vinodsrivastava.wordpress.com

Working with ASP objects

INTRODUCTION

An **object** is a bundle of methods and properties, held in memory on the server , that we , the developer , can manipulate to accomplish programming tasks.

Software objects feature **properties**(that describe the object) , **methods** (that allow the object to do things for your code) and **event** (code that is executed automatically when a certain situation occurs). Thus, an object is described in three categories. These are properties, methods and events.

Properties it describe something about an object. Properties are something read/write . A read/write

property can tell we something about the current state of the object , but we can also assign a new value to the property to deliberately change the state of the object.

Methods the task that an object can perform is called methods. Methods are actions that you can perform with or on an object .the cade in method is executed when it is called.
For example :-response.write ("The name is " & name)
Here **response.write** invokes the write method of the response object.

Events when an object tells the user to do something that happened is called an event.

ASP OBJECT MODEL

An object model is a representation of a set of objects and their relationships to one another. These relationships can take the form of containment, where one object is embedded inside of another .or they can take the form of parent-child relationship, where one object has a set of child objects associated with it.

The ASP objects are :

- **Request object :**provides methods and properties that enable your ASP's to retrieve information from the client, including information from forms.
- **Response object :**provides methods and properties that enable your ASP's to send information, including HTML , to the client.
- **Application object :**used to store information about your web application that can be made available to clients requesting pages from your site.
- **Session object :**used to store information about a particular client's browsing session.
- **Server object :**provides methods and properties that enable your ASP's to communicate with server.

These objects are "intrinsic" to ASP. They are always available to server memory and always ready to use.

<u>REQUEST OBJECT</u> the request object is used to access data sent from the browser to server, most often transmitted from forms appearing on the web page .

<u>Request object collection</u>

Request.QueryString

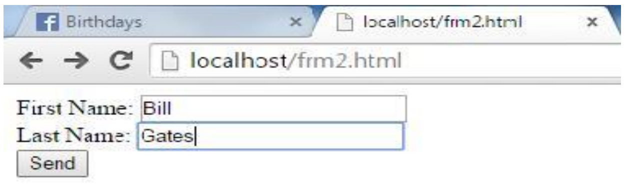
This collection is used to retrieve the values of the variables in the HTTP query string.

- **Information sent from a form with the GET method is visible to everybody (in the address field) and the GET method limits the amount of information to send.**

  Example

  <u>Code of frm2.html</u>

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

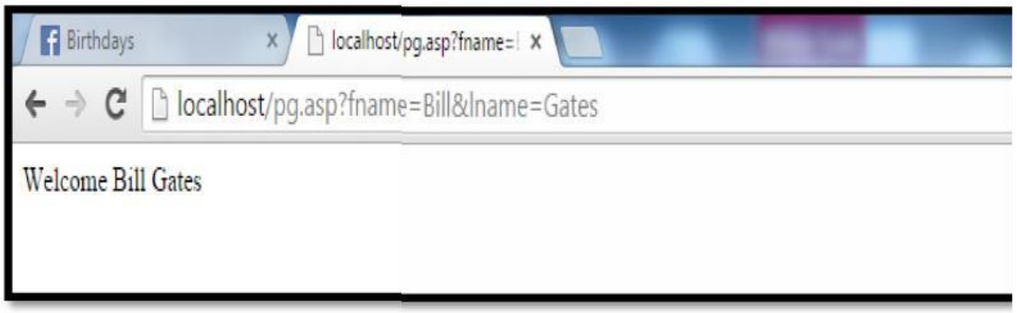


  <u>pg.asp code</u>

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

- **If a user typed "Bill" and "Gates" in the form example above, the url sent to the server would look like this:**

  http://localhost/pg.asp?fname=Bill&lname=Gates



Welcome Bill Gates

- ***The example above writes this into the body of a document:***

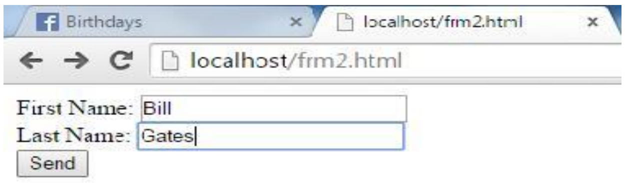
  -<u>Welcome Bill Gates</u>

Request.Form

- **It is used to retrieve the values of form elements posted to the HTTP request body, using the POST method of the <Form> Tag.**
- **Information sent from a form with the POST method is invisible to others.**
- **The POST method has no limitsyou can send a large amount of information.**

