**Unit -4**

**Working with ADO.NET, Overview of Data Objects**

**ADO.NET**

ADO.NET is a set of classes (a framework) to interact with data sources such as databases and XML files. ADO is the acronym for **ActiveX Data Objects**. It allows us to connect to underlying data or databases. It has classes and methods to retrieve and manipulate data.

The following are a few of the .NET applications that use ADO.NET to connect to a database, execute commands and retrieve data from the database.

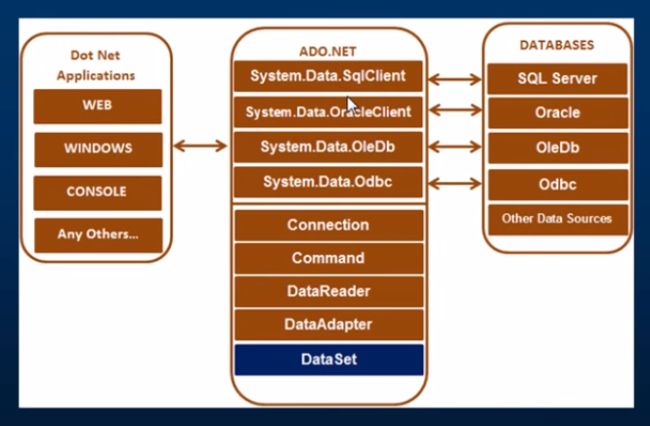
* ASP.NET Web Applications
* Console Applications
* Windows Applications.

Various Connection Architectures

There are the following two types of connection architectures:

1. **Connected architecture:** the application remains connected with the database throughout the processing.
2. **Disconnected architecture:** the application automatically connects/disconnects during the processing. The application uses temporary data on the application side called a DataSet.

**Understanding ADO.NET and it's class library**

****

In this diagram, we can see that there are various types of applications (Web Application, Console Application, Windows Application and so on) that use ADO.NET to connect to databases (SQL Server, Oracle, OleDb, ODBC, XML files and so on).

## Important Classes / Objects in ADO.NET

We can also observe various classes in the preceding diagram. They are:

1. Connection Class
2. Command Class
3. DataReader Class
4. DataAdaptor Class
5. DataSet.Class

### 1. Connection Class

In ADO.NET, we use these connection classes to connect to the database. These connection classes also manage transactions and connection pooling..

### 2. Command Class

The Command class provides methods for storing and executing SQL statements and Stored Procedures. The following are the various commands that are executed by the Command Class.

* **ExecuteReader:** Returns data to the client as rows. This would typically be an SQL select statement or a Stored Procedure that contains one or more select statements. This method returns a DataReader object that can be used to fill a DataTable object or used directly for printing reports and so forth.
* **ExecuteNonQuery:** Executes a command that changes the data in the database, such as an update, delete, or insert statement, or a Stored Procedure that contains one or more of these statements. This method returns an integer that is the number of rows affected by the query.
* **ExecuteScalar:**This method only returns a single value. This kind of query returns a count of rows or a calculated value.
* **ExecuteXMLReader:** (SqlClient classes only) Obtains data from an SQL Server 2000 database using an XML stream. Returns an XML Reader object.

### 3. DataReader Class

The DataReader is used to retrieve data. It is used in conjunction with the Command class to execute an SQL Select statement and then access the returned rows.

### 4. DataAdapter Class

The DataAdapter is used to connect DataSets to databases. The DataAdapter is most useful when using data-bound controls in Windows Forms, but it can also be used to provide an easy way to manage the connection between your application and the underlying database tables, views and Stored Procedures.

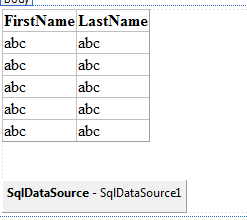
**5. DataSet Class**

The DataSet is the heart of ADO.NET. The DataSet is essentially a collection of DataTable objects. In turn each object contains a collection of DataColumn and DataRow objects. The DataSet also contains a Relations collection that can be used to define relations among Data Table Objects.

