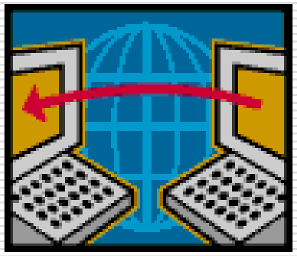
ASP (Active Server Pages)



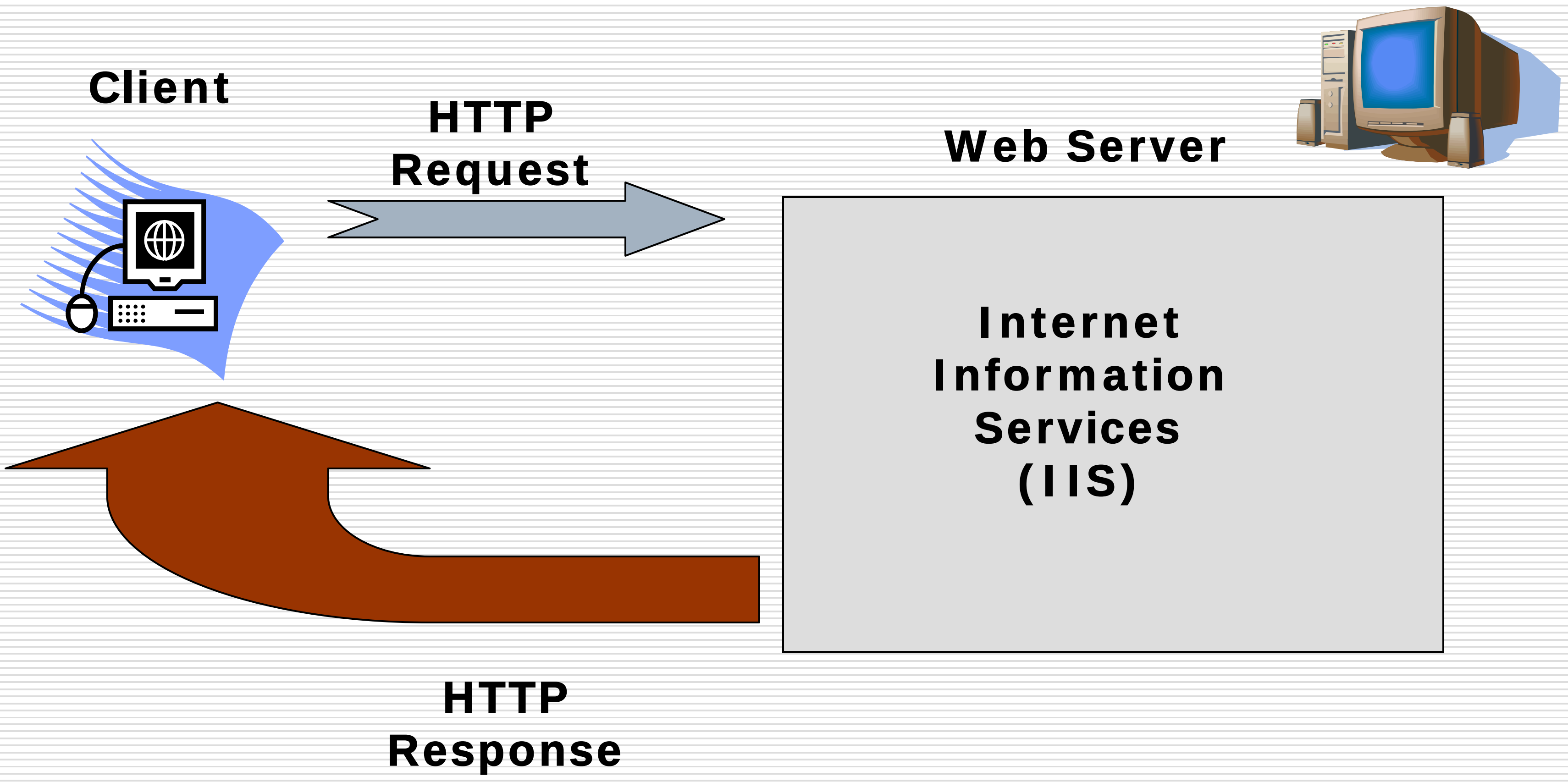
Prerequisites

- Knowledge of Hyper Text Markup Language (HTML).
- Knowledge of life cycle of web page from request to response.
- Knowledge of Scripting language like vbscript, javascript, jscript

