**ASP**

Active Server Pages or Classic ASP. Server side technology introduced by Microsoft for creating dynamic web pages using scripting languages eg. vb script. ASP is a program that runs inside Internet Information Services (IIS) that runs on all versions of Windows.

The default scripting language used for writing ASP is VBScript, although other scripting languages like JScript (Microsoft's version of JavaScript) can use. An ASP file can contain text, HTML, XML, and scripts. Scripts in an ASP file are executed on the server.

Classic ASP pages have the extension **.asp** when a page with the extension .asp is requested by a browser the web server knows to interpret any ASP contained within the web page before sending the HTML produced to the browser. This way all the ASP is run on the web server and no ASP will ever be passed to the web browser.

The page must be requested through a web server that supports ASP, this is why ASP stands for Active Server Pages, no server, no active pages.

<!DOCTYPE html>  
<html>  
<body>  
<%response.write("My first ASP script!")%>  
</body>  
</html>

**How Does ASP Differ from HTML?**

* HTML is static web page. It has predefined tags
* It is used to display information in web page.
* 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

**What can ASP do for you?**

* 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 ASP code cannot be viewed from the browser
* Clever ASP programming can minimize the network traffic

**ASP.NET**

ASP.NET is server side web technology developed for web application to create dynamic web pages and web sites with HTML, CSS, JavaScript and server scripting using Fully Fledged programming languages supported by .NET. Web pages have extension .aspx

ASP.NET is a unified Web development model that includes the services necessary for you to build enterprise-class Web applications with a minimum of coding. ASP.NET is part of the .NET Framework, and when coding ASP.NET applications you have access to classes in the .NET Framework. You can code your applications in any language compatible with the common language runtime (CLR), including Microsoft Visual Basic, C#, JScript .NET, and J#. These languages enable you to develop ASP.NET applications that benefit from the common language runtime, type safety, inheritance, and so on.

ASP.NET includes:

* A page and controls framework
* The ASP.NET compiler
* Security infrastructure
* State-management facilities
* Application configuration
* Health monitoring and performance features
* Debugging support
* An XML Web services framework
* Extensible hosting environment and application life cycle management
* An extensible designer environment

**ASP.NET Page Life-Cycle Events**

**PreInit**

Raised after the start stage is complete and before the initialization stage begins.

Use this event for the following:

• Check the IsPostBack property to determine whether this is the first time the page is being processed. The IsCallback and IsCrossPagePostBack properties have also been set at this time.

• Create or re-create dynamic controls.

• Set a master page and Theme dynamically.

• Set initialization.

If the request is a postback, the values of the controls have not yet been restored from view state. If you set a control property at this stage, its value might be overwritten in the next event.

**Init**

Raised after all controls have been initialized and any skin settings have been applied. The Init event of individual controls occurs before the Init event of the page. Use this event to read or initialize control properties.

**PreLoad.**

Raised after the page loads view state for itself and all controls, and after it processes postback data that is included with the Request instance.

**Load**

The Page object calls the OnLoad method on the Page object, and then recursively does the same for each child control until the page and all controls are loaded. The Load event of individual controls occurs after the Load event of the page.

Use the OnLoad event method to set properties in controls and to establish database connections.

**PreRender**

Raised after the Page object has created all controls that are required in order to render the page, including child controls of composite controls. The Page object raises the PreRender event on the Page object, and then recursively does the same for each child control. The PreRender event of individual controls occurs after the PreRender event of the page.

Use the event to make final changes to the contents of the page or its controls before the rendering stage begins.

**Render**

It’s now time to send the output to the browser. If you would like to make some changes to the final HTML which is going out to the browser, you can enter your HTML logic here.

This is not an event; instead, at this stage of processing, the Page object calls this method on each control. All ASP.NET Web server controls have a Render method that writes out the control's markup to send to the browser.

If you create a custom control, you typically override this method to output the control's markup. However, if your custom control incorporates only standard ASP.NET Web server controls and no custom markup, you do not need to override the Render method.

**Unload**

Raised for each control and then for the page. Page object is unloaded from the memory.

In controls, use this event to do final cleanup for specific controls, such as closing control-specific database connections.

For the page itself, use this event to do final cleanup work, such as closing open files and database connections, or finishing up logging or other request-specific tasks.