**Understanding Data Source Controls : SqlDataSource control**

The SqlDataSource data source control **represents data in an SQL relational database to data-**bound controls. You can use the SqlDataSource control in conjunction with a data-bound control to retrieve data from a relational database and to display, edit, and sort data on a Web page with little or no code.

Example:



In above example , we have taken one GridView Control and one SqlDataSource control.

Source Code of this is following:

<div>

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"

DataSourceID="SqlDataSource1">

<Columns>

<asp:BoundField DataField="FirstName" HeaderText="FirstName"

SortExpression="FirstName" />

<asp:BoundField DataField="LastName" HeaderText="LastName"

SortExpression="LastName" />

</Columns>

</asp:GridView>

<br />

<br />

<asp:SqlDataSource ID="SqlDataSource1" runat="server"

ConnectionString="<%$ ConnectionStrings:ConnectionString %>"

SelectCommand="SELECT \* FROM [Student]"></asp:SqlDataSource>

</div>

We need to only configure the SqlDataSource with database. And provide data source to Gridview.. Steps are following:

1. Click on right Arrow of SqlDataSource Control and click on configure data source. It will open configure data source window.
2. Choose connection and click on next.
3. Select Specify columns from a table. In which you need to select appropriate table and provide condition if any and click on next.
4. Click on finish.
5. Configure GridView. Click on right side Arrow of GridView and select data source. And run your web form.

**Working with DataBound Controls:**

**Gridview:**

The GridView control displays the values of a data source in a table. Each column represents a field, while each row represents a record. The GridView control supports the following features:

* Binding to data source controls, such as SqlDataSource.
* Built-in sort capabilities.
* Built-in update and delete capabilities.
* Built-in paging capabilities.
* Built-in row selection capabilities.
* Programmatic access to the GridView object model to dynamically set properties, handle events, and so on.
* Multiple key fields.
* Multiple data fields for the hyperlink columns.
* Customizable appearance through themes and styles.

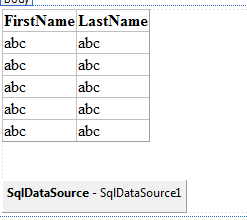
Creating a GridView

1. <asp:GridView ID="gridService" runat="server">
2. </asp:GridView>

we perform the following operations on GridView.

* Bind data to GridView column
* Edit data in GridView
* Delete rows from GridView
* Update row from database

**Example:**



In above example , we have taken one GridView Control and one SqlDataSource control.

Source Code of this is following:

<div>

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"

DataSourceID="SqlDataSource1">

<Columns>

<asp:BoundField DataField="FirstName" HeaderText="FirstName"

SortExpression="FirstName" />

<asp:BoundField DataField="LastName" HeaderText="LastName"

SortExpression="LastName" />

</Columns>

</asp:GridView>

<br />

<br />

<asp:SqlDataSource ID="SqlDataSource1" runat="server"

ConnectionString="<%$ ConnectionStrings:ConnectionString %>"

SelectCommand="SELECT \* FROM [Student]"></asp:SqlDataSource>

</div>

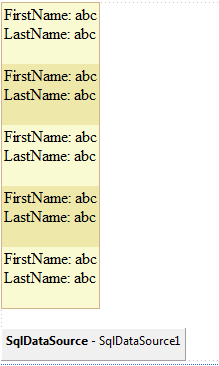
We need to only configure the SqlDataSource with database. And provide data source to Gridview.. Steps are following:

1. Click on right Arrow of SqlDataSource Control and click on configure data source. It will open configure data source window.
2. Choose connection and click on next.
3. Select Specify columns from a table. In which you need to select appropriate table and provide condition if any and click on next.
4. Click on finish.
5. Configure GridView. Click on right side Arrow of GridView and select data source. And run your web form.

**Data List:**

* DataList is an Unformatted Data Control like repeater control in ASP.NET.
* DataList Controls are used to display a list of Items.
* The DataList control is useful for displaying data in any repeating structure.
* Eval() and Bind() can be used to bind data in DataList.

