

## Introduction of ASP.NET

ASP.NET is the next generation ASP, but it's not an upgraded version of ASP. ASP.NET is an entirely new technology for server-side scripting. ASP.NET is a part of the Microsoft .NET framework, and a powerful tool for creating dynamic and interactive web pages. ASP.NET is a server side scripting technology that enables scripts (embedded in web pages) to be executed by an Internet server

- ASP.NET is a Microsoft Technology.
- ASP stands for Active Server Pages
- ASP.NET is a program that runs inside IIS.
- IIS (Internet Information Services) is Microsoft's Internet server.
- IIS comes as a free component with Windows servers.
- IIS is also a part of Windows 2000 and XP Professional.

## ASP.NET File

- An ASP.NET file is just the same as an HTML file
- An ASP.NET file can contain HTML, XML, and scripts
- Scripts in an ASP.NET file are executed on the server
- An ASP.NET file has the file extension ".aspx"

## Working of ASP.NET

- When a browser requests an HTML file, the server returns the file.
- When a browser requests an ASP.NET file, IIS passes the request to the ASP.NET engine on the server.
- The ASP.NET engine reads the file, line by line, and executes the scripts in the file.
- Finally, the ASP.NET file is returned to the browser as plain HTML.

## Difference Between ASP and ASP.NET

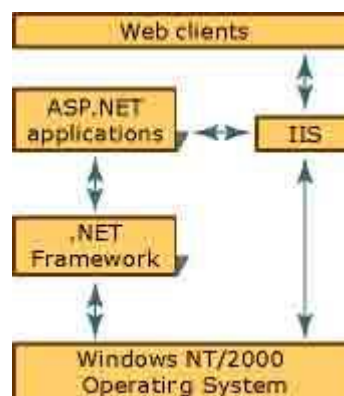
1. Code behind in asp. net allows separation of business logic from UI which is not possible in asp.
2. ASP. Net uses ADO .Net objects which are represented using XML and hence they are lightweight and can travel through firewalls. ASP uses ADO record sets which are binary COM objects heavier than ADO.Net counterparts and cannot travel through firewalls.
3. ASP.Net supports object oriented programming. ASP is procedural.
4. ASP.Net is compiled and hence performs better. ASP is pure scripting and hence interpreted at the time of page load.
5. ASP.Net has caching and exception handling which is not available with ASP.
6. ASP.NET has better language support, a large set of new controls, XML-based components, and better user authentication.
7. ASP.NET provides increased performance by running compiled code.
8. ASP.NET code is not fully backward compatible with ASP

## Advantage of ASP.NET

1. ASP.NET drastically reduces the amount of code required to build large applications.
2. With built-in Windows authentication and per-application configuration, your applications are safe and secured.

3. It provides better performance by taking advantage of early binding, just-in-time compilation, native optimization, and caching services right out of the box.
4. The ASP.NET framework is complemented by a rich toolbox and designer in the Visual Studio integrated development environment. WYSIWYG editing, drag-and-drop server controls, and automatic deployment are just a few of the features this powerful tool provides.
5. Provides simplicity as ASP.NET makes it easy to perform common tasks, from simple form submission and client authentication to deployment and site configuration.
6. The source code and HTML are together therefore ASP.NET pages are easy to maintain and write. Also the source code is executed on the server. This provides a lot of power and flexibility to the web pages.
7. All the processes are closely monitored and managed by the ASP.NET runtime, so that if process is dead, a new process can be created in its place, which helps keep your application constantly available to handle requests.
8. It is purely server-side technology so, ASP.NET code executes on the server before it is sent to the browser.
9. Being language-independent, it allows you to choose the language that best applies to your application or partition your application across many languages.
10. ASP.NET makes for easy deployment. There is no need to register components because the configuration information is built-in.
11. The Web server continuously monitors the pages, components and applications running on it. If it notices any memory leaks, infinite loops, other illegal activities, it immediately destroys those activities and restarts itself.
12. Easily works with ADO.NET using data-binding and page formatting features. It is an application which runs faster and counters large volumes of users without having performance problems.

## Architecture of ASP.NET



- The configuration of ASP.NET is managed by information stored in XML-format in a configuration file (Web.Config).
- The cache allows for improved performance of ASP.NET, as the most commonly requested pages would be served from the ASP.NET cache.
- State management services for ASP.NET are provided by the ASP.NET state service.
- The .NET Framework provides the Common Language Runtime (CLR) , which compiles and manages the execution of ASP.NET code, and the class libraries, which offer prebuilt programmatic functionality for Web Forms, XML support, and exception handling.
- ADO.NET provides ASP.NET with connections to databases.

## 3-Tier Architecture in ASP.NET with C#

3-Tier architecture generally contains UI or Presentation Layer, Business Access Layer (BAL) or Business Logic Layer and Data Access Layer (DAL).

### Presentation Layer (UI)

Presentation layer contains pages like .aspx or windows form where data is presented to the user or input is taken from the user.

### Business Access Layer (BAL) or Business Logic Layer

BAL contains business logic, validations or calculations related with the data..

### Data Access Layer (DAL)

DAL contains methods that helps business layer to connect the data and perform required action, might be returning data or manipulating

data (insert, update, delete etc).

## Designing 3-Tier Architecture



## Data Access Layer (DAL) Code

*Code for Data Access Layer*

```

using System;

using System.Data;

using System.Configuration;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Web.UI.HtmlControls;

using System.Data.OleDb;

public class PersonDAL
{
    string connStr =
ConfigurationManager.ConnectionStrings["connectionstring"].ToString();
    public PersonDAL()
    {
    }

    public int Insert(string name, string address, int age)
    {
        OleDbConnection conn = new OleDbConnection(connStr);

        conn.Open();

        OleDbCommand dCmd = new OleDbCommand("insert into insertrecord
values(@P,@q,@r)",conn);
        try
        {
            dCmd.Parameters.AddWithValue("@p", name);
            dCmd.Parameters.AddWithValue("@q", address);
            dCmd.Parameters.AddWithValue("@r", age);
            return dCmd.ExecuteNonQuery();

        }

        catch
        {
            throw;

        }

        finally
        {
    }
}

```

```
dCmd.Dispose();  
  
conn.Close();  
  
conn.Dispose();  
  
}  
  
}  
}
```

## Business Access Layer (BAL) Code

*Code for Business Access Layer*

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public class PersonBAL
{
    public PersonBAL()
    {
    }
    public int Insert(string name, string address, int age)
    {
        PersonDAL pDAL = new PersonDAL();

        try
        {
            return pDAL.Insert(name, address, age);
        }

        catch
        {
            throw;
        }

        finally
        {
            pDAL = null;
        }
    }
}
```

## Code for .cs file

```
using System;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        //Lets validate the page first

        if (!Page.IsValid)

            return;

        int intResult = 0;

        PersonBAL pBAL = new PersonBAL();

        // Instantiate the object we have to deal with

        string name = txtname.Text;
        string address = txtaddress.Text;
        int age = Int32.Parse(txtAge.Text);
        try
        {
            intResult = pBAL.Insert(name, address, age);
            if (intResult > 0)

                lblMessage.Text = "New record inserted successfully.";

            else

                lblMessage.Text = "FirstName [<b>" + txtname.Text + "</b>]
                alredy exists, try another name";

        }
        catch (Exception ee)
        {

            lblMessage.Text = ee.Message.ToString();

        }

        finally
```

```

    {
        pBAL = null;
    }
}

```

## Presentation Layer

ADD RECORDS

Name

Address

Age

## Code for .aspx page

```

<%@ Page
Language="C#" AutoEventWireup="true" CodeVirtual="Default.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
    <style type="text/css">
        .style1
        {
            width: 100%;
            height: 215px;
            background-color: #FFFFCC;
        }
        .style2
        {
            width: 271px;
        }
        .style3
        {
            width: 271px;
            height: 44px;
        }
        .style4
        {
            height: 44px;
        }
    </style>
</head>
<body>
    <div class="style1">
        <div class="style2">
            <div class="style3">
                <div class="style4">
                    <input type="button" value="Save Record" />
                </div>
            </div>
        </div>
    </div>
</body>
</html>

```





- **Page\_Load:** The server controls are loaded in the page object. View state information is available at this point, so this is where you put code to change control settings or display text on the page.
- **Page\_PreRender:** The application is about to render the page object.
- **Page\_Unload:** The page is unloaded from memory.
- **Page\_Disposed:** The page object is released from memory. This is the last event in the life of a page object.
- **Page\_Error:** An unhandled exception occurs.
- **Page\_AbortTransaction:** A transaction is aborted.
- **Page\_CommitTransaction:** A transaction is accepted.
- **Page\_DataBinding:** A server control on the page binds to a data source.

## Control in ASP.NET

- HTML Controls - Traditional HTML tags
- Web Server Controls - New ASP.NET tags
- Validation Server Controls - For input validation

## HTML Server Controls

HTML server controls are HTML tags understood by the server.

HTML elements in ASP.NET files are, by default, treated as text. To make these elements programmable, add a `runat="server"` attribute to the HTML element. This attribute indicates that the element should be treated as a server control. The `id` attribute is added to identify the server control. The `id` reference can be used to manipulate the server control at run time.

**Note:** All HTML server controls must be within a `<form>` tag with the `runat="server"` attribute. The `runat="server"` attribute indicates that the form should be processed on the server. It also indicates that the enclosed controls can be accessed by server scripts.

## DESIGN OF HTML CONTROL



```
<script language="javascript"
type="text/javascript">

function Submit1_onclick() {
alert("click me");
}
</script>
</head>
<body>
  <form id="form1" runat="server">
    <div>
      <input id="Text1" type="text" value="anil" />
      <input id="Submit1" type="submit"
value="submit" onclick="return Submit1_onclick()"/>
    </div>
  </form>
</body>
```

```
/></div>
</form>
</body>
</html>
```

## Web Server Controls

- Web server controls are special ASP.NET tags understood by the server.
- Like HTML server controls, Web server controls are also created on the server and they require a `runat="server"` attribute to work.
- However, Web server controls do not necessarily map to any existing HTML elements and they may represent more complex elements.

### The syntax for creating a Web server control is:

```
<asp:control_name id="some_id" runat="server" />
```

## DESIGN OF WEB SERVER CONTROL



### .ASPX CODE

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Untitled Page</title>

</head>
<body>
  <form id="form1" runat="server">
    <div style="width: 246px; background-color: #66FFFF;">
      Name&nbsp;
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
      <br />

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

      <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
        Text="click me" />
      <br />
      <br />
    </div>
  </form>
</body>
```

```
</div>
</form>
</body>
</html>
```

## .CS CODE

```
protected void Button1_Click(object sender, EventArgs e)
{
    TextBox1.Text = "wiliam";
    TextBox2.Text = "hello world";
}
```

## Validation Server Controls

## DESIGN OF VALIDATION SERVER CONTROL



The screenshot shows a web form with two input fields: "First Name:" and "Last Name:". Below the "Last Name:" field is a "Submit" button. A Microsoft Internet Explorer error dialog box is displayed in the foreground. The dialog box has a yellow warning triangle icon and contains the following text:

- First Name is required
- Last Name is required

At the bottom of the dialog box is an "OK" button.