<u>Code of frm2.html</u>

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



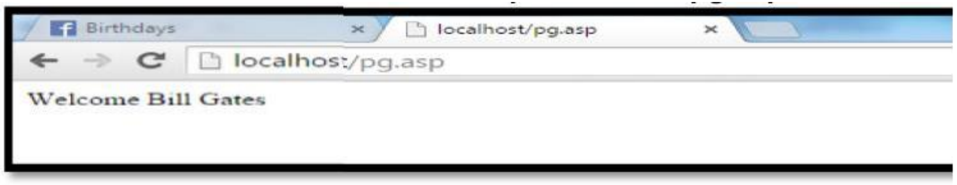
First Name: Bill
Last Name: Gates
Send

<u>pg.asp code</u>

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

- **If a user typed "Bill" and "Gates" in the form example above, the url sent to the server would look like this**:

http://localhost/pg.asp



Welcome Bill Gates

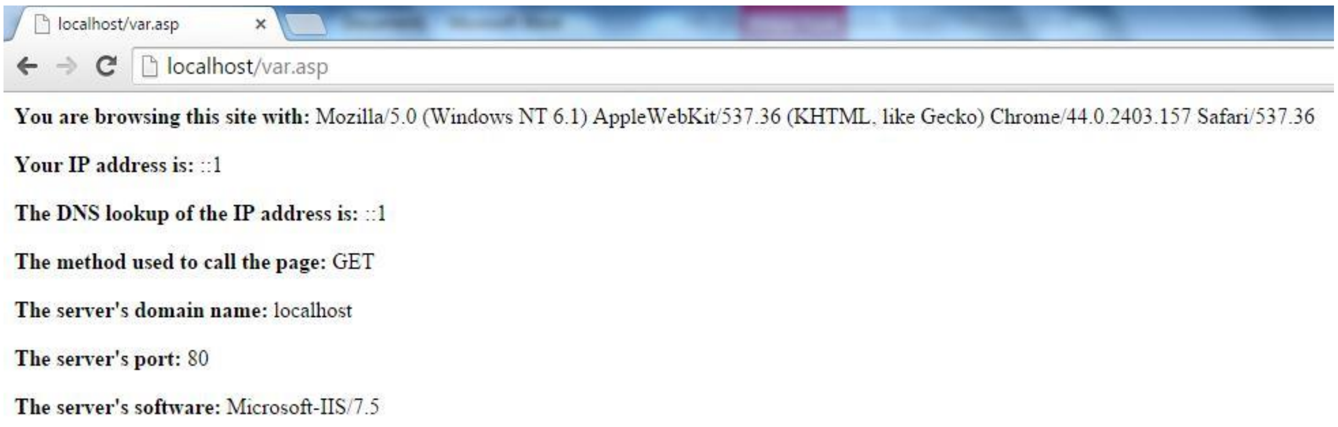
<u>ServerVariables</u> the server variables collection holds the entire HTTP headers and also additional items of information about the server and request.

- It is used to retrieve the values of predetermined environment variables.These values originate when client requests the server.
- *Syntax:<%* **Request.ServerVariables (*server environmentvariable*)%>**

- ***<u>HTTP_USER_AGENT</u>***it lists the compatibility , name and version of the browser .
- ***<u>REMOTE_ADDR</u>***unmapped user-name string send in by the user.
- ***<u>REMOTE_HOST</u>*** the name of the host making request.
- ***<u>SERVER_NAME</u>*** the server's host name , IP address as it would appear in self-referencing URLs .
- ***<u>SERVER_PORT</u>*** the port number to which the request was sent.
- ***<u>SERVER_SOFTWARE</u>***the name and version of the server software that answer the Request and run the gateway.
- ***<u>REQUEST_METHOD</u>***used to make the request. For HTTP , this is GET, HEAD , POST etc.

