ASP.NET

The first release of the .NET Framework 1.0 in early 2002, Microsoft has put a lot of effort and development time into ASP.NET, the part of the .NET Framework that enables you to build rich web applications. This first release meant a radical change from the older Microsoft technology to build web sites called *Active Server Pages* (ASP), now often referred to as *classic* ASP. The introduction of ASP.NET 1.0 and the associated Visual Studio .NET 2002 gave developers the following benefits over classic ASP:

 A clean separation between presentation and code. With classic ASP, your coding logic was often scattered throughout the HTML of the page, making it hard to make changes to the page later.

 a feature-rich development tool (called Visual Studio .NET) that allowed developers to create and code their web applications visually.

 a choice between a number of *object-oriented programming* languages, of which Visual Basic .NET and C# (pronounced as C-Sharp) are now the most popular.

 Access to the entire .NET Framework, which for the first time meant that web developers had a unified and easy way to access many advanced features to work with databases, files, e-mail, networking tools, and much more. Despite the many advantages of ASP.NET over the older model, using ASP.NET also meant an increase of complexity and the knowledge you needed to build applications with it, making it harder for many new programmers to get started with ASP.NET.

After the initial release in 2002, Microsoft released another version of the .NET Framework (called .NET 1.1) and the development IDE Visual Studio .NET in 2003. Many people saw this as a service pack for the initial release, although it also brought a lot of new enhancements in both the framework and the development tools. In November 2005, Visual Studio 2005 and ASP.NET 2.0 were released. To the pleasant surprise of many developers around the world, Microsoft had again been able to drastically improve and expand the product, adding many features and tools that helped reduce the complexity that was introduced with ASP.NET 1.0. **New wizards and smart controls made it possible to reduce the code required to build an application, decreasing the learning curve for new developers and increasing the productivity**. The version, ASP.NET 3.5, builds on top of the successful ASP.NET 2.0 release, leaving many of the beloved features in place, while adding new features and tools in other areas.

What is Classic ASP?

Microsoft's previous server side scripting technology ASP (Active Server Pages) is now often called classic ASP. ASP 3.0 was the last version of classic ASP.

ASP.NET is NOT ASP

ASP.NET is the next generation ASP, but it's not an upgraded version of ASP.

ASP.NET is an entirely new technology for server-side scripting.

ASP.NET is the major part of the Microsoft's .NET Framework.

What is ASP.NET?

ASP.NET is a server side scripting technology that enables scripts (embedded in web pages) to be executed by an Internet server.

- ASP.NET is a Microsoft Technology
- ASP stands for Active Server Pages
- ASP.NET is a program that runs inside IIS
- IIS (Internet Information Services) is Microsoft's Internet server
- IIS comes as a free component with Windows servers
- IIS is also a part of Windows 2000 and XP Professional

Some of the differences between ASP.NET and earlier web development platforms include the following:

• ASP.NET features a completely object-oriented programming model.

• ASP.NET gives you the ability to code in any supported .NET language (including Visual Basic, C#, and many other languages that have third-party compilers).

• ASP.NET is dedicated to high performance. ASP.NET pages and components are compiled before execution, instead of being interpreted every time they’re used.

Important Facts about ASP.NET

Fact 1: ASP.NET Is Compiled, Not Interpreted

One of the major reasons for performance degradation in classic ASP pages is its use of interpreted script code. Every time an ASP page is executed, a scripting host on the web server needs to interpret the script code and translate it to lower-level machine code, line by line. This process is notoriously slow. ASP.NET applications are always compiled—in fact, it's impossible to execute C# or Visual Basic code without it being compiled first. .NET applications actually go through two stages of compilation:

In the first stage, the C# code you write is compiled into an intermediate language called Microsoft Intermediate Language (MSIL), or just IL. This first step is the fundamental reason that .NET can be language-interdependent. Essentially, all .NET languages (including C#, Visual Basic, and many more) are compiled into virtually identical IL code. This first compilation step may happen automatically when the page is first requested. The compiled file with IL code is an assembly.

