CONTENTS

WEB DESIGN USING ASP

SYLLABUS

UNIT I

Introduction to ASP: What is ASP? - ASP Model - Scripting Languages - Delimeters single expressions-statements - Including other files. Understanding objects: Application object-lock - Unlock - Events-application on end - application on start - Request object - Properties of the Response object - Methods of the Response object - Session object - The globall.asa file.

UNIT II

Understanding components: The advertisement rotator component - The text stream component - Properties of the textstream object. Working with users; the input function - Retrieving from data - Using text boxes and text areas.

UNIT III

Cookies: Working with Cookies - Application of Cookies - Created by ASP page - Drawbacks of using cookies - Web Browser Compability Issues - Using Cookies in ASP Applications - An ASP application that uses cookies.

UNIT IV

Working with files and the File system: Considering Performance and Data Protection - Executing a SQL. Statement with the connection Object - Understanding session and connection pooling - Working with recordsets - Recordset cursor and locking types - Understanding ADO cursors - Advanced Methods and Properties of the Recordest Object - Paging Through a recordset. Working with the command object. Creating stored Procedures - Executing stores procedures with the connection object - Receiving Parameter information.

UNIT I

LESSON

1

INTRODUCTION TO ASP

CONTENTS

1.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Understand active server pages
- Understand client-server model
- Discuss running ASP pages
- Discuss using ASP without IIS or PWS
- Discuss creating your first web page
- Understand ASP scripts
- Discuss the ASP process
- Discuss what is variable
- Understand the big if and for-next loops

1.1 INTRODUCTION

ASP was "born" in November 1996 when Microsoft announced its design of an Active Platform. The Active Platform reflects Microsoft's ideas about how a desktop computer and a server computer should communicate. It consists of two parts: the Active Desktop and the Active Server. The Active Desktop refers to the client side, or the user's side, where HTML files are displayed on a web browser. The Active Server refers to the server-side component. This consists of pages that can be interpreted by the server, hence the term Active Server Pages.

Over the past couple of years, we have seen some major changes concerning the Internet. Initially, the internet served as a medium for members of government and education institutions to communicate. With the advent of the World Wide Web, the internet became a multimedia, user friendly environment. Originally the internet served as a place for enthusiasts to create personal home pages, but as more people began going online the internet transformed into an informational resource for the common man.

1.2 ACTIVE SERVER PAGES

As the internet has matured into a viable market place, Web Site design has changed in step. In the early days of the World Wide Web, HTML was used to create static Web pages. Now you can create dynamic Web pages in many ways. Microsoft solution to building dynamic Web pages is through the use of Active Server Pages (ASP).

Active Server Pages contains two parts:

- Programmatic code
- Embedded HTML

The programmatic code can be written in a number of scripting languages. (Scripting language is a particular syntax used to execute commands on a computer)

Some popular Web related scripting languages include VBScript and JavaScript. When creating an ASP page, you can use one of four programming languages:

- VBScript-Similar to visual basic syntax, the most commonly used scripting language for active server pages.
- Jscript-Similar to java script.
- PerlScript-Similar to Perl.
- Python-A powerful scripting language commonly used for Web development.

Most ASP pages are created using VBScript as it has the most English-like syntax of the four scripting languages and is similar to Visual basic syntax.

Asp page contain embedded HTML. This allows for existing static Web pages to be easily converted into dynamic ASP pages. An ASP page must contain an .ASP extension.

Note: Many large Web sites use active server pages to serve dynamic Web content. For example, Buy.com and Dell.com use active server pages to build interactive, dynamic Web sites.

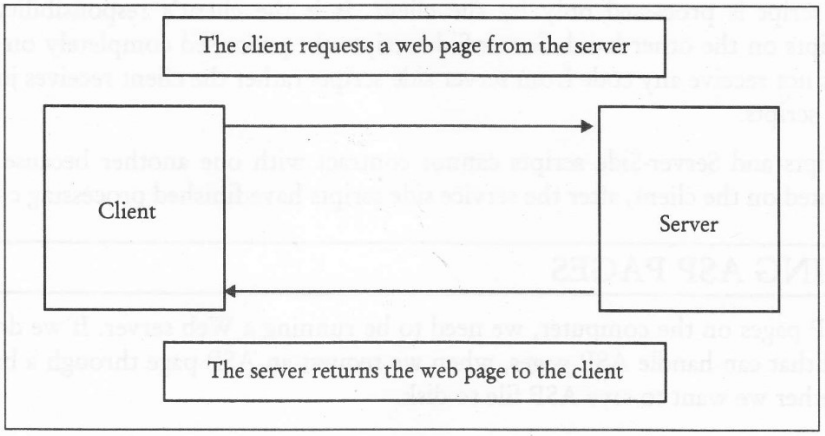
1.3 UNDERSTANDING CLIENT-SERVER MODEL

In a *client-server model,* two computers work together to perform a task. A client computer requests some needed information from a server computer. The server returns this information and the client acts on it. The internet runs on a client-server model. With internet the server is a particular Web server and the client is a Web browser.

