



# An Analysis of Web Development Tool

by

Mr. Kittikhun Kiattichaiprasop

A Final Report of the Six-Credit Course  
IG 6998 E-Commerce Practicum

Submitted in Partial Fulfillment  
of the Requirements for the Degree of  
Master of Science  
in Internet and E-Commerce Technology  
Assumption University

July 2003

**An Analysis of Web Development Tool**

by  
Mr. Kittikhun Kiattichaiprasop

A Final Report of the Six-Credit Course  
IC 6998 E-Commerce Practicum

Submitted in Partial Fulfillment  
of the Requirements for the Degree of  
Master of Science  
in Internet and E-Commerce Technology  
Assumption University

July 2003



Project Title            An Evaluative Analysis of Interactive Data Retrieval Program

Name                     Mr. Kittikhun Kiattichaiprasop

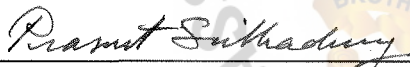
Project Advisor        Rear Admiral Prasart Sribhadung

Academic Year         July 2003


---


The Graduate School of Assumption University has approved this final report of the six-credit course, IC 6998 E-Commerce Practicum, submitted in partial fulfillment of the requirements for the degree of Master of Science in Internet and E-Commerce Technology.

Approval Committee:

  
(Rear Admiral Prasart Sribhadung)  
Dean and Advisor

  
(Prof. Dr. Srisakdi Charmonman)  
Chairman

  
(Dr. Ketchayong Skowratananont)  
Member

  
(Assoc. Prof. Somchai Thayarnyong)  
MUA Representative

July 2003

## ABSTRACT

This is a documentary research project studying ASP and ASP.NET program. The report includes the history of ASP and ASP.NET, what they are used for, and their features. A comparison on ADO and ADO.NET was analyzed. Moreover, it includes a real case study using ASP.NET.

The finding shows that ASP.NET can communicate with the database better than ASP and it can retrieve data faster than ASP.





## ACKNOWLEDGEMENTS

Several people have made contributions to this project. The writer would like to acknowledge their efforts and thank them for their contributions through the making of this project.

He would like to thank Rear Admiral Prasart Sribhadung, his project advisor, for his valuable suggestions and advice given in preparation of this project. Also the writer would like to thank all the members of MS(IEC) committee, Prof.Dr. Srisakdi Charmonman, Dr. Ketchayong Skowratananont.

This project could not have been completed without the assistance and the will power given by his parents in full support. He also wishes to express his acknowledgement to his brothers and aunt for their timely assistance and information provided to him while carrying out the data collection required for his project.

Finally, he would like to extend his gratefulness to all his friends for their support and friendship, thank you.

## TABLE OF CONTENTS

<u>Chapter</u>	<u>Page</u>
ABSTRACT	i
ACKNOWLEDGEMENTS	ii
LIST OF FIGURES	vi
LIST OF TABLES	vii
I. INTRODUCTION	1
1.1 Background of the Project	1
1.2 Objectives of the Project	3
1.3 Scope of the Project	3
1.4 Research Methodology	3
II. ASP OVERVIEW	4
2.1 An Overview of the Technologies	5
2.2 Static Web Pages vs Dynamic Web Page	14
2.3 Simple Web Theory	23
III. ASP HISTORY	32
3.1 Learning from the History of ASP	32
3.2 The ASP Time Line	40
IV. ASP	42
4.1 What Is ASP?	42
4.2 How Does ASP Work?	47
4.3 Advantages and Disadvantages of ASP	49
V. .NET FRAMEWORK	51
5.1 .NET Background and Purposes	51

<u>Chapter</u>	<u>Page</u>
5.2 What Is the Roles of .NET Framework?	51
5.3 What Are Advantages of .NET Framework?	53
5.4 What Is .NET?	53
5.5 Why We Need .NET?	56
5.6 .NET Vision?	60
VI. ASP.NET	64
6.1 What Is ASP.NET?	64
6.2 Weaknesses in ASP2 Model	65
6.3 Weaknesses in the ASP3 Model	66
6.4 The Need for a New ASP Model	66
6.5 Reviewing the Basic of the ASP.NET	74
6.6 Benefits of ASP.NET	75
6.7 ASP.NET Example	77
6.8 How Does ASP.NET Work?	80
VII. ASP VS ASP.NET	83
VIII. COMPARATIVE ANALYSIS: ADO AND ADO.NET	90
8.1 ADO and ADO.NET Overview	90
8.2 What Is ADO?	91
8.3 What Is ADO.NET?	91
8.4 Why We Need Another Data Access Model?	94
8.5 Comparisons of ADO and ADO.NET	97
IX. COMPARATIVE DATABASE CONNECTION BETWEEN ASP AND ASP.NET	100
9.1 Database Connection Forms	100



<u>Chapter</u>	<u>Page</u>
9.2 ASP and ASP.NET Database Connection code	105
X. CASE STUDY	117
10.1 Straight Ahead Ministries	117
10.2 <a href="http://www.netballnorthharbour.co.nz">www.netballnorthharbour.co.nz</a>	119
XI. COMPARATIVE ANALYSIS: TOTAL COST OF ASP AND ASP.NET	124
XII. CONCLUSIONS AND RECOMMENDATIONS	127
12.1 Conclusions	127
12.2 Recommendations	129
BIBLIOGRAPHY	131



## LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
2.1 A Welcome Message Page	15
2.2 Static Web Page	16
2.3 Welcome Page Enhanced	17
2.4 Client-side Dynamic Web Page	20
2.5 Server-side Dynamic Web Page	21
2.6 HTTP Protocol	24
2.7 HTTP Request and HTTP Response	26
6.1 The Result from test.aspx File	79
6.2 HTML Source for test.asp after the Execution	79
6.3 Without ASP.NET	80
6.4 With ASP.NET	81
7.1 Error Message from ASP.NET	85
7.2 Error Message from ASP	86
9.1 OLEDB Architecture	101
9.2 OLEDB Providers and ODBC Drivers	101
9.3 The Result from ASP	108
9.4 The Result form ASP.NET	110
9.5 A Page Modified by ASP	114
9.6 A Page Modified by ASP.NET	115
10.1 Original Site	121
10.2 New Site	121

## LIST OF TABLES

<u>Table</u>	<u>Page</u>
9.1 Provider Names to Connect a Database	104
9.2 Performance Comparison	104
12.1 ASP Hardware Specification	129
12.2 ASP.NET Hardware Specification	129





## I. INTRODUCTION

### 1.1 Background of the Project

Internet is more than a network of computers-it is really a network of networks. Beyond that, Internet is also a network of services and resources, a library, a database, and a community of people from all walks of life ready to answer questions, listen, and share. The Internet is a worldwide computer network, linking more than two million computers. Furthermore, the Internet is estimated to grow by more than 100,000 additional computers every month. Therefore, the Internet applied to commerce-is called "E-commerce"- is a general concept covering any form of business transaction or information exchange executed using information and communication technologies. E-commerce takes place between companies, between companies and their customers, or between companies and public administrations. It includes electronic trading of goods, services and electronic material.

Online commerce is impossible without working on the security and reliability problems. While some are attacking them by taking the sensitive material offline. Many others are using public key cryptography to encrypt sensitive information, to digitally-sign orders and responses, and to make sure that message has been transmitted unmodified such as secure server, digital authentication, encryption, merchant software, and electronic payment software, etc.

Many different approaches can be taken to the problem of transferring money from one individual to another, ranging from existing methods. For examples, bank draft authorization and credit cards, to indirect use of existing methods such as registering your payment information with a third party who processes all transactions, or registering it with the vendor outside of the public network and using an account

number online etc. All the way the use of digital currencies that permit the digital movement of cash endorsed by digital signatures.

To create an effective e-commerce site, the programs are important and necessary. ASP (Active Server Page) program is the one of many programs that can be used to support those tasks such as Dell Online, Barnes and Noble, 1-800-Flowers and the Microsoft site itself. With all of the acronyms floating around the web, ASP might seem like just another development language. On the contrary, it is a hugely powerful database tool, which is in use by most major corporate sites. ASP is primarily a database access language, which allows you to quickly and easily get at information in a database. You can have an inventory page update with the latest items to sell, with current pricing and photographs. You can have your counters track up-to-the-second, creating reports that are truly current. You can create rotating ads, contests, bulletin boards, and more.

ASP.NET, the latest version of Active Server Page, is Microsoft's technology for building dynamic, database-driven Web sites. ASP.NET is a key component to Microsoft's .NET Framework. It brings many great things to the table like, Performance, Flexibility, Simplicity, Manageability, Scalability, Security, Tools and many, many more. This will be one of the technologies that empowers the next generation of the Internet. One of the great things about ASP.NET is if you know Visual Basic or VB script then you already know a lot about ASP. But the cool thing about ASP.NET is that you don't have to use VB program it, you can program in VB, C# and Jscript, plus other languages being incorporated into ASP.NET like Cobol, Perl and more. ASP.NET represents a radical departure from previous versions of Active Server Pages. If you have the program with earlier versions of Active Server Pages and you have not been exposed to the new features of ASP.NET, prepare to be shocked.

## **1.2 Objectives of the Project**

- (1) Studies what ASP and ASP.NET is.
- (2) Compare General issues between ASP and ASP.NET.
- (3) Understand why we need ASP.NET.
- (4) Compare Data Connection ASP and ASP.NET.

## **1.3 Scope of the Project**

- (1) Compare General issues between ASP and ASP.NET.
- (2) Compare Data Connection ASP and ASP.NET

## **1.4 Research Methodology**

This research is using Documentary Research.





## II. ASP OVERVIEW

Since 1996, ASP programmers have faced one upgrade after another, often with no extremely visible advantages until version 3.x-it has been quite a wild ride. Now we have the first significant improvement in ASP programming within our grasp ASP.NET. Our reliance on a watered-down version of Visual Basic has been alleviated now that more powerful version of Visual Basic or the latest version of C++: C#, which is more Web friendly. ASP.NET allows programmers and developers to work with both VB.NET within the same ASP.NET page. .NET itself is a milestone for Microsoft; it marks Microsoft's entry into the "run once, run everywhere" compiler market alongside Java and Ruby. .NET is also notable for its extreme flexibility; unlike the other choices available, .NET allows the programmer to use any number of .NET – compliant languages to create its code and have it run anywhere through the robust .NET Framework. Visual Basic and C++ have undergone changes as well; Visual Basic was already somewhat Web-oriented through its sibling, Visual Basic Script (VBS).

With .NET in general, Visual Basic and VBS are now one and the same. All of the Web-oriented abilities of VBS have been given to Visual Basic and it has received a significant retooling of the language and syntax. Many previous problems, such as poor memory management and object control, have been resolved by the .NET Common Language Runtime (CLR) and internal programming additions, such as the inclusion of the Try/Catch error-handling system and more low-level abilities than before. All in all, Visual Basic can now be called a true programming language.

C++ retained all the aspects that made it a powerful programming language, such as its excellent object control and error-handling techniques, in its new version, C#. It has now gained a very good as well as being more Web-based, a trait that can be

attributed to the .NET Framework and ASP.NET. It is expected that many programmers will still use C# for object control while combining it with Visual Basic's ease of use for GUI and presentation.

This project is meant to show all ASP programmers, new and old, just how powerful ASP.NET now is. Unlike ASP 1.x through 3.x, which worked in Windows 95 through the Personal Web Server tool, you will need at least Windows 2000, all the latest service packs, Internet Explorer 6, IIS 5.x (up to date), and the NET SDK installed.

## **2.1 An Overview of the Technologies**

If we mention old style ASP. Not all of the technologies work in the same way as ASP.NET, but they all allow the user to achieve the same end-result - that of dynamic web applications. If ASP.NET is not an ideal solution to your problems, then you might want to consider these following technologies, taking into account the following questions:

- (1) Are they supported on the platform you use?
- (2) Are they difficult to learn?
- (3) Are they easy to maintain?
- (4) Do they have a long-term future?
- (5) Do they have extra capabilities, such as being able- to parse XML?
- (6) Are a lot of people already using them - are there a lot of tools available?
- (7) Are the supports, skills, and knowledge required use them available?

We are now going to give a quick overview of technologies.

### 2.1.1 Client-Side Technologies for Providing Dynamic Content

Each of these technologies rely on a module (or plug-in) built into the browser to process the instructions. The client-side technologies are a mishmash of scripting languages, controls, and fully fledged programming languages.

#### **JavaScript**

JavaScript is the original browser scripting language, and is not to be confused with Java. Java is a complete application programming language in its own right. Netscape had originally developed a scripting language, known as LiveScript, to add interactivity to their web server and browser range. It was introduced in the release of the Netscape 2 browser. When Netscape joined forces with Sun and in the process, they changed its name to JavaScript. JavaScript borrows some of its syntax and basic structures from Java (which in turn borrowed ideas from C), but has a different purpose - and evolved from different origins (LiveScript was developed separately to Java).

For example, while JavaScript can control browser behavior and content, it isn't capable of controlling features such as file handling. In fact, JavaScript is actively prevented from doing this for security reasons. Think about it: you wouldn't want a web page capable of deleting files on your hard drive, now would you? Meanwhile, Java can't control the browser as a whole, but it can do graphics and perform network and threading functions.

JavaScript is much easier to learn than Java. It is designed to create small, efficient, applications that can do many things, from performing repetitive tasks, to handling events generated by the user (such as mouse clicks, keyboard responses, and so on).

Microsoft introduced their own version of JavaScript, known as JScript, in Internet Explorer 3.0 and has supported it ever since right up to, and including IE6. It



has only minor differences from the Netscape version of the language, although in older versions of both browsers, the differences were originally quite a lot wider.

## **VBScript**

In Internet Explorer 3.0, Microsoft also introduced their own scripting language, VBScript, which was based on their Visual Basic programming language. VBScript was intended to be a direct competitor to JavaScript. In terms of functionality, there isn't much difference between the two, it is more a matter of personal preference - VBScript has a similarly reduced functionality. Visual Basic developers sometimes prefer VBScript because VBScript is, for the most part, a subset of Microsoft's Visual Basic language. However, it enjoys one advantage that makes it more attractive to novice programmers, in that, unlike JavaScript, it isn't case-sensitive and is therefore less fussy about the particulars of the code. Although this "advantage", makes it a lot slower and less efficient.

The biggest drawback is that there isn't a single non-Microsoft browser that supports VBScript for client side scripting. For a short while there were some proprietary plug-ins for Netscape that provided VBScript support, but these never took off. You'll find that JavaScript is much more widely used and supported. If you want to do client-side scripting of web pages on the Internet then JavaScript is the only language of choice. Indeed Microsoft themselves have replaced VBScript in their .NET framework, with VB.NET. VBScript should only be considered when working on Internet pages where it is known that all clients are IE on Windows.

With both of JavaScript and VBScript there is a module, known as a script engine, built into the browser that dynamically processes the instructions, or as it is known in this case.

## **ActiveX Controls**

An ActiveX control is a self-contained program (or component), written in a language such as C++ or Basic. When added to a web page, it provides a specific piece of client-side functionality, such as timer, client authentication, or database access. ActiveX controls are added to HTML pages <object> tag, which is now part of the HTML standard. ActiveX controls can be executed by browser when they are embedded in a web page.

There is a catch. ActiveX controls were developed by Microsoft, and despite being compatible with the HTML standard, they are not supported on any Netscape browser prior to version 6 (which, at time of writing, was still in beta) without an ActiveX plug-in. Without this, they will only function on Internet Explorer. Also, unlike VBScript, ActiveX is able to manipulate items on the user's machine such as the files or Windows registry. For this reason it is very often considered a security risk and is not even allowed through firewalls. Consequently, ActiveX controls still cannot really be considered either a common or a cross-platform way of making your pages dynamic and are failing out of use.

## **Java Applets**

Java is a cross-platform language for developing applications. When Java first hit the Web in the mid 1990s, it created a tremendous stir. The idea is to use Java code in the form of applets, which are essentially Java components that can be easily inserted into web pages with the aid of the <applet> tag.

Java enjoys better functionality than scripting languages, offering better capabilities in areas such as graphic functions and file handling. Java is able to provide these powerful features without compromising security because the applets run in what is known as a sandbox - which prevents a malicious program downloaded from the web

from doing damage to your system. Java also boasts strong database support through JDBC.

Microsoft and Netscape browsers both have built-in Java support via something known as the Java Virtual Machine (JVM), and there are several standard `<object>` and non-standard `<applet>` tags that are used to add Java applets to a web page. These tags tell the browser to download a Java file from a server and execute it with the Java Virtual Machine built into the browser. Of course, this extra step in the web page building phase means that Java applets can take a little while to download, and can take even longer to process once on the browser. So, while smaller Java applets (that provide features such as drop-down menus and animations) are very popular on the Web, larger ones are still not as widespread as scripted pages.

Although the popularity of Java today isn't quite what some people expected, it makes an ideal teaching tool for people wishing to break out into more complex languages; and its Versatility makes it suited for programming web applications.

### **Curl**

A very recent innovation comes from a company partly set up by Tim Berners-Lee (the innovator behind the Web and the HTML language). Curl is another programming language like Java, but unlike Java, where a second file (or more) has to be downloaded with the HTML file, it completely replaces the HTML source and the Java files. It relies on a Curl plug-in having been installed on your browser first, and currently only works on very recent browsers. The advantage are that the download time is faster than Java, and also you don't have to worry about integrating different languages into the page, as Curl is capable of providing the same features as both Java and JavaScript.

Curl is still in the very early stages of development, although the first version has been released, and more details can be obtained at <http://www.curi.com>.

### Sever-Side Technologies for Providing Dynamic Content

Each of these technologies rely on a modular attachment added onto the web server rather than the browser. Consequently, only HTML and any client-side script, are sent back to the browser by the web server. In other words, none of the server-side code is sent back. Server-side technologies have a more consistent look and feel than client-side ones, and it does not take that much extra learning to move between some of the server-side technologies (excepting CGI).

#### CGI

The Common Gateway Interface (CGI) is a mechanism for creating scripts on the server, which can then be used to create dynamic web applications. CGI is a module that is added to the web server. It has been around for quite a bit longer than even ASP, and right now, a large proportion of dynamically created web pages are created using CGI and a scripting language. However, it is incorrect to assume that CGI does the same job as ASP.NET or ASP. Rather, CGI allows the user to invoke another program (such as a Perl script) on the web server to create the dynamic web page, and the role of CGI is to pass the user supplied data to the this program for processing. However, it does provide the same end result - a dynamic web application. You should be aware that CGI has some severe shortcomings:

- (1) It is not easy for a beginner to learn how to program such modules.
- (2) CGI requires a lot of server resources, especially in a multiuser situation.
- (3) It adds an extra step to our server-side model of creating dynamic content: namely, it is necessary to run a CGI program to create the dynamic page, before the page is processed on the server.



What is more, the format in which CGI receives and transmits data means that the data is not easily manipulated by many programming languages, so you need one with good facilities for manipulating text and communicating with other software. The most able programming languages that can work on any, operating system for doing this are C, C++ and Perl. While they can adequately do the job for us, they are some of the more complex languages to learn. Visual Basic does not offer adequate text handling facilities, and is therefore rarely used with CGI.

Despite this, CGI is still very popular with many big web sites, particularly those running on UNIX operating systems. It also runs on many different platforms, which will ensure its continued popularity.

### ASP

Active Server Pages (ASP) is now dubbed "Classic ASP". ASP commonly relied on either of the JavaScript or VBScript scripting languages (although it was also possible to use any scripting language installed on Windows, such as PerlScript) to create dynamic web pages. ASP is a module (the asp.dll file) that you attach to your web server, and it then processes the JavaScript/VBScript on the web server, and turns it into HTML, before sending it into the server, rather than doing it on the browser.

ASP lets us use practically any of the functionality provided by Windows, such as database access, e-mailing, graphics, networking and system functions, and all from within a typical ASP page. However, it is very, very slow performance wise. It is also restricted to using only scripting languages. It can not do all the things that a fully-fledged programming language can. Secondly, languages, being like "junior" versions of full programming languages, took a lot of shortcuts to make the languages smaller. Some of these shortcuts make their programs longer and more complicated than is

otherwise necessary. As we are going to see, ASP.NET rectifies a lot of this by code more structured, easier to understand, and shorter.

## **JSP**

JavaServer Pages (JSP) is a technology that allows you to combine markup (HTML or XML) with Java code to dynamically generate web pages. The JSP specification is implemented by several web servers, as opposed to ASP which is only supported under IIS, and plug-ins are available that allow you to use JSP with IIS 4.0/5.x. One of the main advantages of JSP is the portability of code between different servers. JSP is also very powerful, faster than ASP, and instantly familiar to Java programmers. It allows the Java program to leverage the aspects of the Java2 platform such as JavaBeans and the Java2 libraries. JavaServer Pages isn't directly related ASP, but it does boast the ability to embed Java code into your web pages using server-side tags.

## **ColdFusion**

ColdFusion (<http://www.macromedia.com/software/coldfusion/>) also enables servers to access data as the server builds an HTML page. ColdFusion is a module installed onto your web server. Like ASP, ColdFusion pages are readable by any browser. ColdFusion also utilizes a proprietary set of tags, which are processed by the ColdFusion Server software. This server software can run on multiple platforms, including IIS, Netscape Enterprise Server and Unix/Apache. The major difference is that while ASP.NET solutions are built primarily with programming languages and objects, ColdFusion utilizes HTML-like tags, which encapsulate functionality. A drawback is that the ColdFusion software does not come for free and indeed you could find yourself paying well in excess of a thousand dollars for the privilege of running Cold Fusion on your web server.

## PHP

PHP (originally Personal Home Pages, but more recently PHP HyperText Preprocessor) is another scripting language for creating dynamic web pages. When a visitor opens the page, the server processes the PHP commands and then sends the results to the visitor's browser, just as with ASP.NET or ColdFusion. Unlike ASP.NET or ColdFusion, however, PHP is open-source and cross-platform. PHP runs on Windows NT and many Unix versions, and it can be built as an Apache module and as a binary that can run as a CGI. When built as an Apache module, PHP is especially speedy. A downside is that you have to download PHP separately and go through a series of quite complex steps to install it and get it working on your machine. Also PHP's session management was non-existent until PHP 4, and still is even now, inferior to ASP's even now.