The second level of compilation happens just before the page is actually executed. At this point, the IL code is compiled into low-level native machine code. This stage is known as just-in-time (JIT) compilation. Figure 1 shows this two-step compilation process.

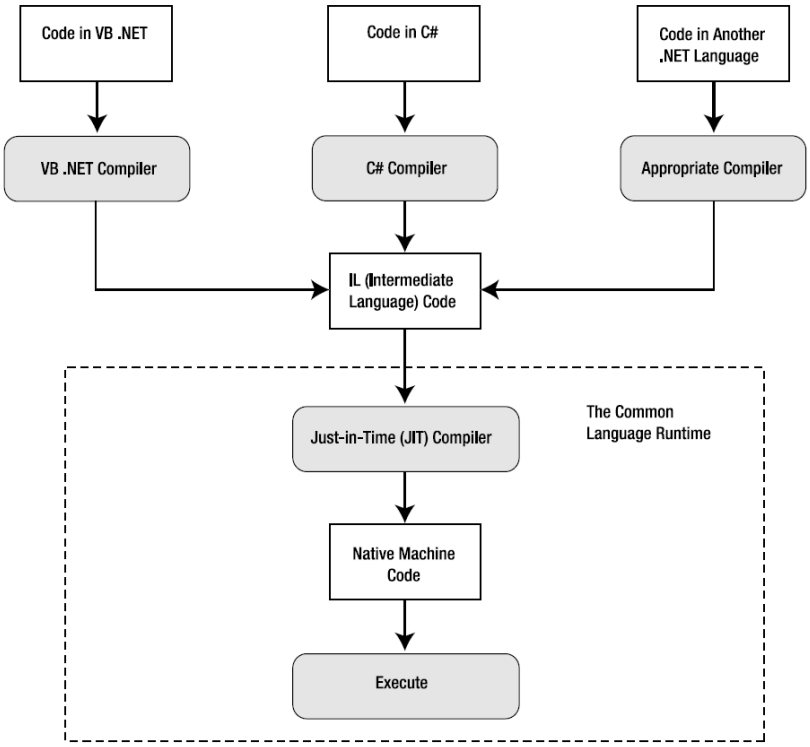

Figure 1: Compilation in an ASP.NET web page

Fact 2: ASP.NET Is Multilanguage

Though you'll probably opt to use one language over another when you develop an application, that choice won't determine what you can accomplish with your web applications. That's because no matter what language you use, the code is compiled into IL. IL is a stepping stone for every managed application.

To understand IL, it helps to consider a simple example. Take a look at this code written in C#:

```
using System;
namespace HelloWorld
{
public class TestClass
{
static void Main(string[] args)
{
Console.WriteLine("Hello World");
}
}

}
```

This code shows the most basic application that's possible in .NET a simple command-line utility that displays a single, predictable message on the console window. Now look at it from a different perspective. Here's the IL code for the Main() method:

```
.method private hidebysig static void Main(string[] args) cil managed
{
.entrypoint
// Code size 13 (0xd)
.maxstack 8
IL_0000: nop
IL_0001: ldstr "Hello World"
IL_0006: call void [mscorlib]System.Console::WriteLine(string)
IL_000b: nop
IL_000c: ret
} // end of method TestClass::Main
```

Fact 3: ASP.NET Is Object-Oriented

ASP.NET is truly object-oriented. Not only does your code have full access to all objects in the .NET Framework, but you can also exploit all the conventions of an OOP (object-oriented programming) environment. For example, you can create reusable classes, standardize code with interfaces and extend existing classes with inheritance. One of the best examples of object-oriented thinking in ASP.NET is found in server-based controls. Server-based controls are the epitome of encapsulation.

Here's a quick example with a standard HTML text box that you can define in an ASP.NET web page:

```
<input type="text" id="myText" runat="server" />
```

With the addition of the runat="server" attribute, this static piece of HTML becomes a fully functional server-side control that you can manipulate in C# code. You can now work with events that it generates, set attributes, and bind it to a data source. For example, you can set the text of this box when the page first loads using the following C# code:

```
void Page_Load(object sender, EventArgs e)
{
myText.Value = "Hello World!";
}
```

Technically, this code sets the Value property of an Html Input Text object. The end result is that a string of text appears in a text box on the HTML page that's rendered and sent to the client.

