NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

E_COMMERCE WEB SITE USING ASP.NET

Graduation Project
COM400

Student: Kıvanç Meram (20010618)

Supervisor: Mr. Kaan Uyar

Nicosia-2005

ACKNOWLEDGEMENTS

It is my pleasure to take this opportunity to express my greatest gratitude to man individuals who have given me a lot of supports during my four-year Undergraduation program in the **Near East University**. Without them, my Graduation Project would not have been successfully completed on time.

First of all, I would like to express my thanks to my supervisor **Mr. Kaan Uyar** for supervising my project. Under the guidance of him I successfully overcome many difficulties and I learned a lot about web designing. He always helped me a lot and I felt remarkable progress during his supervisior. Also I thank for giving his time during the my study and my advisering.

Third,I also want to thank all my friends and specially **Osman Yüksel, Caner Çakır, Ergin Özgen, Aykut Yapıcıkardeşler, Mustafa Sıkıcı, Buğra Demiricioğlu,H. Irmak and Emrah Erdumlu, Elif Gökçen, Deniz Akaltun, Cem Muzaffer Hançer,** who supported and helped me all the time.

Finally, special thanks for my family, especially my parents for being patientfull during my undergraduate degree study. I could never have completed my study without their encouragement and endless support.

ABSTRACT

Commerce Web sites were once considered the great frontier of untapped profit and potential. Build one and you could get rich. The site didn't work properly or bring in paying customers. People soon realized the fallacy of this belief, and the pendulum swung in the opposite direction. Commerce sites came to be through of as a guaranteed money-losing proposition, a sink hole into which you could pour money and never see a return.

With the exception of the eternal optimists who still believe that commerce sites will always succeed and pessimists who still believe they will always fail, most people now understand the reality: commerce sites are an integral part of doing business today. There is no guarantee, but a well-built commerce site that follows a solid business plan has a good chance of contributing to an organization's success.

TABLE OF CONTENTS

INTRODUCTION

Now a days the computer science both hardware and software is being developed over the previous years, programming is always providing the scientific by a systematic development.

Electronic commerce is electronic trading, in which a supplier provides goods or services to a customer in return for payment. A special case of electronic trading is electronic retailing, where the customer is an ordinary consumer rather than another company. Howevwer, while these special cases are of considerable economic importance, they are just particular examples of the more general case of any form of business operation or transaction conducted via electronic media. Other equally valid examples includes internal transactions within a single company or provision of information to an external organisation without charge.

Simply we can say that the electronic commerce is modern way to make you shopping throug the internet.This project, I'm trade to apply of web-based commerce method. I made to write on online e-commerce site. On those pages, The customer can be ordered books.

For the implementation of the project, I used a windows based operation system, Windows XP, Internet Information Server (IIS V5.1). The program language I was used Active Server Pages.NET (ASP.NET) with Visual Basic.NET. As tools for implementation and debugging I used ASP.NET Web Matrix, Microsoft Visual Studio.NET 2003 and Microsoft .NET Framework SDK v1.1, for animations Swish Max, for storing database tables Microsoft SQL Server 2000 Personal Edition.

CHAPTER ONE

.NET FRAMEWORK

1.1. Overview of The .Net Framework

The .NET Framework is designed from the ground up to allow developers of both Web and traditional applications to build their applications more efficiently and enable them to work more flexibly. One of the most significant features of the .NET Framework is that it enables code written in multiple languages to work together seamlessly. Figure 1.1 shows the structure of the .NET Framework at a very high level.

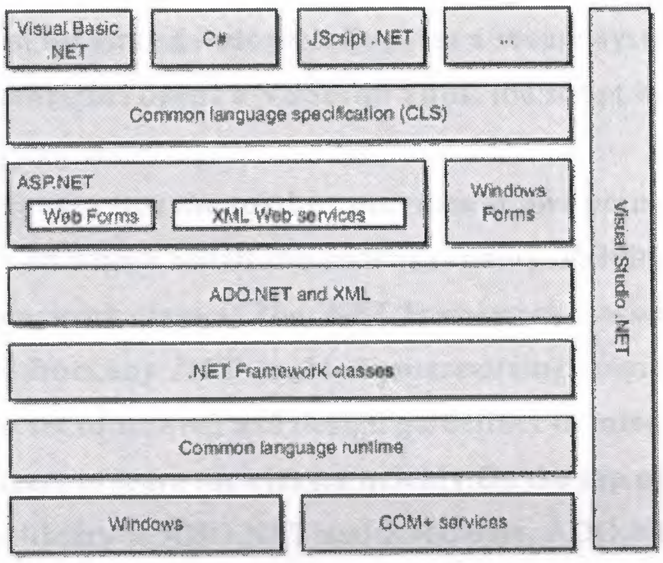


Figure1.1 The .NET Framework architecture

Underlying the entire framework are system services. In the current implementation, this base is the Win32 API and COM+ services, although the abstraction would allow any operating system to provide the services, in theory if not in practice. Traditionally, applications have called the operating system's API directly. In the Win32 programming world, this model is difficult for Visual Basic programmers because some APIs require using data structures that are convenient for C/C++ programmers but much less convenient for Visual Basic programmers.

Layered on top of the system services is the common language runtime. The runtime loads and runs code written in any language that targets the runtime. Code targeted to the runtime is called managed code.

The runtime also provides integrated, pervasive security. Previous Win32 environments provided security only for file systems and network resources, if at all. For example, file security on Microsoft Windows NT and Microsoft Windows 2000 is available only for volumes formatted using NTFS. The runtime provides code access security that allows developers to specify the permissions required to run the code. At load time and as methods are called, the runtime can determine whether the code can be granted the access required. Developers can also explicitly specify limited permissions, meaning that code designed to do something simple and not very dangerous can seek the minimal permissions. Compare this situation to today's VBScript -enabled mail readers, such as Microsoft Outlook, that have been targeted by virus developers. Even on a secure system, if a user with Administrator rights opens a VBScript virus, the script can do whatever the administrator can do.

The role-based security that the runtime provides allows permissions to be set based on the user on whose behalf the code is running. Relying on the runtime are the .NET Framework classes. The .NET Framework classes provide classes that can be called from any .NET-enabled programming language. The classes follow a coherent set of naming and design guidelines in mind, making it easier for developers to learn the classes quickly.On the top of the .NET Framework class library is ADO.NET and XML data. ADO.NET is a set of classes that provide data access support for the .NET Framework. ADO.NET is based on ADO but is designed to work with XML and to work in a disconnected environment.

On top of ADO.NET and XML lies specific support for two different types of applications. One is the traditional client application that uses Windows Forms, a combination of what Visual Basic and the Microsoft Foundation Class Library (MFC) had to offer. The other type of application available is ASP.NET, including Web Forms, and XML Web services.On top of ASP.NET and the Windows Forms is the common language specification (CLS) and the languages that follow the CLS. The CLS is a set of rules that a CLS - compliant language needs to follow, ensuring that each language has a common set of features.

1.2. Introduction to Microsoft Intermediate Language

Although this description of the workings of ASP.NET and the .NET Framework might sound a lot like a description of the way a Java Virtual Machine (JVM) works, ASP.NET and JVM are different. A Java compiler creates byte code, and that byte code is passed through the JVM at runtime. This approach is slightly different than using an intermediate language to generate native code at runtime, but that slight difference has enormous implications with respect to performance. Java's use of byte code is really nothing new. Long ago, other environments used this same structure and generally failed, partly because the hardware wasn't up to the task and partly just because the Internet didn't exist. What the.NET Framework offers that is genuinely different is code that isn't interpreted at runtime but rather becomes native code that is executed directly. One of Java's strengths (and also something that can drive developers crazy at times) is the tight security the Java/JVM model provides. The .NET Framework provides the same level of security, along with the ability to run native code, provided the user has the proper security clearance.One significant advantage that the .NET Framework offers over Java and the JVM is the choice of programming language. If you target the JVM, you must use Java. Java is a perfectly fine programming language, but it's just one language. Developers comfortable with Visual Basic or C++ would have to spend time learning how to use the Java/JVM model. The .NET Framework allows developers to work in whatever language they're most comfortable with, from Visual Basic and C# to Eiffel and COBOL. In theory, as with Java, MSIL can be compiled and run in any environment that supports the runtime. As of this writing, this environment includes only the Intel architecture running Microsoft Windows, but it's safe to assume that the runtime will become available in other environments as well. What makes the potential for multiple platforms possible is the just-in-time(jit) compiler. Figure 2.1 shows the compilation and execution process.

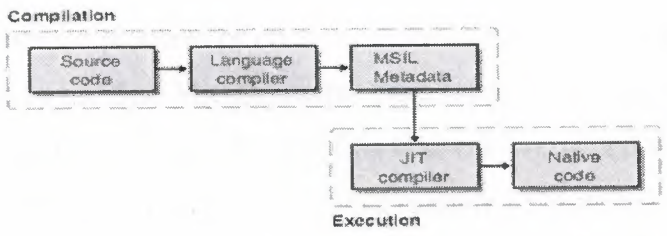


Figure 1.2 The compilation and execution of managed code

When you think about it, compiling an application from assembly code such as MSIL should impose some burden on the performance of the application. In practice, the overhead seems to be a difference small enough that in most cases no one will notice. Part of the reason for this low cost is certainly cleverness on the part of the developers of the JIT compiler, but just as much of the credit goes to the way programs are commonly used. Generally, not every single line of code within a program is used each time the program is run. For example, code related to error conditions might virtually never be executed. To take advantage of this fact, rather than compile the entire MSIL code into a native executable file at the start, the JIT compiler compiles code only as it is needed,and it then caches the compiled native code for reuse. The mechanics of the JIT compilation are fairly straightforward. As a class is loaded, the loader attaches a stub to each method of the class. The first time the method is called, the stub code passes control to the JIT compiler, which compiles the MSIL into native code. The stub is then modified to point to the native code just created, so subsequent calls go directly to the native code.

CHAPTER TWO

The .NET Framework Objects and Languages

2.1. Overview of .NET Framework Objects

When developing real-world systems today, you'll encounter two significant problems: one is the problem of making software work on multiple platforms, and the other is the problem of enabling the various pieces of an application written in different languages to communicate. As you'll see in this chapter, the .NET Framework offers elegant solutions to both these problems. But first, let's review a little history.

One attempt to solve the problem of creating software that will work on multiple platforms has been to use Sun Microsystems' Java programming language. To run Java, a computer must have a Java Virtual Machine (JVM), which will interpret the Java byte code at runtime. Because JVMs are available in browsers for multiple platforms, it would appear that Java has solved part of the problem. In reality, however, there can be incompatibility in the execution of the same Java byte code even on the same platform. For example, in a recent Java project, I needed to use radio buttons but without any text associated with them. I accomplished this by setting the radio button text to an empty string.

This approach worked, but in Microsoft Internet Explorer, when the radio button with no text was selected, a small dotted-line box appeared next to the radio button where the text would have been. The solution seemed simple: instead of not setting the text of the radio button or setting the text of the radio button to an empty string, I explicitly set the text of the radio button to null. This remedy worked for a time. Unfortunately, when a new version of Netscape Navigator came out, setting the text of the radio button to null not only didn't work, but also actually caused the browser to end hard, displaying an error message referencing some C++ source code. So much for Java's cross- platform compatibility.

In the beginning of the PC revolution, cross-platform compatibility was a much bigger requirement. With so many slightly different variants of PCs, as well as other platforms, having a single development environment was very important. Several circumstances have minimized this issue. First, Intel x86 assembly code has become close to a universal assembly language. Virtually any application of any significance these days is available for an Intel-based machine. Even other hardware platforms, notably the Apple Macintosh, provide emulation environments that allow Intel-based applications to run.

The second important change that has affected the issue of cross-platform compatibility has been the explosion of the Internet. The Internet provides a single platform that allows applications from a variety of platforms to work on virtually any other platform, including even the newer ones, such as wireless devices. For many applications, HTML, along with client-side JavaScript, provides a rich enough environment. Of course, in some places, the Internet boom has increased the requirement for cross-platform execution— notably in creating richer user interfaces on the client side—and here's where Java has found a place.

As I mentioned at the beginning of the chapter, another difficulty for software developers today is enabling the various pieces of an application written in different languages to communicate. Currently, a number of languages and technologies are used on the dominant platform (Microsoft Windows running on an Intel processor). Common languages include Microsoft Visual Basic, C/C++, and Borland Delphi. Less common, but still used, are languages such as COBOL, Fortran, and PERL.

From the first days of Windows development, it has been possible to call into dynamic- link libraries (DLLs) from virtually any significant program development environment, but that doesn't mean it's always been easy. For example, something as simple as passing a string as a parameter that will accept some information can cause great problems. In most programming languages, you must ensure that before the string is passed in, it has sufficient allocated space. This task isn't something that many programmers in some programming environments are used to doing. For instance, in Visual Basic, strings are managed, and if you pass a string into another function by reference, the string can have information added to it without worrying about who allocated the space. User-defined data types are much worse, and on at least one occasion not so long ago, the way that Visual Basic padded members of user-defined types wreaked havoc on many a program that had relied on structures being packaged just so.In recent years, COM has been the glue that holds components from the various languages together. COM provides a least common denominator approach to things like data types and does nothing to address issues involved with using the Win32 API. Using the Win32 API from Visual Basic requires some very un-Visual-Basic-like data structures, and the Win32 API can often be difficult to use from other languages as well. The string type supported by COM is *BSTR*, a not entirely friendly type for C/C++ programmers.

The .NET Framework offers solutions to all these problems. First it provides a system of data types that can be marshaled between multiple .NET languages without any loss of fidelity. Developers using the .NET Framework will no longer have to worry about what language might be consuming the class or component they're writing. They can spend more time solving the problem at hand and less time worrying about how the C++ client for the server program is going to interpret a string or a currency type.

Next the .NET Framework provides a virtual execution environment that addresses the need for portability without forsaking performance. Applications built on the .NET platform run as native applications on whatever platform they're running on. I'll explain the technological magic that allows this to occur in the following sections.

2.2. The .NET Solution to Type Compatibility

One of the traits that distinguishes any great programming environment is a well thought out object model. It's difficult to work with a patchwork of poorly designed objects and continue to create world-class software. Given a good object model, you can easily extend it with your own code.

The underlying support for the object model of the .NET Framework is the type system the framework offers.

Let me clarify a few terms here. When I talk about the *type* of a variable, I'm talking about what the variable is designed to hold. For example, if a variable is an integer type, you wouldn't expect that setting it equal to "dog" or "Fred" would work. Likewise, if the type were a date type, 7/24/1956 would be a reasonable value, but 7 wouldn't be. Classic Active Server Pages (ASP) programmers are used to a development language that doesn't use variables with types. More accurately, *every* variable is a single type: *Variant*. Thus, a variable can hold 7 in one line and "Fred" in the next.

Many beginning programmers find having a single data type convenient, but more experienced programmers realize the mess that this limitation can cause. Although forcing you to explicitly change variables from one type to another can be more work, it does ensure that you're converting a variable in a way you intended.

Figure 2.1 shows the relationship between the various types the .NET Framework supports.

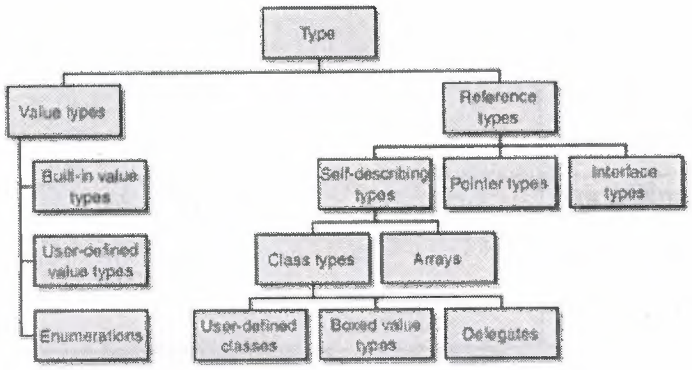


Figure 2-1 : The .NET Framework type system

2.2.1. Value Types

Value types refer to generally small types that are represented as a series of bits. For example, native C/C++ and Visual Basic 6.0 both have *int* and *long* types used to represent numbers. These types are commonly used for much of the processing within any program.

Class Name	**CLS Compliant**	**Description**
System.Byte	Yes	Unsigned 8-bit integer
System.Sbyte	No	Signed 8-bit integer
System.Int16	Yes	Signed 16-bit integer
System.Int32	Yes	Signed 32-bit integer
System.Int64	Yes	Signed 64-bit integer
System.Uint16	No	Unsigned 16-bit integer
System.Uint32	No	Unsigned 32-bit integer
System.Uint64	No	Unsigned 64-bit integer
System.Single	Yes	32-bit Floating point number
System.Double	Yes	64-bit Floating point number
System.Boolean	Yes	True or False value
System.Char	Yes	Unicode 16- bit character
System.Decimal	Yes	96-bit decimal number
System.IntPtr	Yes	Singed integer that is platform dependent
System.UintPtr	No	Unsinged integer that is platform dependent
System.Object	Yes	Root Object
System.String	Yes	Fixed length string of Unicode Chacters

Table 2-2 : Various Value Types in the .NET Framework

CHAPTER THREE

Visual Basic.Net

3.1. Overview of Visual Basic .NET

Seldom has a company the size of Microsoft taken such a chance with one of its flagship products as Microsoft has done with Visual Basic .NET. Visual Basic .NET maintains much of the ease of use that has made Visual Basic famous, but it does so while breaking virtually all existing programs. Furthermore, ASP programmers accustomed to Visual Basic Scripting Edition (VBScript) face a learning curve to be able to take full advantage of what Visual Basic .NET has to offer.

That said, the changes to Visual Basic should also silence the critics who often berate Visual Basic as a toy language. Among the major complaints of programmers who are not fans of Visual Basic is the error handling—sometimes called "On Error Goto Hell" error handling. In fact, the Visual Basic error handling *can* be made to work correctly, but in practice, it's difficult to get right, and it's often handled badly. VBScript's error

handling was even more limited, making the error handling available in Visual Basic look good, which was bad news for ASP programmers. The changes to Visual Basic's error handling are just one of several areas in Visual Basic that have improved dramatically inVisual Basic .NET, albeit at the cost of compatibility with all existing code.

3.2 Out with the Old!

In many respects, Visual Basic is a victim of its own success. There's a joke about the universe being created in seven days: God was able to do it because there was no installed base. I expect the Visual Basic team can appreciate this punch line all too well. Making changes in the primary development platform for many Windows developers is a tricky business. Each new version has brought along new features but for the most part has allowed older code to continue to function. Visual Basic .NET marks a break with that tradition.

Such drastic changes are required for a lot of reasons. The most significant is that the underlying platform Visual Basic .NET is written for is no longer Win32 but rather the.NET Framework. This in and of itself requires many changes. For example, although it's possible to use the exception handling offered by the .NET Framework while continuing with the earlier "On Error Goto" model, doing so would have been at the price of fully exploiting the new framework. Before we get into all the new features offered, it's worthwhile to take a moment to look at the two biggest compatibility issues between Visual Basic 6.0 and Visual Basic .NET, which involve the *Set* statement and the default calling convention.

3.3 The *Set* Statement Goes Away

One of the many areas in which Visual Basic could be confusing to newcomers was in its use of the *Set* keyword. For example, if you wanted to create an instance of an ActiveX control with the ProgID *Foo.Bar*, you would use code such as the following:

```
Dim foo As Foo.Bar
Set foo = New Foo.Bar
```

Creating an object requires using the *Set* keyword. Unfortunately, many developers don't have a good understanding of what exactly is and is not an "object" from the Visual Basic point of view; I've seen more than a few programmers who play with using or not using the *Set* keyword in a vain attempt to get their programs going. Sometimes the result is a working program, because the presence or absence of the *Set* keyword was the problem, but as often as not the real problem remains hidden until you look at the code more closely.

Why was the *Set* keyword ever used? In Visual Basic 6.0 and earlier, objects had default properties that didn't require a parameter. So if an object *foo* had a parameterless default property called *bar*, without using *Set* there was the chance for ambiguity, as in the following example:

```
Dim f as foo
Dim o as Object foo =o
```

In this case, it's unclear whether *foo.bar* should be set to *o* or whether *foo* should be set to *o*. Visual Basic .NET eliminates the need for using *Set* by eliminating parameterless default properties. More than eliminating the need for *Set*, in Visual Basic .NET, the *Set* keyword is no longer allowe

3.4. Default Parameter Calling Conventions

The second area that will require significant source code changes in the move to Visual Basic .NET involves changes to the way parameters are passed to functions and subroutines. In earlier versions of Visual Basic, by default, all parameters were passed in by reference. A parameter passed in by reference means that instead of getting a copy of the parameter, the parameter is really a pointer to the parameter being passed.

Consider the following code that could be used in Visual Basic 6.0:

```
Private Sub Command1_Click()
Dim l As Long
Dim OldL As Long
Dim t As Long
l = CLng(Timer()) OldL = l
t = CallingByReference(l)
MsgBox "l was " & CStr(OldL) & " but is now " & l
End Sub
Function CallingByReference(Ref As Long) As Integer
Ref = Ref Mod 60
CallingByReference = Ref
End Function
```

Running this code any time (after 12:01 a.m.) will result in two different values, as

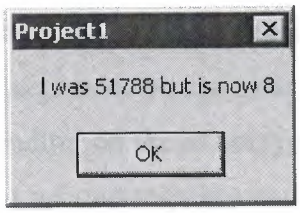

```
Project1
l was 51788 but is now 8
OK
```

Figure 3-1 : Message box displayed after calling *CallingByReference*

The ability to modify parameters is often useful, but it can sometimes confuse beginners. For example, a beginning programmer glancing at this code won't see the relationship between the variable *l* and the variable *Ref* in *CallingByReference*.

Of course, in Visual Basic 6.0 and earlier, you could always declare the parameter explicitly to be passed by value. Here's a Visual Basic 6.0 function that uses call by value:

```
Function CallingByValue(ByVal Ref As Long) As Integer
Ref = Ref Mod 60
CallingByValue = Ref
End Function
```

By using the *CallingByValue* function rather than *CallingByReference*, the *l* value isn't modified. Figure 3-3 shows a sample message box after using *CallingByValue* instead.

It's good form to explicitly declare the calling convention to avoid any confusion, and that will be the standard for the Visual Basic .NET programs that follow.

3.5. In with the New!

Although for some, the break with compatibility will be the big news about Visual Basic.NET, the far more important news is about the improvements to the language. The pain of the compatibility breaks will be temporary, but the gain from the new features will be long lasting. For developers familiar with working under the constraints of VBScript in ASP, the improvements are nothing short of earth shattering.

3.6. Inheritance and Polymorphism

In recent versions, Visual Basic has tried to become a more object-oriented language, with some success. To be considered object oriented, a language must meet three primary requirements. The language must be *polymorphic*, meaning that you can call a method of an object and, depending on the exact type of the object, different underlying methods are called. A second requirement for a language to be considered object oriented is *encapsulation.* Encapsulation means that there is a separation between what the object exposes and the internal workings of the object.

For example, if an object exposes a collection of strings, it shouldn't expose details of implementation, such as whether the collection of strings is stored in an array, a linked list, or a stack. Perhaps the most important requirement is *inheritance.* Inheritance is the ability to derive one type from another. For example, given a simple class:

```
Public Class Base
Public Function foo
System.Console.Writeline("Base Foo")
End Function
End Class
```

we could create another class:

```
Public Class Derived
Inherits Base
Public Function bar
System.Console.Writeline("Derived Bar") End Function
End Class
```

If we created an instance of class *Derived* and called *foo*, "Base Foo" would be displayed on the console.

Inheritance is a convenient way to reuse code, well beyond the cutting and pasting that has often been the standard technique for code reuse in the past. For example, imagine a set of classes representing shapes. All shapes have some characteristics in common—for instance, they might have a position as well as a length and a width. You might also have some common actions that the shapes would take—for example, *Draw* or *Move*. Using inheritance, a hierarchy of shapes could be created, all originally descended from the class *Shape*, which might look like this (in abbrevi ated form):

```
MustInherit Class Shape
Private myX as Integer
Private myY as Integer Public Sub New()
myX = 0
myY = 0
End Sub
Public Property X Get
X = myX
End Get
Set
myX = Value
Draw() End Set
End Property
```

```
Public Property Y
GetY = myY End Get Set
myY = Value
Draw()
End Set
End Property
MustOverride Function Draw()
End Class
Class Square
Inherits Shape
Overrides Function Draw()
' A Square-Specific Implementation
End Function
End Class
```

In this simple example, if you create an instance of class *Square* named *s*, setting the property *s.X* will call the *Set* property as defined in *Shape* and call the *Draw* method that is part of the *Square* class. Furthermore, if the *Square* object *s* is passed to a method that takes a *Shape* object, when *Draw* is called on the object in that method, the *Draw* associated with the *Square* object is called.
Classes can have behaviors with the same name. The ability of the language to determine, based on the type of object, what behavior is used when requested is called *polymorphism.*

3.7. A Word About Multiple Inheritance

Visual Basic .NET doesn't support multiple inheritance—there can be only one *Inherits* keyword per class. In some object models (notably C++),multiple inheritance is used as a way to allow, for example, a *Dog* object to derive from both *Animal* and *Pet.* Single inheritance isn't a terrible limitation, and it eliminates the possibility of method ambiguity. For example, if *Dog* is derived from both *Animal* and *Pet,* and if both hierarchies have a method *MakeNoise,* there could be ambiguity over exactly which method should be called.

You can get around this single inheritance restriction in many ways. In this case, *Animal* could be used as the base class, *Pet* could be derived from *Animal,* and *Dog* could be derived from *Pet.* This is *not* multiple inheritance because there is only a single *Inherits* at each level. (This solution would eliminate the *PetRock* class because a Pet Rock might be considered a pet, but it isn't an animal.)
An alternative solution is to create *Animal*, from that derive a *Dog* class, and then also have the *Dog* class implement the *Pet* interface. An *interface* is like a class except that it contains only methods, and the methods aren't implemented at the interface level. A class can implement an interface simply by declaring that it does implement the interface by using the *Implements* keyword and by providing methods that match each of the methods in the interface. Methods can implement any number of interfaces.

3.8. Structured Exception Handling

There are two general models for handling errors. The first model makes reporting errors the responsibility of any given function, with any code that calls the function responsible for taking action based on the report of an error. This approach is typified by code such as the following:

```
Ret = SomeFunc(SomeParam) If Ret = 0 then
' An error occurred, so do something about it.' End If
Continue processing.
```

This kind of error handling has several problems. Using it often requires error processing to be mixed up with returned results. For example, in C, the *fopen* function returns a file pointer that can be used for other functions that require a file pointer, such as *fgets* and thelike. If the file can't be opened, however, *fopen* returns not a file pointer but *NULL*, indicating that an error occurred opening the file. Thus, the return value from the function is either a file handle *or* something entirely different, a signal that an error occurred.
Many developers can live with the lack of purity of the returned value, but most developers don't always remember to check the return result for the exceptional value indicating an error. In practice, most C programmers *do* check for the return code from calls to *fopen* because it has a fairly large chance of failure. However, many C programmers do *not* check for errors in functions like *fputs* because that function,

using a valid file pointer, fails relatively rarely. Thus, many file writes will fail because the disk is full or, for other reasons, go unnoticed.

