# What Is Visual Studio?

Having a good background in Visual Studio and knowing its history and features can help you understand all the chapters in this book better. Of course, such an understanding is a prerequisite for reading this book, but it is likely that readers have acquired their knowledge from different resources and from their daily experience with this IDE, and therefore may be unfamiliar with some aspects of it. This chapter describes a few things that may be helpful as you read the book.

First, I'll provide a short history of Visual Studio and the main features and enhancements in each of its versions. After that I'll cover Visual Studio 2008 (code-named Orcas) and its new features and requirements, and the installation process. Then, after an overview of the Visual Studio IDE, you will learn about extensibility in Visual Studio and the Visual Studio SDK.

# Introduction to Visual Studio

Visual Studio is one of the most famous Microsoft products and is specifically designed for development goals. It's the most popular tool for all developers who use Microsoft development technologies and is widely used by developers around the world. It is built on an Integrated Development Environment (IDE) and enables users to build different types of applications using Microsoft programming languages.

Visual Studio, which is abbreviated as VS by community members (and is used frequently throughout this book), supports several platforms and programming languages "out of the box," that is, without requiring additional installations, plug-ins, expansion packs, or products.

Because a good background in the history of VS and how its features have improved in each version can help you get a good picture of what Visual Studio is and how it has reached this point in its development, this chapter provides a short history of Visual Studio and its features over the past 11 years and offers a brief introduction to Visual Studio 2008 and its installation process. Chapter 2 discusses the .NET Framework and its correlation with Visual Studio and different pieces of this development framework. Other chapters discuss all aspects of Visual Studio extensibility, with a primary focus on the latest version, Visual Studio 2008.

# History

Visual Studio is an old and famous Microsoft product that was created for development purposes. In 1997 and later, Microsoft built some visual development languages for its Windows operating system. It also built an IDE to support these languages in a visual manner and named it Visual Studio. Visual Studio has changed a lot during the past 11 years. Not only has it become richer, it has enhanced the way it integrates with Microsoft programming languages and technologies. Over the years, Microsoft also added several new features to support database technologies, markup languages, web development tools, unit-testing features, and team work management.

In the following sections, we'll take a look at the history of six versions of Visual Studio released during these years, including all their major improvements, changes, and new features.

### **Visual Studio 97**

1997 was the first year in which Microsoft tried to build a single environment for developing in multiple languages. In that year, it introduced Visual Studio as an IDE for development, and it included the following elements and languages:

- □ Visual Basic 5.0
- $\Box$  Visual C++ 5.0
- □ Visual FoxPro 5.0 (for xBase programming)
- □ Visual J++ 1.1 (for Java programming for Windows)
- □ Visual InterDev (for creating dynamic web pages with ASP)
- □ MSDN (Microsoft Developer Network)

Visual C++, Visual J++, Visual InterDev, and MSDN used the same environment, named Developer Studio, while Visual Basic and Visual FoxPro had their own environments.

Visual Studio 97 was a first attempt and a first version, but even then it was a great development environment. Back then, every developer could enjoy working in this IDE. Current versions of Visual Studio are very different, in terms of both the user interface and the features that are included.

The internal version of Visual Studio 97 was 5.0.

### Visual Studio 6.0 (98)

In 1998, Microsoft tried to integrate its development tools more consistently than in the past and took the first steps toward what we now have as the .NET Framework.

In that year, Microsoft released Visual Studio 6.0 with the same components as Visual Studio 97, but updated their version numbers to 6.0. Even Visual J++ and Visual InterDev version numbers were updated to 6.0 from 1.1 and 1.0, respectively.

Visual Studio 6.0 included the last version of Visual Basic as we knew it, because current Visual Basic (known as Visual Basic .NET) is different from that earlier Visual Basic in some fundamental ways. Visual Studio 6.0 was also the last version of Microsoft development tools for the Windows 9x platform.

At this point, Visual J++ died, and Microsoft no longer supported any programming language for the Java Virtual Machine (JVM).

Other changes were reflected in the number of environments. Visual Studio 98 had one more environment than Visual Studio 97. Visual J++ and Visual InterDev left the environment they had and got their own environments.