Example:



We need to only configure SqlDataSource and DataList control.

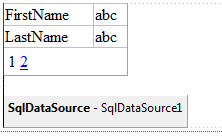
The following Templates can be used to display the Structure of DataList:

*<AlternatingItemTemplate>  
<edititemtemplate>  
<footertemplate>  
<headertemplate>  
<itemtemplate>  
<selecteditemtemplate>  
<separatortemplate>*

**Details View**

* Details View is a formatted data control like Grid View Control in ASP.NET.
* The Details View control displays only a single data record at a time, even if its data source exposes multiple records.
* CRUD operations are possible in DetailsView.
* **asp:BoundField** is used to bind data in DetailsView.

Example:



We need to only configure SqlDataSource and DetailsView control.

The following are the some important templates of DetailsView Control:

*<HeaderTemplate>  
<FooterTemplate>  
<PagerTemplate>  
<Fields>*

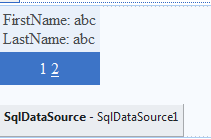
**Form View**

The working of FormView control is same as DetailView control but the default UI of FormView control is different.

DetailView control displays the records in tabular format. FormView control does not have predefined layout but it depend upon template.

According to the need of application, you can display the data with the help of template.

Example:



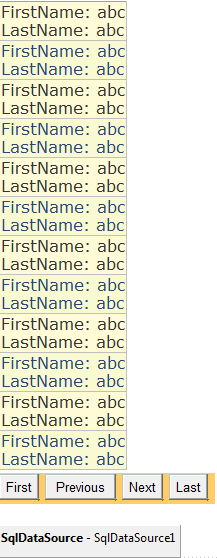
We need to only configure SqlDataSource and FormView control.  
  
You can insert, update, delete and paging the record in FormView.

**List View**

The List View control displays columns and rows of data and allows sorting and paging.

It is by far the most popular data display control, and is ideal for understanding how data display controls interact with data retrieval controls and code.

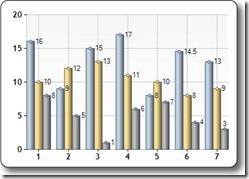
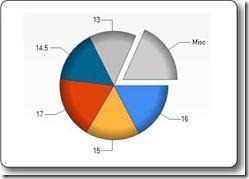
Example:



We need to only configure SqlDataSource and ListView control.

**Chart**

The "Chart Helper" can create chart images of different types with many formatting options and labels. It can create standard charts like area charts, bar charts, column charts, line charts, and pie charts, along with more specialized charts like stock charts.

The data you display in a chart can be from an array, from a database, or from data in a file.

In code behind add the following namespaces:

using System;

using System.Data;

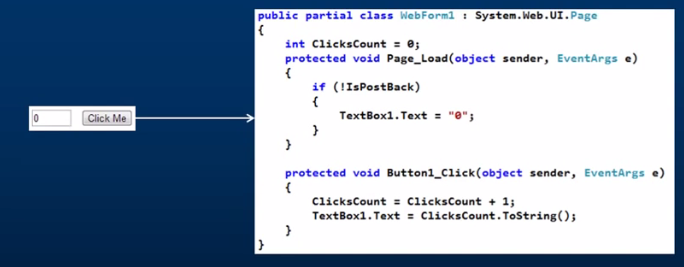
using System.Data.SqlClient;

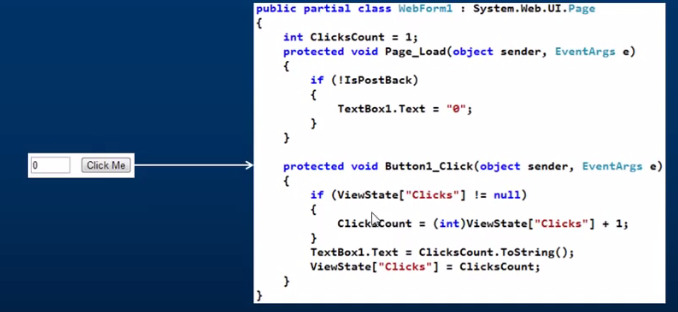
using System.Web.UI.DataVisualization.Charting;