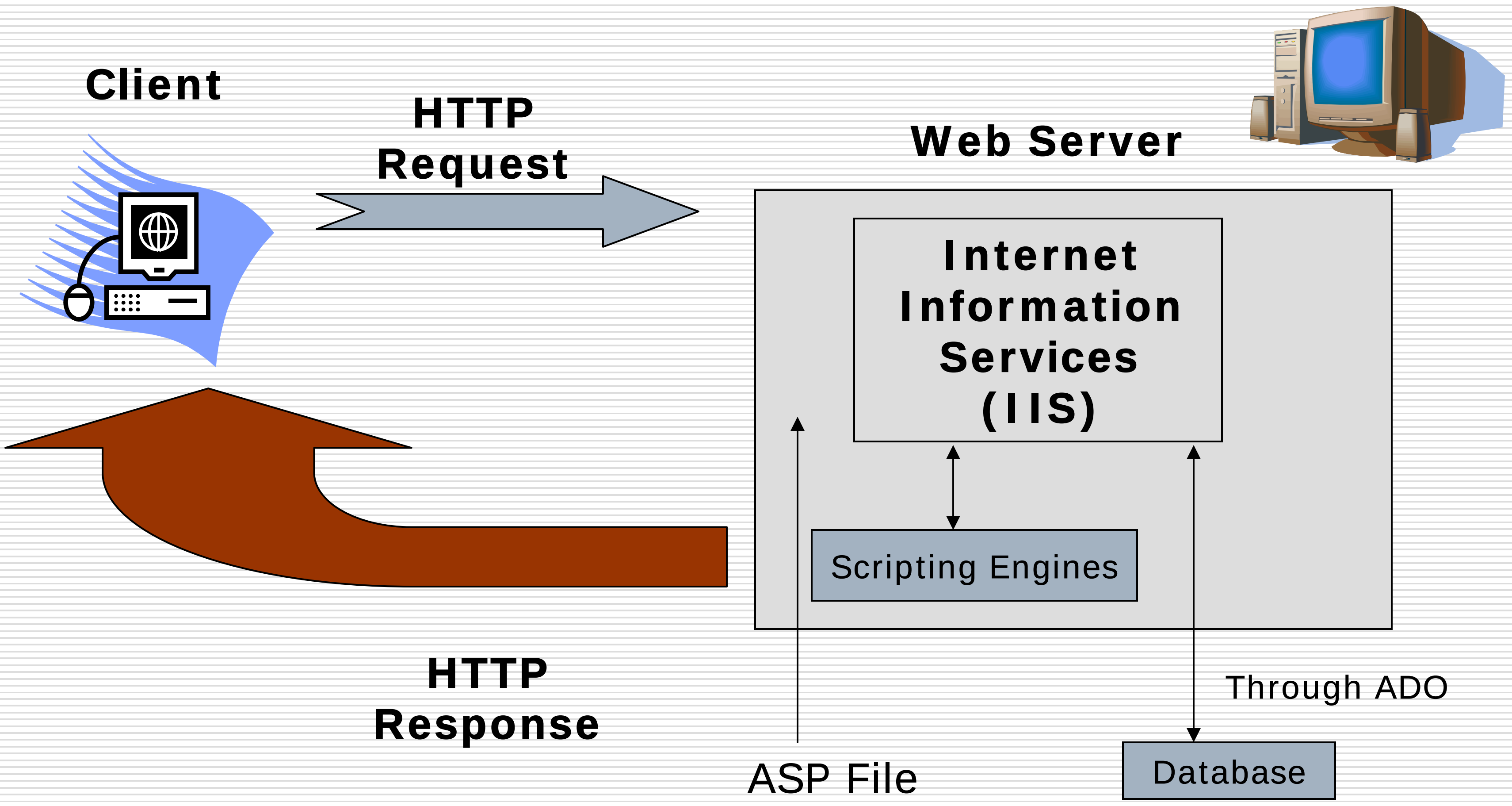
How Does ASP Differ from HTML?

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

Life Cycle of a HTML Page



Life Cycle of ASP Page



What is **A**ctive **S**erver **P**ages **(ASP)**?

- As the name suggests, ASP represents pages that are executed on server side.
- ASP stands for Active Server Pages
- ASP is a program that runs inside IIS
- IIS stands for Internet Information Services
- IIS comes as a free component with Windows 2000 ,Windows XP and Windows 2000/2003 server.
- PWS is a smaller - but fully functional - version of IIS
- PWS can be found on your Windows 95/98 CD
- When a client machine requests an ASP page, request is being sent to server. Server processes the request with the help of C:\WINDOWS\system32\inetsrv\asp.dll file and sends back the response to the client machine.

ASP Compatibility

- ASP is a Microsoft Technology
- To run IIS you must have Windows NT 4.0 or later
- To run PWS you must have Windows 95 or later
- Chili ASP is a technology that runs ASP without Windows OS
- Instant ASP is another technology that runs ASP without Windows

What is an ASP File?

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

What you can do with ASP?

- Dynamically edit, change or add any content of a Web page
- Respond to user queries or data submitted from HTML forms
- Access any data or databases and return the results to a browser
- Customize a Web page to make it more useful for individual users
- The advantages of using ASP instead of CGI and Perl, are those of simplicity and speed
- Provide security since your ASP code can not be viewed from the browser
- Clever ASP programming can minimize the network traffic

Important: Because the scripts are executed on the server, the browser that displays the ASP file does not need to support scripting at all!

How to Run ASP on your own PC

- You can run ASP on your own PC without an external server. To do that, you must install Microsoft's Personal Web Server (PWS) or Internet Information Services (IIS) on your PC.
- **If you are serious about using ASP, you should have at least Windows 98, Second Edition.**
- **If you are really serious about using ASP, you should go for Windows 2000.**

How to install PWS and run ASP on Windows 95

- Personal Web Server (PWS) is not shipped with Windows 95 !!
- To run ASP on Windows 95, you will have to download "Option Pack" from Microsoft.

How to install PWS and run ASP on Windows NT

- Personal Web Server (PWS) is not shipped with Windows NT !!
- To run ASP on Windows NT, you will have to download "Windows NT 4.0 Option Pack" from Microsoft.

How to install PWS and run ASP on Windows 98

- Open the Add-ons folder on your Windows98 CD, find the PWS folder and run the setup.exe file.
- An Inetpub folder will be created on your harddrive. Open it and find the wwwroot folder.
- Create a new folder, like "MyWeb", under wwwroot.
- Use a text editor to write some ASP code, save the file as "test1.asp" in the "MyWeb" folder.
- Make sure your Web server is running - The installation program has added a new icon on your task bar (this is the PWS symbol). Click on the icon and press the Start button in the window that appears.
- Open your browser and type in "http://localhost/MyWeb/test1.asp", to view your first ASP page.

How to install IIS and run ASP on Windows 2000

- From your **Start Button**, go to **Settings**, and **Control Panel**
- In the Control Panel window select **Add/Remove Programs**
- In the Add/Remove window select **Add/Remove Windows Components**
- In the Wizard window check **Internet Information Services**, **click OK**
- An **Inetpub folder** will be created on your harddrive
- Open the Inetpub folder, and find a folder named **wwwroot**
- **Create a new folder**, like "MyWeb", under wwwroot.
- **Use a text editor** to write some ASP code, save the file as "test1.asp" in the "MyWeb" folder
- Make sure your Web server is running - The installation program has added a new icon on your task bar (this is the IIS symbol). Click on the icon and press the Start button in the window that appears.
- **Open your browser** and type in "http://localhost/MyWeb/test1.asp", to view your first ASP page

How to install IIS and run ASP on Windows XP Professional

- Insert the Windows XP Professional CD-Rom into your CD-Rom Drive
- From your **Start Button**, go to **Settings**, and **Control Panel**
- In the Control Panel window select **Add/Remove Programs**
- In the Add/Remove window select **Add/Remove Windows Components**
- In the Wizard window check **Internet Information Services**, **click OK**
- An **Inetpub folder** will be created on your harddrive
- Open the Inetpub folder, and find a folder named **wwwroot**
- **Create a new folder**, like "MyWeb", under wwwroot.
- **Use a text editor** to write some ASP code, save the file as "test1.asp" in the "MyWeb" folder
- Make sure your Web server is running - its status can be checked by going into the **Control Panel**, then **Administrative Tools**, and double-click the "**IIS Manager**" icon
- **Open your browser** and type in "http://localhost/MyWeb/test1.asp", to view your first ASP page

Note: You cannot run ASP on Windows XP Home Edition.