Visual Studio 6.0 (98) was the last link of a chain that ended at this point. After Visual Studio 6.0, Microsoft focused mainly on building a single compiler and environment by introducing the .NET Framework.

## Visual Studio .NET 2002

2002 saw a revolution in Microsoft development tools and programming languages. After some years of hard work and planning, Microsoft released the .NET Framework, Visual Studio .NET, and the .NET programming languages. This marked the first point of complete integration and consistency between Microsoft development products. The beta version of Visual Studio 2002 was available to MSDN subscribers in 2001, but the final version appeared on the market in 2002.

The .NET Framework 1.0 was the most important release that year. It introduced managed code to Microsoft developers. In the .NET Framework, programs compile to *Microsoft Intermediate Language* (*MSIL*) or *Common Intermediate Language* (*MIL*), rather than compile to machine language. You'll read more about the .NET Framework in Chapter 2, "The .NET Framework."

Visual Studio .NET 2002 (with 7.0 as the internal version) was the first version of Visual Studio that required NT-based operating systems for installation. It could support four languages out of the box:

- □ Visual Basic .NET: This was the first member of the Basic family languages that could support object-oriented programming 100 percent. It differed in some fundamental ways with Visual Basic 6.0, which had been one of most popular programming languages for years. Visual Basic .NET was (and still is) the most popular programming language around the world.
- □ **C#:** This was a completely new language in the .NET Framework 1.0. Later, it would catch the interest of many developers. It still has a growing number of fans, and it's likely to be the most popular Microsoft programming language for the near future. C# uses C syntax but is simpler than other Visual languages with C-family syntax, such as Visual C++.
- □ **Visual J#:** Like C#, J# was a new language introduced in the .NET Framework. It can be considered a replacement for Visual J++, which died after Visual Studio 6.0. It has a Java syntax, but unlike Visual J++ it can build applications only for the .NET Framework, not the Java Virtual Machine (JVM).
- □ **Visual C++:** Like its preceding versions, Visual C++ is a programming language with a C-family syntax. It is still available as part of the .NET Framework with enhancements.

Using these programming languages and a single compiler, the .NET Framework provides a rich set of tools to build various kinds of desktop, web, and embedded-device applications. It includes

several technologies, such as ASP.NET, ADO.NET, and web services. The .NET Framework and its related technologies are covered in Chapter 2, "The NET Framework," in more detail.

## Visual Studio .NET 2003

A year after Visual Studio .NET 2002 was released, Microsoft shipped Visual Studio .NET 2003 (with 7.1 as the internal version), which could support the .NET Framework 1.1, the new version of the .NET Framework.

The .NET Framework 1.1 had built-in support for building mobile applications and included some other new features, as well as bug fixes. It was the first stable version of the .NET Framework and it has been widely used. Visual Studio .NET 2003 was even more stable and popular than the 2002 release.

There were four editions for this 2003 version:

- Professional
- Enterprise Developer
- Enterprise Architect
- □ Academic

The Enterprise Architect version offered built-in support for Microsoft Visio, which enables *Unified Modeling Language (UML)* for application architecture.

Microsoft released Service Pack 1 for Visual Studio .NET 2003 in September 2006.

### Visual Studio 2005

In October 2005, Microsoft released the new version of the .NET Framework, the .NET Framework 2.0, and a new version of Visual Studio that was code-named Whidbey and later used 2005 as its version number. The internal version of Visual Studio 2005 is 8.0. At this point, Microsoft removed the ".NET" suffix from the name of its products such as Visual Studio and Visual Basic, bit it still supports the .NET Framework.