**ASP.NET - Web Server Controls**

Web server controls are special ASP.NET tags understood by the server.

Web server controls are also created on the server and they require a runat="server" attribute to work.

The syntax for creating a Web server control is:

<asp:control\_name id="control\_id" runat="server" />

Example

<asp:Button ID="Button1" runat="server" Text="Submit" />

**Introduction to ADO.NET**

ADO.NET is an object-oriented set of libraries that allows us to interact with data sources. Commonly, the data source is a database, but it could also be a text file, an Excel spreadsheet, or an XML file

**Data Providers**

ADO.NET allows us to interact with different types of data sources and different types of databases. Since different data sources expose different protocols, we need a way to communicate with the right data source using the right protocol .

ADO.NET provides a relatively common way to interact with data sources, but comes in different sets of libraries for each way you can talk to a data source. These libraries are called Data Providers and are usually named for the protocol or data source type they allow you to interact with.

###### Table 1. ADO.NET Data Providers are class libraries that allow a common way to interact with specific data sources or protocols. The library APIs have prefixes that indicate which provider they support.

|  |  |  |
| --- | --- | --- |
| **Provider Name** | **API prefix** | **Data Source Description** |
| ODBC Data Provider | Odbc | Data Sources with an ODBC interface. Normally older data bases. |
| OleDb Data Provider | OleDb | Data Sources that expose an OleDb interface, i.e. Access or Excel. |
| Oracle Data Provider | Oracle | For Oracle Databases. |
| SQL Data Provider | Sql | For interacting with Microsoft SQL Server. |
|  |  |  |

### ADO.NET Objects

##### **The SqlConnection Object**

The SqlConnection object creates a link (or connection) to a specified data source. This object must contain the necessary information to discover the specified data source and to log in to it properly using a defined username and password combination and other parameters.

**Windows Authentication**

SqlConnection conn = new SqlConnection("Data Source=server\_name;Initial Catalog=database\_name;Integrated Security=True");

SqlConnection conn = new SqlConnection("Server=SqlServer\_Name;Database=Database\_Name;Integrated Security=SSPI");

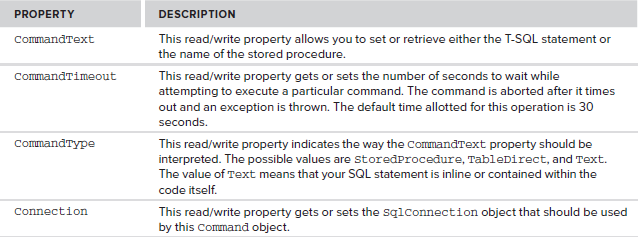
SSPI: Security Support Provider Interface

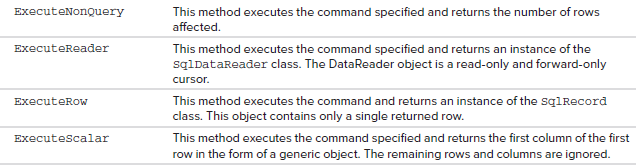
**Sql Authentication**

SqlConnection conn = new SqlConnection ("server=SqlServer\_Name;database=Database\_Name;uid=User;password=Password;");

##### **The SqlCommand Object**

The SqlCommand object uses the Connection object to execute SQL queries. These queries can be in the form of inline text, stored procedures, or direct table access. If the SQL query uses a SELECT clause, the result set it returns is usually stored in either a DataSet or a DataReader object. The Command object provides a number of **Execute** methods that you can use to perform various types of SQL queries.





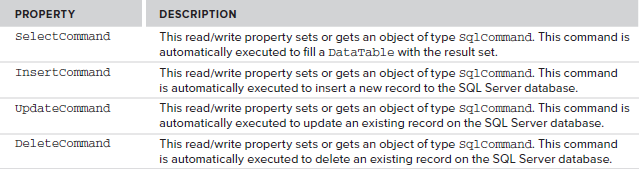
##### **The SqlDataReader Object**

The SqlDataReader object is a simple forward-only and read-only cursor. It requires a live connection with the data source and provides a very efficient way of looping and consuming all or part of the result set. This object cannot be directly instantiated. Instead, you must call the ExecuteReader method of the Command object to obtain a valid DataReader object. When using a DataReader object, be sure to close the connection when you are done using the data reader. If not, then the connection stays alive.