The second model for error handling is exception handling. In this sort of system, an error throws an exception and that exception bubbles up the stack until an appropriate handler is found. Although Visual Basic offers a sort of exception handling using the *On Error* statement, the form in which it was exposed wasn't the most convenient. For VBScript programmers in ASP, the options were even more limited because the system didn't allow all the control over exception handling Visual Basic or Visual Basic for Applications (VBA) allowed.

The preferred sort of exception handling is *structured exception handling*. Although structured exception handling is more a feature of the .NET Framework than of Visual Basic .NET per se, it is a critical change that will allow developers to create far more robust and reliable applications. The general form of structured exception handling is shown here:

```
Try
 ' Some code that might throw an exception
Catch e As Exception
' Handle the error. Finally
' Used to do things that should always be done,
' whether or not an exception occurs
End Try
```

Any code that might throw an exception should be placed within a *Try* block. It's possible that some or all exceptions might not be appropriately handled at this level. If that's the case, the exception can be rethrown (using the *Throw* keyword) and the *Finally* block will still be executed. For example, if within the *Try* block you're opening a database connection, the *Finally* block should be where the database connection will be closed because that block of code will always be executed. Within the *Finally* block, you might need to ensure that the database connection is in fact open because the exception could've been thrown before the database connection was successfully opened. You can have multiple *Catch* blocks so that you can catch specific exceptions. The *Finally* block allows all cleanup for the code in the *Try* block to appear in only a single place, rather than existing once for when the code executed normally and once in each of the *Catch* blocks.

3.9. Function Overloading

Function overloading allows the existence of multiple functions with the same name, differing only in the parameters. For example, if you were creating a method to send a string to a browser, you might declare several functions, each named *Write.* One version would accept a string as a parameter; another, an integer; and yet another, a *DateTime* object.

If you're a VBScript programmer, you might wonder what the big deal is. In the old ASP object model under VBScript, you could, for example, call the *Response.Write* method with a string—or an integer, or a date—and it would seem to work as you would expect. There's a subtle difference, however. In VBScript, all variables are the *Variant* type, a sort of chameleon variable that becomes whatever is poured into it. The *Response.Write* method simply takes whatever is passed and writes the resulting string to the HTML stream. Function overloading is different in that the specific *Write* methods provided will be called based on the type of argument. If the *Write* method is called with an argument that can't be converted implicitly to one of the types that one of the overloaded *Write* functions expects, a compile time error is generated.

Overloading can also be used to cleanly extend existing systems without breaking existing code. For example, if a *Write* method exists that accepts a string, and if there were an option to write with a color, a *Write* method that accepts a string and a color could be created. The code inside the existing *Write* method that takes a string could then be replaced cleanly with a call to the new *Write* method that accepts a string and a color, with the color being passed in the default color. Existing consumers of the *Write* method would be none the wiser, and the natural extension to the *Write* method could be used in new code.

3.10. Stronger Typing of Variables

One of the big changes for ASP programmers moving from VBScript to Visual Basic .NET is the introduction of stronger typing of variables. Although it was possible to require declaration of variables in VBScript, it wasn't possible to declare specific types. Statements such as the following were possible:

```
Dim X
X="Hello There" X=7
```

```
Response.Write(X)
```

In this example, the variable *X* is set to a string, and in the next line, it's set to an integer. The result of the *Response.Write* method is a string containing the number *7*. This is possible because all variables in VBScript are the *Variant* type. To help catch potential data conversion errors, Visual Basic .NET has a new statement named *Option Strict* that is stricter than *Option Explicit.* Using *Option Strict* will cause Visual Basic .NET to generate an error if a data type conversion results in data loss, if a variable is undeclared, or if a variable is late bound.. This isn't news to most non-Visual Basic programmers, but for Visual Basic .NET programmers trying to create professional, reliable applications, it's a huge step forward.

3.11. Short-Circuit Evaluation

Another problem that C/C++ programmers coming to Visual Basic face is the way in which logical expressions are evaluated. For example, imagine that you have the following code in an ASP page:

```
While rs.EOF=False And rs("Grouping")=thisGroup
' Do something for all members of "thisGroup".
Wend
```

Programmers used to C and C++ will presume that if *rs.EOF* is *True*, the evaluation of the expression will end. In VBScript and Visual Basic 6.0, this isn't the case. In this example, even if *rs.EOF* is *True, rs("Grouping")* will be evaluated, causing an error to be raised. Of course, once *rs.EOF* is *True*, we really don't care about the other part of the expression because by definition it has no meaning.

Visual Basic .NET includes two new logical operators (*AndAlso* and *OrElse*) that are used for *short-circuit evaluation* of expressions. In the preceding example, you could replace the *And* operator with *AndAlso*:

```
While rs.EOF=False AndAlso rs("Grouping")=thisGroup
' Do something for all members of "thisGroup".
Wend
```

Once *rs.EOF* evaluates to *True*, the program can just stop the expression evaluation because it's guaranteed that the expression in total can never evaluate to *True*. We can use this evaluation order to our advantage by ordering the parts of a logical expression from least expensive to evaluate to most expensive. However, you need to remember that short-circuit evaluation means that parts of an expression might not always run, which can cause side effects. *And* and *Or* continue to operate as they did in Visual Basic6.0 and earlier, forcing evaluation of all parts of a predicate.

3.12. Miscellaneous Changes

Here's a list of some of the other changes in Visual Basic .NET. All arrays use zero-based indexing. There are ways around this, using classes from the .NET Framework, but within the language itself, all arrays are zero-based. One interesting change designed to help port existing code is what happens when you declare an array. Cons ider the following declaration:

```
Dim a(5) as Integer
```

The result will be a *six* element array, from *a(0)* through *a(5)*. This allows developers to continue using arrays as they have in the past.Developers who are creating cross-language components need to be aware of this behavior and explicitly document how the component will base the array.The *Option Base* statement isn't supported.

Arrays don't have a fixed size. Arrays can be declared with a size; declared without a size and sized by calling *New;* or declared, initialized, and sized in a single statement, like this:

```
Dim Month() As Integer = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
```

In Visual Basic .NET, you can resize arrays with the *ReDim* statement. In Visual Basic 6.0, you couldn't resize arrays with a specified size.String lengths can't be explicitly declared.

ReDim can't be used as a declaration. The variable must be declared using *Dim* first.

The Currency data type is no longer supported. The Decimal data type can be substituted.

The *Type* statement is no longer supported. Use the *Structure...End Structure* construction instead. Each *Structure* member must have an access modifier: *Public, Protected, Friend, Protected Friend,* or *Private. Dim* can be used—in which case, the access to the member is *Public.*

Multiple variable declarations on a single line without the type repeated result in all variables on the line being declared as the same type, as in the following example:

```
Dim I, J as Integer
```

In Visual Basic 6.0, this line will result in *I* being a Variant and *J* being an integer, but in Visual Basic .NET, both *I* and *J* are integers.Variables declared within a block of code have block-level scope rather than procedure-level scope. Thus, if a variable *I* is declared within a *While* block, it's visible only within the block. Note that the lifetime of the variable is the same as that of the procedure, so if a block that declares a variable will be entered more than once, the variable should be initialized on each entry to the block.Parentheses are always required when you're calling procedures with nonempty parameter lists.

Rather than *While* and *Wend*, Visual Basic .NET uses *While* and *End While. Wend* isn't supported.

IsNull is replaced by *IsDBNull*, and *IsObject* is replaced by *IsReference.*

CHAPTER FOUR

Overview of ASP.NET

4.1. Internet Standards

Before, we dive into the evolution of ASP, we should review some basic Web client/server fundamentals. At the highest level, communication in a Web-based environment occurs between two entities: (1) a Web client (most commonly a Web browser such as Internet Explorer or Netscape Navigator), which is an application that requests files from a Web server, and (2) a Web server, which is a software application, usually residing on a server, that handles client requests.

It's easy to deduce that a server is a computer that serves something. In a Web environment, a server "serves" HTTP responses. A server generally has more processing power than a personal computer (PC) in order to handle a large number of simultaneous client requests. A Web server is a server that is capable of handling Web, or HTTP, requests. In the Microsoft world, this Web server is one part of Internet Information Services (IIS).

Web browsers and servers communicate using a protocol called Transmission Control Protocol/Internet Protocol (TCP/IP). A protocol is simply a set of rules and procedures that define how two entities communicate. TCP/IP is actually composed of two parts, TCP and IP. TCP, often referred to as a transport protocol, wraps data in a digital envelope, called a packet, and ensures that the data is received in the same state in which it was sent. IP, a network protocol, is responsible for routing packets over a network, like the Internet.

In addition to TCP/IP, Web clients and servers use a higher-level protocol, called HyperText Transfer Protocol (HTTP). To clarify, let us use the analogy of sending a letter through the mail. The letter is analogous to HTTP. When writing a letter, you'll probably write it in a language that the receiver understands, right? So, if you were a Web browser or server you would write your letter in HTTP rather than English. The envelope, which contains a mail-to and return address, is analogous to TCP and your friendly mail carrier is analogous to IP. The mail carrier ensures that your letter is delivered to the correct street address, in the correct city, in the correct state. Likewise, IP ensures that your TCP packet is delivered to the correct IP address.

HTTP is a request-response type protocol that specifies that a client will open a connection to a server and then send a request using a very specific format. The server will then respond and close the connection. HTTP has the ability to transfer Web pages, graphics, and any other type of media that is used by a Web application. Effectively HTTP is a set of messages that a Web browser and server send back and forth in order to exchange information. The simplest HTTP message is GET, to which a server replies by sending the requested document. In addition to GET requests, clients can also send POST requests. POST requests are used most commonly with HTML forms and other operations that require the client to transmit a block of data to the server.

4.2. The Evolution of ASP

Although it may seem as though Microsoft's Active Server Pages (ASP) technology has been around forever, it is actually a relatively new technology, introduced in 1996. Prior to ASP, developers were able to create active Web sites on a Microsoft platform using the Common Gateway Interface (CGI) and Internet Server Application Programming Interface (ISAPI), each of which played a part in the evolution of ASP.CGI was the first widely accepted technique of delivering dynamic Web content. CGI is effectively a method of extending the functionality of a Web server to enable it to dynamically generate HTTP responses using a program typically written in C or a scripting language such as Perl. This allowed page content to be personalized for the user and con- structed from information stored in a database. Although powerful, CGI did have several shortcomings. For each HTTP request that a CGI application receives, a new process is created. After the request has been handled, the process is killed. Repeatedly creating and killing processes proved be a tremendous burden for even the most capable of Web servers.

Along came Microsoft's Active Server platform, which addressed the technical limitations of CGI programming. The Active Server platform was, and really still is, a set of tools that developers can utilize to write Web applications. Microsoft's Active Server platform didn't however originally include Active Server Pages, ASP. Developers were forced to write ISAPI extensions or filters.

ISAPI extensions and CGI are very similar with one major exception. Unlike CGI applications that are generally implemented as executables (EXE) on the Windows platform, ISAPI extensions are implemented as Dynamic Link Libraries (DLLs), which means they are loaded into memory only once, on first demand, and then stay resident in the same process as IIS.

Therefore, ISAPI extensions do not suffer the same performance problems as CGI applications. Additionally, ISAPI extensions are multithreaded, which means that they can manage concurrent requests without degrading system performance.

Like ISAPI extensions, ISAPI filters are multithreaded, implemented as DLLs, and run in the same memory space as IIS. However, ISAPI filters are not invoked by client requests. Instead, ISAPI filters do exactly as their name implies — they filter or intercept and optionally process HTTP requests. ISAPI filters are actually quite useful in many situations, particularly web server logging and security. However, because ISAPI filters act on every HTTP request, they should be used sparingly to avoid severe performance problems.

As useful and powerful as ISAPI extensions and filters are, they can be difficult for novice programmers to develop. ISAPI DLLs must written in C++; and, even though Visual C++ does provide a wizard to assist with the task, this proved to be quite a barrier. Recognizing this issue, Microsoft released several short-lived Active Platform development products that were actually based on ISAPI. These included dbWeb and Internet Database Connector (IDC), which evolved into ASP.

In 1996, Microsoft released Active Server Pages and as they say "the rest is history." ASP allows developers to execute code inline within a Web page. Although, ASP technology is still a relatively new way to create dynamic Web sites, during its short life span, it has evolved to become one of the foremost dynamic Web site development products. This is probably due to the ease with which complex pages and applications can be created, com- bined with the ability to use custom components and existing Microsoft and third party commercial components through the Component Object Model (COM/COM+) architecture.

Since 1996, there have been several versions of ASP. In 1998, Microsoft introduced ASP 2.0 as part of the Windows NT 4.0 Option Pack. With ASP 2.0 and IIS 4.0, an ASP application and its associated components could be created in a memory space separate from the Web servers space to improve fault tolerance. In 2000, with the much anticipated release of Windows 2000 (and IIS

5.0), Microsoft unveiled ASP 3.0. To us, differences between the capabilities of ASP 2.0 and 3.0 appeared to be quite limited. However, running on Windows 2000, ASP's performance was greatly improved.While ASP is powerful and incredibly simple to use, it does have the following drawbacks:

ASP code can get complicated very quickly. ASP code tends to be unstructured and gets really messy. Tons of server-side code intermixes with client-side script code and HTML. After awhile it becomes difficult to figure out what is going on. If you have a few free hours to blow, try reading someone else's ASP code and you'll see what we mean. It can be a truly painful experience.

To do anything in ASP you have to write code. ASP has no actual component model. Developers tend to start at the top of a page and zip right down to the bottom, executing database queries, running business logic, and generating HTML along the way.

Code is mixed with presentation. This causes problems when developers and designers work together. Supporting internationalization and multiple client types is difficult.

The combination of ASP and IIS isn't always the most reliable of platforms. Sorry, Mr. Gates! However, in Microsoft's defense, this instability isn't necessarily — or even probably — a platform issue. Microsoft, by making the Active Platform so open, gave developers the ability to create applications that could quite easily bring IIS to its knees. Developing an ASP application is one thing, developing a good, effi- cient, reliable ASP application is another. Anyway, ASP fault tolerance could have been a little better.

Deploying an ASP application that utilizes COM can be difficult. COM objects must be registered and are locked by the operating system when being used. As a result, managing a production application, especially in a Web farm, or a Web application that utilizes more than one Web server, proved to be quite challenging.

4.3. The Benefits of ASP.NET

Microsoft, realizing that ASP does possess some significant shortcomings, developed ASP.NET. ASP.NET is a set of components that provide developers with a framework with which to implement complex functionality. Two of the major improvements of ASP.NET over traditional ASP are scalability and availability. ASP.NET is scalable in that it provides state services that can be utilized to manage session variables across multiple Web servers in a server farm. Additionally, ASP.NET possesses a high performance process model that can detect application failures and recover from them.

Along with improved availability and scalability, ASP.NET provides the following additionalbenefits:

- **Simplified development:** ASP.NET offers a very rich object model that developers can use to reduce the amount of code they need to write.
- **Language independence:** ASP pages must be written with scripting. In other words, ASP pages must be written in a language that is interpreted rather than com- piled. ASP.NET allows compiled languages to be used, providing better performance and cross-language compatibility.
- **Simplified deployment:** With .NET components, deployment is as easy as copying a component assembly to its desired location.
- **Cross-client capability:** One of the foremost problems facing developers today is writing code that can be rendered correctly on multiple client types. For example, writing one script that will render correctly in Internet Explorer 5.5 and Netscape Navigator 4.7, and on a PDA and a mobile phone is very difficult, if not impossible, and time consuming. ASP.NET provides rich server-side components that can automatically produce output specifically targeted at each type of client.
- **Web services:** ASP.NET provides features that allow ASP.NET developers to effortlessly create Web services that can be consumed by any client that understands HTTP and XML, the de facto language for inter-device communication.
- **Performance:** ASP.NET pages are compiled whereas ASP pages are interpreted. When an ASP.NET page is first requested, it is compiled and cached, or saved in memory, by the .NET Common Language Runtime (CLR). This cached copy can then be re-used for each subsequent request for the page. Performance is thereby improved because after the first request, the code can run from a much faster compiled version.

Probably one of the most intriguing features of ASP.NET is its integration with the .NET CLR. The CLR executes the code written for the .NET platform. The .NET compilers target the.NET runtime and generate intermediate language (IL) binary code (kind of like Java and byte code). The code generated by .NET compilers cannot be run directly on the processor because the generated code is not in machine language. During runtime, the .NET compilers convert this intermediate code to native machine code and that machine code is eventually run on the processor. Additionally, the .NET compilers also produce metadata that describes the code. The .NET runtime loads metadata information for performing different tasks like resolving method calls, loading different dependent modules, marshaling data from one component to another, and so on. Since the .NET runtime produces binary code that is later compiled, effectively any language that is CLR compliant and can generate IL code can be used to write ASP.NET applications and components.

.NET offers many programmatic improvements and features, one of which is a new version of ActiveX Data Objects (ADO) called, not surprisingly, ADO.NET. ADO.NET provides a suite of data handling and binding facilities. The Web is an inherently disconnected environment: a Web application connects to a datasource, manipulates the data, reconnects to the data- source, and updates the data. ADO.NET has been designed to work in a disconnected fashion, which increases data sharing. Additionally, ADO.NET treats data in a very loose, multidimensional, object-oriented way through a strongly typed object model. With ADO, all data is represented in two dimensions, rows and columns. With ADO.NET these n-dimensional data representations of data are called *datasets*. Iterating through, updating, and deleting related tables in a dataset is exceptionally simple.

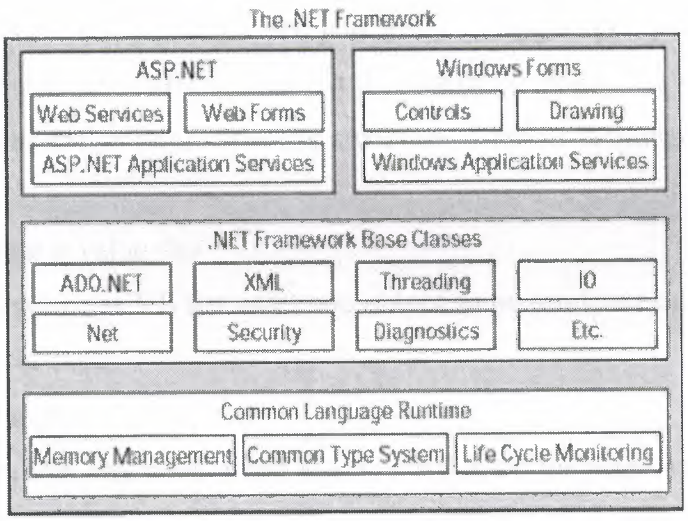


Figure 4.1. The .Net Framework

4.4. Creating an ASP.NET Application

After you've set up the development environment for ASP.NET, you can create your first ASP.NET Web application. You can create an ASP.NET Web application in one of the following ways:

Use a text editor: In this method, you can write the code in a text editor, such as Notepad, and save the code as an ASPX file. You can save the ASPX file in the directory C:\inetpub\wwwroot. Then, to display the output of the Web page in

Internet Explorer, you simply need to type http://localhost/<filename>.aspx in the Address box. If the IIS server is installed on some other machine on the network, replace "localhost" with the name of the server. If you save the file in some other directory, you need to add the file to a virtual directory in the Default WebSite directory on the IIS server. You can also create your own virtual directory and add the file to it.

Use the VS.NET IDE: In this method, you use the IDE of Visual Studio .NET to create a Web page in a WYSIWYG manner. Also, when you create a Web application, the application is automatically created on a Web server(IIS server). You do not need to create a separate virtual directory on the IIS server.

From the preceding discussion, it is obvious that the development of ASP.NET Web applications is much more convenient and efficient in Visual Studio .NET. ASP.NET Web pages consist of HTML text and the code. The HTML text and the code can be separated in two different files. You can write the code in Visual Basic or C#. This separate file is called the *code behind file*. In this section, you'll create simple Web pages by using VB as well as C#. Before you start creating a Web page, you should be familiar with basic ASP.NET syntax. At the top of the page, you must specify an @ Page directive to define page- specific attributes, such as language. The syntax is given as follows:

<%@ Page attribute = value %>

To specify the language as VB for any code output to be rendered on the page, use the following line of code:

<%@ Page Language = "VB" %>

This line indicates that any code in the block, <%%>, on the page is compiled by using VB.

To render the output on your page, you can use the Response.Write() method. For example, to display the text "hello" on a page, use the following code:

<% Response.Write("Hello") %>

You can use HTML tags in the argument passed to the Response.Write() method. For example, to display the text in bold, you use the following code:

<% Response.Write("<B> Hello </B>") %>

For dynamic processing of a page, such as the result of a user interaction, you need to write the code within the <Script> tag. The syntax of the <Script> tag is given as follows:

```
<Script runat="server" [language=codelanguage]>
code here
</Script>
```

4.5. In this syntax . . .

runat="server" indicates that the code is executed at the server side.

[language=codelanguage] indicates the language that is used. You can use VB, C#, or JScript .NET. The square brackets indicate that this attribute is optional. If you do not specify this attribute, the default language used is VB.

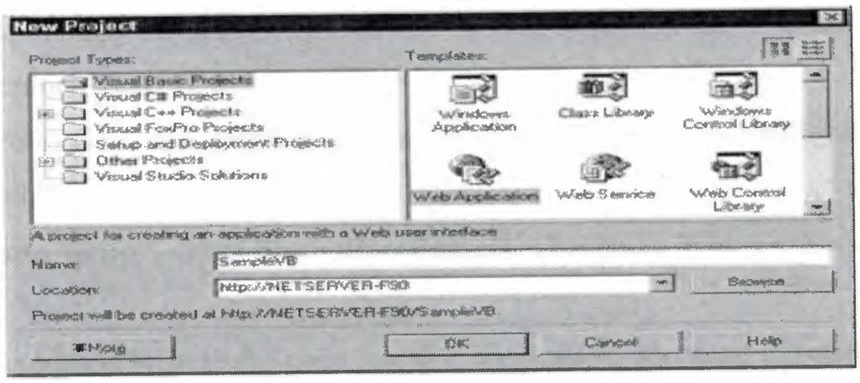
After gaining an understanding of the basic ASP.NET page syntax, you can now create a simple ASP.NET Web application. In the following sections, you'll create a simple Web application by using VB and C#. To do so, you'll use the VS.NET IDE.

4.6. Creating a Visual Basic Web Application

You can create an ASP.NET application using Visual Basic by creating a Visual Basic Web Application project. To do so, complete the following steps:

1. Select File > New > Project. The New Project dialog box appears.
2. Select Visual Basic Projects from the Project Types pane.

3.Select ASP.NET Web Application from the Templates pane. The Name box contains a default name of the application. The Location box contains the name of a Web server where the application will be created. However, you can change the default name and location. In this case, the name of the sample application is SampleVB. The New Project dialog box now appears as shown in Figure 4.2.



A project for creating an application with a Web user interface

Name: SampleVB

Location: http://NETSERVER-F90

Project will be created at http://NETSERVER-F90/SampleVB.

Figure 4.2. The New Project dialog box

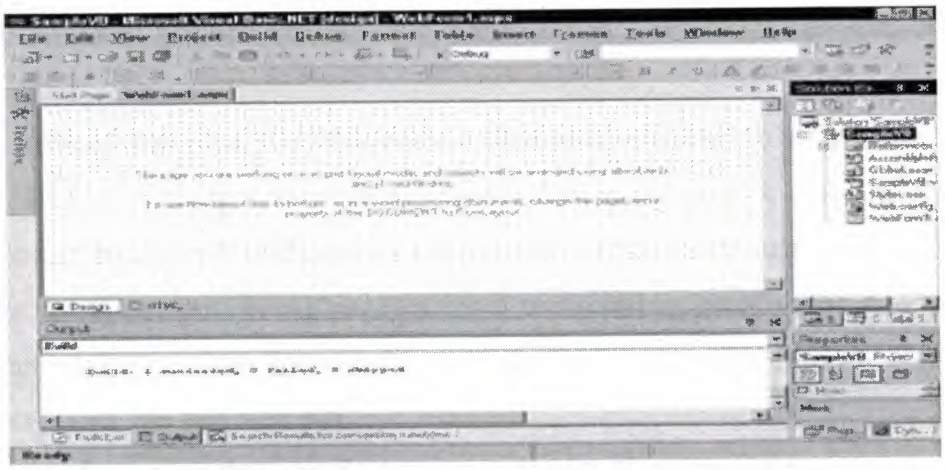


Figure 4.3. The VS.NET window with a new project

The WebForm1.aspx file is displayed in Design mode by default. To view the file in HTML mode, click HTML at the bottom of the WebForm1.aspx file window.
As you can see in HTML view, the language to be used on the page is VB. Any HTML text or code (in the <%%> block) within the <Body></Body> block is rendered on the page when it is displayed in a Web browser.
The default background color of a page is white. You can change the background color of a page by setting the bgcolor attribute of the <Body> element. When you set this attribute, you are prompted to pick the color, as shown in Figure 4.4.

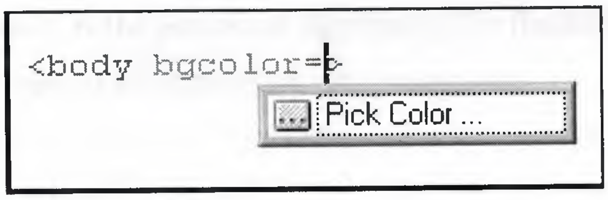

```
<body bgcolor=
```

Figure 4.4. Setting the bgcolor attribute

When you select a color from the color palette, the corresponding color code is set as the value of the bgcolor attribute. A sample of such code is given as follows:
<Body bgcolor="#ccccff">
Write the following code within the <Body> </Body> element to display the text "Hello World":
<% Response.Write(" <Font Size=10> <Center> <B> Hello World </B> </Center> </Font>") %>
After you complete writing the code for your application, you need to build your

application so that you can execute it on a Web server. To build the project, choose Build.

When you build a project, the Web Form class file is compiled to a Dynamic Link Library (DLL) file along with other executable files in the project. The ASPX file is copied to the Web server without any compilation. You can change the ASPX file (only the visual elements of the page) without recompiling, because the ASPX file is not compiled. Later, when you run the page, the DLL and ASPX files are compiled into a new class file and then run.