Visual Studio 2005 was enhanced to support the .NET Framework 2.0. It had the following features and changes:

- □ ASP.NET 2.0
- ADO.NET 2.0
- Generics in .NET 2.0
- Enhanced IntelliSense
- Addition of new project types
- A local web server to test ASP.NET applications without Internet Information Services (IIS)
- □ Support for 64-bit applications

The other major change in VS 8.0, in comparison to its predecessors, was that its editions were introduced in a different way. It had the following editions:

- □ Express
- □ Standard
- Professional
- □ Tools for Office
- Team System

There were two new editions for VS 2005: Express and Team System. Express editions are free and suitable for small business and individuals who just want to build small applications. Four express editions are available for all available languages in the .NET Framework, as well as a Visual Web Developer to create ASP.NET applications. Express editions don't offer all the professional features of Visual Studio, however.

There are five Team System editions. Four of them have a major role in building software: Developers, Architects, Testers, and Database Designers. The other one, Visual Studio Team Suite edition, has all these four capabilities in a single edition.

After the initial release of Visual Studio, some other products were shipped along with it for special purposes:

- □ **Visual Studio Team System (VSTS):** This is suitable for team work. It uses the Visual Studio 2005 IDE, but it adds some applications such as Visual Studio Team Foundation Server, a powerful source control, which includes a free client as an add-in for Visual Studio and great integration with VS.
- □ **Visual Studio Tools for Office (VSTO):** This enables the use of Microsoft Office document data in other applications and Microsoft Office APIs. It also enables developers to use Microsoft Office controls in other applications.
- □ **Visual Studio Tools for Applications (VSTA):** This is the enhanced version of Visual Basic for Applications (VBA). VSTA has a customized version of the VS 2005 IDE that includes a runtime that can be used in other applications via the .NET object model.

Service Pack 1 for Visual Studio 2005 was released in December 2006. This service pack replaced the ASP.NET websites with ASP.NET web applications as the default project type for ASP.NET applications. This was the default option in Visual Studio 2002 and 2003 as well, and was a change requested by many community members.

Microsoft launched the .NET Framework 3.0 in 2006, when Visual Studio 2005 was the latest version of Visual Studio. Thus, there are add-ins for Visual Studio 2005 that enable development for *Windows Presentation Foundation, Windows Communication Foundation,* and *Windows Workflow Foundation* as main new technologies in the .NET Framework 3.0. You'll read more about these technologies in Chapter 2.

You can find more information about Visual Studio 2005 from Wrox's *Professional Visual Studio* 2005, by Andrew Parsons and Nick Randolph (ISBN: 9780764598463).

## Visual Studio 2008

The latest available version of Visual Studio (sixth version) is code-named Orcas and should be launched by Microsoft by the time you read this book, with 9.0 as the internal version. It was released earlier (at the end of 2007) to MSDN subscribers. The next version of Visual Studio (code-named Hawaii) is under development.

Visual Studio 2008 is the first version that supports three different versions of the .NET Framework out of the box: 2.0, 3.0, and 3.5. The .NET Framework versions 2.0 and 3.0 were available before Visual Studio 2008, but the .NET Framework 3.5 shipped with Visual Studio Orcas. The default framework for VS 2008 assemblies is 3.5, but you can choose which version of the .NET Framework you prefer to compile your assemblies with.

Orcas focuses primarily on Windows Vista and Office 2007, as well as new features in the .NET Framework 3.0 and 3.5, while keeping all the goodies from Visual Studio 2005.

Here's a list of new features and changes in Visual Studio 2008:

- Built-in support for Windows Presentation Foundation and an excellent designer, originally called Cider, for XAML layouts
- Built-in support for Windows Communication Foundation
- Built-in support for Windows Workflow Foundation, and a powerful workflow designer
- □ A new language feature in .NET 3.5, LINQ, which is available in new versions of Visual Basic and C# and a LINQ to SQL designer (to define type mappings for SQL data)
- □ Addition of Microsoft Silverlight projects, a new type of web project that brings the great features of Windows Presentation Foundation and XAML to web applications
- □ JavaScript IntelliSense and debugger
- □ A powerful XHTML/CSS editor
- □ J# is no longer included

Visual Studio 2008 ships with the same editions as Visual Studio 2005. Visual Studio Team System 2008 was originally named Rosario and was released with Visual Studio 2008.

You can read more about Visual Studio 2008 in Wrox's *Professional Visual Studio 2008*, by Nick Randolph and David Gardner (ISBN: 9780470229880).

