Introduction

Welcome to the latest Microsoft product, Visual Studio .NET, which unleashes the next generation of application development. Visual Studio .NET provides a complete development environment in which you can develop a variety of applications ranging from Windows applications to ASP.NET Web applications and Web services. Visual Studio .NET provides you a number of programming languages to choose from for developing applications, enabling you to leverage your current development skills instead of needing to refrain.

In *Visual Studio .NET All-in-One Desk Reference For Dummies*, we take a straightforward approach to telling you about Visual Studio .NET and the different programming languages that it offers. Besides giving you lots of concepts in plain and simple English, we've included lots of code samples that help you put the concepts to work. The goal of this book is to take the anxiety and stress out of mastering the assortment of technologies available with Visual Studio .NET. We hope that you find the book a clear and straightforward resource for exploring Visual Studio .NET.

About This Book

This book is your friendly and approachable guide that will help you master Visual Studio .NET. The way this book is organized makes you grasp the technologies easier and faster. This book, however, isn't meant to be read from front to back. The book is divided into seven mini books — each mini book focusing on a specific technology. Each chapter in each mini book is divided into sections, each of which is self-contained. Here are some of the topics that we cover in this book:

- Key components of the .NET Framework
- An overview of Visual Studio .NET along with its installation
- The Visual Studio .NET IDE
- Features of Visual Basic .NET, Visual C++ .NET, and Visual C#
- Application development using Visual Basic .NET, Visual C++ .NET, and Visual C#
- ASP.NET Web application development
- ASP.NET Web services development
- Application deployment

Conventions Used in This Book

Keeping things consistent makes them easier to understand. In this book, those consistent elements are *conventions.* Notice how the word *convention* is in italics? In this book, we put new terms in italics, and then we define them so that you know what they mean.

This book contains lots of code. All code in this book appears in monofont type, as shown here:

```
MsgBox("Hello World")
```

When URLs (Web addresses) appear within a paragraph, they appear in monofont and look like this: `www.microsoft.com`.

Any programming keywords or method names used within a paragraph also appear in monofont type, like this: `Display_Details`.

What You Don't Have to Read

When you read this book, remember that it is not meant to be read from front to back. However, when you read a specific chapter, it is advisable to read the entire content of a section. But, if you read a chapter just for reference, you can very well skip sidebars without missing crucial information.

Foolish Assumptions

Making foolish assumptions can make a fool out of the person who makes them. Throwing caution to the wind, we make a few assumptions about you, the readers of this book:

- You're already familiar with programming languages in the previous version of Visual Studio, Visual Studio 6.0.
- You have a basic knowledge of HTML and XML.
- You know the difference between Web browsers and Web servers.
- You have a basic understanding of Internet Information Server (IIS).

In addition to these assumptions, we also assume that you've installed Visual Studio .NET software on your computer. It'll be great if you have a working Internet connection because you can then directly download the applications provided in this book from `www.dummies.com/extras/VS.NETAllinOne`.

How This Book Is Organized

We've divided *Visual Studio .NET All-in-One Desk Reference For Dummies* into seven mini books. Each mini book covers a different technology separately. Organizing the book as an assortment of several books facilitates your search for the topics that you want to explore. Any time that you require help or information on a specific topic, use the table of contents to locate your topic. This section provides a breakdown of the mini books and what you'll find in each of them.

Book I: Visual Studio .NET Overview

Book 1 provides you an overview of Visual Studio .NET, where you'll find answers to some of your most fundamental questions:

- What is the .NET initiative?
- What are the components of Visual Studio .NET?
- How do you install Visual Studio .NET?

Book II: Using the Visual Studio .NET IDE

Book 2 is a good resource to get you started with Visual Studio .NET. In this book, you'll get a closer look of the common Integrated Development Environment (IDE) provided with Visual Studio .NET. This book takes you through the complete Visual Studio .NET interface and tells you how to use the various IDE tools.

Book III: Visual Basic .NET

Book 3 introduces you to the new features of Visual Basic .NET. Additionally, this book delves into the programming intricacies with Visual Basic .NET. Some of the areas that we explore in this book are

- Windows Forms
- Variables
- Program flow control
- Procedures
- Classes
- Error handling
- Database applications with Visual Basic .NET

Book IV: Visual C++ .NET

Book 4 is a perfect resource for you to explore programming with Visual C++ .NET. Some of the topics that we cover in this book are

- MFC applications
- Database applications with Visual C++ .NET
- ATL server projects
- Managed extensions in C++

Additionally, this book teaches you how to debug and handle exceptions in Visual C++ .NET applications. You can also read about upgrading the existing application to Visual C++ .NET applications.