A web server is a computer that contains all the web pages for a particular web site and has special software installed to send these web pages to web browsers that request them.

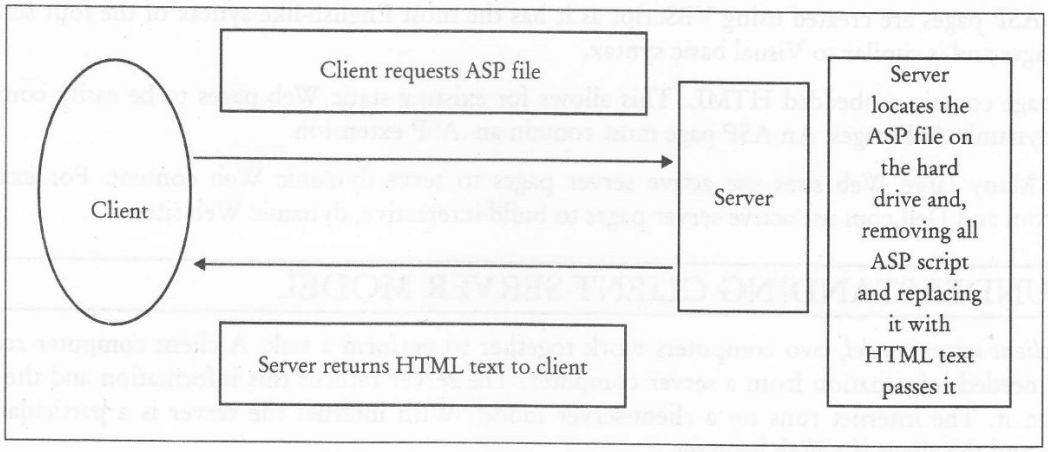
The following steps occur when a web browser visits a static web page:

- The client (web browser) locates the web server specified by the first part of the URL.
- The client then requests the static web page specified by the second part of the URL(/index.htm).
- The web server sends the contents of that particular file to the client in HTML format.
- The client receives the HTML sent by the server and renders it for the user.



When a web browser requests an ASP page, the following steps occur:

- The client locates the web server specified by the first part of the URL
- The client then requests the ASP page specified by the second part of the URL(/default.asp)
- The web server reads the ASP file and processes the code.
- After the ASP page has been completely processed by the web server, the output is sent in HTML format to the client.
- The client receives the HTML sent by the server and renders it for the user.



How ASP differs from Client-Side Scripting Technologies?

Client-Side Scripting is programmatic code in an HTML file that runs on the browser. It is denoted by <SCRIPT> HTML tag. It is commonly written using the JavaScript programming language due to the fact that Netscape Navigator only supports the JavaScript scripting language for Client-Side scripting. ASP scripts are server -side scripts.

While using ASP, its code exists on the server only. ASP code, which is surrounded by the <% and %> delimiters is processed completely on the server. The client cannot access this ASP code.

Differences

A Client-Side script is processed only by the client. It is the client's responsibility to execute all Client-Side scripts on the other hand; Server-Side scripts are processed completely on the web server. The client does not receive any code from server side scripts rather the client receives just the output of the Server-Side scripts.

Client-Side scripts and Server-Side scripts cannot contract with one another because the Client-Side script are executed on the client, after the service side scripts have finished processing completely.

1.4 RUNNING ASP PAGES

To execute ASP pages on the computer, we need to be running a Web server. If we don't have a Web server installed that can handle ASP pages, when we request an ASP page through a browser, a dialog is received whether we want to save ASP file to disk.

A Web server capable of handling ASP pages processes the request to ASP page before it is sent to the client. During this processing two things occur:

- The programmatic code in the ASP page is interpreted by the Web server.
- The Web server informs the browser that it is sending HTML information, at which point the Web server sends the output of the ASP page. The browser, receiving this raw HTML renders it for the user.