# Extensibility

Visual Studio is a tool for development and for extending other things, but it's extensible itself. Obviously, the built-in features of Visual Studio won't be enough for some developers, who need something more or something special. Many current VS features are community requests from developers, and some of them have been implemented by third-party components and open-source projects before they were incorporated into Visual Studio. I won't bother reviewing the necessity of extensibility features in a tool like Visual Studio, because you already know many of the reasons from your own experience. Instead, this section provides a brief introduction to the extensibility features in this tool.

Fortunately, Microsoft has put some extensibility features into Visual Studio to help developers extend VS easily. Add-ins, macros, and packages are three common ways of extending Visual Studio and they have been a part of VS extensibility features for a long time.

Add-ins enable users to gain access to VS's underlying APIs for IDE, in order to automate tasks such as coding and deployment. Macros are a way to automate frequently repeated tasks in Visual Studio. Developers can create macros by recording them or writing them programmatically. Packages, created using the Visual Studio SDK, enable a deeper integration between the IDE and programming languages.

In addition, compared to the 2005 version, a few new extensibility options have been added to Visual Studio, such as visualizers and Visual Studio Shell. This book will cover all these extensibility options in enough detail to make readers familiar with extensibility in Visual Studio. Initially, documentation and community activities related to Visual Studio extensibility were weak before the release of Visual Studio 2008. However, after a Microsoft TechEd 2007 conference (when Visual Studio 2008 was in beta 1 stage), the Visual Studio Extensibility team at Microsoft tried to improve this situation. One such improvement was new documentation about VS extensibility. This book enhances that documentation and should be a good resource for the Visual Studio community, whose members are asking for more information.

Extensibility features for Visual Studio aren't available in Express free editions. You must have a commercial edition (Professional or above) to get all the benefits of extensibility features. You'll read more about extensibility later in this book, but I wanted to introduce the various options here.

# **Visual Studio SDK**

The Visual Studio SDK (Software Development Kit) is a free downloadable package for Visual Studio that contains a set of documentation, code samples, and other materials to help you develop applications that integrate with Visual Studio.

The Visual Studio SDK contains all three extensibility options mentioned in the previous section (addins, macros, and packages) as well as other options that are new or not as well known.

This Visual Studio 2008 SDK offers a new extensibility option: *Visual Studio Shell*. This new tool enables you to build your applications based on a core foundation of VS and with a similar look to its IDE. This is helpful, as developing such applications gets easier, and the user interface more familiar, for many users.

Visual Studio Shell works in two modes: integrated mode and isolated mode. Integrated mode is good for programming languages, whereas isolated mode is helpful for special tools. You'll read more about the Visual Studio SDK later in the book.

# Installation

Although all of the code samples and discussions in this book are about Visual Studio 2008, the latest available version, most of the material is applicable to previous versions with few or no changes.

Here, I'm referring to installation of the Visual Studio 2008 Team System edition. I chose this edition because it's becoming more widely used than other editions. In addition, all its features and capabilities are also available in other commercial editions. You can download a free 90-day trial of this edition from the Microsoft downloads site in order to work through this book and test its sample code. For the purposes of this book, there is no difference between the Professional or later editions of Visual Studio, and you can even use the Standard edition for many of the chapters.

In the following sections, I talk about system requirements and the installation process of Visual Studio 2008 Team Suite edition, but other editions have very similar requirements and the same installation process.

### System Requirements

According to Microsoft's documentation, the supported operating systems for Visual Studio 2008 are various editions of Windows Vista, Windows 2003, and Windows XP.

Minimum installation requirements for Visual Studio 2008 Professional edition are as follows:

- □ Processor: 1.6 + GHz
- □ RAM: 384 + MB of available physical RAM
- □ Display: 1024 × 768

Recommended settings, however, are as follows:

- □ Processor: 2.2 + GHz
- □ RAM: 1024 + MB of available physical RAM
- □ Display: 1280 × 1024

For Windows Vista, you need a 2.4 GHz CPU and 768 MB of RAM.