Book V: Visual C# .NET

This book gets you started with Visual C#. In addition to providing a basic language feature, this book tells you how to create Windows applications and Windows services by using Visual C#.

Book VI: Associated Technologies and Enhancements

This book is a good guide for you to explore the field of programming for the Web with ASP.NET. The book builds a solid ground for ASP.NET and gets you started with it. We cover the following major topics in this book:

- Web Forms and the Web Forms Server controls
- Data binding with server controls
- Mobile Web applications
- Using ADO .NET
- HTTP handlers
- Caching
- Application security

Book VII: Creating and Deploying Web Services and Other Visual Studio .NET Solutions

Book 7 presents advanced features of Visual Studio .NET in such a simple manner that you'll easily master these advanced features while being tempted to discover them. This book teaches you how to create Web services and Web service clients. Additionally, you'll find out how to deploy different types of applications created in Visual Studio .NET.

Icons Used in This Book

To make your experience with the book easier, we use various icons in the margins of the book to indicate particular points of interest.

This icon flags technical details that are informative and interesting but not critical to write code in any of the .NET languages. You can skip these if you want.

This icon marks important directions to keep you out of trouble. These paragraphs contain facts that can keep you from having nightmares.

This icon flags useful information that acts as a hint or a tip for performing certain tasks in an easy and efficient manner.

This icon is a friendly reminder or a marker for something that you want to make sure that you cache in your memory for later use. Don't skip these gentle reminders.

Where to Go from Here

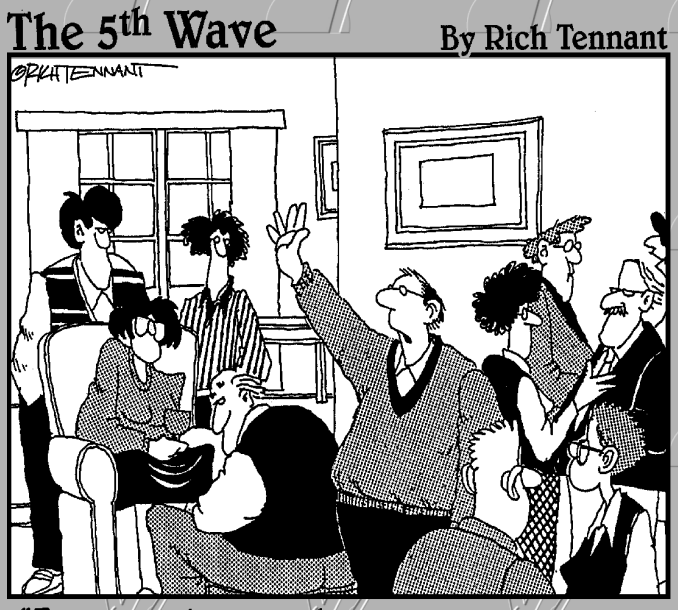
Just as Visual Studio .NET is a single product that hosts multiple technologies, *Visual Studio .NET All-in-One Desk Reference For Dummies* is certainly a great collection of multiple books that provides you with several of these technologies. Now, you're ready to use this book and discover different technologies that Visual Studio .NET hosts. No doubt, the table of contents stands as a ready reference to navigate you to your topic of interest.

We wish you good luck on your journey to explore the host of technologies with Visual Studio .NET.

Enjoy!

Book I

Visual Studio .NET Overview



"Excuse me – is anyone here NOT talking about the .NET initiative?"

Contents at a Glance

Chapter 1: Exploring the .NET Initiative

In This Chapter

- **Finding out about the .NET initiative**
- **Getting to know products in the .NET suite**

Visual Studio .NET is the result of the .NET initiative — an initiative that didn't originate on its own but was brought about by you and me, the customers. We brought about this initiative through increased collaboration, mostly through the Internet, and partly through other communication channels. Increased collaboration resulted in a gradual shift from desktop computing to distributed computing. Microsoft, as always, was quick to realize this shift and launched its .NET initiative to capitalize on it. The .NET initiative gained momentum in the second quarter of 2000 and brought about the development of new applications and services.