If one does not have a Web server installed and tries to view an ASP page through the browser, the second step is not accomplished. That is, the browser isn't informed that the ASP page contains HTML code. The browser does not know what to do with .ASP file. If web server is installed and has associated the .ASP file extension with itself, the program is launched. Example: Both Visual InterDev and Microsoft Front page associate the .ASP extension. If you have one of these programs installed on the computer and an attempt is made to view an ASP page through the browser before the installation of web server, Visual InterDev is launched automatically. If however, there are no programs installed on the computer that associate the .ASP file extension then one is prompted with the dialog.

1.4.1 Setting up a Personal Web Server

When creating a professional website, it is important that a web site run on computer that has Windows NT Server or Windows 2000 installed. Microsoft created a stripped-down version of its professional Web server called Personal Web Server (PWS) which is intended to run on Microsoft Windows 95 or 98 or Microsoft Windows NT workstation.

After the installation of PWS run the Setup program. Next step is what type of installation to perform. We can customize what components and documentation to install by selecting the Custom option. The default options are the most suitable so we recommend just selecting the typical installation.

After the selection of the Typical or Minimum installation a dialog prompts us to enter the directory to select as our default Web publishing home directory. If we choose to perform the Custom install, we will see this screen after we select which components to install.

Example: A Web page is requested from a Web server with a URL,

http://www.InventoryREsellers/Inventory/ShowInventory.asp

The first part of the URL i.e. www.InventoryREsellers.com is the domain name. This name identifies what Web server this Web site exists on. The remainder of the URL determines what directory and file the user is requesting.

In the above example, user is requesting a file named ShowInventory.asp from the /Inventory directory. /Inventory/ShowInventory.asp is referred to as the virtual address. The Web server needs to map the virtual address to a physical address. Also there is need to specify the root physical address. After the selection of Web's root physical address, the setup program starts installing the needed files to the computer. When the installation is complete, a need to reboot the computer arises.

After rebooting a new icon appears on the System Tray, double clicking on which brings up the Personal Web Manager dialog which contains five panes each pane presenting different information and serving a different purpose. The default pane is the Main pane which contains all the statistical information about the Web site's activity e.g. stats on requests per hour, requests per day, visitors per hour etc.

Another useful pane is the Advanced pane which contains the Web site's directory structure and three Web site properties that can be altered. The first, Enable Default Document, determine whether to load a default document if the user does not request a specific file in the URL. Secondly, it allows Directory Browsing. Lastly Advanced pane has the option of Save Web Site Activity Log.

1.4.2 Setting up Internet Information Server

Internet Information Server (IIS) is Microsoft's professional Web server. The latest version of IIS is 5.0 which ships with Windows 2000. To install IIS 5.0 following steps are done:

- Choose start, programs, administrative tools, configure your server.
- A dialog box appears that has several configuration options in the left pane. Choose the bottommost option from the left pane labeled Advanced.
- On clicking this four new options appear: Cluster service, Message Queuing, Support Tools and Optional Components. Click the last option. In the right hand pane, a description of the Option Components option appears, as well as a hyperlink labeled start.
- Click the Start hyperlink-this launches the Windows Components Wizard. Through this wizard you can install or uninstall optional components. Scroll down till the Internet Information Services (IIS) option is seen.
- To decide what IIS components to install, click IIS components in the Windows 2000 Components Wizard and then click the Details button. A list of components that can be installed with IIS is displayed.
- Common Files, Documentation, Internet Information Services Snap-In and World Wide Web Server should surely be selected.
- After the selection of all the IIS subcomponents that are to be installed, click OK, which takes you back to the Windows 2000 Components Wizard.
- To start installing IIS 5.0, click the Next button. When the installation is complete, one can access the Internet Services Manager.

1.4.3 Using ASP without IIS or PWS

ASP Pages can run even without IIS or PWS as your Web server. A couple of companies have created software to allow ASP pages to run on various Web servers and platforms.

One of these products is Halcyon Software's Instant ASP, abbreviated as iASP. Another product created Chili! Soft, is Chili! ASP. These products can run on many non-IIS Web servers as the following:

- Apache
- Sun Web Server
- Java Web Server
- Netscape Enterprise Server.

These products can also run on a number of platforms, including

- Linux
- Sun Solaris
- Apple Mac OS
- IBM/AIX.