How to install IIS and run ASP on Windows Server 2003 (Windows .NET Server)

- When you start the Windows Server 2003, you should see the **Manage Your Server wizard**
- If the wizard is not displayed, go to **Administrative Tools,** and select **Manage Your Server**
- In the wizard, click **Add or Remove a Role**, click Next
- Select **Custom Configuration**, click Next
- Select **Application Server role**, click Next
- Select **Enable ASP.NET**, click Next
- Now, the wizard may ask for the **Server 2003 CD**. Insert the CD and let it run until it is finished, then click the Finish button
- The wizard should now show the Application Server role installed
- Click on **Manage This Application Server** to bring up the **Application Server Management Console (MMC)**
- Expand the **Internet Information Services (IIS) Manager**, then expand your server, and then the Web Sites folder
- You should see the Default Web Site, and it should not say (Stopped)
- IIS is running!
- In the **Internet Information Services (IIS) Manager** click on the **Web Service Extensions** folder
- Here you will see that **Active Server Pages are Prohibited** (this is the default configuration of IIS 6)
- Highlight **Active Server Pages** and click the **Allow** button
- ASP is now active!

Important

- You cannot view the ASP source code by selecting "View source" in a browser, you will only see the output from the ASP file, which is plain HTML. This is because the scripts are executed on the server before the result is sent back to the browser.

The Basic Syntax Rule

An ASP file normally contains HTML tags, just like an HTML file. However, an ASP file can also contain **server scripts**, surrounded by the delimiters **<%** and **%>**. Server scripts are **executed on the server,** and can contain any expressions, statements, procedures, or operators valid for the scripting language you prefer to use.

Write Output to a Browser

The response.write command is used to write output to a browser. The following example sends the text "Hello World" to the browser:

```
<html>
<body>
<%response.write("Hello World!")%>
</body>
</html>
```

Contd..

There is also a shorthand method for the response.write command. The following example also sends the text "Hello World" to the browser:

```
<html>
<body>
<%="Hello World!"%>
</body>
</html>
```

Variables

- A variable is used to store information.
- 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.

ASP Variables

- ASP variables are declared using VBScript declaration type.
- Assuming VbScript is used, this is how you declare variable:
  - DIM varaibleName or Const variableName.
- If you want reject undeclared variables, use <% Option Explicit %> at the beginning of your page. You would often see these two lines:
  <%@ Language="Vbscript" %>
  <% Option Explicit %>
  at the beginning of ASP pages.
- First line sets the language and the second line watches undeclared variables. VBScript and JavaScript variables are variant type variable, which means they can take any type of values. Any variable name must start with letter or underscore.

Program With Variables

```
<html>
    <body>
    <%
      dim h
      h="Hello World"
      response.write("Say: " & h)
    %>
    </body>
    </html>
```

Variable Example

```
<%
        Dim name, email, age
        name=”John M”
         email=”you@you.com”
        age=35
         response.write(“Your Name: “ & name & "<br>")
         response.write(“Your Email: “ & email & "<br">)
         response.Write(“Your age: “ & age)
%>
```

Program to Display Current Time

```
<html>
  <body>
  It's now <%=Time()%>
  </body>
  </html>
```

ASP Arrays

- An array is an indexed list of things called elements or group of related variables. For example, say that we want declare variables for list of cars like this; Dim car1, car2, car2, car3,....
- Here is how you would declare list of cars using array: Dim cars(3); We simply declared array that takes 4 items
- To assign values to this array, do these:
  cars(0)="Jeep Grand Cherokee"
  cars(1)="Jeep Wrangler"
  cars(2)="Jeep Liberty"
  cars(3)="Jeep Cherokee Briarwood" Use this statement to write an item in the array: response.write(cars(3)) This will write the 4th car on the list.

Example

```
<%

Dim Cars(3)

cars(0)="Jeep Grand Cherokee"
cars(1)="Jeep Wrangler"
cars(2)="Jeep Liberty"
cars(3)="Jeep Cherokee Briarwood"

response.write(cars(0) & "<br>")
response.write(cars(1) & "<br>")
response.write(cars(2) & "<br>")
response.write(cars(3) & "<br>")
%>
```

VBScript

You can use several scripting languages in ASP. However, the default scripting language is VBScript:

```
<html>
<body>
<%response.write("Hello World!")%>
</body>
</html>
```

The example above writes "Hello World!" into the body of the document.

JavaScript

To set JavaScript as the default scripting language for a particular page you must insert a language specification at the top of the page:

```
<%@ language="javascript"%>
<html>
<body>
<%Response.Write("Hello World!")%>
</body>
</html>
```

Note: Unlike VBScript - JavaScript is case sensitive. You will have to write your ASP code with uppercase letters and lowercase letters when the language requires it.

Other Scripting Languages

- ASP is shipped with VBScript and JScript (Microsoft's implementation of JavaScript). If you want to script in another language, like PERL, REXX, or Python, you will have to install script engines for them.

  Important: Because the scripts are executed on the server, the browser that displays the ASP file does not need to support scripting at all!

ASP Sub Procedures

- ASP Sub Procedures are a collection of ASP statements that perform a task, and are executed by an *event procedure*. Event Procedures are any clickable objects, or onload event. Sub procedures do not return a value, but executes it's content on "call".

Procedures Examples

```
<%

    Sub GetInfo()
        dim name,telephone,fee
        name="Mr. John Doe"
        telephone="555-5555"
        fee=20
        Response.write("Name: "& name &"<br>")
        Response.write("Telephone: "& telephone &"<br>")
        Response.write("Fee: "& fee &"<br>")
    End Sub

    GetInfo()
%>
```

This example simply declares, populates, & writes three variables in the asp sub procedure, 'GetInfo'. This sub is executed right after the end sub.