Example

```
<html>
<body>
<p><b>You are browsing this site with:</b>
<%Response.Write(Request.ServerVariables("http_user_agent"))%></p>
<p><b>Your IP address is:</b>
<%Response.Write(Request.ServerVariables("remote_addr"))%></p>
<p><b>The DNS lookup of the IP address is:</b>
<%Response.Write(Request.ServerVariables("remote_host"))%></p>
<p><b>The method used to call the page:</b>
<%Response.Write(Request.ServerVariables("request_method"))%></p>
<p><b>The server's domain name:</b>
<%Response.Write(Request.ServerVariables("server_name"))%></p>
<p><b>The server's port:</b>
<%Response.Write(Request.ServerVariables("server_port"))%></p>
<p><b>The server's software:</b>
<%Response.Write(Request.ServerVariables("server_software"))%></p>
</body>
</html>
```



You are browsing this site with: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/44.0.2403.157 Safari/537.36

Your IP address is: ::1

The DNS lookup of the IP address is: ::1

The method used to call the page: GET

The server's domain name: localhost

The server's port: 80

The server's software: Microsoft-IIS/7.5

<u>Cookies</u>cookies are text files written by the client browser , containing information sent by a server, which reside on the user's computer.

<u>Clientcertificate</u>it is a digital certificate exchanged between clients and server to verify the identity of the user attempting to contact the server.

<u>Property and Methods</u>

TotalBytes

It specifies the total number of bytes the client has sent in the body of the request. property is read-only

Syntax:

Counter = **Request.TotalBytes**

Counter - Specifies a variable to receive the total number of bytes that the client sends in the request. Example:

<% Response.Write(Request.TotalBytes) %>

- ***<u>Binaryreadmethod</u>*** it retrieves data sent to the server from the clients as part of a POST request sent by a form and store it in a Safe Array of bytes.

RESPONSE OBJECT the response object is used by a script to send information from the server to the browser.

Response Object – Properties

- **Buffer**

  It provides control of when the data is to be sent to the client..

  Syntax: **<%Response.Buffer [=** ***flag*****] %>**

  Flag **is True or False. This line is put on the top of the ASP page.**

  When it is set to **True** (default value)the server won't respond to client until whole page is processed or until Flush or End method are called (i.e, whole page is kept in a buffer and showed after completion).

  If it is set to **False** then server streams the page to the client as it is created.

- **CacheControl**

  It is used to control whether the page will be cached by the proxy server or not.

  Syntax:**Response.CacheControl [=** ***Cache Control Header*** **]**

  Cache Control Header **- Value is Public or Private**

  Default value is **Private** – This setting tells the proxy server that the contents of an ASP are private to a particular user and should not be cached.

- **Expires**

  It specifies the length of time after which page caching on a browser expires and fresh copy is retrieved.

  Syntax:**Response.Expires [=** ***number*****]**

  number **-The time in minutes after which the page caching expires**.

  – ***Example:***

  <%Response.Expires = 5%>

   The page will be removed form cache and a fresh page picked up after 5minutes.

- **ExpiresAbsolute**

  – This property specifies the date and time at which a page caching on a browser expires.When a page will not be changed very often, to extend the amount of time that the browser uses its copy of a page from the cache.

  Syntax:**Response.ExpiresAbsolute [= [*****date*****] [*****time*****]]**

  number **- The time in minutes before the page expires**

– *Example:***<% Response.ExpiresAbsolute=#Apr 1,2001 00:00:00# %>**

 The page will be removed form cache and a fresh page picked up on 1st April'2001.

Response object methods

Write method This method outputs a specified string to the browser OR outputs the value of an expression to the browser.

Syntax:<%**Response.Write(String or Variable or Function)%>**

Example:

```
<% Response.Write("Hello World") %>  'Display String Hello World
<% Response.Write(Time) %>            ' Display Current Time
      – Equivalent is the Output directive <%= %>
<%= Time%>
```

Redirect

Redirect method it instructs thebrowser to connect to a different URL.

Syntax ***<%response.redirect “contect.html ”%>***

Clear

Clear method empties the current page buffer without outputting the contents of the buffer.

This method will cause a run-time error if Response.Buffer has not been set to TRUE.

Syntax: **<%Response.Clear%>**

Flush

Flush method sends any previously buffered output to the client immediately, but continues processing the script.

This method will cause a run-time error if Response.Buffer has not been set to TRUE.

