfolio backlogs	X	-	-
Visualize project data with work item chart authoring	X	-	-
Send feedback requests to users and stakeholders	X	-	-
Manage and run tests and test plans	-	-	-
Host team projects on-premises as well as in the cloud	-	-	-

■ **Note** For more details about prices and the differences between online editions, see Visual Studio's website at `http://www.visualstudio.com/products/visual-studio-online-overview-vs`.

Guided Tour of the Visual Studio UI

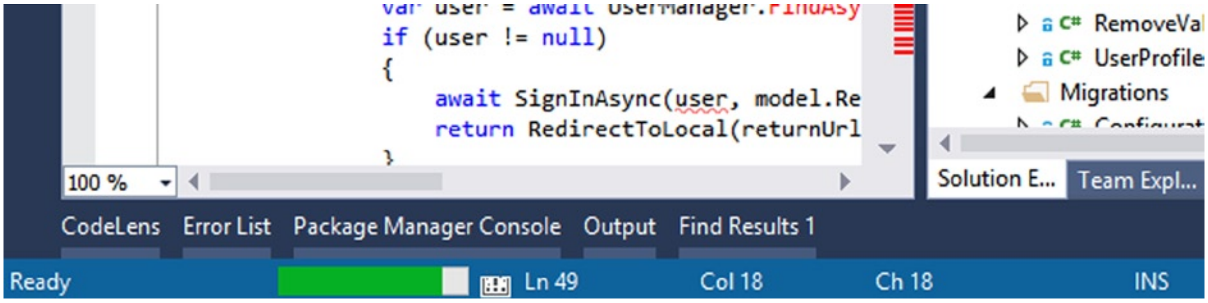
Visual Studio has been enhanced a lot since its creation. The software is composed of several portions that are worth exploring. This section explains most of the useful parts of the IDE.

The General UI

Before going deeper into Visual Studio's 2013 features, let's do a quick overview of its user interface (UI).

■ **Note** Because the Ultimate edition includes every Visual Studio feature, it will be used for screen shots throughout this book.

First, the Visual Studio menu is at the top of the screen. It offers different possible actions grouped by theme. For example, the main menu has **File**, **Edit**, **View**, **Project**, **Build**, **Debug**, **Team**, **Tools**, **Test**, **Architecture**, and **Analyze** tabs that all contain submenus. Second, toolbars are located under the menus. You can configure toolbars by adding tools or removing buttons and bars as you wish. Toolbars must always be located under the menu and over the main code editor. This means that you can move toolbars to specific locations, but you cannot move them just anywhere. The third part of the UI is the code editor, located between the toolbar and the footer. The footer, shown in Figure 1-1, provides information about the state of the compilation and the position of the caret.



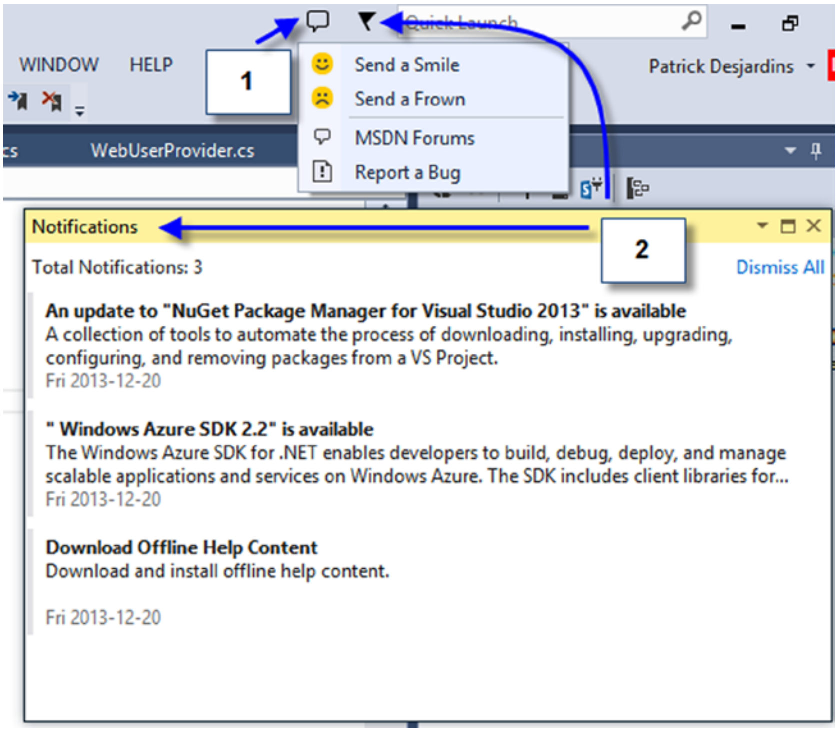
```
var user = await UserManager.FindAsy
if (user != null)
{
    await SignInAsync(user, model.Re
    return RedirectToLocal(returnUrl
}
```

Figure 1-1. *Footer information*

Figure 1-1 shows that the current state is Ready, which tells you that it is ready for the next project to be compiled in this example. The second indicator displays a progress bar and appears only when compiling or publishing. Following that, the icon that looks like a box indicates what type of progress is occurring. Figure 1-1 has the icon for compilation. Next to this icon is information about the caret position. It indicates that it is located on line 49, column 18, which is also character 18. Finally, on the far right is the keyboard mode, which is currently Insert (the alternative would be Override).

■ **Note** You can double-click the line, column, or character in the footer to open the *Go To Line* dialog or press the shortcut, Ctrl+G.

Visual Studio offers an easy way to communicate feedback to Microsoft. At the top right of the screen, a small shootout icon is available. Click it to send feedback with an attached screenshot of the current state of your Visual Studio. Label (1) in Figure 1-2 highlights the **feedback** feature.



Total Notifications: 3

An update to "NuGet Package Manager for Visual Studio 2013" is available
A collection of tools to automate the process of downloading, installing, upgrading, configuring, and removing packages from a VS Project.
Fri 2013-12-20

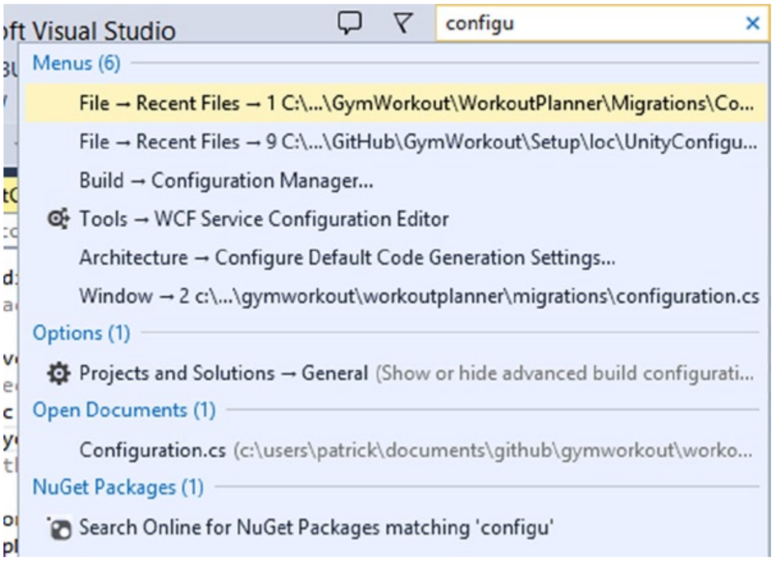
" Windows Azure SDK 2.2" is available
The Windows Azure SDK for .NET enables developers to build, debug, deploy, and manage scalable applications and services on Windows Azure. The SDK includes client libraries for...
Fri 2013-12-20

Download Offline Help Content
Download and install offline help content.
Fri 2013-12-20

Figure 1-2. *Feedback and notifications*

The top menu also contains an icon in the shape of a flag (2). This one is for available notifications. If you click the flag, the notification panel opens. If the notification flag is black, a new notification is available. If the flag is white, no notification is available.