1.4.4 Creating Your First Web Page

To create ASP pages, we need access to a computer with a Web server that supports Active Server Pages technology. In running ASP pages we need to install two Microsoft Web server: Personal Web Server and Internet Information Server. After the Web server has been installed, we can create ASP pages in our Web site's root physical directory, or in subdirectories of the root physical directory and view the result of these ASP pages through a standard Web browser. Because ASP pages are processed completely on the server- side and only return HTML to the client, any Web browser can be used to view ASP pages. There are no restrictions on the client-side. VBScript scripting language is the most commonly used scripting language for ASP pages.

Example: imagine that depending on the time of the day we want a Web page to display a different message. If the time is 11:00 AM, we want to display Good Morning!, whereas the time is 5:00 PM, you want to display Good Evening!. Using static HTML pages we would have to edit the HTML page twice a day- once before noon and once after, altering the Web page and changing its message. However with ASP pages, we can use programmatic code to determine the current time and display a custom message based on the time.

Displaying a different message depending on the time of day

```
<%@ Language=VBScript %>
<%  Option Explicit  %>
<HTML>
<BODY>
The current time is  <%=Time ()%>
<P>
<%
If DatePart ("h", Time()) >=12 then
'Is it afternoon
Response. Write "Good Evening!"
Else
'Is it before noon
Response. Write "Good Morning!"
End if
%>
</BODY>
</HTML>
```

1.5 WRITING ASP SCRIPTS

1.5.1 .ASP File

Active Server Pages file is the one where its file name includes .asp as its extension. An .asp file is defined as a text file and can be a blend of the following:

- Text
- HTML tags
- Script commands. A script command informs your computer to do something, like assigning a value to a variable.

For creating an .asp file, rename any HTML file, replace the existing .htm or .html file name extension with .asp. Save the new file in a Web publishing directory to make the .asp script file available to Web users. When you see the file with your browser, you will see that ASP processes and returns HTML, same as before. When you add scripts to your HTML, ASP really begins to work for you.

1.5.2 What is a Script?

Script means a small relatively limited interpreted language. A script is defined as a spring of script commands. For example, a script can allocate a value to a ***variable.*** A variable is a named storage location that can include data, like a value.

Inform the Web server to send something, such as the value of a variable, to a browser. An instruction that sends a value to a browser is an ***output expression.*** Commands can be combined into ***procedures.*** A procedure is a named series of commands and statements that acts as a unit.

Implementing a script sends the sequence of commands to a scripting engine, which interprets and transmits them to your computer. Scripts are written in languages which are having specific rules; so, if you want to use a specified scripting language, your server must run the scripting engine that is familiar with the language.

There are two ways to make the translation:

- We can translate the code at runtime, which means the computer reads a line or block of code from our code file, which makes the translation and then executes the code. This process is called interpreting the code and its done by a program called an interpreter.
- We can translate the code and store the resulting code in a file. When we execute a file the computer reads the machine instructions directly. This kind of translation is not performed at runtime, a program called a compiler translates the code before the computer begins to execute it. The compiler and the interpreter do much the same thing, but the compiler is more efficient because it is not under any time constraint. It compiles the code off line. The interpreter is less efficient, because it has to translate the code to machine instructions and run it almost instantaneously.

We need to know what interpreted code is. Computers don't understand code as we write it. Instead, other programs translate the code we write into machine instructions.

ASP gives scripting engines for the VBScript and JScript scripting languages. VB Script is a ***primary scripting language*** that is, the language that ASP presumes that you are using if you don't specify a language.

1.5.3 ASP Syntax

ASP provides a situation that processes scripts that you encompass into your HTML pages. That is ASP is not a scripting language. You are required to learn the **syntax**, or rules, by which ASP operates. This is necessary to use ASP successfully.

Delimiters

A delimiter is a series of characters that indicate the beginning or end of a unit. HTML tags are distinguished from text by ***delimiters***. These delimiters are the less than (<) and greater than (>) symbols, in case of HTML

Also, delimiters differentiate ASP script commands and output expressions from both text and HTML tags. The delimiters <% and %> are used by ASP to enclose script commands. For example, the command <% music= "Guitar" %> assigns the value guitar to the variable music.

ASP uses the delimiters <%= and %> to encircle output expressions. That is, the output expression <%= music %> sends the value guitar (the current value of the variable) to the browser.

Single Expressions

ASP delimiters can contain any expression which is justifiable for your primary scripting language. For example, the following line produces text ending with the current server time:

This page was last refreshed at <%= Now %>.

Here, the Web server returns the value of the VBScript function **Now** to the browser together with the text.

