ASP

Purbanchal University: BCA V

What is ASP?

- In the language of Microsoft:
  - Active Server Pages is an open, compile-free application environment in which you can combine HTML, scripts, and reusable ActiveX server components to create dynamic and powerful Web-based business solutions.
  - Active Server Pages enables server side scripting for IIS with native support for both VBScript and JScript.
- Translated into plain English, that reads –
  - Active Server Pages (ASPs) are Web pages that contain ***server-side scripts*** in addition to the usual mixture of text and HTML tags.
  - Server-side scripts are special commands you put in Web pages that are processed before the pages are sent from the server to the web-browser of someone who's visiting your website.

What Can You Do with Active Server Pages?

- There are many things you can do with Active Server Pages. Few are listed below:
  - You can display date, time, and other information in different ways.
  - You can make a survey form and ask people who visit your site to fill it out, send emails, save the information to a file, etc
  - You can have a database, which people can access via the web. People can get information from the database as well as update or insert information into it.
  - You can password-protect certain sections of your site, and make sure that only authorized users can see that information.
  - The possibilities are virtually endless. Most widgetry that you see on web pages nowadays can be easily done using ASP.

What is localhost?

- Let us first see, what we mean by a hostname.
  - Whenever you connect to a remote computer using it's URL, you are in effect calling it by its hostname.
  - For example, when you type in **http://www.google.com/** you are really asking the network to connect to a computer named www.google.com.
  - It is called the "hostname" of that computer.
- Localhost is a special hostname.

- It always references your own machine.
- So what you just did, was to try to access a webpage on your own machine (which is what you wanted to do anyway.)
- For testing all your pages, you will need to use localhost as the hostname.
- By the way, there is also a special IP address associated with localhost, that is 127.0.0.1
- So you could as well have typed: **http://127.0.0.1/** and would have received the same page.
- To access pages in a virtual directory called myscripts for example, you should type in: **http://localhost/myscripts/** in the address bar.

First Program in ASP

- The first Example in ASP which displays "Hello World".

```
<HTML>
<HEAD>
<TITLE>Hello, World !</TITLE>
</HEAD>
<BODY>
    <%
        Response.Write "Hello, World!"
    %>
</BODY>
</HTML>
```

- The above program can be written in the following way:

```
<HTML>
<HEAD>
<TITLE>Hello, World !</TITLE>
</HEAD>
<BODY>
    <%= "Hello, World!" %>
</BODY>
</HTML>
```

Displaying Current Date and Time

- The following program displays the current date and time of the web-server.

```
<HTML>
<HEAD>
<TITLE>Displaying Date and Time !!!</TITLE>
</HEAD>
<BODY>
    <%
        Response.Write now
    %>
</BODY>
</HTML>
```

- The output of the above program is: **day/month/year hrs:min:sec (7/10/2000 12:35:31 AM).**

The Big If

- The simplest of constructs, found in every language, is the If-Then-Else statement.
- Example of using If – Then – Else statement.

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

For - Next Loops

- The syntax of For – Next Loops is as follows:

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

- Here vbCrLf is used to display the output in the next line.

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
%>
```

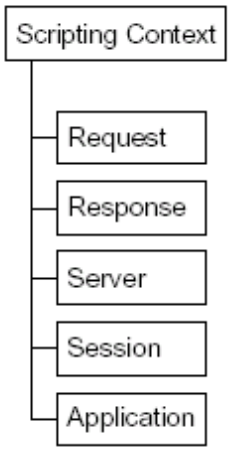
Select Case:

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":
        Response.Write "You chose 1"
    Case "2":
        Response.Write "You chose 2"
    Case "3":
        Response.Write "You chose 3"
    Case "4":
        Response.Write "You chose 4"
  End Select
%>
```

The Object Model

- An Object Model is simply a hierarchy of objects that you may use to get services from.
- In the case of ASP, all commands are issued to certain inbuilt objects that correspond to the Client Request, Client Response, the Server, the Session & the Application respectively.
- All of these are for global use