Finally, the top right contains a **Quick Launch** textbox. If you start typing, a context panel appears with related information. Quick Launch provides a search within all code file names, build configurations, tools, architecture, and modeling files. It also allows you to jump to every configuration possible inside Visual Studio, and open documents and even NuGet packages. Figure 1-3 shows its results.


configigu

Menus (6)

File → Recent Files → 1 C:\...\GymWorkout\WorkoutPlanner\Migrations\Co...
File → Recent Files → 9 C:\...\GitHub\GymWorkout\Setup\loc\UnityConfigu...
Build → Configuration Manager...
Tools → WCF Service Configuration Editor
Architecture → Configure Default Code Generation Settings...
Window → 2 c:\...\gymworkout\workoutplanner\migrations\configuration.cs

Options (1)

Projects and Solutions → General (Show or hide advanced build configurati...

Open Documents (1)

Configuration.cs (c:\users\patrick\documents\github\gymworkout\worko...

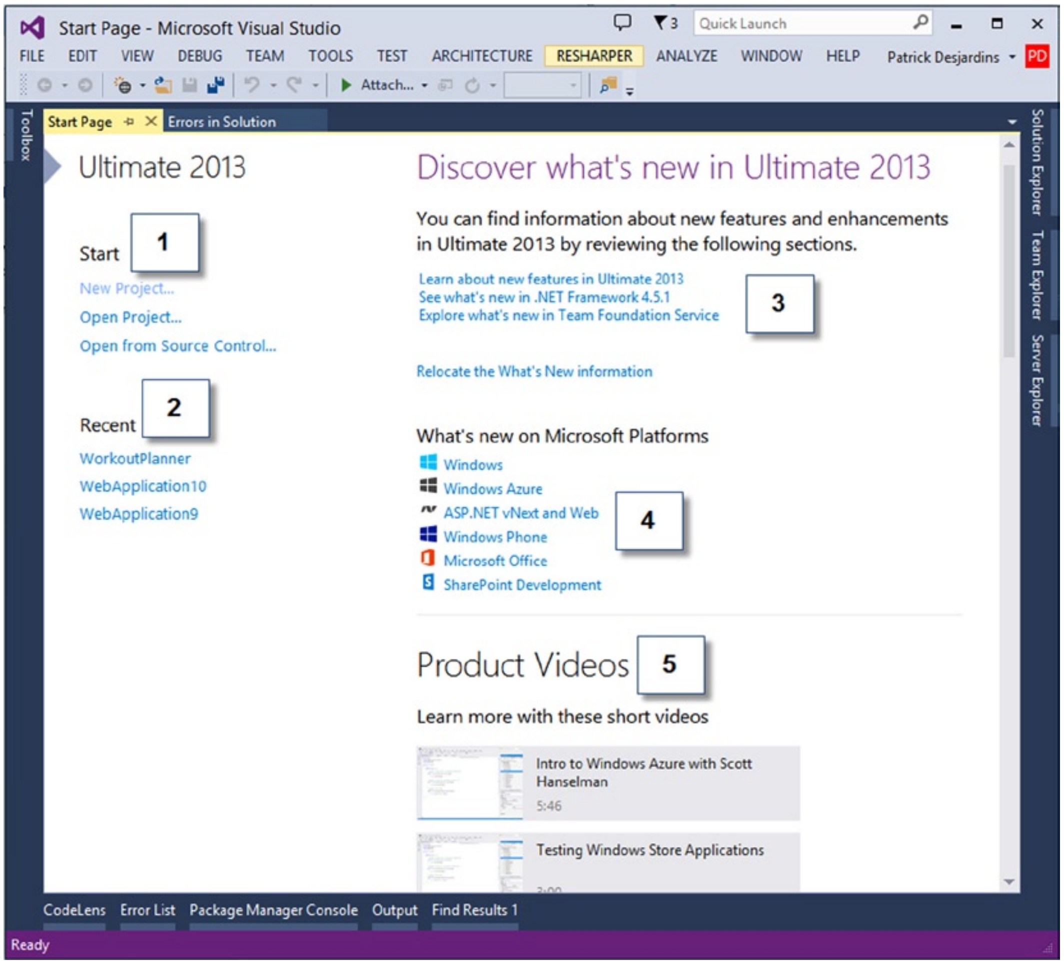
NuGet Packages (1)

Search Online for NuGet Packages matching 'configu'

Figure 1-3. *Quick Launch results*

The Start Page

Visual Studio consists of several panels and toolbars. When Visual Studio opens, the first panel displayed is the **Start page**, shown in Figure 1-4.


Start Page - Microsoft Visual Studio
Quick Launch
FILE EDIT VIEW DEBUG TEAM TOOLS TEST ARCHITECTURE RESHARPER ANALYZE WINDOW HELP
Patrick Desjardins PD
Attach...
Toolbox
Start Page
Errors in Solution
Solution Explorer
Team Explorer
Server Explorer
CodeLens Error List Package Manager Console Output Find Results 1
Ready
1
2
3
4
5

Ultimate 2013

Start

New Project...

Open Project...

Open from Source Control...

Recent

WorkoutPlanner

WebApplication10

WebApplication9

Discover what's new in Ultimate 2013

You can find information about new features and enhancements in Ultimate 2013 by reviewing the following sections.

Learn about new features in Ultimate 2013
See what's new in .NET Framework 4.5.1
Explore what's new in Team Foundation Service

Relocate the What's New information

What's new on Microsoft Platforms

Windows
Windows Azure
ASP.NET vNext and Web
Windows Phone
Microsoft Office
SharePoint Development

Product Videos

Learn more with these short videos

Intro to Windows Azure with Scott Hanselman
5:46

Testing Windows Store Applications
3:00

Figure 1-4. *Visual Studio Start page*

The Start page contains various types of information. In Figure 1-4, labels (1) and (2) point out the two portions of the Start page that developers use the most. The **Start** link at (1) lets you create a new project, open an existing one on a hard drive, or open an existing project from a source control. These functions are also available in the File menu (Figure 1-5).