Syntax **<%response.flush%>**

End

***End* method causes the server to stop processing the script and sent the buffered output. Any further script instructions are not processed, nor is any remaining HTML sent.**

Syntax **<%response.end%>**

–

Binarywrite method this method sends text to the browser without character-set conversions.

Appendtolog method this method adds text to the web server log entry for this request

Syntax: **<%Response.AppendToLog *string*%>**

.

*Example:***<% Response.AppendToLog "My custom log message“ %>**

Example

```
<% Response.Buffer= “True”%>
<html>
<body>
<%Response.Write("1. India" & "<br>")
Response.Write("2. USA" & "<br>")
Response.clear
Response.Write("3. Kuwait" & "<br>")
Response.Flush
Response.Write("4. UK" & "<br>")
Response.clear
Response.Write("5. Germany" & "<br>")
Response.Ends
Response.Write("6. Japan" & "<br>")
Response.Write("7. France" & "<br>")
%>
</body>
</html>
```

OUTPUT

3. Kuwait

5 Germany

Explanation:

Response. Clear will erase first two buffered output that is 1. India 2. USA

But continue processing so 3. Kuwait will be buffered

Response.Flush will send current buffered output immediately to browser and continue processing so 3. Kuwait will be displayed and 4.Uk go to buffer

Response. Clear will erase it from buffer but process continue so Germany will go to buffer

Response.End the current buffer to Browser and stop further processing of script

So after Germany displayed on Browser no further

Application object

It is used to share information among all users of a given application.This information can be changed in one page and automatically gets reflected on all other pages.

<u>Application variables</u> one of the features of an application is that we can store information that can be accessed by all clients accessing the application. This information is store in what is known as an application scope variable.

- **Store and Retrieve Variable Values**
  - **Application variables** can be accessed and changed by any page in the application.
  - **Creating Application variables:**

    <% Application(“Greetings")=“Welcome” %>
  - **Retrieving an Application variable:**

    <% Response.Write(Application("Greetings")) %>
  - Once an Application variable has been assigned a value, it retains that value till the Web-Server shuts down.
- There are two methods **Remove** and **RemoveAll** to remove items from the contents collection.
- **Remove Method** removes specific item from the collection whereas **RemoveALL** removes all items of the collection.

- **Lock Method**
  - The **Lock** method **blocks other clients from modifying the variables** stored in the **Application** object, ensuring that only one client at a time can alter or access the **Application** variables who was currently accessing it.

    ***Syntax:* <%Application.Lock%>**
  - *Example:* To Count the no. of visitors to the Web Site

```
<% Application.Lock
        NumClicks = Application("NumVisits")
        NumClicks = NumClicks + 1
        Application("NumVisits") = NumClicks
        Application.Unlock %>
```

To avoid 2 users to click exactly at the same time.

- **UnLock Method**
  - The **Unlock** method enables other clients to modify the variables stored in the **Application** object after it has been locked using the Lock method.

    ***Syntax:* <% Application.UnLock %>**
  - If **Unlock** method is not explicitly called then the server unlocks the locked **Application** object when the .asp file ends or times out.

- **OnStart Event**
  - This event is written in **Global.asa** file and is triggered once when the **first page located in application is requested**.It is not triggered again until after IIS service is stopped or application is unloaded.

    Syntax:

```
    <SCRIPT  LANGUAGE=ScriptLanguage RUNAT=Server>
    Sub Application_OnStart
        . . .
    End Sub
</SCRIPT>
```

These events are coded in the Global.asa file.

- ❖ This event is fired before the **Session_OnStart** event.
- ❖ Only the Application and Server built-in objects are available in this event.
- ❖ Referencing the Session, Request, or Response objects in this event causes an error.

- **OnEnd Event**
  - This event is written in **Global.asa** file and is triggered when the **application quits or web server is stopped** by OS.

    Syntax:

    ```
    <SCRIPT LANGUAGE=ScriptLanguage RUNAT=Server>
    Sub Application_OnEnd
            . . .
    End Sub
    </SCRIPT>
    ```

❖ This event is fired afer the **Session_OnEnd** event.
❖ Only the Application and Server built-in objects are available in this event.
❖ The MapPath method of server object can't be called in this event.

Session Object

- The **Session object** is used to **store information** about **each user** entering the Web-Site and are available to all pages in one application.
- Common information stored in session variables are user's name, id, and preferences.
- The server **creates a new Session object** for each new user, and **destroys the Session object** when the session **expires** or is **abandoned** or the user logs out.