**State Management:**

* View State:
* There are 5 types of state management techniques available with ASP.Net:
  + Query Strings
  + Cookies
  + **View State**
  + Session State
  + Application State
* View state are hidden fields when the form is posted to store state on behalf of a client.
* Each time a page is posted to itself, the contents of the \_\_VIEWSTATE field are sent as part of the post; which means viewState will help to pass value across multiple posts to the same page, but not across different pages.
* The primary use of view state is for controls to retain their state across post-backs, but it can also be used as a mechanism for storing generic client-specific state between post-backs to the same page.

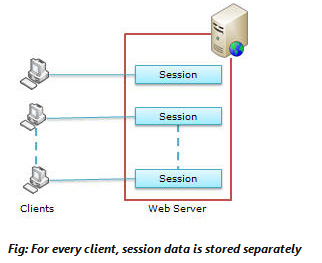
**Example:**





**What is Session?**

Web is **stateless**, which means a new instance of a web page class is re-created each time the page is posted to the server. As we all know, HTTP is a stateless protocol, it can't hold client information on a page. If the user inserts some information and move to the next page, that data will be lost and the user would not be able to retrieve that information. What do we need here? We need to store information. Session provides a facility to store information on server memory. It can support any type of object to store along with our own custom objects. For every client, session data is stored separately, which means session data is stored on a per client basis.



State management using session is one of the best ASP.NET features, because it is secure, transparent from users, and we can store any kind of object in it. Along with these advantages, some times session can cause performance issues in high traffic sites because it is stored in server memory and clients read data from the server.

#### **Advantages:**

* It helps maintain user state and data all over the application.
* It is easy to implement and we can store any kind of object.
* Stores client data separately.
* Session is secure and transparent from the user.

#### **Disadvantages:**

* Performance overhead in case of large volumes of data/user, because session data is stored in server memory.
* Overhead involved in serializing and de-serializing session data, because in the case of StateServer and SQLServer session modes, we need to serialize the objects before storing them.

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 variable:**

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()  
%>

**.aspx file:**

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Session.aspx.cs" Inherits="Session" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:Label ID="Label1" runat="server" Text="UserName"></asp:Label>

<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

<br />

<asp:Label ID="Label2" runat="server" Text="Password"></asp:Label>

<asp:TextBox ID="TextBox2" runat="server" TextMode="Password"></asp:TextBox>

<br />

<asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1\_Click" />

</div>

</form>

</body>

</html>

**C# file:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

public partial class Session : System.Web.UI.Page

{

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

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

SqlConnection cn = new SqlConnection(ConfigurationManager.ConnectionStrings["con"].ConnectionString);

SqlCommand cmd = new SqlCommand("select count(\*) from LoginData where username =@un and password = @pw",cn);

cmd.Parameters.AddWithValue("un", TextBox1.Text);

cmd.Parameters.AddWithValue("pw", TextBox2.Text);

cn.Open();

int i = (int)cmd.ExecuteScalar();

cn.Close();

if (i == 1)

{

Session["uname"] = TextBox1.Text;

Response.Redirect("welcome.aspx");

}

else

{

Response.Write("wrong username or password");

TextBox1.Text = "";

TextBox2.Text = "";

}

}

}

**What is Cookies?**

A **cookie** is a small bit of text that accompanies requests and pages as they go between the Web server and browser. The **cookie** contains information the Web application can read whenever the user visits the site.

Advantages :

* Cookie does not required any server because it is store in client.
* Cookie is easier to implement.
* We can configure the cookie when the browser session ends.

Disadvantages :

* User can delete cookie.
* It can store only string data types.
* In very rare case you can store cookie because it has not security.