PHP's language syntax is similar to C and Perl. This might prove a barrier to people with no prior programming experience, but if you have a background in either language then you might want to take a look. PHP also has some rudimentary object-oriented features, providing a helpful way to organize and encapsulate your code.

## ASP.NET

So why are you telling me about all these other technologies if we are only going to be learning about ASP.NET you might be wondering? Hopefully you will see a similarity between the technologies, and this will aid your understanding of ASP.NET.

ASP.NET also relies on a module attached to the web server. However, the ASP.NET module (which is a physical file called `aspnet_isapi.dll`) doesn't do all of the work itself, it passes some on to the .NET Framework to do the processing for it.

## 2.2 Static Web Pages VS Dynamic Web Pages

### 2.2.1 What Is a Static Web Page?

If you surf around the Internet today, you will see that there are a lot of static web pages out there. What do we mean by a static web page? Essentially, it is a page whose content consists of some HTML code that was typed directly into a text editor and saved as an .htm or .html file. Thus, the author of the page has already, completely determined the exact content of the page, in HTML, at some time before I user visits the page.

Static web pages are often quite easy to spot; sometimes you can pick them out by just looking at the content of the page. The content (text, images, hyperlinks, etc.) and appearance of a static web page is always the same - regardless of who the page, or when they visit, or how they arrive at the page, or any other factors.

For example, suppose we create a page called welcome.htm for our website, by writing some simple HTML like this:

```
<html>
<head><title>A Welcome Message</title></head>
<body>
  <h1>Welcome</h1>
  Welcome to our humble web site. Please feel free to view our
  <a href="conterts.htm">List of contents</a>. <br><br>
```

If you have any difficulties,. you can <a href=<mailto:webmaster@ksc.au.edu>>  
send e- mail to webmaster</a>

```
</body>
</html>
```

Whenever any client comes to our site to view this page, it will look like this (Figure 2.1). The content of the page was determined before the request was made at the time the webmaster saved the .htm file to disk.



Figure 2.1. A Welcome Message Page.

#### How Are Static Web Pages Served?

Ok, so let us think for a moment about how a static, pure-HTML page finds its way onto a client browser as Figure 2.2:

- (1) A web author writes a page composed of pure HTML, and saves it within an .htm file on the server.
- (2) Sometime later, a user types a page request into their browser, and the request is passed from the browser to the web server.
- (3) The web server locates the .htm page and converts it to an HTML stream.
- (4) The web server sends the HTML stream back across the network to the browser.



- (5) The browser processes the HTML and displays the page.

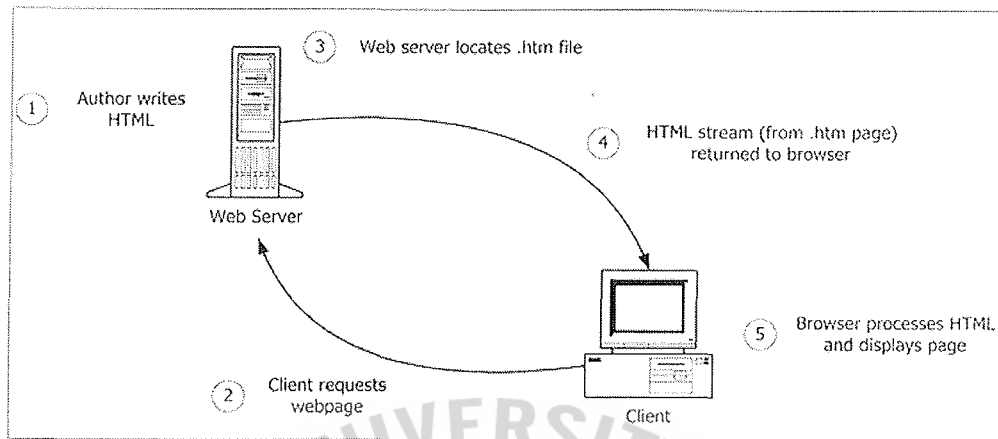


Figure 2.2. Static Web Page.

Static, pure-HTML files like Welcome.htm make perfectly serviceable web pages. We can even spruce up the presentation and usability of such pages by adding more HTML to create frames and tables. However, there is only so much we can achieve by writing pure HTML, precisely because their content is completely determined before the page is ever requested.

#### Limitations of Static Web Pages

For example, suppose we want to enhance our Welcome page - so that it displays the current time or a special message that is personalized for each user. These are simple ambitions, but they are impossible to achieve using HTML alone. If you are not convinced, try writing a piece of HTML for a web page that displays the current time, like Figure 2.3:

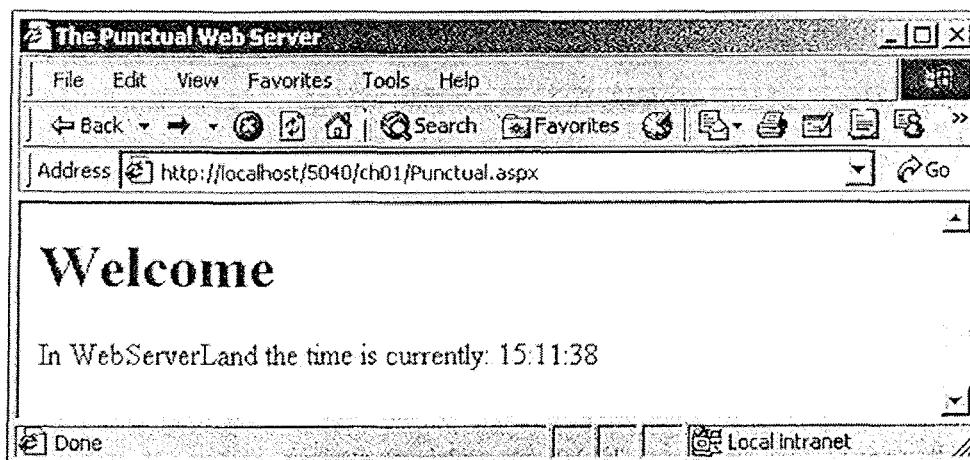


Figure 2.3. Welcome Page Enhanced.

As you type in the HTML, you will soon realize the problem - you know that the user will request the page sometime, but you don't know what the time will be when they do so! Hard-coding the time into your HTML will result in a page that will always claim that the time is the same (and will almost always display the wrong time).

In other words, trying to write pure HTML for a web page that displays the time - but you cannot be sure of the exact time that the web page should display until the time the page is requested. It can't be done using HTML alone.

Also HTML offers no features for personalizing your web pages, each web page that is served is the same for every user. There is also no security with HTML, the code is there for everybody to view, and there is nothing to stop you from copying somebody else's HTML code and using it in your own web page. Static pages might be very fast as quick as copying a small file over a network, but they are quite limited without any dynamic features.

Since we cannot create our page by saving our hard-coded HTML into a file before the page is requested, what we need is a way to generate the HTML after the page is requested. There are two ways of doing this; we will look at them both now.

#### What Is a Web Server?

A web server is a piece of software that manages web pages and makes them available to 'client' browsers - via a local network or over the Internet. In the case of the Internet, the web server and browser are usually on two different machines, possibly many miles apart. However, in a more local situation, we might set up a machine that runs the web server software, and then use a browser on the same machine to look at its web pages. It makes no difference whether we access a remote web server (that is, a web server on a different machine to our browser application) or a local one (web server and browser on the same machine), since the web server's function - to make web pages available to all - remains unchanged. It might well be that you are the only person with access to our web server on your own machine, as would be case if you were running a web server from our home machine. Nevertheless, the principles remain the same.

While there are many web servers available (the commonest ones being Apache, IIS and Iplanet's Enterprise server). This is because it is the only web server that will run ASP.NET. The web server comes as part of the installation for both Windows 2000 and Windows XP. IIS version 5.0 comes with Windows 2000, and IIS version 5.1 with Windows XP; however, there is very little to distinguish the two, and we shall treat them in this chapter as the same product.

#### 2.2.2 What Is Dynamic Web Page?

To fully understand the nature of dynamic web pages, we first need to look at the limitations of what we can and cannot do with a static web page. There are two ways of providing dynamic web page content as:

## Client-Side Dynamic Web Pages

In the client-side model, modules (or plug-ins) attached to the browser do all the work of creating dynamic pages. The HTML code is typically sent to the browser along with a separate file containing a set of instructions, which is referenced from within the HTML page. However, it is also quite common to find these instructions intermingled with the HTML codes. The browser then uses them to generate pure HTML for the page when the user requests the page - in other words, the page is generated dynamically on request. This produces a HTML page, which is sent back to the browser.

So, in this model as Figure 2.4, our set of five steps now becomes six:

- (1) A web author writes a set of instructions for creating HTML, and saves it within an .htm file. The author also writes a set of instructions in a different language. This might be contained within the .htm file, or within a separate file.
- (2) Sometime later, a user types a page request into their browser, and the request is passed from the browser to the web server.
- (3) The web server locates the .htm page, and may also have to locate a second file that contains the instructions.
- (4) The web server sends both the newly created HTML stream and instructions back across the network to the browser.
- (5) A module within the browser processes the instructions and returns it as HTML within the .htm page - only one page is returned, even if two were requested.
- (6) The HTML is then processed by the browser which displays the page.

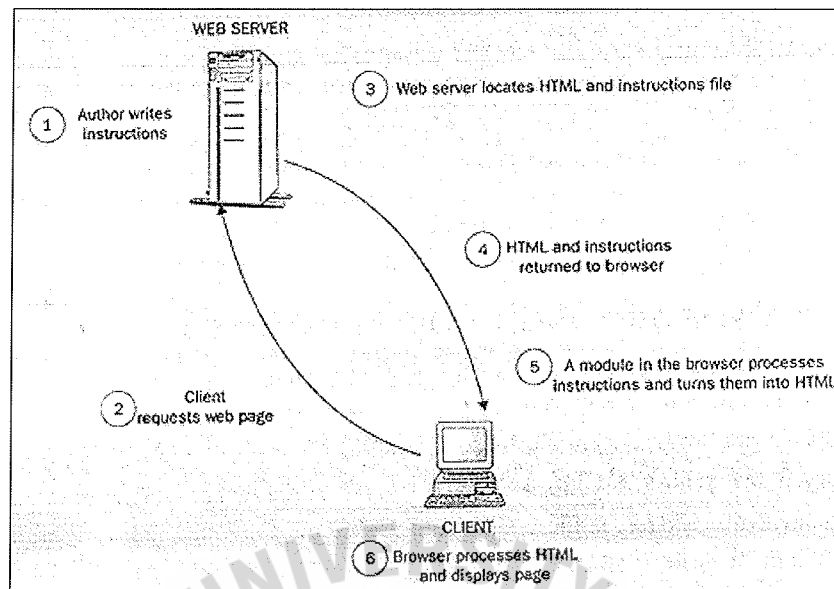


Figure 2.4. Client-side Dynamic Web Page.

Client-side technologies have fallen out of favor in recent times, as they take a long time to download, especially if we have to download a second file with a separate set of instructions. In some cases, we might have to download several files of separate instructions. A second drawback is that each browser interprets these instructions in different ways, so we have no way of guaranteeing that if Internet Explorer understands them, whether Netscape Navigator or Opera will. Another major drawback is that it is a problem to write client-side code that uses server-side resources such as databases, because it is interpreted at client-side. Also all code for client-side scripting is available to everybody, which can be undesirable.

#### Server-Side Dynamic Web Pages

With the server-side model, the HTML source is sent to the web server with an intermingled set of instructions. Again this set of instructions will be used to generate HTML for the page at the time the user requests the page. Once again, the page is



generated dynamically upon request. Our set of five steps once more becomes six, however, with the subtle twist regarding where the processing of instructions is done as Figure 2.5:

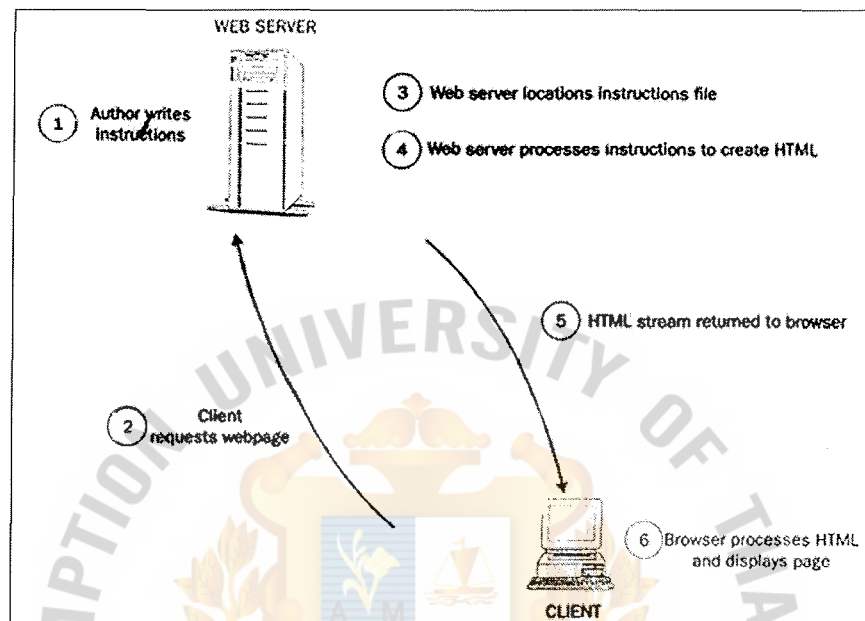


Figure 2.5. Server-side Dynamic Web Page.

- (1) A web author writes a set of instructions for creating HTML, and saves these instructions within a file.
- (2) Sometime later, a user types a page request into their browser, and the request is passed from the browser to the web server.
- (3) The web server locates the file of instructions.
- (4) The web server follows the instructions in order to create a stream of HTML.
- (5) The web server sends the newly created HTML stream back across the network to the browser.
- (6) The browser processes the HTML and displays the page.

The twist is that all the processing is done on the server, before the page is sent back to the browser. One of the key advantages this has over the client-side model is that only the HTML code describing the finished page is actually sent to the browser. This means that our page's logic is hidden away on the server, and that we can safely assume that most browsers should be able to at least have a go at displaying it. ASP.NET as you might have gathered, follows the server-side model.

In fact either process of serving a dynamic web page is only slightly different from the process of – there is just one extra step involved (Step 5 on the client or Step 4 on the both cases this difference is crucial -the HTML that defines the web page is not, page has been requested. For example, we can use either technique to instructions for creating a page that displays the current time:

```
<html>
<head><title>The Punctual Web Servers</title></head>
<body>
  <hl>Welcome</hl>
  In Webserverland, the time is exactly
  <INSTRUCTOM: write code to display the current time>
</body>
</html>
```

In this case, we can compose most of the page using pure HTML. It is just that we cannot hard-code the current time. Instead, we can write a special code (which would replace the italicized line here) that instructs the web server to generate that bit of HTML during Step 5 on the client, or Step 4 on the server, at the time the page is requested.

## 2.3 Simple Web Theory

The web server's main job is to make your web pages available to all and sundry. Another job of the web server is to provide an area (typically in a directory or folder structure) in which to organize and store your web pages, or whole web site.

When you use the Web to view a web page, you will automatically be making contact with a web server. The process of submitting your URL is called 'making a request' to the server. The server interprets the URL, locates the corresponding page, and sends back the code to create the page as part of what is called the response to the browser. The browser then takes the code it has received from the web server and compiles a viewable page from it. The browser is referred to as a client in this interaction, and the whole interaction as a client-server relationship.

### 2.3.1 Client-Server

This term describes the workings of the Web, by outlining the distribution of tasks. The server (the web server) stores, interprets data, and distributes data (that is compiled into web-pages), and the client (browser) accesses the server to get at the data. From now on, whenever we use the term client, we are just referring to the browser.

To understand what is going on in greater detail, we need to briefly discuss how the client and server communicate over the Internet using the HTTP protocol.

### 2.3.2 The HTTP Protocol

The Internet is a network of interconnected nodes. It is designed to carry information from one place to another. When the user tells the browser to fetch a web page, a message is sent from the browser to the web server.

This message is sent using Hypertext Transfer Protocol (or HTTP). HTTP is the protocol used by the World Wide Web in the transfer of information from one machine

to another - when you see a URL prefixed with http://, you know that the internet protocol being used is HTTP.

The message passed from the browser to the web server asking for a particular web page is known as an HTTP request. When the web server receives this request, it checks its stores to find the appropriate page. If the web server finds the page, it bundles up the HTML in an HTTP response, and sends this back across the network to the browser. If the web server cannot find the requested page, it issues a response that features an appropriate error message, and dispatches that page to the browser. Here is an illustration of the process as Figure 2.6:

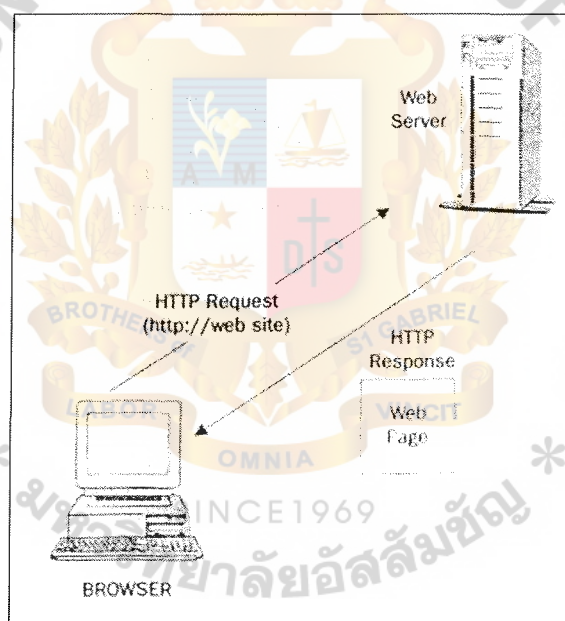


Figure 2.6. HTTP Protocol.

HTTP is known as a stateless protocol. This is because HTTP does not know whether the HTTP request that has been made, is part of an ongoing correspondence, or just a single message. Just as the same way your postman will not know whether your letter is, the first asking your local hi-fi company for a refund on the piece of junk they

sold you, or the fifteenth one penned in giant green capital letters demanding that they give you the refund and a brand new stereo system on top.

The reason HTTP is stateless, is that it was only intended to retrieve a single web page for display. The Internet would be very slow and might even collapse if permanent connections needed to be maintained between browsers and servers, as people moved from one page to another. Think about the extra work HTTP would have to do if it had to worry about whether you had been connected for one minute or whether you had been idle for an hour, and needed disconnecting. Then multiply that by- a million for all the other users. Instead, HTTP makes the connection and delivers the request. And then returned the response and disconnects. The downside of this is that HTTP cannot distinguish between difference requests however, and cannot assign different priorities, so it will not be able to tell whether a particular HTTP request is the request of a user, or the request of a virus infected machine. That has been set up to hit a government web server 1000 times a minute. It will treat all requests equally with the same status, as there are no ways for HTTP to determine where the request originated.

### How HTTP Works

When a request for a web page is sent to the server, this request chains more than just the desired URL. There is a lot of extra information that is sent as part of the request. This is also true of the response - the server sends extra information back to the browser.

The information that is passed within the HTTP message is generated automatically, and the user does not have to deal with it directly, so you do not need to worry about transmitting such information yourself. While you do not have to worry about creating this information yourself, you should be aware that it is being passed between machines as part of the HTTP request and HTTP response. It is because the



ASP.NET code that we write can allow us to have a direct effect on the exact content of this information.

Every HTTP message assumes the same format (whether it is a client request or a server response). We can break this format down into three sections: the request/response line, the HTTP header, and the HTTP body. The content of these three sections is dependent upon whether the message is an HTTP request or HTTP response - so we will take these two cases separately. Let us just pause and illustrate our understanding of the process now:

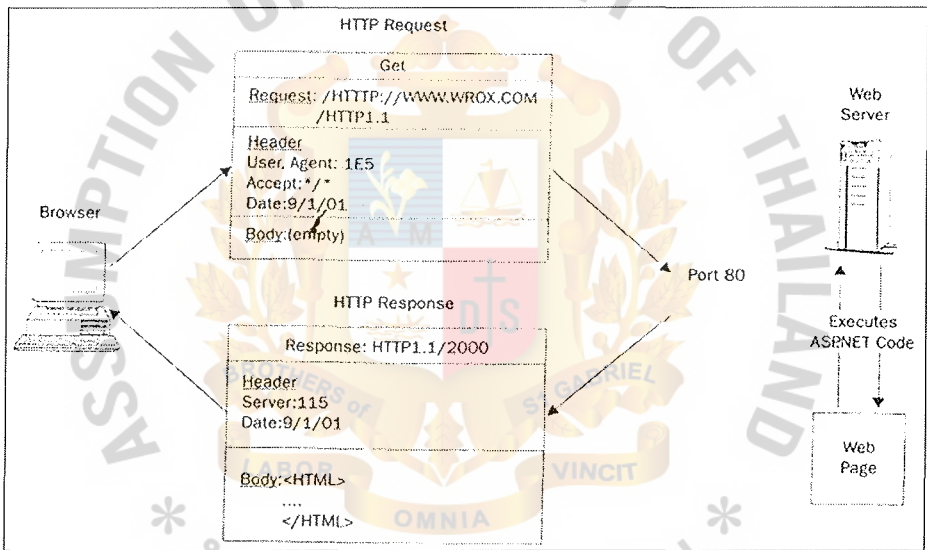


Figure 2.7. HTTP Request and HTTP Response.

We can see that the HTTP request and HTTP response have broadly similar structures, and that there is information common to both that is sent as part of the HTTP header.

There are other pieces of information that can only be known to either the browser or the server, and are only sent as part of either the request or response, so it makes sense to examine their constituent parts in greater detail. These pieces of information,

such as the server name, the date, the acceptance code returned by the server if it finds a web page, are all termed environment or server variables. They are easily readable when using ASP.NET code.