For this book, I used Visual Studio 2008 Team System edition on Microsoft Windows Vista Ultimate x86, but nothing is different from other operating systems.

#### **Installation Process**

After running the Visual Studio setup file, you will see a window like the one shown in Figure 1-1.



Figure 1-1: Visual Studio 2008 Setup Window

By clicking the first option, Install Visual Studio 2008, you can begin the installation process. The second option installs the MSDN library, and the third option installs service releases, which keep your Visual Studio installation up to date.

The installation process for Visual Studio 2008 is similar to that for Visual Studio 2005. After opening the installer, you need to accept the terms and license agreement and choose the appropriate type of installation (Default, Full, or Custom) to install Visual Studio at the specified path.

After finishing the installation, you can launch your Visual Studio IDE. On its first run, Visual Studio asks you to choose your preferred environment settings based on the programming language and the type of development that you frequently use (see Figure 1-2).

🐼 Choose Default Environment Settings				
Visual Studio Team System 2008				
Before you begin using Visual Studio for the first time, you need to specify the type of development activity you engage in the most, such as Visual Basic or Visual C#. Visual Studio uses this information to apply a predefined collection of settings to the development environment that is designed for your development activity.				
You can choose to use a different collection of settings at any time. From the Tools menu, choose Import and Export Settings and then choose Reset all settings.				
🕼 Allow Visual Studio to download and display online RSS content				
Choose your default environment settings:				
General Development Settings	Description:			
Visual Basic Development Settings	customizes the environment to maximize code editor screen space and improve the visibility			
Visual C# Development Settings	of commands specific to C#. Increases			
Visual C++ Development Settings Web Development Settings	productivity with keyboard shortcuts that are			
	designed to be easy to learn and use.			
	Start Visual Studio Exit Visual Studio			

Figure 1-2: Choosing your preferred settings at first startup

# Installing the SDK

You can download the Visual Studio 2008 SDK 1.0 from http://tinyurl.com/3brqyy. This installer contains everything you need to set up the Visual Studio SDK. This is the only available version at the time of writing, but Microsoft releases new versions of the Visual Studio SDK on a regular basis, so you may be able to get a newer version by the time you read this.

# An Overview of the IDE

This section provides you with an overview of the IDE. I know that most readers are already familiar with these principles, but it is important that every reader has seen them before stepping into the content. After a short introduction to the structure of the IDE, I'll introduce the main elements of the Visual Studio environment.

### **General Structure of the IDE**

The first time you load Visual Studio, you will see an environment like the one shown in Figure 1-3.



Figure 1-3: Visual Studio 2008 IDE

The VS IDE has a tabular structure, which means that it opens the main content in several tabs in the center of the IDE. It also contains several windows that can be docked to one of corners, as well as a collection of different toolbars for development purposes. The Visual Studio IDE is completely customizable — you can change the layout and look of most of its elements. Using the Solution Explorer, you navigate through your solutions and projects, choose your code files, and open them in new tabs. Other windows enable you to easily accomplish necessary tasks.

Windows and toolbars play the main role in using Visual Studio for development, and you'll use them to achieve the benefits of different features of Visual Studio.

# Main Elements of the IDE

This book describes how to extend Visual Studio. Any extension of Visual Studio has its own goals, but regardless of the goals of an extension, it extends one of the main parts of this environment. Because most of the time you are dealing with some of the main elements of the VS IDE for extensibility, it's worthwhile to review all these elements in one place and learn what they do. Therefore, in the next few sections, you'll learn about all these windows and how they help you do your work.

#### Solution Explorer

The Solution Explorer, shown in Figure 1-4, is an important window in Visual Studio. It enables you to navigate between your solutions, projects, and project files in an hierarchical manner, using a tree control.



Figure 1-4: Solution Explorer

By clicking on each item, a new tab opens in the middle of the IDE and you can see the content of that item. You can also gain access to the properties of each item by right-clicking on it. Some common tasks are available for each item in the right-click menu.

#### **Properties Window**

The Properties window, shown in Figure 1-5, is another common window in Visual Studio.