Here is the execution result:

```
Name: Mr. John Doe
Telephone: 555-5555
Fee: 20
```

Procedures Examples

You can pass an argument to the asp sub procedure, and provide the value(s) when calling it.

This is an asp sub procedure that accepts an argument:

```
<html>
<head>
<%sub vbproc(num1,num2)
response.write(num1*num2)
end sub%>
</head>
<body>
<p>Result: <%call vbproc(3,4)%>
</p>
</body>
</html>
```

ASP Function Procedures

- ASP Function Procedures are a series of *VBscript* statements enclosed by the 'Function', and 'End Function' statements. Function procedures are similar to a 'Sub procedure', but can *also* return a value. An asp function procedure can accept arguments (constants, variables, or expressions) that are passed to it by calling a procedure).

Non-parameterized asp function procedure

```
<%
    function userName()
    userName="MaryLou"
    end function
%>
```

This function procedure remembers a username. You can use this function any number of times in your program, and you can change it once to maintain it- use 'userName()' to call this function.

For example: response.write("The username is: "&userName())

Parameterized asp function procedure

```
<%
    Function total(price)
            dim tax
            tax=price*.09
            total=price+tax
            end Function
%>
```

The price value is provided when calling the function-

Example: response.write("The total price is: " & total(50))

Example

```
<%
   Function profit(sellPrice, cost)
      dim pr
      profit=sellPrice-cost
   End Function
   dim currProfit
   currProfit=profit(1280,890)
   Response.write("Profit: $"&currProfit)
   %>
```

Calling from VBScript and JavaScript

- When calling a VBScript or a JavaScript procedure from an ASP file written in VBScript, you can use the "call" keyword followed by the procedure name. If a procedure requires parameters, the parameter list must be enclosed in parentheses when using the "call" keyword. If you omit the "call" keyword, the parameter list must not be enclosed in parentheses. If the procedure has no parameters, the parentheses are optional.
- When calling a JavaScript or a VBScript procedure from an ASP file written in JavaScript, always use parentheses after the procedure name.

If—Else--End if

Using if statement, ASP has the ability to make distinctions between different possibilities. For example, you might have a script that checks if a variable consists certain type of values.

```
<%
    dim n
    n =1
    if n= 1 then
        response.write("N has the value equal to 1")
    else
        response.write("N is not equal to one")
    end if
    %>
```

Nested If Statement

- You would be able to check variety of conditions based on deversified values.
- The following example is nested if statement to check two conditions:

```
<%
    dim fruit1, fruit2, in_store
    fruit1="Apple"
    fruit2="Orange"
    in_store=1
    if in_store=1 then
    %>
      <%=fruit1%> is in store
    <%
    elseif in_store=2 then
    %>
      <%=fruit2%> is in store
    <%
    else
      response.write("No value specified")
    end if
    %>
```

Case Statement

Case statement can be used instead of if statement suitably when one condition could have multiple possibilities. Following example illustrates the use of case statement

```
<% Dim dat
    dat=WeekDay(Date)
    %>
    <%
    Select Case dat
      case 1
        response.write("Today is Sunday")
      case 2
        response.write("Today is Monday")
      case 3
        response.write("Today is Tuesday")
      case 4
        response.write("Today is Wednesday")
      case 5
        response.write("Today is Thursday")
      case 6
        response.write("Today is Friday")
        case 7
        response.write("Today is Saturday")
      end select
    %>
```

ASP Loop Statements

- Loops are set of instructions that repeat elements in specific number of times. Counter variable is used to increment or decrement with each repetition of the loop. The two major groups of loops are, For..Next and Do..Loop. While..Wend is another type of Do..Loop. The For statements are best used when you want to perform a loop in specific number of times. The Do and While statements are best used to perform a loop an undetermined number of times.

Looping Structures - For Loop

```
<%
    Dim counter
    counter=0
    for counter = 0 to 5
      response.write("The counter is: "&counter&"<br>")
    next
    %>
```

The above example increments a variable counter from 0 to 5 The values of counter are incremented by 1 on each run before 6

We can modified how the values are incremented by adding step # to the for counter statement.

Looping Structures - For Loop

```
<%
Dim counter
counter=0
for counter = 0 to 5 step 2
   response.write("The counter is: "&counter&"<br>")
next
%>
```

Looping Structures - For Loop

```
<%
   Dim counter
   counter=0
   for counter = 5 to 0 step -1
     response.write("The counter is: "&counter&"<br>")
   next
   %>
```

Looping Structures - Do Loop

- The Do..Loop structure repeats a block of statements until a specified condition is met. There are three types of Do..Loops.
  - Do..Until
  - Do..While
  - While..Wend.
- Do..While and While..Wend performs a loop statement as long as the condition being tested is true while Do..Until performs a loop statement as long as the condition tested is false. In both cases, you have a choice to perform the test at start of the loop or at the end of the loop

Looping Structures - Do Until Loop

```
<%
   Dim counter
   counter=5
   Do Until counter=0
     response.write("The counter is: "&counter&"<br>")
   counter=counter-1
   loop
   %>
```

Looping Structures
- Do Until Loop

- You can also accomplish the loop this way:

```
<%
    Dim counter
    counter=5
    Do Until counter=0
      response.write("The counter is: "&counter&"<br>")
    counter=counter-1
    loop
    %>
```

Looping Structures - Do While Loop

```
<%
Dim my_num
my_num=1
Do While my_num <=10
    Response.Write(my_num & "<br>“)
    my_num = my_num +1
Loop
%>
```

Looping Structures - While Loop

```
<%
dim x
x=1
While x<10
    Response.write(x & "<br>")
    x=x+1
Wend
%>
```

Session Variables

- Session variables are used to store information about ONE single user, and are available to all pages in one application. Typically information stored in session variables are name, id, and preferences.

Application Variables

- Application variables are also available to all pages in one application. Application variables are used to store information about ALL users in a specific application.