In this chapter, we explore the world of .NET, discussing the products and services associated with .NET and discovering how Visual Studio .NET takes the lead role in the .NET initiative.

Understanding the .NET Initiative

Today, users often encounter on a daily basis applications that are user-friendly but not platform-independent. For example, when you visit a Web site, you can register on the site and that registration information can be used by the Web site administrator to offer customized products. The same is pertinent for applications, such as Microsoft Word, installed on your computer in which you can store your preferences. Comparatively, if you have two applications running on different platforms (such as Linux and Windows together), they may not offer complete interoperability — meaning that you can't run Windows applications on a Linux platform.

You can therefore conclude that customizing an application for a user is easy, but you have very limited options for customizing an application for another. The .NET initiative aims to bridge this customizability gap between applications.

But what are the benefits of customizing one application for another? If you've not experimented with applications that communicate with each other, you may not know all their benefits. Application interoperability enables you to access existing data and functionality from different applications that may run on different platforms.

Take this idea a little further: If an application exposes its functionality over the Internet, your application can access the Internet application from anywhere. Thus, you've got two applications that talk over the Internet. This is exactly what .NET tries to achieve.

Check out how .NET enables applications to communicate with one another. As an example, take two applications that need to interoperate, looking closely at some of the inherent barriers and how they're overcome.

- **Geography:** Applications that need to interoperate can be located in different parts of the world.
- **Platform:** Applications may run on different platforms.
- **Language:** These applications may have been developed in different programming languages without the slightest degree of similarity.

Consider possible solutions to these obstacles. The simplest solution that we (and probably you, too) can think for the geographical requirement is to run the applications through the Internet. The Internet is the fastest and the easiest way to break geographical barriers. For the platform problem, you need to have a common language that can make the two applications talk with each other. XML (eXtensible Markup Language) has fast emerged as this common language and is also the industry standard for describing and transporting data. Therefore, XML is used to exchange data and information between .NET applications. XML also solves the language difference problem. Even if applications don't understand each other's language, they can communicate as long as they are based on XML.

Thus, you can create applications that don't require user interaction but still communicate with each other to serve a common purpose. We call these applications *services* to distinguish them from the applications that we normally use. See how smart you already are? You now know three important keywords to use together: XML, Web (or Internet), and services — XML Web services! XML Web services are the most significant outcome of the .NET initiative.

With XML Web services, you can provide software products as services. Users don't need to purchase software. Instead, they can subscribe to a service and use the service as long as they need it. Microsoft is coming up with its own set of Web services, known as My Services. These services are based on the Microsoft Passport authentication service, the same service that runs Hotmail.

The .NET initiative includes a suite of products, from .NET Enterprise servers to the Visual Studio .NET development platform, that are centered on XML Web services. But before we move on to a description of these products, let us examine two main benefits that the .NET initiative offers.

Interoperation of client devices

Microsoft software, such as Windows CE .NET and Windows XP, can be used to operate handheld computers, laptops, and PCs. (Windows CE .NET is the next version of Windows CE that's used for wireless communication.) Windows XP extensively uses XML to implement features such as remote assistance and Web publishing. On the other hand, Windows CE .NET supports XML 3.0, thereby enabling you to access Web services over mobile devices. Therefore, by using these software products, you can access Web services on client devices, such as laptops and handhelds. Web services can provide important data to these client devices so that the data is easily accessible.

Enhancements to user experience

.NET offers a good opportunity to enhance user experiences. In fact, this is the essential quality of the .NET initiative. Imagine the convenience to a customer who doesn't have to stop at the next ATM to transfer money from one bank account to another. Such user experiences are possible by using XML Web services. Examples of some existing XML Web services that enhance user experiences are the Microsoft Passport authentication service, and the Remote Assistance feature of Windows XP.

Solution providers can subscribe to the Microsoft Passport authentication service to enable Passport authentication on their Web sites. When you log on to a Passport-enabled Web site, your log-on information is stored on your computer as a cookie. When you browse to another Passport-enabled site, the cookie enables the other site to recognize you as an authenticated user and provide customized services. This mechanism is depicted in Figure 1-1.