### The HTTP Request

The browser sends the HTTP request to the web server, and it contains the following three listed elements:

- (1) The Request line
- (2) The HTTP header
- (3) The HTTP body

### The Request Line

The first line of every HTTP request is the request line, which contains three pieces of information:

- (1) An HTTP command, known as a method
- (2) The file name, and the path in the server directory structure of the resource that the client is requesting from the server.
- (3) The version number of HTTP

So, an example request line might look like this:

GET/Testpage.htm HTTP/1.1

The method is used to tell the server how to handle the request, and usually consists of a GET or POST command. These basically tell the server to find some particular data. There are a number of other methods supported by HTTP - including PUT, DELETE, TRACE, CONNECT, and OPTIONS.

### The HTTP Header

The next bit of information sent is the HTTP header. This contains details of what document types the client will accept back from the server, like the type of browser that

has requested the page, the date, and general configuration information. The HTTP request's header contains information that falls into three different categories:

- (1) General: contains information about either the client or server, but not specific to one or the other.
- (2) Entity: contains information about the data being sent between the client and server.
- (3) Request: contains information about the client configuration and different types of acceptable documents.

An example- HTTP header might look like this:

Accept: \*/\*

Accept-Language: en-us

Connection: Keep-Alive

Host: www.wrox.com

Referer: <http://webdev.wrox.co.uk/books/SampleList.aspx?bookcode=5040>

User-Agent: Mozilla (XII; I; Linux 2.0.32 i586)

As you can see, the HTTP header is composed of a number of lines; each line contains the description of a piece of HTTP header information, and its value. For example, the user agent line refers to the type of browser that made the request. The accept-language indicates the human-readable language used within the web page; in this case, US English. There are many different lines that can be contained in a HTTP header, and most of them are optional, so HTTP has to indicate when it has finished transmitting the header information. To do this, a blank line is used.

### The HTTP Body

The HTTP request body will contain any data that is being sent to the server - for example, data that the user typed into an HTML. Otherwise, the HTTP request body

will be empty. Data can actually be sent in the URL line (thus, still leaving the request body empty).

### The HTTP Response

The HTTP response is sent by the server back to the client browser, and contains the following three elements:

- (1) The Response line
- (2) The HTTP header
- (3) The HTTP body

### The Response Line

The response line contains only two bits of information:

- (1) The HTTP version number
- (2) An HTTP status code that reports the success or failure of the request

An example response line might look like this:

HTTP/1.1 200 OK

First of all - fall, we can see the HTTP version number - this is not Significant, there are only two versions 1.0 and 1.1 and it just tells the sever which format was used to package up the request. The version number is followed by the status code. This example returns the HTTP status code 200, which represents the message "OK". This denotes the success of the request, and that the response contains the required page or data from the server. Error code values are three-digit numbers, where the first digit indicates the class of the response. There are five classes of response:

- (1) 100-199: These codes are informational - they indicate that the request is currently being processed.
- (2) 200-299: These codes denote success - that the web server received and carried out the request successfully.

- (3) 300-399: These codes indicate that the request has not been performed, because the information required has been moved.
- (4) 400-499: These codes denote a client error-that the request was incomplete, incorrect, or impossible.
- (5) 500-599: These codes denote a server error-that the request appeared to be a valid, but that server failed to carry it out.

### The HTTP Header

The HTTP response header is similar to the request header, which we discussed above. In the HTTP response, the header information again falls into three types:

- (1) General: contains information about either the client or server, but is not specific to one or the other
- (2) Entity: contains information about the data being sent between the client and the server
- (3) Response: information about the server sending the response, and how it can deal with the response

Once again, the header consists of a number of lines, and uses a blank line to indicate that the header information is complete. Here is a sample of what a header might look like, with the name of each line down the side:

HTTP/1.1 200 OK	- the response line
Date: Mon, 1st Nov 1999, 16:12:23 GMT	- the general header
Server: Microsoft-IIS/5.0	- the response header
Last-modified: Fri, 29th Oct 1999, 12:08:03 GMT	- the entity heads

We have already discussed the first line, and the second is self-explanatory. On the third line, server, indicates the type of software the web server is running, and as we



are requesting a file somewhere on the web server, the last bit of information refers to the last time the page we are requesting was modified.

### The HTTP Body

If the request was successful, then the HTTP response body contains the HTML code (together with any script that is to be executed by the browser), ready for the browser to use. Additional HTTP requests are used to retrieve any other resource, such as images, dictated by the HTML code returned after the first request.



### III. ASP HISTORY

#### 3.1 Learning from the History of ASP

You can trace the history of ASP right back to 1995 and the momentous occasion when Microsoft realized they were falling behind in a fundamental shift in the industry by not embracing the Internet. Up until that point Microsoft had been developing their proprietary technologies, tools, and network's protocols for the Microsoft Network; all of a sudden they needed an Internet strategy and fast. Microsoft has gone from a position of playing catch-up to one close to dominance with the Internet Explorer Web browser having a strangle-hold on the Web browsing market, and Internet Information Server (IIS) installed at the majority of Fortune 1000 companies.

##### 3.1.1 The Origins of ASP

Back in the mid'90s, when the commercial Web world was still young, there was not a great deal of choice of tools for the Web developer who wanted to make his or her Web site a truly useful place to do business. The choices were limited in both available server-side programming platforms and also desktop development tools to produce the solutions. In the end, the programmer was stuck with clumsy Common Gateway Interface (CGI) programs using compiled languages such as C, Delphi, and Visual Basic, or interpreted scripting languages like Perl or Rexx, and operating system shell scripts on systems such as UNIX.

In early 1996 Microsoft had a first stab at improving the situation by including the Internet Server Application Programming Interface (ISAPI) technology as part of Internet Information Server. ISAPI is an extension to the Windows Win32 API. It was developed as a way to create Web server software that interacts with the inner

workings of Internet Information Server, bringing what was claimed to be a five-fold increase in performance. As you can well imagine from this description, as well as the immediate performance increase, it also had a side effect of increasing the complexity of the development for the programmer. It was not for the faint hearted, and it takes some serious hardcore programming knowledge to do ISAPI applications right. As well as ISAPI, Microsoft encouraged developers to embrace their Internet Database Connector (IDC) technology. This was a new way to connect Web sites to backend databases through Open Database Connectivity (ODBC).

The ISAPI and IDC technologies lifted Microsoft's youthful and as yet unproven Web server from being a glorified file server to being a basic interactive application server platform for the first time.

Other vendors had tools out there, and several were very popular, such as Netscape Livewire. Livewire was a technology that ran under Netscape's Web server and used a version of JavaScript for page logic, and also used Java components. Unfortunately, Livewire had similar limitations to ISAPI in that it was a compiled technology and the server needed stopping and starting to make changes visible.

### 3.1.2 Why ASP Was Needed?

Not all Web developers have the programming skills needed to write ISAPI applications, and because ISAPI requires the compilation of programs, there are extra steps in producing an ISAPI-based site that slow development down. Novice and intermediate programmers found the need to learn an industrial-strength language, such as C++, and compile even the simplest of their page logic into .dll files a real barrier.

Visual Basic programs, although easier to develop, when used for CGI, performed poorly and the overhead hogged resources. Other languages such as Perl

require the Web server to launch a separate command-line program to interpret and execute the requested scripts, increasing page-load time and reducing server performance. CGI itself hogs resources because every page request forces the Web servers to launch and kill new processes and communicate across these processes. This is time consuming and also uses up precious RAM.

Another problem facing development teams in the mid'90s was the fact that a Web site is a mixture of Hypertext Markup Language (HTML) and logic. They needed a way to mix the programmer's code with the designer's page-layout HTML and designs without one messing up the other. There were many solutions to this problem, ranging from custom template systems to Server Side Include (SSI) statements that told the server to execute code based on special HTML comment tags.

Database-driven interactivity was another challenge. The demand for complex Web sites had just kicked off, and developers needed to supply that demand in a manageable fashion, but the tools available did not make this an easy task. Those who could achieve it demanded rewards that matched the difficulty of what they were being asked to do.

What was needed was a solution for the rest of us. It needed to be a simple scripted text-based technology like Perl, so developers could tweak and alter their pages without compilation and with simple text-editing tools such as Notepad. It needed to have low resource requirements while keeping high performance; therefore it needed to be executed within the server environment just like ISAPI, but without the complexity. Designers and cross-discipline teams demanded that it should include SSI and template features to make integrating page layouts simpler to manage. To be truly popular, it should run off a language that would be easy to pick up and was familiar to a large community of developers.

### 3.1.3 Why ASP Was Not Originally Embraced?

Active Server Pages was not an overnight success, though understandably it did capture the imagination of a large sector of the development community, particularly those already well versed in Visual Basic programming or Visual Basic for applications scripting.

Others who did not have an investment in Visual Basic knowledge found the limitations of Visual Basic, and by extension Visual Basic Scripting, reasons to avoid the technology. Faults included poor memory management, the lack of strong string management abilities, such as Regular Expressions, found in other established languages. When compared to CGI with Perl, ASP was found lacking.

At that time, Internet Information Server was in its infancy, and take-up was low, despite Microsoft's public relations juggernaut going into full flow after the is on to current versions company's much-reported dramatic turnaround. In comparison of the software it seems very poor, but it was still competitive on performance.

Until 1997, back-end Web programming was pretty much owned by CGI and Perl. High-performance Web sites usually had a mix of C-compiled programs for the real business engine, and Perl for the more lightweight form processing.

There was a fair amount of doubt and suspicion around Microsoft's Internet efforts, including IIS and Internet Explorer, and ISAPI had not done all that much to bring across a huge sector of the development community. Despite this uncertain atmosphere, Microsoft saw many Windows NT 4 licenses being bought specifically for Web hosting and development increasing. Third-party support for anything other than small components was initially slow, but, as with all Microsoft products, after the first couple of releases they usually get things right, and ASP was no exception.



Whereas Perl had a huge community of developers led by the heroic figure of Larry Wall, the ASP developer was not yet well supported. A Perl programmer was encouraged from the top to share and make his or her code open. Therefore, the community thrived, with every conceivable solution or library just a few clicks away at the Comprehensive Perl Archive Network (CPAN) site, or at one of the many other web sites and news groups. Contrast this with the ingrained competitive and financially led philosophies of the third-party component vendors in the Windows Distributed Internet Applications (DNA) world. Of course, it did not take the ASP community long to grow to be the loving, sharing success it is now.

#### 3.1.4 Developing ASP 1.x

ASP 1 was an upgrade to Internet Information Server 2, bringing it up to version 3, and was installed as an optional downloaded component. The public beta was first made available in October 1996 and the final release was a factor in IIS quickly overtaking Netscape in the server market.

Around the same period, Microsoft had purchased and further developed a Web site authoring tool called FrontPage that brought with it a new organizational and hosting concept of the FrontPage Web, enabling the developer to deploy Web applications in drag and drop style without using the File Transfer Protocol (FTP). This concept would be carried through into Microsoft Visual Interdev, Microsoft's new HTML and ASP editing environment.

ASP 1 was surprisingly feature-rich for a version 1 product. It included much of the revolutionary functionality ASP that today's programmers take for granted. For example, ActiveX Data Objects that shield the programmer from differences in database implementations, with record sets to easily access and navigate database query results, and the ability to mix and match logic and presentation code in the

same page. Programmers found the limitations of some areas frustrating, for example, options for reading and writing to the file system; but overall, ASP 1 was a breath of fresh air, and many developers quickly and eagerly adopted it.

### 3.1.5 Developing ASP 2.x

Once ASP 1 had settled and become established, Microsoft released a new version of Internet Information Server and an upgrade to ASP, with a combined download called the Windows NT 4 Option Pack. This time, ASP was built in to the Web server setup and was not seen as an extra. The Web server was a big improvement, with better support and functionality all round and the addition of a Simple Mail Transfer Protocol (SMTP) Mail service.

With ASP 2, the technology matured to the point where developers could really implement powerful, large-scale solutions. Big-name companies adopted the Microsoft platform for their high traffic transactional-sites and the technology proved itself time and again against the demands of serving up millions of page views.

From launch, ASP 2 showed improvements across the board, such as increased file system functionality, added components, and language improvements. Third-party developers released components into the market place that filled in every conceivable gap in functionality, and developers were producing their own bespoke components through ASP's Component Object Model (COM)-based architecture.

Developer tools also had upgrades, with Visual Interdev becoming much improved and better integrated into the Visual Studio suite, with access to Visual Source Safe for source control. Third-party tool vendors had also developed their own solutions, with many wizard-style developers' toolkits and integrated environments coming to market, such as the popular Macromedia Ultradev.

Microsoft extended the language code with incremental releases of the language runtime Scripting Engines, allowing for improvements in the languages, such as support for Regular Expressions, without the need for full new versions of Active Server Pages.

### 3.1.6 Major Changes with ASP 2.0

Moving to Active Server Pages 2 brought the developer into a more stable and feature-rich environment. All aspects of the technology were tuned and tweaked, and programmers really felt that things had settled into a stable technology. This new found confidence was in part due to the evidence of successful transactional sites actually showing that the platform could deliver, but also the fact that the technology had been boosted under the hood with tighter integration with Microsoft Transaction Server (MTS). In fact, IIS 4 was rebuilt to be a MTS application, and so ASP and MTS components were actually running in the same processes. Another improvement was the work with Microsoft Message Queue. This allow ASP and components to communicate across networks, ideal for language applications with complex backend requirements, for example, e-commerce systems integrating with existing legacy enterprise resource planning (EPP) infrastructures.

### 3.1.7 Developing ASP 3.0

With the release of windows 2000, Active Server Pages 3 was available. Performance was increased considerably by the addition of a step in the execution of the pages that checked for a previously cached version of the compiled page, and the compiler checking for script elements rather than always processing the page line by line.

The Windows 2000 operating system and features in IIS5 that included the option to selectively separate out Web applications and processes addressed stability issues.

Functionally, it did not have many revolutionary additions (perhaps they were waiting for .NET, which was already on the drawing board at Microsoft), but developers did get several features. They had been asking for, such as server-side redirects to replace the Hypertext Transfer Protocol (HTTP)-header client-side implementation, better error handling, and dynamic includes.

#### 3.1.8 Final Changes to Original ASP Model

With version 3, Microsoft introduced the concept of server scriptlets. These were COM objects that were developed as Extensible Markup Language (XML)-based text files. This enabled programmers to rapidly prototype multi-tiered application business logic without the “change, recompile, upload, stop the server, register, test, change” cycle of component development.

ASP and ActiveX Data Objects (ADO) were given a boost in capability with the addition of XML-processing abilities. XML was, at this point, a massive deal in the developer community, and Microsoft wanted to appear to be fully embracing it, and so the whole of Microsoft’s product line seemed to be receiving an XML makeover.

As well as the new script execution changes mentioned earlier, it included many other performance improvements, such as the ability of the Web server to self-tune, checking adding threads when needed, and having response buffering on by default.

### 3.2 The ASP Time Line

Before looking at ASP.NET, let's briefly take a look at the short but eventful history of Active Server Pages to see how we got to where we are today:

- (1) December 1995: Microsoft makes a dramatic U-turn and announces that their whole product lineup will be refocused to embrace the Internet. Up until this point they had largely ignored the Internet market and had fallen dangerously behind the competition.
- (2) February 1996: Microsoft releases Internet Information Server to the public for free download. Microsoft spokespeople claim that the server offers a four-fold increase in performance over Netscape Netsite server. IIS includes ISAPI and IDC technologies.
- (3) With the release of Windows NT 4, IIS version 2 is bundled, while IIS 1 is available for Windows NT 3.51.
- (4) October 1996: Microsoft releases the public beta for IIS 3 as an optional upgrade to IIS 2. The major change with this version is the inclusion of a new development environment called Active Server Pages, formerly known under its project name of "Denali." As part of their public relations campaign, Microsoft claims they are beating Netscape 2-1 in the server market. IIS no longer supports MIPS and NT 3.51.
- (5) August 1997: Microsoft releases ASP 2 with IIS 4. IIS now includes the Microsoft Management Console (MMC) to make administering the server more straightforward, and the SMTP server is now bundled, having previously been a part of the Commercial package. IIS and ASP are now tightly integrated with Microsoft Transaction Server, and this is seen as a



real step forward in making the platform a credible choice for large-scale deployment.

- (6) 1998-2000: Microsoft started releasing incremental versions of the language Scripting Engines, adding language features and functionality without the need for full ASP version updates, such as the addition of Regular Expressions for VBScript programmers.
- (7) With the release of Windows 2000 with IIS 5, Active Server Pages 3 became available. ASP 3 allowed for server-side redirects, better error support, ADO.2.5 with support for XML, and caching of compiled code. IIS 5 enabled the administrator to finely separate processes to prevent crashing of the server.
- (8) July 2000: .NET makes their first public announcement, revealing their new C# language, promising to deliver better functionality and flexibility than ever before, and promising support for a wide variety of Internet standards.

## IV. ASP

### 4.1 What Is ASP?

ASP stands for Active Server Pages, and it is free! It's built into Windows 2000, and can easily be installed on Windows 95/98/NT, hence the question whether you should say "What Is ASP?" or "What Are ASP?" Active Server Pages represents a new paradigm in computer development.

- (1) Early data programming (pre-database): For a quick recap, the programmers of several decades ago created both the program and the data. The data was in a special file whose very structure was defined by the program. In short, the data could only be read by that specific program. Each program module that accessed the data file would begin by "formatting" the data. It would read a record and then break it down as account number = 10 characters, customer name = 25 characters, etc. If an error in the program had the account number as 9 characters, then the program would assume that the tenth character was actually the first character of the customer name. Errors like these could result in trashing the data. If the customer came along with a request to add a new field or change the size of a field, the programmer would have to write a conversion program to operate on the existing data and then change every single program module that accessed the data. Data integrity had to be maintained by the program. If the program wasn't written properly, you could wind up with missing account numbers or two customers with the same account number.

- (2) Database programming: Somewhere down the line, someone got the bright idea to develop a database external to the program. As soon as the program opened the data file, the database itself defined the fields. The account number could only be ten characters (assuming that the database defined it as such). The program could never read part of one field and part of another by accident. If a field had to be enlarged or a new one added, it was done in the database and all the program modules automatically knew about it. At first, data integrity (missing account numbers, duplicate account numbers) was handled by the program. Eventually, databases improved so that they handled these rules. Once these rules were coded into the database, the programmer didn't have to worry about forgetting these rules in the program code. The database itself would issue a warning if these rules were violated.
- (3) Early Networks (pre Client-server): Early programming was done on mainframes and minis. Here the user sat at a "dumb terminal" and all the processing took place at the CPU (central processing unit) which was housed in the mainframe. The CPU was an expensive piece of equipment designed to handle many user requests. But then PCs and PC Networks proliferated. The central CPU where the data resided (the server) was not really designed to handle multiple requests simultaneously. So PC servers were mostly file servers, where their purpose was to keep the files in a central area accessible to many users. But the actual processing took place at the local user's machine. So when a user requested to see all clients who had an outstanding balance, the server sent the entire client file over to the user's machine, where the local CPU filtered the data to show only

those with an outstanding balance. Transmitting all these unnecessary records over the LAN wasted a good deal of time.

What we really needed was to do the same with program code as we did with the data. Three-tiered architecture, with languages such as ASP, is the result. Instead of just the database server, there is now a database server and an application server. The users communicate with the application server. The application server makes requests of the database server, creates the code for the page the user needs to see and sends just that page to the user. The only application that resides on the user's machine is the browser needed to communicate with the application server and view the pages being received. Browsers are fairly standard and don't get updated as frequently as a custom application. If the application is updated, the user gets the new version as soon as he hits refresh or reloads the browser. Distributing the application is also easier. There is usually no code to install on the user's machine. The application server is connected to the Internet or Intranet. The database server is connected. The user is given the URL to the application server. Anything that needs to be installed on the user's machine is "pushed" to the user by the application.

Active Server Pages (ASP) can be defined as code that can be put into an html page to make it respond dynamically to user requests. ASP allows you to take advantage of server-side scripting. Furthermore, ASP provides an array of objects and components that manage the interaction between the browser and the web server.

Scripting languages such as VBScript and JScript are used to manipulate these objects. ASP is not actually a language in itself. Meaning there is no ASP code, but VBScript or JScript or whichever scripting language you decide you want to use. VBScript is the most widely used language for ASP.

ASP is written in a mixture of VBScript (server side) and JavaScript (client side). If the server is able to decode it, it can replace ASP code with HTML code and send back to the user a plain HTML page. If the user clicks view-source, he won't see the ASP code, he will see the HTML code that the ASP code produced. For an example, view our online catalog from the link on the left. The ASP page opens a database of our products, formats an html page and displays the items. It turns the item code into a clickable link to another ASP page. When you click the link, that ASP page formats another HTML page with information about the specific item. To more understand ASP, it's important to know the difference between ASP and regular HTML pages.

In a HTML page, everything on it is static (i.e. it's just a page that the server takes out from it's 'filing cabinet' and sends to the user's browser).

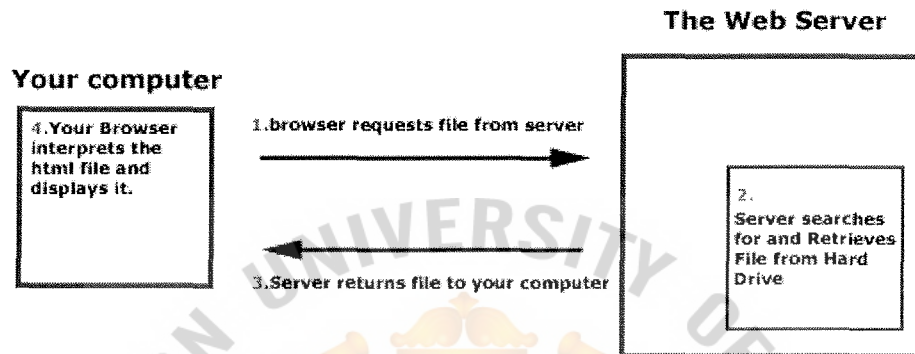
In an ASP, it is dynamic. For example, the webmaster actually typed into the page is a set of instructions for the server to go and fetch, gather or calculate some kind of data. And then compile it in a nice orderly way and 'write' all the information into html and send the page back to the user's browser.