User Input

- The Request object may be used to retrieve user information from forms.
- **Form example:**

```
<form method="get" action="simpleform.asp">
First Name: <input type="text" name="fname" /><br>
Last Name: <input type="text" name="lname" /><br><br>
<input type="submit" value="Submit" />

</form>
```

- User input can be retrieved in two ways: With **Request.QueryString** or **Request.Form**

Request.QueryString

- The Request.QueryString command is used to collect values in a form with method="get". Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send.
- If a user typed "Bill" and "Gates" in the form example above, the URL sent to the server would look like this:

http://www.du.ac.in/simpleform.asp?fname=Bill&lname=Gates

Contd..

- Assume that the ASP file "simpleform.asp" contains the following script:

```
<body>Welcome
<%response.write(request.querystring("fname"))
response.write(" " &
request.querystring("lname"))%>

</body>
```

The browser will display the following in the body of the document:

Welcome Bill Gates

Request.Form

- The Request.Form command is used to collect values in a form with method="post". Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.
- If a user typed "Bill" and "Gates" in the form example above, the URL sent to the server would look like this:

http://www.du.ac.in/simpleform.asp

Contd..

- Assume that the ASP file "simpleform.asp" contains the following script:

```
<body>
Welcome <%response.write(request.form("fname"))
response.write(" " & request.form("lname"))%>
</body>
```

The browser will display the following in the body of the document:

Welcome Bill Gates

Processing forms using ASP

- You can process HTML forms using these two powerful ASP objects, *Response* and *Request*. **Response** outputs the value to a page and **Request** retrieves values from an object. Take a look at the following example

Example 1-Userform.html

```
<form name=”userForm” method=”post”
    action=”userForm.asp”>
      Enter a user name: <input type=”text”
    name=”userName” size=”20”>
      Enter a password: <input type=”password”
    name=”password” size=”20”>
      <input type=”submit” name=”submit”
    value=”Send”>
    </form>
```

We just created HTML form and tell the browser to process the form using the file *"userForm.asp"*. The following is userForm.asp file that writes the values from the form.

Userform.asp

```
<html>
<head>
<title>Process form Info</title>
</head>
<body>
You have typed the user name
   <%=Request.Form("userName")%> and
   the password
   <%=Request.Form("password")%>.
</body>
</html>
```

Form Processing Example 2

```
<html>
<head>
<title>Form Example 2</title>
</head>
<body>
<p><b>This example process basic form elements</b>
<form method="POST" action="formProcess.asp">
<p>Your name: <input type="text" name="Name" size="20"><br>
Status: <input type="radio" value="Customer" name="status">Customer   <input
type="radio" name="status" value="Visitor">Visitor<br>
Do you own any of these trucks:<br>
<input type="checkbox" name="truck" value="Land Cruiser">Land Cruiser<br>
<input type="checkbox" name="truck" value="Sequoia">Sequoia<br>
<input TYPE="checkbox" name="truck" value="4Runner">4Runner<br>
<input TYPE="checkbox" name="truck" value="Highlander">Highlander<br>
<input TYPE="checkbox" name="truck" value="Tundra Access Cab">Tundra Access Cab<br>
Car of Choice:<select size="1" name="car">
<option value="MR2 Spyder">MR2 Spyder</option>
<option value="Celica">Celica</option>
<option value="Matrix">Matrix</option>
<option value="Avalon">Avalon</option>
<option value="Camry">Camry</option>
<option value="Corolla">Corolla</option>
<option value="Echo">Echo</option>
<option value="Prius">Prius</option>
<option value="RAV4 EV">RAV4 EV</option>
</select><br>
Enter some general comments about what you think about Toyota cars:<br>
<textarea rows="5" name="Comments" cols="50"></textarea><br>
<align="center"><input type="submit" value="Submit" name="submit"><br>
</form>
</body>
</html>
```

Code for formProcess.asp

```
<html>
<head>
<title>Result of your information</title>
</head>
<body>
<%
dim name, status, truck, car, comments
name=Request.Form("Name")
status=Request.Form("status")
car=Request.Form("car")
comments=Request.Form("comments")
truck=Request.Form("truck")
%>
Your name: <b><%=name%></b><br>
Status: <b><%=status%></b><br>
Your favourite car is: <b><%=car%></b><br>
You currently own these trucks:<b> <%=truck%></b><br>
Your comments about Toyota products:<b><%=comments%></b>
</body>
</html>
```

Writing to a Text File using ASP

- You can write to or read from a text file using ASP. The following is simple example that illustrates how to create text file and write some information to it.

```
<html>
<title>Create txt file </title>
<body>
<%
Set fileObj=Server.CreateObject("Scripting.FileSystemObject")
set file1=fileObj.CreateTextFile("c:\inetpub\wwwroot\asp\TextFile.txt")
file1.WriteLine("This is what goes to the text file that would be
     created")
file1.WriteLine("This is the second line of the text file")
file1.Close
set file1=nothing
set fileObj=nothing
%>
</body>
</html>
```

Reading from a Text File

- Reading from a text file is also very easy and similar to writing to it. The following example illustrates how to read from a text file.

```
<html>
<body>
<%
Set fileObj=Server.CreateObject("Scripting.FileSystemObject")
Set listFile=fileObj.OpenTextFile(Server.MapPath("\asp\textfile.txt"), 1)
do while listFile.AtEndOfStream = false
Response.Write(listFile.ReadLine)
Response.Write("<br>")
loop
listFile.Close
Set listFile=Nothing
Set fileObj=Nothing
%>
</body>
</html>
```

- To display all the lines at once without line breaks, simply replace the lines starting with do while and ending with loop to Response.Write(listFile.ReadAll).
- To display the first line of the text file use, Response.Write(listFile.ReadLine).
- To skip line of a text file use, listFile.SkipLine
- To skip part of line of a text use,listFile.Skip(2). This will skip 2 characters.