Validation server controls are used to validate user-input. If the user-input does not pass validation, it will display an error message to the user.

## ■ ASPX CODE

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Untitled Page</title>

</head>
<body>
  <form id="form1" runat="server">
    <div style="width: 414px; background-color: #66FFFF;">
      Name 
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    </div>
  </form>
</body>
</html>
```

```

        <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server"
        ControlToValidate="TextBox1" ErrorMessage="Blank are not
allowed"
        SetFocusOnError="True"></asp:RequiredFieldValidator>
        <br />
Address
        <asp:TextBox ID="TextBox2"
runat="server"
onTextChanged="TextBox2_TextChanged"></asp:TextBox>
        <br />
        <br />
        <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
Text="click me" />
        <br />
        <br />
    </div>
</form>
</body>
</html>

```

## .CS CODE

```

protected void Button1_Click(object sender, EventArgs e)
{
    TextBox1.Text = "123";
    TextBox2.Text = "hello world";
    Response.Write("hi display textbox 1 value"+TextBox1.Text);
}

```

## Label Control in ASP.NET

It is used to provide the static information on the webform.

### Common properties for all the server side controls

<b>Id</b>	Used to specify the unique identifier for the control placed on the webform..
<b>runat="Server"</b>	all control run on sever
<b>Text</b>	It is used to set or get the value from the server side control..

The id attribute is used to uniquely identify the <asp:label> control so you can refer to it in your ASP.NET code. The runat="server" attribute tells the server to process the control and generate HTML code to be sent to the client.

### The class hierarchy for this control is as follows

1. Object
2. Control
3. Web Control

#### 4. Label

##### Syntax of Label Control

```
<asp: Label ID="Label1" runat="server" Text="This is a label control">
```

```
</asp: Label>
```

##### Simple Example of Label Control

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = "Hi I AM Label control";
}
```

##### Click On Button

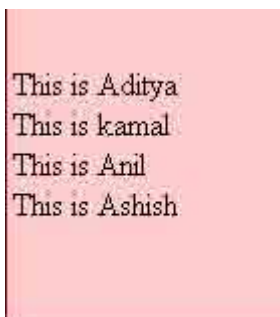


##### How To Write Line Break In ASP.NET Label Control

To write text in lines in standard ASP.NET label control you need to use HTML tag `<br />`. This is example label control that contains text in Four Lines lines:

```
<asp:Label ID="Label1" runat="server"
Text="This is Aditya<br />This is kamal<br /> This is Anil<br/> This is Ashish">
</asp:Label>
```

##### Example



## TextBox Control in ASP.NET

TextBox Control is used for user to enter the input like any string eg., Name, Password , number etc., Asp.NET TextBox itself is a class which is present under System.Web.UI.WebControls namespace.

It is used to accept SingleLine, MultiLine or Password characters as the input.

### Syntax Of Textbox

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

### Properties

Property	Description
AutoCompleteType	Specifies the AutoComplete behavior of a TextBox
AutoPostBack	A Boolean value that specifies whether the control is automatically posted back to the server when the contents change or not. Default is false
CausesValidation	Specifies if a page is validated when a Postback occurs
Columns	The width of the textbox
MaxLength	The maximum number of characters allowed in the textbox
ReadOnly	Specifies whether or not the text in the text box can be changed
Rows	The height of the textbox (only used if TextMode="Multiline")
Runat	Specifies that the control is a server control. Must be set to "server"
TagKey	
Text	The contents of the textbox
TextMode	Specifies the behavior mode of a TextBox control (SingleLine, MultiLine or Password)
ValidationGroup	The group of controls that is validated when a Postback occurs
Wrap	A Boolean value that indicates whether the contents of the textbox should wrap or not
OnTextChanged	The name of the function to be executed when the text in the textbox has changed

### Example

### .CS CODE

```
protected void Button1_Click(object sender, EventArgs e)
{
    ListBox1.Items.Add("Name:" + TextBox3.Text);
    ListBox1.Items.Add("father Name:" + TextBox4.Text);
    ListBox1.Items.Add("Course:" + TextBox5.Text);
    ListBox1.Items.Add("college:" + TextBox6.Text);
}
```

## Output



## Button Control in ASP.NET

Button control is generally used to post the form or fire an event either client side or server side. When it is rendered on the page, it is generally implemented through `<input type=submit>` HTML tag. However, if `UserSubmitBehavior` property is set to false then control will render out as `<input type=button>`.

Its properties like `BackColor`, `ForeColor`, `BorderColor`, `BorderStyle`, `BorderWidth`, `Height` etc. are implemented through style properties of `<input>` tag. You can set its `Text` property either by setting `Text` property in the .aspx page or from server side page. (other properties can also be set from both pages)

The Button control is used to display a push button. The push button may be a submit button or a command button. By default, this control is a submit button.

A submit button does not have a command name and it posts the page back to the server when it is clicked. It is possible to write an event handler to control the actions performed when the submit button is clicked.

A command button has a command name and allows you to create multiple Button controls on a page. It is possible to write an event handler to control the actions performed when the command button is clicked.

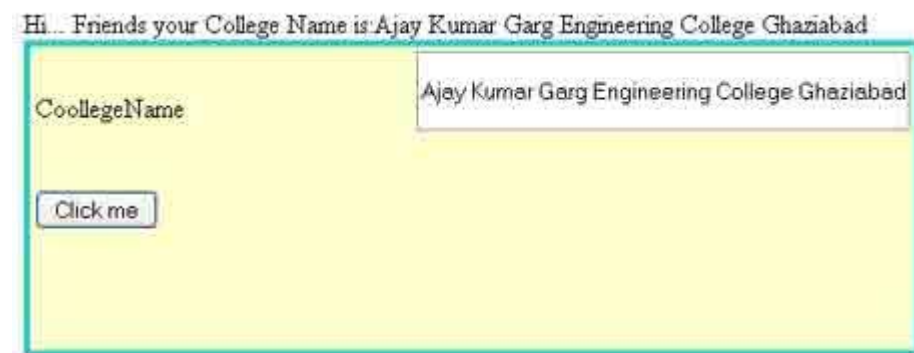
## Properties

Properties	Description
<code>CausesValidation</code>	Specifies if a page is validated when a button is clicked
<code>CommandArgument</code>	Specifies additional information about the command to perform



CommandName	Specifies the command associated with the Command event
OnClickClick	Specifies the name of the function to be executed when a button is clicked
PostBackUrl	Specifies the URL of the page to post to from the current page when a button is clicked
runat	Specifies that the control is a server control. Must be set to "server"
Text	Specifies the text on a button
UseSubmitBehavior	Specifies whether or not a button uses the browser's submit mechanism or the ASP.NET postback mechanism
ValidationGroup	Specifies the group of controls a button causes validation, when it posts back to the server

### Example



### .Cs Code

```
protected void Button1_Click(object sender, EventArgs e)
{
    Response.Write("Hi... Friends your College Name is:" + TextBox1.Text);
}
```

## Image Button in ASP.NET

ImageButton control is used to display an image that responds to mouse clicks. Specify the image to display in the control by setting the ImageUrl property.

Both the Click and Command events are raised when the ImageButton control is clicked.

By using the OnClick event handler, you can programmatically determine the coordinates where the image is clicked. You can then code a response based on the values of the coordinates. Note the origin (0, 0) is located at the upper-left corner of the image.

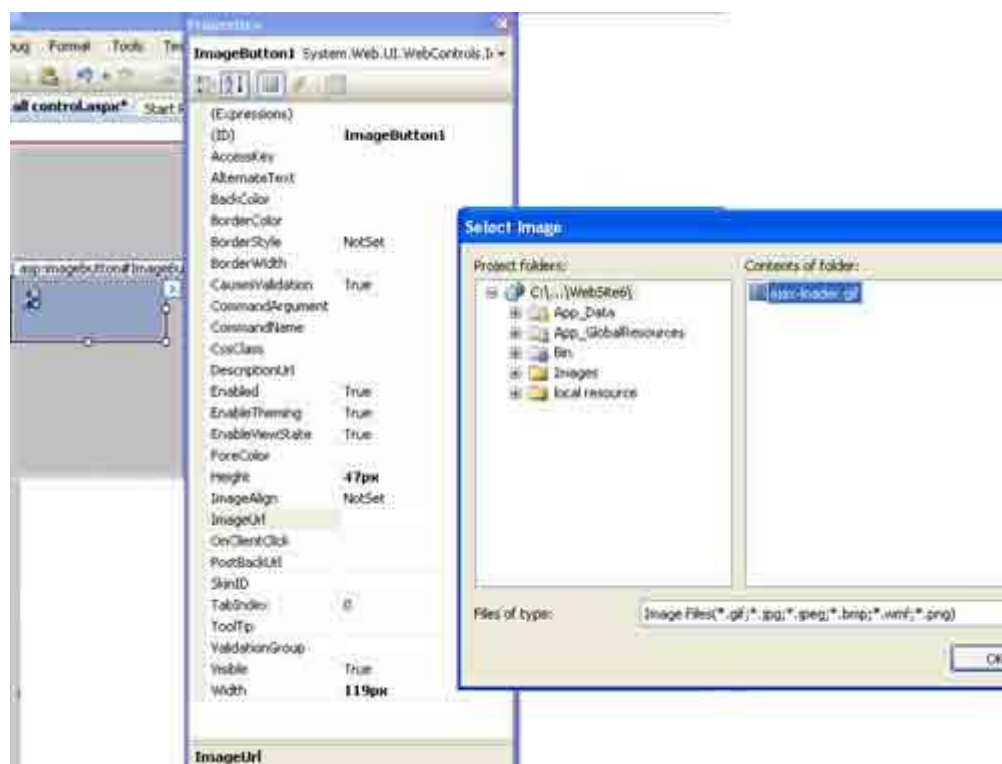
You can use the OnCommand event handler to make the ImageButton control behave like a command button. A command name can be associated with the control by using the CommandName property. This allows multiple ImageButton controls to be placed on the same Web page. The value of the CommandName property can then be programmatically identified in the OnCommand event handler to determine the appropriate action to perform when each ImageButton control is clicked. The

CommandArgument property can also be used to pass additional information about the command, such as specifying ascending order.