##### **The SqlDataAdapter Object**

The SqlDataAdapter is a special class whose purpose is to bridge the gap between the disconnected DataTable objects and the physical data source. The SqlDataAdapter provides a two-way data transfer mechanism. It is capable of executing a SELECT statement on a data source and transferring the result set into a DataTable object. It is also capable of executing the standard INSERT, UPDATE, and DELETE statements and extracting the input data from a DataTable object.

The commonly used properties offered by the SqlDataAdapter class.



The SqlDataAdapter class also provides a method called Fill(). Calling the Fill() method automatically executes the command provided by the SelectCommand property, receives the result set, and copies it to a DataTable object.

**DataSet**

DataSet objects are in-memory representations of data. They contain multiple Datatable objects. **The DataSet is specifically designed to help manage data in memory** and to support disconnected operations on data

**DataTable**

The DataTable object represents a logical table in memory. It contains rows(DataRow), columns(DataColumn), primary keys, constraints, and relations with other DataTable objects.

using System.Data.SqlClient;

using System.Data;

/// <summary>

/// Summary description for DatabaseConnection

/// </summary>

public class DBConnection

{

SqlConnection conn;

public DatabaseConnection()

{

string strConn="Data Source=BINOD-PC;Initial Catalog=COSMOS;Integrated Security=True;";

conn = new SqlConnection(strConn);

}

public int ExecuteNonQuery(string sql)

{

SqlCommand cmd = new SqlCommand();

try

{

cmd.CommandText = sql;

cmd.CommandType = CommandType.Text;

cmd.Connection = conn;

int result = cmd.ExecuteNonQuery();

return result;

}

catch (Exception ex)

{

throw ex;

} cmd.Connection.Open();

finally

{

if (cmd.Connection.State != ConnectionState.Closed)

{

cmd.Connection.Close();

}

}

}

public DataTable ExecuteDataTable(string sql) //This is method to select operation

{

DataSet ds = new DataSet();

try

{

SqlDataAdapter SqlAdapter = new SqlDataAdapter(sql, conn);

SqlAdapter.Fill(ds);

if (ds.Tables.Count > 0)

return ds.Tables[0];

else

return null;

}

catch (Exception ex)

{

throw ex;

}

}

}

**2.How to use above Database Connection Class**

Create **test.aspx**

Write this code on **button click** event of **test.aspx.cs** to insert data

protected void btnSave\_Click(object sender, EventArgs e)

{

string insertQuery="insert into student values('"+ txtFirstName.Text +"','"+ txtLastName.Text +"')";

DBConnection objDB = new DBConnection(insertQuery);

objDB.ExecuteNonQuery();

}

Write this code **Page\_Load** event of **test.aspx.cs** to display data in **GridView**

protected void Page\_Load(object sender, EventArgs e)

{

DataTable objDT = new DataTable();

string selectQuery="select \* from student";

DBConnection objDB = new DBConnection(selectQuery);

objDT=objDB.ExecuteDataTable();

GridView1.DataSource = objDT;

GridView1.DataBind()

}



When click on this button display student list.

The below code is to connect database and execute select statement to display students.

protected void DisplayStudent\_Click(object sender, EventArgs e)

{

SqlConnection conn = new SqlConnection("Data Source=.\\sqlexpress;Initial Catalog=Sep2015\_IT;Integrated Security=True");

DataSet ds = new DataSet();

DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter("select \* from student", conn);

da.Fill(ds);

if (ds.Tables.Count > 0)

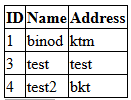
dt= ds.Tables[0];

GridView1.DataSource = dt;

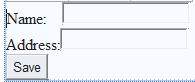
GridView1.DataBind();

}

Result:



The Form to save student information.



When click on student button then save data into student table. The below code is to connect database and execute insert statement.

protected void Save\_Click(object sender, EventArgs e)

{

SqlConnection conn = new SqlConnection("Data Source=.\\sqlexpress;Initial Catalog=Sep2015\_IT;Integrated Security=True");

SqlCommand cmd = new SqlCommand("insert into student values(' " + txtName.Text + "','" + txtAddress.Text + "')", conn);

cmd.Connection.Open();

cmd.ExecuteNonQuery();

cmd.Connection.Close();

}