**Aspx file:**

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="cookiedemo.aspx.cs" Inherits="cookiedemo" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

</head>

<body runat="server" id="BodyTag">

<form id="form1" runat="server">

<div>

<asp:DropDownList runat="server" id="ColorSelector" autopostback="true" OnSelectedIndexChanged="ColorSelector\_SelectedIndexChanged" >

<asp:ListItem value="White" selected="True">Select color...</asp:ListItem>

<asp:ListItem value="Red">Red</asp:ListItem>

<asp:ListItem value="Green">Green</asp:ListItem>

<asp:ListItem value="Blue">Blue</asp:ListItem>

</asp:DropDownList>

</div>

</form>

</body>

</html>

**C# file:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

public partial class cookie : System.Web.UI.Page

{

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

{

if (Request.Cookies["BackgroudColor"] != null)

{

BodyTag.Style["Backgroud-color"] = ColorSelector.SelectedValue;

}

}

protected void ColorSelector\_SelectedIndexChanged(object sender, EventArgs e)

{

BodyTag.Style["Backgroud-color"] = ColorSelector.SelectedValue;

HttpCookie cookie = new HttpCookie("BackgroudColor");

cookie.Value = ColorSelector.SelectedValue;

cookie.Expires = DateTime.Now.AddMinutes(1);

Response.SetCookie(cookie);

}

}

**Compare / Differentiate Session and Cookie**

|  |  |
| --- | --- |
| Session | Cookie |
| * It can store any data type. * It is store in cache / server side. * It is secured than cookie because it can store at server side. * It stores data in binary format. * For all situation we can use session. | * It can store only string data type. * It stores in client side only. * It is less secured than session because it store at client side only. * It stores data in text format. * For some specific situation we use cookie. |

**What is connection class in ado.net explain with example**

“Most basic process of ADO.NET that use in almost every process and that always check our database (.mdf) file is connected with our web page or not.”

* It has main two functions :
  + What type of database we are going to connect .
  + The path/locatoin of database file (.mdf)

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| **.open ()** | **It will open the database connection** |
| **.close ()** | **It will close the database connection.** |
| **.dispose ()** | **It will free primary memory if multiple connection opens** |
| **.state ()** | **It will check the connection with various stage like is it open / close / continue** |

In asp.net connection set in web.config file that has below code and it will use in normal web page ( ex. default.aspx)

Source code for web.config file :

**<configuration>**

**<conncetionStrings>**

**<add name=”con” connectionString=”<paste connection string from property>” />**

**</connectionString>**

**<system.web>**

**<connection debug =”true” …..>**

**</system.web>**

**</configuration>**

**What is Command class in ado.net explain with example**

“Main class of ADO.NET that perform all major process like Insert – Update – Delete and Select from Front end to back end and back end to front end is known as command class”

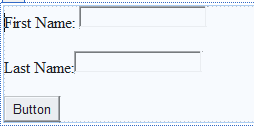
It has main two functions :

* + It perform Insert – Update – Delete with connection only
  + It perform Select query with connection + Data reader

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| **.executeQuery()** | **To perform Insert – Update – Delete** |
| **.executeReader ()** | **To perform single or group record search with data reader** |
| **.executeScaler ()** | **To perform Single primary key base record** |

**Example of Insert record from web page to database**

**Example of Connection string + command class**

****

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

public partial class welcome : System.Web.UI.Page

{

protected void Page\_Load(…………………………)

{

}

protected void Button1\_Click(………………)

{

SqlConnection cn = new SqlConnection(ConfigurationManager.ConnectionStrings["con"].ConnectionString);

SqlCommand cmd = new SqlCommand("insert into Student values(@FirstName, @LastName)",cn);

cmd.Parameters.AddWithValue("@FirstName", TextBox1.Text);

cmd.Parameters.AddWithValue("@LastName", TextBox2.Text);

cn.Open ();

cmd.ExecuteNonQuery ();

cn.Close();

}

}

**Explain Data Adapter of ADO.NET with suitable example**

“A class that perform back to front process with select query is known as data adapter “

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| **.fill ()** | **It transfer select query records to dataset** |

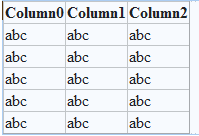
**Explain Data set class of ADO.NET with suitable example**