## Property

Property	Description
CausesValidation	Specifies if a page is validated when an ImageButton control is clicked
CommandArgument	Additional information about the command to perform
CommandName	The command associated with the Command event
GenerateEmptyAlternateText	Specifies whether or not the control creates an empty string as an alternate text
OnClickClientClick	The name of the function to be executed when the image is clicked
PostBackUrl	The URL of the page to post to from the current page when the ImageButton control is clicked
ValidationGroup	Specifies that the control is a server control. Must be set to "server"
runat	The group of controls for which the ImageButton control causes validation when it posts back to the server

## How to Set Image On Button



## Output



## Hyperlink control in ASP.NET

The Hyperlink Web server control is used to create a link to another Web page. The text in the HyperLink control is specified using the Text property. We can also display a image on this control instead of text.

**Properties:** These are the following properties of the Hyperlink control.

**Text** - The text to be shown for the link.

**ImageUrl** - gets\sets The image to be displayed in the hyperlink.

**NavigateUrl** - The URL to navigate to when the hyperlink is clicked.

**Target** - The target property allows us to set new page/content to be displayed in a new browser window, same window, etc. The values are as follows:

\_blank: displays the linked content in a new window without frames.

\_parent: displays the linked content in the immediate frameset parent.

\_self: displays the linked content in the frame with focus.

\_top: displays the linked content in the full window without frames.

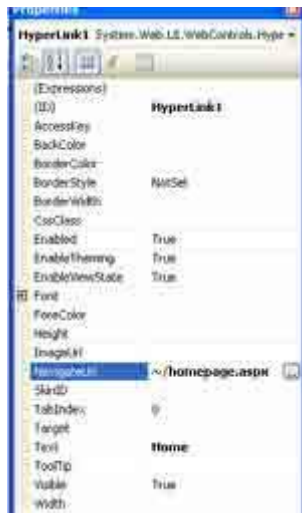
**ToolTip** - ToolTip displayed when the mouse is over the control.

## Example

### Design



### Right Click on Home



## Output

**When you click On Hyperlink it will show home Page of website**



## Literal Control in ASP.NET

- The Literal Control is similar to the Label Control as they both are used to display static text on a web page.
- The Literal Control is not inherited from WebControl namespace.
- The Literal Control doesn't provide substantial functionality but Literal text is programmable.
- It doesn't add any HTML elements to the web page. This control makes it possible to add HTML code directly in the code designer window without switching to design view and clicking the HTML button to edit the HTML.
- You cannot apply a style to a literal control.
- Unlike Label control, there is no property like BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, Height etc. for Literal control. That makes it more powerful, you can even put a pure HTML contents into it.

The Literal control is used to display text; that is, it renders static text on a Web page without adding additional HTML tags. It passes content directly to the client browser unless you use the Mode property to encode the content.

### Syntax of Literal Control

```
<asp:Literal ID="LiteralText" runat="server" Text="This is example of  
Literal"></asp:Literal>
```

**Important Properties of Asp.NET Webserver Literal control are given below.** You can Set /Get these properties at Design time or at Runtime.

**Mode** - Enumerator - To specify how to render literal control in a web page.

**Text** - Accept String - Commonly Used property to Get/Set the text that you want to render in your webpage

**There are three Mode enumerator available. These are**

1. **PassThrough** : If you set this property, then the content will not be modified and rendered as is. For eg., if string contains <hr> tag then its dependent on your browser, of how it handles <hr> tag.
2. **Encode** : If you set this property then content will be encoded and sent to browser for eg., if your string contains <hr> tag, then your string will be converted to &lt;Hr> and sent to browser.
3. **Transform** : If you set Mode property to Transform then the string render depends upon the type of the markup.

**Events available in Asp.Net Web server Literal Control are:**

Click  
Init  
Load  
PreRender  
UnLoad

**The difference between a Label Control and a Literal Control?**

1. The main difference is, you can apply style to a Label control where as you can not apply styles in a literal control.
2. Label control renders as span tag in a webpage while Literal Control only shows the text without any tag associated to it.
3. Literal control does not maintain its viewstate.
4. It's always a good practice to use literal control when you want to display a plain text in your web page. ASP.NET Literal itself is a class which is present under [System.Web.UI](#) namespace.

### ASP.NET LinkButton Control

Link Button Control is similar to Button Control. Everything is similar in properties and events, except that user can see it as a Link.

## Syntax Of Link Button

```
<asp:LinkButton ID="LinkButton1" runat="server" >Home</asp:LinkButton>
```

## Important Properties

**AccessKey:** specify a key that navigates to the Button control.

**CommandArgument:** specify a command argument that is passed to the Command event.

**CommandName:** specify a command name that is passed to the Command event.

**Enabled:** disable the LinkButton control.

**OnClientClick:** specify a client-side script that executes when the LinkButton is clicked.

**PostBackUrl:** post a form to a particular page.

**TabIndex:** specify the tab order of the LinkButton control.

**Text:** label the LinkButton control.

**Focus:** set the initial form focus to the LinkButton control.

## LinkButton Event

Event	Description
Click	occurs when the LinkButton control is clicked by the user.
Command	occurs when the LinkButton control is clicked and passes its associated <b>CommandName</b> and <b>CommandArgument</b> to the server side event handler.

## Example

### .CS Code

```
protected void LinkButton1_Click(object sender, EventArgs e)
{
    Label1.Text = "This is my Home Page";
}

protected void LinkButton2_Click(object sender, EventArgs e)
{
    Label1.Text = "My Company name R4r TechSoft Solution ";
}

protected void LinkButton3_Click(object sender, EventArgs e)
```

```
{
    Label1.Text = "Welcome";
}
```



## DropDownList in ASP .NET

A DropDownList is also commonly known as combo box. It can contain multiple data members, but unlike a normal list box the users can choose only one value from this control. Though the functionality of this DropDownList is much like a Single Row Select List Box, a DropDownList can save a lot of GUI space as it is rendered on a Single line and is expanded only when the user clicks on the Control.

This DropDownList is provided as a Server Control in ASP .Net like many other controls. This DropDownList can be used to add data manually or even for dynamic binding with data base.

**A simple declaration for DropDownList can be done as below.**

```
<asp:DropDownList ID="DropDownList1" runat="server" Height="47px" Width="73px">
</asp:DropDownList>
```

### Properties Of DropDownList Control

Properties	Discription
BorderColor	enables to get or set the border color of the web server control. [Not Applicable]
BorderStyle	enables to get or set the border style of the control. [Not Applicable]
BorderWidth	enables to get or set the border width for the control. [Not Applicable]
SelectedIndex	enables to get or set an Integer value to specify the zero-based index for the currently selected list item.
AppendDataBoundItems	enables to get or set a Boolean value to indicate whether to clear the list items before binding the data to the control.
AutoPostBack	enables to get or set a Boolean value that indicates whether to postback the web page when list item of DropDownList control is clicked.
DataTextField	enables to get or set the name of the data source field that provides the text for the list items.
DataTextFormatString	enables to get or set the string formatting to control the text of each list item that is to be displayed.
DataValueField	enables to get or set the name of the data source field that provides the value for the list items.
Items	enables to manage the list items of the

	DropDownList control.
SelectedItem	enables to get the currently selected list item in DropDownList control.
SelectedValue	enables to get the value property of selected list item that returns ListItem object
Text	enables to get or set the text for the control.
ValidationGroup	enables to get or set the name of the group of controls to which it belongs to and causes validation when posts back to the server.
DataMember	enables to get or set the name of the list of data that the data-bound control binds to. The list of data items may have more than one data members.
DataSourceID	enables to get or set the ID of the DataSource control from which the data-bound control retrieves its list of data items.
DataSourceObject	enables to get an object that implements the IDataSource interface and provides the access to the data items.
DataSource	enables to get or set the object from which the data-bound control retrieves its list of data items.
CausesValidation	enables to get or set a Boolean value to indicate whether to perform the validation when any list item of the control is clicked.

## Events

**SelectedIndexChanged:** Occurs when selected item of list control changes between postbacks to the server.

**TextChanged:** Occurs when Text and SelectedValue properties change between postbacks to the server.

## Example:



## .CS Code:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        DropDownList1.Items.Add("pototo");
        DropDownList1.Items.Add("Tamato");
    }
}
```



```

DropDownList1.Items.Add("brinjal");
DropDownList1.Items.Add("Lime");
DropDownList1.Items.Add("Pea");

////////////////////////////////////
DropDownList2.Items.Add("Jeans");
DropDownList2.Items.Add("Shirt");
DropDownList2.Items.Add("Cap");
DropDownList2.Items.Add("Blaser");
DropDownList2.Items.Add("Tie");
////////////////////////////////////

DropDownList3.Items.Add("Lather");
DropDownList3.Items.Add("Canvas");
DropDownList3.Items.Add("Sports");
DropDownList3.Items.Add("Sandle");
DropDownList3.Items.Add("Sleeper");

////////////////////////////////////
DropDownList4.Items.Add("Fan");
DropDownList4.Items.Add("TybeLight");
DropDownList4.Items.Add("Plug");
DropDownList4.Items.Add("Holder");
DropDownList4.Items.Add("Wire");
}

protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    ListBox1.Items.Add("Vegitables:" + DropDownList1.SelectedItem.Text);