The advantages of the Passport authentication service extend beyond just being able to use the same log-on information on more than one site. The user and the service provider can derive many advantages, including

- **Ease of deployment:** A solution provider subscribing to the Microsoft Passport authentication service doesn't have to deploy the infrastructure to host and maintain the authentication service. This saves deployment and operational costs.
- **Customized service:** The information provided by a user on one Web site can be accessed on another site. Consequently, solution providers can offer a complete package of services customized for specific users.

- **Better market coverage:** If a solution provider is associated with a well-known service, the market coverage of the solution provider is enhanced. For example, if you trust the Microsoft Passport authentication service, you can be confident when supplying sensitive information, such as a credit card number, to a Web site that uses this service.

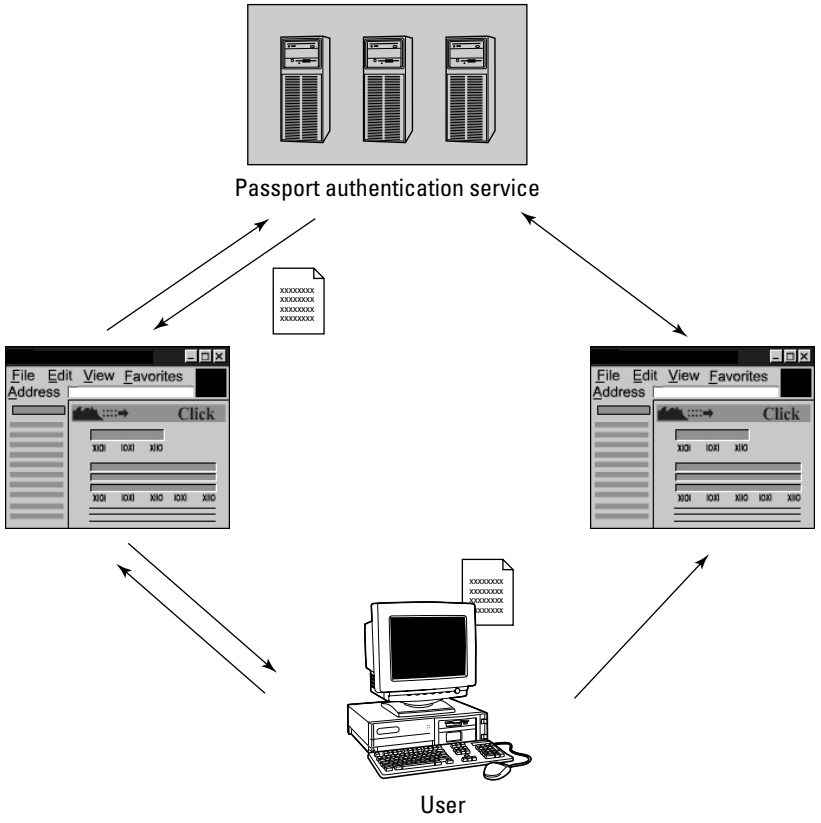


Figure 1-1: More than one Web site can benefit from the Passport authentication service.

Products in the .NET Suite

Microsoft developed a range of products and services for its .NET initiative. Ranging from operating systems to Enterprise servers and development platforms, Microsoft offers a complete suite to build and deploy applications for the .NET initiative. We cover these products briefly in the following sections.

.NET Enterprise servers

The .NET initiative, upon complete implementation, will require many processes to run all applications simultaneously and manage the flow of information between applications and services.

Microsoft developed .NET Enterprise servers as a set of servers to help you achieve high interoperability and availability. The .NET Enterprise servers are Windows.NET Server, Application Center 2000, BizTalk Server 2000, Commerce Server 2000, Content Management Server 2001, Exchange Server 2000, Host Integration Server 2000, Internet Security and Acceleration Server 2000, Mobile Information 2001 Server, SharePoint Portal Server 2001, and SQL Server 2000. In Figure 1-2, you can view each of these servers along with their roles in the .NET Enterprise environment.