Fact 4: ASP.NET Is Multidevice and Multibrowser

One of the greatest challenges web developers face is the wide variety of browsers they need to support. Different browsers, versions, and configurations differ in their support of HTML. Web developers need to choose whether they should render their content according to HTML 3.2, HTML 4.0, or something else entirely—such as XHTML 1.0 or even WML (Wireless Markup Language) for mobile devices. This problem, fueled by the various browser companies, has plagued developers since the World Wide Web Consortium (W3C) proposed the first version of HTML. ASP.NET addresses this problem in a remarkably intelligent way.

ASP.NET File?

- An ASP.NET file is just the same as an HTML file.
- An ASP.NET file can contain HTML, XML, and scripts.
- Scripts in an ASP.NET file are executed on the server.
- An ASP.NET file has the file extension ".aspx".

How Does ASP.NET Work?

- When a browser requests an HTML file, the server returns the file.
- When a browser requests an ASP.NET file, IIS passes the request to the ASP.NET engine on the server.
- The ASP.NET engine reads the file, line by line, and executes the scripts in the file. Finally, the ASP.NET file is returned to the browser as plain HTML.

During the processing of the page, three important areas can influence the way the page eventually ends up in the browser:

 Static text: Any static text, like HTML, CSS, or JavaScript code you place in a page, is sent to the browser directly.

 ASP.NET server controls: These controls are placed in your ASPX page and when they are processed, they emit HTML that is inserted in the page.

 Programming code: You can embed code, like Visual Basic .NET or C#, directly in a page. In addition, you can place code in a separate code file, called a Code Behind file. This code can be executed by the runtime automatically, or based on a user's action.

A First Look at ASP.NET Markup

The markup for ASP.NET Server Controls is similar to that of HTML. It also has the notion of tags and attributes, using the same angle brackets and closing tags as HTML does. However, there are also some differences. For starters, most of the ASP.NET tags start with an asp: prefix. For example, a button in ASP.NET looks like this:

<asp:Button ID="Button1" runat="server" Text="Button" />

Note how the tag is self-closed with the trailing slash (/) character, eliminating the need to type a separate closing tag.

You create a new Web Site Project by choosing:

File ➪ New Web Site or File ➪ New ➪ Web Site from Visual Web Developer's main menu.

Choosing the Right Web Site Template

The New Web Site dialog box in VWD contains different web site templates, each one serving a distinct purpose.

Figure 2 shows the New Web Site dialog box in VWD. You can open this dialog box by choosing File New Web Site, or File New Web Site depending on your version of VWD.

The top section of the Templates area shows the ASP.NET web site templates that are installed by default. The second part, labeled My Templates, contains a link to search for templates online. In addition, when you have created your own templates, or have templates installed from other parties, they show up in this area as well.

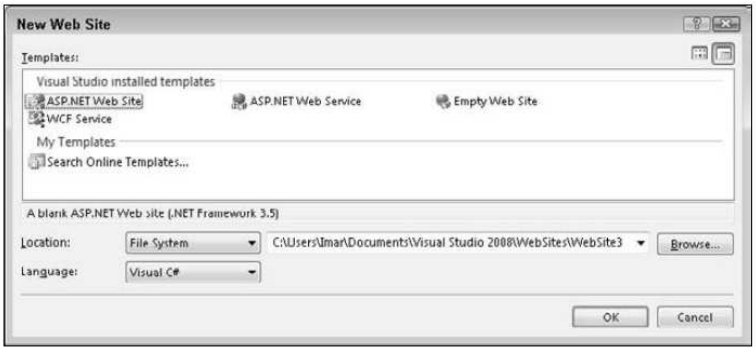

Figure 2: New Web Site Dialog Box in VWD