Request: To get information from the user

Response: To send information to the user

Server: To control the Internet Information Server

Session: To store information about and change settings for the user's current Web-server session

Application: To share application-level information and control settings for the lifetime of the application

The Request object

- The Request object retrieves the values that the client browser passed to the server during an HTTP request.
- Syntax: Request*[.collection|property|method](variable).*
- **Collections**
  1. *ClientCertificate*

     To get the certification fields from the request issued by the Web browser.
  2. *Cookies*

     The values of cookies sent in the HTTP request.
  3. *Form*

     The values of form elements in the HTTP request body.
  4. *QueryString*

     The values of variables in the HTTP query string.
  5. *ServerVariables*

     The values of predetermined environment variables.
- **Properties**
  1. *TotalBytes*

     Read-only. Specifies the total number of bytes the client is sending in the body of the request.

Methods

1. BinaryRead

Retrieves data sent to the server from the client as part of a POST request. Variable parameters are strings that specify the item to be retrieved from a collection or to be used as input for a method or property.

The Response object

- The Response object is used to send information to the user. The Response object supports only Cookies as a collection (to set cookie values). The Response object also supports a number of properties and methods.
- **Syntax:** *Response.collection|property|method*
- **Collections**

1. Cookies

Specifies cookie values. Using this collection, you can set cookie values.

- **Properties**

1. Buffer

Indicates whether page output is buffered.

2. CacheControl

Determines whether proxy servers are able to cache the output generated by ASP.

3. Charset

Appends the name of the character set to the content-type header.

4. ContentType

Specifies the HTTP content type for the response.

5. Expires

Specifies the length of time before a page cached on a browser expires.

6. ExpiresAbsolute

Specifies the date and time on which a page cached on a browser expires.

7. IsClientConnected

Indicates whether the client has disconnected from the server.

8. Status

The value of the status line returned by the server.

- **Methods**

1. AddHeader

Sets the HTML header name to value.

2. AppendToLog

Adds a string to the end of the Web server log entry for this request.

3. BinaryWrite

Writes the given information to the current HTTP output without any character-set conversion.

4. Clear

Erases any buffered HTML output.

5. End

Stops processing the .asp file and returns the current result.

6. Flush

Sends buffered output immediately.

7. Redirect

Sends a redirect message to the browser, causing it to attempt to connect to a different URL.

8. Write

Writes a variable to the current HTTP output as a string.

The Server object

- The Server object provides access to methods and properties on the server. Most of these methods and properties serve as utility functions.
- **Syntax:** Server.*property|method*
- **Properties**

1. ScriptTimeout

The amount of time that a script can run before it times out.

- **Methods**

1. CreateObject

Creates an instance of a server component. This component can be any component that you have installed on your server (such as an ActiveX).

2. MapPath

Maps the specified virtual path, either the absolute path on the current server or the path relative to the current page, into a physical path.

3. URLEncode

Applies URL encoding rules, including escape characters, to the string.

The Session object

- You can use the Session object to store information needed for a particular user-session.
- Variables stored in the Session object are not discarded when the user jumps between pages in the application; instead, these variables persist for the entire user-session.
- The Web server automatically creates a Session object when a Web page from the application is requested by a user who does not already have a session.

- The server destroys the Session object when the session expires or is abandoned.
- One common use for the Session object is to store user preferences.
- Note Session state is only maintained for browsers that support cookies.
- **Syntax:** Session.*collection|property|method*
- **Collections**

1. Contents

Contains the items that you have added to the session with script commands.

2. StaticObjects

Contains the objects created with the <OBJECT> tag and given session scope.

- **Properties**

1. CodePage

The codepage that will be used for symbol mapping.

2. LCID

The locale identifier.

3. SessionID

Returns the session identification for this user.