The internet user brings up a web page which has the extension .asp. When the browser requests the ASP file from the web server, instead of processing the page like a normal .html or .htm page, the ASP is processed on the server. ASP processes the requested file from top to bottom, executing any script commands contained in the file, and produces pure 100% HTML code. The resultant html is then sent back to the browser for the user to view. Because your script runs on the server, the Web server does all of the processing and standard HTML pages can be generated and sent to the browser.



The good news is that ASP pages are thus browser independent, so a web page with .asp extension can be viewed by all browsers. Have a look at the Figure 4.1 below. They are pretty simplistic and self-explanatory.

#### How a Html page is displayed



#### How an Asp page is displayed

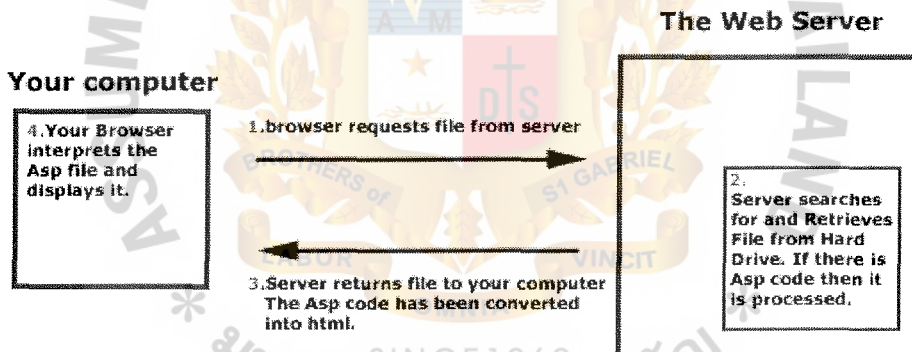


Figure 4.1. HTML vs ASP Processes.

As a webmaster, what you type into your .asp page is actually mostly Scripts (this is then the instruction mentioned above), usually in *Visual Basic Script* (*VBScript*) or *JScript* and some html. Mostly, webmasters use ASP so they can do some cool stuff and especially, use information from a Database in their web site.

At no time did you have to download any application to connect to our database. ASP sent it all to you in a format your browser could handle. The webmasters longer have to create a new page every time they add a product to their line. They no longer have to review their pages to make sure it reflects the latest pricing. All it takes is an entry in our database and ASP does the rest. Most people believe that is much faster to create fully functional service side scripts in ASP then say CGI. You are able to connect to a database with just a few lines of code or mix and match ASP and HTML. Another big advantage is the flexibility of how you write the code. Most people use VB SCRIPT or JSCRIPT. For example is as:

```
<HTML>
<TITLE>My First ASP!</TITLE>
<BODY>
<CENTER>My First ASP!
<BR>
<% Response.Write("Welcome to ASP World!") %>
<BR>
All done!</CENTER>
</BODY>
</HTML>
```

Any web pages containing ASP cannot be run by just simply opening the page in a web browser. The page must be requested through a web server that supports ASP, this is why ASP stands for Active Server Pages, no server, no active pages.

#### **4.2 How Does ASP work?**

Instead of creating HTML documents (.html or .htm), you create ASP documents (.asp). In fact, you can rename your HTML documents to ASP and they

will continue to work exactly as expected. Typically, ASP code and HTML code are both intermingled in the same ASP document. This is somewhat confusing for developers new to ASP, but it quickly becomes understood.

So how does ASP work? Let's assume you have a web server setup and that it is running IIS and that you have both HTML and ASP documents on your web site. Lets now assume someone visits your web site. When a request to view an HTML document (.html or .htm) is made by a browser visiting your web site, the IIS running on your web server simply loads the requested HTML document and sends it to the browser that made the request. When a request is made for an ASP document (.asp), the IIS managing your web server does not simply load and send the document to the browser making the request. Instead, IIS processes the requested ASP document, looking to "run" or "execute" any embedded ASP script code it finds. This script code can be written in VBScript or JavaScript.

In summary, any document with an ASP extension is loaded and processed differently by IIS than an HTML document. An ASP file can have no ASP code present, or it can have lots of ASP code in it. It doesn't matter. Because the document has a .asp extension, IIS automatically loads and scans the document looking to run script information. Keep in mind that all this processing is done on the web server itself, and not on the client computer. The end result is that the web server is responsible for processing ASP files and delivering HTML content back to connected web browsers.

Another cool thing about ASP is that only HTML code is sent back to the web browser. This means that your ASP source code (i.e. VBScript and JavaScript) is never sent to the browser so nobody can see it! This is done for you automatically by

IIS on the web server. The embedded ASP script is ripped out by IIS as it is processed.

### 4.3 Advantages and Disadvantage of ASP

Advantages of ASP are:

- (1) Script Language-independent: ASP allows to use VBScript, Jscript or Perl (“PerlScript”) and have it executed on the server without having to learn another scripting language.
- (2) Utilize COM components from your web server - reuse any functionality built using COM components for your company’s software product can on your website through ASP pages. ASP is the connection between the conventional software and the web site.
- (3) ODBC links to any data source.
- (4) You can use server-side Active components.
- (5) No worries about installation, maintenance, upgrading and management of software, which enables companies to concentrate on their core activities.
- (6) IT maintenance and management costs are reduced because large in-house computers are no longer required.
- (7) ASP allows has the Header () function in one file that every page on the site includes. If we want to change the page header, we change only one file and other page will be included.

Disadvantages of ASP:

- (1) Only on Windows that has IIS or personal web server program.
- (2) SLOW! Comparable ASP code even on a dual-processor 600 MHz Pentium III. A server will execute significantly slower than a PHP solution on a 133 MHz single processor machine.
- (3) No compilation - but compilers do find errors. Debugging is trial-and-error.





## **V. .NET FRAMEWORK**

### **5.1 .NET Background and Purposes**

.NET was introduced to the public in July 2000 at Microsoft's Professional Developers Conference. This technology had been in development for more than two years, under very heavy wraps. We had seen various aspects of what was to become .NET (at that time called "Next Generation Windows Services") at different times in the preceding year. .NET makes our job as developers quite a bit easier for a multitude of tasks.

Microsoft .NET represents a revolution in application development-not just for Web application development, but for Windows applications as well. Moving information from anywhere to anywhere is the basic message of .NET. This means that information should be able to flow from a mainframe to a phone or a wireless device and anything in between. The key to making this information flow possible is Microsoft .NET's heavy reliance on standards-based protocols and formats, such as XML and SOAP. Another key factor is that .NET has been specifically designed with the Internet in mind.

To make the .NET vision a reality, companies must make many changes not just in technology, but also in philosophy. It can be a challenge for corporations to fully grasp the .NET vision, despite the many attempts to explain and demonstrate the different scenarios in which .NET is useful.

### **5.2 What Is the Roles of .NET Framework?**

Microsoft wants to create the standard to every devices connections, so all devices can communicate each other without installing any driver program. This technology is called ".NET Framework". It is not operating system. It is a program

and Microsoft has the future plan for deploying this technology to be installed all devices and causes all are one system. .NET Framework has 3 classes as:

- (1) Programming Language: It is the language form that can be implemented .NET and they are:
  - (a) C#: it is the new language that Microsoft develops it from C++ and JAVA.
  - (b) VB.NET: it comes from Visual Basic version 6.0 development.
  - (c) JScript.net: Microsoft develops it from JScript that is JavaScript.
- (2) Base Classes Library: It is like a set of command that you can apply to it by writing it on the code program. Most of commands are usually used by a programmer. This makes the program developer creates them for any convenient. Library language is formed as “include”. For ASP, the library language is the component. .NET will have the basic library language, therefore a programmer can use any program language development by taking the basic library language on the code. If we do not have .NET, each program language has its own library and so we have to change the code from one language to other one.
- (3) Common Language Runtime(CLR): It is the most important language of .NET because CLR has the responsibility for making the various programs to become standard form. It is called “Intermediate Language(IL)”. When we run any program, CLR will evaluate the status of a computer that we use it and CLR will compile it to become the appropriate program for that computer. Therefore, we can apply to any program on any computer.

### 5.3 What Are Advantages of .NET Framework?

The benefits and advantages of .NET Framework are brief as:

- (1) Library Standard form: According to library standard form, we do not worried about a program language has any library. We can use any library for any program language.
- (2) Independence OS: Many organizations or many personal computer users implement the difference operating system, there is no problem on .NET framework. We can still use any program on difference operating system.
- (3) Any language development: We do not need to studying the new language for developing the program language and we can use any program based on your skill or knowledge.
- (4) Good execution environment controls: According to standard system, it can manage the system such as memory management, etc. This gains the computer has the good performance and reduce the computer hang.
- (5) High Security: .NET has the ability to determine the right's usage or permission of each user which a part of program can be accessed.

### 5.4 What Is .NET?

In fact, .NET is a catchall term that embraces Microsoft's core strategy, plans, and vision for the foreseeable future. At the heart of this strategy is the .NET Framework, which provides the core. The Framework itself consists of several components, of which ASP.NET is just one.

.NET is designed to help solve many fundamental problems faced by programmers. It takes care of a great deal of the hard work involved in building large, reliable applications. It also blurs the line between writing applications to run locally on your own machine and writing applications that can be accessed over the

Web. What's more, it doesn't bring with it all the overheads traditionally associated with "simple" programming frameworks - that is, we don't need to write complex code in a high-powered language to get some fairly impressive speed out of our .NET programs.

We can break down our discussion of the .NET Framework into a few specific topics:

- (1) MS Intermediate Language - all the code we write is compiled into a more abstract before it's executed.
- (2) The Common Language Runtime (CLR) - this is a complex system responsible for executing the code on the computer. It takes care of all the tasks involved in talking to Windows and IIS.
- (3) The .NET Framework Class Libraries - these are code libraries containing a mass of tremendously useful functionality, which we can very easily bolt into our own applications to make complex tasks much more straightforward.
- (4) The .NET Languages - these are simply programming languages that conform to certain specific structural requirements (as defined by the Common Language Specification), and can therefore be compiled.
- (5) ASP.NET - this is how the .NET Framework exposes itself to the Web, using IIS to manage simple pages of code, so that they can be compiled into full .NET programs. These are then used to generate HTML that can be sent out to browsers.

There are the main pieces of .NET:

- (1) The .NET Vision - the idea that all devices will some day be connected together by a global broadband network (that is, the Internet), and that software will become a service provided over this network.
- (2) The .NET Framework - new technologies like ASP.NET that makes .NET more than just a vision, providing concrete services and technologies so that developers can build applications to support the needs of users connected to the Internet today.
- (3) The .NET Enterprise Servers - server products like SQL 2000 and BizTalk 2000 that are used by .NET Framework applications, but are not currently written using the .NET Framework. All future versions of these server products will support .NET, but will not necessarily be rewritten using .NET.

For developers, another important piece of the .NET platform is of course developer tools. Microsoft also has a major new update of Visual Studio called Visual Studio .NET that is the premier development environment for .NET. However, you can still develop .NET applications using Notepad or any other tools you prefer, which is what a lot of the Microsoft development teams do.

In summary, .NET means that all devices on the world will be connected like the net. They have the ability to communicate together such as not only we can play the Internet from any where as a computer, a notebook, a mobile phone etc. but also we can play it on TV, a refrigerator, etc.



## 5.5 Why We Need .NET?

The reasons why we need .NET are:

### 5.5.1 .NET - A Clean Start

When programming applications for Windows platform today, there are a myriad of programming languages and technologies that we can use. Depending on what programming language you choose, the technologies available are typically very different, and can often be restricted. For example, a C/C++ programmer who has to write a GUI Application can either use the Microsoft Foundation Classes (MFC), the Windows Template Library (WTL), or the lower level WIN32 APIs. A VB programmer has to use the VB forms package. This approach's problems are:

- (1) Microsoft spends more time developing two or more competing technologies, rather than focusing on improving one shared technology.
- (2) The availability of so many technologies doing the same thing confuses people.
- (3) Multi-faceted developers who know multiple languages have to learn multiple technologies to achieve the same results.
- (4) Companies have to predominantly invest in one language, since cross-training can be time consuming and expensive.
- (5) Not all languages will necessarily expose all of the same functionality, or be as productive as each other. For example, with C/C++ and MFC today we can easily write applications with toolbars and windows. With VB, we have to buy a third-party package or write the functionality ourselves.

With .NET there is now just one clean object-oriented way of accessing the functionality of the .NET Framework and building applications. All the best and most

commonly used features of existing technologies have been merged together into a single framework.

For example, when developing GUI applications you use Windows Forms. Windows Forms is a consistent GUI framework that exposes the same set of classes to any language supported by the .NET Framework. All languages typically also have the same Visual Designers. This makes the development of GUI applications simple. You use one technology, and it does not matter what language you use. The same simplicity also applies to building web applications using ASP.NET. No longer you have to choose between writing VB Web Classes, ISAPI Extensions, or ASP, so you just use ASP.NET. It provides all of the features application developers need, and again all languages are equal and can access exactly the same functionality.

Of course, there are some downsides to .NET. If you're writing applications that require absolute performance, such as real-time applications, or applications like SQL Server, .NET v1.0 might not be the platform for you. Although .NET has huge benefits to the average application developer, it simply doesn't have the raw performance of a well-written C/C++ application, although certain aspects of a .NET application (such as memory allocation) are faster than C/C++. For reasons of performance and investment, many Microsoft teams will not be rewriting their applications using .NET, instead they will be .NET enabling them. For example, the next major release of SQL Server will enable you to write stored procedures using .NET languages like VB and C#.

#### 5.5.2 No More Language Functionality Debates

If you have programmed using VB before, no doubt you have been aware that C/C++ is a much more powerful environment for low-level development, and suffers from far fewer limitations than VB. With .NET, all programming languages are first-

class citizens. This means you can implement solutions in a programming language that your developers are productive with, without any penalties. With version 1.0 of .NET there will be four languages shipped by Microsoft:

- (1) VB.NET
- (2) C#
- (3) jScript.NET
- (4) MC++

There are no significant technical difference factors between these languages; so again, it's a matter of personal preference and/or company benefits. One caveat to note is that some languages may perform marginally better (about 5%) than others. The C# is marginally faster than VB, and MC++ (Managed C/C++) is faster than C# since it optimizes the output it creates. At the end of the day performance really comes down to the abilities of the compiler writers to generate good code, and how long the compiler has been under development for.

If performance is crucial to your applications, you may want to do some basic performance testing before choosing a language. Eventually, one would assume all languages would be as performance as each other. You can deploy the language that will give you the most productivity.

Anything that we can do from within a .NET class we can do in an ASP.NET page. This means that rather than having to always use components to develop our web application, we now have the choice of when, how and if we use them. This flexibility in ASP.NET stems from the fact that all ASP.NET pages are converted into classes and compiled into a DLL behind the scenes. Of course, just because we now have this new-found flexibility, doesn't mean we should be silly and forget

everything we've learnt in the past. We should still use components to encapsulate data access and other common functionality used in our applications.

### 5.5.3 Multiple Platform Support

.NET has been designed with multiple platform support as key feature. For version 1.0 of .NET, this means that code written using the .NET Framework can run on all versions of Windows: Windows 95, 98, 98SE, Windows NT, Windows 2000, Windows XP, and so on. Depending upon the class libraries used, the same code will also execute on small devices under operating systems like Windows CE, which will run a special compact edition of .NET. However, unlike Java, .NET does not promise that all classes will work under all platforms.

Rather than restricting the class libraries available in .NET to cover functionality that's only available on all platforms, Microsoft has included rich support for all platforms. As developers, it's down to us to make sure we only use .NET classes supported on those platforms (although it's expected that Microsoft will provide tools to help with this process). An exciting prospect for companies is that the .NET code you write today will also work under 64-bit versions of Windows without change.

Targeting multiple platforms with .NET does introduce the potential -for well-known Java problems to hit companies. Since the code is being compiled dynamically on different platforms, the compilation process will result in different native code.

Looking forward, it is expected that .NET will run under other platforms like UNIX, although it is unlikely that the whole of the .NET Framework will be supported, probably just the languages and the base class libraries.

#### 5.5.4 Performance

Since day one, an important design goal for .NET has been great performance and scalability. For .NET to succeed, companies must be able to migrate their applications, and not suffer from poor performance due to the way code is executed by the CLR (Common Language Runtime). To ensure optimal performance, the CLR compiles all application code into native machine code at some point. This conversion can either be done just in time as an application runs (on a method-by-method basis), or when an application is first installed. The compilation process will automatically make use of the microprocessor features available on different platforms, something traditional Windows applications could never do, unless you shipped different binaries for different platforms.

With the first version of ASP.NET you can expect well-written web applications to run two to four times faster than equivalent ASP applications, with similar gains in the area of scalability.

#### 5.6 .NET Vision?

For years now Microsoft has been investing heavily in the Internet, both in terms of product development, technology development, and consumer marketing. You can think of any Microsoft product or technology that isn't web? You can think of any marketing material Microsoft has released that isn't Internet-centric. The reason for this Internet focus is that Microsoft are betting their future on the success of the Internet and other open standards such as XML succeeding and gaining widespread adoption. They are also betting that they can provide the best development platform and tools for the Internet in a world of open standards.

The .NET Framework provides the foundations and plumbing on which the Microsoft .NET vision is built. Assuming the .NET vision becomes reality, one day



very soon the whole world will be predominantly Internet enabled, with broadband access available just about anywhere, at any time. Devices of all sizes will be connected together over this network, trading and exchanging information at the speed of light. The devices will speak common languages like XML over standardized or shared protocols such as HTTP, and these devices will be running a multitude of software on different operating systems and devices. This vision is not specific to Microsoft, and many other companies like IBM and Sun have their own spin on it.

The .NET Framework provides the foundation services that Microsoft sees as essential for making their .NET vision a reality. It's all well and good having a global network and open standards like XML that make it easier for two parties to exchange data and work together, but history has shown that great tools and technologies that implement support for standards are an important ingredient in any vision. Marketing dribble alone doesn't make applications, great developers with great tools and a great platform do. Enter the .NET Framework.

ASP.NET provides the tools and technologies needed to write applications that can seamlessly and easily communicate over the Internet (or any other network such as an Intranet) using open standards like XML (eXtensible Markup Language). The .NET Framework also solves many of the problems developers face today. For example, ever cursed at having to shutdown ASP applications to replace component files? Ever wished you didn't have to register components, or spend hours trying to track down binary compatibility or versioning problems? The good news is that the .NET Framework provides a solution to problems like these. No more registering components or shutting down applications to upgrade them!

Even better news is that the .NET Framework also solves many of the problems you're likely to experience in the future. For example, ever considered how you're going to adapt your applications or web sites to run on or support small hand-held devices? Have you thought about the impact of the up and coming 64-bit chips from Intel? Microsoft has, and these are all catered for as part of .NET Framework.

So, the whole push towards the Internet stems from Microsoft's belief that all devices (no matter how small or large) will one day be connected to a broadband network: the Internet. We will all benefit in unimaginable ways from the advantages this global network will bring - your fridge could automatically send orders out to your local supermarket to restock itself, or your microwave could download the cooking times for the food you put in it, and automatically cook it. Wouldn't that be cool?

These ideas might sound a little futuristic, but manufacturers are already working on prototypes. Imagine if you were part of a team working on one of these projects. Where would you start? How many technologies and protocols would you need to use? How many are languages? How many are different compilers? Just thinking about some of these fairly elementary issues make my brain hurt. However, this is just the tip of the iceberg.

If a fridge were going to restock itself automatically, wouldn't it be cool to have it connect via the Internet to the owner's local supermarket, or any other supermarket available in some global supermarket directory. The supermarket systems and fridge would need to exchange information in some standard format like XML, ordering the goods and arranging delivery. Delivery times would have to be determined, probably by the fridge, based upon the owner's electronic diary (maybe stored on a mobile

device or in a central Internet location - Hailstorm for example), telling the fridge when the owner will be at home to accept the delivery.

Our lives as developers really are going to change a lot in the future, especially when web services are widely adopted. The Internet has already changed our lives and careers dramatically, and that change isn't slowing down. More and more devices are going to get connected, and if we are going to adapt quickly to these changes, we need a great tool set that enables us to meet the time-to-market requirements of the Internet. A tool set that also provides a consistent development strategy, no matter what type of development we're doing.



## VI. ASP.NET

### 6.1 What Is ASP.NET?

Unfortunately, the Internet still has bandwidth limitations and not every person is running the same web browser. These issues make it necessary to stick with HTML as our mark-up language of choice. This means that web pages will not look quite as amazing as a fully fledged application running under Windows, but with a bit of skill and creative flair, you can make some rather amazing web applications with ASP.NET. ASP.NET processes all code on the server (in a similar way to a normal application). When the ASP.NET code has been processed, the server returns the resultant HTML to the client. If the client supports JavaScript, then the server will use it to make the clients browser experience quicker and easier.

ASP.NET (Active Server Pages) is a new and extended technology to the earlier classic ASP, introduced by Microsoft Corporation, which fully supports Microsoft .NET Framework for developing next generation web Applications. It supplies all the required user interfaces under the name “WebForms” and also works with all .NET languages like Visual C#.NET, Visual Basic .NET etc. WebForms are used to create user interfaces for the web pages. Till now, you have to apply specific HTML tags like <Input> and <Select> for creating user interfaces and ASP scripting using VBScript or JavaScript. But with the introduction of ASP .NET, you need not apply them any more. All you have to do is to call the custom GUI classes defined in the System.Web.UI.WebControls namespace of the .NET Framework.

Moreover, System.Web namespace provides necessary classes, methods and properties for developing client - server applications and System.Web.UI namespace

interacts with other .NET language like C#, VB.NET. Therefore, a C# file containing some methods or C# syntax can be easily called in your ASP applications.

## **6.2 Weaknesses in the ASP 2 Model**

Failings in the ASP 2 model were most noticeable when the platform was contrasted against newcomers and developments in other technologies, such as Java Server Pages (JSP), Perl 5, PHP, and ColdFusion.

The main contender for ASP mind-share in Microsoft's most-needed marketplace, large-scale blue chip projects, was Java Server Pages. Microsoft could discuss the others as low-rent small to medium business and hobbyist technologies, and had an army of certified solutions companies and consultants to take care of those. On the other hand, products from Microsoft's biggest competitors, such as IBM, Oracle, and Sun, supported Java, and these companies had massive opinion forming clout in the world's largest corporations. As well as products such as IBM Net. Commerce (now Websphere), other vendors such as ATG and Broadvision were releasing application servers based around Java. To make matters worse, Microsoft could not claim to have the better technology.