    //DropDownList1.Items.Clear();
}
protected void ListBox1_SelectedIndexChanged(object sender, EventArgs e)
{
}
protected void DropDownList2_SelectedIndexChanged(object sender, EventArgs e)
{
    ListBox1.Items.Add("Cloths:" + DropDownList2.SelectedItem.Text);
}
protected void DropDownList3_SelectedIndexChanged(object sender, EventArgs e)
{
    ListBox1.Items.Add("Showes:" + DropDownList3.SelectedItem.Text);
}
protected void DropDownList4_SelectedIndexChanged(object sender, EventArgs e)
{
    ListBox1.Items.Add("Electronics:" + DropDownList4.SelectedItem.Text);
}
}

```

## ListBox Control in ASP.NET

The **ASP.Net ListBox control** provides the functionality to display a list of items in the form of menu that allows single or multiple items selection. You can select the multiple list items of ListBox control by changing their selected state. The selected items of ListBox return the Boolean type result as true or false that determine the selected state of an individual list item or multiple list items based on the C# code logic applied on it. If the list item is in **selected state** then it returns true and if it is in un-selected state then the associated list item returns false. You can use the ASP.Net ListBox web server control to get the user input as multiple items selected under a specific group. A simple example of ListBox control is displaying a list of items to display country, state or cities that can be used to select multiple locations to search any desired object based on filter criteria specified by using multiple selection of items of ListBox controls.

### Properties:

**Rows:** The number of rows displayed in the list.

**SelectionMode:** Allows single or multiple selections.

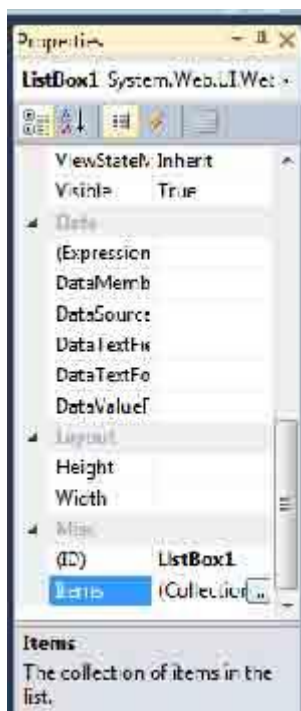
**Items:** The collections of items in the list.

**ForeColor:** Color of the text within the control.

### For example:

Drag the control ListBox and a Button on the form and select ListBox control property items to add the items in list.

Such as:



### Handling ListBox Control Events :

The *SelectedIndexChanged* event in the ListBox class is fired whenever the *SelectedIndex* in the ListBox changes as and when your web page postbacks to the Web Server.

In order for the Postback to happen when the user select the values of the ListBox, you need to set the property *AutoPostBack* = true

```
private void lstBox_SelectedIndexChanged(object sender, System.EventArgs e)
{
    Response.write(lstBox.SelectedItem.Value);
}
```

### Example:

**Registration Form**

Name: ANIL YADAV

Location: kanpur

CollegeList: A.K.G Ghaziebad

Qualification: MCA  
M.TECH  
MBA  
B.TECH  
BCA  
BBA  
PGDCA

[Show Detail](#)

Course: MCA  
M TECH  
BBA  
PGDCA  
Name: ANIL YADAV  
Location: kanpur  
College Name: A.K.G  
Ghanabad

### **.aspx Code:**

```
<%@ Page Language="C#" AutoEventWireup="true" CodeVirtual="simple  
databinding.aspx.cs" Inherits="simple_databinding" %>  
  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
  
<html xmlns="http://www.w3.org/1999/xhtml">  
<head runat="server">  
    <title>Untitled Page</title>  
    <style type="text/css">  
        .style1  
        {  
            width: 25%;  
            border: 1px solid #000080;  
            background-color: #FFCC99;  
            height: 112px;  
        }  
    
```



```

        <tr>
            <td class="style3">
                <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
                    Text="Show Detail" />
            </td>
            <td class="style2">
                <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
            </td>
        </tr>
        <tr>
            <td class="style3">
                &nbsp;</td>
            <td class="style2">
                &nbsp;</td>
        </tr>
    </table>
</div>
</form>
</body>
</html>

```

#### .CS Code:

```

public void bindgrid() //databinding In DropDownList1
{
    OleDbConnection con;
    OleDbDataAdapter da;
    con = new OleDbConnection("provider=microsoft.jet.oledb.4.0;Data
source=D:\\invent.mdb");
    da = new OleDbDataAdapter("select * from purchase", con);
    //DataTable dt = new DataTable();
    DataSet ds = new DataSet();
    da.Fill(ds);
    DropDownList1.DataSource = ds.Tables[0];
    DropDownList1.DataTextField = "ship_location";
    DropDownList1.DataBind();
    for (int i = 0; i < DropDownList1.Items.Count; i++)
    {
        DropDownList1.Items[i].Attributes.Add("style", "color:" + ds.Tables[0].
Rows[i]["categorycolor"].ToString());
    }
}

public void bindlist() //databinding In DropDownList2
{
    OleDbConnection con;
    OleDbDataAdapter da;
    con = new OleDbConnection("provider=microsoft.jet.oledb.4.0;Data
source=D:\\invent.mdb");
    da = new OleDbDataAdapter("select * from collegeList ", con);
}

```

```

//DataTable dt = new DataTable();
DataSet ds = new DataSet();
da.Fill(ds);

DropDownList2.DataSource = ds.Tables[0];

DropDownList2.DataTextField = "colname";
DropDownList2.DataBind();
}

protected void Page_Load(object sender, EventArgs e)
{
    bindgrid();// calling function for dropdownlist1

    bindlist(); // calling function for dropdownlist2

    // Bind data in Listbox
    if (!Page.IsPostBack)
    {
        ListBox1.Items.Add("MCA");
        ListBox1.Items.Add("M.TECH");
        ListBox1.Items.Add("MBA");
        ListBox1.Items.Add("B.TECH");
        ListBox1.Items.Add("BCA");
        ListBox1.Items.Add("BBA");
        ListBox1.Items.Add("PGDCA");
    }
    string loc="";
    protected void Button1_Click(object sender, EventArgs e)
    {
        for (int i = 0; i < ListBox1.Items.Count; i++) // Select Multiple items from
        Listbox
        {
            if (ListBox1.Items[i].Selected == true)
            {
                loc += ListBox1.Items[i].Text + "<br>";
            }
        }

        Label1.Text = "Course::" + " " + loc + "Name::" + TextBox1.Text + "" + "<br>" +
"Location::" +
DropDownList1.SelectedItem.Text + "<br>" + "College Name::" +
DropDownList2.SelectedItem.Text; ;
    }
}

```

## CheckBox in ASP.NET

It is used to select the Boolean value true or false, but not the multiple selections. The CheckBox control creates a check box on the Web Forms page that allows the user to switch between a true or false state. You can specify the caption to display in the control by setting the Text property. The caption can appear either on the right or left of the

check box. Set the TextAlign property to specify the side that the caption appears on.

Note Because the <asp:CheckBox> element has no content, you can close the tag with /> instead of using a separate closing tag.

To determine whether the CheckBox control is checked, test the Checked property. The CheckedChanged event is raised when the state of the CheckBox control changes between posts to the server. You can provide an event handler for the CheckedChanged event to perform a specific task when the state of the CheckBox control changes between posts to the server.

Note: When creating multiple CheckBox controls, you can also use the CheckBoxList control. The CheckBoxList control is easier to use for creating a set of check boxes using data binding, while the individual CheckBox control gives you greater control over layout.

By default, the CheckBox control does not automatically post the form to the server when it is clicked. To enable automatic posting, set the AutoPostBack property to true.

### **Properties**

Property	Description
<b>AutoPostBack</b>	Specifies whether the form should be posted immediately after the Checked property has changed or not. Default is false
<b>CausesValidation</b>	Specifies if a page is validated when a Button control is clicked
<b>Checked</b>	Specifies whether the check box is checked or not
<b>InputAttributes</b>	Attribute names and values used for the Input element for the CheckBox control
<b>LabelAttributes</b>	Attribute names and values used for the Label element for the CheckBox control
<b>runat</b>	Specifies that the control is a server control. Must be set to "server"
<b>Text</b>	The text next to the check box
<b>TEXTALIGN</b>	On which side of the check box the text should appear (right or left)
<b>ValidationGroup</b>	Group of controls for which the Checkbox control causes validation when it posts back to the server
<b>OnCheckedChanged</b>	The name of the function to be executed when the Checked property has changed

### **Example:**

## Output:

Subject	Fees
Name	ANIL YADAV
<input checked="" type="checkbox"/> C	1000
<input type="checkbox"/> C++	1500
<input checked="" type="checkbox"/> JAVA	3500
<input type="checkbox"/> ORACLE	2000
<input checked="" type="checkbox"/> COMPLETE NET	1200
<input type="checkbox"/> C#	1000

Calculate Fees

MR. ANIL YADAV  
Your Total Fees: 5700

## .aspx Code:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeVirtual="Checkbox.aspx.cs"
Inherits="Checkbox" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Untitled Page</title>
  <style type="text/css">
    .style1
    {
        width: 34%;
        height: 520px;
        border: 4px solid #000080;
        background-color: #FFFF99;
    }
    .style2
    {
        height: 133px;
        width: 43px;
    }
    .style3
    {
        height: 23px;
        width: 43px;
    }
    .style4
    {
        height: 21px;
        width: 43px;
    }
    .style5
    {
```



```
width: 176px;
}
.style6
{
height: 23px;
width: 176px;
}
.style7
{
height: 21px;
width: 176px;
}
.style8
{
height: 133px;
width: 176px;
}
.style9
{
width: 43px;
}
.style10
{
width: 49px;
}
.style11
{
height: 23px;
width: 49px;
}
.style12
{
height: 21px;
width: 49px;
}
.style13
{
height: 133px;
width: 49px;
}
</style>
</head>
<body>
<form id="form1" runat="server">
<div>

<table class="style1">
<tr>
<td class="style5">
Subject</td>
<td class="style9">
Fees</td>
<td class="style10">
&nbsp;</td>
</tr>
<tr>
<td class="style5">
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;& Name&nbsp;</td>
<td class="style9">
```

```

        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    </td>
    <td class="style10">
        &nbsp;   </td>
</tr>
<tr>
    <td class="style5">
        <asp:CheckBox ID="CheckBox1" runat="server" Text="C" />
    </td>
    <td class="style9">
        <asp:Label ID="Label1" runat="server" Text="1000"></asp:Label>
    </td>
    <td class="style10">
        &nbsp;   </td>
</tr>
<tr>
    <td class="style6">
        <asp:CheckBox ID="CheckBox2" runat="server" Text="C++" />
    </td>
    <td class="style3">
        <asp:Label ID="Label2" runat="server" Text="1500"></asp:Label>
    </td>
    <td class="style11">
        &nbsp;   </td>
</tr>
<tr>
    <td class="style6">
        <asp:CheckBox ID="CheckBox3" runat="server"
            oncheckedchanged="CheckBox3_CheckedChanged" Text="JAVA" />
    </td>
    <td class="style3">
        <asp:Label ID="Label3" runat="server" Text="3500"></asp:Label>
    </td>
    <td class="style11">
        &nbsp;   </td>
</tr>
<tr>
    <td class="style7">
        <asp:CheckBox ID="CheckBox4" runat="server" Text="ORACLE" />
    </td>
    <td class="style4">
        <asp:Label ID="Label4" runat="server" Text="2000"></asp:Label>
    </td>
    <td class="style12">
        &nbsp;   </td>
</tr>
<tr>
    <td class="style5">
        <asp:CheckBox ID="CheckBox5" runat="server"
            oncheckedchanged="CheckBox5_CheckedChanged" Text="COMPLETE
.NET" />
    </td>
    <td class="style9">
        <asp:Label ID="Label5" runat="server" Text="1200"></asp:Label>
    </td>
    <td class="style10">
        &nbsp;   </td>
</tr>
</tr>