4. Timeout

The timeout period for the session state for this application, in minutes.

- **Methods**

1. Abandon

This method destroys a Session object and releases its resources.

- **Events**

Scripts for the following events are declared in the global.asa file.

1. Session_OnEnd

2. Session_OnStart

The Application object

- The Application object can store information that persists for the entire lifetime of an application (a group of pages with a common root).
- Generally, this is the whole time that the IIS server is running.
- This makes it a great place to store information that has to exist for more than one user (such as a page counter).
- Because the Application object is shared by all the users, threading can be a nightmare to implement.
- You can use the Application object to share information among all users of a given application.
- Because the Application object can be shared by more than one user, there are Lock and Unlock methods to ensure that multiple users do not try to alter a property simultaneously.

- **Syntax:** Application.*method*
- **Collections**
  1. *Contents*

     Contains all of the items that have been added to the Application through script commands.
  2. *StaticObjects*

     Contains all of the objects added to the session with the <OBJECT> tag.
  3. *Lock*

     The Lock method prevents other clients from modifying Application object
- **Properties.**

  1. *Unlock*

     The Unlock method allows other clients to modify Application object properties.
- **Events**
  1. *Application_OnEnd*
  2. *Application_OnStart*

Data Manipulation in ASP

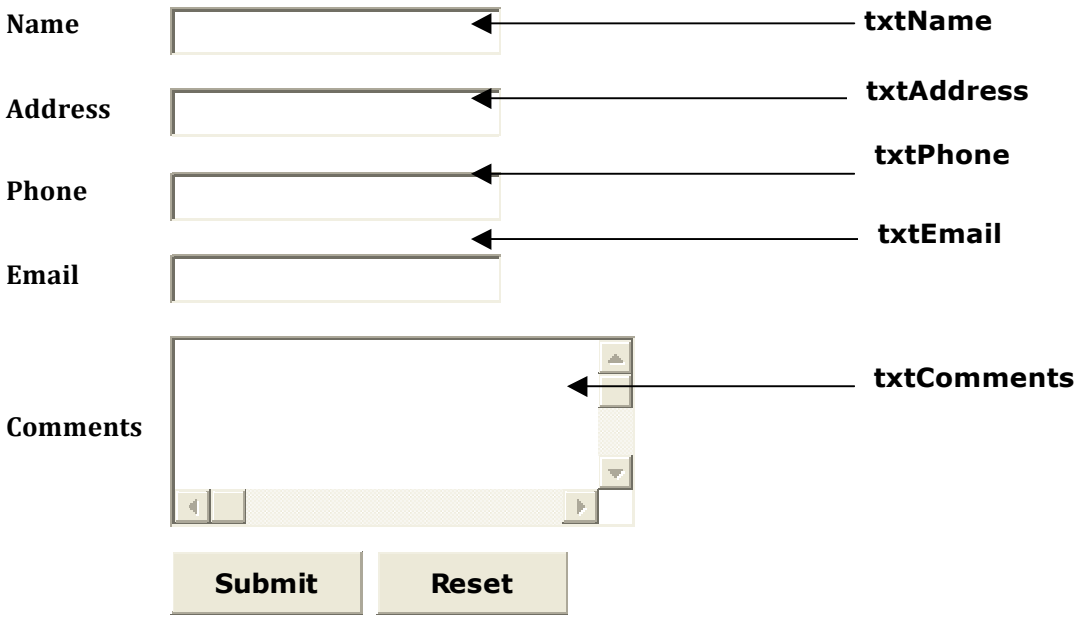
- Lets imagine the database file: **database.mdb** with the table **comments**. The table **comments** has the following fields:

Fields	Data-type
id (primary key)	auto-number
name	text
address	text
phone	number
email	text
comments	memo

- The inserting operating looks as below:

Inserting data in Database using ASP

- For inserting the values in database we first have to create a database.
- **SQL** provides **insert into** statements for inserting data into database.