JSP was outperforming and out-scaling ASP, plus the application servers and host operating systems proved time and again to be more robust and stable, and had lower cost of ownership and higher uptime!

The Java Server Pages and Servlets technologies allowed performance gains against ASP 2 partly because the code is compiled before execution. The Java language also had better error handling, object orientation, housekeeping, and variable typing. ASP, on the other hand, was based around interpreted scripting and languages that were compromised shadows of their already flawed parents.



### **6.3 Weakness in the ASP 3 Model**

Despite the great achievements of Active Server Pages, particularly in the areas of speed and stability, the platform was still based on incomplete scripting languages of VBScript and JScript, and third-party languages such as Perl.

Scripting languages required the developer to compromise coding standards and bolsters the application with components written in a second language, usually C++ or VB. The languages were not properly objects oriented, although they were object-aware, and could never perform very well whenever they required an interpreter to execute.

The reliance on the systems administrator for Web server configurations was also a problem; the administrator must register components, settings, and permissions on the server, and so deployment was not as simple as just uploading your files.

### **6.4 The Need for a New ASP Model**

It was evident that Microsoft would require a fundamental change to bring ASP up to the standard of industrial-strength programming. Active Server Pages was a technology based on the foundations of COM, ActiveX and COM technology provided much of its strength, but also many of its limitations. Microsoft would need to have a long hard look at COM to see how it could improve, and these changes would be bound to affect ASP. At the same time, Microsoft realized that the developers' playing field was changing. With new standards arriving all the time, particularly in information-sharing and distributed applications using XML, such as Simple Object Access Protocol (SOAP) and XML-RPC. Web services were becoming all the rage; Java was everywhere, and XML was taking the developer community by storm. A new version of ASP was not going to be enough to meet

these demands; the changes must be more far-reaching if they were not just going to catch up but also take the lead against such tough challenges.

In today's world, we do not have to contend just with different Web browsers but also with different distribution channels and modes of operation, with mobile phones and computers, interactive digital TV, intelligent appliances, digitally networked homes, and possibly moving from Web pages to disposable applications and Web services.

No doubt, as Microsoft was looking at their own technologies they must have analyzed the competition. As they announced the .NET framework, they also introduced a new language for the twenty-first century, C#. C# and .NET would address all of the criticisms, provide for a whole new way of looking at applications and the Web, and replace everything that had gone before, including Microsoft's flagships Visual C++, Visual Basic, and Active Server Pages.

ASP has achieved enormous success as a way of developing web sites, so why is a new version needed? Simply put, ASP has not evolved to take into account the way it's now being used. ASP has led to problems:

- (1) ASP is a scripted language, relying mainly on VBScript and JScript. Other languages are available if we install an interpreter, but it is still interpreted. The two disadvantages of interpreted languages are the lack of strong types (as supported by typed languages such as Visual Basic and C/C++), and the lack of a compiled environment. ASP does cache code, but it's still interpreted, and this inevitably leads to performance and scalability problems.
- (2) ASP does not provide an inherent structure for applications. In the days of static web pages, we used to see small, focused source files. With the

dynamic concept of ASP, it was possible to build code into the web page, again leading to problems. There is the eternal worry of mixing code and content, which can be a problem if you have a mixed team, with certain people designing the HTML and the interface, with different people doing the other coding. Having two sets of people working on the same files is asking for trouble. Another problem was the ability to make the code complex, leading to larger source files. Include files allow a certain amount of structure and code reuse, but it was never really a great solution.

- (3) The world of browser compatibility has morphed into device compatibility. Whilst the majority of Web access still takes place from a PC and browser, how long will that remain the case? Mobile devices are becoming more prevalent, and more powerful, leading to more problems designing sites. If you want your web site to obtain maximum reach you need to contend with these devices, and this means writing code to detect the device and render the appropriate content.
- (4) Standards compatibility also plays a big part in Web development. XHTML is becoming more widely accepted, XML and XSL/T are both now widely used, and talking to mobile devices might also mean support for WML. Support for these standards means that our ASP applications not only have to work with existing standards, but also be easily upgradeable to support future standards.

These are just some of the problems we will encounter when building ASP applications, but they aren't the only ones. The rapidly changing nature of the Internet often requires rapid changes to applications. For languages that have strong

development environments, practices such as componentization, code reuse, rapid development, and so on, are a great boon to a developer, but this sort of support is lacking in ASP. The rise of Business-to-Business applications, and peer-to-peer data sharing also brings great challenges to the developer.

ASP.NET was written from the ground up to meet these needs. Not only does it answer many of the questions posed by the existing development environment, but also provides great extensibility, and brings great tool support. At its minimum, all you require is the ASP.NET redistributable, which is freely available, and you can continue to use your favorite editor of choice (come on, admit it – it's Notepad). This gives us access to everything possible with ASP.NET, including multi-language support. For a richer environment you can use Visual Studio.NET, where you get the drag and drop support, colored code, context-sensitive help and tooltips, and all of the usual great editing features that Visual Studio has brought in the past.

The reasons for using ASP.NET are:

- (1) Developer Productivity
  - (a) Easy Programming Model. ASP.NET makes building real world Web applications dramatically easier. ASP.NET server controls enable an HTML-like style of declarative programming that let you build great pages with far less code than with classic ASP. Displaying data, validating user input, and uploading files are all amazingly easy. Best of all, ASP.NET pages work in all browsers, including Netscape, Opera, AOL, and Internet Explorer.
  - (b) Flexible Language Options. ASP.NET lets you leverage your current programming language skills. Unlike classic ASP, which supports only interpreted VBScript and JScript, ASP.NET now supports more

than 25 .NET languages (including built-in support for VB.NET, C#, and JScript.NET-no tool required), giving you unprecedented flexibility in your choice of language.

- (c) Great Tool Support. You can harness the full power of ASP.NET using any text editor-even Notepad! But Visual Studio .NET adds the productivity of Visual Basic-style development to the Web. Now you can visually design ASP.NET Web Forms using familiar drag-drop-double click techniques, and enjoy full-fledged code support including statement completion and color-coding.
- (d) Rich Class Framework. Application features that used to be hard to implement, or required a 3rd-party component, can now be added in just a few lines of code using the .NET Framework. The .NET Framework offers over 4,500 classes that encapsulate rich functionality like XML, data access, file upload, regular expressions, image generation, performance monitoring and logging, transactions, message queuing, SMTP mail, and much more!

The Enterprise versions of Visual Studio .NET deliver life-cycle features to help organizations plan, analyze, design, build, test, and coordinate teams that develop ASP.NET Web applications. These include database modeling (conceptual, logical, and physical models), testing tools (functional, performance and scalability), and enterprise frameworks and templates, all available within the integrated Visual Studio .NET environment.

## (2) Improved Performance and Scalability

ASP.NET lets you use serve more users with the same hardware.



- (a) Compiled execution. ASP.NET is much faster than classic ASP, while preserving the “just hit save” update model of ASP. However, no explicit compile step is required! ASP.NET will automatically detect any changes, dynamically compile the files if needed, and store the compiled results to reuse for subsequent requests. Dynamic compilation ensures that your application is always up to date, and compiled execution makes it fast. Most applications migrated from classic ASP.
  - (b) Rich output caching. ASP.NET output caching can dramatically improve the performance and scalability of your application. When output caching is enabled on a page, ASP.NET executes the page just once, and saves the result in memory in addition to sending it to the user. When another user requests the same page, ASP.NET serves the cached result from memory without re-executing the page. Output caching is configurable, and can be used to cache individual regions or an entire page. Output caching can dramatically improve the performance of data-driven pages by eliminating the need to query the database on every request.
  - (c) Web-Farm Session State. ASP.NET’s session state lets you share session data user-specific state values across all machines in your Web farm. Now a user can hit different servers in the web farm over multiple requests and still have full access to her session.
- (3) Enhanced Reliability

ASP.NET ensures that your application is always available to your users. Memory Leak, DeadLock and Crash Protection. ASP.NET

automatically detects and recovers from errors like deadlocks and memory leaks to ensure your application is always available to your users.

For example, say that your application has a small memory leak, and that after a week the leak has tied up a significant percentage of your server's virtual memory. ASP.NET will detect this condition, automatically start up another copy of the ASP.NET worker process, and direct all new requests to the new process. Once the old process has finished processing its pending requests, it is gracefully disposed and the leaked memory is released. Automatically, without administrator intervention or any interruption of service, ASP.NET has recovered from the error.

#### (4) Easy Deployment

ASP.NET takes the pain out of deploying server applications.

- (a) “No touch” application deployment. ASP.NET dramatically simplifies installation of your application. With ASP.NET, you can deploy an entire application as easily as an HTML page: just copy it to the server. No need to run regsvr32 to register any components, and configuration settings are stored in an XML file within the application.
- (b) Dynamic update of running application. ASP.NET now lets you update compiled components without restarting the web server. In the past with classic COM components, the developer would have to restart the web server each time he deployed an update. With ASP.NET, you simply copy the component over the existing DLL.

ASP.NET will automatically detect the change and start using the new code.

- (c) **Easy Migration Path.** You do not have to migrate your existing applications to start using ASP.NET. ASP.NET runs on IIS side-by-side with classic ASP on Windows 2000 and Windows XP platforms. Your existing ASP applications continue to be processed by ASP.DLL, while new ASP.NET pages are processed by the new ASP.NET engine. You can migrate application by application, or single pages. And ASP.NET even lets you continue to use your existing classic COM business components.

(5) **New Application Models**

ASP.NET extend your application's reach to new customers and partners.

- (a) **XML Web Services.** XML Web services allow applications to communicate and share data over the Internet, regardless of operating system or programming language. ASP.NET makes exposing and calling XML Web Services simple. Any class can be converted into an XML Web Service with just a few lines of code, and can be called by any SOAP client.

Likewise, ASP.NET makes it incredibly easy to call XML Web Services from your application. No knowledge of networking, XML, or SOAP is required.

- (b) **Mobile Web Device Support.** ASP.NET Mobile Controls let you easily target cell phones, PDAs-over 80 mobile Web devices-using ASP.NET. You write your application just once, and the mobile

controls automatically generate WAP/WML, HTML, or iMode as required by the requesting device.

## 6.5 Reviewing the Basic of the ASP.NET

Microsoft has done a great job of bringing ASP and their older languages into the twenty-first century with .NET. ASP.NET, using VB. NET, is now a full- fledged object-oriented Web application development platform, and has seen many improvements; but the past legacy languages should not hold back a new initiative as massive as .NET, so Microsoft developed a new headline-grabbing language for the .NET Framework, called C#.

The following are some key points about ASP.NET:

- (1) ASP.NET is a key part of the wider Microsoft .NET initiative, Microsoft's new application development platform.
- (2) .NET is both an application architecture to replace the Windows DNA model and a set of tools, services, applications and servers based around the .NET Framework and common language runtime (CLR).
- (3) Rather than just being ASP 4 or an incremental upgrade, ASP.NET is a complete rewrite from the ground up, using all the advanced features .NET makes available.
- (4) ASP.NET can take advantage of all that .NET has to offer, including support for around 20 or more .NET languages from C# to Perl.NET, and the full set of .NET Framework software libraries.
- (5) Web applications written in ASP.NET are fast, efficient, manageable, scalable, and flexible, but, above all, easy to understand and to code!

- (6) Components and Web applications are all compiled .NET objects written in the same languages, and they offer the same functionality, so no need to leave the ASP environment for purely functional reasons.
- (7) You'll have less need for third-party components. With a few lines of code, ASP.NET can talk to XML, serve as or consume a Web service, upload files, "screen scrape" a remote site, or generate an image.

## 6.6 Benefits of ASP. NET

ASP.NET offers several important advantages over previous Web development models:

- (1) Enhanced Performance. ASP.NET is compiled Common Language Runtime code running on the server. Unlike its interpreted predecessors, ASP.NET can take advantage of early binding, just-in-time compilation, automatic resource optimization, runtime profiling, automatic memory management, enhanced exception handling, and caching services, right out-of-the-box, this improves the performance before you start coding.

For example, a database table will spend the majority of its execution time connecting to the database and querying the information. ASP.NET comes with a data-caching module. This data-caching module allows you to specify what data on an ASP page to cache and on what conditions to empty the cache and re-query the data-store. ASP.NET ships with Performance Counters which system administrators can use to gather application metrics. They can be used to measure the performance of either a single instance of an ASP.NET application or all ASP.NET applications combined on a computer.



- (2) World-class Tool Support. The ASP.NET framework is complimented by a rich toolbox and designer in the Visual Studio integrated development environment. WYSIWYG editing, drag-and-drop server controls, and automatic deployment are just a handful of features this powerful tool provides.
- (3) Power and Flexibility. Because ASP.NET is based on the Common Language Runtime, the power and flexibility of that entire platform is made available to web application developers. The Common Language Runtime's Base Class libraries, Messaging, and Data Access solutions are all seamlessly accessible from the web. ASP.NET is also language-independent, so you can choose a language that best applies to your application, or partition your application across many languages. Further, Common Language Runtime interoperability guarantees that your existing investment in COM-based development is preserved when migrating to ASP.NET.
- (4) Simplicity. ASP.NET makes it easy to perform common tasks, from simple form submission and client authentication to deployment and site configuration. For example, the ASP.NET Page Framework allows you to build user interfaces that cleanly separate application logic from presentation code, and handle events in a simple, VB-like forms processing model. Additionally, the Common Language Runtime simplifies development with managed code services like automatic reference counting and garbage collection.
- (5) Manageability. ASP.NET employs a text-based, hierarchical configuration system, which simplifies applying settings to your server environment and

web applications. Because configuration information is stored as plaintext, new settings may be applied without the aid of local administration tools. This “zero local administration” philosophy extends to deploying ASP.NET applications as well. An ASP.NET application is deployed to a server simply by copying the necessary files to the server. No server restart is required, even to deploy or replace running compiled code!

- (6) Scalability and Availability. ASP.NET has been designed with scalability in mind, with features specifically tailored to improve performance in clustered and multi-processor environments. Further, processes are closely monitored and managed by the ASP.NET runtime, so that if one misbehaves (leaks, deadlocks), a new process can be created in its place, which helps keep your application constantly available to handle requests.
- (7) Customizability and Extensibility. ASP.NET delivers a well-factored architecture that allows developers to “plug-in” their code at the appropriate level. In fact, it is possible to extend or replace any sub-component of the ASP.NET runtime with your own custom-written component. Implementing custom authentication or state services has never been easier.
- (8) Security. With built in Windows authentication and per-application configuration, you can be assured your applications are secure.

## 6.7 ASP.NET Example

- (1) Open up a text editor and type in the following code:

```
<html>
```

```
<head><title>ASP.NET Test Example</title></head>
```

```
<body>
```

```
<h1>Welcome</h1>
```

This message comes from HTML.

```
<br>
```

```
<asp:label id="label" runat="server" text="This message comes from ASP.net "/>
```

```
</body>
```

```
</html>
```

- (2) Save this page as test.aspx.

When you save the file, you should double-check that your new file has the correct suffix. It should be .aspx, since this is how you tell the web server that the page contains ASP.NET code. Be aware that Notepad (and many other text editors) consider .txt to be the default. So in the Save or Save As dialog, make sure that you change the Save As type it to read All Files, or All Files(\*.\*), or enclose the path and filename in quotes.

- (3) Now start up your browser and it will display as Figure 6.1:

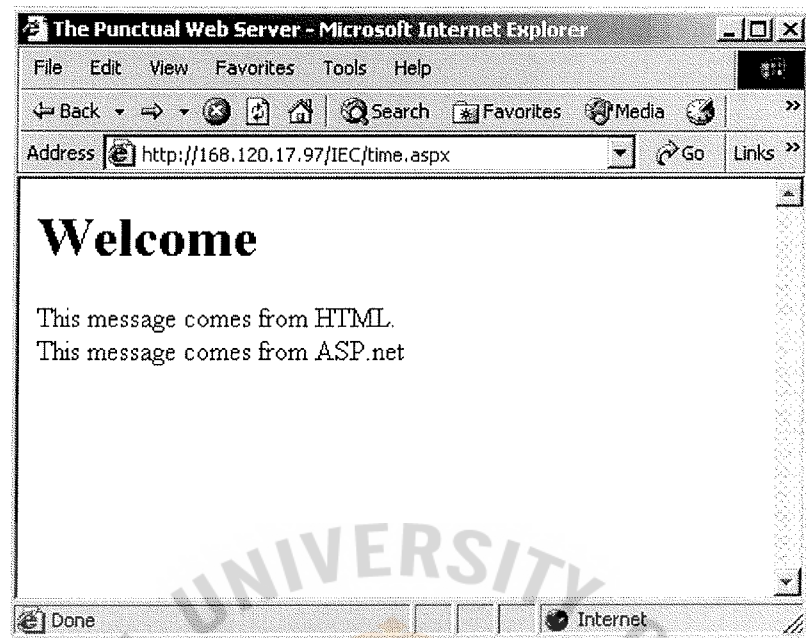


Figure 6.1. The Result from test.aspx File.

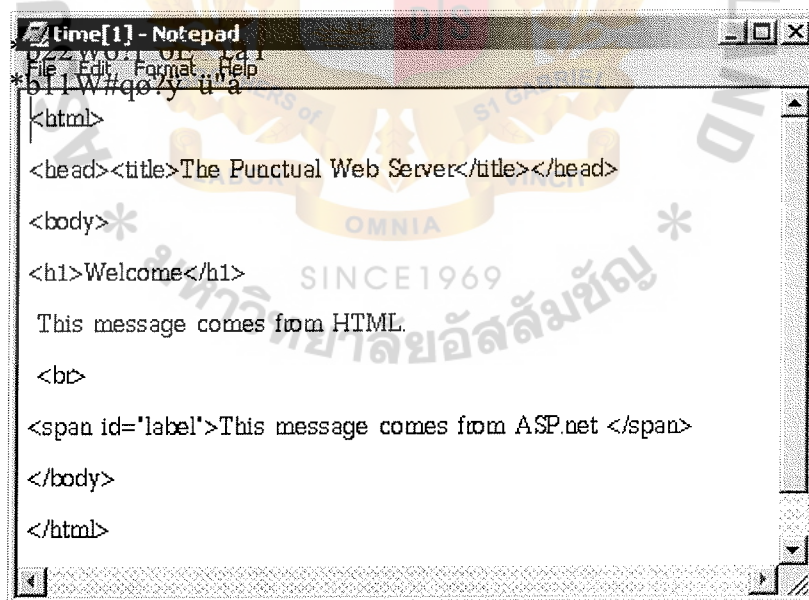


Figure 6.2. HTML Source for test.aspx after the Execution.

(4) Now on your browser select View Source or similar, (depending on which browser you are using) from the browser menu to see the HTML source that was sent from the web server to the browser. The result is shown below as Figure 6.2. You can see that there is no ASP.NET code to be seen, the ASP.NET code has been processed by the web server and used to generate pure HTML, which is hard-coded into the HTML source that is sent to the browser.

(5) As we mentioned before, you can expect this to work in any browser - because the ASP.NET is processed on the web server. If you have another browser available, try it out.

### 6.8 How Does ASP.NET Work?

For a better understanding on how ASP.NET works, let you see the traditional way that a Web page works (without ASP.NET) as Figure 6.3.

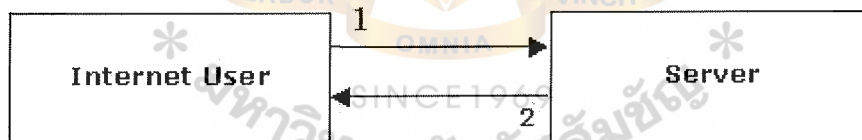


Figure 6.3. Without ASP.NET.

- (1) Any internet user (client) is asking for internet-page.
- (2) The Web-Server sends the physical content of the page back.

As you can see in the above image, calling a static page is very simple process. A client is asking/demanding for a Web page. For that you will need a connection between your client (by using IE, Netscape etc.) and the server - this is done through



the Internet. The file must exist on the server otherwise you will get a "404 File not found" error. The server reads the requested file and sends it back to the client. It does not matter who the client is or when the request reaches the server, the result will always be the same - until the file on the server is modified.

Now let us see how this will look with the usage of ASP.NET as Figure 6.4:

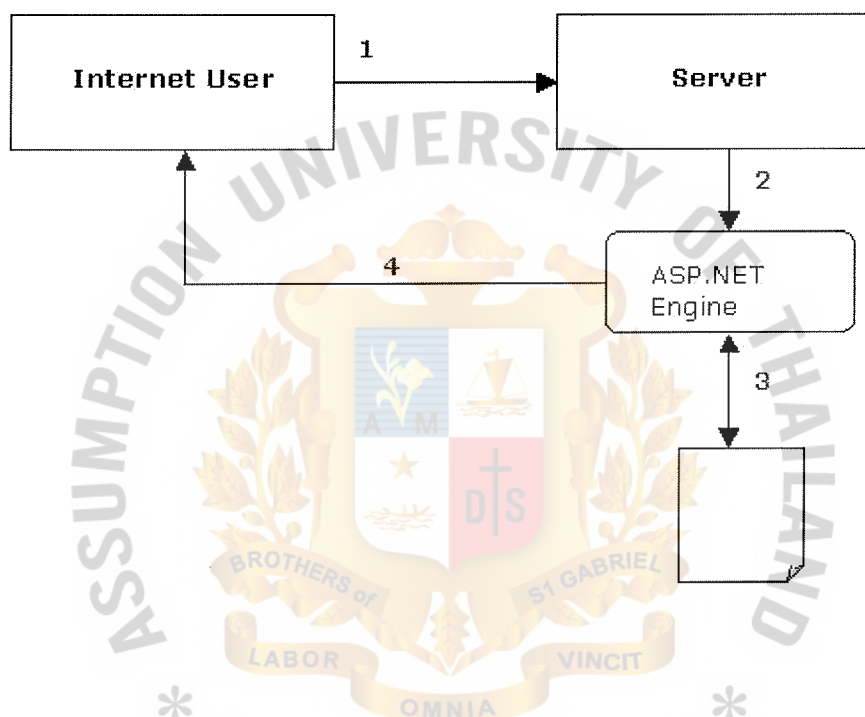


Figure 6.4. With ASP.NET.