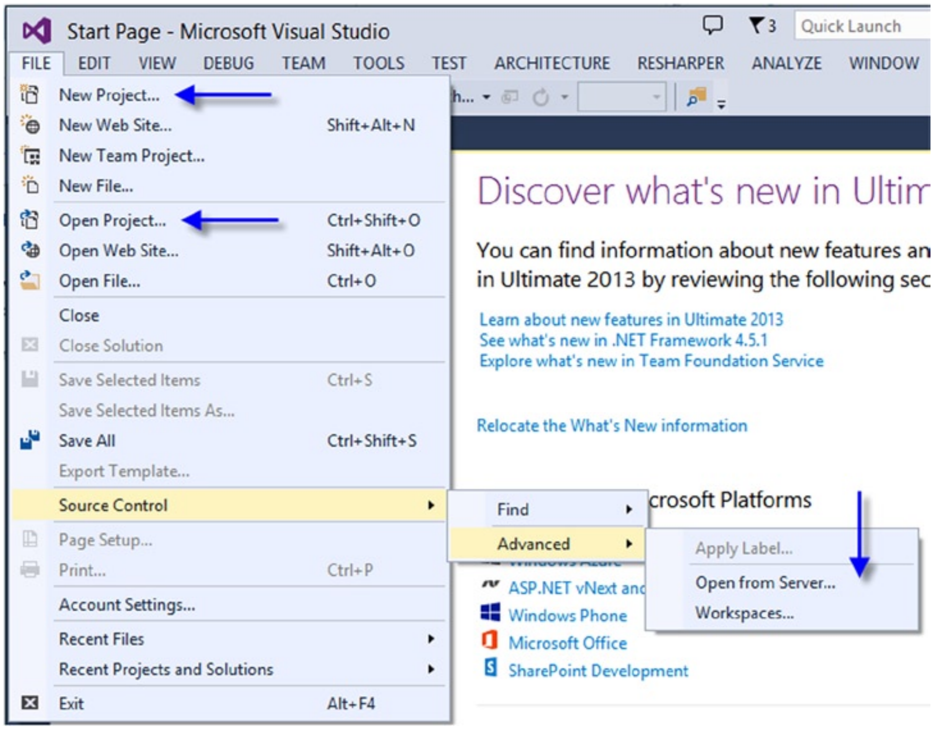

Start Page - Microsoft Visual Studio
Quick Launch
FILE EDIT VIEW DEBUG TEAM TOOLS TEST ARCHITECTURE RESHARPER ANALYZE WINDOW
New Project...
New Web Site... Shift+Alt+N
New Team Project...
New File...
Open Project... Ctrl+Shift+O
Open Web Site... Shift+Alt+O
Open File... Ctrl+O
Close
Close Solution
Save Selected Items Ctrl+S
Save Selected Items As...
Save All Ctrl+Shift+S
Export Template...
Source Control
Page Setup...
Print... Ctrl+P
Account Settings...
Recent Files
Recent Projects and Solutions
Exit Alt+F4
Find
Advanced
Apply Label...
Open from Server...
Workspaces...
Discover what's new in Ultim
You can find information about new features an
in Ultimate 2013 by reviewing the following sec
Learn about new features in Ultimate 2013
See what's new in .NET Framework 4.5.1
Explore what's new in Team Foundation Service
Relocate the What's New information
crosoft Platforms
ASP.NET vNext and
Windows Phone
Microsoft Office
SharePoint Development

Figure 1-5. *To start working in Visual Studio, choose New Project, Open Project, or Open from Source Control*

The **Recent** link at (2) offers a history of the recent solutions that you have open. Often, developers work on the same solution for a while. Opening one of these solutions is a great shortcut to get up and running quickly when you need to open Visual Studio.

The links at (3), (4), and (5) in Figure 1-4 are always at the right side of the Start page. This part is dynamic and updated by Microsoft. At (3) you'll find links to websites for information about new features and enhancements of the edition you are working on. In the screen shot, it is news concerning the Visual Studio Ultimate edition. At (4), **What's New on Microsoft Platforms**, are links to Microsoft platforms. This information is rarely used compared to the **Product Videos** at (5). This displays a short tutorial video about Visual Studio tips and features. Under the Video section, Visual Studio also has an Announcements section, which reports an all-new version of Visual Studio and discusses what is coming out soon.

The Start page is a panel like most of Visual Studio's screens. All panels have the same common features: One feature is that they can be closed. To close a panel, the mouse needs to hover over the panel header or the panel needs to be the one that is active. In both scenarios, an X icon appears. In Figure 1-6, the Start Page tab is shown because it is open on the screen, but you can see that it is also activated because the focus is on it. This is not the case for the Errors In Solution panel to the right of the Start Page panel. This panel is open but not activated, and hovering with the mouse has no effect. To close a panel, select the X icon.

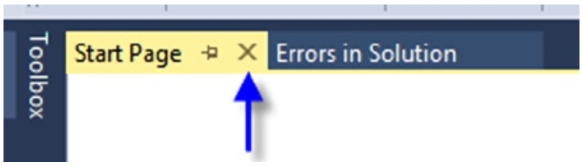
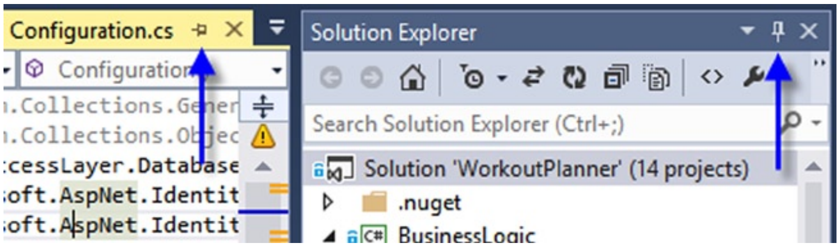


Figure 1-6. *Click the X to close a panel*

Pinning Panels

To activate a panel, just click on the caption. Two icons appear. One is the close icon, as we just saw, and the other is a pin icon (Figure 1-7). The pin icon is a second common feature that allows you to make a panel sticky on the screen. Otherwise, the default behavior is that once a panel loses its focus, it collapses to give more space. If you want to keep the panel open, click once on its pin icon.


Configuration.cs
Configuration
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'WorkoutPlanner' (14 projects)
.nuget
BusinessLogic

```
1.Collections.Gener
1.Collections.Objec
:cessLayer.Database
;oft.AspNet.Identit
;oft.AspNet.Identit
```

Figure 1-7. *Two different states of the pin icon*

Figure 1-7 shows at the left a C-Sharp file named `Configuration.cs` that is open but not pinned. This one stays open because it is a code file. Having a code panel pinned has no effect except that all pinned code files are grouped together. However, at the right of Figure 1-7, the Solution Explorer is pinned. This panel stays open even if the focus is on the left panel. Usefully, you can navigate quickly with the Solution Explorer while having the code editor open. Having a code file pinned will let you close all of the pinned or unpinned panels, depending on which option you choose. To access this option, right-click any code file and a context menu will appear, as shown in Figure 1-8.