- In the above table, the names of all text boxes are indicated by the arrow.
- Let imagine the above table is in the page **feedback.asp**.
- In this file, **<form action="feedback_result.asp" method="post" name="myform">**

- Now, in the file **feedback_result.asp** we will write the following asp codes to insert the values inserted by user in the above field, in the table **comments.**

```
/************************** feedback_result.asp ********************************/
<%@LANGUAGE="VBSCRIPT"%>
<%option explicit%>
<%
  dim conn, str1, strcon
  set conn=server.createobject("adodb.connection")
  strcon = "PROVIDER=MSDASQL;"
  strcon = strcon & "DRIVER={Microsoft Access Driver (*.mdb)};"
  strcon = strcon & "DBQ=" & Server.mappath("database.mdb")&";"
  conn.open=strcon
%>

<%
   dim var1,var2,var3,var4,var5
   var1 = request.Form("txtName")
   var2 = request.Form("txtAddress")
   var3 = request.Form("txtPhone")
   var4 = request.Form("txtEmail")
   var5 = request.Form("txtComments")
   sql = "insert into feedback ( name, address, phone, email, comments) values ('" & var1 & "','" & var2 &     "','" &
   var3 & "','" & var4 & "','" & var5 & "')"
   set rs = conn.execute(sql)
 Response.Write ("Message Inserted successfully !!!")
%>
```

- First line of the ASP file should be **<%@LANGUAGE = "VBSCRIPT"%>**.
- The line **<%option explicit%>** tells that the variable should be declared.
- **request.Form** is used to retrieve the value from the text-field from the page **feedback.asp** into the page **feedback_result.asp**.
- **insert into** statement is used to insert values into the database.
- **Response.Write** is used to write the string in the browser.
    - The statement, Server.CreateObject() is used to create an instance of a COM object which has the ProgID ADODB.Connection.

- strcon = "PROVIDER=MSDASQL;" strcon = strcon & "DRIVER={Microsoft Access Driver (*.mdb)};", is a string expression that tells our object where to locate the database, and more importantly, what type the database is – whether it is an Access database, or a Sybase database, or else, is it Oracle.

Retrieving data from Database using ASP

- The following line of codes retrieves the data from the database and display in the web-browser.
- The codes are written in the file **view_feedback.asp** file.
- We need **SQL SELECT** statement for retrieving data from the database.
- Lets imagine the same table as before to retrieve the data from the database.

```
/************************** view_feedback.asp ********************************/
<%@LANGUAGE="VBSCRIPT"%>
<%option explicit%>
<%
  dim conn, str1, strcon
  set conn=server.createobject("adodb.connection")
  strcon = "PROVIDER=MSDASQL;"
  strcon = strcon & "DRIVER={Microsoft Access Driver (*.mdb)};"
  strcon = strcon & "DBQ=" & Server.mappath("database.mdb")&";"
  conn.open=strcon
%>

<%
   dim sql1, rs, var1,var2,var3,var4,var5
   sql1 = "select * from feedback"
   set rs = conn.execute(sql1)
   do while not rs.eof
            Response.Write ( rs("name") )
            Response.Write ( rs("address") )
            Response.Write ( rs ("phone") )
            Response.Write ( rs ("email") )
            Response.Write ( rs ("comments") )
            rs.movenext
   Loop
%>
```