- (1) Any Internet user (Client) is asking for internet-page.
- (2) Server sends the inquiry to the ASP.NET engine.
- (3) ASP.NET creates the required new web page for the user.
- (4) ASP.NET sends the created page back the Client.

The request of a dynamic Web page differs much from the static one. After the server has received the inquiry, it will be sent to the ASP.NET engine. This checks whether the requested file exists. If the file exists, ASP.NET will not simply send the

content back, instead it will create a new dynamic page as HTML that will be sent back to the client. This dynamic page can look different from user to user. If the file does not exist, it will send back the message to let the user know that request cannot process because there is no file.



## VII. ASP VS ASP.NET

The first difference an experienced ASP developer will notice is that VBScript support has been dropped in favor of VB.NET. This is not as much of a hurdle as it sounds like, as the syntax is quite similar, and VB.NET is a full-fledged language and so provides a lot richer environment than VBScript ever could.

As described above, all ASP.NET languages are object oriented, event driven, and server compiled. This brings many benefits, especially where improvements were needed most, namely, performance, stability, scalability, and manageability.

With Classic ASP, you pretty much had to code your whole application from scratch. ASP.NET has several labor-saving additions to make life easier. Web forms introduce a new Visual Basic Rapid Development-style way of looking at forms in Web pages. With Web Forms, the developer uses new form components that you can add in the traditional way or through code, and they enable the programmer to call on server-side event-driven programming and true separation of layout and logic. You can separate the layout code and functions by using code behind pages that use inheritance to add methods to the form. .NET form controls maintain the session state so the users input remains when the page is submitted, and the controls' property values are available to the ASP code without resorting to querying the request object.

The framework foundation class libraries contain exciting new features, previously only available from third parties such as the System.Drawing tools, which enable you to build dynamic images on the fly, built-in browser-based file upload and system network services for working with TCP/IP and DNS.

With Web Services and built-in support for SOAP you can distribute code and applications. Your ASP.NET scripts can consume services across the Web, and publish and expose routines as services just as easily.

Deployment, including server configuration, is mostly just a matter of transferring files with configuration that was previously only available from the MMC now implemented with XML files. Now you do not need to register and not register components, and the server can handle multiple versions of the same component without conflicts.

Mission critical services has increased support with load balancing and several state-management options, including the ability to store state information in an SQL Server database and pass the session ID on the URL to avoid requiring the user to have cookies. The core differences between ASP and ASP.NET are:

- (1) Any language for writing script: The old version of ASP can use only the script as VBScript and Jscript, but ASP.NET has the full version of language which has the three basic languages are C#, VB.NET, and Jscript.NET. Microsoft have the future plan to develop the language for all.
- (2) Flexibility on program: ASP.NET can provide the allowance to us coding the program by using multi-language on same file. This is good for a programmer who can take any language that he wants to code. For example, the loop of VB is coded easier than C# but its functions usage is more easiest, therefore the loop should use VB and the functions should use C#.
- (3) More Libraries: Some application on ASP is not created easily and has to use various components to support it, but ASP.NET can solve that

problem by a library such as a mail and uploading library, etc. Therefore, we can create the application for the free host that it has the limitation on the components.

- (4) Easy controls: It is the addition from ASP version that is no controls. This control provides us to create the home page effectiveness and efficiency, so we do not worry about the old or new browser that supports a language. Another advantage is the controls can be used to instead of any script that we often use them. For example, when we need to display the data in form of a table on the home page and if we use ASP, so we have to code a program so complicated and many lines. On contrast, if we use the ASP.NET, it has less code line and causes to good execution for program.
- (5) Independence hardware: Any device can communicate together such as Palm, PDA, WAP mobile phone and etc.

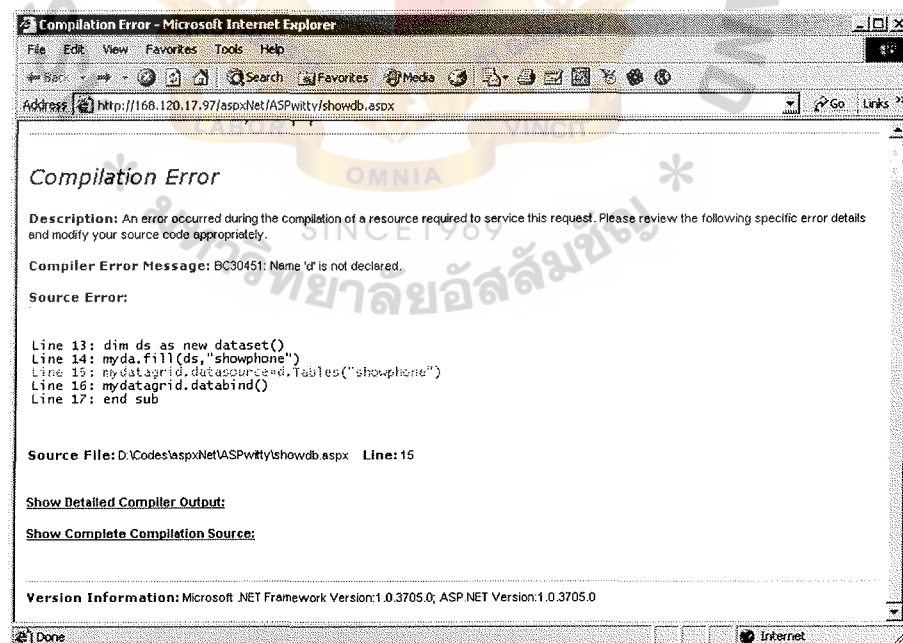


Figure 7.1. Error Message from ASP.NET.



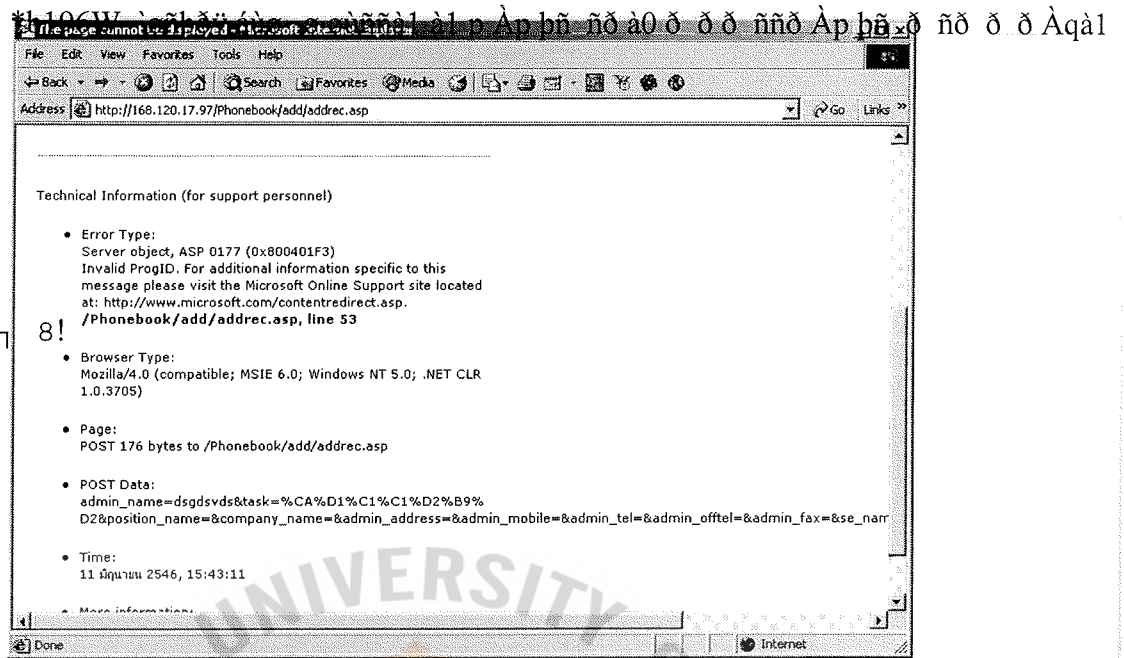


Figure 7.2. Error Message from ASP.

- (6) Error detection easily: If we use the ASP and there is an error on the program, it will inform you which line has the error as Figure 7.2. On the other hand, if we use ASP.NET and there is the error on the program, it will inform you more details with the solutions as Figure 7.1.
- (7) Event self-checking within the homepage: There is the various events self-checking from uploading a page till closing a page, so we can set up any event easily.
- (8) HTML and ASP separate: The HTML and ASP source will be mixed together on the file but the new ASP.NET version will separate each part which one is HEML and which one is ASP.
- (9) Web From Controls: When we use ASP with HTML to server-side, we have the problems as:

- (a) Consistency. We are still stuck with the rather non-intuitive nature of some HTML controls. Why for example, is there an “INPUT” tag for single-line text entry, but a “TEXTAREA” tag for multi-line text entry? Surely a single control where we specify the rows and columns makes more sense?
- (b) User Experience. How do we easily write sites that render rich content for browsers such as IE, while also preserving compatibility with down level browsers? HTML doesn’t have the ability to change its content depending on the browser - we have to write the code for that.
- (c) Devices. How do we write sites that cope with devices other than browsers? WAP-phones, PDAS, and even fridge have browsers nowadays. Like the browser issue, we’d have to manually write code for this.

The ASP.NET can tackle the above problems by:

- (a) Providing a consistent naming standard. For example, all text entry fields are handled by the “TextBox” control. For the different modes (multi-line, password, etc.) we just specify attributes.
- (b) Providing consistent properties. All server controls use a consistent set of properties, making it easier to remember. For example, the “Text” field of a “TextBox” is more intuitive than a value field.
- (c) Providing a consistent event model. Traditional ASP pages often have large amounts of code handling the posting of data, especially when one page provides multiple commands. With ASP.NET we

wire up controls to event procedures, giving our server-side code more structure.

- (d) Emitting pure HTML, or HTML plus client-side JavaScript. With one minor exception (which is intentional) the server controls emit HTML 3.2 by default, giving great cross-browser compatibility. This can be changed so that by default we target up-level browsers such as IE, where the controls will emit HTML 4.0 and DHTML, providing a richer interface. All the user ever sees is the HTML content, not the server controls.
- (e) Emitting device-specific code. Certain controls will emit HTML when requested by a browser, but WML when requested by a WAP-phone. The control handles the detection of the device and the generation of the correct markup.

Let's take a quick look at a simple example to show how these controls work:

```
<html>
<script language = "VB" runat = "server">
Public Sub btn_Click(Sender As Object, E As EventArgs)
'some code goes here
End Sub
</script>
<body>
<form runat = "server">
Press the button: <asp:Button runat="server" Text="Press Me", OnClick="btn_Click,
runat="server"/>
</form>
```

</body>

</html>

The server control in this example is a button, added to the page using the asp: Button element. There are several things to note about this control:

- (1) It has the runat = “server” attribute set, to tell ASP.NET that it should process this control,
- (2) It uses the “Text” attribute to set the text to be shown on the button. This is consistent with other controls.
- (3) It uses the “OnClick” attribute to identify the event procedure to be run when the button is clicked. Since this is a server control this event procedure runs on the server.



## VIII. COMPARATIVE ANALYSIS: ADO AND ADO.NET

### 8.1 ADO and ADO.NET Overview

There are some areas in which ADO and ADO.NET resemble one another closely and others in which they are wildly different. Let us first look at areas of similarity.

Both models require a connection to a data store to fetch data, and the code for getting a connection is similar for both. Both models have Connection and Command objects. The Connection object is extremely similar in both models. The Command object is conceptually similar in its function for both, but the object model changes. However, the similarity in these areas means that code to create and open a connection, and use a Command object to execute commands, will look immediately familiar to the experienced ADO developer.

Both models manipulate collections of rows and fields, but their techniques of manipulation are different. Both models have support for transactions through the Connection object, and this functionality uses similar code in both models. And both models can be bound to controls for automatic data handling. \*

At that point the similarities begin to diminish. In ADO, the main construct for holding data to be manipulated is the Recordset. It is a set of rows or records holding data that was typically fetched from some data store. ADO.NET has no Recordset object. The functional equivalents to a Recordset in ADO.NET depend on the type of data access you need. The two main ones are called a DataReader and a DataSet.

There are other significant differences. For example, ADO.NET has no functionality at all for pessimistic concurrency, in which records are locked when they are accessed and remain locked until the lock is released by the accessing code.



This type of concurrency has no place in a stateless Web environment because it depends on extensive maintenance of state information. The only type of concurrency available in ADO.NET is optimistic concurrency, in which records are checked for changes when an attempt is made to rewrite a record.

## **8.2 What Is ADO?**

ADO stands for Active Data Object. It is based on the concept of a 24/7 (24 hours, 7 days a week) “connected world,” such as is found on a corporate local area network (LAN). You can create a RecordSet; can connect it to a data source. The RecordSet stays “plugged in,” if you will, to the data source, and changes to the data are posted to the data store immediately. A Microsoft data access technology that enables your client applications to access and manipulate data from a database server or any other data store through an OLEDB provider. Sometimes called just Active Data Objects. ADO provides to:

- (1) Allow access to all types of data
- (2) Support simple object model
- (3) Support asynchronous queries
- (4) Support client-side and server-side cursors
- (5) Support disconnected RecordSets
- (6) Can define and use disconnected Recordsets

A programmer can use ADO to access database with general programming languages or script languages. ADO is supported by a lot of languages (e.g. Visual Basic, Java or Visual C++) including ASP.

## **8.3 What Is ADO.NET?**

Traditional data access with ADO revolves around the fundamental data storage object the RecordSet. The technique used there is to create a connection to a data

store using either an OLEDB (Object Linking and Embedding Database) provider or an ODBC through OLEDB driver (depending on the data store and the availability of the provider). Then execute commands against it that return a RecordSet object containing the appropriate data. This can be done using a Command object, or directly against the Connection object. Alternatively, to insert or update the data, we simply execute a SQL statement or a stored procedure within the data store using the Connection object or Command object directly, without returning a RecordSet object.

Data access in .NET follows a broadly similar principle, but uses a different set of objects. So, switching to .NET does not involve learning a completely different technique. However, the objects we use are quite different, providing much better performance with more flexibility and usability.

The .NET data accesses object model is based around one fundamental object - the DataSet. This replaces the RecordSet from traditional ADO. It provides many new features that make complex data access techniques much more efficient, while remaining as easy to use as the RecordSet object. The main difference is that a DataSet object can hold more than one table (in other words more than one RowSet) from the same data source, as well as the relationships between them.

We can create a DataSet from existing data in a data store, or fill it directly with data one record at a time using code. It also allows us to manipulate the data held in the DataSet's tables and build and/or modify the relationships between the tables within it. Each table within a DataSet maintains details of the original values of the data as we work with it, so that these changes can be pushed back into the data store at a later date. The DataSet also contains metadata describing the table contents, such as the columns' types, rules, and keys. Remember that the whole ethos is to be able to work accurately and efficiently in a disconnected environment.

The DataSet object can also persist its contents, including more than one data table or RowSet, directly as XML, and load data from an XML document that contains structured data in the correct format. In fact, XML is the standard persistence format for data sets in .NET -bringing it more into line with the needs of disconnected and remote clients.

Since data access has become very important in all types of applications – windows and web, Microsoft has brought out a new data access technology – ADO.NET – to access data. This is successor to ADO, which was used to access data prior to .NET. This is a new Data Access Technology that the Microsoft .Net Framework uses to access Databases/Data providers such as SQL, OleDb, ODBC etc and also it is main exchange of Data that is done XML, this makes is able to interoperate with basically any system on the internet.

It uses a disconnected form of technology in the sense that once Data is retrieved, that client that retrieved the data gets disconnected from the main Database until it is time to Update its changes, by using the disconnected idea the following are achieved:

- (1) There is an overall increase in performance of your network because the Webserver/clients do not have to stay connected to the DataSource.
- (2) This relieves the Developer of have to implement his own disconnected RecordSet in his Code (ASP).

.NET has brought so many changes to the way one writes programs. Once of the key changes has been ADO.NET. It is an improvement over ADO. Though ADO was used in windows and web applications for quite some time, it suffers from the following limitations. The following are the some of the problems with ADOs:

- (1) Data is stored in RecordSet - a data structure that is stored in memory and needs a connection to the database throughout the lifetime of the RecordSet. This exhausts connections on the database. Though this problem is solved using disconnected recordsets and RDS (Remote Data Services), it involves extra programming.
- (2) Establishing relationship between multiple tables is difficult. Each table should have its own RecordSet and after taking a row from master table (first RecordSet); we have to search for its child records in child table (second RecordSet). Otherwise we have to join both the tables and put the joined data into the RecordSet. The first approach involves more programming effort and later approach needs more space in memory, as master record is stored along with each child record.
- (3) Does not support XML (Extensible Markup Language)
- (4) Data is transmitted across machines using COM marshalling. This supports the data types that are defined in COM standard and in some cases it requires that existing data to be converted to the standard data type. This will also have problem in Internet, as Firewalls may not permit low-level data transfer.

Whether you have faced with any of the problems mentioned above or not, you have to move on to ADO.NET. If not anything else (assuming you don't need anything more than what ADO provides) at least it will provide all that your ADO is providing.

#### **8.4 Why We Need Another Data Access Model?**

Data access techniques in Visual Basic have evolved quickly in the last few years. We have gone from local access using things like the Access Jet engine, to

client-server access using databases such as Oracle and SQL Server, and then to Internet access, all since 1993.

In an attempt to keep up, Microsoft has introduced several models for accessing data. Data Access Objects (DAO) were introduced for local access.

DAO proved insufficient to work well with client-server architectures, so Remote Data Objects (RDO) came next. RDO proved less than ideal and was only used widely for a couple of years. Microsoft learned from that attempt and introduced ADO.

Like RDO, ADO was designed for connection-based, client-server architectures, but it was introduced just about the time the Web became an important factor in creating business systems. Microsoft responded with regular versions of ADO, rolling out new capabilities

Despite Microsoft's attempt to retrofit Web-oriented capabilities on ADO, developers began to see limitations in ADO for Web development. The most important were:

- (1) Connection-based use of ADO required unacceptable overhead. Architectures to avoid continuous connections required more development work.
- (2) RDS (Remote Data services) worked for distributed access but imposed a variety of limitations. Using it effectively involved a steep learning curve.
- (3) XML became a de facto standard for data interchange on the Web, but ADO was designed before XML became important, and so ADO's integration with XML was weak.
- (4) Early versions of ADO were mutually incompatible. This was fixed in later versions but only after developers had been forced to make many



conversions to newer versions just to keep software systems on one server in synch with a single version of ADO.

We can summarize the most important problems into two areas. ADO is not ideal for distributed architectures, and it does not integrate well with XML. These are the biggest problems with ADO that Microsoft needed to address as it moved to .NET.

We should move to ADO.NET because .NET is the future for programmers working with Microsoft technologies. So, we have to move on to new technologies that are made available on .NET. ADO is gone and now is ADO.NET and ADO.NET does come with some new interesting and important features. The reasons why we need the new data access model are:

#### Scalability

This is very crucial for Web applications. ADO.NET provides a new component called DataSet. It does not need constant connection to the database. It connects to database, retrieves data and disconnects from database. It allows the data stored in memory to be manipulated. It again connects to the database when changes made to data in memory are to be made to database.

That means the connection time to database is kept to absolute minimum. This is an important development since connections of the database are a bottleneck. This process allows more number of clients to connect to database thus increasing scalability (the ability to scale application up/down) of the application. That also means though more number of clients is accessing the database, the number of connections at any given time will not significantly increase and thus enabling system provide more scalability.

## Performance

ADO.NET increases performance. It uses XML to transmit the data whereas ADO used COM marshalling. COM marshalling spent a lot of time in converting data types of the database to data type defined in COM. This overhead removed from ADO.NET.

ADO.NET supports DataSet, which is a mini database in memory. It supports constraints and relationship between tables in memory. This enables master-detail relationship between tables in memory.

## XML Support Is Built-in

As XML is becoming new way of storing information, XML support in ADO.NET is much wanted. ADO.NET supports storing data in XML documents and retrieving information from XML documents.

Apart from the ADO.NET stores data in XML format internally and transmits data using XML. This enables any application – whether or not ADO.NET based – can receive the data sent by an ADO.NET component. This becomes an important feature while developing Web Services, which are based on protocol SOAP that also uses XML.

## 8.5 Comparisons of ADO and ADO.NET

There are differences between ADO and ADO.NET:

- (1) Modify from objects to libraries: The old version of ADO uses the objects, but ADO.NET employs a class form that is the principle of Base Classes' .NET framework. Therefore, you can import the namespace file into the program.
- (2) XML database connection: ADO does not support the database connection of XML, but ADO.NET does.

- (3) Database connectionless: If you use ADO for the database connection, the connection will be processed all the time till you write the close command. This makes utilize the resource of a server and it is time-assuming that you write the close command always. The source code will be long and sometimes you may forget writing it. ADO.NET, it will copy the data from the database to “Catch” and then it disconnects to database automatically. However, if you need to connect the database, ASP.NET allows that.
- (4) There are the specific libraries to connect database of SQL server and XML: the connection to database has to have the middle connection that can communicate between database such as OLEDB, ODBC, etc. ASP requires ODBC, but ADO.NET has the capability to connect the database by using a namespace. The namespaces are employed for connecting database of SQL server or XML without the middle connection. It is convenient and high-speed for connection to database. However, the connection to database by using a middle connection is still processed.
- (5) DataSet, DataReader replacing RecordSet: ADO needs the RecordSet object to manage the data in the form of Table, but ADO.NET applies to the DataSet and DataReader object instead. This provides the alternative to a user or a programmer for managing the data.
- (6) Data Transmission by using XML: If there is the firewall on the server, when you transmit the data to that server, you cannot transmit it because the firewall will protect all damaged things to it. ADO has this problem, but ADO.NET will not be. Because you can transmit the data file in from of XML to server via the firewall on HTTP protocol as same as HTML.