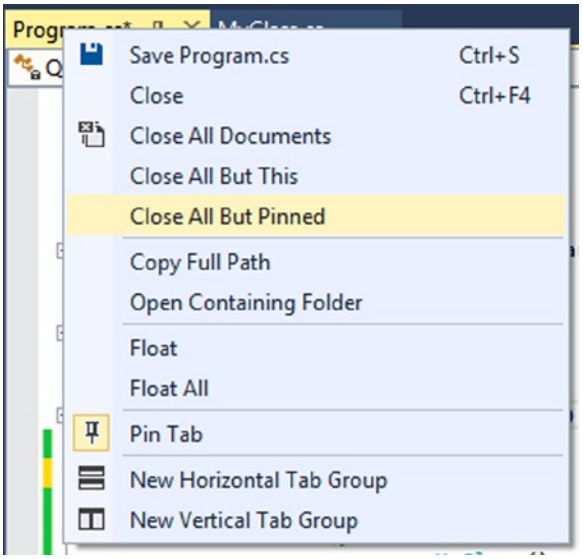
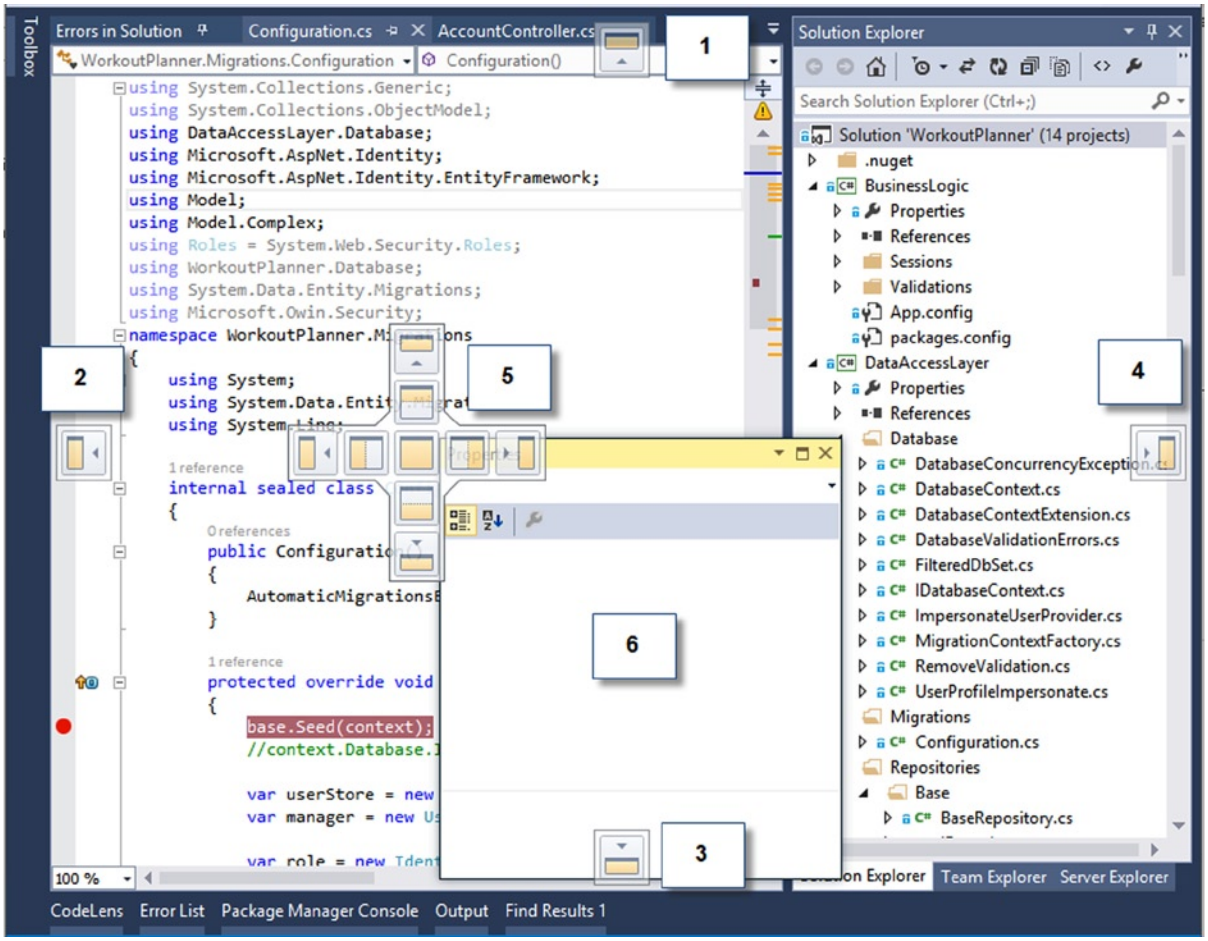


Figure 1-8. *Choose the Close All But Pinned option to view only a single panel*

Docking Panels

Panels can be docked. Visual Studio is a big container in which panels can be moved around and docked. A panel is docked when it is in one of six possible positions: left, right, top, bottom, center, or its own group outside Visual Studio.

To move a panel, click its caption and then drag it into one of the six locations highlighted in Figure 1-9.



```
using System.Collections.Generic;
using System.Collections.ObjectModel;
using DataAccessLayer.Database;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
using Model;
using Model.Complex;
using Roles = System.Web.Security.Roles;
using WorkoutPlanner.Database;
using System.Data.Entity.Migrations;
using Microsoft.Owin.Security;
namespace WorkoutPlanner.Mi      ons
{
    using System;
    using System.Data.Entit      rat
    using System.Linq;

    1 reference
    internal sealed class
    {
        0 references
        public Configurati
        {
            AutomaticMigrationsE
        }

        1 reference
        protected override void
        {
            base.Seed(context);
            //context.Database.I

            var userStore = new
            var manager = new U

            var role = new Ident
```

Figure 1-9. Docking positions

Once you have clicked the title of a panel and have started to move your cursor, keeping it pressed on the panel, you will see a docking indicator like those labeled in Figure 1-9. Clicking at (1) will dock the panel at the top of the screen. If you release the mouse at this position, the panel will be placed over the three panels open (Errors in Solution, Configuration.cs, and AccountController.cs). If you drop it at position(2), the current panel will be grouped with the Toolbox panel. By default, the panel will be open but once the focus is lost, the panel will be under the Toolbox caption. It is the same for the bottom position, (3), and the position at the right (4).

The placeholder at (5) is unique because it allows for nine different positions. The center one is the most used. In the example in Figure 1-9, the center position would put the panel with the three others on the main panel (Errors in Solution, Configuration.cs, and AccountController.cs). Those placeholders that are at each end create a new column of panels. For example, the one at the far right of (5) would create a unique panel group to the right of the existing one, which already contains three open panels (Solution Explorer, Team Explorer, and Server Explorer). On the other hand, an inner placeholder at (5) places the panel at the position desired within the center panel. The inner one is rarely used except when you want to have two code files open side by side.

The last docking position is shown at (6) but could be anywhere outside Visual Studio. Often, developers with more than one screen want to have a toolbox, Solution Explorer, or other panels on their second screen. This allows more space for the code and lets you have more panels open and pinned.

■ **Note** Usually developers have the Solution Explorer and Unit Test pinned on their screen.

Navigation

Visual Studio incorporates many features to help you work efficiently while developing. One main aspect where Visual Studio 2013 shines is navigating between files. This is important because developers must change task files hundreds of times every day. This section describes navigation tools that are available directly on the Visual Studio user interface. They are the **Solution Explorer**, the **Search Solution Explorer**, and the **Quick Launch**. Other features for navigation exist but are related to code directly and will be discussed later in the book.