Updating data into Database using ASP

- The following line of codes updates the fields of database.
- We have to use SQL UPDATE statement to update the database.
- This lines of codes, for example is written in a file: update_feedback.asp.
- Let's imagine the same table as before for updating table.
- Let the same table (HTML) exist for updating table as in the case of inserting data.

```
/************************** update_feedback.asp ********************************/
<%@LANGUAGE="VBSCRIPT"%>
<%option explicit%>
<%
  dim conn, str1, strcon
  set conn=server.createobject("adodb.connection")
  strcon = "PROVIDER=MSDASQL;"
  strcon = strcon & "DRIVER={Microsoft Access Driver (*.mdb)};"
  strcon = strcon & "DBQ=" & Server.mappath("database.mdb")&";"
  conn.open=strcon
%>

<%
   dim a, b, c, d, e, rs, sql
   a = request.Form("txtName")
   b = request.Form("txtAddress")
   c = request.Form("txtPhone")
   d = request.Form("txtEmail")
   e = request.Form("txtComments")
   sql = "update feedback set name = '"& a & "', address = '"& b & "', phone = '"& c & "', email = '"& d & "' ,
     comments = '"& e & "' where      serial="&id
   set rs = conn.execute(sql)
%>
```

Deleting data from Database using ASP

- Let's imagine the same table for deleting data from database.
- The following codes are written in the file: delete_feedback.asp.
- In this file we have passed the id, which has to be deleted, from the next file.

- The following codes delete the records from the database.
- We use SQL DELETE statement to delete the data from the table.

/********** delete_feedback.asp **********/

```
<%@LANGUAGE="VBSCRIPT"%>
<%option explicit%>
<%
  dim conn, str1, strcon
  set conn=server.createobject("adodb.connection")
  strcon = "PROVIDER=MSDASQL;"
  strcon = strcon & "DRIVER={Microsoft Access Driver (*.mdb)};"
  strcon = strcon & "DBQ=" & Server.mappath("database.mdb")&";"
  conn.open=strcon
%>
<%
        dim fno, rs, sql
        fno = request.QueryString("nid")
        sql = "delete from feedback where id=" &fno
        set rs = conn.execute(sql)
        Response.Write("Record Deleted Successfully !!!")
%>
```

SESSION

When you are working with an application on your computer, 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 open the application and when you close it. However, 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 user's computer and it contains information that identifies the user. This interface is called the Session object.

The Session object stores information about, or change settings for a user session.

Variables stored in a 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, use the **Timeout** property.

The example below sets a timeout interval of 5 minutes:

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

Use the **Abandon** method to end a session immediately:

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

Note: The main problem with sessions is WHEN they should end. We do not know if the user's last request was the final one or not. So we do not know how long we should keep the session "alive". Waiting too long for an idle session uses up resources on the server, but if the session is deleted too soon the user has to start all over again because the server has deleted all the information. Finding the right timeout interval can be difficult!

Tip: Only store SMALL amounts of data in session variables!

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".

You can also store user preferences in the Session object, and then access that preference to choose what page to return to the user.

The example below specifies a text-only version of the page if the user has a low screen resolution:

```
<%If Session("screenres")="low" Then%>
  This is the text version of the page
<%Else%>
  This is the multimedia version of the page
<%End If%>
```

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()
%>
```

Loop Through the Contents Collection

The Contents collection contains all session variables. You can loop through the Contents collection, to see what's stored in it:

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

dim i
For Each i in Session.Contents
  Response.Write(i & "<br />")
Next
%>
```

Result:

```
username
age
```

If you do not know the number of items in the Contents collection, you can use the Count property:

```
<%
dim i
dim j
j=Session.Contents.Count
Response.Write("Session variables: " & j)
For i=1 to j
  Response.Write(Session.Contents(i) & "<br />")
Next
%>
```

Result:

```
Session variables: 2
Donald Duck
50
```

Loop Through the StaticObjects Collection

You can loop through the StaticObjects collection, to see the values of all objects stored in the Session object:

```
<%
dim i
For Each i in Session.StaticObjects
  Response.Write(i & "<br />")
Next
%>
```

Advanced Server-side Issues

- The important issues that should be covered during the server-side programming are listed below:

1. Security

- Security is one of the most important issues in the server-side.
- Starting from the submission of the user data to the delivery of the server response the data must be protected in every node of the Internet.
- Protection of data stored in the server is also important.
- Robust encryption systems should be implemented for data transfer from either side – client or server.