The output of the page that you developed is displayed in Figure4.5

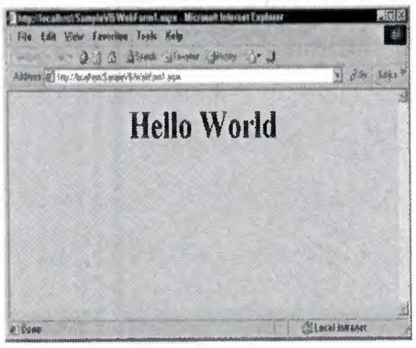


Figure 4.5. A sample output of the Web page

4.7. Deploying an ASP.NET Web Application

After creating and testing your ASP.NET Web applications, the next step is deployment. *Deployment* is the process of distributing the finished applications (without the source code) to be installed on other computers.

In Visual Studio .NET, the deployment mechanism is the same irrespective of the programming language and tools used to create applications. In this section, you'll deploy the "Hello World" Web application that you created. You can deploy any of the application that was created by using VB or C#. Here, you'll deploy the application created by using VB. To do so, follow these steps:

1. Open the Web application project that you want to deploy. In this case, open the SampleVB project.
2. Select File □ Add Project □ New Project to open the Add New Project dialog box.
3. From the Project Types pane, select Setup and Deployment Projects. From the Templates pane, select Web Setup Project.

4. Change the default name of the project. In this case, change it to "SampleVBDeploy."

5. Click OK to complete the process. The project is added in the Solution Explorer window.

Also, a File System editor window appears to the left, as shown in Figure27. The editor window has two panes. The left pane displays different items. The right pane displays the content of the item selected in the left pane.

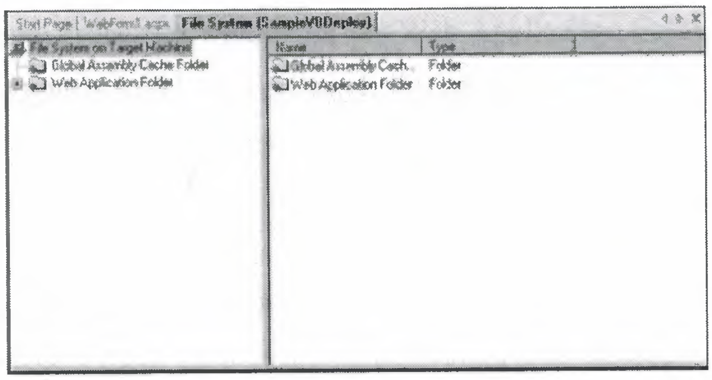

Start Page | WebForm1.aspx | File System (SampleVBDeploy)
File System on Target Machine
Global Assembly Cache Folder
Web Application Folder
Name
Type
Global Assembly Cach...
Folder
Web Application Folder
Folder

Figure 4.6. The Deployment editor

6. Select Web Application Folder in the left pane of the File System editor window. Then, from the Action menu, select Add > Project Output to open the Add Project Output Group dialog box, shown in Figure 4.7.

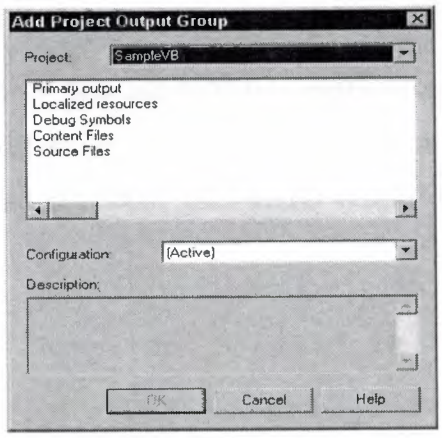


Figure 4.7. The Add Project Output Group dialog box

7. Verify that SampleVB is selected in the Project drop-down list. Then, select Primary Output and Content Files from the list.

8. Click OK. The output files and content files of the SampleVB project are added to the solution.

9. Select Web Application Folder in the File System editor and select Properties Window from the View menu to open the Properties window.

10.Set the Virtual Directory property to a folder, <folder name>, that would be the virtual directory on the target computer where you want to install the application. By default, this property is set to SampleVB Deploy, which is the name of the Web Setup project that you added. In this case, set the property to Deployed Application.

11. In the same Properties window of the Web Application Folder, set the DefaultDocument property to WebForm1.aspx. This property is used to set the default Web Forms page for the application.

12. Build the solution by selecting Build Solution from the Build menu.

13. After the solution is built successfully, a Sample VBDeploy . msi file is created in the Debug directory of the Web Setup project. The default path is \documents and settings\<*login name*>\My Documents\Visual Studio Projects\SampleVB\SampleVBDeploy\Debug\SampleVBDeploy.msi.

14. Copy the Sample VBDeploy.msi file to the Web server computer (c:\inetpub\wwwroot) where you want to deploy the application.

15. Double-click the Sample VBDeploy.msi file on the target computer to run the installer.

After the installation is complete, you can run your application on the target computer. To do so, start Internet Explorer and enter http://<computername> /DeployedApplication in the address box. The "Hello World" page that you developed is displayed.

CHAPTER FIVE

Database for ASP.NET

5.1. What is a Database?

A **database** is essentially an electronic means of storing **data** in an **organized** manner. Data can be anything that a business or individual needs to keep track of and that, prior to computers, could have only been tracked on one or more paper documents. Once stored, data in the database can be retrieved, processed, and displayed by programs as **information** to the reader. The actual structure that a database uses to store data can take one of many different forms, each which offers certain advantages when that information is to be retrieved or updated. In the next section, we will look at how storing the database in a flat file structure differs from a relational database structure, and the advantages and disadvantages that each of those presents.

5.2. The Microsoft SQL Server 2000 Desktop Engine

We use the Microsoft SQL Server Desktop Engine for database development throughout this book. Before wading too far into this topic, it is worthwhile to first understand what Microsoft SQL Server
2000 is, what different editions of it are available, and how the Desktop Engine we will be using in this book compares with other editions of SQL Server 2000.

5.3. Microsoft SQL Server 2000 Defined

Microsoft SQL Server 2000 is a relational database management system that can be used by individuals or businesses for storing and managing data. It also offers powerful functionality for data analysis and reporting. There are actually seven versions of Microsoft SQL Server 2000 to choose from. Two of these, the Enterprise and Standard Editions, are for deployment on servers in production environments. The other five versions each have a special purpose and are not licensed for deployment on production servers. Each of the seven versions of SQL Server are briefly described below:

- **SQL Server 2000 Enterprise Edition** – This is the most comprehensive version of SQL Server 2000 and supports the full set of SQL Server 2000 features. This version is most appropriate for large organizations that need to manage immense amounts of data quickly and efficiently.

- **SQL Server 2000 Standard Edition** – This version of SQL Server 2000 supports many of the available features, with the notable exception of those that enable the quick and efficient management of large amounts of data. Hence, this version is primarily aimed at small to medium sized organizations that do not have the complex database requirements of larger firms. SQL Server 2000 Standard Edition is nonetheless an extremely powerful version of SQL Server and supports Analysis Services (with a few exceptions), Replication, Full-Text Search, Data Transformation Services, English Query, and other advanced SQL Server features.
- **SQL Server 2000 Personal Edition** – This version of SQL Server 2000 supports basically the same features as the Standard Edition, with the exception of transactional replication. Additionally, Analysis Services and Full-Text Search are only available on certain operating systems with this edition. This version is most appropriate for users who spend some time disconnected from the network but access SQL Server data on their local machine while disconnected. A common example would be mobile users – say, a company's sales force who require access to data while out in the field. This version limits the number of concurrent database activities that can be running at any one time. This simply means that it isn't designed to handle a great many users or database activities.
- **SQL Server 2000 Windows CE Edition** – This version of SQL Server 2000 runs on mobile devices that run under Windows CE. It is a compact edition of SQL Server 2000 and allows relational databases to be stored and managed on a Windows CE device for later synchronization with the main database. It also allows users to manage a SQL Server database remotely over the Internet from their CE device.
- **SQL Server 2000 Developer Edition** – This version of SQL Server 2000 supports all available features just like the Enterprise Edition, with the proviso that it not be deployed on a production server. As the name indicates, this version is designed for developers, consultants, and solution providers while developing and testing SQL applications.

- **SQL Server 2000 Evaluation Edition** – This version is a fully functional version of SQL Server 2000 Enterprise Edition that stops working after 120 days. It allows organizations to evaluate the full product without charge.
- **SQL Server 2000 Desktop Engine** – This is a redistributable version of the SQL Server database engine. This means that you can include it in your setup programs for applications that use SQL Server to store data. The Desktop Engine doesn't include any of the SQL Server 2000 graphical user interface tools, such as SQL Server Enterprise Manager, so other products(such as Visual Studio .NET Server Explorer, Access, or SQL Server 2000 APIs) must be used to create and manage databases stored in this version of SQL Server. (Note: This is not the same version as the SQL Server 7 Desktop Edition. The SQL Server 7 Desktop Edition became the Personal Edition in SQL Server 2000. The SQL Server 2000 Desktop Engine was called the **Microsoft Data Engine**, or **MSDE**, in SQL Server 7).

5.4. Creating a Database

The first step in building a database with SQL Server is to actually create the database. That's right. SQL Server is a piece of software that runs on a computer, or server. Once the SQL Server software is installed you can create a database (or databases) with the SQL Server software that is then managed by that SQL Server software. Many people refer to SQL Server as a database, which it is, sort of. SQL Server is actually an application, a Relational Database Management System (RDBMS), which can contain multiple databases.

Creating the database using Enterprise Manager and perform the following steps:

1. Expand the SQL Server Group item, if it isn't already expanded, in the Enterprise Manager tree. Once expanded you should see a list of SQL Servers that are regis- tered with Enterprise Manager.
2. Right-click the SQL Server in which you want to create the Music database.
3. Select New > Database.
4. You see the Database Properties dialog box. On the General tab, enter **Music** in the Name field. The Database Properties dialog box allows you to control other features of your database such as file growth, maximum database size, transaction log files, and so on. For the sake of brevity, accept the defaults.

You have created a SQL Server database using Enterprise Manager. If you want to create a database with T-SQL, follow these steps :

1. Select Start > Programs > Microsoft SQL Server > Query Analyzer to open SQL Server's Query Analyzer.
2. You see the Connect to SQL Server dialog box. Select the SQL Server on which you would like to create the Music database from the SQL Server drop-down box. Select the Use SQL Server authentication radio button. Now enter the appropriate authentication information in the Login Name and Password fields.
3. In the Query Analyzer window, enter the following T-SQL statement:

```
E master
GO
CREATE DATABASE Music ON PRIMARY
( NAME = MusicData,FILENAME = ‘C:\MSSQL7\data\MusicData.mdf’)
```

In step 3, you essentially created a database named Music and specified that the data should be stored in the MusicData.mdf file located, in this example, in the C:\MSSQL7\data directory. The CREATE DATABASE statement accepts many other parameters, such as MAXSIZE, FILEGROWTH, SIZE, and so on. However, again, for the sake of brevity, you used the SQL Server defaults.

5.5. Creating SQL Server Tables

Now that you have a database, Music, you can add tables to it. If you recall from the previous session, the Music database contains several tables including t_bands, t_band_members, t_albums, and so on. Figure 4-3 shows the schema for the Music database.

You are not going to create every table in the Music database, but hopefully, based on the tables you do create, you will be able to build the remaining tables. So, go create the t_bands table:

1. In Enterprise Manager, right-click on the Music database node and select New > Table.
2. You see the Choose Name dialog box as shown in Figure 4-4. Enter **t_bands** in the "Enter a name for the table:" textbox and click OK. The table design grid is now ready for you to enter column information.
3. In the design grid, enter **band_id** in the Column Name field of the first row as shown in Figure 4-5. In the Datatype column, select int to signify that the

4. band_id field will contain integer type data. On the same row, deselect the Allow Nulls checkbox and select the Identity checkbox. Click the Set Primary Key button
(it looks like a key) on the SQL Server toolbar to make the band_id column the primary key for the t_bands table.
5. Create the **band_title**, **music_type_id**, and **record_company_id** columns, using Figure 4-5 as a guide.
6. Right-click the t_bands table design grid as shown in Figure 4-5. You see the Properties dialog box.
7. Select the Indexes/Keys tab and click the New button to create a new index on the band_title column.
8. Select band_title from the Column name drop-down box and enter **IX_band_title** in the Index name text box.
9. Select the Create UNIQUE checkbox and the Index option button, and click the Close button (as shown in Figure 4-6).
10. Save and close the t_bands design grid .

5.6. The ADO.NET Object Model

The figure below shows a simplified view of the primary objects in the ADO.NET object model. Of course, the reality of the class library is more complicated, but we'll deal with the intricacies later. For now, it's enough to understand what the primary objects are and how they typically interact.

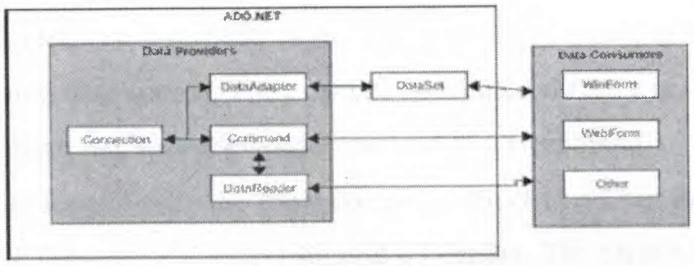


Figure 5.1. Ado.Net

The ADO.NET classes are divided into two components: the Data Providers (sometimes called Managed Providers), which handle communication with a physical data store, and the DataSet, which represents the actual data. Either component can communicate with data consumers such as WebForms and WinForms.

5.6.1 Data Providers

The Data Provider components are specific to a data source. The .NET Framework includes two Data Providers: a generic provider that can communicate with any OLE DB data source, and a SQL Server provider that has been optimized for Microsoft SQL Server versions 7.0 and later. Data Providers for other databases such as Oracle and DB2 are expected to become available, or you can write your own.

The two Data Providers included in the .NET Framework contain the same objects, although their names and some of their properties and methods are different. To illustrate, the SQL Server provider objects begin with SQL (for example, SQLConnection), while the OLE DB objects begin with OleDB (for example, OleDbConnection).

The Connection object represents the physical connection to a data source. Its properties determine the data provider (in the case of the OLE DB Data Provider), the data source and database to which it will connect, and the string to be used during connecting. Its methods are fairly simple: You can open and close the connection, change the database, and manage transactions.

The Command object represents a SQL statement or stored procedure to be executed at the data source. Command objects can be created and executed independently against a Connection object, and they are used by DataAdapter objects to handle communications from a DataSet back to a data source.

Command objects can support SQL statements and stored procedures that return single values, one or more sets of rows, or no values at all.

A DataReader is a fast, low-overhead object for obtaining a forward-only, read-only stream of data from a data source. They cannot be created directly in code; they are created only by calling the *ExecuteReader* method of a Command.

The DataAdapter is functionally the most complex object in a Data Provider. It provides the bridge between a Connection and a DataSet. The DataAdapter contains four Command objects: the SelectCommand, UpdateCommand, InsertCommand, and DeleteCommand. The DataAdapter uses the SelectCommand to fill a DataSet and uses the remaining three commands to transmit changes back to the data source, as required.

Microsoft ActiveX Data Objects (ADO)	In functional terms, the Connection and Command objects are roughly equivalent to their ADO counterparts (the major difference being the lack of support for server-side cursors), while the DataReader functions like a firehose cursor. The DataAdapter and DataSet have no real equivalent in ADO.

5.6.2. DataSets

The DataSet is a memory-resident representation of data. Its structure is shown in the figure below. The DataSet can be considered a somewhat simplified relational database, consisting of tables and their relations. It's important to understand, however, that the DataSet is always disconnected from the data source—it doesn't "know" where the data it contains came from, and in fact, it can contain data from multiple sources.

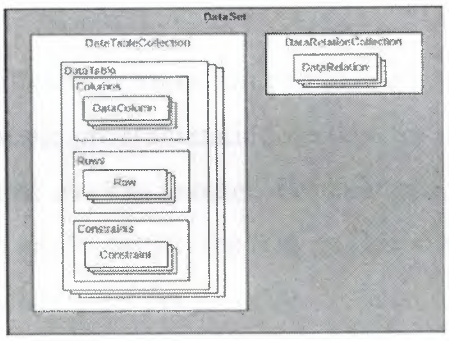


Figure 5.2. DataSet

The DataSet is composed of two primary objects: the DataTableCollection and the DataRelationCollection. The DataTableCollection contains zero or more DataTable objects, which are in turn made up of three collections: Columns, Rows, and Constraints. The DataRelationCollection contains zero or more DataRelations.
The DataTable's Columns collection defines the columns that compose the DataTable. In addition to ColumnName and DataType properties, a DataColumn's properties allow you to define such things as whether or not it allows nulls (AllowDBNull), its maximum length (MaxLength), and even an expression that is used to calculate its value(Expression).
The DataTable's Rows collection, which may be empty, contains the actual data as defined by the Columns collection. For each Row, the DataTable maintains its

original, current, and proposed values. As we'll see, this ability greatly simplifies certain kinds of programming tasks.

ADO The ADO.NET DataTable provides essentially the same functionality as the ADO Recordset object, although it obviously plays a very different role in the object model.

The DataTable's Constraints collection contains zero or more Constraints. Just as in a relational database, Constraints are used to maintain the integrity of the data. ADO.NET supports two types of constraints: ForeignKeyConstraints, which maintain relational integrity (that is, they ensure that a child row cannot be orphaned), and UniqueConstraints, which maintain data integrity (that is, they ensure that duplicate rows cannot be added to the table). In addition, the PrimaryKey property of the DataTable ensures entity integrity (that is, it enforces the uniqueness of each row).

Finally, the DataSet's DataRelationCollection contains zero or more DataRelations. DataRelations provide a simple programmatic interface for navigating from a master row in one table to the related rows in another. For example, given an Order, a DataRelation allows you to easily extract the related OrderDetails rows. (Note, however, that the DataRelation itself doesn't enforce relational integrity. A Constraint is used for that.)

5.6.3 Binding Data to a Simple Windows Form

The process of connecting data to a form is called *data binding.* Data binding can be performed in code, but the Microsoft Visual Studio .NET designers make the process very simple. In this chapter, we'll use the designers and the wizards to quickly create a simple data bound Windows form.

Important If you have not yet installed this book's practice files, work through "Installing and Using the Practice Files" in the Introduction, and then return to this chapter.

5.6.3.1 Adding a Connection and Data Adapter to a Form Roadmap

The first step in binding data is to create the Data Provider objects. Visual Studio provides a Data Adapter Configuration Wizard to make this process simple. Once the Data Adapter has been added, you can check that its configuration is correct by using the Data Adapter Preview window within Visual Studio.

5.6.3.1.1. Add a Connection to a Windows Form

1. Open the Employees Form project from the Visual Studio Start Page.
2. Double-click Employees.vb (or Employees.cs if you're using C#) in the Solution Explorer to open the form.

Visual Studio displays the form in the form designer.

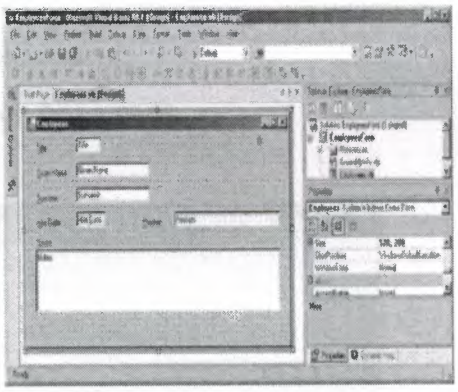
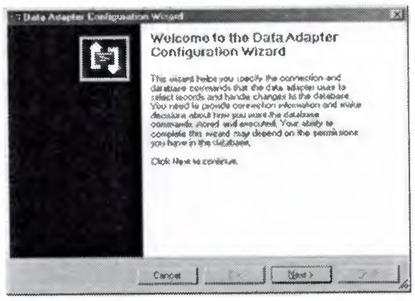


Figure 5.3. Visual Studio.Net

3. Drag a SQLDataAdapter onto the form from the Data tab of the Toolbox.

Visual Studio displays the first page of the DataAdapter Configuration Wizard.



Data Adapter Configuration Wizard

Welcome to the DataAdapter Configuration Wizard

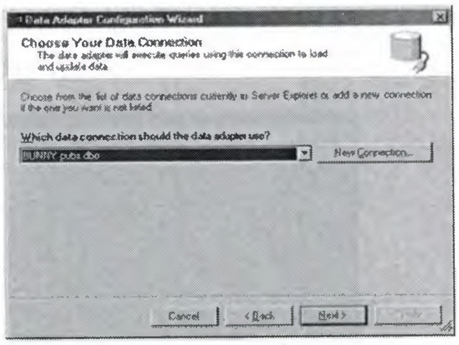
This wizard helps you specify the connection and database commands that the data adapter uses to select records and handle changes to the database. You need to provide connection information and make decisions about how you want the database commands stored and executed. Your ability to complete this wizard may depend on the permissions you have in the database.

Click Next to continue.

Figure 5.4. Data Adapter ConfigurationWizard

4. Click Next.

The DataAdapter Configuration Wizard displays a page asking you to choose a connection.



Data Adapter Configuration Wizard

Choose Your Data Connection
The data adapter will execute queries using this connection to load and update data.

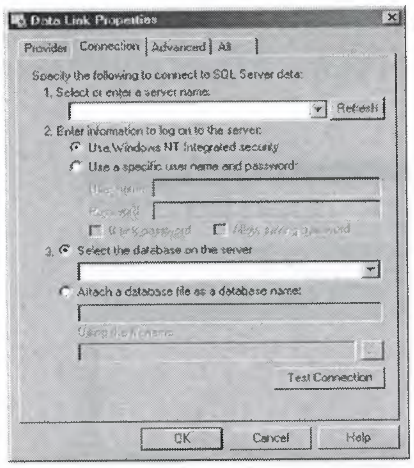
Choose from the list of data connections currently in Server Explorer or add a new connection if the one you want is not listed.

Which data connection should the data adapter use?
BUNNY.pubs.dbo

Figure 5.5. Data Adapter ConfigurationWizard

5. Click New Connection.

The Data Link Properties dialog box opens.



Data Link Properties

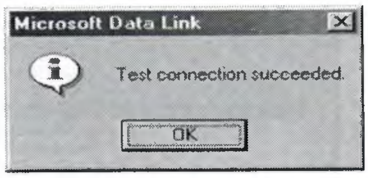
Provider | Connection | Advanced | All

Specify the following to connect to SQL Server data:
1. Select or enter a server name:
2. Enter information to log on to the server:
(●) Use Windows NT Integrated security
( ) Use a specific user name and password:
User name:
Password:
☐ Blank password ☐ Allow saving password
3. (●) Select the database on the server:
( ) Attach a database file as a database name:
Using the filename:

Figure 5.6. Data Link Properties Dialog Box

6. Specify the name of your server, the appropriate logon information, select the Northwind database, and then click Test Connection.

The DataAdapter Configuration Wizard displays a message indicating that the connection was successful.

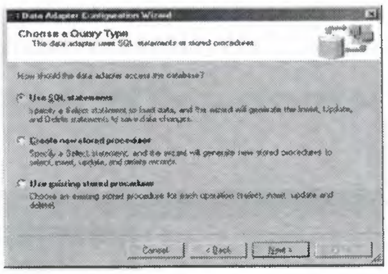

Microsoft Data Link

Test connection succeeded.

Figure5.7. Message Box

7. Click OK to close the message, click OK to close the Data Link Properties dialog box, and then click Next to display the next page of the DataAdapter Configuration Wizard.

The DataAdapter Configuration Wizard displays a page requesting that you choose a query type.


Data Adapter Configuration Wizard

Choose a Query Type
The data adapter uses SQL statements or stored procedures.

How should the data adapter access the database?

(●) Use SQL statements
Specify a Select statement to load data, and the wizard will generate the Insert, Update, and Delete statements to save data changes.

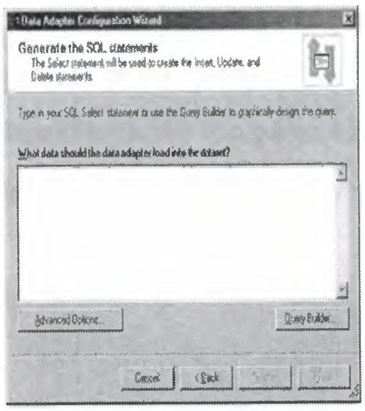
( ) Create new stored procedures
Specify a Select statement, and the wizard will generate new stored procedures to select, insert, update, and delete records.

( ) Use existing stored procedures
Choose an existing stored procedure for each operation (select, insert, update and delete).

Figure 5.8.Choose A Query Type

8. Verify that the Use SQL statements option is selected, and then click Next ,

The DataAdapter Configuration Wizard displays a page requesting the SQL statement(s) to be used.


Data Adapter Configuration Wizard

Generate the SQL statements
The Select statement will be used to create the Insert, Update, and Delete statements.

Type in your SQL Select statement or use the Query Builder to graphically design the query.

What data should the data adapter load into the dataset?

Figure5.9

9. Click Query Builder.

The DataAdapter Configuration Wizard opens the Query Builder and displays the Add Table dialog box.