The Solution Explorer is a panel (Figure 1-10) that has been available since almost the first version of Visual Studio. Searching within the panel now allows filtering. This is an important feature for a large solution. To access the Solution Explorer, choose **View ➤ Solution Explorer.**+

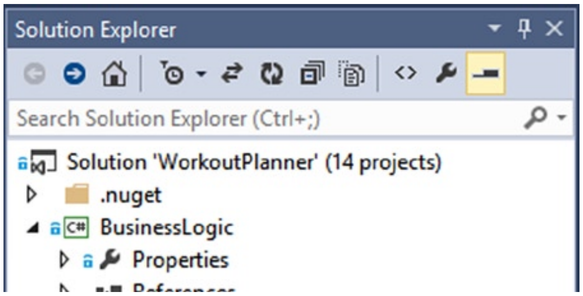

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'WorkoutPlanner' (14 projects)
.nuget
BusinessLogic
Properties

Figure 1-10. *Solution Explorer panel*

The **Solution Explorer** has its own toolbar that contains several icons:

- **Left and right arrow icons:** Allow you to move between previous searches.
- **House icon:** Toggles the navigation between a search and the initial state, which is to display every file and project.
- **Clock icon:** Allows a fast way to see only those files that have been modified since the last commit to the source control.

- **Double arrows icon:** Allows synchronizing the current open file with the Solution Explorer. This means that if you have a file open and you click this icon, the file name will be selected in the Solution Explorer, allowing you to navigate back to the file structures from a file that is open. This is a very useful command when you are navigating with other mechanisms and you want to come back to the Solution Explorer.
- **Round arrows icon:** Refreshes the file with the source control.
- **Multiple sheets icon:** Allows collapsing every project and folder. After you've been working for some time, solutions can become messy, so it's useful to be able to collapse everything for a cleaner view of the solution.
- **Double files icon:** Allows displaying files in the Solution Explorer even if they are not part of the solution. This lets you easily add files by right-clicking them, and you can add them to solutions without using Add Existing Item.
- **Less and greater icon:** Switches the file into the code editor. This is only useful when a file can be opened in either a designer mode or a code editor mode. For example, that is the case for the WinForm application.
- **Wrench icon:** Allows switching to the Property Panel for the active file. When the Property Panel opens, the properties displayed may be for the file, control, or HTML tag you have selected and not the file itself. This button automatically displays the properties of the file.
- **Toggle icon:** Accesses the **Preview Tab** feature that has the file open automatically when a file is clicked in the Solution Explorer. When a file is opened with the Preview Tab without being edited, it is closed automatically when the focus moves to another file.

The Solution Explorer has under its toolbar a textbox for the **Search Solution Explorer** feature (Figure 1-11). This tool performs a search within all files inside the Solution Explorer. The shortcut is Ctrl+;. You can search within the file contents and by file name. You can customize these options by clicking the arrow next to the search text box.

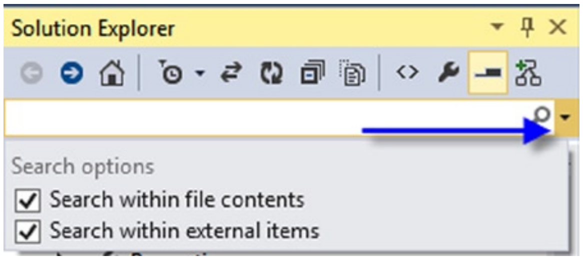


Figure 1-11. *Solution Explorer search*

The **Quick Launch** tool was briefly discussed earlier, but it allows for navigating, too. It is available at the top of the Visual Studio UI, or by pressing Ctrl+Q. You can navigate to the most recently used file by adding `@mru` followed by the keyword desired. You can also navigate to something inside a menu by adding a prefix to your search with `@menu`. The prefix to navigate directly to an option is `@opt`, to NuGet it is `@nuget`, and to search open documents it is `@doc`.

Properties

Visual Studio's Properties pane allows you to change a property depending on which element is active on the screen. The shortcut key is F4. If you select a file, the Properties pane will show the file properties. If you select a C-Sharp file, nothing should be displayed. However, if you select a file with a `.cshtml` extension, it gives you possible HTML attributes depending on where your cursor is located. If you select a resource file, properties for resource files are displayed, and so on.

Command Window

The Command window has been present for many years in Visual Studio and it allows you to execute commands directly in the IDE. To access the Command window, press Ctrl+Alt+A. You can also access it by choosing **Views ➤ Other Windows ➤ Command Window**. For example, if you want to build your entire solution, just type **`Build.BuildSolution`** and the solution is compiled.

Since Visual Studio 2010, the Command Window has provided IntelliSense, which helps you explore possible commands and eliminate the need to type everything. When debugging, IntelliSense can evaluate an expression if you use the question mark followed by a space and the name of the expression; for example, **`? variable1`**. It is possible to use the up and down arrow keys to navigate through the history of commands. You can also open a file by starting a command with the keyword `of` followed by a space and the file name. For more information, see the "IntelliSense" section later in this chapter.

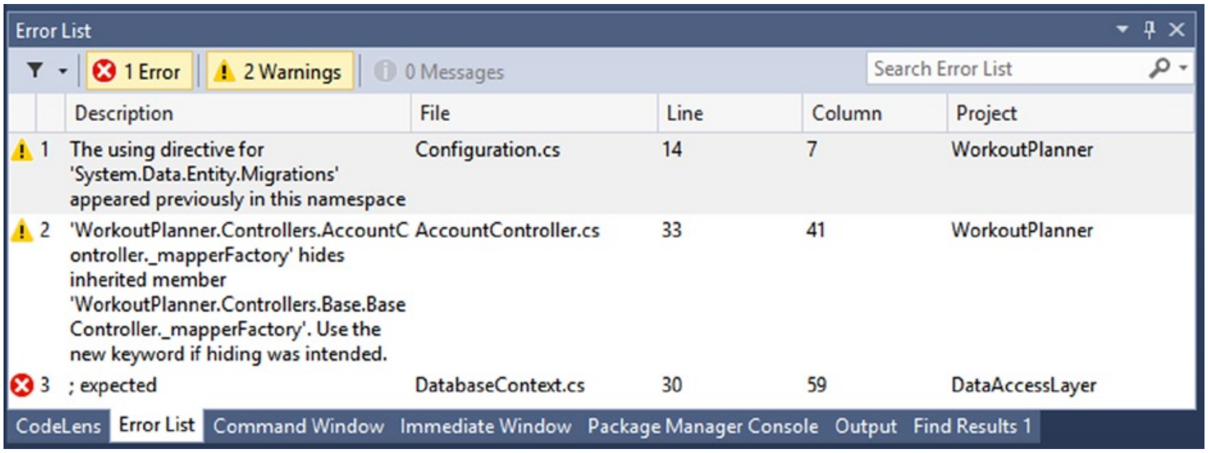
If you want to know the Visual Studio command aliases, like what the question mark does, you can enter **`alias`** into the Command window, and the full list of aliases and their related commands will be displayed.

Immediate Window

The **Immediate** window evaluates expressions when using the question mark, just like with the Command window. This time, only typing the variable name is enough. To access the Immediate window, press Ctrl+Alt+I or choose **Debug ➤ Windows ➤ Immediate Window**. The Immediate window is useful only at debug time and shows a subset of what the Command window can provide. It is a panel that you can pin when debugging and hide when you are developing.

Error List

The Error List panel is where every error, warning, and information message is listed (Figure 1-12).