Statements

In VBScript and other scripting languages, a ***statement*** is a complete unit that represents one kind of action, declaration, or definition. The conditional **If...Then...Else** statement is a common VBScript statement that is given below:

```
<%
If Time >=#12:00:00 PM# And Time < #04:00:00 PM#  Then
  greeting = "Good Afternoon!"
Else
  greeting = "Hi!"
End If
%>
```

This statement holds either the value "Good Afternoon!" or the value "Hi!" in the variable greeting. No values are sent to the client browser. The lines shown below send the value, in blue, to the client browser:

```
<FONT COLOR="BLUE">
<%= greeting %>
</FONT>
```

So, a user viewing this script before 03:00 PM (in the time zone) would see

Good Afternoon!

A user viewing the script at or after 03:00 PM would see

Hi!

Including HTML in a Statement

HTML text could be included between the sections of a statement. For example, the script shown below, which combines HTML within an **If...Then...Else** statement, gives the same result as the script in the previous section:

```
<FONT COLOR="BLUE">
<% If Time > = #12:00:00 PM# And Time < #03:00:00 PM#  Then %>
Good Afternoon!
<% Else %>
Hi!
<% End If %>
</FONT>
```

If the condition is true, then the Web Server sends the HTML that follows the condition ("Good Afternoon") to the browser; otherwise, it sends the HTML that follows **Else** ("Hi") to the browser.

Script Tags

The statements, expressions, commands, and procedures that are used in script delimiters must be justifiable for the default primary scripting language. ASP is provided with the default primary scripting language set to VBScript. You can use other scripting languages with ASP; HTML script tags <SCRIPT> and </SCRIPT> are used, in combination with the LANGUAGE and RUNAT attributes, to encircle complete procedures written in any language for which you have the scripting engine.

For example, the following .asp file processes the JScript procedure **MyFunction**.

```
<HTML>
<BODY>
<% Call MyFunction %>
</BODY>
</HTML>
<SCRIPT RUNAT=SERVER LANGUAGE=JSCRIPT>
  function  MyFunction ()
  {
```

```
    Response.Write("MyFunction Called")
  }
</SCRIPT>
```

Including Other Files

This is a mechanism which is used to insert information into a file before processing. ASP implements only the **#INCLUDE** pre-processing directive. This directive can be used to insert the content of another file into an .asp file. The following syntax is used:

```
<!--#INCLUDE VIRTUAL|FILE="filename"-->
```

where you must type either **VIRTUAL** or **FILE**, which are keywords that indicate the type of path which is used to include the file, and ***filename*** is the path and file name of the file you want to include.

Using the Virtual Keyword

The Virtual keyword is used to show a path beginning with a ***virtual directory***. For example, if a file named app.inc is located in a virtual directory named /Myapp, the following line would insert the contents of app.inc into the file containing the line:

```
<!--#INCLUDE VIRTUAL="/myapp/app.inc"-->
```

Using the File Keyword

The File keyword is used to show a ***relative*** path. A relative path starts with the directory that invloves the including file. For example, if you have a file in the directory Myapp, and the file Header1.inc is in Myapp\Headers, the following line would insert Header1.inc in your file:

```
<!--#INCLUDE FILE="headers/header1.inc"-->
```

Including Files: Tips and Cautions

An included file can contain other files. An .asp file can also contain the same file more than once, provided that the <INCLUDE> statements do not create a loop. For example, if the file file1.asp includes the file file2.inc, file2.inc must not in turn include file1.asp. ASP detects such loops are detected by ASP, produces an error message, and stops processing the requested .asp file.

ASP files are included before implementing script commands. So, a script command cannot be used to build the name of an included file. For example, the script shown below would not open the file Header1.inc because ASP attempts to implement the **#Include** directive before it assigns a file name to the variable name.

```
<!-- This script will fail -->
<% name=(header1 & ".inc") %>
<!--#include file="<%= name %>"-->
```

Scripts commands and procedures must be entirely contained within the script delimiters <% and %>, the HTML tags <SCRIPT> and </SCRIPT>, or the HTML tags <OBJECT> and </OBJECT>. That is, you cannot open a script delimiter in an including .asp file, then close the delimiter in an included file; the script or script command must be a complete unit. For example, the following script would not work:

```
<!-- This script will fail -->
```

```
<%
For i = 1 To n
  statements in main file
  <!--#include file="header1.inc" -->
Next
%>
The following script would work:
<%
For i = 1 to n
  statements in main file
%>
<!--#include file="header1.inc" -->
<% Next %>
```