```

```

        <td class="style5">
            <asp:CheckBox ID="CheckBox6" runat="server" Text="C#" />
        </td>
        <td class="style9">
            <asp:Label ID="Label6" runat="server" Text="1000"></asp:Label>
        </td>
        <td class="style10">
            &nbsp;</td>
    </tr>
    <tr>
        <td class="style5">
            <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
                Text="Calculate Fees" />
        </td>
        <td class="style9">
            &nbsp;</td>
        <td class="style10">
            &nbsp;</td>
    </tr>
    <tr>
        <td class="style8">
            &nbsp;</td>
        <td class="style2">
            <asp:Label ID="Label7" runat="server" Text="Label"></asp:Label>
        </td>
        <td class="style13">
            &nbsp;</td>
    </tr>
</table>

</div>
</form>
</body>
</html>

```

### .CS Code:

```

int p, q, r, s, t, u;
protected void Button1_Click(object sender, EventArgs e)
{
    if (CheckBox1.Checked == true)
    {
        p = Int32.Parse(Label1.Text);
    }
    if (CheckBox2.Checked == true)
    {
        q = Int32.Parse(Label2.Text);
    }
    if (CheckBox3.Checked == true)
    {
        r = Int32.Parse(Label3.Text);
    }
    if (CheckBox4.Checked == true)
    {
        s = Int32.Parse(Label4.Text);
    }
}

```

```

    }
    if (CheckBox5.Checked == true)
    {
        t = Int32.Parse(Label5.Text);
    }
    if (CheckBox6.Checked == true)
    {
        u = Int32.Parse(Label6.Text);
    }
    int k = p + q + r + s + t + u;
    Label7.Text = "MR.." + TextBox1.Text + "<br>" + "Your Total Fees::" + k.ToString();
}

```

## CheckBoxList Control in ASP.NET

The ASP.NET CheckBoxList control is used to create a defined list of options in which a user can choose from. With this control, you can select one item or multiple items. Each selectable item in a CheckBoxList control is considered a ListItem element. This control can be loaded manually through ASP.NET code (like my example) as well as from a data source. The CheckBoxList control is part of the System.Web.UI.WebControls namespace.

### Important Properties:

SelectedValue	Gets the value of first selected item.
SelectedIndex	Gets or Sets the index of the first selected item.
SelectedItem	Gets the first selected item
TextAlign	Gets or Sets the alignment of the checkbox text.
DataTextField	Name of the data source field to supply the text of the items. (No need to set when you are adding items directly into .aspx page.)
DataValueField	Name of the data source field to supply the value of the items. (No need to set when you are adding items directly into .aspx page.)
DataSourceID	ID of the datasource component to provide data. (Only used when you have any DataSource component on the page, like SqlDataSource, AccessDataSource etc.)
DataSource	The datasource that populates the items in the checkboxlist box. (Generally used when you are dynamically generating the items from Database.)
AutoPostBack	true/false. If true, the form is automatically posted back to the server when user click any of the checkbox. It will also fire OnSelectedIndexChanged method.
AppendDataBoundItems	true/false. If true, the statically added item (added from .aspx page) is maintained when adding items dynamically (from code behind file) or items are cleared.
OnSelectedIndexChanged	Method name that fires when user click any of the checkbox in the list. (Fires only when AutoPostBack=true.)
Items	Gets the collection of the items from the list.
RepeatLayout	table/flow. Gets or Sets the layout of the checkboxes when rendered.





```

{
    for (int i = 0; i < CheckBoxList1.Items.Count; i++)
    {
        if (CheckBoxList1.Items[i].Selected == true)
        {
            loc += CheckBoxList1.Items[i].Text + "<br>";
        }
    }

    Label1.Text = "Items::" + " " + loc;
}

```

## RadioButton Control in ASP.NET

RadioButton control is used to give single select option to the user from multiple items. When it is rendered on the page, it is implemented through `<input type=radio></input>` HTML tag. Its properties like BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, Height etc. are implemented through style properties of `<input>`.

### Important Properties:

AutoPostBack	Form is automatically posted back when Radio button selection is changed.
CausesValidation	true/false. If true, Form is validated if Validation control has been used in the form.
Checked	true/false. If true, Radio button is selected by default.
OnCheckedChanged	Fires when Radio button selection changes. This works only if AutoPostBackproperty is set to true.
ValidationGroup	Used to put a radio button under a particular validation group. It is used when you have many set of form controls and by clicking a particular button you want to validate a particular set of controls only.
GroupName	It is used a group a set of radion buttons so only one of them can be selected a time.

### Event:

**CheckedChanged Event:** It enables you to handle the PostBack event of radiobutton control when user changes its state from selected state. When user clicks the radio button it raises the CheckedException event only if the AutoPostBack property of RadioButton control is set to true

### Example:

### Output:



### .CS Code:

```
protected void Button3_Click(object sender, EventArgs e)
{
    if (RadioButton1.Checked == true)
    {
        Label2.Text = "Your Answer is Wrong";
    }
    if (RadioButton2.Checked == true)
    {
        Label2.Text = "Your Answer is Wrong";
    }
    if (RadioButton3.Checked == true)
    {
        Label2.Text = "Your Answer is Right " + "<br>" + "Yow win $100";
    }
    if (RadioButton4.Checked == true)
    {
        Label2.Text = "Your Answer is Wrong";
    }
}
```

## RadioButtonlist Control in ASP.NET

RadioButtonList Control is used for selecting one predefined (available in RadioButtonList) value. That means user can be able to select only one a time. For e.g., it is useful for selecting the User Sex , Highest Qualification etc., ASP.NET RadioButtonList itself is a class like other controls which are present under System.Web.UI.WebControls namespace.

### Important Properties

**AutoPostBack** - Accept Boolean Value - If True, then page will submit to the server if the index of the Radiobutton is changed

**CausesValidation** - Accept Boolean Value - Used to Validate the page when RadioButton click event fires.

**DataTextField** - Accept String Value - Used to set the column Name from the DataSource which user Can see (eg., Name).



**DataValueField** - Accept String Value - Used to set the column Name from the DataSource which user Can not see (eg., ID).

**Enabled** - Accept Boolean Value- Used to Get/Set Radio button Enabled

**Height** - Accept IntegerValue - Used to Set/Get height of the RadioButton

**RepeatColumns** - Accept Integer Value - Used to set how Many RadioButtons you want to display in one line.

**RepeatDirection** - Enumerator - "Vertical" or "Horizontal"

**Width** - Accept IntegerValue - Used to set the width of the RadioButton

#### Events :

**DataBound** - This event will fire when you call the assigned dataSource property and you write this line of code to bind your RadioButtonList

```
myRadioButtonList.DataBind();
```

**SelectedIndexChanged** - This event will fire when user has changed his/her selection and RadioButtonList's "AutoPostBack" property is set to true;

#### Example:

#### Output:



#### .CS Code:

```
protected void Page_Load(object sender, EventArgs e)
{
    if(!Page.IsPostBack)
    {
        RadioButtonList1.Items.Add("Salman Khan");
        RadioButtonList1.Items.Add("Sharukh Khan");
        RadioButtonList1.Items.Add("Amir Khan");
        RadioButtonList1.Items.Add("Hritik");
        RadioButtonList1.Items.Add("Ajay Devgan");
        RadioButtonList1.Items.Add("Ranveer Kapoor");
    }
}
```

```

RadioButtonList1.Items.Add("Akshay Kumar");
RadioButtonList1.Items.Add("Sahid kapoor");
RadioButtonList1.Items.Add("Sanjay Dutt");
}

}
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = "Your Favrouite Actor::> " + RadioButtonList1.SelectedItem.Text;
}

```

## Image Control in ASP.NET

**ASP.Net 2.0 Image control** provides the functionality to display images on ASP.Net web page. The image server control renders using HTML img tag in the web browser. As a web server control it can be accessed at server side that enables you to set the images dynamically using C# code or by specifying the image URL retrieved from SQL database table. It provides properties similar to HTML img tag that allow you to set the path of image file that you want to display on web page along with its alignment and alternate text properties that render as attributes of HTML img tag in the web browser. You can get or set these properties at design time as well as using C# code programmatically. The ASP.Net image control allows you to display the images only for design purposes or as a part of the page content but does not support any server events to execute the server side code if a user clicks on it or moves the mouse over it.

### Properties:

1. **AlternateText:** You can specify the text value for this property that appears as tooltip for the displayed image in the web browsers. All web browsers do not support this property to display the text as tooltip. The web browsers display this alternate text value if the specified image does not load for any reason like unreachable or unavailable at that time. This property renders as alt attribute of img tag.
2. **DescriptionUrl:** You can specify the URL of the web page containing the full description regarding the image.
3. **GenerateEmptyAlternateText:** It is a Boolean type property that allows or disallows to generate the empty alt attribute while rendering the image control.
4. **ImageAlign :** It enables you to specify the alignment of image as a part of inline content or web page elements.
5. **ImageUrl :** You can specify the URL of the image that you want to display on the web page.
6. **ToolTip:** It enables you to specify the tooltip text for the image control. The tooltip property generates the title attribute of img tag that appears as tooltip in some browsers.

Apart from these properties there some other CSS based properties that enable you to set the border, background color, height and width of the image. The CSS based properties include: BackColor, BorderColor, BorderStyle, BorderWidth, CssClass, Height and Width that allow you to customize the appearance of image.

### Syntax:

```
<asp:Image ID="Image1" runat="server" Height="215px"ImageUrl="~/Images/Sunset.jpg" />
```

### Example:

				
Name	location	Email	Description	Date
Ashish Kumar Gupta	Ghaziabad	Ashish07akg@gmail.com	very Nice Article	1/12/2011 11:22 AM
Anil	Delhi	anil.rkgit@gmail.com	Very nice Article Thanks	1/12/2011 11:25 AM
Anil	Delhi	anil.rkgit@gmail.com	Very nice Article Thanks	1/12/2011 11:30 AM

### Design:



## ImageMap Control in ASP.NET

ImageMap control is used to create an image that contains clickable hotspot region. When user click on the region, the user is either sent to a URL or a sub program is called Imagemap. When it is rendered on the page, it is implemented through `<img />` HTML tag.