2. Performance

- The performance should not degrade below the acceptable level when a number of users connect to the application at the same time.
- This is a major performance issue.
- If databases are used in the application, creating and maintaining a number of connection pools can significantly increase the performance of the application.
- The performance of the application can be increased by adding multiple servers for the same application.
- This is called Server Farm.
- If one Server is very busy, other Server can respond to the new user's requests.
- Google search engine consists of a forest of servers – more than 3500 servers.
- That's why you get a search result within a friction of a second.

3. Load balancing

- It is possible in Internet that more requests may come to some servers and they get heavily loaded, but some of the servers do not get so many requests and stay idle.
- Therefore, it is desirable that all the servers get a fair amount of time.
- The process is called **load balancing** and involves the use of some networking tools to measure the loads on the server.

4. Session management

- It is maintaining the user identification and other properties throughout a browser session.
- The same information can be stored persistently on the client side using cookies or on the server-side using database.

- The session management in a web farm is bit complicated.
- In such case, we store the state information in one of the relational databases that is accessible from all servers in the farm.
- Every-time the user logs in check his/her record in the database and gets the state information.

5. Internationalization

- A web application will attract more users if users can access the application in the language of their choice.
- Several tools are available that translate the content of the page while delivering it to the client.
- This is another reason why Google is the most popular search engine in the Internet.

Download all source code for Feedback from http://sarojpandey.com.np/academic/

1. Default.asp

```
<h3>Database Operation With ASP</h3>
<a href="Feedback.asp">Feedback</a>
<p> <a href="Select.asp">View All</a></p>
```

2. dbConnect.inc

```
<%
dim conn, strcon
set conn=server.createobject("adodb.connection")
strcon = "PROVIDER=MSDASQL;"
strcon = strcon & "DRIVER={Microsoft Access Driver (*.mdb)};"
strcon = strcon & "DBQ=" & Server.mappath("database.mdb")&";"
conn.open=strcon
%>
```

3. Feedback.asp

```
<h2 align="center">Feedback Form</h2>
<form name="form" method="post" action="Insert.asp">
  <table width="28%" border="1" align="center">
    <tr>
      <td width="35%">Name :</td>
      <td width="65%"><input name="txtName" type="text" id="txtName"></td>
    </tr>
    <tr>
      <td>Address :</td>
      <td><input name="txtAddress" type="text" id="txtAddress"></td>
    </tr>
    <tr>
      <td>Phone No :</td>
      <td><input name="txtPhone" type="text" id="txtPhone"></td>
    </tr>
    <tr>
      <td>Email :</td>
      <td><input name="txtEmail" type="text" id="txtEmail"></td>
    </tr>
    <tr>

      <td>Comments :</td>
      <td><textarea name="txtComment" id="txtComment"></textarea></td>
    </tr>
    <tr>
      <td height="28" colspan="2"> <div align="center">
          <input type="submit" name="Submit" value="Submit">
          <input type="reset" name="Submit2" value="Reset">
        </div></td>
    </tr>
  </table>
  <div align="center"></div>
</form><p>
<a href="Default.asp">Home</a></p>
```

4. Select.asp

```
<%@LANGUAGE="VBSCRIPT"%>
<!--#include file="dbConnect.inc"-->
<%
    sql = "select * from Feedback"
    set rs = conn.execute(sql)
    Response.Write("<table border=1><tr><td><b>S. No</td><td><b>Name</td><td><b>Address</td><td><b>Phone</td><td><b>
Email</td><td><b>Comments</td></tr>")
    i=1
    do while not rs.eof
        d=rs("id")
        Response.Write("<tr><td>" & i & "</td><td><a href=edit.asp?id=" & d & ">" & rs("name") & "</a></td><td>" &
        rs("address") & "</td><td>" & rs ("phone") & "</td><td>" & rs("email") & "</td><td>" & rs ("comments") & "</td
</tr>")
        rs.movenext
        i=i+1
    Loop
    Response.Write("</table>")
%>
<p><a href="Default.asp">Home</a></p>
```