1.5.4 The Separation of Script by the Server from Content

When the ASP engine parses the file, it separates server side script from content in two ways:

- As a programmer we must separate our script from content with a delimiter, which is an arbitrary character or sequence of characters that separates values. Delimited text files often have only a single delimiter - the comma or tab character. ASP script has two delimiters, one to mark the beginning of a code block and another to mark the end of code block. The ASP code start delimiter is a left angle bracket and a percent sign, and the end delimiter is the percent sign followed by a right angle bracket : for example <% your code here %>
- We may also separate code from script using a more traditional, HTML (Hyper text marking language) like syntax. Using this method, the code begins with a </script> tag. To use this syntax with ASP script, we write the script as follows :

```
<script language="Jscript" RunAt="server">
'your code here
</script>
```

We note the text RunAt="server" in the <script> tag. If you don't include the text the server will ignore the script, but the browser will try to execute the script on the client side. In other words <script> tags by default, delimit the clients script. To force the ASP engine to process scripts as server side code we must add the RunAt="server" attribute.

1.5.5 How does the Server process Script?

When the server invokes the ASP scripting engine, it parses the file and inserts any include files. It then collects the script portion by finding all the script delimiter. The ASP treats all non script contents as strings that it writes verbatim back to the client. Next, the scripting engine creates variables and external components invoked by the script. Finally, it begins the script, executing each command and sending results to the client.

1.6 THE ASP PROCESS

ASP process helps us perform several operations that are difficult or impossible with straight HTML. The goal of a website and application is to respond to a client as quickly as possible making the site an interactive experience rather than an exercise in patience.

Make If ...Then decisions:

1. Most programming is about making simple decisions. For example, when a user selects an item from a drop down list our application will do one thing or another depending upon which item is selected. That implies that our application can make a decision. In pseudo code:

```
If user clicked TX then
'Save users selection in a variable
Session("LastUsers selection") = "TX"
Show list of state parks in texas
Elself user clicked TN then
'Save user's selection in a variable
Session ("LastUser selection") = "TN"
Show list of state parks in Tennessee
End If
```

   The code shows that we are going to make a decision based on which state abbreviation are user selected from a list of states. We want to save the user selection for future use, and then display a new list of state parks for selected state.

   Similarly, when the user return to the list after looking at the state park list, we would probably want to leave the list in the same state it was in when the user left it - with wither TX or TN selected.

2. ***Processing of Information From Clients:*** ASP provides several native objects that let us access data sent by client. We can store data, make decisions, alter the data, send it back to the client - basically we can do whatever we want. The ASP objects makes it easy to process client data.

3. ***Access Database and Files:*** ASP by itself has no database connectivity or file reading abilities, however we can use the ADO objects installed with ASP to access the database.

4. ***Format Responses:*** With ASP we can respond differently to different clients. This means that we can take the results of database query or data from a file, format it on a fly, and return the result to a client. For example, we can hide or show data based on clients' permission level. In a student report card application, for example, we would want to let teachers and school administrator view and change data for their own students. But we would not want to let students change their data. Similarly, we would want students to see their own report cards, not those of other students.

   With ASP we can use the same data set for all users, but format the responses based on identity. We can also easily change the look and feel of the response, change colours and controls, adjust to multiple screen resolutions and more, by running conditions specific code on the server.

5. *Launching and Communicating with Com Objects:* ASP can launch Component Object Model (COM) objects. A COM object can contain almost any kind of functionality, but all COM objects have one thing in common - they can communicate with one another.

6. *Control Transactions:* We can use ASP to create mission critical transactional applications. A transaction is any set of tasks that must either all complete successfully or fail completely. An example is a financial transaction. When we pay by a cheque or a credit card, the bank must remove the money from our account and credit the merchant's account. If either part fails, we want both parts to fail. Forcing failure is a difficult process, because it means we have to undo the parts of the transaction that have already been completed.

1.6.1 What does Response/Write do?

When a browser requests a data from the server, the server responds, with a redirect message, the requested data, or an error.

The response has two main parts:

- The Response header contains directives or information about the impending content, such as the type of content, the expiration date, etc. The response body contains the data.
- For HTML requests, the response data comes directly from the web server, which reads and returns the contents of an HTML page. For ASP pages, the response data comes from the ASP object called the response sheet.