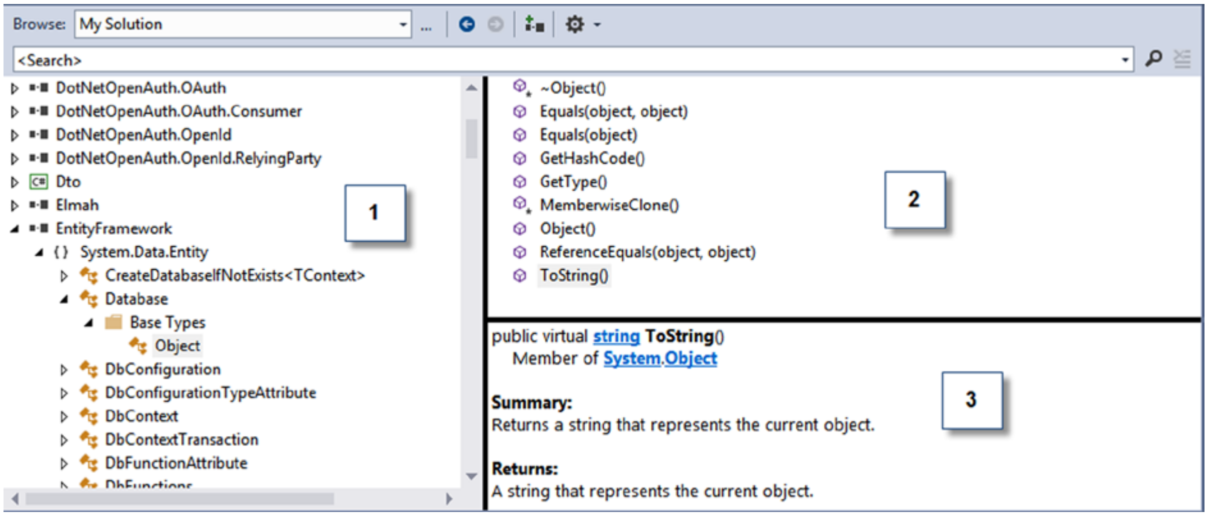
	Description	File	Line	Column	Project
⚠ 1	The using directive for 'System.Data.Entity.Migrations' appeared previously in this namespace	Configuration.cs	14	7	WorkoutPlanner
⚠ 2	'WorkoutPlanner.Controllers.AccountController._mapperFactory' hides inherited member 'WorkoutPlanner.Controllers.Base.BaseController._mapperFactory'. Use the new keyword if hiding was intended.	AccountController.cs	33	41	WorkoutPlanner
✖ 3	; expected	DatabaseContext.cs	30	59	DataAccessLayer

Figure 1-12. *Error List panel*

The funnel icon on this panel allows you to filter errors, warnings, and messages by project, document, or only open documents. Then, in the resulting set, it is possible to filter only errors or warnings or messages by choosing the type you want to filter. It is also possible to search through the list. The list is updated automatically for most errors, but some additional information can be discovered only at compilation time. This is one of the most useful panels and should be always open.

The Object Browser

The **Object Browser** (Figure 1-13) gives you information about the DLL of your project, DLLs referenced by your project, and any DLL or COM object on your computer. To access the Object Browser, choose **View ➤ Object Browser** or press Ctrl+Alt+J. By default, the preset shows only your solution, which lists all your projects and all the referenced DLLs.



public virtual string ToString()
Member of System.Object

Summary:
Returns a string that represents the current object.

Returns:
A string that represents the current object.

Figure 1-13. *Object Browser*

At (1) is the list of all libraries listed in the current solution, known as the **Object Panel**. It currently shows the Browse list for My Solution. The Entity Framework library is open for the `System.Data.Entity` namespace. The Database class is also open and its content is seen at the right. At (2) is the **Member Panel**. When a method is selected, the Description Panel (3) is open with a definition of the method.

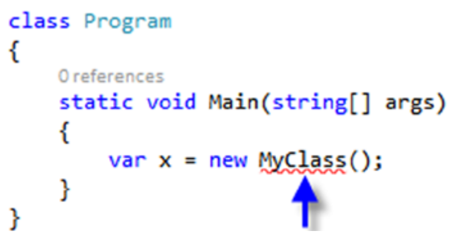
The Object Browser is useful for learning more about libraries but is used less and less now that it's possible to search on the Internet for richer information. The Internet contains examples of use, which the Object Browser does not. It is also less used because some other tools allow you to dig into the class of the variable. Even if the class is from a library outside your solution, the Object Browser can jump into the file definition or even be able to decompile the code. In both cases, however, you can have a good idea of the signature of classes without having to use the Object Browser.

Code vs. Debug View

Visual Studio's User Interface (UI) has two different modes: Code and Debug. It is important to be aware of this because you may be confused when Visual Studio is changing modes. When you open panels and docks at the location you desire, Visual Studio can save these placements depending on whether you are coding or debugging. This allows you to have different panels open depending on what you are doing. For example, you may not want to see the Code Stack open when developing but want this one open and pinned to be activated all the time when debugging.

Smart Tag

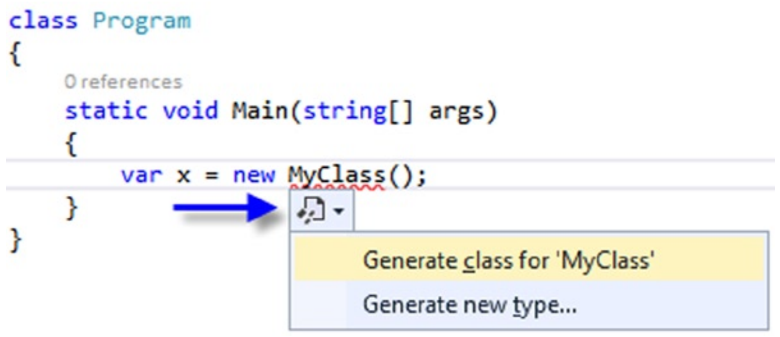
Visual Studio offers real time information about your code with smart tags. This feature first alerts you where it can make suggestions about your code by flagging those items with a red underline (Figure 1-14).



```
class Program
{
    0 references
    static void Main(string[] args)
    {
        var x = new MyClass();
    }
}
```

Figure 1-14. *Red underline highlights where a Smart tag will appear*

When you put your cursor over the word with this indicator, the smart tag appears, showing multiple possible actions (Figure 1-15). You can also open this list of possible actions by pressing Ctrl+.(period).



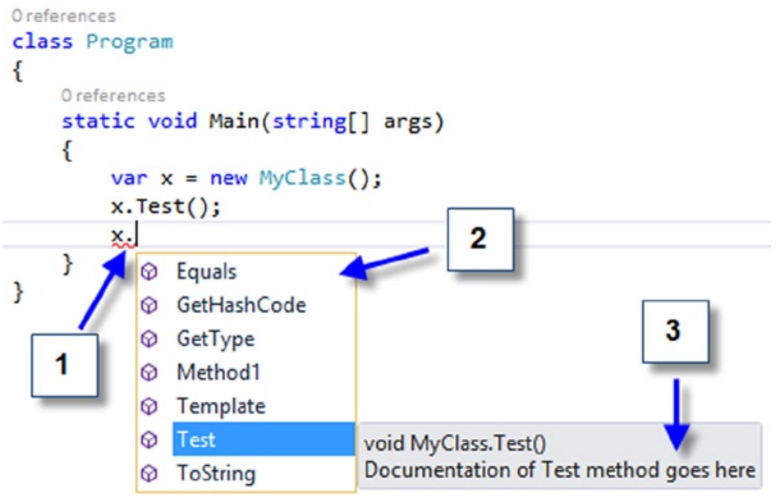
```
class Program
{
    0 references
    static void Main(string[] args)
    {
        var x = new MyClass();
    }
}
```

Figure 1-15. *Smart tag with two suggestions*

IntelliSense

IntelliSense is a Microsoft Visual Studio code completion feature. You can type just a few keystrokes and press Tab or Enter, and the complete word will be written for you. IntelliSense groups some other features that we will discuss here, like List Members, Parameter Info, Quick Info, and Complete Word. IntelliSense has been around since 1996 and has been improved to be more intelligent. It uses your current code but also draws on other libraries to figure out what variables, methods, or namespaces are available. Since the release of .NET, IntelliSense has triggered when the developer types; no shortcut is needed.