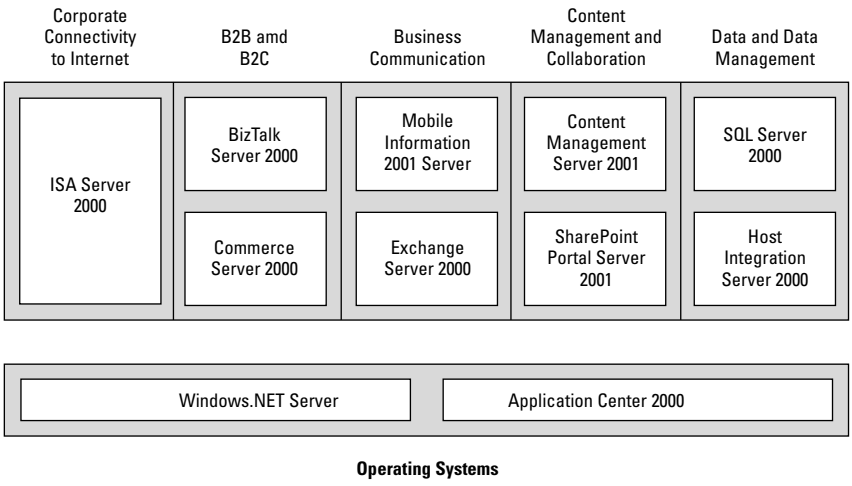


Figure 1-2: Each .NET Enterprise server has a specific role.

We briefly describe each of these servers here.

Windows.NET Server

Windows.NET Server is the operating system for other .NET Enterprise servers. It has a native support for XML and SOAP (Simple Object Access Protocol). The server also integrates with the Microsoft Passport service so that users can use a single log-on ID for accessing all sites that subscribe to this service.

SOAP is an XML-based protocol that's used for exchanging information in a distributed setup. The protocol includes three parts that describe the message that's being transmitted and the rules to process the message.

Application Center 2000

Microsoft Application Center 2000 is a high-availability management and deployment tool for Web applications built on Microsoft Windows 2000. Application Center 2000 provides clustering and is designed for customers who need Web applications with high scalability and availability.

A *cluster* is a group of servers that works like a single unit. By implementing clustering, Application Center 2000 exposes a group of Web servers as a single server on the Internet. If one or more servers in the cluster fail, other servers can take the incoming traffic and prevent a site from going offline.

Application Center 2000 supports Network Load Balancing (NLB) and Component Load Balancing (CLB). NLB is the same as a cluster of Web servers. NLB enables Application Center 2000 to remove a server from a network when that server fails. CLB organizes and manages Web and COM+ applications by using a centralized management console.

BizTalk Server 2000

The advent of .NET has created a key business requirement for the integration of business processes running across different organizations. BizTalk Server 2000 enables you to accomplish exactly that. The server provides extensive support for XML and uses advanced technologies, such as Visio 2000, to describe business processes. BizTalk Server 2000 presents advanced capabilities to manage and exchange data in the XML format as well as access data in non-XML format.

Commerce Server 2000

Commerce Server 2000 is the Microsoft solution to the high customizability needs of e-commerce Web sites. Organizations are deploying Commerce Server 2000 to deliver personalized content to users. The strength of the software is in its ability to provide customized solutions to businesses.

Microsoft also provides two solution sites with Commerce Server 2000. These sites can be used to develop business-to-business (B2B) and business-to-commerce (B2C) Web sites. The coding of solution sites and other Web sites deployed on Commerce Server 2000 is done using Active Server Pages (ASPs). Therefore, Commerce Server 2000 provides a good opportunity to create Web applications in ASP.NET for deploying Commerce Server Web sites.

Content Management Server 2001

Content Management Server 2001 is used to manage content published on Web sites. The server includes sample Web sites and the customization code that you can use to create dynamic Web sites to deploy content.

These Web sites can include presentation templates that help you plan site design and layout. You can apply different presentation templates to quickly change the appearance of a site.

Content Management Server 2001 integrates with other .NET Enterprise Servers, such as Windows 2000 Advanced Server, Microsoft SQL Server 2000, and Microsoft Commerce Server 2000.

Exchange Server 2000

Exchange Server 2000 is used to manage networking and messaging infrastructure. The server offers built-in calendar services, contact and task management capabilities, and discussion groups. When an organization deploys Exchange Server 2000, users can access their e-mail messages, schedules, and contacts through any Web browser. This eliminates constraints to information availability.

Exchange Server 2000 also offers extensive development opportunities. These include unified support for XML that enables vendors to develop solutions for Exchange users. Users can employ these solutions to access important information on devices such as mobile phones and palmtops.