Properties 👻 👎		
WebApplication1 Solution Properties -		
(Name)	WebApplication1	
Active config	Debug Any CPU	
Description		
Path	C:\Users\Keyvan Nayyeri\I	
Startup project	WebApplication1	
(Name)		
The name of the solution file.		

Figure 1-5: Properties window

This window shows all available properties of an item and can sort them alphabetically or by group. Using this window, you can set the properties of a solution, project, file, control, or any other item easily.

#### Toolbox

The Toolbox, shown in Figure 1-6, is a window that contains a list of controls that you can drag and drop into your Windows Forms, web pages, XAML windows, or workflow designer.

In the toolbox, controls are grouped and are displayed based on the type of application that you're developing. You can also add new groups and custom controls manually.

Toolbox 👻 🛨 🗙
Standard
🗆 Data
Pointer
🕮 GridView
DataList
🗐 DetailsView
- FormView
E ListView
Repeater
₩≥» DataPager
SqlDataSource
AccessDataSource
LinqDataSource
BobjectDataSource
, XmlDataSource
SiteMapDataSource
Validation
Navigation
🗄 Login
WebParts
AJAX Extensions
Reporting
HTML
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

Figure 1-6: The Visual Studio Toolbox

#### **Server Explorer**

The Server Explorer window, shown in Figure 1-7, is another window that enables you to navigate through local and remote servers in order to gain access to databases as well as resources and services of servers.



Figure 1-7: Visual Studio's Server Explorer window

#### **Class View**

This window, shown in Figure 1-8, displays all the available classes in a solution (in addition to built-in classes in the .NET Framework) and groups them according to their namespaces.

Class View	<b>→</b> ₽ ×
≝   ← ⇒   ≝ •   &	
<search></search>	• 🕨 🛒
WebApplication1   Project References  VebApplication1  VebApplication1  VebApplication1  Base Types  VebApplication2  Project References  Project References Project References Project References Project References Project References Project References Project References Project References Project References Project References Project Reference	
Page_Load(object, System.Eve	ntArgs)
🖏 Solution Explorer 🕁 Class View	

Figure 1-8: The Class View window

#### **Error List**

All errors in your code, either at the time of writing the code or upon compilation, are shown in the Error List window, shown in Figure 1-9.

Error List           O Errors         ① 0 Warnings           ① 0 Errors         ① 0 Warnings	es			<b>-</b> ₽×
Description	File	Line	Column	Project

Figure 1-9: Visual Studio's Error List window

This window lists all warnings and errors. By clicking on an item in the list, you can access the original code that caused the warning or error.

#### Output

The last important Visual Studio window is the Output window, shown in Figure 1-10. This window provides a list of messages about all the processes that are occurring in Visual Studio.

Output	· · · · · · · · · · · · · · · · · · ·	1 >	<
Show output from: Build	•   🗟   🖨 🕒   🛒   🖬		
Rebuild All started: Pr C:\Windows\Microsoft.NET\Frame Compile complete 0 errors, WebApplication1 -> C:\Users\Ke ========= Rebuild All: 1 succ	oject: WebApplication1, Configuration: Debug Any CPU work\v3.5\Csc.exe /noconfig /nowarn:1701,1702 /errorreport:prompt / 0 warnings eyvan Nayyeri\Documents\Visual Studio 2008\Projects\WebApplication1\ eeeded, 0 failed, 0 skipped ===================================	wa We	•
			÷
<		Þ.	
Error List Output			

Figure 1-10: The Output window

Most of the actions and tasks in Visual Studio have a corresponding command, and these commands generate output. The output of actions is helpful for debugging and finding the status of a task. You can also write your own texts into the Output window for debugging purposes.

# Summary

In this first chapter of the book, you took a brief walk through the history of Visual Studio over the past 11 years, including the features and changes of the six versions of this product that have been released during this period. After that, you learned about the extensibility options in Visual Studio and were introduced to the Visual Studio SDK. In addition, you learned the installation requirements and procedures for Visual Studio. Finally, you took a quick tour of the Visual Studio IDE and its main elements.