### Hot spots features:

- There is no limit on number of hotspots each image may contain.
- Each hotspot is characterized by various attributes like shape, location and size.
- Overlapping hotspots are perfectly valid.
- Hot spots are defined using x and y coordinates.
- Hot spots can be assigned Image URL's and are capable of postback.

### Different types of hot spots:

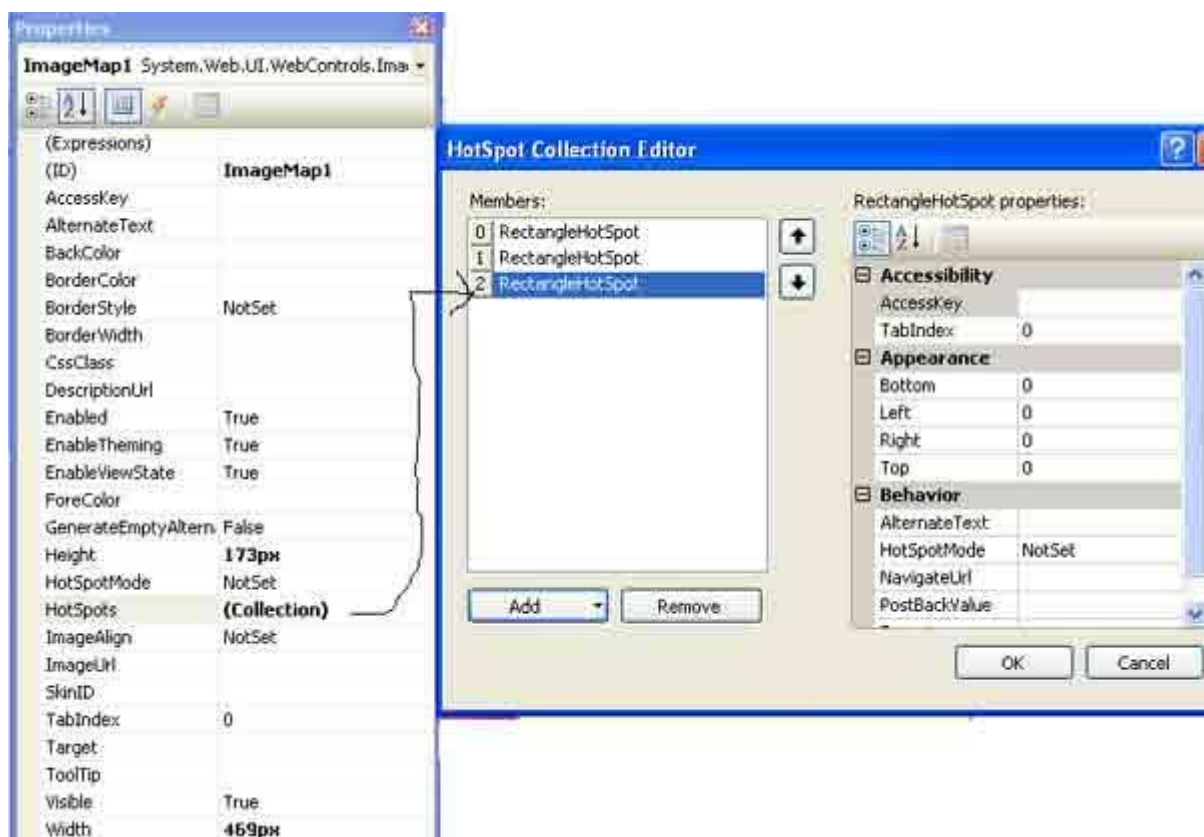
There are three different types of hot spots offered by ImageMap control. They are:

- CircleHotspot
- RectangleHotspot
- PolygonHotspot

**CircleHotspot:** CircleHotspot defines circle shaped hot spot region in an ImageMap control. To define the region for a circle hot spot, we should define X and Y coordinates for circle as well as radius property which usually is the distance from the center of circle to the edge.

**RectangleHotspot:** RectangleHotspot defines rectangle shaped hot spot region in an ImageMap control. To define the region for a Rectangle hot spot, we define Top, Bottom, Left and Right coordinates. Similar is the case for the Polygon hot spot.

### Rightclick On Imagemap Control :



### Example:



if you click the left side of the image you see below image



if you click the Right side of the image you see below image



## Calendar control In ASP.net

This control displays a one-month calendar that allows the user to select dates and move to the next and previous months.

### Properties:

Property	Description
DayHeaderStyle	The style for displaying the names of the days
DayNameFormat	The format for displaying the names of the days. Can take one of the following values: <ul style="list-style-type: none"><li>• FirstLetter</li><li>• FirstTwoLetters</li><li>• Full</li><li>• Short</li></ul>
SelectMonthText	The text displayed for the month selection link
SelectWeekText	The text displayed for the week selection link
TitleFormat	The format for the title of the calendar. Can take one of the following values: <ul style="list-style-type: none"><li>• Month</li><li>• MonthYear</li></ul>
TodaysDate	Today's date
VisibleDate	The date that specifies the month that is currently visible in the calendar
WeekendDayStyle	The style for weekends

Here I show how we can use Calendar control in asp.net. Create a web form name Calendar.aspx, and then add a Calendar control and TextBox control. Here we use OnSelectionChanged event in calendar control. When someone selects a date from Calendar, the TextBox control shows the selected date.

#### .ASPX Code:

```
<asp:TextBox ID="TextBox1" runat="server" Height="32px" Width="234px"></asp:TextBox>

"FFCC99#"=BorderColor "server"=runat "Calendar1"=ID asp:Calendar>
"Dashed"=BorderStyle
"Calendar1_SelectionChanged"=onselectionchanged
<"Please Select Date"=ToolTip "01-11-2010"=SelectedDate
</ "FFFFCC#"=BackColor TitleStyle>
<asp:Calendar/>
```

#### .CS Code:

```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
    TextBox1.Text = Calendar1.SelectedDate.ToString(); //To
display date in TextBox
}
```

#### Output:



## FileUpload Control in ASP.net

Finally ASP.NET 2.0 has a **FileUpload** control, which allows developers to drop the control on a page and let it browse a file and upload it on the server. To create a control, simply drop the FileUpload control from Toolbox to a Web page.

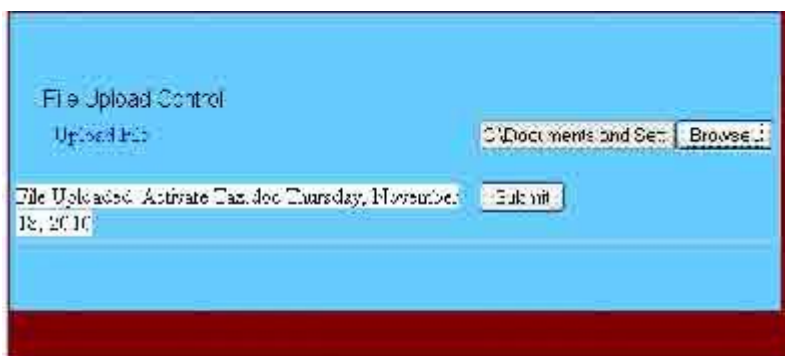
### .ASPX CODE

```
<asp:FileUpload ID="FileUpload1" runat="server" />
<asp:Label ID="Label1" runat="server" BackColor="White"
Text="Label"></asp:Label>
<asp:Button ID="Button1" runat="server" onclick="Button1_Click"
Text="Submit" />
```

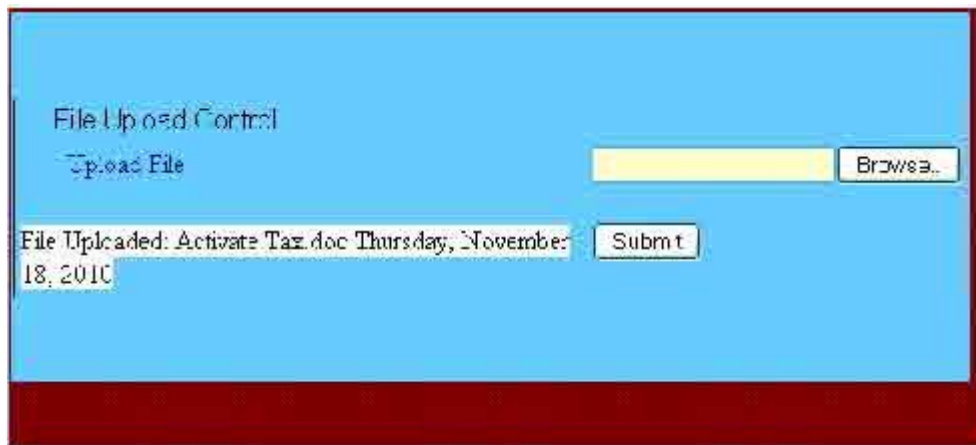
### .CS CODE

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (FileUpload1.HasFile)
    {
        FileUpload1.SaveAs(@"D:\temp\"
+FileUpload1.FileName);
        Label1.Text = "File Uploaded: " +
FileUpload1.FileName + "
"+System.DateTime.Now.ToLongDateString();
    }
    else
    {
        Label1.Text = "No File Uploaded.";
    }
}
```

### OUTPUT:



After Click on Submit Button file Saved in Folder Temp



MultiView is a simpler way to declare multiple views (<asp:View> is a separate control that is used as a child control of MultiView control) and show only one at a time. It has no default interface like Calendar, you get only whatever you add in its different views. When it is rendered on the page, it is implemented through the HTML tag you place inside views.

You can navigate through views either by setting its ActiveViewIndex property or by putting buttons with preset command names. To Navigate through different views, you need to put Buttons/LinkButton/ImageButton on the views with CommandName as NextView(to navigate to next view) and PrevView(to navigate to previous view).

Following are some important properties that are very useful.

ActiveViewIndex	Gets or sets the active view index.
OnActiveViewChanged	Method name that fires when Active View Changed (When you click Next or Previous buttons).

DEMO : MultiView

[Show Source Code](#)

**Header of View 1**

Navigate Views:

Navigate Views:

```
// MultiView //////////////////////////////////////
<asp:MultiView ID="MutliView1" runat="Server" ActiveViewIndex="0"
OnActiveViewChanged="AutoSelectDropDown">
  <asp:View ID="View1" runat="server">
    <table style="border:1px solid #c0c0c0;">
      <tr style="background-color:#c0c0c0;">
        <th>
          Header of View 1
        </th>
      </tr>
      <tr>
        <td>
          <asp:TextBox ID="textBox1"

```