Property/Method	Description
AddHeader Method	Adds a header value to the returned page.
AppendToLog Method	Appends a string to the Message field of the IIS log
Binary Write Method	Used to write non-textual data to the browser. We can use this method to send image, audio, or other binary to the browser.
Buffer Property	Buffers the character stream sent to the browser until page processing completes.
Cache Control Property	Controls whether proxy servers should cache the page
Charset Property	Controls the character set the browser uses to render the page. The default is ISO=LATIN-1
Clear Method	Clears the contents of the server response Buffer.Ifyou are in the middle of processing a page and some condition causes you. To want to restart the page you can clear the buffer by issuing the Response. Binary Write Method to send binary data.
Cookies Collection	You use this method to write both per session and persistent cookies to client browser.
End Method	Ends the response. This command takes effect immediately. The server will not process any code or HTML that follows the response. End command.
Expires Property	Sets the time interval, in minutes, before the page expires. Until the specified period elapses the browser may redisplay the page from cache. After the specified period. The browser must return to the server to redisplay the page.
Expires Absolute Property	Sets an exact time when the page expires. Before the specified time, the browser may Re-display the page from cache.

Contd...

Flush Method	Sends any buffered content to the browser we might want to do this for long pages Note that that the Response. Flush method raises an error if you have not set the response. Buffer to True.
Pics Property	Adds an HTTP header value containing a platform for internet content selection (PICS) label.
Redirect Method	Sends a header to the browser telling it to request a new page. You specify the URL for the page along with any Query String variables and values.
Status Property	Sets the value of the status line returned by the server. You may have seen this error before-404 not found. You set the status properly to return a specific number and explanation to the browser. The HTTP specification defines the set of valid status values and their meanings.
Write Method	Writes text and HTML content to the browser.

1.6.2 What does the ASP Script Return to Browser?

The process the web server uses to return a response to the browser is complex, but returning a response using the Response object could not be easier. We return a response by using the write method of the response object:

```
<%
Response.Write''The Response. Write Method is great!"
%>
```

Note the bracket percent (<%) delimiters around the code. The delimiters separate script from the content in an ASP page.

1.6.3 The Response.Write Method

The <%=Shortcut, What's the <%@ LANGUAGE=VBSCRIPT%>?

```
<%@ Language=VBScript%>
 <html>
 <head>
 </head>
 <body>
 <%
 Response. Write "The response. Write method is great!"
 %>
 </body>
 </html>
```

Two features differentiate this file from a standard HTML file. First, the top line in the file, <%@ language=VBScript %>, tells the ASP engine that the default scripting language used for the page is VBScript(the @ sign is required and denotes the script directive , as opposed to script code). Second; the Response.Write line contains the content. The content is not in HTML; we now control the content with script. We can change the script to return any response we want.

1.7 WHAT IS VARIABLE?

A variable is used to store information.

1.7.1 Storing Information in a Variable

If the variable is declared outside a procedure it can be changed by any script in the ASP file. If the variable is declared inside a procedure, it is created and destroyed every time the procedure is executed

Different variables in ASP are mentioned below:

Dim 'em first

To "Dim" it means to Dimension it. That's VB lingo. A variable is declared in _VBScript using the Dim keyword.

```
<%
Dim myVar
%>
```

VB programmers will notice here, that we have not included any indication of the type of the said variable. E.g. Dim myString as String, or Dim myString$.

In _VBScript, all variables are variants. Their type is determined automatically by the runtime interpreter, and the programmer need not (and should not) bother with them.

By default, _VBScript does not force requiring variable declaration. That is, it is allowed to use a variable directly without declaring it first. However, experienced programmers know the importance of making it compulsory to declare all your variables first – without that, the bugs that may result are very difficult to detect. Considering this, we have here a simple directive to make variable declaration compulsory.

```
<%
Option Explicit
Dim myVar
%>
```

Remember that Option Explicit must necessarily be the first statement of your ASP page, otherwise a server error is generated.

To illustrate what I mean, if you had a page that read:

```
<%
Pi = 3.141592654
Response.Write Pi
%>
```

This is a perfectly valid page. You will get the value of Pi written back to the page, as you really expected.

Now, using the Option Explicit directive as above, let's rewrite the same page as follows:

```
<%
Option Explicit
```

```
Dim Pi
Pi = 3.141592654
%>
```

The Big If

The simplest of constructs, found in every language, is the If-Then-Else statement. I think it's familiar to everyone, so let's start with an example.

```
<%
If OK = True Then
  Response.Write "OK"
Else
  Response.Write "Error"
End If
%>
```

Some important points to note:

- The condition after the If must be followed by the Then keyword. This is unlike C, C++ or Java which do not require the Then keyword to follow.
- If only a single statement is to be executed in the Then block, it may directly follow the Then on the same line. If there are multiple statements to execute in the Then-block, the first statement should begin on the line after Then.
- The Else-block, as in most languages, is optional.
- The complete set of statements in the Then-block as well as the Else-block need to be "closed" by the End If keyword. This is very important, and the source of many hard-to-locate errors! Take care not to forget it!