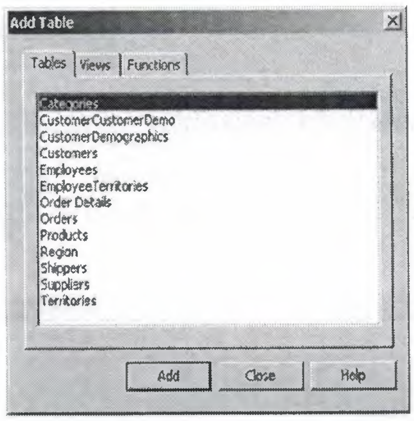


Figure 5.10. Add Table

10. Select the Employees table, click Add, and then click Close.

The Add Table dialog box closes, and the Employees table is added to the Query Builder,

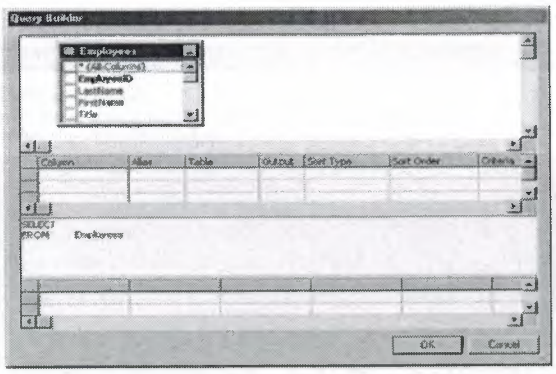
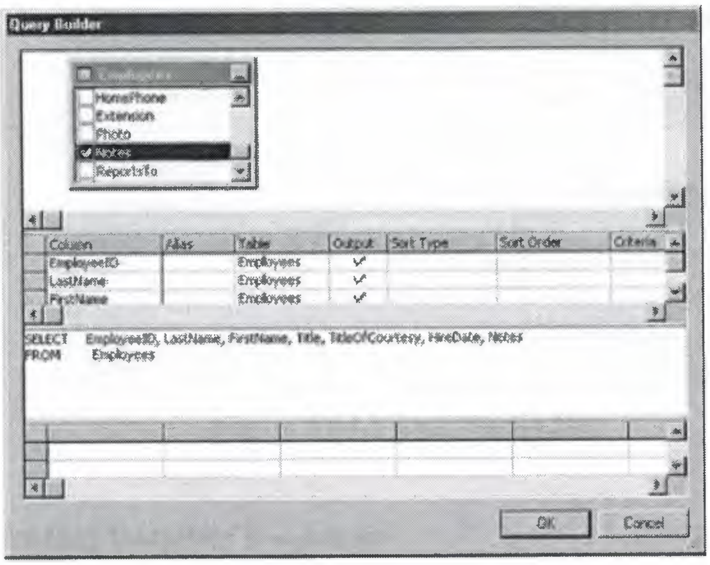


Figure 5.11.Query Builder

11. Add the following fields to the query by selecting the check box next to the field name in the top pane: *EmployeeID, LastName, FirstName, Title, TitleOfCourtesy, HireDate, Notes.*

The Query Builder creates the SQL command.



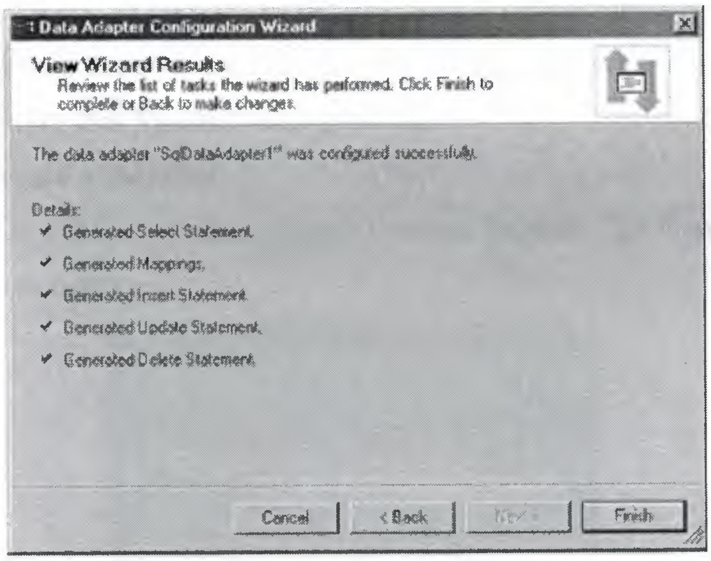
Column	Alias	Table	Output	Sort Type	Sort Order	Criteria
EmployeeID		Employees	✓			
LastName		Employees	✓			
FirstName		Employees	✓			

```
SELECT    EmployeeID, LastName, FirstName, Title, TitleOfCourtesy, HireDate, Notes
FROM      Employees
```

Figure 5.12.

12. Click OK to close the Query Builder, and then click Next.

The DataAdapter Configuration Wizard displays a page showing the results of adding the Connection and DataAdapter objects to the form.



View Wizard Results
Review the list of tasks the wizard has performed. Click Finish to complete or Back to make changes.

The data adapter "SqlDataAdapter1" was configured successfully.

Details:
✓ Generated Select Statement.
✓ Generated Mappings.
✓ Generated Insert Statement.
✓ Generated Update Statement.
✓ Generated Delete Statement.

Figure 5.13 Data Adapter Configuration Wizard

13. Click Finish to close the DataAdapter Configuration Wizard. The DataAdapter Configuration Wizard creates and configures a SQLDataAdapter and a SQLConnection, and then adds them to the Component Designer.

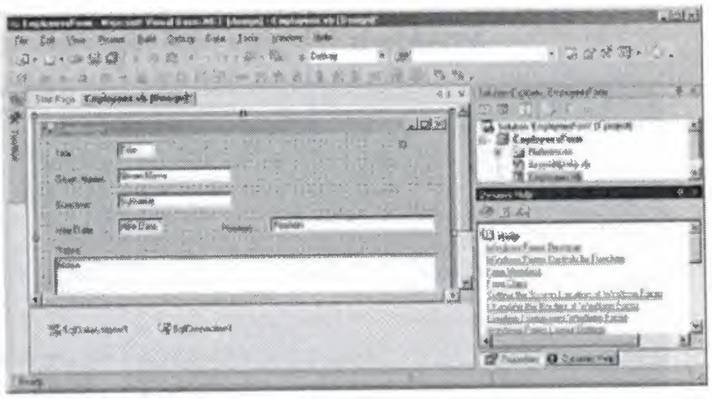


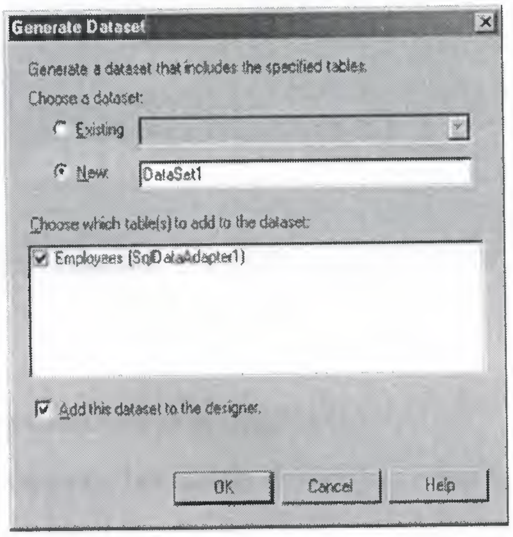
Figure 5.14.Visual Studio.Net

5.6.3.2. Creating DataSets Roadmap

The Connection and DataAdapter objects handle the physical communication with the data store, but you must also create a memory -resident representation of the actual data that will be bound to the form. You can bind a control to almost any structure that contains data, including arrays and collections, but you'll typically use a DataSet. As with the Data Provider objects, Visual Studio provides a mechanism for automating this process. In fact, it can be done with a simple menu choice, although because Visual Studio exposes the code it creates, you can further modify the basic DataSet functionality that Visual Studio provides.

5.6.3.2.1. Create a DataSet

1. On the Data menu, choose Generate Dataset. The Generate Dataset dialog box opens.



Generate Dataset

Generate a dataset that includes the specified tables.

Choose a dataset:

( ) Existing

(•) New: DataSet1

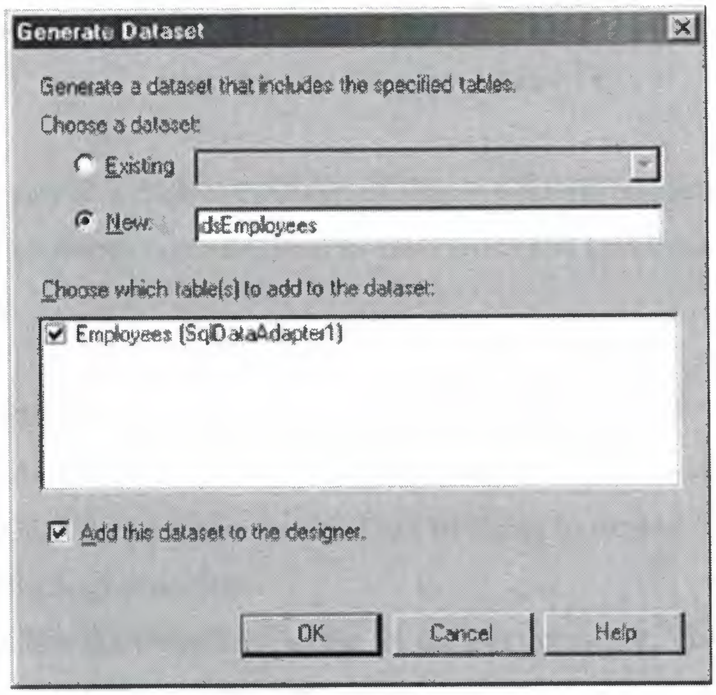
Choose which table(s) to add to the dataset:

☑ Employees (SqlDataAdapter1)

☑ Add this dataset to the designer.

Figure5.15.Generate Dataset

2. In the New text box, type **dsEmployees**



Generate Dataset

Generate a dataset that includes the specified tables.

Choose a dataset:

( ) Existing

(●) New: dsEmployees

Choose which table(s) to add to the dataset:

☑ Employees (SqlDataAdapter1)

☑ Add this dataset to the designer.

Figure5.16.

3. Click OK.

Visual Studio creates the DataSet class and adds an instance of it to the bottom pane of the forms designer.

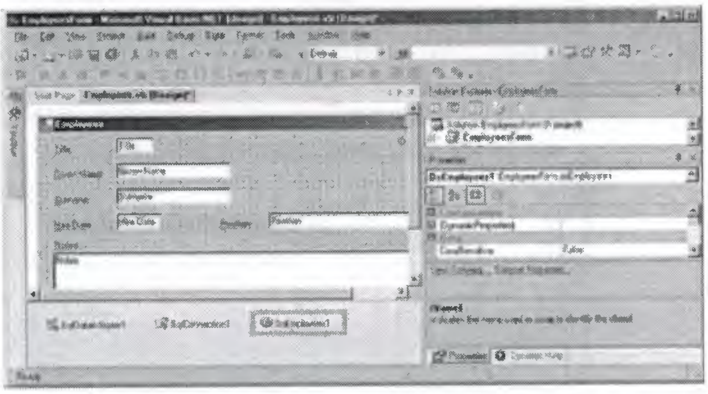


Figure5.17.

5.6.4. Simple Binding Controls to a Data Set

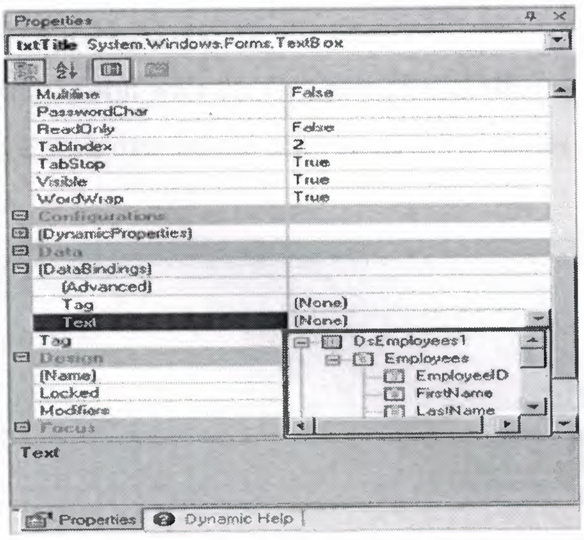
The .NET Framework supports two kinds of binding: simple and complex. *Simple binding* occurs when a single data element, such as a date, is bound to a control. *Complex binding* occurs when a control is bound to multiple data values, for example, binding alist box to a DataSet that contains a list of Order Numbers.

Roadmap We'll examine simple and complex data binding in more detail in Chapters 10 and 11.

Almost any property of a control can support simple binding, but only a subset of Windows and Web Forms controls (such as Data Grids and List Boxes) can support complex binding.

5.6.4.1. Bind the Text Property of a Control to a Data Set

1. Click the txt Title text box in the forms designer to select it.
2. Click the plus sign next to Data Bindings to expand the Data Bindings properties.
3. Click the drop-down arrow for the Text property. Visual Studio displays a list of available data sources.
4. In the list of avail able data sources for the Text property, click the plus sign next to the DsEmployees1 data source, and then click the plus sign next to the Employees Data Table.


Properties

txtTitle System.Windows.Forms.TextBox

Multiline	False
PasswordChar	
ReadOnly	False
TabIndex	2
TabStop	True
Visible	True
WordWrap	True
Configurations	
(DynamicProperties)	
Data	
(DataBindings)	
(Advanced)	
Tag	(None)
Text	(None)
Tag	
Design	
(Name)	
Locked	
Modifiers	
Focus	

DsEmployees1
Employees
EmployeeID
FirstName
LastName

Text

Figure 5.18.Properties

5. Click the Title Of Courtesy column to select it.
6. Repeat steps 1 through 5 to bind the Text property of the remaining controls to the columns of the Employees Data Table.

5.6.5. Loading Data into the Data Set

We now have all the components in place for manipulating the data from our data source, but we have one task remaining: We must actually load the data into the DataSet.

If you're used to working with data bound forms in environments such as Microsoft Access, it may come as a surprise that you need to do this manually. Remember, however, that the ADO.NET architecture has been designed to operate without a permanent connection to the database. In a disconnected environment, it's appropriate, and indeed necessary, that the management of the connection be under programmatic control.

5.6.5.1. Visual Basic .NET

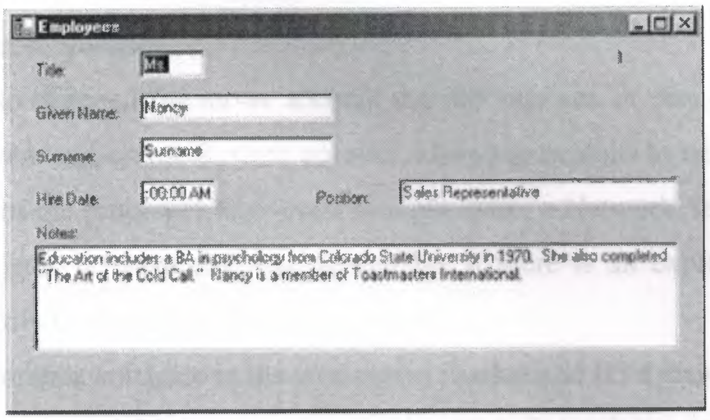
1. Press F7 to view the code for the form.
2. Expand the region labeled "Windows Form Designer generated code" and navigate to the New Sub.
3. Add the following line of code just before the end of the procedure:

SqlDataAdapter1.Fill(DsEmployees1).

4. Press F5 to build and run the program.

Visual Studio displays the form with the first row displayed.

5. Admire your data bound form for a few minutes and then close the form.



Employees

Title: Ms

Given Name: Nancy

Surname: Surname

Hire Date: 00:00 AM

Position: Sales Representative

Notes: Education includes a BA in psychology from Colorado State University in 1970. She also completed "The Art of the Cold Call." Nancy is a member of Toastmasters International.

Figure 5.19.

CHAPTER SIX

Internet Information Server

6.1. What is IIS?

IIS (Microsoft Internet Information Services or Server) is a set of Internet based services for Windows machines. Originally supplied as part of the Option Pack for Windows NT, they were subsequently integrated with Windows 2000 and Windows Server 2003. The current (Windows 2003) version is IIS 6.0 and includes servers for FTP, SMTP, NNTP and HTTP/HTTPS. Earlier versions also included a Gopher server.

The web server itself can not directly perform server side processing but can delegate the task to ISAPI applications on the server. Microsoft provides a number of these, including ones for Active Server Pages and ASP.NET. Third parties have provided support for PHP and Perl languages in the same way.

Internet Information Services is designed to run on Windows server operating systems. A restricted version that supports one web site and a limited number of connections is also supplied with Windows XP Professional.

IIS has been attributed with a number of security exploits, most of which were in fact issues within the lesser used ISAPI handlers. With Windows Server 2003 Microsoft finally elected to turn off all ISAPI handlers by default thereby giving the web server a much more secure "out of the box" configuration.

Microsoft has also changed the server account that IIS runs on. In versions of IIS before 6.0, all the features were run on the System account, allowing exploits to run wild on the system. Under 6.0 many of the processes have been brought under a Network Services account which has fewer privileges. In particular this means that if there is an exploit on that feature, it wouldn't necessarily compromise the entire system.

Apache is the dominant software in the web server market and IIS's main competitor. Solaris Operating Environment/J2EE also competes in the enterprise web services arena.

6.2. Installing ASP.NET (IIS 6.0)

ASP.NET is supported on the Windows Server™ 2003 family, Windows 2000 (Professional, Server, and Advanced Server), and Windows XP Professional for both client and server applications.

A server running a member of the Microsoft Windows Server 2003 family can be configured as an application server, with ASP.NET as an option that you can enable while configuring the application server role. To deploy ASP.NET Web applications to a production server, you must be sure to enable the ASP.NET and IIS roles on the production server before you distribute the application.

If you want to install ASP.NET on a domain controller, there are special steps you must take to make the installation work correctly

ASP.NET, along with the .NET Framework version 1.1, is installed as a part of Windows Server 2003. You simply need to add it as a new program from Control Panel or enable it by using the Configure Your Server wizard.

To install ASP.NET on a server running Windows Server 2003 using the Configure Your Server wizard

1. From the Start menu, click Manage Your Server; in the Manage Your Server window, click Add or remove a role.
2. In the Configure Your Server Wizard, click Next, and in the Server Role dialog box, check Application Server (IIS, ASP.NET) and then click Next.
3. In the Application Server Options dialog box, select the Enable ASP.NET check box, click Next, and then click Next again.
4. If necessary, insert your Windows Server 2003 installation CD in the CD-ROM drive and then click Next.
5. When the installation is complete, click Finish.

To install ASP.NET on a server running Windows Server 2003 using the Add or Remove Programs dialog box

1. From the Start menu, point to Control Panel, and then click Add or Remove Programs.

2. In the Add or Remove Programs dialog box, click Add/Remove Windows Components.
3. In the Add or Remove Programs dialog box, click Add/Remove Windows Components.
4. When the Windows Components wizard has finished configuring Windows Server 2003, click Finish

To enable ASP.NET in IIS Manager on a server running Windows Server 2003

1. From the Start menu, click Run.
2. In the Open box in the Run dialog box, type inetmgr, and then click OK.
3. In IIS Manager, expand the local computer and then click Web Service Extensions.
   In the details pane, right-click ASP.NET and then click Allow. The status of ASP.NET changes to Allowed.

CHAPTER SEVEN

ASP.NET Security

7.1. The .NET Security Model

The .NET Security model is a complex, multi-layered, and highly configurable system. Since part of the purpose of the .NET framework is to allow for "mobile code" to be distributed to users on multiple platforms, security has become a major concern. Mobile code is an application or piece of software that is transmitted from a server to a local system (or other device) to be executed locally. This section provides an overview of several aspects of the .NET security model. The focus for this section is on the configuration of .NET, specifically the configuration options for
securing deployment of Web applications. Configurations of the framework itself, such as defining policies and code groups, as well as security features inherent in the CLR are addressed later in this paper.

7.2. Web Application Security

There are several important features available for configuration for ASP.NET applications. Permissions for Web applications are regulated by the .NET Framework. Each Web application is comprised of a number of assemblies, all of which may have different security permissions granted to them by the CLR. The way these permissions are established is determined by the configuration of the .NET Framework on the server that is executing these applications.When building Web applications, configuration options may provide additional security beyond default settings. These options provide authentication mechanisms and authorization control to allow for granular support over who may use these applications and what features of the application are available to that user. System-wide policies may also be put in place to enforce rules that will apply to all installed Web applications.

These options are configured by changing the configuration parameters in the Web applications' configuration file, which is located in the directory where the application is installed. This file, written in XML and named "web.config," defines a set of policies that will be applied to the application. The worker process will also recursively examine parent directories for the existence of web.config files. Lists of permissions will be built from policies supplied in each web.config file encountered in the order that they are read (from the root directory of the application up to the root directory of the Web server).

Finally, a global configuration file, machine.config, located in the %SYSTEMROOT%\Microsoft.NET\(version)\CONFIG directory, defines policies that will be applied to all installed Web applications. Since this policy is read last, all previously read web.config files will install policies with a higher precedence than those contained in the global configuration file.

7.3. .NET Framework Security Mechanisms Role-Based Security

Role-Based security refers to the security context under which an assembly is run. Assemblies have associated privilege levels that define what resources may be accessed at execution time. The framework establishes the privileges granted to a particular assembly when *authentication* takes place.

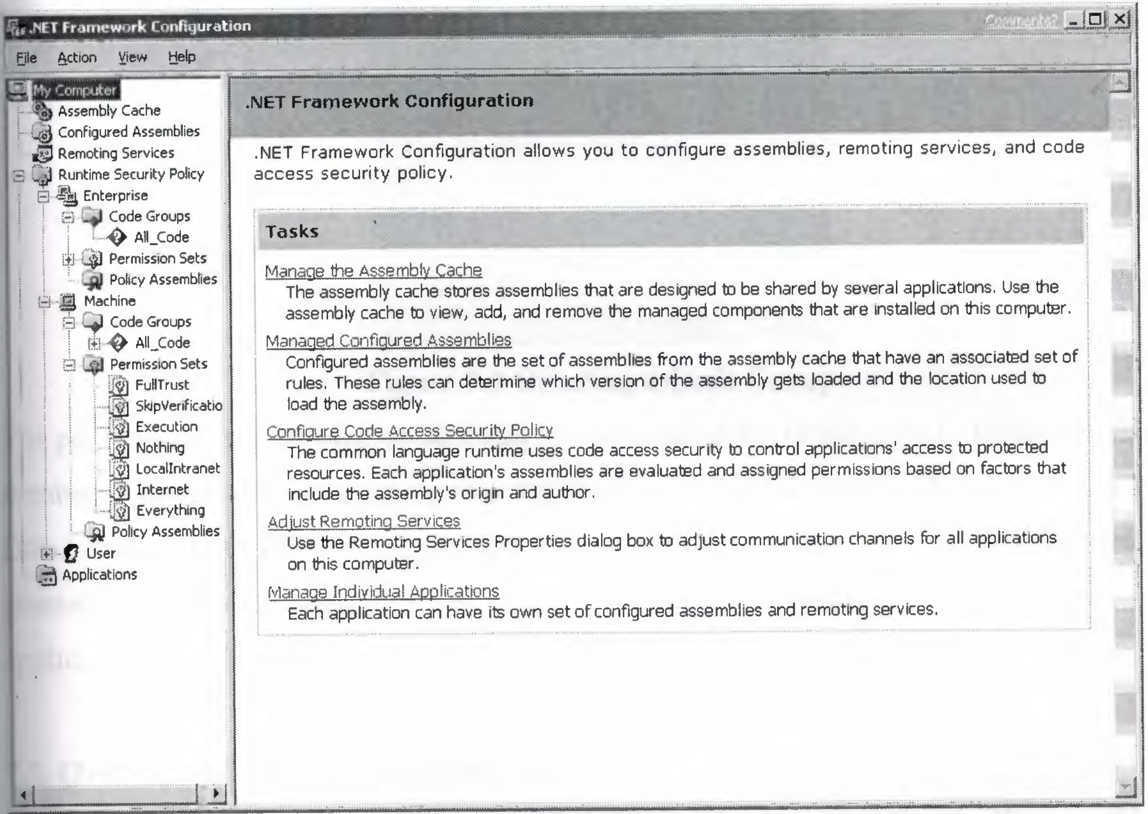
The application programmer defines rules related to roles in an XML configuration file, such as specifing policies which govern privileges the application has when it is invoked, as well as who is permitted to use it. This is achieved by specifying rules for both *authentication* and *authorization.* Once authenticated and authorized, an Identity is established and associated with the running assemblies principal. Identities associated with a principal may either be accounts and roles on the system, or custom identities that are created by the application programmer.

7.4. Evidence-Based Security

When an assembly is being prepared for execution, the .NET framework must decide what resources the application may access. This is accomplished by evaluating the *evidence* associated with the assembly and allocating rights based on this evidence.

Evidence refers to the collection of information describing various aspects of the origin of the assembly. The purpose of this information is to establish the level of trust that should be granted to the assembly for execution on the local machine. The information present in assemblies depends on what is known about the assembly in question. Information supplied could include:

Valid Digital Signatures – If an assembly is signed and the signature can be verified, digital signatures may be used to ensure that the code is trustworthy if it comes from a reputable source.



.NET Framework Configuration

.NET Framework Configuration allows you to configure assemblies, remoting services, and code access security policy.

Tasks

Manage the Assembly Cache
The assembly cache stores assemblies that are designed to be shared by several applications. Use the assembly cache to view, add, and remove the managed components that are installed on this computer.

Managed Configured Assemblies
Configured assemblies are the set of assemblies from the assembly cache that have an associated set of rules. These rules can determine which version of the assembly gets loaded and the location used to load the assembly.

Configure Code Access Security Policy
The common language runtime uses code access security to control applications' access to protected resources. Each application's assemblies are evaluated and assigned permissions based on factors that include the assembly's origin and author.

Adjust Remoting Services
Use the Remoting Services Properties dialog box to adjust communication channels for all applications on this computer.

Manage Individual Applications
Each application can have its own set of configured assemblies and remoting services.

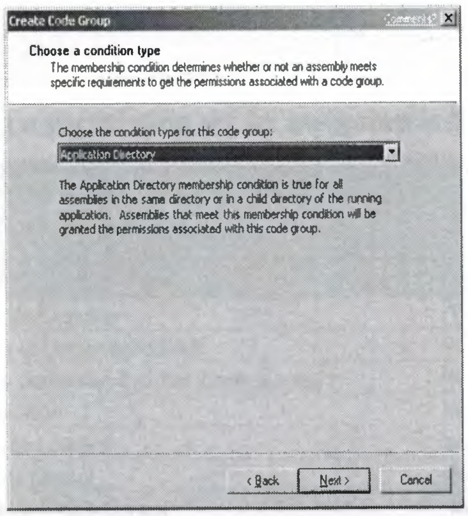
Figure 7.1. .NET Framework Configuration.

URL or Site – The origin of assemblies may also help establish some level of trust if the source is reputable.

Zone – The zone from where the assembly originated can be used to determine whether privileges should be granted to it. Untrusted zones, such as the Internet in a default setup, have a low amount of privileges granted to it.

Path – The path where the assembly is physically located on disk is accessible. This may be useful to lower privileges of assemblies created by particular users or Web applications that will be serving anonymous requests.

Evidence is used to categorize assemblies into *code* groups, which refer to the different sets ofevidence given and associated privileges. Each assembly that is going to be executed is categorized in a code group so that privileges may be determined and applied. After privileges for an assembly have been discovered, the framework checks what permissions the assembly requests to function. If assigned less privileges than the assembly requires, it is not run.


Create Code Group

Choose a condition type

The membership condition determines whether or not an assembly meets specific requirements to get the permissions associated with a code group.