“A class that works like bridge between data adapter and gridview and provide select query records to front-end web page is known as dataset “

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| **.tables()** | **It is collection of records that can connect to grid view directly** |

**Load all the records from student table to datagridview in asp.net web page.**

**Example of Connection string + data adapter + dataset**

****

**In this example we can display data from database in to GridView using following code.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

public partial class GridView : System.Web.UI.Page

{

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

{

SqlConnection cn = new SqlConnection(ConfigurationManager.ConnectionStrings["con"].ConnectionString);

SqlDataAdapter adp = new SqlDataAdapter("select \* from Student", cn);

DataSet ds = new DataSet();

adp.Fill(ds);

GridView1.DataSource = ds;

GridView1.DataBind();

}

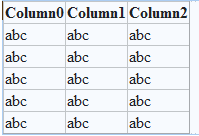
}

**Explain SqlDataReader class of ADO.NET with suitable example**

“A class of ADO.NET that always perform logical perform or sometimes execute with command class to perform select query for back to font process is known as Datareader”

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| **.read ()** | **To perform single record found or not ( ex. login process user found or not )** |

**DataReader:**

****

**Load all the records from student table to datagridview in asp.net web page.**

**Example of Connection string + command class + datareader + GridView**

**We can read data from database and display in GridView using following code.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

public partial class Conn\_usingDataReader\_GridView : System.Web.UI.Page

{

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

{

SqlConnection cn = new SqlConnection(ConfigurationManager.ConnectionStrings["con"].ConnectionString);

SqlCommand cmd = new SqlCommand("select \* from Student", cn);

cn.Open();

SqlDataReader rdr = cmd.ExecuteReader();

GridView1.DataSource = rdr;

GridView1.DataBind();

cn.Close();

}

}

**Differenciate:**

|  |  |
| --- | --- |
| **Grid View** | **DetailsView** |
| GridView is intended to primarily to display/update multiple records. | DetailsView is intended for the displaying of a single record. |
| In GridView paging and navigation is not optional. | In DetailsView optional support for paging and navigation. |
| GirdView shows all record like HTML table. | DetailsView as its designed for master detail relationships. |
| Displays the values of a data source in a table where each column represents a field and each row represents a record. | Displays the values of a single record from a data source in a table, where each data row represents a field of the record. |
| GridView control enables you to sort and edit records. | DetailsView control allows you to edit, delete and insert records. |
| Grouping is not allow in GridView. | In detailsview grouping is allow. |
| If you want more customizable flow layout, GridView is not best option. | In detailview we can customize flaw layout easily. |
| If default paging is used, GridView will load complete dataset in memory, which decreases website performance. | In detailview single record is display at a time. So it will not create problem of performance. |

|  |  |
| --- | --- |
| **GridView** | **FormView** |
| GridView is intended to primarily to display/update multiple records | Fromview is used to insert/update the single record. |
| In GridView complete control over creating is gridview while insert/update/delete | It will gives complete control over creating form while inserting/updating a record |
| GridView ables to validate against a data source schema and does supply foreign key filed. | FormView control is unable to validate against a data source schema and doesn’t supply advanced editing features like foreign key field. |
| GridView has default rendering as HTML table tag. | FormView has not default rendering of its own. |
| The Gridview control enables you to sort records. | FormView does not allow to sort records. |

|  |  |
| --- | --- |
| Master Web Page | Normal Web Page |
| * Extension of this page is .master. * Main purpose is to design the website. * It never execute / render. * It has by default content place holder as server side control. * For any website Master page are one – two or very few. | * Extension of this page is .aspx * Main purpose is to perform transactions of website. * It always execute / render. * It does not has content place holder as server side control * For any website Normal pages are maximum as per requirement. |

|  |  |
| --- | --- |
| Connected Architecture | Disconnected Architecture |
| * Where connection declared with compulsory use of connection method like .open () and .close () * It use with **command class and datareader** class OF ADO.NET | * Where connection declare with **not necessary** use of connection method like .open () and .close () * It use with **dataadapter and dataset** class of ADO.NET |