```

runat="Server"></asp:TextBox>
        <asp:Button ID="btnSubmit" runat="Server" Text="Submit"
OnClick="SubmitFrom1stView" />
    </td>
</tr>
<tr>
    <td>
        <hr />
        Navigate Views:<br />
        <asp:Button ID="cmdNext1" runat="server" Text="Next
View >" CommandName="NextView" Font-Size="8pt" />
    </td>
</tr>
</table>
</asp:View>
<asp:View ID="View2" runat="Server">
    <table style="border: 1px solid #c0c0c0;">
        <tr style="background-color: #c0c0c0;">
            <th>
                Header of View 2
            </th>
        </tr>
        <tr>
            <td>
                <asp:Image ID="img1" runat="Server"
ImageUrl="~/images/samples/NewAdvance.gif" AlternateText="DotNetFunda.com"
/>
            </td>
        </tr>
        <tr>
            <td><hr />
                Navigate Views:<br />
                <asp:Button ID="Button4" runat="server" Text="< Prev
View" CommandName="PrevView" Font-Size="8pt" /> |
                <asp:Button ID="Button1" runat="server" Text="Next View
>" CommandName="NextView" Font-Size="8pt" />
            </td>
        </tr>
    </table>
</asp:View>
<asp:View ID="View3" runat="Server">
    <table style="border: 1px solid #c0c0c0;">
        <tr style="background-color: #c0c0c0;">
            <th>
                Header of View 3
            </th>
        </tr>
        <tr>
            <td>
                <asp:Image ID="Image1" runat="Server"
ImageUrl="~/images/samples/DotNetFunda3.gif"
AlternateText="DotNetFunda.com" />
            </td>
        </tr>
        <tr>
            <td>
                <hr />
                Navigate Views:<br />
                <asp:LinkButton ID="Button2" runat="server" Text="<
Previous" CommandName="PrevView" /> |
                <asp:LinkButton ID="Button3" runat="server" Text="Next
>" CommandName="NextView" />
            </td>
        </tr>
    </table>
</asp:View>

```

```

        </td>
    </tr>
</table>
</asp:View>
<asp:View ID="View4" runat="Server">
    <table style="border: 1px solid #c0c0c0;">
        <tr style="background-color: #c0c0c0;">
            <th>
                Header of View 4
            </th>
        </tr>
        <tr>
            <td>
                <asp:Image ID="Image2" runat="Server"
ImageUrl="~/images/samples/DotNetFunda2.gif"
AlternateText="DotNetFunda.com" />
            </td>
        </tr>
        <tr>
            <td>
                <hr />
                Navigate Views:<br />
                <asp:Button ID="Button5" runat="server" Text="< Prev
View" CommandName="PrevView" Font-Size="8pt" />
            </td>
        </tr>
    </table>
</asp:View>
</asp:MultiView>

// Server Side Event //////////////////////////////////////
protected void Page_Load(object sender, EventArgs e)
{
    // fires only when Form is posted back
    if (!IsPostBack)
    {
        dropViews.DataSource = MutliView1.Views;
        dropViews.DataTextField = "ID";
        dropViews.DataBind();
    }
}

// Fires when 1st view button is clicked
protected void SubmitFrom1stView(object sender, EventArgs e)
{
    lblMessage.Text = "<hr />TextBox Value is: <b>" + textBox1.Text +
"</b>";
}

// Fires when DropDown selected index changes
protected void ChangeTheViews(object sender, EventArgs e)
{
    MutliView1.ActiveViewIndex = dropViews.SelectedIndex;
}

// Fires when Active Window changes.
// When Next or Previous button is clicked.
protected void AutoSelectDropDown(object sender, EventArgs e)
{

```

```
dropViews.SelectedIndex = MutliView1.ActiveViewIndex;
}
```

## Wizard control in ASP.NET

The Wizard control is a perfect fit when you need to develop a web site that requires multiple steps to collect and display the data.

### Advantages of using a Wizard Control

- \* A header for the title of a particular step
- \* A sidebar for left bar navigation
- \* collection of steps
- \* types of steps

### Wizard Layout

A Header appears at the top of each WizardStep display. The easiest way to display a common heading is by coding the Wizard's Header Text property as is done above. If no header is provided, none is displayed.

A Sidebar menu appears along the left side of the display. It presents a menu of links to the Wizard Steps that comprise the sequence of page displays. A Side Bar is optional and can be suppressed by coding the Wizard's **DisplaySideBar**="False" property. The Side Bar is centered vertically within the total height of the Wizard.

### .ASPX CODE

[Download ASPX CODE](#)

### .CS Code

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

public partial class Wizard : System.Web.UI.Page
{
    protected void Page_Load(object sender,
```

```

EventArgs e)
{
    Label2.Visible = false;
    Label3.Visible = false;
    Label4.Visible = false;
    Label5.Visible = false;
    // Wizard1.Visible = false;

}
protected void Button1_Click(object sender,
EventArgs e)
{
    if (TextBox1.Text == "ashish" &&
    TextBox2.Text == "gupta")
    {
        Label1.Text = "Login Are successfully";

    }
    else
        Label1.Text = "username And password
wrong";
}

protected void Button3_Click(object sender,
EventArgs e)
{
    Label2.Text = "Name:" + "" +
    TextBox3.Text;
    Label3.Text = " Father Name:" + "" +
    TextBox4.Text;
    Label4.Text = "Address:" + "" +
    TextBox5.Text;
    Label5.Text = "phone no:" + "" +
    TextBox6.Text;
    Label2.Visible = true;
    Label3.Visible = true;
    Label4.Visible = true;
    Label5.Visible = true;

}
}

```

## OUTPUT:

### 1.LOGIN PAGE

[Login](#)  
[Registration](#)  
[Contact Us](#)

**Login Page**

Username:

Password:

Login Are successfully

## 2.REGISTRATION

[Login](#)  
[Registration](#)  
[Contact us](#)

**REGISTRATION PAGE**

Name:

Father Name:

Address:

Phone no.:

Name:amit  
 Father Name:sunil  
 Address:ghaziabad  
 phone no:9990512118

## 3.Contact Us

[Login](#)  
[Registration](#)  
[Contact us](#)

**Contact Us**



Ghaziabad:

NH-24 Shanti Nagar,  
 Near Kamla Hall, Ghaziabad  
 Uttar Pradesh, India

## Navigation Control Example in ASP.NET

### SiteMapPath Control:

The SiteMapPath control is used to add a site map (breadcrumbs) to the website. A site map is a way to present all folders and pages of the website.

Its properties like BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, Height etc. are implemented through style properties of <span/> tag. Generally current page reference is not rendered as a link, however, it can be made clickable by setting RenderCurrentNodeAsLink=true.

### Following are some important properties that are very useful

PathSeparator	Gets or sets Path separator text. (By default it is >.)
NodeStyle	Sets the style of all nodes that will be displayed.
CurrentNodeStyle	Sets the style on node that represent the current page.
RootNodeStyle	Sets the style on the absolute root node.
PathSeparatorStyle	Sets the style of path separator.

### The following sitemap file is used in this tutorial:

```
<?xml version="1.0" encoding="utf-8" ?>

<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >

  <siteMapNode url="" title="" description="">

    <siteMapNode url="Default2.aspx" title="Home" description="" />

    <siteMapNode url="Default3.aspx" title="About Us" description="" />

    <siteMapNode url="Default.aspx.aspx" title="Product" description="" />

    <siteMapNode url="Adrotator.aspx.aspx" title="Contact us" description="" />

  </siteMapNode>

</siteMap>
```

Rules for creating a sitemap file:

- The XML file must contain a <siteMap> tag surrounding the content

- The <siteMap> tag can only have one <siteMapNode> child node (the "home" page)
- Each <siteMapNode> can have several child nodes (web pages)
- Each <siteMapNode> has attributes defining page title and URL

### **Dynamic Menu:**

The <asp:Menu> control displays a standard site navigation menu.

### **Code Example:**

```
<asp:SiteMapDataSource id="nav1" runat="server" />

<form runat="server">
<asp:Menu runat="server" DataSourceId="nav1" />
</form>
```

The **<asp:Menu>** control in the example above is a placeholder for a server created navigation menu.

The data source of the control is defined by the **DataSourceId** attribute. The **id="nav1"** connects it to the **<asp:SiteMapDataSource>** control.

The **<asp:SiteMapDataSource>** control automatically connects to the default sitemap file (**web.sitemap**).

### **TreeView:**

The <asp:TreeView> control displays a multi level navigation menu.

The menu looks like a tree with branches that can be opened or closed with + or - symbol.

### **Code Example:**

```
<asp:SiteMapDataSource id="nav1" runat="server" />

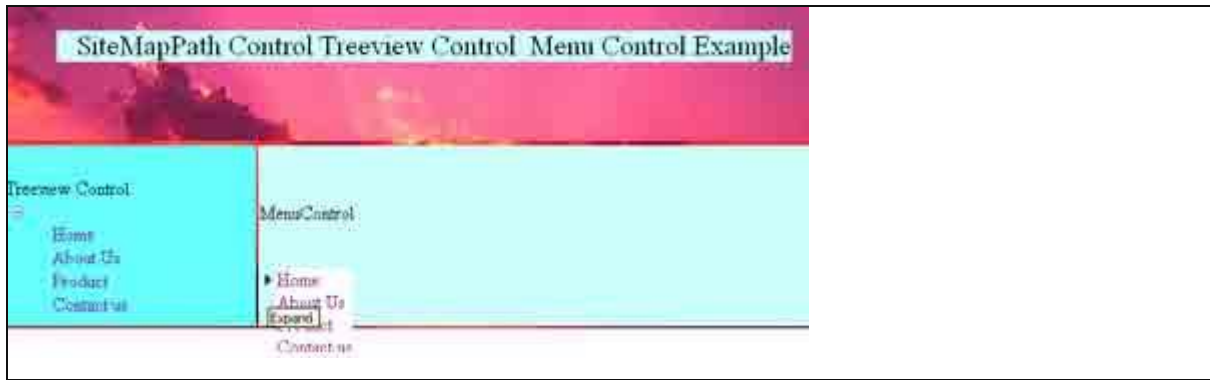
<form runat="server">
<asp:TreeView runat="server" DataSourceId="nav1" />
</form>
```

The **<asp:TreeView>** control in the example above is a placeholder for a server created navigation menu.

The data source of the control is defined by the **DataSourceId** attribute. The **id="nav1"** connects it to the **<asp:SiteMapDataSource>** control.

The **<asp:SiteMapDataSource>** control automatically connects to the default sitemap file (**web.sitemap**).