Choose the condition type for this code group: Application Directory

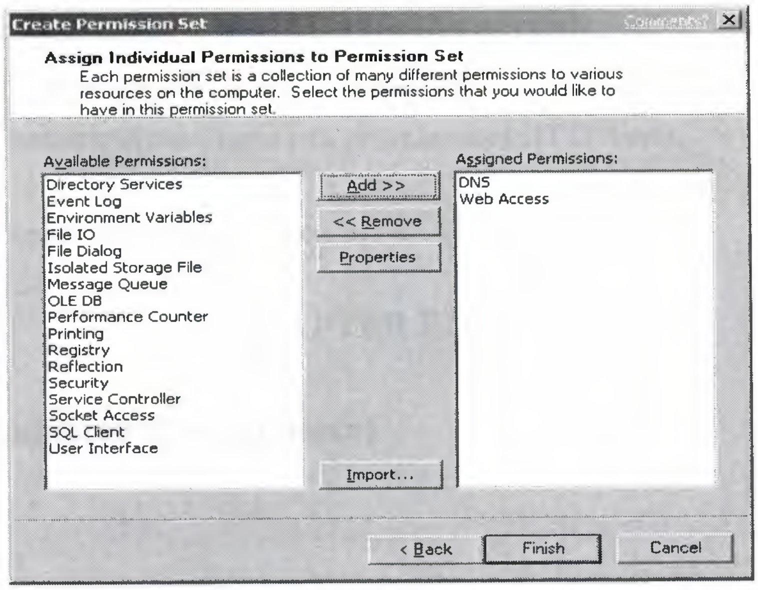
The Application Directory membership condition is true for all assemblies in the same directory or in a child directory of the running application. Assemblies that meet this membership condition will be granted the permissions associated with this code group.

Figure 7.2 Creating a Code Group

The permissions refer to what objects may be accessed on the target system. There are a large number of objects that an assembly may require to perform its function. These objects include DataAccess, FileIO, DNS, EventLog, and Environment. By default, very low privilege sets are granted to assemblies originating from unknown authors on the Internet, so malicious applications cannot make use of the file system or tamper with the Registry.

7.5. Overview of ASP.NET Security (IIS 6.0)

Most Web sites need to selectively restrict access to some portions of the site. You can think of a Web site as somewhat analogous to an art gallery. The gallery is open for the public to come in and browse, but there are certain parts of the facility, such as the business offices, that are accessible only to people with certain credentials, such as employees. When a Web site stores its customers' credit card information in a database, for example, ASP.NET helps protect the database from public access. ASP.NET security addresses this and many other security issues.

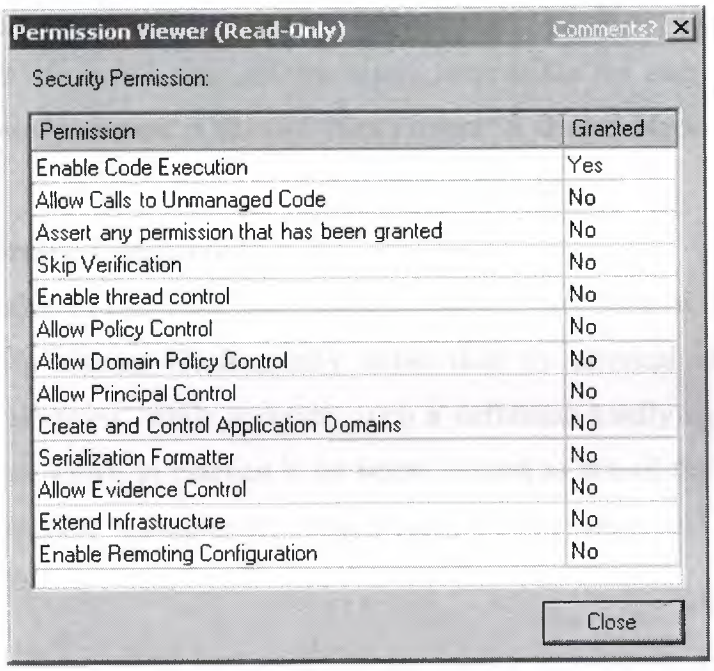


Assign Individual Permissions to Permission Set
Each permission set is a collection of many different permissions to various resources on the computer. Select the permissions that you would like to have in this permission set.

Available Permissions:
Directory Services
Event Log
Environment Variables
File IO
File Dialog
Isolated Storage File
Message Queue
OLE DB
Performance Counter
Printing
Registry
Reflection
Security
Service Controller
Socket Access
SQL Client
User Interface

Assigned Permissions:
DNS
Web Access

Figure 7.3. Creating a permission set by assigning individual permissions.



Security Permission:

Permission	Granted
Enable Code Execution	Yes
Allow Calls to Unmanaged Code	No
Assert any permission that has been granted	No
Skip Verification	No
Enable thread control	No
Allow Policy Control	No
Allow Domain Policy Control	No
Allow Principal Control	No
Create and Control Application Domains	No
Serialization Formatter	No
Allow Evidence Control	No
Extend Infrastructure	No
Enable Remoting Configuration	No

Figure 7.4. Permission levels for a specific permission set

ASP.NET, in conjunction with Internet Information Services (IIS), can authenticate user credentials such as names and passwords using any of the following authentication methods:

- Windows: Basic, digest, or Integrated Windows Authentication (NTLM or Kerberos).
- Microsoft Passport authentication
- Forms authentication
- Client Certificate authentication

ASP.NET helps control access to site information by comparing authenticated credentials, or representations of them, to NTFS file system permissions or to an XML file that lists authorized users, authorized roles (groups), or authorized HTTP verbs.

The topics in this section describe the specifics of ASP.NET security.

CHAPTER EIGHT

Electronic Commerce (E-Commerce)

8.1 Introduction

This paper provides a brief introduction to electronic commerce. It discuss the nature of electronic commerce, consider its scope and impact, and outlines several examples. It then identifies a number of open issues and the actors responsible for addressing those issues. Finally, it gives a brief overview of the G-7 Pilot Project "A Global Marketplace for SMEs".

8.2 What is Electronic Commerce?

One possible definition of electronic commerce would be: "any form of business transaction in which the parties interact electronically rather than by physical exchanges or direct physical contact". However, while accurate, such a definition hardly captures the spirit of electronic commerce, which in practice is far better viewed as one of those rare cases where changing needs and new technologies come together to revolutionise the way in which business is conducted.

Modern business is charecterised by ever-increasing supply capabilities, ever-increasing global competition, and ever-increasing customer exeptions. In response, businesses throughout the world are changing both their organisations and their operations. They are flattening old hierarchical structures and eradicating the barriers between company divisions. They are lowering the barriers between the company and its customers and suppliers. Business processes are being re-designed so that they cross these old boundaries. We now see many examples of processes that span the entire company and even processes that are jointly owned and operated by the company and its customers or suppliers.

Electronic Commerce is a means of enabling and supporting such change on a global scale. It enables companies to be more efficient and flexible in their internal operations, to work more closely with their suppliers, and to be more responsive to the needs and exeptions of their customers. It allows companies to select the best suppliers regardless of their geographical location and to sell to a global market.

One special case of electronic commerce is electronic trading, in which a supplier provides good or services to a customer in return for payment. A special case of electronic trading is electronic retailing, where the customer is an ordinary consumer rather than another company. However, while these special cases are of considerable economic importance, they are just particular examples of the more general case of any form of business operation or transaction conducted via electronic media. Other equally valid examples include internal transactions within a single company or provision of information to an external organisation without charge.

Electronic Commerce is technology for change. Companies that choose to regard it only as an "add on" to their existing ways of doing business will gain only limited benefit. The major benefit will accrue to fully exploit the opportunities offered by electronic commerce.

CHAPTER NINE

E-COMMERCE WEB SITE USING ASP.NET

9.1. What is an e-commerce web site?

Web-based catalog sales site based on the ASP.NET Commerce Starter Kit. Users can browse the catalog, add items to their personal shopping cart, and when they've finished shopping, check out and finalize the sale. Casual browsers can view catalog items freely. A search facility allows users to look for items using any word in the description. If they like, users can add product reviews to the description.

To purchase items, users must be registered as authenticated users by providing account information: an email address and password.

Authenticated users can also take advantage of two components accessible remotely over the Web: an "instant order" component that allows them to create orders remotely and a component that allows them to track orders remotely. These two features illustrate Web Services, the ASP.NET facility that allows you to publish components that are accessible using a Web protocol such as SOAP or HTTP.

9.2. Application Architecture

The base UI of the Commerce Web Site is created using ASP.NET web pages (.aspx files). Reusable UI widgets, such as the navigation menus, are implemented as ASP.NET user controls (.ascx files). User controls are also used to create dynamic page content, such as the list of most popular items.

The data for the Commerce Web Site is stored in a SQL Server database, and accessed via stored procedures. The ADO.NET database code to access these stored procedures is then encapsulated in a component layer. The remote-order and order tracking facilities are implemented as ASP.NET web services.

9.3 Application Code

9.3.1. alsobought_ascx

```
<%@ Control Language="VB" %>
<%@ import Namespace="System.Data.SqlClient" %>
<script runat="server">
 Public ProductID As Integer
Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
 ' Obtain list of products that people who "also bought" an item have purchased. Databind to
list control
Dim productCatalogue As ASPNET.StarterKit.Commerce.ProductsDB = New
ASPNET.StarterKit.Commerce.ProductsDB() alsoBoughtList.DataSource =
productCatalogue.GetProductsAlsoPurchased(ProductID) alsoBoughtList.DataBind()
' Hide the list if no items are in it
If alsoBoughtList.Items.Count = 0 Then
 alsoBoughtList.Visible = False
End If
 End Sub
</script>
<table width="95%" cellpadding="0" cellspacing="0" border="0">
  <tr>
<td><asp:Repeater ID="alsoBoughtList" runat="server">
<HeaderTemplate>
```

```
<tr><td class="MostPopularHead">&nbsp;Customers who bought this also bought
</td></tr>
 </HeaderTemplate>
 <ItemTemplate>
 <tr>
  <td bgcolor="#d3d3d3">asp:HyperLink class="MostPopularItemText" NavigateUrl='<%#
"ProductDetails.aspx?ProductID=" & DataBinder.Eval(Container.DataItem,
"ProductID")%>' Text='<%#DataBinder.Eval(Container.DataItem, "ModelName")%>'
runat="server" /><br>
  </td></tr>
 </ItemTemplate>
 <FooterTemplate>
   <tr>
     <td bgcolor="#d3d3d3">
</td>  </tr>
</FooterTemplate>
</asp:Repeater>
</td>
</tr>
</table>
```

9.3.2. header_ascx

```
<%@ Control %>
<table cellspacing="0" cellpadding="0" width="100%" border="0">
<tr>
<td colspan="2" background="images/grid_background.gif" nowrap>
<table cellspacing="0" cellpadding="0" width="100%" border="0">
<tr>
<td colspan="2">
<img src="images/most_secretive_place.gif">
</td>
<td align="right" nowrap>
<table cellpadding="0" cellspacing="0" border="0">
<tr valign="top">
```

```
<td align="center" width="65">
<a href="Login.aspx" class="SiteLinkBold"><img src="images/sign_in.gif" border="0">
 Sign In</a></td>
<td align="center" width="75">
<a href="OrderList.aspx" class="SiteLinkBold"><img src="images/account.gif" border="0">
Account</a></td>
<td align="center" width="55">
<a href="ShoppingCart.aspx" class="SiteLinkBold"><img src="images/cart.gif"border="0">
Cart</a></td>
<td align="center" width="65">
<a href="InstantOrder.asmx" class="SiteLinkBold"><img src="images/services.gif"
border="0">Services</a></td>
<tr>
</table>
</td>
<td width="10">
</td>
</tr>
</table>
</td>
</tr>
<tr>
<td colspan="2" nowrap><form method="post" action="SearchResults.aspx" id="frmSearch"
name="frmSearch">
<table cellspacing="0" cellpadding="0" width="100%" border="0">
<tr bgcolor="#9D0000"><td background="images/modernliving_bkgrd.gif">
<img align="left" src="images/modernliving.gif">
</td>
<td width="94" align="right" bgcolor="#9D0000"><img src="images/search.gif">
</td><td width="120" align="right" bgcolor="#9D0000">
<input type="text" name="txtSearch" ID="txtSearch" SIZE="20">
</td><td align="left" bgcolor="#9D0000">
```

```
&nbsp;<input type="image" src="images/arrowbutton.gif" border="0" id="image1"
name="image1">&nbsp;</td></tr></table>
</form></td></tr></table>
```

9.3.3 Menu.ascx

```
<%@ Control Language="VB" %>
<%@ import Namespace="System.Data.SqlClient" %>
<%@ outputcache duration="3600" varybyparam="selection" %>
<script runat="server">
 Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
' Set the curent selection of list
Dim selectionId As String = Request.Params("selection")
If Not selectionId Is Nothing Then
MyList.SelectedIndex = CInt(selectionId)
End If
' Obtain list of menu categories and databind to list control
Dim products As ASPNET.StarterKit.Commerce.ProductsDB = New
ASPNET.StarterKit.Commerce.ProductsDB()
MyList.DataSource = products.GetProductCategories()
MyList.DataBind()
End Sub
</script>
<table cellspacing="0" cellpadding="0" width="145" border="0">
 <tr valign="top"><td colspan="2">
<a href="default.aspx"><img src="images/logo.gif" border="0"></a></td>
</tr><tr valign="top"><td colspan="2">
<asp:DataList id="MyList" runat="server" cellpadding="3" cellspacing="0" width="145"
SelectedItemStyle-BackColor="dimgray" EnableViewState="false">
<ItemTemplate><asp:HyperLink class="MenuUnselected" id="HyperLink1" Text='<%#
DataBinder.Eval(Container.DataItem, "CategoryName") %>' NavigateUrl='<%#
"productslist.aspx?CategoryID=" & DataBinder.Eval(Container.DataItem, "CategoryID") &
"&selection=" & Container.ItemIndex %>' runat="server" />  </ItemTemplate>
```

```
<SelectedItemTemplate><asp:HyperLink class="MenuSelected" id="HyperLink2"
Text='<%# DataBinder.Eval(Container.DataItem, "CategoryName") %>' NavigateUrl='<%#
"productslist.aspx?CategoryID=" & DataBinder.Eval(Container.DataItem, "CategoryID") &
"&selection=" & Container.ItemIndex %>' runat="server" /></SelectedItemTemplate>
</asp:DataList></td></tr><tr>  <td width="10"></td><td>
<br><br><br><br><br><br>
<a href="docs/docs.htm" target="_blank" class="SiteLink">Commerce Starter
Kit<br>Documentation</a></td></tr></table>
```

9.3.4 PopularItems.ascx

```
<%@ Control Language="VB" %>
<%@ import Namespace="System.Data.SqlClient" %>
<%@ outputcache duration="3600" varybyparam="None" %>
<%@ import Namespace="ASPNET.StarterKit.Commerce" %>
<script runat="server">
Public ProductID As Integer
Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
' Obtain list of favorite items
Dim products As ProductsDB = New ProductsDB()
' Databind and display the list of favorite product items  productList.DataSource =
products.GetMostPopularProductsOfWeek()
productList.DataBind()
' Hide the list if no items are in it
If productList.Items.Count = 0 Then
productList.Visible = False
End If
End Sub
</script>
<table width="95%" cellpadding="0" cellspacing="0" border="0">
<asp:Repeater ID="productList" runat="server">
<HeaderTemplate><tr><td class="MostPopularHead">&nbsp;Our most popular items this
week</td> </tr>

</HeaderTemplate>
```

```
<ItemTemplate><tr>
<td bgcolor="#d3d3d3">&nbsp;<asp:HyperLink class="MostPopularItemText"
NavigateUrl='<%# "ProductDetails.aspx?ProductID=" &
DataBinder.Eval(Container.DataItem, "ProductID")%>'
Text='<%#DataBinder.Eval(Container.DataItem, "ModelName")%>' runat="server"
/><br></td></tr>
</ItemTemplate><FooterTemplate><tr>  <td bgcolor="#d3d3d3">
</td></tr></FooterTemplate></asp:Repeater></table>
```

9.3.5. ReviewList.ascx

```
<%@ Control Language="VB" %>
<%@ import Namespace="System.Data.SqlClient" %>
<script runat="server">
Public ProductID As Integer
Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
 ' Obtain and databind a list of all reviews of a product
Dim productReviews As ASPNET.StarterKit.Commerce.ReviewsDB = New
ASPNET.StarterKit.Commerce.ReviewsDB()
MyList.DataSource = productReviews.GetReviews(ProductID)
MyList.DataBind()
' Update navigation link for users to add a new review
AddReview.NavigateUrl = "ReviewAdd.aspx?productID=" + ProductID.ToString()
End Sub
</script>
<br>
<br>
<table cellspacing="0" cellpadding="0" width="100%" border="0"> <tr>
<td class="SubContentHead">&nbsp;Reviews
  <br></td></tr>
   <tr><td></td></tr><tr><td><asp:Hyperlink id="AddReview" runat="server">
<img align="absbottom" src="images/review_this_product.gif"  border="0">
</asp:Hyperlink<br>
        <br>
</td></tr><tr><td>
```

```
<asp:DataList ID="MyList" runat="server" width="500" cellpadding="0"
cellspacing="0"><ItemTemplate>
<asp:Label class="NormalBold" Text='<%#DataBinder.Eval(Container.DataItem,
"CustomerName")%>' runat="server" />
<span class="Normal">says...
</span><imgsrc='images/ReviewRating<%#DataBinder.Eval(Container.DataItem,
"Rating")%>.gif'>
<br>
<asp:Label class="Normal" Text='<%#DataBinder.Eval(Container.DataItem,
"Comments")%>' runat="server" /></ItemTemplate><SeparatorTemplate>
<br></SeparatorTemplate></asp:DataList></td></tr></table>
```

9.3.6 AddCard.ascx

```
<%@ Page Language="VB" %>
<script runat="server">
Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
If Not Request.Params("ProductID") Is Nothing Then
Dim cart As ASPNET.StarterKit.Commerce.ShoppingCartDB = New
ASPNET.StarterKit.Commerce.ShoppingCartDB()
' Obtain current user's shopping cart ID
Dim cartId As String = cart.GetShoppingCartId()
' Add Product Item to Cart
 cart.AddItem(cartId, CInt(Request.Params("ProductID")), 1)
End If
Response.Redirect("ShoppingCart.aspx")
End Sub
</script>
```

9.3.7. CheckOut.aspx

```
<%@ Page Language="VB" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%><%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx"
%>
<%@ import Namespace="System.Data.SqlClient" %>
```

```
<script runat="server">
Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
If Page.IsPostBack = False Then
' Calculate end-user's shopping cart ID
dim cart as ASPNET.StarterKit.Commerce.ShoppingCartDB = New
ASPNET.StarterKit.Commerce.ShoppingCartDB()
Dim cartId As String = cart.GetShoppingCartId()
' Populate datagrid with shopping cart data
MyDataGrid.DataSource = cart.GetItems(cartId)
MyDataGrid.DataBind()
' Update total price label
TotalLbl.Text = String.Format("{0:c}", cart.GetTotal(cartId))
End If
End Sub
Sub SubmitBtn_Click(ByVal sender As Object, ByVal e As ImageClickEventArgs)
Dim cart As ASPNET.StarterKit.Commerce.ShoppingCartDB = New
ASPNET.StarterKit.Commerce.ShoppingCartDB()
' Calculate end-user's shopping cart ID
Dim cartId As String = cart.GetShoppingCartId()
' Calculate end-user's customerID
Dim customerId As String = User.Identity.Name
If (Not cartId Is Nothing) And (Not customerId Is Nothing) Then
' Place the order
Dim ordersDatabase As ASPNET.StarterKit.Commerce.OrdersDB = New
ASPNET.StarterKit.Commerce.OrdersDB()
Dim orderId As Integer = ordersDatabase.PlaceOrder(customerId, cartId)
'Update labels to reflect the fact that the order has taken place
Header.Text = "Check Out Complete!"
Message.Text = "<b>Your Order Number Is: </b>" & orderId
SubmitBtn.Visible = False
End If
End Sub
</script>
<html><head>
```

```
<link href="ASPNETCommerce.css" type="text/css" rel="stylesheet" /></head>
<body bottommargin="0" leftmargin="0" background="images/sitebkgrd.gif" topmargin="0"
rightmargin="0" marginwidth="0" marginheight="0"><table cellspacing="0"
cellpadding="0" width="100%" border="0"><tbody>
<tr><td colspan="2">
<ASPNETCommerce:Header id="Header1" runat="server"></ASPNETCommerce:Header>
</td></tr><tr>
<td valign="top">
<ASPNETCommerce:Menu id="Menu1" runat="server"></ASPNETCommerce:Menu>
<img height="1" src="images/1x1.gif" width="145" /> </td>
<td valign="top" nowrap="nowrap" align="left" width="100%">
<table height="100%" cellspacing="0" cellpadding="0" width="100%" align="left"
border="0"><tbody><tr valign="top"><td nowrap="nowrap">
 <br /><form runat="server">
<img src="images/1x1.gif" width="24" align="left" />
<table cellspacing="0" cellpadding="0" width="100%" border="0"><tbody><tr>
<td class="ContentHead"><img height="32" src="images/1x1.gif" width="60" align="left"
/><asp:Label id="Header" runat="server">Review and Submit Your Order</asp:Label>
<br /></td></tr></tbody></table>
<img height="1" src="images/1x1.gif" width="92" align="left" />
<table height="100%" cellspacing="0" cellpadding="0" width="550" border="0">
<tbody><tr valign="top"><td class="Normal" width="100%"><br />
<asp:Label id="Message" runat="server">Please check all the information below to be sure
it's correct.</asp:Label><br />
<br />
<asp:DataGrid id="MyDataGrid" runat="server" AutoGenerateColumns="false"
AlternatingItemStyle-CssClass="CartListItemAlt" ItemStyle-CssClass="CartListItem"
FooterStyle-CssClass="cartlistfooter" HeaderStyle-CssClass="CartListHead"
ShowFooter="true" Font-Size="8pt" Font-Name="Verdana" cellspacing="0"
cellpadding="4" GridLines="Vertical" BorderColor="black" width="90%">
<Columns><asp:BoundColumn HeaderText="Product Name" DataField="ModelName" />
<asp:BoundColumn HeaderText="Model Number" DataField="ModelNumber" />
<asp:BoundColumn HeaderText="Quantity" DataField="Quantity" />
```

```
<asp:BoundColumn HeaderText="Price" DataField="UnitCost" DataFormatString="{0:c}"
/>
<asp:BoundColumn HeaderText="Subtotal" DataField="ExtendedAmount"
DataFormatString="{0:c}" /></Columns>
</asp:DataGrid><br />
<br /><b>Total: </b><asp:Label id="TotalLbl" runat="server"></asp:Label><p>
<asp:ImageButton id="SubmitBtn" onclick="SubmitBtn_Click" runat="server"
ImageURL="images/submit.gif"></asp:ImageButton></p>
</td> </tr></tbody>  </table></form>
</td></tr></tbody></table></td></tr></tbody></table></body></html>
```

9.3.8.CreditCartForm.aspx

```
<%@ Page %>
<script runat="server">
Sub Button_Click( s As Object, e As EventArgs)
If IsValid Then
Response.Redirect( "CheckOut.aspx" )
End If
End Sub
Sub ValidateCCNumber( s As Object, e As ServerValidateEventArgs )
Dim intCounter As Integer
Dim strCCNumber As String
Dim blnIsEven As Boolean = False
Dim strDigits As String = ""
Dim intCheckSum As Integer = 0
' Strip away everything except numerals
For intCounter = 1 To Len( e.Value )
If IsNumeric( MID( e.Value, intCounter, 1 ) ) THEN
strCCNumber = strCCNumber & MID( e.Value, intCounter, 1 )
End If
Next
' If nothing left, then fail
If Len( strCCNumber ) = 0 Then
e.IsValid = False
```

```
Else
' Double every other digit
for intCounter = Len( strCCNumber ) To 1 Step -1
if blnIsEven Then
strDigits = strDigits & cINT( MID( strCCNumber, intCounter, 1 ) ) * 2
ELSE
strDigits = strDigits & cINT( MID( strCCNumber, intCounter, 1 ) )
END IF
blnIsEven = ( NOT blnIsEven )
Next
' Calculate CheckSum
For intCounter = 1 To Len( strDigits )
intCheckSum = intCheckSum + cINT( MID( strDigits, intCounter, 1 ) )
Next
' Assign results
e.IsValid = (( intCheckSum Mod 10 ) = 0 )
End If
End Sub
Sub DropDownList1_SelectedIndexChanged(sender As Object, e As EventArgs)
End Sub
</script>
<html>
<head>
<title>CustomValidatorLuhn.aspx</title>
</head>
<body>
<form runat="Server">
<p align="center">
<strong><font style="BACKGROUND-COLOR: blue">Credit Card Query
Form</font></strong> </p>
      <p>
        Name Surname:<asp:TextBox id="TextBox1" runat="server"></asp:TextBox>
      </p>
      <p>
```

```
        Valid Date:&nbsp;
        <asp:DropDownList id="DropDownList1" runat="server"
OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged" DataValueField="1">
            <asp:ListItem Value="1">1</asp:ListItem>
            <asp:ListItem Value="2">2</asp:ListItem>
            <asp:ListItem Value="3">3</asp:ListItem>
            <asp:ListItem Value="4">4</asp:ListItem>
            <asp:ListItem Value="5">5</asp:ListItem>
            <asp:ListItem Value="6">6</asp:ListItem>
            <asp:ListItem Value="7">7</asp:ListItem>
            <asp:ListItem Value="8">8</asp:ListItem>
            <asp:ListItem Value="9">9</asp:ListItem>
            <asp:ListItem Value="10">10</asp:ListItem>
            <asp:ListItem Value="11">11</asp:ListItem>
            <asp:ListItem Value="12">12</asp:ListItem>
            <asp:ListItem></asp:ListItem>
        </asp:DropDownList>
        /<asp:DropDownList id="DropDownList2" runat="server" Width="68px">
            <asp:ListItem Value="05">05</asp:ListItem>
            <asp:ListItem Value="06">06</asp:ListItem>
            <asp:ListItem Value="07">07</asp:ListItem>
            <asp:ListItem Value="08">08</asp:ListItem>
            <asp:ListItem Value="09">09</asp:ListItem>
            <asp:ListItem Value="10">10</asp:ListItem>
            <asp:ListItem Value="11">11</asp:ListItem>
            <asp:ListItem Value="12">12</asp:ListItem>
            <asp:ListItem Value="13">13</asp:ListItem>
            <asp:ListItem Value="14">14</asp:ListItem>
            <asp:ListItem Value="15">15</asp:ListItem>
        </asp:DropDownList>
    </p>
    <p>
        CVC:<asp:TextBox id="TextBox4" runat="server" Width="60px"></asp:TextBox>
    </p>
```

```
        Enter your credit card number:
        <asp:TextBox id="txtCCNumber" Runat="Server" MaxLength="20"
Columns="20"></asp:TextBox>
        <asp:CustomValidator id="CustomValidator1" Runat="Server" Text="Invalid Credit
Card Number!" Display="Dynamic" OnServerValidate="ValidateCCNumber"
ControlToValidate="txtCCNumber"></asp:CustomValidator>
        <br />
      </p>
      <p>
        <asp:Button id="Button1" onclick="Button_Click" Runat="Server"
Text="Submit!"></asp:Button>
      </p>
    </form>
</body>
</html
```