Adding Data to Access Database

- To add data to a database table, you need an existing database plus the table to add the data to. Let us assume that you have Access Database file name *FeedBack.mdb* in the same folder as this file with the following table:

tblFeeds

FieldName	**DataType**	**FieldSize**
user_id	Autonumber	8
Name	Text	45
Comments	Text	200

Form Validation

- User input should be validated on the browser whenever possible (by client scripts). Browser validation is faster and you reduce the server load.
- You should consider using server validation if the user input will be inserted into a database. A good way to validate a form on the server is to post the form to itself, instead of jumping to a different page. The user will then get the error messages on the same page as the form. This makes it easier to discover the error.

```
<html>
<head>
<title> Adding to database example </title>
<script type="text/javascript">
<!--
function validate()
{
  if(document.form.name.value=="")
  {
   alert("Name is missing");
   return false;
  }
  if(document.form.comments.value.length<8)
  {
   alert("Not enough comments entered");
   return false;
  }
  else
  {
   return true;
  }
}
//-->
</script>
</head>
<body>
<form name="form" method="post" action="save.asp">
Name: <input type="text" name="name" maxlength="45"> <br>
Comments: <textarea cols="20" rows="8" name="comments"
maxlength="200"> </textarea><br>
<input type="submit" name="Save" value="Submit" onClick="return validate();">
</form>
</body>
</html>
```

Save.asp

```
<%
Dim Conn
Dim Rs
Dim sql
'Create an ADO connection and recordset object
Set Conn = Server.CreateObject("ADODB.Connection")
Set Rs = Server.CreateObject("ADODB.Recordset")
'Set an active connection and select fields from the database
Conn.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("FeedBack.mdb")
sql= "SELECT name, comments FROM tblFeeds;"

Rs.CursorType = 2
Rs.LockType = 3

Rs.Open sql, Conn    'Open the recordset with sql query

Rs.AddNew 'Prepare the database to add a new record and add
Rs.Fields("name") = Request.Form("name")
Rs.Fields("comments") = Request.Form("comments")

Rs.Update   'Save the update
Rs.Close
Set Rs = Nothing
Set Conn = Nothing
response.redirect("view.asp")
%>
```

View.asp

```
<%
Dim Conn
Dim Rs
Dim sql
Set Conn = Server.CreateObject("ADODB.Connection")
Set Rs = Server.CreateObject("ADODB.Recordset")
Conn.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
       Server.MapPath("FeedBack.mdb")
sql= "SELECT name, comments FROM tblFeeds;"
Rs.Open sql, Conn
Do While not Rs.EOF
  Response.Write
       ("========================================"&"<b
       r>")
  Response.Write ("Name: " & "<font color='red'>" & Rs("name") & "</font>")
  Response.Write ("<br>")
  Response.Write ("Comment: " & "<font color='red'>" & Rs("comments") &
       "</font>")
  Response.Write ("<br>")
  Rs.MoveNext
Loop
Rs.Close
Set Rs = Nothing
Set Conn = Nothing
%>
```

Update a record

- There are more than one way to do things. For this example, we are going to list items from the database so that you can select a record using radio button.

```
<html>
<body >
Select name to update.
<%
Dim Conn, Rs, sql
Set Conn = Server.CreateObject("ADODB.Connection")
Set Rs = Server.CreateObject("ADODB.Recordset")
Conn.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("FeedBack.mdb")
sql= "SELECT * FROM tblFeeds;"
Rs.Open sql, Conn
Response.Write "<FORM name=‘Update’ method='post' action='toUpdateT.asp'>"
Response.Write "<table border=1 cellspacing=0>"
Response.Write "<tr>"&"<td colspan='3' align='center'>"&"Select a comment to update and
click select"&"</td>"&"</tr>"
Response.Write "<tr>"&"<th align='center' colspan='2'>"&"Name"&"</th>"&"<th
align='center'>"&"Comment"&"</th>"&"</tr>"
if NOT Rs.EOF then
Do While not Rs.EOF
    Response.Write ("<tr>")
    Response.Write ("<td>"&"<input type='radio' name='ID'
value="&Rs("user_id")&">"&"</td>")
    Response.Write ("<td>"&Rs("name")&"</td>")
    Response.Write ("<td>"&Rs("comments")&"</td>")
    Response.Write ("</tr>")
    Rs.MoveNext
Loop
else
   Response.Write("No records found")
end if
Response.Write("<tr>"&"<td colspan='3' align='center'>"&"<input type ='submit'
name='submit' value='Select' >"&"</td>"&"</tr>")
Response.Write "</table>"
Rs.Close
Set Rs = Nothing
Set Conn = Nothing
%>
</form>
</body>
</html>
```

Toupdatet.asp

```
<html>
<body>
<form name="updated" action="updateComment.asp" method="post">
<%
Dim ID, name, comments
ID= Request.Form("ID")
Session(“id”)=ID
name = Request.Form("name")
comments=Request.Form("comments")
if ID="" then
   Response.Write "You did not select a name to update!"
Else
Dim Conn, Rs, sql
Set Conn = Server.CreateObject("ADODB.Connection")
Set Rs = Server.CreateObject("ADODB.Recordset")
Conn.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" & Server.MapPath("FeedBack.mdb")
sql= "Select * FROM tblFeeds WHERE user_id="&ID
Rs.Open sql, Conn
if NOT Rs.EOF then
%>
<table border=1 cellspacing=0>
<tr><td colspan="2" align="center">Update and save</td></tr>
<tr>
   <td>Name: </td>
   <td><input type="text" name="name" size="30" maxlength="45" value="<%=Rs("name")%>"></td>
</tr><tr>
   <td>Comment: </td>
   <td><input type="text" name="comments" size="30" maxlength="250" value="<%=Rs("comments")%>"></td>
</tr><tr>
   <td colspan="2" align="center"><input type="submit" name="submit" value="Save"></td>
</tr>
</table>
</form>
<%
else
   Response.Write("Record does not exist")
end if
   Conn.Close
   Set Conn = Nothing
End If
%>
</body>
</html>
```

Updatecomment.asp

```
<html>
<body>
<%
Dim name,comments, user_id
ID = session(“id")
name = Request.Form("name")
comments=Request.Form("comments")
if name="" OR comments="" then
Response.Write "A field was left empty, please try again!"
Else
Dim Conn ,Rs, sql
Set Conn = Server.CreateObject("ADODB.Connection")
Set Rs = Server.CreateObject("ADODB.Recordset")
Conn.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("FeedBack.mdb")
sql= "Update tblFeeds Set name='"& name & "', comments='" & comments &"' WHERE
user_id=" & ID
Rs.Open sql, Conn
Conn.Close
Set Rs=Nothing
Set Conn = Nothing
Response.Write "Successfully Updated"
End If
%>
</body>
</html>
```

Delete a record

- We are going to use two files in order to delete a record. First file *(toDelete.asp)* is to view all the records and the second file *(deleteComment.asp)* is to delete selected record.