Host Integration Server 2000

Microsoft Host Integration Server 2000 can help you access data from legacy systems. Thus, data on systems such as AS/400, Unix, and DB2 can be made available for an enterprise solution. By using Host Integration Server 2000, you can create distributed applications that best utilize the information on host systems.

Internet Security and Acceleration Server 2000

Internet Security and Acceleration Server 2000 (ISA Server 2000) is used to manage corporate connectivity to the Internet. As the name suggests, ISA Server imparts security and accelerated speed during Internet access.

To enable security with ISA Server 2000, a system administrator can limit the Web sites accessible to corporate employees. The server has built-in security mechanisms to prevent unauthorized users from accessing a corporate network and breaching its security.

ISA Server 2000 also speeds up Internet access by caching Web pages that are visited frequently. Consequently, when a user requests for a cached Web page, the page is retrieved from the internal cache instead of the Web site. This enables organizations to save expenses of unnecessary connectivity to the Internet.

Mobile Information 2001 Server

Microsoft Mobile Information 2001 Server (MIS 2001) is a gateway for mobile users to access their corporate data and the intranet. MIS 2001 also includes Outline Mobile Access in its integrated package. This enables users to access information, such as e-mail messages, tasks, calendars, and contacts on mobile devices by using Microsoft Outlook.

SharePoint Portal Server 2001

SharePoint Portal Server 2001 helps you create Web portals for collating information from different sources into a central location. SharePoint Portal Server 2001 offers a number of features, such as version tracking, document publishing, and controlling role-based access to help you streamline document management.

Just like Visual SourceSafe 6.0 (VSS), SharePoint Portal Server 2001 records the history of a document to help you track changes and eliminate the possibility of a user changing another user's modifications. To edit a document, a user must first check out the document. This prevents other users from changing the document until the first user checks it in. Each time that a document is checked in, a new version number is assigned to the document and the previous version is archived.

SQL Server 2000

Microsoft SQL Server 2000 is a relational database management system (RDBMS) that provides improved turnaround time, lower transaction costs, and high availability. SQL Server 2000 provides a graphical user interface (GUI) and a development environment for creating data-driven applications.

SQL Server 2000 has built-in support for the XML format. You can import XML-format data to databases and export data from database tables to an XML format. XML processing capabilities make SQL Server 2000 an effective solution for providing data access in XML Web services.

.NET Framework

The .NET Framework is a platform for creating XML Web services. It provides the necessary classes, namespaces, and assemblies to create such applications. The .NET Framework consists of three components:

- **Common language runtime (CLR):** The common language runtime manages execution of code at runtime. The CLR ensures efficient memory and thread management and safety of the executing code. CLR is so named because it ensures interoperability between programming code

that's written in different Visual Studio .NET applications. As a result, you can run an application coded in one language in another language. You can read more about cross-language interoperability in Book 1, Chapter 2.

- **Class library:** The .NET Framework includes the class library that has a comprehensive collection of object-oriented classes that you can use to develop Windows and Web applications.
- **ASP.NET:** ASP.NET is an important component of the .NET Framework. ASP.NET provides advanced capabilities, such as efficient database access and easy-to-use Application and Session state capabilities. ASP.NET applications can be created using any .NET language, such as Visual Basic .NET or Visual C# .NET (that's *C sharp*, not *C pound*)

By including ASP.NET in .NET Framework, Microsoft ensures that you can code in ASP.NET without using the Visual Studio .NET development platform. However, you'll find it easier to code ASP applications in Visual Studio .NET because Visual Studio .NET provides server controls that make it easy to code.

A simple way to shift from your existing ASP applications to ASP.NET is by changing the file extensions from `.asp` to `.aspx`. This ensures that your application runs in the .NET Framework environment. Subsequently, you can update the code incrementally to optimize your application for Visual Studio .NET.

See Book 1, Chapter 2 for the details on the components of the .NET Framework that we describe here.

The .NET Framework is essentially the backbone of all your endeavors in Visual Studio .NET although you may not directly use it in your journey through Visual Studio .NET. This framework provides key components that are used by Visual Studio .NET to achieve interoperability and tight integration between programming languages. For example, you use the classes of the .NET Framework to develop Visual Studio .NET applications.