5. Insert.asp

```
<%@LANGUAGE="VBSCRIPT"%>
<!--#include file="dbConnect.inc"-->
<%
    dim var1,var2,var3,var4,var5
    var1 = request.Form("txtName")
    var2 = request.Form("txtAddress")
    var3 = request.Form("txtPhone")
    var4 = request.Form("txtEmail")
    var5 = request.Form("txtComment")
    sql = "insert into Feedback ( name, address, phone, email, comments) values ('" & var1 & "','" & var2 & "','"
    & var3 & "','" & var4 & "','" & var5 & "')"
    conn.execute(sql)
    Response.Redirect("Default.asp")
%>
```

6. Edit.asp

```
<%@LANGUAGE="VBSCRIPT"%>
<!--#include file="dbConnect.inc"-->
<%
        no=request.QueryString("id")
        sql = ("select * from Feedback where id="& no)
        set rs = conn.execute(sql)
        ID=rs("id")
      a =rs("name")
      b = rs("Address")
      c = rs("Phone")
      d = rs("Email")
      e = rs("Comments")
%>
<h2 align="center">Feedback Form</h2>
<form name="form" method="post" action="update.asp?id=<%=rs("id")%>">
```

```
  <table width="28%" border="1" align="center">
      <tr>
      <td width="35%">ID :</td>
      <td   width="65%"><input   name="txtID"   type="text"   readonly="true"
value="<%=ID%>"></td>
    </tr>
    <tr>
      <td width="35%">Name :</td>
      <td   width="65%"><input   name="txtName"   type="text"   id="txtName"
value="<%=a%>"></td>
    </tr>
    <tr>
      <td>Address :</td>
      <td><input      name="txtAddress"      type="text"      id="txtAddress"
value="<%=b%>"></td>
    </tr>
    <tr>
      <td>Phone No :</td>
      <td><input       name="txtPhone"       type="text"       id="txtPhone"
value="<%=c%>"></td>
    </tr>
    <tr>
      <td>Email :</td>
      <td><input       name="txtEmail"       type="text"       id="txtEmail"
value="<%=d%>"></td>
    </tr>
    <tr>
      <td>Comments :</td>
      <td><textarea        name="txtComment"        cols="18"        rows="4"
id="txtComment"><%=e%></textarea></td>
    </tr>
    <tr>
      <td height="28" colspan="2"> <div align="center">
          <input name="Update" type="submit" id="Update" value="Update">
          <input name="Delete" type="button" id="Delete" value="Delete">
          <a href="delete.asp?id=<%=rs("id")%>">Delete</a></div></td>
    </tr>
  </table>
  <div align="center"></div>
</form>
<p><a href="Default.asp">Home</a></p>
```

7. Update.asp

```
<%@LANGUAGE="VBSCRIPT"%>
<!--#include file="dbConnect.inc"-->
<%

    no=request.QueryString("id")

    a = request.Form("txtName")
    b = request.Form("txtAddress")
    c = request.Form("txtPhone")
    d = request.Form("txtEmail")
    e = request.Form("txtComment")
    sql = "update Feedback set name = '"& a & "', address = '"& b & "', phone = '"& c & "', email = '"& d & "'
comments = '"& e & "' where id="& no
    set rs = conn.execute(sql)
    Response.Write("Record Successfully Updated.<br><a href=Default.asp>Home</a>")
%>
```

8. Delete.asp

```
<%@LANGUAGE="VBSCRIPT"%>
<!--#include file="dbConnect.inc"-->
<%
    no=request.QueryString("id")
    sql="delete from feedback where id=" &no
    set rs=conn.execute(sql)
    response.Write("Record Deleted Successfully.<br><a href=Default.asp>Home</a>")
%>
```

And you came to the end…