Paper Code: CSCS351 # WEB TECHNOLOGY

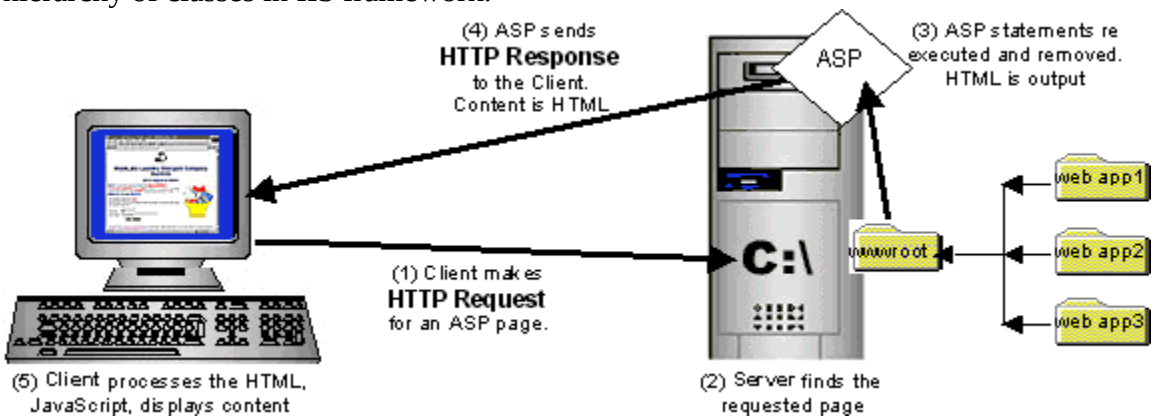
MODULE – V Introduction to ASP – Database Management with ASP: Database access with ADO, working with ADO's Connection object, Using Command objects, Working with ADO's Recordset.

What is ASP

ASP is a web development platform, which provides a programming model, a comprehensive software infrastructure and various services required to build up robust web applications for PC, as well as mobile devices.

ASP works on top of the HTTP protocol, and uses the HTTP commands and policies to set a browser-to-server bilateral communication and cooperation.

ASP is a part of Microsoft Windows platform. ASP applications are compiled codes, written using the extensible and reusable components or objects present in IIS framework. These codes can use the entire hierarchy of classes in IIS framework.



ASP uses scripting on the server to generate content that is sent to the client's web browser via HTTP response. The ASP interpreter reads and executes all script code between <% and %> tags, the result of which is content generation. These scripts were written using VBScript, JScript, or PerlScript. The @Language directive, the <script language="manu" runat="server" /> syntax or server configuration can be used to select the language. In the example below, Response.Write Now() is in an HTML page; it would be dynamically replaced by the current time of the server.

Server side	Client Side
The server's current time:	The server's current time:
<%	
Response.Write Now()	8/11/2015 6:24:45 PM
%>	

Web pages with the .asp filename extension use ASP, although some web sites disguise their choice of scripting language for security purposes by using the more common .htm or .html extensions. Pages with the .asp (or) .aspx extension use compiled ASP.NET; however, ASP.NET pages may still include some ASP scripting. The introduction of ASP.NET led to use of the term Classic ASP for the original technology.

Sun Java System ASP (formerly ChiliSoft ASP) was a popular and reportedly complete emulator,[4] but it has been discontinued.

Command Object

The ADO Command object is used to execute a single query against a database. The query can perform actions like creating, adding, retrieving, deleting or updating records.
If the query is used to retrieve data, the data will be returned as a RecordSet object. This means that the retrieved data can be manipulated by properties, collections, methods, and events of the Recordset object.

The major feature of the Command object is the ability to use stored queries and procedures with parameters.

ProgID

```
set objCommand=Server.CreateObject("ADODB.command")
```

Properties

Property	Description
ActiveConnection	Sets or returns a definition for a connection if the connection is closed, or the current Connection object if the connection is open
CommandText	Sets or returns a provider command
CommandTimeout	Sets or returns the number of seconds to wait while attempting to execute a command
CommandType	Sets or returns the type of a Command object
Name	Sets or returns the name of a Command object
Prepared	Sets or returns a Boolean value that, if set to True, indicates that the command should save a prepared version of the query before the first execution
State	Returns a value that describes if the Command object is open, closed, connecting, executing or retrieving data

Connection Object

The ADO Connection Object is used to create an open connection to a data source. Through this connection, you can access and manipulate a database.