Visual Studio .NET

Visual Studio .NET, the Microsoft platform for developing Web services and Windows applications, comprises a number of languages that share a common set of classes and a development environment. The platform consists of Visual Basic .NET, Visual C#, Visual FoxPro, and Visual C++ .NET.

Take a look at the bigger picture of the .NET initiative to help you see its components, Visual Studio .NET and the .NET Framework, in the right perspective. Figure 1-3 depicts the role of Visual Studio .NET in the .NET initiative.

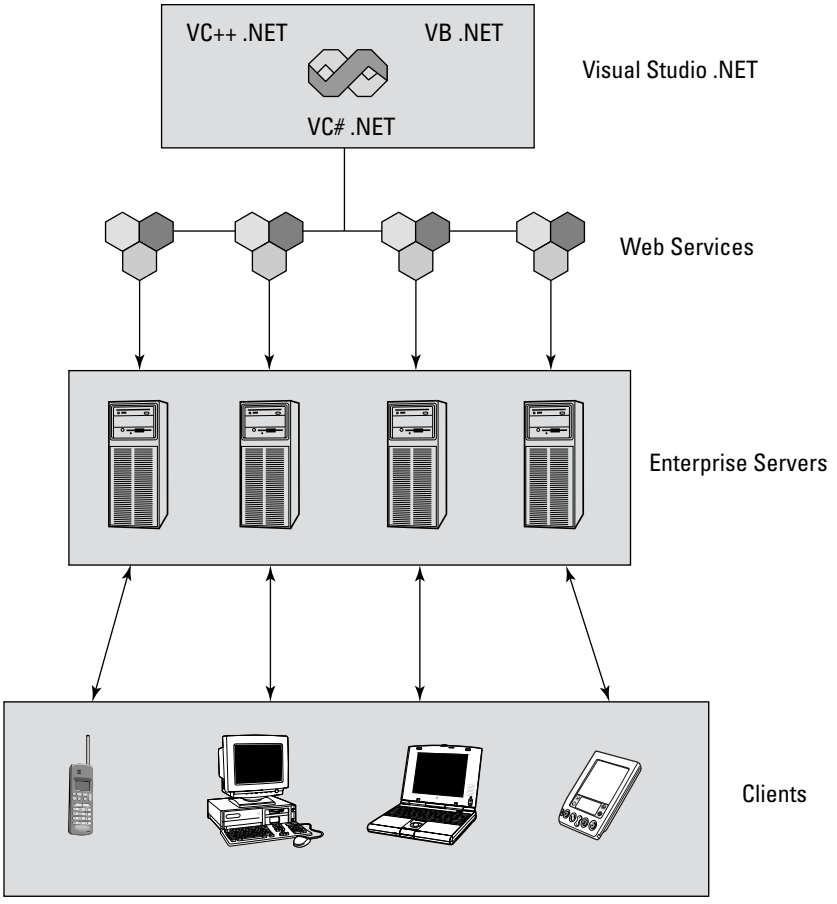


Figure 1-3: Visual Studio .NET is an important component of the .NET initiative.

Note these two immediate advantages of Visual Studio .NET. (For more on the advantages of Visual Studio .NET, turn to Book 1, Chapter 4.)

- **Common IDE across languages:** All Visual Studio .NET languages have the same IDE (Integrated Development Environment). Therefore, after you grasp the basics of working on one language, it's easy to switch to and share tools and applications across languages. This is a feature distinct from Visual Studio 6.0, in which Visual Basic and Visual InterDev have an IDE different from that of other languages. We discuss more on IDE in Book 2.

- **Easy ASP programming:** Visual Studio .NET also compliments ASP.NET programming in the .NET Framework. For example, when you use Visual Studio .NET to create ASP.NET Web applications, you can take advantage of tools such as WYSIWYG (what you see is what you get) editors for Web pages and code-aware editors for statement completion to develop your application. Visual Studio .NET also enables you to compile and debug your ASP.NET application when you create it. These features give you a welcome break from the tedious process of manually creating ASP Web pages by using a text editor, such as Notepad, and then deploying applications on IIS (Internet Information Services).

WYSIWYG editors enable you to know the output of your application as you design it. For example, if you use Microsoft FrontPage 2000, you'll know how your Web page will look as you create it.