Todelete.asp

Select name to delete.

```
<%
Dim Conn, Rs, sql
Set Conn = Server.CreateObject("ADODB.Connection")
Set Rs = Server.CreateObject("ADODB.Recordset")
Conn.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("FeedBack.mdb")
sql= "SELECT * FROM tblFeeds;"
Rs.Open sql, Conn
Response.Write "<FORM name='Delete' method='post' action='DeleteComment.asp'>"
Response.Write "<table border=1 cellspacing=0>"
Response.Write "<tr>"&"<td colspan='3' align='center'>"&"Select a comment to delete
and click delete"&"</td>"&"</tr>"
Response.Write "<tr>"&"<th align='center' colspan='2'>"&"Name"&"</th>"&"<th
align='center'>"&"Comment"&"</th>"&"</tr>"
Do While not Rs.EOF
Response.Write ("<tr>")
Response.Write ("<td>"&"<input type='radio' name='ID'
value="&Rs("user_id")&">"&"</td>")
Response.Write ("<td>"&Rs("name")&"</td>")
Response.Write ("<td>"&Rs("comments")&"</td>")
Response.Write ("</tr>")
Rs.MoveNext
Loop
Response.Write("<tr>"&"<td colspan='3' align='center'>"&"<input type ='submit'
name='submit' value='Delete' onClick='return validate();'>"&"</td>"&"</tr>")
Response.Write "</table>"
Response.Write "</form>"
Rs.Close
Set Rs = Nothing
Set Conn = Nothing
%>
```

Deletecomment.asp

```
<%
Dim ID
ID = Request.Form("ID")
if ID="" then
  Response.Write "You did not select a name to delete!"
Else
Dim Conn
Dim Rs
Dim sql
Set Conn = Server.CreateObject("ADODB.Connection")
Set Rs = Server.CreateObject("ADODB.Recordset")
Conn.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
     Server.MapPath("FeedBack.mdb")
sql= "Delete FROM tblFeeds WHERE user_ID=" & ID


Rs.Open sql, Conn
Conn.Close
Set Conn = Nothing
Response.Write "Successfully Deleted"
End If
%>
```

Virtual Includes

- The virtual include or include file is a command that instructs the browser to display a text file (.txt, .html, .htm, .shtml, .asp, etc). Virtual is used if the impended file is located in different directory.
  For example, if you want display standard or same information more than one page. You can create seperate file for this information and include whatever other file you want the content to be displayed. It makes easier to maintain the information by changing one file instead of going through all the files.

- Here is how you include file virtualy
  `<!--#include virtual="/files/file.asp" -->`
- or you can include it this way if the file is located in the current directory
  `<!--#include file="file.asp" -->`

What is a Cookie?

- A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With ASP, you can both create and retrieve cookie values.

- Each time the same computer access the page, the cookie will also be retrieved if the expiry date has future value.

How to Create a Cookie?

- The "Response.Cookies" command is used to create cookies.

  Note: The Response.Cookies command must appear BEFORE the <html> tag.

  In the example below, we will create a cookie named "firstname" and assign the value "Alex" to it:

  ```
  <%Response.Cookies("firstname")="Alex"%>
  ```

- It is also possible to assign properties to a cookie, like setting a date when the cookie should expire:

```
<%Response.Cookies("firstname")="Alex"
Response.Cookies("firstname").Expires=#May 10,2002#%>
```

How to Retrieve a Cookie Value?

- The "Request.Cookies" command is used to retrieve a cookie value.
- In the example below, we retrieve the value of the cookie named "firstname" and display it on a page:

```
<%fname=Request.Cookies("firstname")
response.write("Firstname=" & fname)%>
```

Output: Firstname=Alex

A Cookie with Keys

- If a cookie contains a collection of multiple values, we say that the cookie has Keys.
- In the example below, we will create a cookie collection named "user". The "user" cookie has Keys that contains information about a user:

```
<%Response.Cookies("firstname")="Alex"
Response.Cookies("user")("firstname")="John"
Response.Cookies("user")("lastname")="Smith"
Response.Cookies("user")("country")="Norway"
Response.Cookies("user")("age")="25" %>
```

Read all Cookies

Now we want to read all the cookies sent to a user. The example below shows how to do it (note that the code below checks if a cookie has Keys with the HasKeys property):

```
<html>
<body>

<% dim x,y
for each x in Request.Cookies
      response.write("<p>")
      if Request.Cookies(x).HasKeys then
            for each y in Request.Cookies(x)
                        response.write(x & ":" & y & "=" & Request.Cookies(x)(y))
                        response.write("<br/>")
            next
      else
            Response.Write(x & "=" & Request.Cookies(x) & "<br/>")
      end if
      response.write "</p>"
next %>
</body>
</html>
```

Output

```
firstname=Alex
user:firstname=John
user:lastname=Smith
user:country=Norway
user:age=25
```

The following example domonstrates how to retrieve some of the useful server variables

```
<%
agent = Request.ServerVariables("http_user_agent") 'Gets the browser type
IP = Request.ServerVariables ("REMOTE_ADDR") 'Retrieves the user IP Address
dnsIP = Request.ServerVariables("remote_host") 'Retrieves the remote host IP
Address
serverName = Request.ServerVariables("server_name") 'Retrieves the page domain
name
referer = request.servervariables("http_referer") 'Retrieves the referer url
scriptName=request.servervariables("script_name") 'Retrieves current page
serverPort=request.servervariables("server_port") 'Retrieves server port
serverSoftware=request.servervariables("server_software") 'Retrieves server software
Url=request.servervariables("URL") 'Retrieves page url
method=request.servervariables("Request_Method") 'Retrieves request mehtod .. get
or post
%>
<%
Response.Write("<b>User Agent: </b>"&agent &"<br>")
Response.Write("<b>IP Address:</b> "&IP &"<br>")
Response.Write("<b>Remote host IP:</b> "&dnsIP &"<br>")
Response.Write("<b>Server Domain name: </b>"&serverName &"<br>")
Response.Write("<b>Referer page:</b> "&referer &"<br>")
Response.Write("<b>Script Name: </b>"&scriptName &"<br>")
Response.Write("<b>Server Port: </b>"&serverPort &"<br>")
Response.Write("<b>Server Sortware:</b> "&serverSoftware &"<br>")
Response.Write("<b>Page url: </b>"&Url &"<br>")
Response.Write("<b>Request Method:</b> "&method &"<br>")
%>
```