IntelliSense uses a feature called **List Members** (Figure 1-16). This displays every member of an object when the object is followed by a dot.



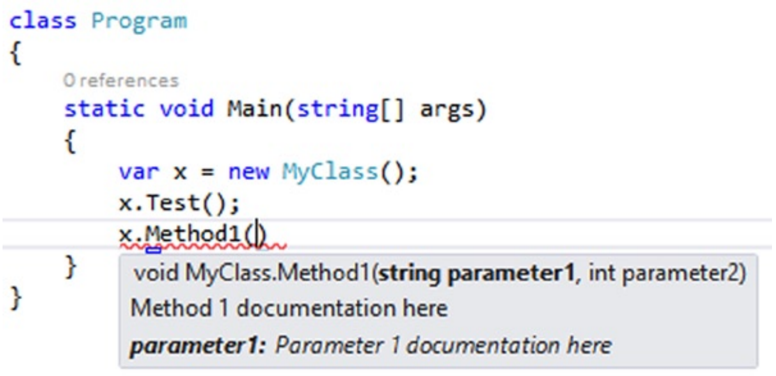
```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        var x = new MyClass();
        x.Test();
        x.
    }
}
```

```
void MyClass.Test()
Documentation of Test method goes here
```

Figure 1-16. *The IntelliSense List Member display*

At label (1) in Figure 1-16 is the dot, the trigger that tells IntelliSense to use the List Members feature. It opens the list of possible options shown at (2). Furthermore, if a comment has been written, it is displayed, as shown in the **Quick Info** popup at (3), which we will discuss shortly.

Another feature that kicks in once you choose the method of an object is **Parameter Info** (Figure 1-17). It gives information about every type of parameter and displays the comment written for the parameter.



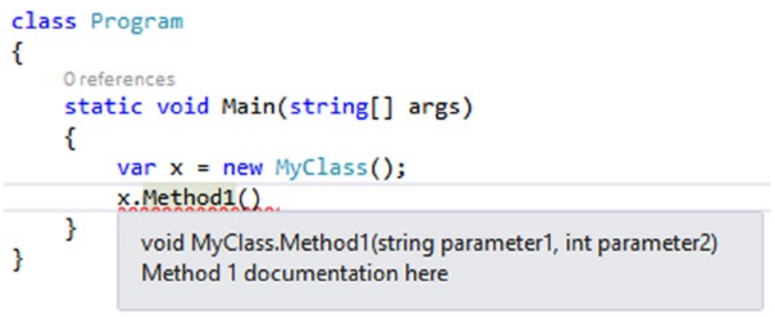
```
class Program
{
    0 references
    static void Main(string[] args)
    {
        var x = new MyClass();
        x.Test();
        x.Method1()
    }
}
```

```
void MyClass.Method1(string parameter1, int parameter2)
Method 1 documentation here
parameter1: Parameter 1 documentation here
```

Figure 1-17. *The IntelliSense Parameter Info display*

It is possible to trigger this feature by using the shortcut Ctrl+Shift+Spacebar or by using the Parameter Info button on the editor toolbar. If a method is overloaded, you can use the up and down arrow keys to navigate through the whole list of overloaded methods.

The **Quick Info** displays documentation about different methods (Figure 1-18). You can trigger it by using the menu or by using the shortcut Ctrl+I. A popup is shown with the return type of the method, parameters, and the comment written for the method. You can see Quick Info when you have the List Member open and also when you hover over an existing method.



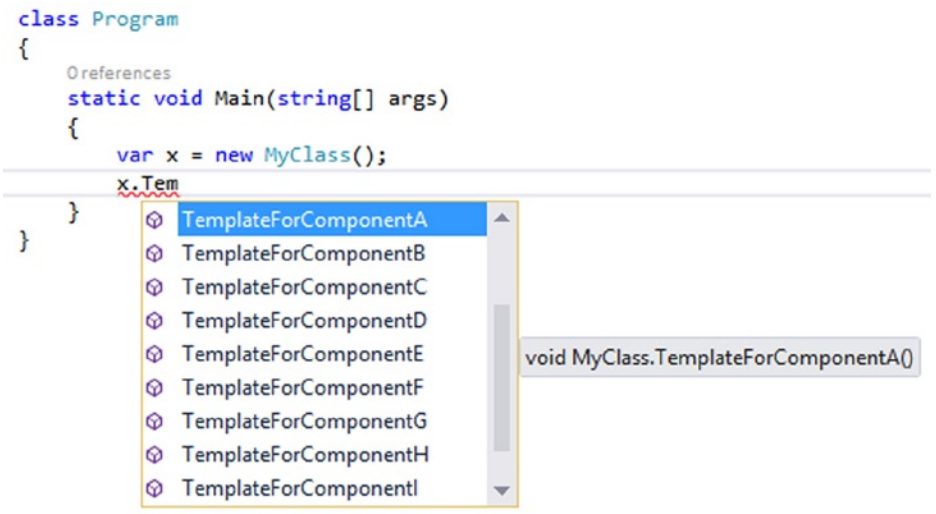
```
class Program
{
    0 references
    static void Main(string[] args)
    {
        var x = new MyClass();
        x.Method1()
    }
}
```

```
void MyClass.Method1(string parameter1, int parameter2)
Method 1 documentation here
```

Figure 1-18. *The IntelliSense Quick Info display*

Another addition to IntelliSense is the **Complete Word** feature, which completes the word you are typing if Visual Studio can figure out without a doubt which variable or method has been written. The shortcut is Ctrl+Spacebar.

In Visual Studio 2013, IntelliSense has been improved by allowing Pascal Case to filter a long **Method List.** It allows you to search by entering only the first letter of each word in a method name. For example, Figure 1-19 shows a list of several methods that start with the same name.