- **Store and Retrieve Variable Values**
  - The most important thing about the Session object is that you can store variables in it, like this:

    <% Session("username")="Tripti"
    Session("age")=24 %>
  - When the value is stored in a session variable it can be reached from any page in the ASP application by using: Session("username"):

    Welcome **<%Response.Write(Session("username"))%>**
  - You can also store user preferences in the Session object, and then access that preference to choose what page to return to the user.

- **Contents Collection**
  - The **Contents** collection contains all the variables that have been added and stored in a **Session object.**
    - *Syntax:* Session.Contents(*Key*)
      Session.Contents("username")

***Methods**:* The **Remove** method can remove a variable from a session.
Session.Contents.Remove(name|Index)
Session.Contents.RemoveAll()
- *Example:* Removes a session variable named "sale":

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

SESSION OBJECTS-PROPERTIES

- **SessionID**
  - The **SessionID** property is a unique identifier that is generated by the server when the session is first created and persists throughout the time the user remains at your web site.

- The session ID is returned as a **LONG** data type and read only property
  - ***Syntax:* <%Session.SessionID%>**
- *Example:* This is used to track where the user goes and records the pages the user visits.

```
<%Dim who,currentpage
who = session.sessionID
currentpage = Request.ServerVariables("script_name")
Response.AppendToLog  who%":" currentpage %>
```

- **TimeOut**
  - The **Timeout** property specifies the time before session ends automatically if client doesn't makes the request.
    - *Syntax:* Session.Timeout [= *nMinutes*]
    - Default value is 20 minutes
- **Abandon Method**
  - The **Abandon** method destroys all the objects stored in a **Session** object and releases their resources.It is used to manually end the session.

    ***Syntax:* <% Session.Abandon() %>**
  - *Example:*

```
<% Session.Abandon()
         Session("userid")=""
        Server.transfer("login.asp")%>
```

<u>SESSION OBJECTS-EVENTS</u>

- **OnStart Event**
  - The **Session_OnStart** event occurs when the server creates a new session, that is new user requests an ASP file and is written in **Global.asa** file.
  - ***Syntax:***

```
<SCRIPT LANGUAGE=ScriptLanguage RUNAT=Server>
 Sub Session_OnStart
      . . .
 End Sub
</SCRIPT>
```

These events are coded in the Global.asa file.

- ❖ The server processes this script prior to executing the requested page.
- ❖ This event is a good time to set any session-wide variables, because they will be set before any pages are accessed.
- ❖ All the built-in objects can be accessed in this event.
- ❖ Session only starts if there is a session_onstart routine.

- **OnEnd Event**
  - The **Session_OnEnd** event occurs when a session is **abandoned** or **times out** or a user has not requested or refreshed a page in the ASP application for a specified period.This event is written in **Global.asa** file.
  - ***Syntax:***

```
<SCRIPT LANGUAGE=ScriptLanguage RUNAT=Server>
Sub Session_OnEnd
       . . .
End Sub
</SCRIPT>
```

The Global.asa file

- This file is an optional file containing declarations of objects, variables, and methods that can be accessed by every page in an ASP application .
- The Global.asa file must be stored in the **root directory** of the ASP application which is identified as the **virtual directory** and each application having **only one Global.asa**..
- Global.asa files can contain only the following:
    - Application events
    - Session events
    - <object> declarations
    - TypeLibrary declarations

Changes to the Global.asa file require a restart of the server to recognize them

- **Standard format of Global.asa**
    - Subroutines are created to handle these events in the Global.asa file:

```
<script language="vbscript" runat="server">
sub Application_OnStart
            ......some vbscript code
end sub
sub Application_OnEnd
            ......some vbscript code
end sub
sub Session_OnStart
            ......some vbscript code
end sub
sub Session_OnEnd
            ......some vbscript code
end sub
</script>
```

- **The Global.asa contains four types of Standard Events:**
    - **Application_OnStart -** This event occurs when the **FIRST user calls the first page from an ASP application**. This event occurs after Web server is restarted or after the Global.asa file is edited as changes to the file require a restart of the server to recognize them. Eg. DB Connectivity.
    - **Application_OnEnd -** This event occurs after the **LAST user has ended the session,** typically when a Web Server stops. Eg. Delete records to clean up settings after the Application stops or write information to log files.
    - **Session_OnStart -** This event occurs **EVERY time a new user requests the first page** in the ASP application.Eg. Login page to be displayed first time a user enters.
    - **Session_OnEnd -** This event occurs **EVERY time a user ends a session**. A user ends a session after a page has not been requested by the user for a specified time (by default this is 20 minutes). A session also ends if the user closes the web browser, or goes to someone else's web page.