Create Login Page with ASP Connected to Database

```
<%
        dim username, password, loginButton
        username=TRIM(Request("username"))
        password=TRIM(Request("password"))
        logButton=Request("loginButton")="Login"
        if logButton then
           Dim Con, sql, rec
           set Con = Server.CreateObject("ADODB.Connection")
           Con.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" & Server.MapPath(“feedback.mdb")
           'Select the record matching the username.
           sql = "SELECT * FROM tblusers WHERE UCase(username)='
        "& UCase(username) & "' AND UCase(password)=' " & UCase(password) & " ' "
           set rec=Con.execute(sql)
           'If no match found, EOF is not true.
           if NOT rec.EOF then
              Response.Redirect("somepage.asp") 'Change to page redirect to after login
           else
              blankError="Invalid username." 'EOF is true, no match found.
           end if
        end if
        %>
        <html>
        <head>
        <title>Login</title>
        </head>
        <body>
        <form name="productForm" method="post" action="<%=Request.ServerVariables("URL")%>">
        <center>
        <table border =1>
        <tr><td colspan="2">
        <%

        if blankError<>"" then
        Response.Write("<center><font color='red' size='3'>"&blankError&"</font></center>")
        end if
        %>
        </td></tr>
        <tr>
        <td><Strong><font face="courier new" size="3">Username:</font></strong></td>
        <td><input type="text" name="username" size="35"></td>
        </tr>
        <tr>
        <td><Strong><font face="courier new" size="3">Password</font></strong></td>
        <td><input type="password" name="password" size="35"></td>
        </tr>
        <tr><td colspan="2" align="center"><input type="submit" name="loginButton" value="Login">
        <input type="reset" name="reset" value="Clear"></td>
        </tr>
        </table>
        </center>
        </form>
        </body>
        </html>
```

The Session object

- When you are working with an application, you open it, do some changes and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are and what you do because the HTTP address doesn't maintain state.
- ASP solves this problem by creating a unique cookie for each user. The cookie is sent to the client and it contains information that identifies the user. This interface is called the Session object.
- The Session object is used to store information about, or change settings for a user session. Variables stored in the Session object hold information about one single user, and are available to all pages in one application. Common information stored in session variables are name, id, and preferences. The server creates a new Session object for each new user, and destroys the Session object when the session expires.

When does a Session Start?

A session starts when:

- A new user requests an ASP file, and the Global.asa file includes a Session_OnStart procedure
- A value is stored in a Session variable
- A user requests an ASP file, and the Global.asa file uses the <object> tag to instantiate an object with session scope

When does a Session End?

- A session ends if a user has not requested or refreshed a page in the application for a specified period. By default, this is 20 minutes.
- If you want to set a timeout interval that is shorter or longer than the default, you can set the **Timeout** property.

- The example below sets a timeout interval of 5 minutes:

```
<% Session.Timeout=5 %>
```

- To end a session immediately, you may use the **Abandon** method

```
<% Session.Abandon %>
```

Store and Retrieve Session Variables

- The most important thing about the Session object is that you can store variables in it.
- The example below will set the Session variable *username* to "Donald Duck" and the Session variable *age* to "50":

```
<% Session("username")="Donald Duck"
Session("age")=50 %>
```

When the value is stored in a session variable it can be reached from ANY page in the ASP application:

- Welcome <%Response.Write(Session("username"))%>
- The line above returns: "Welcome Donald Duck".

Remove Session Variables

- The Contents collection contains all session variables.
- It is possible to remove a session variable with the Remove method.
- The example below removes the session variable "sale" if the value of the session variable "age" is lower than 18:

```
<% If Session.Contents("age")<18 then
Session.Contents.Remove("sale")
End If  %>
```

To remove all variables in a session, use the RemoveAll method:

- `<% Session.Contents.RemoveAll() %>`

The Global.asa file

- The Global.asa file is an optional file that can contain declarations of objects, variables, and methods that can be accessed by every page in an ASP application.
- All valid browser scripts (JavaScript, VBScript, JScript, PerlScript, etc.) can be used within Global.asa.
- The Global.asa file can contain only the following:
  - Application events
  - Session events
  - <object> declarations
  - TypeLibrary declarations
  - the #include directive

ASP Response Object

- The ASP Response object is used to send output to the user from the server. Its collections, properties, and methods are described below:

Collections

Collection	Description
Cookies	Sets a cookie value. If the cookie does not exist, it will be created, and take the value that is specified

Properties

Property	Description
Buffer	Specifies whether to buffer the page output or not
CacheControl	Sets whether a proxy server can cache the output generated by ASP or not
Charset	Appends the name of a character-set to the content-type header in the Response object
ContentType	Sets the HTTP content type for the Response object
Expires	Sets how long (in minutes) a page will be cached on a browser before it expires
ExpiresAbsolute	Sets a date and time when a page cached on a browser will expire
IsClientConnected	Indicates if the client has disconnected from the server
Pics	Appends a value to the PICS label response header
Status	Specifies the value of the status line returned by the server

Methods

Method	Description
AddHeader	Adds a new HTTP header and a value to the HTTP response
AppendToLog	Adds a string to the end of the server log entry
BinaryWrite	Writes data directly to the output without any character conversion
Clear	Clears any buffered HTML output
End	Stops processing a script, and returns the current result
Flush	Sends buffered HTML output immediately
Redirect	Redirects the user to a different URL
Write	Writes a specified string to the output

ASP Request Object

Collections

Collection	Description
ClientCertificate	Contains all the field values stored in the client certificate
Cookies	Contains all the cookie values sent in a HTTP request
Form	Contains all the form (input) values from a form that uses the post method
QueryString	Contains all the variable values in a HTTP query string
ServerVariables	Contains all the server variable values

Properties

Property	Description
TotalBytes	Returns the total number of bytes the client sent in the body of the request

Methods

Method	Description
BinaryRead	Retrieves the data sent to the server from the client as part of a post request and stores it in a safe array

Thank You