9.3.9. Default.aspx

```
<%@ Page Language="VB" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="PopularItems"
Src="_PopularItems.ascx" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<script runat="server">
Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
 ' Customize welcome message if personalization cookie is present
If Not Request.Cookies("ASPNETCommerce_FullName") Is Nothing Then
WelcomeMsg.Text = "Welcome " &
Request.Cookies("ASPNETCommerce_FullName").Value
End If
End Sub
</script>
<html>
<head>
```

```
<link href="ASPNETCommerce.css" type="text/css" rel="stylesheet" />
</head><body bottommargin="0" leftmargin="0" background="images/sitebkgrdnogray.gif"
topmargin="0" rightmargin="0" marginwidth="0" marginheight="0"><table cellspacing="0"
cellpadding="0" width="100%" border="0"><tbody><tr>
<td colspan="2">
<ASPNETCommerce:Header id="Header1" runat="server"></ASPNETCommerce:Header>
</td></tr><tr><td valign="top" width="145"><ASPNETCommerce:Menu id="Menu1"
runat="server"></ASPNETCommerce:Menu><img height="1" src="images/1x1.gif"
width="145" /> </td><td valign="top" nowrap="nowrap" align="left" width=*>
<table height="100%" cellspacing="0" cellpadding="0" width="100%" align="left"
border="0"><tbody><tr valign="top"><td nowrap="nowrap">  <br />
<img src="images/1x1.gif" width="24" align="left" /> <table cellspacing="0"
cellpadding="0" width="100%"><tbody>  <tr>  <td>
<table cellspacing="0" cellpadding="0" width="100%">  <tbody>   <tr>
<td class="HomeHead">
<asp:Label id="WelcomeMsg" runat="server">Welcome to the ASP.NET e-Commerce Web
Site</asp:Label></td>
</tr>  </tbody>
  </table><table cellspacing="0" cellpadding="2" width="600" border="0"> <tbody>
<tr valign="top"><td><table width="300">  <tbody>
<tr valign="top"><td>
<span class="NormalDouble">The Commerce Web Site demonstrates how extraordinarily
simple it is to create powerful, scalable applications and services for the .NET platform.
"click and mortar" retailer whose online presence is based on the ASP.NET Commerce
  Web site. <br /> </span></td>  </tr></tbody></table></td><td align="left">
<img src="ProductImages/sw7.gif" width="309" border="0" />
  <br />
span class="NormalDouble"><i>Blast off in a <a
href="ProductDetails.aspx?productID=373"><b>Asp.Net
Programming with C# </b> </a></i></span></td>
<td>  </td></tr>  <tr valign="top"><td>
<ASPNETCommerce:PopularItems
id="PopularItems1"runat="server"></ASPNETCommerce:PopularItems></td>
```

```
<td>
<br />
<span class="NormalDouble">To give the Commerce&nbsp;Web site &nbsp;a test spin,
simply starting browsing and add any items you want to your shopping cart. </span></td>
<td></td></tr></tbody></table></td></tr></tbody>
</table></td></tr></tbody>
</table></td></tr></tbody></table></body></html>
```

9.3.10. Error.aspx

```
<%@ Page %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<html> <head>
<link rel="stylesheet" type="text/css" href="ASPNETCommerce.css">
</head><body background="images/sitebkgrdnogray.gif" leftmargin="0" topmargin="0"
rightmargin="0" bottommargin="0" marginheight="0" marginwidth="0">
<table cellspacing="0" cellpadding="0" width="100%" border="0">
<tr>  <td colspan="2">
<ASPNETCommerce:Header ID="Header1" runat="server" /></td>
</tr><tr><td valign="top">
<ASPNETCommerce:Menu id="Menu1" runat="server" />
<img height="1" src="images/1x1.gif" width="145">
</td><td align="left" valign="top" width="100%" nowrap>
<table height="100%" align="left" cellspacing="0" cellpadding="0" width="100%"
border="0"> <tr valign="top"><td nowrap>
<br><img align="left" width="32" src="images/1x1.gif">
<table cellspacing="0" cellpadding="0" width="100%"><tr>
<td><table cellspacing="0" cellpadding="0" width="100%">
<tr><td class="HomeHead">  <h3>
  We are sorry, but an error occured during the
<br>processing of your last request.
<br>
```

```
<br>
This could be a result of either illegal input
<br>values, or a bug in our code. Sorry for the inconvenience.
 </h3></td> </tr></table></td></tr></table>
</td></tr></table> </td></tr></table>
</body>
</html>
```

9.3.11 Login.aspx

```
<%@ Page Language="VB" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<script runat="server">
Sub LoginBtn_Click(ByVal sender As Object, ByVal a As ImageClickEventArgs)
' Only attempt a login if all form fields on the page are valid
If Page.IsValid = True Then
' Save old ShoppingCartID
Dim shoppingCart As ASPNET.StarterKit.Commerce.ShoppingCartDB = New
ASPNET.StarterKit.Commerce.ShoppingCartDB()
Dim tempCartID As String = shoppingCart.GetShoppingCartId()
' Attempt to Validate User Credentials using CustomersDB
Dim accountSystem As ASPNET.StarterKit.Commerce.CustomersDB = New
ASPNET.StarterKit.Commerce.CustomersDB()
Dim customerId As String = accountSystem.Login(email.Text,
ASPNET.StarterKit.Commerce.Security.Encrypt(password.Text))
If customerId <> "" Then
' Migrate any existing shopping cart items into the permanent shopping cart
 shoppingCart.MigrateCart(tempCartID, customerId)
' Lookup the customer's full account details
Dim customerDetails As ASPNET.StarterKit.Commerce.customerDetails =
accountSystem.GetCustomerDetails(customerId)
' Store the user's fullname in a cookie for personalization purposes
```

```
Response.Cookies("ASPNETCommerce_FullName").Value = customerDetails.FullName
' Make the cookie persistent only if the user selects "persistent" login checkbox
If RememberLogin.Checked = True Then
Response.Cookies("ASPNETCommerce_FullName").Expires =
DateTime.Now.AddMonths(1)
End If
' Redirect browser back to originating page
 FormsAuthentication.RedirectFromLoginPage(customerId, RememberLogin.Checked)
Else
Message.Text = "Login Failed!"
End If
End If
End Sub
</script>
<html>
<head>
<link rel="stylesheet" type="text/css" href="ASPNETCommerce.css">
</head>
<body background="images/sitebkgrd.gif" leftmargin="0" topmargin="0" rightmargin="0"
bottommargin="0" marginheight="0" marginwidth="0">
<table cellspacing="0" cellpadding="0" width="100%" border="0"><tr>
<td colspan="2"><ASPNETCommerce:Header ID="Header1" runat="server" /></td>
</tr><tr><td valign="top">
<ASPNETCommerce:Menu id="Menu1" runat="server" />
<img height="1" src="images/1x1.gif" width="145"></td>
<td align="left" valign="top" width="100%" nowrap>
<table height="100%" align="left" cellspacing="0" cellpadding="0" width="100%"
border="0"><tr valign="top"><td nowrap>
<br><form runat="server">
<img align="left" width="24" height="1" src="images/1x1.gif">
<table cellspacing="0" cellpadding="0" width="100%" border="0"><tr>
<td class="ContentHead">
```

```
<img align="left" height="32" width="60" src="images/1x1.gif">Sign Into Your
Account
<br>
</td></tr></table>
<img align="left" height="1" width="92" src="images/1x1.gif">
<table height="100%" cellspacing="0" cellpadding="0" border="0">
<tr valign="top"><td width="550">
<asp:Label id="Message" class="ErrorText" runat="server" />
<br><br>
&nbsp;<span class="NormalBold">Email</span>
<br>&nbsp;<asp:TextBox size="25" id="email" runat="server" />&nbsp;
<asp:RequiredFieldValidator id="emailRequired" ControlToValidate="email"
Display="dynamic" Font-Name="verdana" Font-Size="9pt" ErrorMessage="'Name' must not
be left blank." runat="server" />
<asp:RegularExpressionValidator id="emailValid" ControlToValidate="email"
ValidationExpression="[\w\.-]+(\+[\w-]*)?@([\w-]+\.)+[\w-]+" Display="Dynamic"
ErrorMessage="Must use a valid email address." runat="server" />
&nbsp;<span class="NormalBold">Password</span><br>
&nbsp;<asp:TextBox id="password" textmode="password" size="25" runat="server"
/>&nbsp
<asp:RequiredFieldValidator id="passwordRequired" ControlToValidate="password"
Display="Static" Font-Name="verdana" Font-Size="9pt" ErrorMessage="'Password' must
not be left blank." runat="server" /><br>
<br>
<asp:checkbox id="RememberLogin" runat="server" />
<span class="NormalBold">Remember My Sign-In Across Browser Restarts</span>
<br>
<br>
<br>
<asp:ImageButton id="LoginBtn" ImageURL="images/sign_in_now.gif"
OnClick="LoginBtn_Click" runat="server" />
<br>
<br>
```

```
<span class="Normal">&nbsp;If you are a new user and you don't have an account with the
Commerce Starter Kit, then register for one now.</span>
<br>
<br>
<a href="register.aspx"><img border="0" src="images/register.gif"></a>
</td></tr></table></form></td></tr></table></td></tr></table>
</body>
</html>
```

9.3.12 OrderDetails.aspx

```
<%@ Page Language="VB" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<script runat="server">
Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
' Obtain Order ID from QueryString
Dim OrderID As Integer = CInt(Request.Params("OrderID"))
' Get the customer ID too
Dim CustomerId As String = User.Identity.Name
' Obtain Order Details from Database
  Dim orderHistory As ASPNET.StarterKit.Commerce.OrdersDB = New
ASPNET.StarterKit.Commerce.OrdersDB()
Dim myOrderDetails As ASPNET.StarterKit.Commerce.OrderDetails =
orderHistory.GetOrderDetails(OrderID, CustomerId)
' if order was found, display it
If Not (myOrderDetails Is Nothing) Then
' Bind Items to GridControl
GridControl1.DataSource = myOrderDetails.OrderItems
GridControl1.DataBind()
' Update labels with summary details
lblTotal.Text = String.Format("{0:c}", myOrderDetails.OrderTotal)
lblOrderNumber.Text = OrderID.ToString()
lblOrderDate.Text = myOrderDetails.OrderDate.ToShortDateString()
```

```
lblShipDate.Text = myOrderDetails.ShipDate.ToShortDateString()
Else
' otherwise display an error message
MyError.Text = "Order not found!"
detailsTable.Visible = False
End If
End Sub
</script>
<html>
<head>
<link rel="stylesheet" type="text/css" href="ASPNETCommerce.css">
</head>
<body background="images/sitebkgrd.gif" leftmargin="0" topmargin="0" rightmargin="0"
bottommargin="0" marginheight="0" marginwidth="0">
<table cellspacing="0" cellpadding="0" width="100%" border="0"><tr>
<td colspan="2"><ASPNETCommerce:Header ID="Header1" runat="server" />
  </td></tr><tr><td valign="top"><ASPNETCommerce:Menu id="Menu1" runat="server" />
<img height="1" src="images/1x1.gif" width="145"></td>
<td align="left" valign="top" width="100%" nowrap>
<table height="100%" align="left" cellspacing="0" cellpadding="0" width="100%"
border="0"><tr valign="top"><td nowrap>
<br>
<img align="left" width="24" src="images/1x1.gif">
<table cellspacing="0" cellpadding="0" width="100%" border="0"><tr>
<td class="ContentHead"><img align="left" height="32" width="60"
src="images/1x1.gif">Order Details
<br></td></tr></table>
<img align="left" height="15" width="86" src="images/1x1.gif" border="0">
<asp:Label id="MyError" CssClass="ErrorText" EnableViewState="false" runat="Server" />
<table id="detailsTable" height="100%" cellspacing="0" cellpadding="0" width="550"
border="0" EnableViewState="false" runat="server">
<tr valign="top">
  <td width="100%" class="Normal"><br>
```

```
<b>Your Order Number Is: </b>
<asp:Label ID="lblOrderNumber" EnableViewState="false" runat="server" />
<br>
<b>Order Date: </b>
<asp:Label ID="lblOrderDate" EnableViewState="false" runat="server" />
<br><b>Ship Date: </b>
<asp:Label ID="lblShipDate" EnableViewState="false" runat="server" />
<br>
<br>
<asp:DataGrid id="GridControl1" width="90%" BorderColor="black" GridLines="Vertical"
cellpadding="4" cellspacing="0" Font-Name="Verdana" Font-Size="8pt"
ShowFooter="true" HeaderStyle-CssClass="CartListHead" FooterStyle-
CssClass="cartlistfooter" ItemStyle-CssClass="CartListItem" AlternatingItemStyle-
CssClass="CartListItemAlt" AutoGenerateColumns="false" runat="server">
<Columns>
<asp:BoundColumn HeaderText="Product Name" DataField="ModelName" />
<asp:BoundColumn HeaderText="Model Number" DataField="ModelNumber" />
<asp:BoundColumn HeaderText="Quantity" DataField="Quantity" />
<asp:BoundColumn HeaderText="Price" DataField="UnitCost" DataFormatString="{0:c}"
/>
<asp:BoundColumn HeaderText="Subtotal" DataField="ExtendedAmount"
DataFormatString="{0:c}" /></Columns>
</asp:DataGrid>  <br><b>Total: </b>
<asp:Label ID="lblTotal" EnableViewState="false" runat="server" />
</td>  </tr></table>  </td></tr></table></td></tr></table>
</body>
</html>
```

9.3.13 OrderList.aspx

```
<%@ Page Language="VB" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<%@ import Namespace="System.Data.SqlClient" %>
```

```
<script runat="server">
Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
Dim customerID As String = User.Identity.Name
' Obtain and bind a list of all orders ever placed by visiting customer
Dim orderHistory As ASPNET.StarterKit.Commerce.OrdersDB = New
ASPNET.StarterKit.Commerce.OrdersDB()
MyList.DataSource = orderHistory.GetCustomerOrders(customerID)
MyList.DataBind()
' Hide the list and display a message if no orders have ever been made
If MyList.Items.Count = 0 Then
MyError.Text = "You have no orders to display."
MyList.Visible = False
End If
End Sub
</script>
<html>
<head>
<link rel="stylesheet" type="text/css" href="ASPNETCommerce.css">
</head>
<body background="images/sitebkgrd.gif" leftmargin="0" topmargin="0" rightmargin="0"
bottommargin="0" marginheight="0" marginwidth="0">
<table cellspacing="0" cellpadding="0" width="100%" border="0"><tr>
<td colspan="2"><ASPNETCommerce:Header ID="Header1" runat="server" />
</td></tr><tr><td valign="top">
<ASPNETCommerce:Menu id="Menu1" runat="server" />
<img height="1" src="images/1x1.gif" width="145"></td><td align="left" valign="top"
width="100%" nowrap><table height="100%" align="left" cellspacing="0" cellpadding="0"
width="100%" border="0"><tr valign="top">
<td nowrap><br><form runat="server">
<img align="left" width="24" src="images/1x1.gif">
<table cellspacing="0" cellpadding="0" width="100%" border="0">
 <tr>  <td class="ContentHead">
<img align="left" height="32" width="60" src="images/1x1.gif">Account History
```

```
<br></td></tr></table>
<img align="left" height="4" width="110" src="images/1x1.gif"> <font color="red">
<asp:Label id="MyError" class="ErrorText" runat="Server" />
 </font>
 <br>
  <img align="left" height="15" width="84" src="images/1x1.gif" border="0">
<table height="100%" cellspacing="0" cellpadding="0" width="550" border="0">
<tr valign="top"><td width="100%">
<asp:DataGrid id="MyList" width="90%" BorderColor="black" GridLines="Vertical"
cellpadding="4" cellspacing="0" Font-Name="Verdana" Font-Size="8pt"
ShowFooter="true" HeaderStyle-CssClass="CartListHead" FooterStyle-
CssClass="cartlistfooter" ItemStyle-CssClass="CartListItem" AlternatingItemStyle-
CssClass="CartListItemAlt" AutoGenerateColumns="false" runat="server">
<Columns>
<asp:BoundColumn HeaderText="Order ID" DataField="OrderID" />
<asp:BoundColumn HeaderText="Order Date" DataField="OrderDate"
DataFormatString="{0:d}" />
<asp:BoundColumn HeaderText="Order Total" DataField="OrderTotal"
DataFormatString="{0:c}" />
<asp:BoundColumn HeaderText="Ship Date" DataField="ShipDate"
DataFormatString="{0:d}" />
<asp:HyperLinkColumn HeaderText="Show Details" Text="Show Details"
DataNavigateUrlField="OrderID"
DataNavigateUrlFormatString="orderdetails.aspx?OrderID={0}" />
</Columns></asp:DataGrid>
</td></tr></table></form></td></tr></table></td></tr></table>
</body>
</html>
```

9.3.14. Product Detail.aspx

```
<%@ Page Language="VB" EnableViewState="false" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="ReviewList"
Src="_ReviewList.ascx" %>
```

```
<%@ Register TagPrefix="ASPNETCommerce" TagName="AlsoBought"
Src="_AlsoBought.ascx" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<%@ outputcache duration="60" varybyparam="ProductID" %>
<script runat="server">
Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
'Obtain ProductID from QueryString
 Dim ProductID As Integer = CInt(Request.Params("ProductID"))
' Obtain Product Details
Dim products As ASPNET.StarterKit.Commerce.ProductsDB = New
ASPNET.StarterKit.Commerce.ProductsDB()
Dim myProductDetails As ASPNET.StarterKit.Commerce.ProductDetails =
products.GetProductDetails(ProductID)
' Update Controls with Product Detailsdesc.Text = myProductDetails.Description
UnitCost.Text = String.Format("{0:c}", myProductDetails.UnitCost)
ModelName.Text = myProductDetails.ModelName
ModelNumber.Text = myProductDetails.ModelNumber.ToString()
ProductImage.ImageUrl = "ProductImages/" & myProductDetails.ProductImage
addToCart.NavigateUrl = "AddToCart.aspx?ProductID=" & ProductID
ReviewList.ProductID = ProductID
AlsoBoughtList.ProductID = ProductID
End Sub
</script>
<html>
<head>
<link href="ASPNETCommerce.css" type="text/css" rel="stylesheet">
</head>
<body bottomMargin="0" leftMargin="0" background="images/sitebkgrd.gif"
topMargin="0" rightMargin="0" marginwidth="0" marginheight="0">
<table cellSpacing="0" cellPadding="0" width="100%" border="0">
<tr><td colSpan="2">
```

```
<ASPNETCommerce:Header id="Header1" runat="server" /></td></tr>
<tr><td vAlign="top" width=145>
<ASPNETCommerce:Menu id="Menu1" runat="server" />
 <img height="1" src="images/1x1.gif" width="145"></td>
<td vAlign="top" align="left">
<table height="100%" cellSpacing="0" cellPadding="0" width="620" align="left"
border="0">
<tr vAlign="top"><td><br>
<img src="images/1x1.gif" width="24" align="left">
<table cellSpacing="0" cellPadding="0" width="100%" border="0"><tr>
<td class="ContentHead">
<img height="32" src="images/1x1.gif" width="60" align="left"><asp:label
id="ModelName" runat="server" />
<br>
</td></tr></table>
<table cellSpacing="0" cellPadding="0" width="100%" border="0" valign="top">
<tr vAlign="top"><td rowspan="2">
<img height="1" width="24" src="images/1x1.gif">
</td><td width="309"><img height="15" src="images/1x1.gif">
<br>
<asp:image id="ProductImage" runat="server" height="185" width="309" border="0" />
 <br>
<br>
<img height="20" src="images/1x1.gif" width="72"><span class="UnitCost"><b>Your
Price:</b>&nbsp;<asp:label id="UnitCost" runat="server" /></span>
<br>
<img height="20" src="images/1x1.gif" width="72"><span
class="ModelNumber"><b>Model Number:</b>&nbsp;<asp:label id="ModelNumber"
runat="server" /></span>  <br>
<img height="30" src="images/1x1.gif" width="72"><asp:hyperlink id="addToCart"
runat="server" ImageUrl="images/add_to_cart.gif" />
</td> <td><table width="300" border="0">
```

```
<tr><td vAlign="top">
<asp:label class="NormalDouble" id="desc" runat="server"></asp:label>
<br></td></tr></table>
<img height="30" src="images/1x1.gif">
<ASPNETCommerce:AlsoBought id="AlsoBoughtList" runat="server" />
</td></tr><tr></tr></table>
<table border="0"> <tr><td>
<img src="images/1x1.gif" width="89" height="20"></td>
<td width="100%"><ASPNETCommerce:ReviewList id="ReviewList" runat="server" />
</td> </tr></table></td></tr></table></td></tr></table>
</body>
</HTML>
```

9.3.15. ProductList.aspx

```
<%@ Page Language="VB" EnableViewState="false" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="ReviewList"
Src="_ReviewList.ascx" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="AlsoBought"
Src="_AlsoBought.ascx" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<%@ outputcache duration="60" varybyparam="ProductID" %>
<script runat="server">
 Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
'Obtain ProductID from QueryString
Dim ProductID As Integer = CInt(Request.Params("ProductID"))
' Obtain Product Details
Dim products As ASPNET.StarterKit.Commerce.ProductsDB = New
ASPNET.StarterKit.Commerce.ProductsDB()
Dim myProductDetails As ASPNET.StarterKit.Commerce.ProductDetails =
products.GetProductDetails(ProductID)
```

```
' Update Controls with Product Details
desc.Text = myProductDetails.Description
UnitCost.Text = String.Format("{0:c}", myProductDetails.UnitCost)
ModelName.Text = myProductDetails.ModelName
ModelNumber.Text = myProductDetails.ModelNumber.ToString()
ProductImage.ImageUrl = "ProductImages/" & myProductDetails.ProductImage
addToCart.NavigateUrl = "AddToCart.aspx?ProductID=" & ProductID
ReviewList.ProductID = ProductID
AlsoBoughtList.ProductID = ProductID
End Sub
</script>
<html>
<head>
<link href="ASPNETCommerce.css" type="text/css" rel="stylesheet">
 </head>
<body bottomMargin="0" leftMargin="0" background="images/sitebkgrd.gif"
topMargin="0" rightMargin="0" marginwidth="0" marginheight="0">
<table cellSpacing="0" cellPadding="0" width="100%" border="0">
<tr><td colSpan="2">
<ASPNETCommerce:Header id="Header1" runat="server" />
</td></tr><tr>
<td vAlign="top" width=145>
<ASPNETCommerce:Menu id="Menu1" runat="server" />
<img height="1" src="images/1x1.gif" width="145"></td>
<td vAlign="top" align="left">
<table height="100%" cellSpacing="0" cellPadding="0" width="620" align="left"
border="0">
<tr vAlign="top"><td>
<br><img src="images/1x1.gif" width="24" align="left">
<table cellSpacing="0" cellPadding="0" width="100%" border="0">
 <tr><td class="ContentHead">
```

```
<img height="32" src="images/1x1.gif" width="60" align="left"><asp:label
id="ModelName" runat="server" />
<br></td></tr></table>
<table cellSpacing="0" cellPadding="0" width="100%" border="0" valign="top">
<tr vAlign="top">
 <td rowspan="2">
<img height="1" width="24" src="images/1x1.gif">
 </td><td width="309">
<img height="15" src="images/1x1.gif">
<br><asp:image id="ProductImage" runat="server" height="185" width="309" border="0" />
<br>
<br>
<img height="20" src="images/1x1.gif" width="72"><span class="UnitCost"><b>Your
Price:</b>&nbsp;<asp:label id="UnitCost" runat="server" /></span>
<br>
<img height="20" src="images/1x1.gif" width="72"><span
class="ModelNumber"><b>Model Number:</b>&nbsp;<asp:label id="ModelNumber"
runat="server" /></span>
<br>
<img height="30" src="images/1x1.gif" width="72"><asp:hyperlink id="addToCart"
runat="server" ImageUrl="images/add_to_cart.gif" />
</td><td><table width="300" border="0">
<tr>  <td vAlign="top">
<asp:label class="NormalDouble" id="desc" runat="server"></asp:label><br>
</td></tr></table><img height="30" src="images/1x1.gif">
<ASPNETCommerce:AlsoBought id="AlsoBoughtList" runat="server" /></td>
</tr><tr></tr>  </table>
<table border="0">
  <tr><td><img src="images/1x1.gif" width="89" height="20"> </td>
<td width="100%"><ASPNETCommerce:ReviewList id="ReviewList" runat="server" />
  </td> </tr></table></td></tr></table></td></tr></table>
</body>
</HTML>
```