The following are further examples of valid as well as invalid If-constructs:

```
<%
If OK = True Then Response.Write "OK" Else Response.Write "Error"
%>
```

This is valid. Since only one statement is to be executed in each of the Then- & Else-blocks, we do not require an End If. In addition, the Else statement must be on the same line.

```
<%
If OK = True Then Response.Write "OK"
Else Response.Write "Error"
%>
```

This is invalid. Since only one statement belongs to the Then-block, the first part of the construct above is fine. However, the Else cannot continue on the next line in such a case.

A simple rule of thumb to follow, is to use only one form of the If-statement for all your needs:

```
<%
If OK = True
```

```
TheResponse.Write "OK"
 ' ... Any more statements
Else
  Response.Write "Error"
 ' ... Any more statements
End If
%>
```

Incidentally, any line that begins with an apostrophe, ', is a comment; it is ignored by the interpreter. The following line is partly executable and partly a comment.

```
<%
OK = True 'Sets OK to True
%>
```

The comment begins with the apostrophe and continues to the end of the line.

For-Next Loops

The syntax is as follows:

```
<%
For I = 1 to 10
Response.Write "Number = " & I & vbCrLf
Next
%>
```

And the output ...

```
Number          =          1
Number          =          2
Number          =          3
Number          =          4
Number          =          5
Number          =          6
Number          =          7
Number          =          8
Number          =          9
Number          =         10
```

The vbCrLf used in the statement above is a predefined constant that equals the combination of the Carriage-Return character (CR for short), and the Line Feed character (LF for short.) Using it causes the output to continue on the next line.

Without the vbCrLf, our output would have appeared on one long line:

```
Number = 1Number = 2Number = 3Number = 4Number = 5Number = 6Number = 7Number =
8Number = 9Number = 10
```

Let us take a case of nested loops to clarify things:

```
<%
For I = 1 to 8
For j =1 to 8
Response.Write "X"
Next
Response.Write vbCrLf
Next
%>
```

This will draw a nice chessboard pattern on the screen. (You will need to view the source of the page in your browser however. If you look at the page in the browser itself, you will not see the true result.)

A very important point to note is that the Next statement that completes the For does not take an argument. You cannot say:

```
Next I
```

Or

```
Next J
```

This is invalid. Each Next statement encountered is automatically assumed to complete the immediately preceding for statement.

Finally, _VBScript also allows the Step keyword to modify the interval or step-size of the For-loop variable.

```
<%
For I = 1 to 10 Step
Response.Write "Number = " & I & vbCrLf
Next
%>
gives you:
Number = 1
Number = 3
Number = 5
Number = 7
Number = 9
```

The loop counted I in steps of 2, thus taking on only odd values from the entire set of 10.

For Each Object in Collection.

The for-Each construct is unique to _VBScript (and its parent, Visual Basic, of course!) It allows you to iterate through the items in a collection one by one.

```
<%
For Each Member in Team
```

```
Response.Write Member
Next
%>
```

Here, Team is assumed to be a collection of items. This statement is very useful in scenarios, where the size of the collection is not known in advance. Using the For Next statement assures that all items in that collection will be processed, and no "Array Index Out Of Bounds" errors will be generated.

While ... Wend

Again, here is one of the popular looping constructs of all time.

```
<%
While Not RS.EOF
   Response.Write RS.Fields ("Name")
   RS.MoveNext
Wend
%>
```

or,

```
<%
Do While Not RS.EOF
   Response.Write RS.Fields ("Name")
   RS.MoveNext
Loop
%>
```

The While statement executes the statements in its loop until the given condition remains true. The moment it becomes false, the loop halts.

Remember to end the While Statement with the Wend Keyword.

A variation of the While loop that tests the condition after the loop is the Do-loop.

```
<%
Do
   TimePass()
Until ExamDate - Now = 30
%>
```

I hope the meaning is amply clear from the example above.

1.7.2 Select Case

To make a choice between a set of items that can be assigned to a variable, use the Select Case statement.

```
<%
Select Case Choice
Case "1":
```