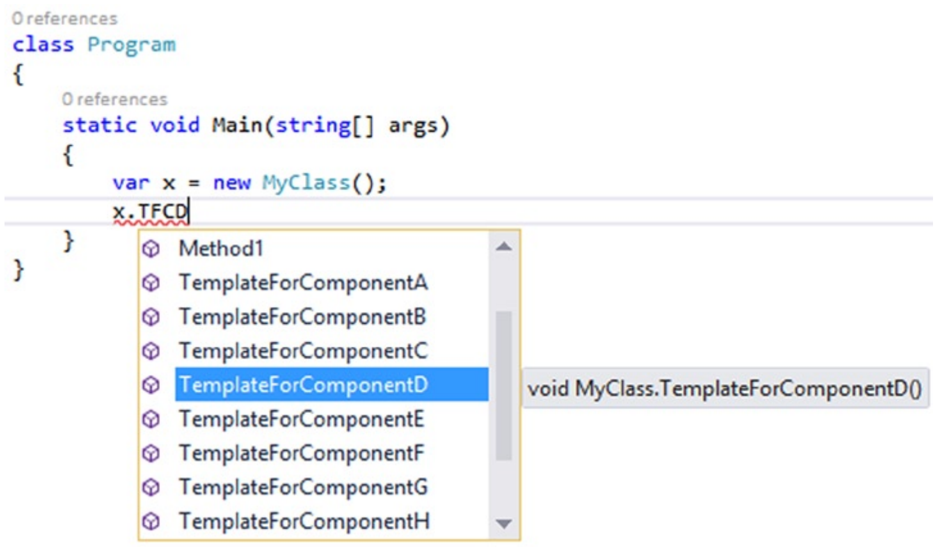
```
class Program
{
    0 references
    static void Main(string[] args)
    {
        var x = new MyClass();
        x.Tem
    }
}
```

```
TemplateForComponentA
TemplateForComponentB
TemplateForComponentC
TemplateForComponentD
TemplateForComponentE
TemplateForComponentF
TemplateForComponentG
TemplateForComponentH
TemplateForComponentI
```

```
void MyClass.TemplateForComponentA()
```

Figure 1-19. *The IntelliSense Long Member List display*

Instead of searching through the list, it is now possible to type **`TFCD`**, for `TemplateForComponentD` (Figure 1-20).



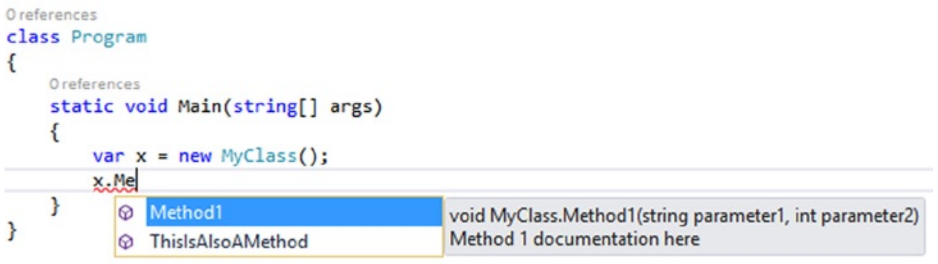
```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        var x = new MyClass();
        x.TFCD
    }
}
```

Method1
TemplateForComponentA
TemplateForComponentB
TemplateForComponentC
TemplateForComponentD
TemplateForComponentE
TemplateForComponentF
TemplateForComponentG
TemplateForComponentH

void MyClass.TemplateForComponentD()

Figure 1-20. *The IntelliSense Pascal Case feature*

Visual Studio 2013 IntelliSense can also now search for not only the first word written but also inside the method name (Figure 1-21). That is, you can search for method names that contain a string as well as those that start with it. For example, if you type **Me** and your class has a method named `Method1` and another named `ThisIsAlsoAMethod`, both will appear, which was not the case in earlier editions.



```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        var x = new MyClass();
        x.Me
    }
}
```

Method1
ThisIsAlsoAMethod

void MyClass.Method1(string parameter1, int parameter2)
Method 1 documentation here

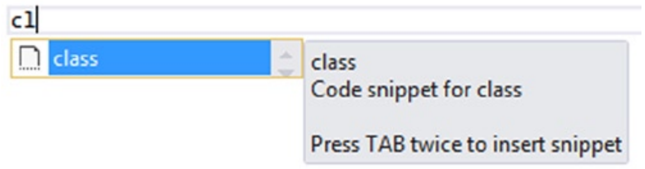
Figure 1-21. *IntelliSense with a Pascal Case filter now allows searches anywhere within a method name*

Code Snippet

Code Snippet allows you to have code written for you quickly after typing a few keywords. Code Snippet also contains templates with placeholders that are required to be filled.

■ **Note** You can find all preinstalled snippets in your Visual Studio folder at `C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC#\Snippets\1033\`. This folder contains subfolders for different types of projects.

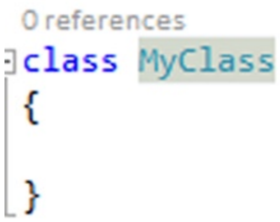
For example, by default, Visual Studio has a Code Snippet named `class`. If you type **cl**, IntelliSense shows the **class** snippet. If you press the Tab key twice, a new class is generated. The first time you press Tab, the letters “cl” expand to the word “class” (Figure 1-22).


```
cl
class
```
class
Code snippet for class

Press TAB twice to insert snippet

Figure 1-22. *Code Snippet with IntelliSense*

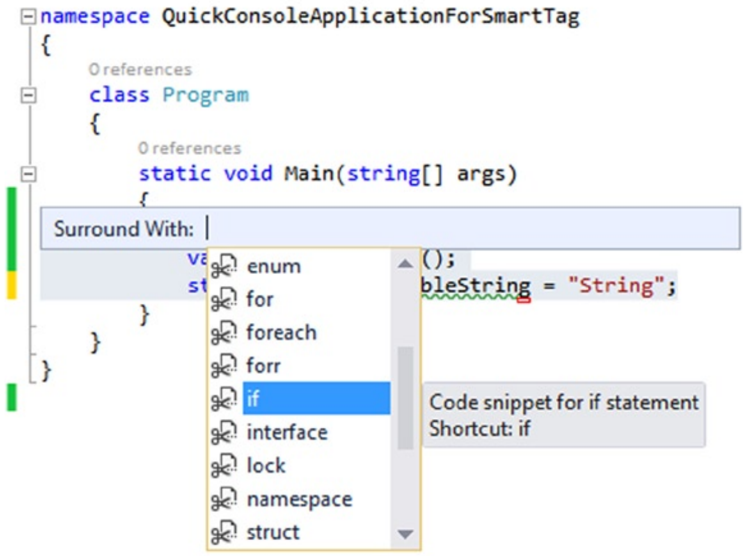
The second time you press Tab, the Code Snippet is launched and a new class is generated with a default name, which is highlighted to allow you to customize the name (Figure 1-23).


```
0 references
class MyClass
{

}
```

Figure 1-23. *Code snippet expanded with default class name*

It is also possible to trigger a Code Snippet of the type Surround by selecting some code and typing Ctrl+K + Ctrl+S. For example, if you select a few lines of code and use the shortcut for Code Snippet surround, you can select the IF template to have the code surrounded by braces (Figure 1-24).



```
namespace QuickConsoleApplicationForSmartTag
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
Surround With: |
            va          ();
            st          bleString = "String";
        }
    }
}
```

Figure 1-24. *Surround code snippet*

You can define your own snippet by adding your snippet to the Visual Studio Snippets folder, and you can edit an existing snippet with Notepad. However, a better way is to use the code editor (**Tools ➤ Code Snippets Editor**) to add your custom Code Snippet folder to Visual Studio. This way, you can copy and paste your folder anywhere you work or share it with your team. The creation of a code snippet is covered later in this book.

Find and Replace

The **Quick Find** feature allows you to search a string by using the shortcut Ctrl+F (Figure 1-25). You can also access it by choosing **Edit ➤ Find and Replace ➤ Quick Find**. It opens a small search window at the top right of the editor. By default, it searches only in the current document but this can be changed to all open documents, the current project, or for the entire solution. In all cases, the keyword searched is highlighted and you can navigate through all that has been previously searched by pressing F3.