9.3.16. Register.aspx

```
<%@ Page Language="VB" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<script runat="server">
Sub RegisterBtn_Click(ByVal sender As Object, ByVal e As ImageClickEventArgs)
' Only attempt a login if all form fields on the page are valid
If Page.IsValid = True Then
' Store off old temporary shopping cart ID
Dim shoppingCart As ASPNET.StarterKit.Commerce.ShoppingCartDB = New
ASPNET.StarterKit.Commerce.ShoppingCartDB()
Dim tempCartId As String = shoppingCart.GetShoppingCartId()
' Add New Customer to CustomerDB database
Dim accountSystem As ASPNET.StarterKit.Commerce.CustomersDB = New
ASPNET.StarterKit.Commerce.CustomersDB()
Dim customerId as String = accountSystem.AddCustomer(Server.HtmlEncode(Name.Text),
Email.Text, ASPNET.StarterKit.Commerce.Security.Encrypt(Password.Text))
If customerId <> "" Then
' Set the user's authentication name to the customerId
FormsAuthentication.SetAuthCookie(customerId, False)
' Migrate any existing shopping cart items into the permanent shopping cart
shoppingCart.MigrateCart(tempCartId, customerId)
' Store the user's fullname in a cookie for personalization purposes
Response.Cookies("ASPNETCommerce_FullName").Value =
Server.HtmlEncode(Name.Text)
' Redirect browser back to shopping cart page
Response.Redirect("ShoppingCart.aspx")
Else
MyError.Text = "Registration failed:&nbsp; That email address is already
registered.<br><img align=left height=1 width=92 src=images/1x1.gif>"
End If
End If
End Sub
</script>
<html>
<head>
<link rel="stylesheet" type="text/css" href="ASPNETCommerce.css">
</head>
<body background="images/sitebkgrd.gif" leftmargin="0" topmargin="0" rightmargin="0"
bottommargin="0" marginheight="0" marginwidth="0">
<table cellspacing="0" cellpadding="0" width="100%" border="0">
<tr><td colspan="2"><ASPNETCommerce:Header ID="Header1" runat="server" />
</td></tr><tr><td valign="top">
<ASPNETCommerce:Menu id="Menu1" runat="server" />
<img height="1" src="images/1x1.gif" width="145">
</td><td align="left" valign="top" width="100%" nowrap>
<table height="100%" align="left" cellspacing="0" cellpadding="0" width="100%"
border="0"><tr valign="top"><td nowrap><br><form runat="server">
<img align="left" width="24" height="1" src="images/1x1.gif">
<table cellspacing="0" cellpadding="0" width="100%" border="0">
<tr><td class="ContentHead">
<img align="left" height="32" width="60" src="images/1x1.gif">Create a New Account
<br></td></tr></table>
<img align="left" height="1" width="92" src="images/1x1.gif">
<asp:Label id="MyError" CssClass="ErrorText" EnableViewState="false" runat="Server" />
<table height="100%" cellspacing="0" cellpadding="0" width="500" border="0">
<tr valign="top">
<td width="550">
<br>
<br><span class="NormalBold">Full Name</span>
<br><asp:TextBox size="25" id="Name" runat="server" />
<asp:RequiredFieldValidator ControlToValidate="Name" Display="dynamic" Font-
Name="verdana" Font-Size="9pt" ErrorMessage="'Name' must not be left blank."
runat="server"></asp:RequiredFieldValidator>
<br>
```

```
<br><span class="NormalBold">Email</span>
<br><asp:TextBox size="25" id="Email" runat="server" />
<asp:RegularExpressionValidator ControlToValidate="Email" ValidationExpression="[\w\.-
]+(\+[\w-]*)?@([\w-]+\.)+[\w-]+" Display="Dynamic" Font-Name="verdana" Font-
Size="9pt" ErrorMessage="Must use a valid email address."
runat="server"></asp:RegularExpressionValidator>
<asp:RequiredFieldValidator ControlToValidate="Email" Display="dynamic" Font-
Name="verdana" Font-Size="9pt" ErrorMessage="'Email' must not be left blank."
runat="server"></asp:RequiredFieldValidator>
<br>
<br>
<span class="NormalBold">Password</span>
<br><asp:TextBox size="25" id="Password" TextMode="Password" runat="server" />
<asp:RequiredFieldValidator ControlToValidate="Password" Display="dynamic" Font-
Name="verdana" Font-Size="9pt" ErrorMessage="'Password' must not be left blank."
runat="server"></asp:RequiredFieldValidator>
<br>
<br><span class="NormalBold">Confirm Password</span>
<br><asp:TextBox size="25" id="ConfirmPassword" TextMode="Password" runat="server"
/><asp:RequiredFieldValidator ControlToValidate="ConfirmPassword" Display="dynamic"
Font-Name="verdana" Font-Size="9pt" ErrorMessage="'Confirm' must not be left blank."
runat="server"></asp:RequiredFieldValidator>
<asp:CompareValidator ControlToValidate="ConfirmPassword"
ControlToCompare="Password" Display="Dynamic" Font-Name="verdana" Font-
Size="9pt" ErrorMessage="Password fields do not match."
runat="server"></asp:CompareValidator>
<br>
<br><asp:ImageButton id="RegisterBtn" OnClick="RegisterBtn_Click"
ImageUrl="images/submit.gif" runat="server" />
<br></td></tr></table></form></td></tr></table></td></tr>
</table>
</body>
</html>
```

9.3.17.1. ReviewAdd.aspx

```
<%@ Page Language="VB" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<script runat="server">
Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
If Page.IsPostBack <> True Then
' Obtain ProductID of Product to Review
 Dim productID As Integer = CInt(Request("productID"))
' Populate Product Name on Page
Dim products As ASPNET.StarterKit.Commerce.ProductsDB = New
ASPNET.StarterKit.Commerce.ProductsDB()
ModelName.Text = products.GetProductDetails(productID).ModelName
' Store ProductID in Page State to use on PostBack
ViewState("productID") = productID
End If
End Sub
Sub ReviewAddBtn_Click(ByVal sender As Object, ByVal e As ImageClickEventArgs)
' Only add the review if all fields on the page are valid
If Page.IsValid = True Then
' Obtain ProductID from Page State
 Dim productID As Integer = CInt(ViewState("productID"))
' Obtain Rating number of RadioButtonList
Dim _rating As Integer = CInt(rating.SelectedItem.Value)
' Add Review to ReviewsDB.  HtmlEncode before entry
Dim review As ASPNET.StarterKit.Commerce.ReviewsDB = New
ASPNET.StarterKit.Commerce.ReviewsDB()
 review.AddReview(productID, Server.HtmlEncode(Name.Text),
Server.HtmlEncode(Email.Text), _rating, Server.HtmlEncode(Comment.Text))
' Redirect client back to the originating product details page
Response.Redirect("ProductDetails.aspx?ProductID=" & productID)

End If
```

```
<br>
This could be a result of either illegal input
<br>values, or a bug in our code. Sorry for the inconvenience.
 </h3></td>  </tr></table></td></tr></table>
</td></tr></table>  </td></tr></table>
</body>
</html>
```

9.3.11 Login.aspx

```
<%@ Page Language="VB" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<script runat="server">
Sub LoginBtn_Click(ByVal sender As Object, ByVal a As ImageClickEventArgs)
' Only attempt a login if all form fields on the page are valid
If Page.IsValid = True Then
' Save old ShoppingCartID
Dim shoppingCart As ASPNET.StarterKit.Commerce.ShoppingCartDB = New
ASPNET.StarterKit.Commerce.ShoppingCartDB()
Dim tempCartID As String = shoppingCart.GetShoppingCartId()
' Attempt to Validate User Credentials using CustomersDB
Dim accountSystem As ASPNET.StarterKit.Commerce.CustomersDB = New
ASPNET.StarterKit.Commerce.CustomersDB()
Dim customerId As String = accountSystem.Login(email.Text,
ASPNET.StarterKit.Commerce.Security.Encrypt(password.Text))
If customerId <> "" Then
' Migrate any existing shopping cart items into the permanent shopping cart
 shoppingCart.MigrateCart(tempCartID, customerId)
' Lookup the customer's full account details
Dim customerDetails As ASPNET.StarterKit.Commerce.customerDetails =
accountSystem.GetCustomerDetails(customerId)
' Store the user's fullname in a cookie for personalization purposes
```

```
Response.Cookies("ASPNETCommerce_FullName").Value = customerDetails.FullName
' Make the cookie persistent only if the user selects "persistent" login checkbox
If RememberLogin.Checked = True Then
Response.Cookies("ASPNETCommerce_FullName").Expires =
DateTime.Now.AddMonths(1)
End If
' Redirect browser back to originating page
 FormsAuthentication.RedirectFromLoginPage(customerId, RememberLogin.Checked)
Else
Message.Text = "Login Failed!"
End If
End If
End Sub
</script>
<html>
<head>
<link rel="stylesheet" type="text/css" href="ASPNETCommerce.css">
</head>
<body background="images/sitebkgrd.gif" leftmargin="0" topmargin="0" rightmargin="0"
bottommargin="0" marginheight="0" marginwidth="0">
<table cellspacing="0" cellpadding="0" width="100%" border="0"><tr>
<td colspan="2"><ASPNETCommerce:Header ID="Header1" runat="server" /></td>
</tr><tr><td valign="top">
<ASPNETCommerce:Menu id="Menu1" runat="server" />
<img height="1" src="images/1x1.gif" width="145"></td>
<td align="left" valign="top" width="100%" nowrap>
<table height="100%" align="left" cellspacing="0" cellpadding="0" width="100%"
border="0"><tr valign="top"><td nowrap>
<br><form runat="server">
<img align="left" width="24" height="1" src="images/1x1.gif">
<table cellspacing="0" cellpadding="0" width="100%" border="0"><tr>
<td class="ContentHead">
```

```
<img align="left" height="32" width="60" src="images/1x1.gif">Sign Into Your
Account
<br>
</td></tr></table>
<img align="left" height="1" width="92" src="images/1x1.gif">
<table height="100%" cellspacing="0" cellpadding="0" border="0">
<tr valign="top"><td width="550">
<asp:Label id="Message" class="ErrorText" runat="server" />
<br><br>
&nbsp;<span class="NormalBold">Email</span>
<br>&nbsp;<asp:TextBox size="25" id="email" runat="server" />&nbsp;
<asp:RequiredFieldValidator id="emailRequired" ControlToValidate="email"
Display="dynamic" Font-Name="verdana" Font-Size="9pt" ErrorMessage="'Name' must not
be left blank." runat="server" />
<asp:RegularExpressionValidator id="emailValid" ControlToValidate="email"
ValidationExpression="[\w\.-]+(\+[\w-]*)?@([\w-]+\.)+[\w-]+" Display="Dynamic"
ErrorMessage="Must use a valid email address." runat="server" />
&nbsp;<span class="NormalBold">Password</span><br>
&nbsp;<asp:TextBox id="password" textmode="password" size="25" runat="server"
/>&nbsp
<asp:RequiredFieldValidator id="passwordRequired" ControlToValidate="password"
Display="Static" Font-Name="verdana" Font-Size="9pt" ErrorMessage="'Password' must
not be left blank." runat="server" /><br>
<br>
<asp:checkbox id="RememberLogin" runat="server" />
<span class="NormalBold">Remember My Sign-In Across Browser Restarts</span>
<br>
<br>
<br>
<asp:ImageButton id="LoginBtn" ImageURL="images/sign_in_now.gif"
OnClick="LoginBtn_Click" runat="server" />
<br>
<br>
```

```
<span class="Normal">&nbsp;If you are a new user and you don't have an account with the
Commerce Starter Kit, then register for one now.</span>
<br>
<br>
<a href="register.aspx"><img border="0" src="images/register.gif"></a>
</td></tr></table></form></td></tr></table></td></tr></table>
</body>
</html>
```

9.3.12 OrderDetails.aspx

```
<%@ Page Language="VB" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<script runat="server">
Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
' Obtain Order ID from QueryString
Dim OrderID As Integer = CInt(Request.Params("OrderID"))
' Get the customer ID too
Dim CustomerId As String = User.Identity.Name
' Obtain Order Details from Database
 Dim orderHistory As ASPNET.StarterKit.Commerce.OrdersDB = New
ASPNET.StarterKit.Commerce.OrdersDB()
Dim myOrderDetails As ASPNET.StarterKit.Commerce.OrderDetails =
orderHistory.GetOrderDetails(OrderID, CustomerId)
' if order was found, display it
If Not (myOrderDetails Is Nothing) Then
' Bind Items to GridControl
GridControl1.DataSource = myOrderDetails.OrderItems
GridControl1.DataBind()
' Update labels with summary details
lblTotal.Text = String.Format("{0:c}", myOrderDetails.OrderTotal)
lblOrderNumber.Text = OrderID.ToString()
lblOrderDate.Text = myOrderDetails.OrderDate.ToShortDateString()
```

```
lblShipDate.Text = myOrderDetails.ShipDate.ToShortDateString()
Else
' otherwise display an error message
MyError.Text = "Order not found!"
detailsTable.Visible = False
End If
End Sub
</script>
<html>
<head>
<link rel="stylesheet" type="text/css" href="ASPNETCommerce.css">
</head>
<body background="images/sitebkgrd.gif" leftmargin="0" topmargin="0" rightmargin="0"
bottommargin="0" marginheight="0" marginwidth="0">
<table cellspacing="0" cellpadding="0" width="100%" border="0"><tr>
<td colspan="2"><ASPNETCommerce:Header ID="Header1" runat="server" />
 </td></tr><tr><td valign="top"><ASPNETCommerce:Menu id="Menu1" runat="server" />
<img height="1" src="images/1x1.gif" width="145"></td>
<td align="left" valign="top" width="100%" nowrap>
<table height="100%" align="left" cellspacing="0" cellpadding="0" width="100%"
border="0"><tr valign="top"><td nowrap>
<br>
<img align="left" width="24" src="images/1x1.gif">
<table cellspacing="0" cellpadding="0" width="100%" border="0"><tr>
<td class="ContentHead"><img align="left" height="32" width="60"
src="images/1x1.gif">Order Details
<br></td></tr></table>
<img align="left" height="15" width="86" src="images/1x1.gif" border="0">
<asp:Label id="MyError" CssClass="ErrorText" EnableViewState="false" runat="Server" />
<table id="detailsTable" height="100%" cellspacing="0" cellpadding="0" width="550"
border="0" EnableViewState="false" runat="server">
<tr valign="top">
  <td width="100%" class="Normal"><br>
```

```
<b>Your Order Number Is: </b>
<asp:Label ID="lblOrderNumber" EnableViewState="false" runat="server" />
<br>
<b>Order Date: </b>
<asp:Label ID="lblOrderDate" EnableViewState="false" runat="server" />
<br><b>Ship Date: </b>
<asp:Label ID="lblShipDate" EnableViewState="false" runat="server" />
<br>
<br>
<asp:DataGrid id="GridControl1" width="90%" BorderColor="black" GridLines="Vertical"
cellpadding="4" cellspacing="0" Font-Name="Verdana" Font-Size="8pt"
ShowFooter="true" HeaderStyle-CssClass="CartListHead" FooterStyle-
CssClass="cartlistfooter" ItemStyle-CssClass="CartListItem" AlternatingItemStyle-
CssClass="CartListItemAlt" AutoGenerateColumns="false" runat="server">
<Columns>
<asp:BoundColumn HeaderText="Product Name" DataField="ModelName" />
<asp:BoundColumn HeaderText="Model Number" DataField="ModelNumber" />
<asp:BoundColumn HeaderText="Quantity" DataField="Quantity" />
<asp:BoundColumn HeaderText="Price" DataField="UnitCost" DataFormatString="{0:c}"
/>
<asp:BoundColumn HeaderText="Subtotal" DataField="ExtendedAmount"
DataFormatString="{0:c}" /></Columns>
</asp:DataGrid>  <br><b>Total: </b>
<asp:Label ID="lblTotal" EnableViewState="false" runat="server" />
</td>  </tr></table>  </td></tr></table></td></tr></table>
</body>
</html>
```

9.3.13 OrderList.aspx

```
<%@ Page Language="VB" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<%@ import Namespace="System.Data.SqlClient" %>
```

```
<script runat="server">
Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
Dim customerID As String = User.Identity.Name
' Obtain and bind a list of all orders ever placed by visiting customer
Dim orderHistory As ASPNET.StarterKit.Commerce.OrdersDB = New
ASPNET.StarterKit.Commerce.OrdersDB()
MyList.DataSource = orderHistory.GetCustomerOrders(customerID)
MyList.DataBind()
' Hide the list and display a message if no orders have ever been made
If MyList.Items.Count = 0 Then
MyError.Text = "You have no orders to display."
MyList.Visible = False
End If
End Sub
</script>
<html>
<head>
<link rel="stylesheet" type="text/css" href="ASPNETCommerce.css">
</head>
<body background="images/sitebkgrd.gif" leftmargin="0" topmargin="0" rightmargin="0"
bottommargin="0" marginheight="0" marginwidth="0">
<table cellspacing="0" cellpadding="0" width="100%" border="0"><tr>
<td colspan="2"><ASPNETCommerce:Header ID="Header1" runat="server" />
</td></tr><tr><td valign="top">
<ASPNETCommerce:Menu id="Menu1" runat="server" />
<img height="1" src="images/1x1.gif" width="145"></td><td align="left" valign="top"
width="100%" nowrap><table height="100%" align="left" cellspacing="0" cellpadding="0"
width="100%" border="0"><tr valign="top">
<td nowrap><br><form runat="server">
<img align="left" width="24" src="images/1x1.gif">
<table cellspacing="0" cellpadding="0" width="100%" border="0">
 <tr>  <td class="ContentHead">
<img align="left" height="32" width="60" src="images/1x1.gif">Account History
```

```
<br></td></tr></table>
<img align="left" height="4" width="110" src="images/1x1.gif"> <font color="red">
<asp:Label id="MyError" class="ErrorText" runat="Server" />
 </font>
 <br>
  <img align="left" height="15" width="84" src="images/1x1.gif" border="0">
<table height="100%" cellspacing="0" cellpadding="0" width="550" border="0">
<tr valign="top"><td width="100%">
<asp:DataGrid id="MyList" width="90%" BorderColor="black" GridLines="Vertical"
cellpadding="4" cellspacing="0" Font-Name="Verdana" Font-Size="8pt"
ShowFooter="true" HeaderStyle-CssClass="CartListHead" FooterStyle-
CssClass="cartlistfooter" ItemStyle-CssClass="CartListItem" AlternatingItemStyle-
CssClass="CartListItemAlt" AutoGenerateColumns="false" runat="server">
<Columns>
<asp:BoundColumn HeaderText="Order ID" DataField="OrderID" />
<asp:BoundColumn HeaderText="Order Date" DataField="OrderDate"
DataFormatString="{0:d}" />
<asp:BoundColumn HeaderText="Order Total" DataField="OrderTotal"
DataFormatString="{0:c}" />
<asp:BoundColumn HeaderText="Ship Date" DataField="ShipDate"
DataFormatString="{0:d}" />
<asp:HyperLinkColumn HeaderText="Show Details" Text="Show Details"
DataNavigateUrlField="OrderID"
DataNavigateUrlFormatString="orderdetails.aspx?OrderID={0}" />
</Columns></asp:DataGrid>
</td></tr></table></form></td></tr></table></td></tr></table>
</body>
</html>
```

9.3.14. Product Detail.aspx

```
<%@ Page Language="VB" EnableViewState="false" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="ReviewList"
Src="_ReviewList.ascx" %>
```

```
<%@ Register TagPrefix="ASPNETCommerce" TagName="AlsoBought"
Src="_AlsoBought.ascx" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<%@ outputcache duration="60" varybyparam="ProductID" %>
<script runat="server">
Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
'Obtain ProductID from QueryString
 Dim ProductID As Integer = CInt(Request.Params("ProductID"))
' Obtain Product Details
Dim products As ASPNET.StarterKit.Commerce.ProductsDB = New
ASPNET.StarterKit.Commerce.ProductsDB()
Dim myProductDetails As ASPNET.StarterKit.Commerce.ProductDetails =
products.GetProductDetails(ProductID)
' Update Controls with Product Detailsdesc.Text = myProductDetails.Description
UnitCost.Text = String.Format("{0:c}", myProductDetails.UnitCost)
ModelName.Text = myProductDetails.ModelName
ModelNumber.Text = myProductDetails.ModelNumber.ToString()
ProductImage.ImageUrl = "ProductImages/" & myProductDetails.ProductImage
addToCart.NavigateUrl = "AddToCart.aspx?ProductID=" & ProductID
ReviewList.ProductID = ProductID
AlsoBoughtList.ProductID = ProductID
End Sub
</script>
<html>
<head>
<link href="ASPNETCommerce.css" type="text/css" rel="stylesheet">
 </head>
<body bottomMargin="0" leftMargin="0" background="images/sitebkgrd.gif"
topMargin="0" rightMargin="0" marginwidth="0" marginheight="0">
<table cellSpacing="0" cellPadding="0" width="100%" border="0">
<tr><td colSpan="2">
```

```
<ASPNETCommerce:Header id="Header1" runat="server" /></td></tr>
<tr><td vAlign="top" width=145>
<ASPNETCommerce:Menu id="Menu1" runat="server" />
 <img height="1" src="images/1x1.gif" width="145"></td>
<td vAlign="top" align="left">
<table height="100%" cellSpacing="0" cellPadding="0" width="620" align="left"
border="0">
<tr vAlign="top"><td><br>
<img src="images/1x1.gif" width="24" align="left">
<table cellSpacing="0" cellPadding="0" width="100%" border="0"><tr>
<td class="ContentHead">
<img height="32" src="images/1x1.gif" width="60" align="left"><asp:label
id="ModelName" runat="server" />
<br>
</td></tr></table>
<table cellSpacing="0" cellPadding="0" width="100%" border="0" valign="top">
<tr vAlign="top"><td rowspan="2">
<img height="1" width="24" src="images/1x1.gif">
</td><td width="309"><img height="15" src="images/1x1.gif">
<br>
<asp:image id="ProductImage" runat="server" height="185" width="309" border="0" />
 <br>
<br>
<img height="20" src="images/1x1.gif" width="72"><span class="UnitCost"><b>Your
Price:</b>&nbsp;<asp:label id="UnitCost" runat="server" /></span>
<br>
<img height="20" src="images/1x1.gif" width="72"><span
class="ModelNumber"><b>Model Number:</b>&nbsp;<asp:label id="ModelNumber"
runat="server" /></span>  <br>
<img height="30" src="images/1x1.gif" width="72"><asp:hyperlink id="addToCart"
runat="server" ImageUrl="images/add_to_cart.gif" />
</td> <td><table width="300" border="0">
```

```
<tr><td vAlign="top">
<asp:label class="NormalDouble" id="desc" runat="server"></asp:label>
<br></td></tr></table>
<img height="30" src="images/1x1.gif">
<ASPNETCommerce:AlsoBought id="AlsoBoughtList" runat="server" />
</td></tr><tr></tr></table>
<table border="0"> <tr><td>
<img src="images/1x1.gif" width="89" height="20"></td>
<td width="100%"><ASPNETCommerce:ReviewList id="ReviewList" runat="server" />
</td> </tr></table></td></tr></table></td></tr></table>
</body>
</HTML>
```

9.3.15. ProductList.aspx

```
<%@ Page Language="VB" EnableViewState="false" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="ReviewList"
Src="_ReviewList.ascx" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="AlsoBought"
Src="_AlsoBought.ascx" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<%@ outputcache duration="60" varybyparam="ProductID" %>
<script runat="server">
 Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
'Obtain ProductID from QueryString
Dim ProductID As Integer = CInt(Request.Params("ProductID"))
' Obtain Product Details
Dim products As ASPNET.StarterKit.Commerce.ProductsDB = New
ASPNET.StarterKit.Commerce.ProductsDB()
Dim myProductDetails As ASPNET.StarterKit.Commerce.ProductDetails =
products.GetProductDetails(ProductID)
```

```
' Update Controls with Product Details
desc.Text = myProductDetails.Description
UnitCost.Text = String.Format("{0:c}", myProductDetails.UnitCost)
ModelName.Text = myProductDetails.ModelName
ModelNumber.Text = myProductDetails.ModelNumber.ToString()
ProductImage.ImageUrl = "ProductImages/" & myProductDetails.ProductImage
addToCart.NavigateUrl = "AddToCart.aspx?ProductID=" & ProductID
ReviewList.ProductID = ProductID
AlsoBoughtList.ProductID = ProductID
End Sub
</script>
<html>
<head>
<link href="ASPNETCommerce.css" type="text/css" rel="stylesheet">
 </head>
<body bottomMargin="0" leftMargin="0" background="images/sitebkgrd.gif"
topMargin="0" rightMargin="0" marginwidth="0" marginheight="0">
<table cellSpacing="0" cellPadding="0" width="100%" border="0">
<tr><td colSpan="2">
<ASPNETCommerce:Header id="Header1" runat="server" />
</td></tr><tr>
<td vAlign="top" width=145>
<ASPNETCommerce:Menu id="Menu1" runat="server" />
<img height="1" src="images/1x1.gif" width="145"></td>
<td vAlign="top" align="left">
<table height="100%" cellSpacing="0" cellPadding="0" width="620" align="left"
border="0">
<tr vAlign="top"><td>
<br><img src="images/1x1.gif" width="24" align="left">
<table cellSpacing="0" cellPadding="0" width="100%" border="0">
 <tr><td class="ContentHead">
```

```
<img height="32" src="images/1x1.gif" width="60" align="left"><asp:label
id="ModelName" runat="server" />
<br></td></tr></table>
<table cellSpacing="0" cellPadding="0" width="100%" border="0" valign="top">
<tr vAlign="top">
 <td rowspan="2">
<img height="1" width="24" src="images/1x1.gif">
 </td><td width="309">
<img height="15" src="images/1x1.gif">
<br><asp:image id="ProductImage" runat="server" height="185" width="309" border="0" />
<br>
<br>
<img height="20" src="images/1x1.gif" width="72"><span class="UnitCost"><b>Your
Price:</b>&nbsp;<asp:label id="UnitCost" runat="server" /></span>
<br>
<img height="20" src="images/1x1.gif" width="72"><span
class="ModelNumber"><b>Model Number:</b>&nbsp;<asp:label id="ModelNumber"
runat="server" /></span>
<br>
<img height="30" src="images/1x1.gif" width="72"><asp:hyperlink id="addToCart"
runat="server" ImageUrl="images/add_to_cart.gif" />
</td><td><table width="300" border="0">
<tr>  <td vAlign="top">
<asp:label class="NormalDouble" id="desc" runat="server"></asp:label><br>
</td></tr></table><img height="30" src="images/1x1.gif">
<ASPNETCommerce:AlsoBought id="AlsoBoughtList" runat="server" /></td>
</tr><tr></tr>  </table>
<table border="0">
  <tr><td><img src="images/1x1.gif" width="89" height="20"> </td>
<td width="100%"><ASPNETCommerce:ReviewList id="ReviewList" runat="server" />
  </td> </tr></table></td></tr></table></td></tr></table>
</body>
</HTML>
```