- (7) Relationship between tables: ADO can use only one table's database. If you need to connect more than one database table within the same homepage, you have to examine the previous database table is already closed because if you do not close it, you cannot connect another database table. On the other hand, ADO.NET has the capability to connect more than one database table at the same page.



## **IX. COMPARATIVE DATABASE CONNECTION BETWEEN ASP AND ASP.NET**

### **9.1 Database Connection Forms**

There are a number of ways to connect to a database. You can use a DSN, a DSN-less connection, or the native OLEDB provider.

Back in the old days, database connectivity was difficult. Everybody had their own database formats, and developers had to know a low level API (Application Programming Interface) for each database they wished to develop for. There was a push for a universal API, an API that would work for numerous data stores. It was about this time that ODBC, or Open Database Connectivity, which was an early attempt at creating this universal API. A number of databases conformed to this standard, and became known as ODBC-compliant databases. ODBC-compliant databases consist of Access, MS-SQL Server, Oracle, Informix, etc.

Well, ODBC was not perfect. It still contained a lot of low-level calls, and was difficult to develop with. Developers had to focus more on low-level communications with the database, as opposed to being able to concentrate on getting the data they needed and using it how they saw fit. Along came Microsoft's solution: DAO, or Data Access Objects.

After DAO came RDO (Remote Data Objects, targeted for distributed database architecture), and then ADO. These have all had their shortcomings, though. According to Microsoft, "ODBC provides native access to SQL data" and "DAO provides high-level objects to data." Even DAO and RDO require the data in a data store to be in SQL format (Structured Query Language). In response to these shortcomings, Microsoft introduced OLEDB, a COM-based data access object which



provides access to all types of data, and even provides access to disconnected data stores.

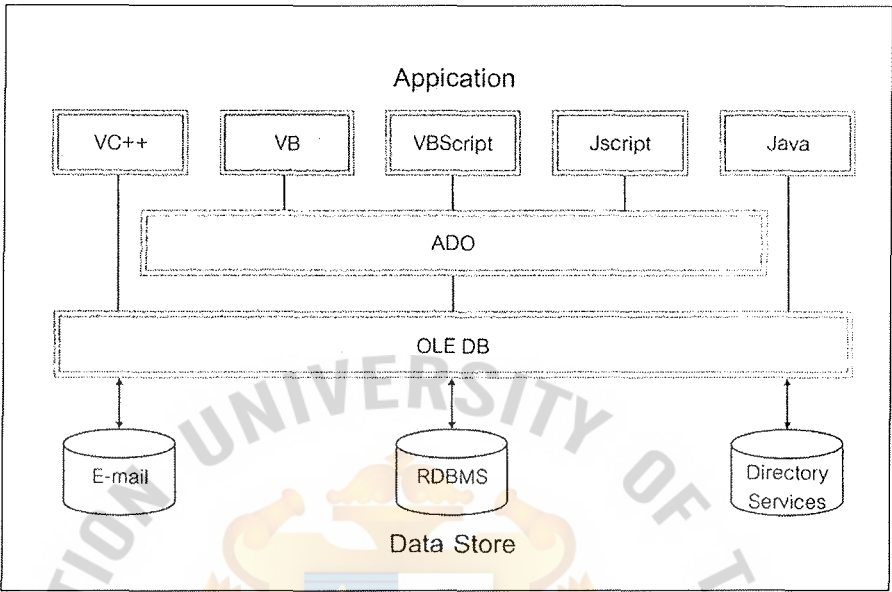


Figure 9.1. OLEDB Architecture.

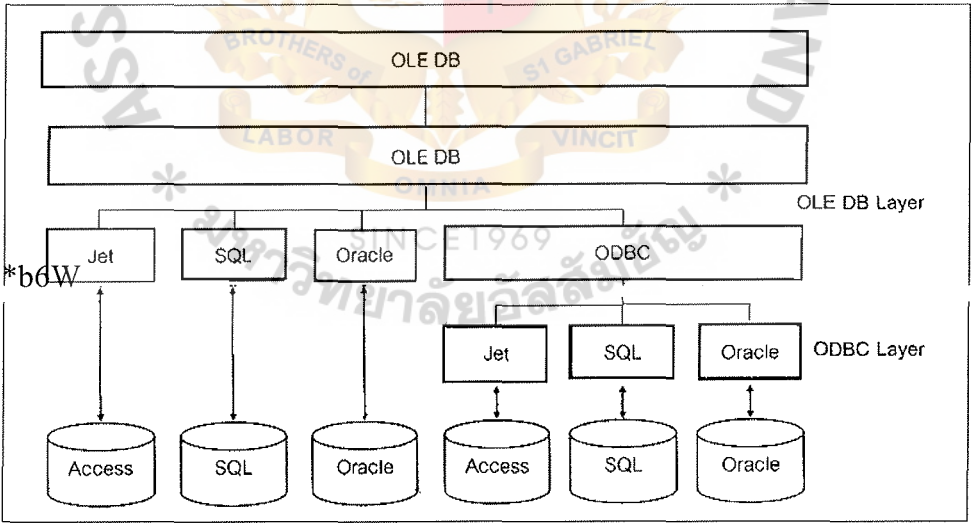


Figure 9.2. OLEDB Providers and ODBC Drivers.

From Figure 9.1, the highest layer is the application that can be a web site or any application written from various languages. Those will access to a database such

as RDBMS (Relational Database Management System) that is a database, Messaging or Directory Services that has OLEDB. It responses to direct the connection between various languages such as Visual C++, JAVA, etc. In the case of a language does not connect to it own OLEDB such as Visual Basic etc and this includes any script such as VBScript, Jscript, etc. Those have the middle connection between application and OLEDB are ADO (ActiveX Data Object).

From Figure 9.2, we can see that the Provider will locate on OLEDB layer, while Drivers is ODBC layer. If we apply OLEDB Providers to connection a database, it is high-speed connection because OLEDB connect directly to a database without going through ODBC. ODBC has to connect the driver of a database at the first time to know the database type and after that it will access to that database.

ADO has three classes for a database connection as:

(1) DSN Connection

The DSN (Data Source Name) connection technique requires the setup of ODBC (Open Database Connectivity) Data Source in the first. ODBC has the responsibility for the middle connection. This technique has the script written shorter than other techniques and more facility than other database connection methods. The drawbacks are:

- (a) It is slow for connection.
- (b) Some host does not allow using this technique because each database connection has to setup ODBC and the setup can be done by only a server administrator.

At the present day, this method is not so popular and Microsoft announced that ASP.NET removes it. The example of DSN Connection is:

```
Set Conn = Server.CreateObject("ADODB.Connection")
```

```
Conn.Open "b"
```

From an example, we declare the variable object in order to open the database by opening ODBC that is referred to "b".

## (2) DSNLess Connection

This technique does not require DSN, but it applies to a driver of each Database. It is more facility than DSN connection because we do not set up any things for the connection. Most of hosts have ability to connect a database by this method.

## (3) OLEDB (Object Linking and Embedding Database) Connection

This is the latest technique for the database connection. Many developers have to test it and it is very high-speed to connect a database than others. The example is:

```
Set Conn = Server.CreateObject("ADODB.Connection")
```

```
Conn.Provider = "Microsoft.Jet.OLEDB.4.0"
```

```
Conn.Open Server.MapPath("\a.mdb")
```

From an example, we will declare the variable by determining "Microsoft.Jet.OLEDB.4.0" in order to connect the database as Microsoft Access type. The position of database is named a.mdb that is located in the root directory of a server.

In addition to ADO can apply to the database as Microsoft Access, but it employs to other database types by settling the provider name. After it is settled, we can connect to a database. The lists of provider name are shown as Table 9.1:

Table 9.1. Provider Names to Connect a Database.

Provider Name	Database Connection Type
ADSDSOObject	Active Directory Services
Microsoft.Jet.OLEDB.4.0	Microsoft Jet database
MSDAIPP.DSO.1	Microsoft Internet Publishing
MSDAORA	Oracle database
MSDAOSP	Simple text files
MSDASQL	Microsoft OLEDB provider for ODBC
MSDataShape	Microsoft Data Shape
MSPersist	Locally saved files
SQLOLEDB	Microsoft SQL server

According to tests done by Wrox in their book ADO 2.0 Programmer's Reference, if we use an OLEDB connection as opposed to a DSN or DSN-less connection, we will see the following improvements as Table 9.2:

Table 9.2. Performance Comparison.

Performance Comparison					
SQL			Access		
	OLEDB	DSN		OLEDB	DSN
Connection Times:	18	82	Connection Times:	62	99
Iterating Time 1,000 Records Times:	2900	5400	Iterating Time 1,000 Records Times:	100	950

Note: these results are published on pages. 232, 233 in Wrox's ADO 2.0 Programmer's Reference. Times are in milliseconds and the iterating through 1,000 Records times were calculated using server-side cursors (there was only a minor



difference in performance between OLEDB & DSN recordset iterations when using client-side cursors.

## 9.2 ASP and ASP.NET Database Connection Code

In this research report, the writer will build both an Active Server Pages (ASP) page and an ASP.NET page, each of which generates an HTML page using data from a database. In both examples, the concept of data access is the same—they both involve a connection to the database. The difference is in the way the codes are employed to connect and retrieve data from a database. It is different between ASP and ASP.NET to connect a database. These are:

The principles of database connection of ASP are:

- (1) To connect a database as:

```
Set Conn = Server.CreateObject("ADODB.connection")
```

- (2) To open the database with identifying a Provider type:

```
Conn = "Provider=Microsoft.Jet.OLEDB.4.0; DataSource=D:\a.mdb"
```

- (3) To employ SQL commands to select data from a database as:

```
sql = "select * from member"
```

- (4) To employ the data retrieved with Recordset object

```
set RS = server.createobject("ADODB.Recordset")
```

```
sql = "select * from member"
```

```
RS.open sql,Conn,1,3
```

- (5) To close the database

```
Conn.close()
```

The principles of database connection of ASP.NET are:

- (1) To import a namespace

```
<%@ import namespace="system.data"%>
```



```
<%@ import namespace="system.data.oledb"%>
```

*Note: syetem.data: is the namespace to connecting a database.*

*system.data.oledb: is the namespace that inform the database connection on OLEDB Provider format. (If we want to change the connection method, we can change the namespace here).*

- (1) To employ the oledbconnection to link a database as:

```
dim myconn as new  
oledbconnection("Provider=Microsoft.Jet.OLEDB.4.0;_  
DataSource=D:\a.mdb")
```

- (2) To employ SQL commands to select data from a database as:

```
dim myda as new oledbdataadapter("Select * From  
book_phone",myconn)
```

*Note oledbdataadapter is to determinethe data that is retrieved from a database by using SQL commands.*

- (3) To employ the data retrieved with Dataset object as:

```
dim ds as new dataset()  
myda.fill(ds,"member")  
mydatagrid.datasource=ds.Tables("member")  
mydatagrid.databind()  
<asp:datagrid id="mydatagrid" runat = "server" />
```

- (4) No close because Dataset has the close property, therefore the system automatically close database connection.

Let see the example, there are 2 files as showdb.asp and showdb.aspx.

### 9.2.1 Creating the ASP Page

You can type the following code:

```
<HTML>

<HEAD>

<TITLE> To retrieve all Records to display on the web by Using ASP </TITLE>

</HEAD>

<BODY>

<%

Set Conn = Server.CreateObject("ADODB.Connection")

Conn = "Provider=Microsoft.Jet.OLEDB.4.0; Data_

Source=D:\Codes\IECASP\Phone_books.mdb"

set RS = server.createobject("ADODB.Recordset")

sql = "select * from book_phone"

RS.open sql,Conn,1,3

response.write "<table border = 1 >"

response.write " <tr><td >ID:</b></td><td>Company</b></td><td>Web</td><td>

Tel</td><td>Fax</td><td>Name</b></td><td>Email</td><td>Position</td><td>

Remark</td></tr>"

total = RS.RecordCount

do while not RS.eof

        response.write"<tr><td>"&rs.fields("ID")&"</td>

<td>"&rs.fields("Company")&"</td><td>"&rs.fields("Web")&"</td><td>"&rs.fields

("Tel")&"</td><td>"&rs.fields("Fax")&"</td><td>"&rs.fields("Name")&"</td>

<td>"&rs.fields("Email")&"</td><td>"&rs.fields("Position")&"</td><td>"&rs.fields

("Remark")&"</td></tr>"
```

```

rs.movenext

loop

response.write "</table>"

rs.close

set rs = nothing

%>

</BODY>

</HTML>

```

After you write the code above, save it as showdb.asp, and run it, the result is shown on Figure 9.3:

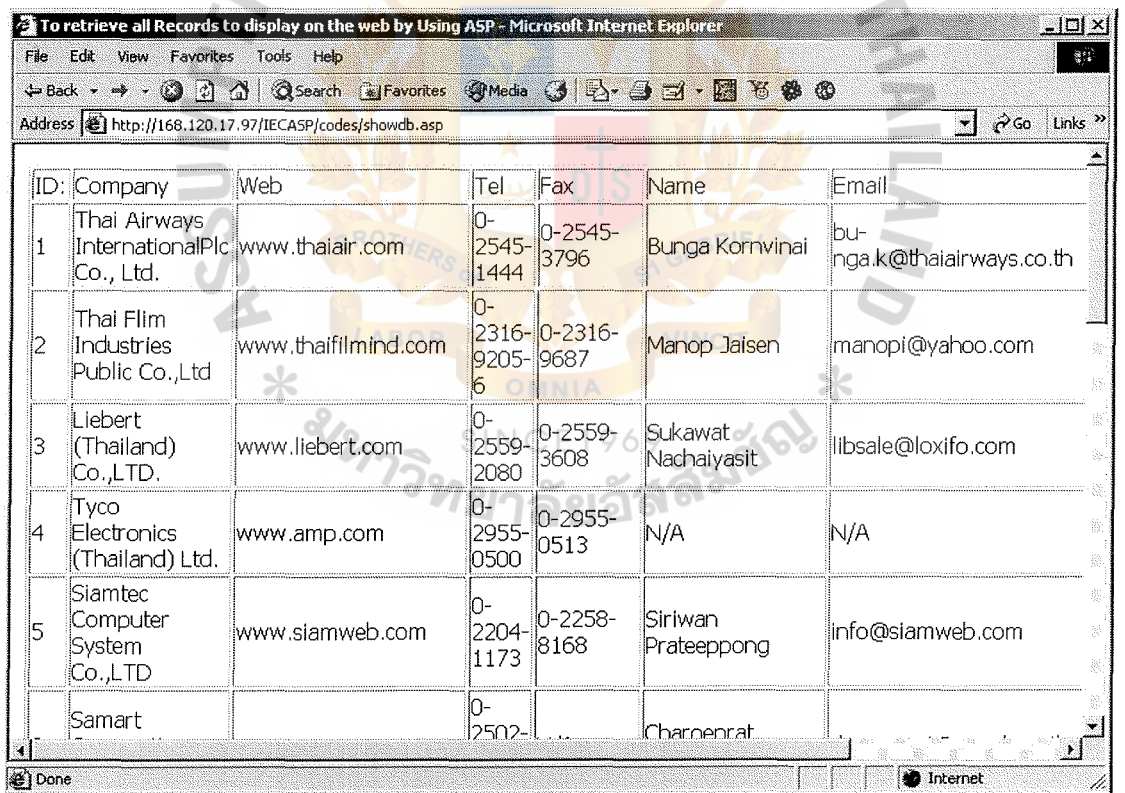


Figure 9.3. The Result from ASP.

### 9.2.2 Creating the ASP.NET Page

You can type the following code:

```
<%@ import namespace="system.data"%>

<%@ import namespace="system.data.oledb"%>

<HTML>

<HEAD>

TITLE> To retrieve all Records to display on the web by Using ASP.NET </TITLE>

</HEAD>

<BODY>

<script language="VB" runat ="server">

sub page_load(Sender As Object, E As EventArgs)

dim myconn as new oledbconnection("Provider=Microsoft.Jet.OLEDB.4.0;Data_

Source=D:\Codes\IECASP\Phone_books.mdb")

dim myda as new oledbdataadapter("Select * From book_phone",myconn)

dim ds as new dataset()

myda.fill(ds,"showphone")

mydatagrid.datasource=ds.Tables("showphone")

mydatagrid.databind()

end sub

</script>

<asp:datagrid id="mydatagrid" runat ="server" />

</BODY>

</HTML>
```



After you write the code above, save it as showdb.aspx, and run it, the result is shown in Figure 9.4:

ID	Company	Web	Tel	fax	Name	Email	Position	Rem
1	Thai Airways International Plc Co., Ltd.	www.thaiair.com	0-2545-1444	0-2545-3796	Bunga Kornvinai	bu-nga.k@thaiairways.co.th	Director-IT Services	-
2	Thai Film Industries Public Co., Ltd	www.thaifilmind.com	0-2316-9205-6	0-2316-9687	Manop Jaisen	manopi@yahoo.com	MIS	-
3	Liebert (Thailand) Co., LTD.	www.liebert.com	0-2559-2080	0-2559-3608	Sukawat Nachaiyasit	libsale@loxifo.com	M.D.	-
4	Tyco Electronics (Thailand) Ltd.	www.amp.com	0-2955-0500	0-2955-0513	N/A	N/A	N/A	N/A
5	Siamtec Computer System Co., LTD	www.siamweb.com	0-2204-1173	0-2258-8168	Siriwan Prateeppong	info@siamweb.com	Mkt. Mgr.	N/A
6	Samart Corporation	www.svoa.com	0-2502-6000	N/A	Charoenrat Vilailak	charoenrat@smark.co.th	CEO	N/A

Figure 9.4. The Result from ASP.NET.

You can see both files have the same result, but the difference is the code. ASP connects the database by using:

```
Set Conn = Server.CreateObject("ADODB.Connection")
Conn = "Provider=Microsoft.Jet.OLEDB.4.0; Data_
Source=D:\Codes\IECASP\Phone_books.mdb"
```

But ASP.NET connects the database by using the new technology of .NET as:



- (1) To bring Namespace to the code.

```
<%@ import namespace="system.data"%>  
<%@ import namespace="system.data.oledb"%>
```

- (2) To determine Connection String for linking a database

```
dim myconn as new  
oledbconnection("Provider=Microsoft.Jet.OLEDB.4.0;Data  
_Source=D:\Codes\IECASP\Phone_books.mdb")
```

- (3) To retrieve the data from a database

```
dim myda as new oledbdataadapter("Select * From  
_book_phone",myconn)
```

If you need to adjust the page to be good looking you can add the following code:



## ASP Code

```

response.write "<table border = 1 cellpadding=0 cellspacing=0
bordercolor=#D8D8D8 >"

response.write "<tr bgcolor=#993333><td <font face ='Verdana' color = #ffffff
size=2><b>ID:</b></font></td><td><font face ='Verdana' color = #ffffff size
=2><b>Company</b></font></td><td><font face ='Verdana' color = #ffffff size
=2><b>Web</b></font></td><td><font face ='Verdana' color = #ffffff size=2>
<b>Tel</b></font></td><td><font face ='Verdana' color = #ffffff size=2><b>Fax
</b></font></td><td><font face ='Verdana' color = #ffffff size=2><b>Name</b>
</font></td><td><font face ='Verdana' color = #ffffff size=2><b>Email</b>
</font></td><td><font face ='Verdana' color = #ffffff size=2><b>Position</b>
</font></td><td><font face ='Verdana' color = #ffffff size=2><b>Remark</b>
</font></td></tr>"total = RS.RecordCount

do while not RS.eof
    for num = 1 to total
        if num mod 2 = 0 then
            changecolor = "#eeeeee"
        else
            changecolor = "#ffffff"
        end if

response.write"<tr bgcolor="&changecolor& "><td><font face ='MS san sarif
size=2>"&rs.fields("ID")&"</font></td><td><font face ='MS san sarif
size=2>"&rs.fields("Company")&"</font></td><td><font face ='MS san sarif
size=2>"&rs.fields("Web")&"</font></td><td><font face ='MS san sarif

```



```

size=2>"&rs.fields("Tel")&"</font></td><td><font face ='MS san sarif'
size=2>"&rs.fields("Fax")&"</font></td><td><font face ='MS san sarif'
size=2>"&rs.fields("Name")&"</font></td><td><font face ='MS san sarif'
size=2>"&rs.fields("Email")&"</font></td><td><font face ='MS san sarif'
size=2>"&rs.fields("Position")&"</font></td><td><font face ='MS san sarif'
size=2>"&rs.fields("Remark")&"</font></td></tr>"

        rs.movenext

    next

loop

    response.write "</table>"

rs.close

set rs = nothing

```

ASP applies to HTML tags to modify the page format, and there is the add more one code:

```

for num = 1 to total
    if num mod 2 = 0 then
        changecolor = "#eeeeee"
    else
        changecolor = "#ffffff"
    end if

    .....

next

```



This is the condition that checks the even record row and add the light grey color to that row (record), otherwise it will be white color. The result from this above code is shown in Figure 9.5.

To retrieve all Records to display on the web by Using ASP - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back

Forward

Home

Search

Favorites

Media

Print

Stop

Reload

Go

Links

Address http://168.120.17.97/IECASP/codes/showdbdecorate.asp

ID	Company	Web	Tel	Fax	Name	Email	Position	Remark
1	Thai Airways International Plc Co., Ltd.	www.thaiair.com	0-2545-1444	0-2545-3796	Bunga Kornvinai	bu-nga.k@thaiairways.co.th	Director-IT Services	
2	Thai Film Industries Public Co.,Ltd	www.thaifilmind.com	0-2316-9205-6	0-2316-9687	Manop Jaisen	manopi@yahoo.com	MIS	
3	Liebert (Thailand) Co.,LTD.	www.liebert.com	0-2559-2080	0-2559-3608	Sukawat Nachaiyasit	lbsale@loxifo.com	M.D.	
4	Tyco Electronics (Thailand) Ltd.	www.amp.com	0-2955-0500	0-2955-0513	N/A	N/A	N/A	N/A
5	Siamtec Computer System Co.,LTD	www.siamweb.com	0-2204-1173	0-2258-8168	Siriwan Prateeppong	info@siamweb.com	Mkt. Mgr.	N/A
6	Samart Corporation Public	www.svoa.com	0-2502-6000-9	N/A	Charoenrat Vilailak	charoenrat@smark.co.th	CEO	N/A
7	APC Thailand	www.apcc.com	0-2264-2885-8	0-2264-2884	Sajji Fungkhajornkiat	aarsirak@apcc.com	Tech. Mgr	N/A
	Mobile Oil		0-					

Done Internet

Figure 9.5. A Page Modified by ASP.