If you want to access a database multiple times, you should establish a connection using the Connection object. You can also make a connection to a database by passing a connection string via a Command or Recordset object. However, this type of connection is only good for one specific, single query.

ProgID

```
set objConnection=Server.CreateObject("ADODB.connection")
```

Properties

Property	Description
Attributes	Sets or returns the attributes of a Connection object
ConnectionString	Sets or returns the details used to create a connection to a data source
CursorLocation	Sets or returns the location of the cursor service
DefaultDatabase	Sets or returns the default database name
Mode	Sets or returns the provider access permission

Methods

Method	Description
BeginTrans	Begins a new transaction
Cancel	Cancels an execution
Close	Closes a connection
CommitTrans	Saves any changes and ends the current transaction
Execute	Executes a query, statement, procedure or provider specific text
Open	Opens a connection
OpenSchema	Returns schema information from the provider about the data source
RollbackTrans	Cancels any changes in the current transaction and ends the transaction

Events

Event	Description
BeginTransComplete	Triggered after the BeginTrans operation
CommitTransComplete	Triggered after the CommitTrans operation
ConnectComplete	Triggered after a connection starts
Disconnect	Triggered after a connection ends
ExecuteComplete	Triggered after a command has finished executing
InfoMessage	Triggered if a warning occurs during a ConnectionEvent operation
RollbackTransComplete	Triggered after the RollbackTrans operation
WillConnect	Triggered before a connection starts
WillExecute	Triggered before a command is executed

The Server object
The server object allows connections to databases (ADO), filesystem, and use of components installed on the server.

ADO Recordset

To be able to read database data, the data must first be loaded into a recordset.

Create an ADO Table Recordset

After an ADO Database Connection has been created, as demonstrated in the previous chapter, it is possible to create an ADO Recordset. In the following example rs is the record set object.

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"

set rs = Server.CreateObject("ADODB.recordset")
rs.Open "SELECT * FROM Customers", conn

do until rs.EOF
  for each x in rs.Fields
    Response.Write(x.name)
    Response.Write(" = ")
    Response.Write(x.value & "<br>")
  next
  Response.Write("<br>")
  rs.MoveNext
loop

rs.close
conn.close
%>

</body>
</html>
```

OUTPUT

CustomerID = ALFKI
CompanyName = ABCD
ContactName = Maria Anders
Address = Obere Str. 57
City = Berlin
PostalCode = 12209
Country = Germany

CustomerID = BERGS
CompanyName = XYZ

ContactName = Christina Berglund
Address = Berguvsvägen 8
City = Luleå
PostalCode = S-958 22
Country = Sweden

ADO Add Records

We may use the SQL INSERT INTO command to add a record to a table in a database.

Add a Record to a Table in a Database

We want to add a new record to the Customers table in the Northwind database. We first create a form that contains the fields we want to collect data from:

```
<html>
<body>

<form method="post" action="demo_add.asp">
<table>
<tr>
<td>CustomerID:</td>
<td><input name="custid"></td>
</tr><tr>
<td>Company Name:</td>
<td><input name="compname"></td>
</tr><tr>
<td>Contact Name:</td>
<td><input name="contname"></td>
</tr><tr>
</table>
<br><br>
<input type="submit" value="Add New">
<input type="reset" value="Cancel">
</form>

</body>
</html>
```

When the user presses the submit button the form is sent to a file called "demo_add.asp". The "demo_add.asp" file contains the code that will add a new record to the Customers table:

```
<html>
<body>

<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"
```

```
sql="INSERT INTO customers (customerID,companyname,"
sql=sql & "contactname,address,city,postalcode,country)"
sql=sql & " VALUES "
sql=sql & "('" & Request.Form("custid") & "',"
sql=sql & "'" & Request.Form("compname") & "',"
sql=sql & "'" & Request.Form("contname")& "')"

on error resume next
conn.Execute sql,recaffected
if err<>0 then
  Response.Write("No update permissions!")
else
  Response.Write("<h3>" & recaffected & " record added</h3>")
end if
conn.close
%>

</body>
</html>
```