9.3.16. Register.aspx

```
<%@ Page Language="VB" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<script runat="server">
Sub RegisterBtn_Click(ByVal sender As Object, ByVal e As ImageClickEventArgs)
' Only attempt a login if all form fields on the page are valid
If Page.IsValid = True Then
' Store off old temporary shopping cart ID
Dim shoppingCart As ASPNET.StarterKit.Commerce.ShoppingCartDB = New
ASPNET.StarterKit.Commerce.ShoppingCartDB()
Dim tempCartId As String = shoppingCart.GetShoppingCartId()
' Add New Customer to CustomerDB database
Dim accountSystem As ASPNET.StarterKit.Commerce.CustomersDB = New
ASPNET.StarterKit.Commerce.CustomersDB()
Dim customerId as String = accountSystem.AddCustomer(Server.HtmlEncode(Name.Text),
Email.Text, ASPNET.StarterKit.Commerce.Security.Encrypt(Password.Text))
If customerId <> "" Then
' Set the user's authentication name to the customerId
FormsAuthentication.SetAuthCookie(customerId, False)
' Migrate any existing shopping cart items into the permanent shopping cart
shoppingCart.MigrateCart(tempCartId, customerId)
' Store the user's fullname in a cookie for personalization purposes
Response.Cookies("ASPNETCommerce_FullName").Value =
Server.HtmlEncode(Name.Text)
' Redirect browser back to shopping cart page
Response.Redirect("ShoppingCart.aspx")
Else
MyError.Text = "Registration failed:&nbsp; That email address is already
registered.<br><img align=left height=1 width=92 src=images/1x1.gif>"
End If
End If
End Sub
</script>
<html>
<head>
<link rel="stylesheet" type="text/css" href="ASPNETCommerce.css">
</head>
<body background="images/sitebkgrd.gif" leftmargin="0" topmargin="0" rightmargin="0"
bottommargin="0" marginheight="0" marginwidth="0">
<table cellspacing="0" cellpadding="0" width="100%" border="0">
<tr><td colspan="2"><ASPNETCommerce:Header ID="Header1" runat="server" />
</td></tr><tr><td valign="top">
<ASPNETCommerce:Menu id="Menu1" runat="server" />
<img height="1" src="images/1x1.gif" width="145">
</td><td align="left" valign="top" width="100%" nowrap>
<table height="100%" align="left" cellspacing="0" cellpadding="0" width="100%"
border="0"><tr valign="top"><td nowrap><br><form runat="server">
<img align="left" width="24" height="1" src="images/1x1.gif">
<table cellspacing="0" cellpadding="0" width="100%" border="0">
<tr><td class="ContentHead">
<img align="left" height="32" width="60" src="images/1x1.gif">Create a New Account
<br></td></tr></table>
<img align="left" height="1" width="92" src="images/1x1.gif">
<asp:Label id="MyError" CssClass="ErrorText" EnableViewState="false" runat="Server" />
<table height="100%" cellspacing="0" cellpadding="0" width="500" border="0">
<tr valign="top">
<td width="550">
<br>
<br><span class="NormalBold">Full Name</span>
<br><asp:TextBox size="25" id="Name" runat="server" />
<asp:RequiredFieldValidator ControlToValidate="Name" Display="dynamic" Font-
Name="verdana" Font-Size="9pt" ErrorMessage="'Name' must not be left blank."
runat="server"></asp:RequiredFieldValidator>
<br>
```

```
<br><span class="NormalBold">Email</span>
<br><asp:TextBox size="25" id="Email" runat="server" />
<asp:RegularExpressionValidator ControlToValidate="Email" ValidationExpression="[\w\.-
]+(\+[\w-]*)?@([\w-]+\.)+[\w-]+" Display="Dynamic" Font-Name="verdana" Font-
Size="9pt" ErrorMessage="Must use a valid email address."
runat="server"></asp:RegularExpressionValidator>
<asp:RequiredFieldValidator ControlToValidate="Email" Display="dynamic" Font-
Name="verdana" Font-Size="9pt" ErrorMessage="'Email' must not be left blank."
runat="server"></asp:RequiredFieldValidator>
<br>
<br>
<span class="NormalBold">Password</span>
<br><asp:TextBox size="25" id="Password" TextMode="Password" runat="server" />
<asp:RequiredFieldValidator ControlToValidate="Password" Display="dynamic" Font-
Name="verdana" Font-Size="9pt" ErrorMessage="'Password' must not be left blank."
runat="server"></asp:RequiredFieldValidator>
<br>
<br><span class="NormalBold">Confirm Password</span>
<br><asp:TextBox size="25" id="ConfirmPassword" TextMode="Password" runat="server"
/><asp:RequiredFieldValidator ControlToValidate="ConfirmPassword" Display="dynamic"
Font-Name="verdana" Font-Size="9pt" ErrorMessage="'Confirm' must not be left blank."
runat="server"></asp:RequiredFieldValidator>
<asp:CompareValidator ControlToValidate="ConfirmPassword"
ControlToCompare="Password" Display="Dynamic" Font-Name="verdana" Font-
Size="9pt" ErrorMessage="Password fields do not match."
runat="server"></asp:CompareValidator>
<br>
<br><asp:ImageButton id="RegisterBtn" OnClick="RegisterBtn_Click"
ImageUrl="images/submit.gif" runat="server" />
<br></td></tr></table></form></td></tr></table></td></tr>
</table>
</body>
</html>
```

9.3.17.1. ReviewAdd.aspx

```
<%@ Page Language="VB" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<script runat="server">
Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
If Page.IsPostBack <> True Then
' Obtain ProductID of Product to Review
 Dim productID As Integer = CInt(Request("productID"))
' Populate Product Name on Page
Dim products As ASPNET.StarterKit.Commerce.ProductsDB = New
ASPNET.StarterKit.Commerce.ProductsDB()
ModelName.Text = products.GetProductDetails(productID).ModelName
' Store ProductID in Page State to use on PostBack
ViewState("productID") = productID
End If
End Sub
Sub ReviewAddBtn_Click(ByVal sender As Object, ByVal e As ImageClickEventArgs)
' Only add the review if all fields on the page are valid
If Page.IsValid = True Then
' Obtain ProductID from Page State
 Dim productID As Integer = CInt(ViewState("productID"))
' Obtain Rating number of RadioButtonList
Dim _rating As Integer = CInt(rating.SelectedItem.Value)
' Add Review to ReviewsDB.  HtmlEncode before entry
Dim review As ASPNET.StarterKit.Commerce.ReviewsDB = New
ASPNET.StarterKit.Commerce.ReviewsDB()
 review.AddReview(productID, Server.HtmlEncode(Name.Text),
Server.HtmlEncode(Email.Text), _rating, Server.HtmlEncode(Comment.Text))
' Redirect client back to the originating product details page
Response.Redirect("ProductDetails.aspx?ProductID=" & productID)

End If
End Sub
</script>
<html>
<head>
<link rel="stylesheet" type="text/css" href="ASPNETCommerce.css">
</head>
<body background="images/sitebkgrd.gif" leftmargin="0" topmargin="0" rightmargin="0"
bottommargin="0" marginheight="0" marginwidth="0">
<table cellspacing="0" cellpadding="0" width="100%" border="0">
  <tr><td colspan="2">
  <ASPNETCommerce:Header ID="Header1" runat="server" />
</td></tr><tr><td valign="top">
<ASPNETCommerce:Menu id="Menu1" runat="server" />
<img height="1" src="images/1x1.gif" width="145"></td>
  <td align="left" valign="top" width="100%" nowrap>
<table height="100%" align="left" cellspacing="0" cellpadding="0" width="100%"
border="0"><tr valign="top">
<td nowrap>
<br>
<form runat="server">
<img align="left" width="24" src="images/1x1.gif">
<table cellspacing="0" cellpadding="0" width="100%" border="0">  <tr>
<td class="ContentHead">
 <img align="left" height="32" width="60" src="images/1x1.gif">Add Review -
<asp:label id="ModelName" runat="server" />
<br>
</td></tr></table>  <br>
<img align="left" width="92" src="Images/1x1.gif">
<table width="500" border="0"><tr valign="top">
<td><table border="0"><tr><td valign="top" width="550">
<span class="NormalBold">Name</span>  <br>
<asp:TextBox size="20" id="Name" runat="server" />
```

```
<asp:RequiredFieldValidator ControlToValidate="Name" Display="Dynamic" Font-
Name="verdana" Font-Size="9pt" ErrorMessage="'Name' must not be left blank."
runat="server"></asp:RequiredFieldValidator>
 <br>
<br>
<span class="NormalBold">Email</span>
<br>
<asp:TextBox id="Email" size="20" runat="server" />
<asp:RequiredFieldValidator ControlToValidate="Email" Display="Dynamic" Font-
Name="verdana" Font-Size="9pt" ErrorMessage="'Email' must not be left blank."
runat="server"></asp:RequiredFieldValidator>
<br>
<br>
<span class="NormalBold">Rating</span>
<br>
<br>
<asp:RadioButtonList ID="Rating" runat="server">
<asp:ListItem text="Five" value="5" selected="True">
<img src="Images/reviewrating5.gif"></asp:ListItem>
<asp:ListItem text="Four" value="4">
<img src="Images/reviewrating4.gif"></asp:ListItem>
<asp:ListItem text="Three" value="3">
<img src="Images/reviewrating3.gif"></asp:ListItem>
 <asp:ListItem text="Two" value="2">
<img src="Images/reviewrating2.gif"></asp:ListItem>
<asp:ListItem text="One" value="1">
<img src="Images/reviewrating1.gif"></asp:ListItem>
</asp:RadioButtonList>
</td></tr> </table>
<br>
<br>
<span class="NormalBold">Comments</span>
<br>
```

```
<asp:TextBox id="Comment" textmode="multiline" rows="7" columns="60" runat="server"
/>
<asp:RequiredFieldValidator ControlToValidate="Comment" Display="Dynamic" Font-
Name="verdana" Font-Size="9pt" ErrorMessage="'Comment' must not be left blank."
runat="server"></asp:RequiredFieldValidator>
<br>
<br>
<asp:ImageButton ImageURL="images/submit.gif" OnClick="ReviewAddBtn_Click"
runat="server" />
<br>
<br>
</td></tr></table></form></td></tr></table></td></tr></table>
</body>
</html>
```

9.3.18. Search Result.aspx

```
<%@ Page Language="VB" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<%@ import Namespace="System.Data.SqlClient" %>
<script runat="server">
Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
' Search database using the supplied "txtSearch" parameter
Dim productCatalogue As ASPNET.StarterKit.Commerce.ProductsDB = New
ASPNET.StarterKit.Commerce.ProductsDB()
MyList.DataSource =
productCatalogue.SearchProductDescriptions(Request.Params("txtSearch"))
MyList.DataBind()
' Display a message if no results are found
If MyList.Items.Count = 0 Then
ErrorMsg.Text = "No items matched your query."
```

```
End If
End Sub
</script>
<html>
<head>
<link rel="stylesheet" type="text/css" href="ASPNETCommerce.css">
</head>
<body background="images/sitebkgrdnogray.gif" leftmargin="0" topmargin="0"
rightmargin="0" bottommargin="0" marginheight="0" marginwidth="0">
<table cellspacing="0" cellpadding="0" width="100%" border="0">
<tr><td colspan="2">
<ASPNETCommerce:Header ID="Header1" runat="server" /></td>
</tr><tr><td valign="top">
<ASPNETCommerce:Menu id="Menu1" runat="server" />
<img height="1" src="images/1x1.gif" width="145"></td>
<td align="left" valign="top" width="100%" nowrap>
<table height="100%" align="left" cellspacing="0" cellpadding="0" width="100%"
border="0"><tr valign="top">
<td nowrap>
<br>
<asp:DataList id="MyList" RepeatColumns="2" runat="server">
<ItemTemplate>
<table border="0" width="300">
<tr>
<td width="25"></td>
<td width="100" valign="middle" align="right">
<a href='ProductDetails.aspx?productID=<%# DataBinder.Eval(Container.DataItem,
"ProductID") %>'>
<img src='ProductImages/thumbs/<%# DataBinder.Eval(Container.DataItem,
"ProductImage") %>' width="100" height="75" border="0"></a>
</td><td width="200" valign="middle">
<a href='ProductDetails.aspx?productID=<%# DataBinder.Eval(Container.DataItem,
"ProductID")%>'>
<span class="ProductListHead">
```

```
<%# DataBinder.Eval(Container.DataItem, "ModelName")%>
</span>
<br>
</a><span class="ProductListItem"><b>Special Price: </b>
<%# DataBinder.Eval(Container.DataItem, "UnitCost", "{0:c}")%>
</span>
<br>
<a href='AddToCart.aspx?productID=<%# DataBinder.Eval(Container.DataItem,
"ProductID")%>'>
<font color="#9D0000"><b>Add To Cart</b></font></a>
</td></tr></table>
</ItemTemplate>
</asp:DataList>
<img height="1" width="30" src="Images/1x1.gif">
<asp:Label id="ErrorMsg" class="ErrorText" runat="server" />
</td></tr></table></td></tr></table>
</body>
</html>
```

9.3.19. ShoppingCart.aspx

```
<%@ Page Language="VB" %>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Header" Src="_Header.ascx"
%>
<%@ Register TagPrefix="ASPNETCommerce" TagName="Menu" Src="_Menu.ascx" %>
<%@ import Namespace="System.Data.SqlClient" %>
<script runat="server">
Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)' Populate the shopping cart
the first time the page is accessed.
If Page.IsPostBack = False Then
PopulateShoppingCartList()
End If
End Sub
```

```
Sub UpdateBtn_Click(ByVal sender As Object, ByVal e As ImageClickEventArgs)
' Update the Shopping Cart and then Repopulate the List
UpdateShoppingCartDatabase()
PopulateShoppingCartList()
End Sub
Sub CheckoutBtn_Click(ByVal sender As Object, ByVal e As ImageClickEventArgs)
' Update Shopping Cart
UpdateShoppingCartDatabase()
' If cart is not empty, proceed on to checkout page
Dim cart As ASPNET.StarterKit.Commerce.ShoppingCartDB = New
ASPNET.StarterKit.Commerce.ShoppingCartDB()
' Calculate shopping cart ID
Dim cartId As String = cart.GetShoppingCartId()
' If the cart isn't empty, navigate to checkout page
If cart.GetItemCount(cartId) <> 0 Then
Response.Redirect("CreditCardForm.aspx")
Else
MyError.Text = "You can't proceed to the Check Out page with an empty cart."
End If
End Sub
Sub PopulateShoppingCartList()
Dim cart As ASPNET.StarterKit.Commerce.ShoppingCartDB = New
ASPNET.StarterKit.Commerce.ShoppingCartDB()
' Obtain current user's shopping cart ID
Dim cartId As String = cart.GetShoppingCartId()
' If no items, hide details and display message
If cart.GetItemCount(cartId) = 0 Then
DetailsPanel.Visible = False
MyError.Text = "There are currently no items in your shopping cart."
Else
' Databind Gridcontrol with Shopping Cart Items
MyList.DataSource = cart.GetItems(cartId)
MyList.DataBind()
```

```
'Update Total Price Label
lblTotal.Text = String.Format("{0:c}", cart.GetTotal(cartId))
End If
End Sub
Sub UpdateShoppingCartDatabase()
Dim cart As ASPNET.StarterKit.Commerce.ShoppingCartDB = New
ASPNET.StarterKit.Commerce.ShoppingCartDB()
' Obtain current user's shopping cart ID
Dim cartId As String = cart.GetShoppingCartId()
' Iterate through all rows within shopping cart list
Dim i As Integer
For i = 0 To MyList.Items.Count - 1
' Obtain references to row's controls
Dim quantityTxt As TextBox = CType(MyList.Items(i).FindControl("Quantity"), TextBox)
Dim remove As CheckBox = CType(MyList.Items(i).FindControl("Remove"), CheckBox)
' Wrap in try/catch block to catch errors in the event that someone types in
 ' an invalid value for quantity
Dim quantity as Integer
Try
 quantity = CInt(quantityTxt.Text)
 ' If the quantity field is changed or delete is checked
  If quantity <> CInt(MyList.DataKeys(i)) Or remove.Checked = True Then
Dim lblProductID As Label = Ctype(MyList.Items(i).FindControl("ProductID"), Label)
If quantity = 0 Or remove.Checked = true Then
cart.RemoveItem(cartId, CInt(lblProductID.Text))
Else
cart.UpdateItem(cartId, CInt(lblProductID.Text), quantity)
End If
End If
Catch
MyError.Text = "There has been a problem with one or more of your inputs."
End Try
Next
```

```
End Sub

</script>
<html>
<head>
<link href="ASPNETCommerce.css" type="text/css" rel="stylesheet" />
</head>
<body bottommargin="0" leftmargin="0" background="images/sitebkgrd.gif" topmargin="0"
rightmargin="0" marginheight="0" marginwidth="0">
<table cellspacing="0" cellpadding="0" width="100%" border="0">
<tbody>
<tr>
<td colspan="2">
<ASPNETCommerce:Header id="Header1" runat="server"></ASPNETCommerce:Header>
</td></tr><tr><td valign="top">
<ASPNETCommerce:Menu id="Menu1"runat="server"></ASPNETCommerce:Menu>
<img height="1" src="images/1x1.gif" width="145" />
</td><td valign="top" nowrap="nowrap" align="left" width="100%">
<table height="100%" cellspacing="0" cellpadding="0" width="100%" align="left"
border="0"><tbody>
<tr valign="top">
<td nowrap="nowrap">
<br />
<form runat="server">
<img src="images/1x1.gif" width="24" align="left" />
<table cellspacing="0" cellpadding="0" width="100%" border="0">
<tbody><tr>
<td class="ContentHead">
<img height="32" src="images/1x1.gif" width="60" align="left" />Shopping Cart
 <br />
</td></tr></tbody>
</table>
<img height="4" src="images/1x1.gif" width="110" align="left" /> <font color="red">
```

```
<asp:Label class="ErrorText" id="MyError" runat="Server"
EnableViewState="false"></asp:Label>

</font>
<br />
<img height="15" src="images/1x1.gif" width="24" align="left" border="0" />
<asp:panel id="DetailsPanel" runat="server">
<img height="1" src="images/1x1.gif" width="50" align="left" />
<table height="100%" cellspacing="0" cellpadding="0" width="550" border="0">
 <tbody><tr valign="top">
<td width="550">
 <asp:DataGrid id="MyList" runat="server" BorderColor="black" GridLines="Vertical"
cellpadding="4" cellspacing="0" Font-Name="Verdana" Font-Size="8pt"
ShowFooter="true" HeaderStyle-CssClass="CartListHead" FooterStyle-
CssClass="CartListFooter" ItemStyle-CssClass="CartListItem" AlternatingItemStyle-
CssClass="CartListItemAlt" DataKeyField="Quantity" AutoGenerateColumns="false">
<Columns>
<asp:TemplateColumn HeaderText="Product&nbsp;ID">
<ItemTemplate>
<asp:Label id="ProductID" runat="server" Text='<%# DataBinder.Eval(Container.DataItem,
"ProductID") %>' />
</ItemTemplate>
</asp:TemplateColumn>
<asp:BoundColumn HeaderText="Product Name" DataField="ModelName" />
<asp:BoundColumn HeaderText="Model" DataField="ModelNumber" />
<asp:TemplateColumn HeaderText="Quantity">
<ItemTemplate>
<asp:TextBox id="Quantity" runat="server" Columns="4" MaxLength="3" Text='<%#
DataBinder.Eval(Container.DataItem, "Quantity") %>' width="40px" />
</ItemTemplate>
</asp:TemplateColumn>
<asp:BoundColumn HeaderText="Price" DataField="UnitCost" DataFormatString="{0:c}"
/>
```

```
<asp:BoundColumn HeaderText="Subtotal" DataField="ExtendedAmount"
DataFormatString="{0:c}" />


<asp:TemplateColumn HeaderText="Remove">
 <ItemTemplate>
<center>
<asp:CheckBox id="Remove" runat="server" />
</center>
</ItemTemplate>
 </asp:TemplateColumn>
</Columns>
</asp:DataGrid>
 <img height="1" src="Images/1x1.gif" width="350" /> <span class="NormalBold">Total:
</span>
<asp:Label class="NormalBold" id="lblTotal" runat="server"
EnableViewState="false"></asp:Label>
<br />
<br />
<img height="1" src="Images/1x1.gif" width="60" />
<asp:imagebutton id="update" onclick="UpdateBtn_Click" runat="server"
ImageURL="images/update_cart.gif"></asp:imagebutton>
<img height="1" src="Images/1x1.gif" width="15" />
<asp:imagebutton id="checkout" onclick="CheckoutBtn_Click" runat="server"
ImageURL="images/final_checkout.gif"></asp:imagebutton>
<br /></td></tr></tbody>
</table></asp:panel>
</form></td></tr></tbody></table></td></tr></tbody></table>
</body>
</html>
```

9.3.20. GLOBAL.ASAX

```
<%@ Import Namespace="ASPNET.StarterKit.Commerce" %>
<%@ import Namespace="System.Threading" %>
<%@ import Namespace="System.Globalization" %>
<script language="VB" runat="server">


Sub Application_BeginRequest(sender As [Object], e As EventArgs)
Try
If Not (Request.UserLanguages Is Nothing) Then
Thread.CurrentThread.CurrentCulture =
CultureInfo.CreateSpecificCulture(Request.UserLanguages(0))
' Default to English if there are no user languages
Else
Thread.CurrentThread.CurrentCulture = New CultureInfo("en-us")
End If
Thread.CurrentThread.CurrentUICulture = Thread.CurrentThread.CurrentCulture
Catch ex As Exception
Thread.CurrentThread.CurrentCulture = New CultureInfo("en-us")
End Try
End Sub
</script>
```

9.3.21. WEB.CONFIG

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <!-- application specific settings -->
  <appSettings>
    <add key="ConnectionString"
value="server=PYHTON;database=Commerce;uid=sa;pwd=;" />
  </appSettings>
  <!-- forms based authentication -->
  <system.web>
            <compilation debug="true" />
    <pages validateRequest="true" />
    <!-- enable Forms authentication -->
    <authentication mode="Forms">
```

```
        <forms name="CommerceAuth" loginUrl="login.aspx" protection="All" path="/" />
      </authentication>
      <!-- enable custom errors for the application -->
      <customErrors mode="RemoteOnly" defaultRedirect="ErrorPage.aspx" />
      <!-- disable session state for application -->
      <sessionState mode="Off" />
       <globalization fileEncoding="utf-8" requestEncoding="utf-8" responseEncoding="utf-
8"/>
    </system.web>
    <!-- set secure paths -->
    <location path="Checkout.aspx">
      <system.web>
        <authorization>
          <deny users="?" />
        </authorization>
      </system.web>
    </location>
    <location path="OrderList.aspx">
      <system.web>
        <authorization>
          <deny users="?" />
        </authorization>
      </system.web>
    </location>
    <location path="OrderDetails.aspx">
      <system.web>
        <authorization>
          <deny users="?" />
        </authorization>
      </system.web>
    </location>
</configuration>
```

9.3.22. InstantOrder.asmx

```
<%@ WebService Language="VB" Class="InstantOrder" %>
Imports System
Imports System.Web.Services
Imports ASPNET.StarterKit.Commerce
Public class InstantOrder : Inherits WebService
 <WebMethod(Description:="The OrderItem method enables a remote client to
programmatically place an order using a WebService.", EnableSession:=false)> _
Public Function OrderItem(ByVal userName As String, ByVal password As String, ByVal
productID As Integer, ByVal quantity As Integer) As OrderDetails
' Login client using provided username and password
Dim accountSystem As ASPNET.StarterKit.Commerce.CustomersDB = New
ASPNET.StarterKit.Commerce.CustomersDB()
Dim customerId As String = accountSystem.Login(userName,
ASPNET.StarterKit.Commerce.Security.Encrypt(password))
If customerId Is Nothing Then
Throw New Exception("Error: Invalid Login!")
End If
' Wrap in try/catch block to catch errors in the event that someone types in
' an invalid value for quantity
Dim qty As Integer = System.Math.Abs(quantity)
If qty = quantity And qty < 1000 Then
' Add Item to Shopping Cart
Dim myShoppingCart As ASPNET.StarterKit.Commerce.ShoppingCartDB = New
ASPNET.StarterKit.Commerce.ShoppingCartDB()
myShoppingCart.AddItem(customerId, productID, quantity)
' Place Order
Dim orderSystem As ASPNET.StarterKit.Commerce.OrdersDB = New
ASPNET.StarterKit.Commerce.OrdersDB()
Dim orderID As Integer = orderSystem.PlaceOrder(customerId, customerId)
' Return OrderDetails
Return orderSystem.GetOrderDetails(orderID, customerId)
Else
' invalid input
```

```
Return Nothing
End If
End Function

<WebMethod(Description:="The CheckStatus method enables a remote client to
programmatically query the current status of an order in the ASPNETCommerce System.",
EnableSession:=false)> _
Public Function CheckStatus(ByVal userName As String, ByVal password As String, ByVal
orderID As Integer) As OrderDetails
' Login client using provided username and password
Dim accountSystem As ASPNET.StarterKit.Commerce.CustomersDB = New
ASPNET.StarterKit.Commerce.CustomersDB()
Dim customerId As String = accountSystem.Login(userName,
ASPNET.StarterKit.Commerce.Security.Encrypt(password))
If customerId = "" Then
Throw New Exception("Error: Invalid Login!")
End If
' Return OrderDetails Status for Specified Order
Dim orderSystem As ASPNET.StarterKit.Commerce.OrdersDB = New
ASPNET.StarterKit.Commerce.OrdersDB()
Return orderSystem.GetOrderDetails(orderID, customerId)
End Function
End Class
```

CONCLUSION

At the beginning the microsoft .NET is explained and is examined. This project answers the what the .NET , Object orientented in .NET ,Visual Basic.Net ,History of ASP,Differences of ASP.NET and ASP, short view above SQL Server 2000 ,Security in web,Internet Information Server and lastly e-commerce project are studied.

Companies world-wide are establish a basic electronic presence on a global open network, learning from the experience, and gradually becoming more sophisticated in their use of the technologies. While the more advanced levels of electronic commerce present substantial challenges, the more basic levels are now well established and supported by "off the shelf" solutions. The best way of gaining the mastery of electronic commerce that will be vital in tomorrow's markets is to try it today.

Technology is improving,May be one day ,We will be slave of Technology,We are walking to make Human Life better.

RERERANCES

http://www.microsoft.com

http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS

http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/1f74eac5-9005-4f91-9137-f63b73eefde8.mspx

Yüksel Inan,Nihat Demirli, *Visual Basic.NET 2003,*İstanbul,2004

Yüksel Inan,Nihat Demirli,*XML ve ASP.NET* ,İstanbul,2004

Yüksel Inan,Nihat Demirli,*SQL Server ADO.NET* ,İstanbul,2004

Prof.Dr.Mithat UYSAL, *Visual Basic.NET ile yazılım geliştirme,*İstanbul,2004

Douglas Reilly, *Designing Microsoft ASP.NET Applications*, Microsoft Press One Microsoft WayRedmond, WA 98052-6399

Mesbah Ahmed, Chris Garrett, Jeremy Faircloth, Chris Payne, ***ASP.NET** Web Developer's Guide,* 800 Hingham Street Rockland, MA 02370, 2002

Danny Ryan and Tommy Ryan,*ASP.NET Your visual blueprint for creating Web applications on the .NET Framework,* Hungry Minds, Inc. 909 Third Avenue NY 10022,2002

Scott Mitchell, Bill Anders, Rob Howard, Doug Seven, Stephen Walther, Christop Wille, and Don Wolthuis, *Draft,*SAMS, 201 West 103rd St., Indianapolis, Indiana, 46290 USA,2001

Jeff Prosise, *Programming Microsoft .NET,* Microsoft Press, 2002

Miridula Parihar,*ASP.NET Bible*,Hungary Minds,909 Third revenue NY,2002

Jason Butler,Tony Caudill,*ASP.NET Database programming Weekend Crash Course*, Hungary Minds,909 Third revenue NY,2002