### ASP.NET Code

```

<asp:datagrid id="mydatagrid"
borderwidth="2"
cellpadding="3"
headerstyle-backcolor="#993333"
headerstyle-font-name="Verdana"
headerstyle-font-bold="true"

```



```

headerstyle-font-size="10"

headerstyle-forecolor="#ffffff"

itemstyle-font-name="MS san serif"

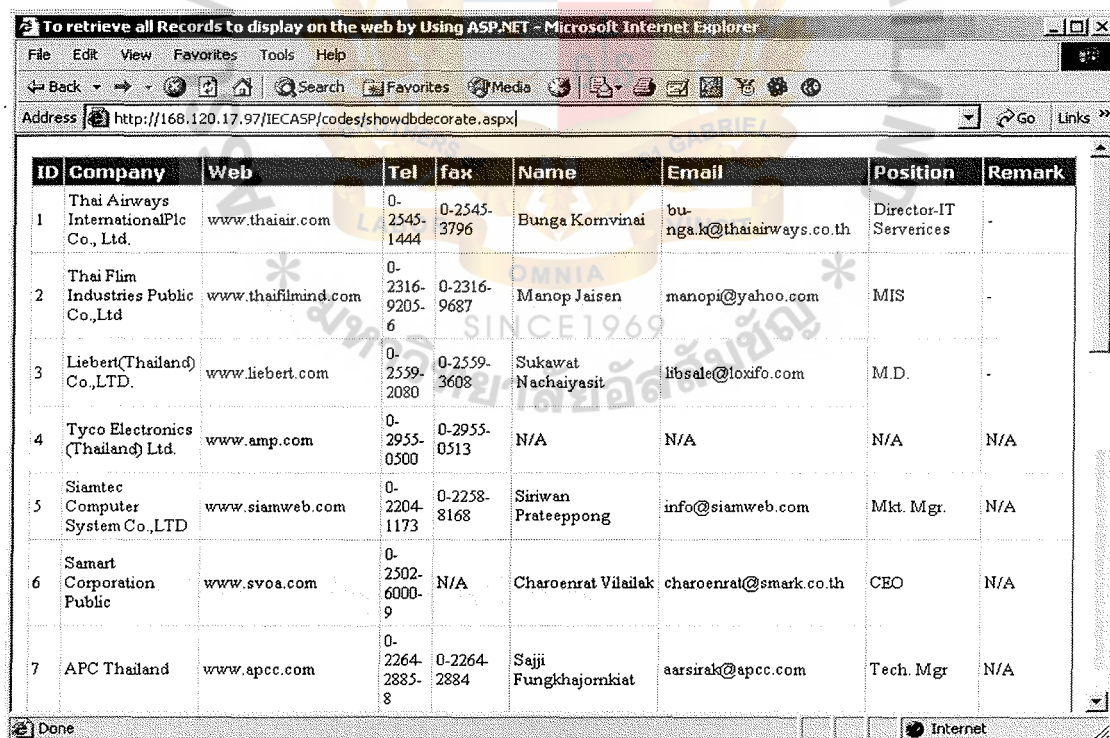
itemstyle-font-size="10"

alternatingitemstyle-backcolor = "#eeeeee"

runat = "server" />

```

You just add some attribute from Datagrid such as borderwidth, cellpadding, headerstyle-backcolor, headerstyle-font-name, itemstyle-font-name and etc. You can run it and see the result as Figure 9.6.



ID	Company	Web	Tel	fax	Name	Email	Position	Remark
1	Thai Airways International Plc Co., Ltd.	www.thaiair.com	0-2545-1444	0-2545-3796	Bunga Kornvinai	bu-nga.k@thaiairways.co.th	Director-IT Services	-
2	Thai Film Industries Public Co., Ltd	www.thaifilmind.com	0-2316-9205-6	0-2316-9687	Manop Jaisen	manopi@yahoo.com	MIS	-
3	Liebert(Thailand) Co.,LTD.	www.liebert.com	0-2559-2080	0-2559-3608	Sukawat Nachaiyasit	libsale@loxfo.com	M.D.	-
4	Tyco Electronics (Thailand) Ltd.	www.amp.com	0-2955-0500	0-2955-0513	N/A	N/A	N/A	N/A
5	Siamtec Computer System Co.,LTD	www.siamweb.com	0-2204-1173	0-2258-8168	Siriwan Prateppong	info@siamweb.com	Mkt. Mgr.	N/A
6	Samart Corporation Public	www.svoa.com	0-2502-6000-9	N/A	Charoenrat Vilailak	charoenrat@smark.co.th	CEO	N/A
7	APC Thailand	www.apcc.com	0-2264-2885-8	0-2264-2884	Saji Fungkhajornkiet	aarsirak@apcc.com	Tech. Mgr	N/A

Figure 9.6. A Page Modified by ASP.NET.



Again, both of result is the same but the difference is the code. ASP.NET applies to code is easy and short. But ASP has the complicated code. This causes to effect to the performance of the system. The shorter the code is, the better the execution is.



## X. CASE STUDY

### 10.1 Straight Ahead Ministries

The implementation of controls to affect every component of this new site except for the actual content has taken development and maintenance to a whole new level. Every aspect of the dynamic features has been done in a way that reduced development cost and will continue to keep the cost of maintenance to a minimum. StraightAhead.org will be a pleasure to maintain. -- Dan Wallace (DSW Consulting, LLC).

Straight Ahead Ministries approached DSW Consulting, LLC, an associate of ours to write a new web site for their organization. DSW Consulting approached us to handle the programming tasks that would be required of the web site. These tasks included login, bookstore, administration as well as a menu system that changes according to who is logged in. The problems are:

- (1) The ability for a user to Login and have the web site show and hide information based on who is logged in.
- (2) A bookstore that can list products and details of those products.
- (3) A shopping cart “swoosh” that is visible when there is something in the shopping cart but not visible when it is empty.
- (4) The ability to add pages to the site and to add them to the menu as they become available. Some of these pages will be restricted based on who is logged in.

The chief problem was to write the code in such a way as to only write it once but have it work through out the site in the same way.

There really are not many options given the requirements. They could write separate pages for each condition, write script in each page that handles all of the various conditions, or with the advent of user controls in ASP.NET, write a control for each major component and let it deal with the part of the site it is responsible for. The solution is ASP.NET and we opted to use the user controls.

The first step was to write a simple menu control that was dynamically created based on the content of an XML file. This XML file's controls that can see the menu item, the text of the menu item, and the URL the menu item will link to. By changing the XML file, we can change the menu. To add a new page, they can now simply change the XML file and post the page and the XML file.

Next they combined a series of menu controls into one main menu control which we dropped into each page. Because each section has a tree view, they created a tree control that also uses the same control as the menu control. This allows us to change one file and affect both areas.

For the "swoosh", we created a control that could display and hide itself based on the condition. We were then able to place that control on each page we needed it on.

When they run this web site, it all worked. The site is very easy to maintain. In fact, Straight Ahead Ministries thought that adding another sub category to the site would be a new project when it is really just maintenance of the existing site.

However, the only thing they should have improved is some way to create hyperlinks that were not dependent on a particular URL. Our next site will incorporate this. Their thought is to have all of the real URL references located in a file. This way, if a web designer needs to link to a page that does not exist he can create the link and add it to the file. When the page finally gets created, the reference

can be adjusted in the links file. Any references to it throughout the site will automatically be updated.

With ASP.NET (and JSP too) a properly designed site can have a lower cost of ownership because the maintenance costs are significantly reduced.

## **10.2 [www.netballnorthharbour.co.nz](http://www.netballnorthharbour.co.nz)**

How to turn a clunky, hard-to-find web site into a Force to be reckoned with? Netball North Harbour required a refreshed web site to promote their semi-professional team The Force and the use of their facilities on the North Shore. In particular they wanted to:

- (1) Improve web content and accessibility,
- (2) Improve interaction and communication with the target audience,
- (3) Increase membership and participation, and
- (4) Gain a website that is easy to use and maintain.

To reach the Netball community and fans of The Force, (typically school-aged girls) they also wanted to:

- (1) Allow access to the site from mobile devices such as WAP phones and other small devices,
- (2) Provide an alert or broadcast service for news and results via email or mobile SMS text messaging, and
- (3) Use additional graphics and animation to make the site more entertaining.

But there were some constraints: Netball North Harbour is a non-profit organization with limited resources to pay for hosting and maintenance. On the other hand they are users of Microsoft FrontPage. Because of these constraints it was clear that a packaged content management solution was not appropriate.



Synergy used its web creation process to produce an attractive, active site that is easy to use. Some key steps used in this process were:

- (1) Synergy Creative team produced several alternative conceptual designs that were presented to Netball North Harbour, and
- (2) Synergy worked closely with Netball North Harbour to design a simple, usable, and logical information model to build the site around.

To allow Netball North Harbour to effectively manage the site and its content, an administration console was developed that included the following features:

- (1) Document publishing engine with submission fields for document title, keywords, description, and validity dates,
- (2) Content administration engine for publishing news, events, and managing advertising, Microsoft FrontPage templates that allowed Netball North Harbour to develop content off-line to a consistent style before publishing, and
- (3) Site membership administration.

And though the site could have been developed using standard Microsoft architecture, Netball North Harbour agreed to allow Synergy to develop the website using Microsoft's new ASP.NET.

A "Forceful" presence is on the web. The web site is interesting, easy to use and already pulling the traffic. The main pages show event and news summaries, plus the requested topic in a clear and interesting style. And the administration console allows Netball North Harbour to manage all their content and advertising for the whole site quickly and easily.

The technology is pretty cool too. Combining Microsoft Visual Studio.NET's rapid application development environment, reuse, and VB.NET's object oriented

features have allowed Synergy to produce the site in less time than other technologies but without comprising on performance or quality.

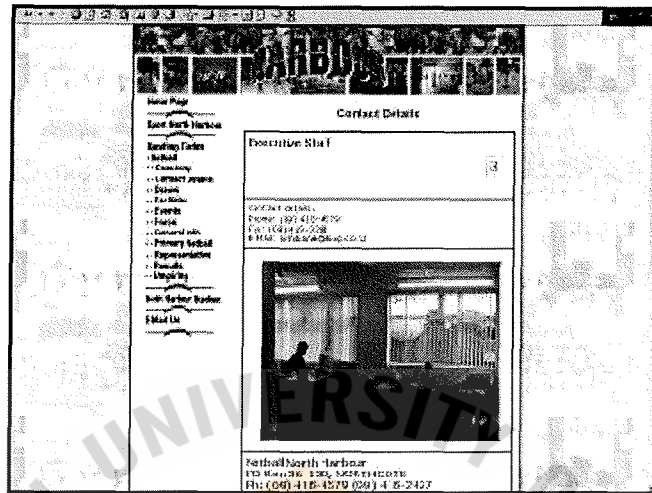


Figure 10.1. Original Site.

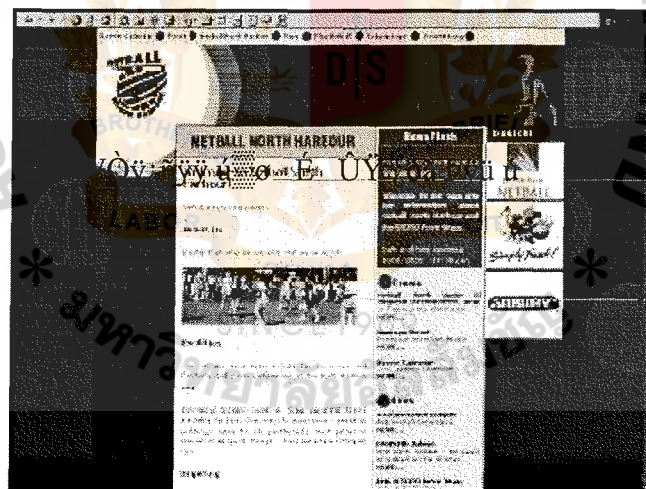


Figure 10.2. New Site.

The resulting site is hosted on a platform shared with other web sites and still performs quickly and reliably without affecting other systems on the server. .NET's simplified deployment model means site changes can be rapidly deployed without

shutting down the site or server. The old site was clunky and out of date as Figure 10.1. The addition of flash animation brings the new site to life as Figure 10.2. It is cool and funky.

Day to day monitoring and maintenance of an ASP.NET site uses the same Internet Service Manager as ASP on IIS 5. No new skills are required. ASP.NET also offers improved deployment and site management:

- (1) File copy deployment,
- (2) Simplified configuration using “web.config” files,
- (3) Declarative caching of web pages and controls for better performance,
- (4) Improved error handling, process isolation and system monitoring to allow the web site to recover from operational faults and even automatically restart when required,
- (5) Using XML Web Service across a firewall ensures that additional ports for handling DCOM and DTC calls do not need to be opened.

Some messages from users on this web site as:

- (1) Synergy has developed a wonderful site and has exceeded our expectations. Even for a computer illiterate like me, they make the development process very user-friendly. People have already started to comment on how great the site is. We are ecstatic! said Teresa Tairi Netball North Harbour Communications Manager and “Force” player.
- (2) “I’ve signed up as a member and can’t wait for the updates. With all the links I can print forms and email the necessary contact people.” said Carol Agnew Force Fan.

- (3) “I like the pictures and the colorful are really identifiable to Harbour. The layout is great and easy to get around, the site looks very professional”  
said D Hamilton Force FanNew Site Cool and funky.





## **XI. COMPARATIVE ANALYSIS: TOTAL COST OF ASP AND ASP.NET**

The cost of ASP and ASP.NET on this research report will concentrate on the performance, processing, programming model, and database connection, etc. These are the following:

- (1) Perfect Execution: ASP has the translation the code in the form of “Interpreter”- it means that it will work only the code line that it displays the result to a user on the screen. On the other hand, ASP.NET has the “Compiler” for the translation. It means that it will translate the whole code lines or a program.
- (2) Easy Programming Model: ASP.NET lets you build the great pages with far less code than with ASP.
- (3) Excellent components: The components will base on XML. The core thing is when we use ASP.NET’s components, we can upload a file to the directory of the administrator that he determines it. After that component will be set up automatically. This can reduce the problem of ASP because only an administrator will set up the ASP version of components. It is not convenient for a program to wait for him to set up the component.
- (4) To request the data from server: On ASP, the server can request the data from only the client, but on ASP.NET, the server can request the data from another server such as we can code the program to retrieve the stock data from the Yahoo web site to display on our homepage.
- (5) Effectiveness on database connection: There is no ODBC for database connection, but it will employ OLEDB.NET Provider. Microsoft develop

commands for MS SQL SERVER because it is fast for connection to the database and easy to manage it.

- (6) **Server Processing:** One of the big problems with ASP is that pages simply define one big function, which started at the top of the page and finished at the bottom. The page content is rendered in the page order, whether it is straight HTML or ASP-generated HTML. Therefore, our logic was dependent upon its position in the page, and there is no way to target HTML controls except by rendering them as part of the stream. Anything we do requires us to write code, and that includes the output of HTML elements. ASP.NET solves this problem by introducing a declarative, server-based model for controls. This is where the concept may seem alien to ASP programmers, because the controls are declared on the server, can be programmed against on the server, but can be event driven from the client. This sounds pretty weird, but it is simple to use.
- (7) **Database connection:** ASP.NET can improve the connection on the database more than ASP. ASP.NET uses OLEDB (Object Link Embedding Database) for the database connection. It provides the excellence on the database connection effectiveness and efficiency because OLEDB has the capability to connect the database based on DSN (Data Source Name) and DSNless, but it has the “provider” that can connect the database high-speed and communicate with relational database. Therefore, it can reduce the processes of the database connection and creating the object and closing it. ASP uses ODBC (Open Database Connectivity) to connect the database. It takes up the time because when ASP connects to the database, ODBC has to search for the DNS name on

the server and if it is found, it will check what the driver is. When ODBC gets the driver, it will use this driver to connect the database and return to ADO (Active X Data Object) that transfers that result to ASP and display on the screen.



## **XII. CONCLUSIONS AND RECOMMENDATIONS**

### **12.1 The Conclusions**

The research result indicates that the capability between ASP and ASP.NET and also their difference. It shows that ASP.NET has more the capability and powerful than ASP. The connection to a database on ASP.NET has the less code than ASP's code. This brings to the good performance of system's execution on the code.

The communication for static HTML works only one way. There is no way to send information back to a Web server. To fix this problem, forms were created. Forms are HTML tags that allow Web page creators to include controls like check boxes, and radio buttons in their Web pages. That way, the user can enter information. It also provides a Submit button that sends the information off to the server. But according to Internet, the server has to be smarter now. It cannot just get requests for pages and send out pages. The server has to know what to do with this information when it gets it. That is where ASP comes in.

An ASP is somewhat similar to a server-side. It can be put into an html page to make it responded dynamically to user requests. ASP allows you to take advantage of server-side scripting. Moreover, it can provide an array of objects and components that manage the interaction between the browser and the web server. ASP makes it much quicker and easier to create highly interactive Web sites. It also makes your pages easier to maintain and update in the future.

However, here are some problems of ASP such as:

- (1) It will process the connections all the time. This utilizes the resource of a server.
- (2) It can access only one database table at a time.



- (3) It cannot transfer the data file over a firewall.
- (4) It has no error detection explanation, etc.

Therefore, ASP.NET was developed to come to fix those problems. It is a new and extended technology to the earlier classic ASP. It can support, more languages than ASP such as XML, C#, etc. ASP.NET communicates with the database better than ASP and retrieves data faster than ASP because it can bring many benefits, especially improvements are needed most, namely, performance, power and flexibility, scalability, tool support, simplicity, and manageability.

- (1) Performance: when you apply to code by using ASP.NET is better performance than ASP because ASP.NET can execute only the code that it uses, not whole code like ASP does such as coding of connecting to a database to display on the web page.
- (2) Power and Flexibility: ASP.NET is language-independent, so you can choose across many languages to your web, but ASP has no capability.
- (3) Scalability: each programmer can use own language that he has the good skill and after that these code can be integrated to one web page. ASP.NET has the capability that can perform in clustered and multi-processor environments, but ASP has no those capability.
- (4) Tool Support; the tool supporting for ASP.NET is Visual Studio.NET. It is better than InterDev tool of ASP.
- (5) Simplicity: the employment of ASP.NET is easier than ASP because ASP.NET has the effectiveness tools.
- (6) Manageability: ASP.NET employs a text-based, a programmer can come to modify it without the aid of local administration, but ASP does not

support this. If you add the new components to a server, you do not restart it unlike ASP.

12.2 Recommendations

The research result indicates that ASP.NET has more capability than ASP. Both of them have the advantages and disadvantages. ADO.NET is much different than ADO. In order to achieve disconnected data access programmers have to use different techniques like disconnected recordsets such as ADO.NET. ADO object model is very small as compared to ADO.NET. ADO.NET provides number of specialized objects to handle very specific tasks. Microsoft has taken care to closely map properties and methods of ADO.NET objects with existing ADO counterparts. As per Microsoft ADO.NET is not a replacement for ADO but an enhancement in the overall data access technology.

Table 12.1. ASP Hardware Specification.

Hardware	Minimum	Recommend
CPU	Intel Pentium 90 MHz above	Intel Pentium 166 Mhz better
RAM	32 MB	More 64 MB is better.
Hard Disk	250 MB is available.	More available is better.

Table 12.2. ASP.NET Hardware Specification.

Hardware	Minimum	Recommend
CPU	Intel Pentium II 300 MHz	Intel Pentium 600 MHz
RAM	96 MB	128 MB
Hard Disk	560-600 MB	1 GB.

We can use both ADO and ADO.NET in our application. A user chooses either ASP or ASP.NET depending on the purpose, the task, the hardware and software specification. For an example, if he does not have many databases, he can apply to ASP to connect to the database, or the high-speed performance is needed, ASP.NET

can be applied. Other important point is the hardware specification, if we do not have enough hardware, we can not employ it. Their conclusions are shown on Table 12.1 and Table 12.2.

For the software, ASP.NET can be executed on Windows 2000 or Windows XP or more versions. On the other hand, ASP can be executed on Window 95 and 98 with installed the Person Web Server program, Windows 2000 and Windows NT with installed Service Pack 3.



## BIBLIOGRAPHY

### English References

1. Ahmed, Mesbah, et al. ASP.NET: Web Developer's Guide. Rockland, MA: Syngress, 2002.
2. Alexander, John and Billy Hollis. Developing Web application with Visual Basic.net and ASP.net. New York: J. Wiley, 2002.
3. Anderson, Richard, et al. Professional ASP.NET. Birmingham: Wrox Pr., 2001.
4. Birdwell, Rob, et al. Beginning ASP.NET: Using VB.NET. Birmingham: Wrox Pr., 2001.

### Thai References

1. จำลอง ครูอุตสาหะ. ASP.NET ฉบับโปรแกรมเมอร์. กรุงเทพฯ: เคทีพี คอมพ์ แอนด์ คอนซัลท์, 2545.
2. ทวีชัย หงษ์สุมาลย์ และ สกวนชัย สุวรรณชีวะศิริ. อินไซด์ ASP และ ASP.NET ฉบับสมบูรณ์. กรุงเทพฯ:บริษัท โปรวิชั่น จำกัด, 2545.
3. สุรัตน์ บันทิตลักษณ์. เก่ง ASP.NET ให้ครบสูตร. กรุงเทพฯ: บริษัท วิดีโอ กรุ๊ป จำกัด, 2544.
4. สัจจะ จรัสรุ่งเรือง และ สมพร จิวรสกุล. เริ่มต้นอย่างมืออาชีพด้วย ASP และ E-Commerce ฉบับสมบูรณ์. กรุงเทพฯ: อินโฟเพรส, 2543.