• Server Object

- The **Server object** provides **access to methods and properties on the Server**
- It enables to work with **external Objects** registered on the Server including **Components that are bundled with IIS**.

Server Object – Properties

- **ScriptTimeOut**
  - The **ScriptTimeout** property specifies the maximum amount of time a script can run before it is terminated.
    - The timeout will not take effect while a server component is processing.
    - The time is specified in seconds. The default value is 90 seconds,
      Syntax:* <%Server.ScriptTimeout = *NumSeconds%>
      - *Example:* <% Server.ScriptTimeout = 100 %>

- **CreateObject Method**
  - The **CreateObject** method is used for creating an instance of a **Server Component (External Object)**. It can be used for any component that is correctly installed on our server.An instance must be explicitly created before using these external objects.
    - *Syntax:* Set MyObj = Server.CreateObject(progID)

    Set MyObj = Server.CreateObject(library.classID)
  - *Example:*
    <%Set MyAd = Server.CreateObject("MSWC.AdRotator") %>
  - Destroying an object
    <%Set MyObj = **Nothing**%>

- **Execute Method**

  - The **Execute** method calls an .asp file and processes it as if it were **part of the calling ASP script**.
    - ***Syntax:* Server.Execute(Path)**
    - *Example:* <% Server.Execute (“abc.asp") %>

- **Transfer Method**
  - The **transfer** method **sends all of the information** that has been assembled for processing by one .asp file to a second .asp file.
    - ***Syntax:* <%Server.Transfer(Path)%>**
    - *Example:* <% Server.Transfer(“abc.asp”) %>

- **MapPath Method**

  - The **MapPath** method returns **maps the specified relative or virtual path** to the corresponding **physical directory path** on the server.
    - ***Syntax:* <%Server.MapPath(“Virtual Path”)%>**
    - *Example:* <%Server.MapPath(www.myfile.com/thefile.asp)%>

    It returns the actual physical path where the file actually resides like
    C:\inetpub\wwwroot\thefile.asp

Example
example.asp

```
<%
Response.Write("Welcome to Faips" &"<br>")
Response.Write(" Explore " &"<br>"))
Server.Execute("lab.asp")
Server.Transfer("School.asp")
Response.Write(" Hope You Enjoyed ")
%>
```

lab.asp
```
<%
Response.Write("The Multimedia Lab " &"<br>")
Response.Write(" Hope You Like the Lab " &"<br>")
%>
```

School.asp
```
<%
Response.Write("School has 5000 Students & 350 teachers" &"<br>")
%>
```

Output
```
Welcome to Faips
Explore
The Multimedia Lab
Hope You Like the Lab
School has 5000 Students & 350 teachers
```

"

Explanation
First two line of example.asp containing response write so they will be displayed as written
Server.Execute(lab.asp) will bring the output of lab.asp into this file so third line will be displayed along first two lines
Server.Transfer(School.asp) will take the first three output and transfer it to school.asp so these will be displayed and then code of school .asp
Since the complete control go to school.asp so no further processing of script in example.asp so the last line response.Write("Hope You Enjoyed") will not be displayed

Error Object

- The **ASPError** object is used to **get information of any error** that occurs in scripts in an ASP page.
- The **ASPError** object is created when **Server.GetLastError** is called, so the error information can only be accessed by using the Server.GetLastError method.

ASPError Object

- **Syntax : <% ASPError.property %>**

Property is read-only giving information about the error.

ASPCode	Returns an error code generated by IIS.
Source	Returns the actual source code of the line that caused the error.
File	Returns name of .asp file being processed when error occurred.
Line	Indicates the line within the .asp file that generated the error.
Column	Indicates column position in the .asp file that generated the error.
Description	Returns a short description of the error
ASPDescription	Returns a detailed description of the ASP-related error.