### **Design Output:**



## Validation Control In ASP.NET

### Introduction

ASP.NET validation controls provide an easy-to-use but powerful mechanism of ensuring that data is entered correctly on the forms. There are 6 validation controls included in the ASP.NET 2.0 and ASP.NET 3.5 versions.

ASP.NET validation controls also provide two ways of validation: Server-side or Client-side. The nice thing about these Validation controls is that it will perform client-side validation when it detects the browser is able (unless client-side validation has been disabled). Thus reducing roundtrips. And it will perform server-side where necessary. This client-side/server-side detection and validation is done without extra work by the developer!

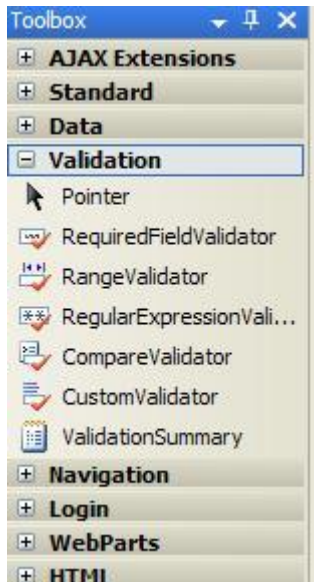
In ASP.NET, there are five(6) Validation controls included. They are:

Validation Server Control	Description
CompareValidator	Compares the value of one input control to the value of another input control or to a fixed value
CustomValidator	Allows you to write a method to handle the validation of the value entered
RangeValidator	Checks that the user enters a value that falls between two values
RegularExpressionValidator	Ensures that the value of an input control matches a specified pattern
RequiredFieldValidator	Makes an input control a required field
ValidationSummary	Displays a report of all validation errors occurred in a Web page

### How to use Validation Control IN Website:

Create a new website project and you have an Empty default.aspx page. In your Solution Explorer double click it and go to its design view. Look for your toolbar on the left hand side of your window. In the Toolbar look for the Validation Section as depicted below





These controls are used to validate the user input provided at the standard controls.

Note: \*If the validation control returns 'false' then the data will not be submitted to the server.

\*A single validation control can validate a single standard control.

\*A single standard control can be binded with any no. of validation controls.

\*Validation scripts should not be altered from ASP. Net 2.0 onwards. If any modifications are performed on the scripts then all the ASP.Net websites which uses validation controls will not function properly.

\*Validation controls can be grouped together from ASP. Net 2.0 onwards

### **Difference Between Server side Control And Client Side Control**

1..Client side validation is processed the client side before submitting the form. The advantage of using the client side validation is it reduces the network traffic since the validation is processed in the client machine itself.

2..Server side validation is processed in the server. Some data cannot be validated in the client side and it has to be validated in the server side. E.g. Date between the two dates in the database.

3..Client-side is faster than server-side as the networking time from client to server is saved.

4..server-side is done on the server. Then the server converts the data into an html page and sends to the browser.

5..server-side is more secure as the user cannot see the code even he does a view-source.

6..The disadvantage of javascript validation is that users can have javascript turned off and therefore can submit invalid data.

7..The difference is that the javascript validation is done on the client side (if the user's browser supports it, so never depend on javascript alone), before the page is submitted to the server. The advantage is that the user gets immediate feedback and does not need to wait for the page to reload. It also reduces server load.

## Required FieldValidator control

It is used to verify if a value has been provided for a control or not.

As the name suggest, this validation control make sure that control mention in Control ToValidate cannot be empty.

### Common properties for all the validation controls:

<b>ControlToValidate</b>	Specifies the name of the control which has to be validated.
<b>ErrorMessage</b>	Specifies the error message to be displayed if the validation control returns 'false'.

**Note:** If customization or if validation controls has to be extended then the "Text" property for the validation controls should be set. If more than one validation control validates a standard control then it is mandatory that the "Display" property for all the validation controls should be set as "Dynamic". Other than the 'RequiredFieldValidator' control, remaining validation controls will not validate the controls if the value for the control is empty.

### .Aspx Code

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

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="TextBox1" ErrorMessage="* Blank are not allowed" SetFocusOnError="True" Display="Dynamic"></asp:RequiredFieldValidator>
```

Output:



## RegularExpressionValidator control

It is used to validate the user input based on the validation expression.

### Meta characters used to prepare the expression for single digit

\d	Accepts a single digit
/D	Accepts a single character
\w	Accepts any character other than a white space (spacebar)
\s	Accepts a space
[A-Za-z]	Accepts upper or lower case characters
[0-9]	Accepts a numerical value
^[0-9]	Accepts any characters other than numeric value

### Occurrences:

It is used to specify the occurrence of the meta characters within the expression.

{number}	Accepts the input if the length of the expression is equal to the specified number E.g.: \d{5} à accepts 5 digits number
{Minnumber, }	Accepts the input if the length after expression is greater than or equal to the specified Minnumber. E.g: [A-Za-z0-9_]{6, }
{Minnumber, Maxnumber}	Accepts the input if the length of the expression is between the specified range. e.g.: \D{6,8}

### Modes:

It is used to specify the occurrence of the meta characters within the expression.

Mode	MinOccurrence	MaxOccurrence
?	0	1
. or *	0	Any
+	1	Any

E.g.: \d{1, } <==> \d+

[ma+am] => accepts 'madam, Malayalam .....

### .ASPX CODE

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

```
<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
ControlToValidate="TextBox1" ErrorMessage="plese enter email ID"
ToolTip="plese Enter email id" ValidationExpression="\w+([-+.']\w+)*@\w+([-
.\w+)*\.\w+([-.\w+)*"></asp:RegularExpressionValidator>
```

### Output:



## Custom Validator control

It is used to validate user input based on the user defined functionality.

OR Custom validator control is used to capture the validation that cannot be handled by the validator controls provided by ASP.NET.

Here user is at the freedom to define his own custom method, both Client side and Server side.

CustomValidator is the only validator control which has the capability to validate the user input at the client side and also at the server side.

Other than the CustomValidator, all the controls will validate the user input at the client side.

### Property

ClientValidationFunction	Specifies the JavaScript function name, which provides the definition to validate the user input at the client side.
--------------------------	--

### Event

ServerValidate	The code written within this event will be used to validate the user input at the server side.
----------------	--

### .ASPX CODE

```
Custom text:<br />
<asp:TextBox runat="server" id="txtCustom" />
<asp:CustomValidator runat="server" id="cusCustom" controltovalidate="txtCustom"
onservervalidate="cusCustom_ServerValidate" errormessage="The text must be exactly
8 characters long!" />
<br /><br />
```

### .CS CODE

```
protected void CustomValidator1_ServerValidate(object sender,
ServerValidateEventArgs e)
{
    if (e.Value.Length == 8)
        e.IsValid = true;
    else
        e.IsValid = false;
}
```

**ValidateEmptyText** is a boolean attribute that if set to true, the input control will be validated even if it is empty.

**ClientValidationFunction** contains name of client validation function.

**OnServerValidate** contains name of server validation function.

## Validation Summary control

This control is used to display the list of all the validation error that has occurred on the page. The error message displayed is the one set for the **ErrorMessage** attribute of the validation control. No error message will be displayed if this attribute is not set.

ASP.NET has provided an additional control that complements the validator controls. The validation summary control will collect all the error messages of all the non-valid controls and put them in a tidy list. The list can be either shown on the web page (as shown in the example above) or with a popup box (by specifying ShowMessageBox="True")..

**DisplayMode** has three options *List*, *BulletList* and *SingleParagraph*

**ShowMessageBox** when set to **true** will display the error as a alert popup

**ShowSummary** will display the error on the page. By default it is true.

### .ASPX CODE

```
<%@ Page Language="C#" AutoEventWireup="true" CodeVirtual="Default4.aspx.cs"
Inherits="Default4" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
    <style type="text/css">
        .style1
        {
            width: 52%;
            border-left-style: solid;
            border-left-width: 1px;
            border-right: 1px solid #C0C0C0;
            border-top-style: solid;
            border-top-width: 1px;
            border-bottom: 1px solid #C0C0C0;
            background-color: #008000;
        }
        .style2
```

```

{
    color: #FFFFFF;
}
.style3
{
    width: 126px;
}
</style>
</head>
<body>
<form id="form1" runat="server">
<div>

    <table class="style1">
        <tr>
            <td class="style2" colspan="2">
                &nbsp; Regular&nbsp; Expression Validator&nbsp;</td>
            <tr>
                <td class="style2">
                    Email ID</td>
                <td class="style3">

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

                </td>

                <td>

                    <asp:RegularExpressionValidator ID="RegularExpressionValidator1"
runat="server"
                    ControlToValidate="TextBox1" ErrorMessage="plese enter email ID"
                    ToolTip="plese Enter email id"
                    ValidationExpression="\w+([-+.]\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-
.])\\w+)*">
</asp:RegularExpressionValidator>

                </td>
            </tr>
            <tr>
                <td class="style2">
                    Name</td>
                <td class="style3">

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

                </td>
                <td>
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
                    ControlToValidate="TextBox2" ErrorMessage="Blank not Allowed"
                    SetFocusOnError="True"></asp:RequiredFieldValidator>

                </td>
            </tr>
            <tr>
                <td class="style2">
                    Age</td>
                <td class="style3">

                    <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>


```

```

        </td>
        <td>
            <asp:RangeValidator ID="RangeValidator1" runat="server"
                ControlToValidate="TextBox3" ErrorMessage="Age should be Between
!8 to 35"
                MaximumValue="35" MinimumValue="18"></asp:RangeValidator>
        </td>
    </tr>
    <tr>
        <td class="style2">
            &nbsp;</td>
        <td class="style3">
            <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Button"
Width="123px" />
        </td>
        <td>
            <asp:ValidationSummary ID="ValidationSummary1" runat="server"
                ShowMessageBox="True" ShowSummary="False" />
        </td>
    </tr>
    <tr>
        <td class="style2">
            &nbsp;</td>
        <td class="style3">
            &nbsp;</td>
        <td>
            &nbsp;</td>
        <td>
            &nbsp;</td>
    </tr>
</table>

</div>
</form>
</body>
</html>

```

### Output::



## Why we use validation controls?

Validation is important part of any web application. User's input must always be validated before sending across different layers of the application.

Validation controls are used to:

- Implement presentation logic.
- To validate user input data.
- Data format, data type and data range is used for validation.

## Validation is of two types:

1. Client Side
2. Server Side

Client side validation is good but we have to be dependent on browser and scripting language support.

Client side validation is considered convenient for users as they get instant feedback. The main advantage is that it prevents a page from being postback to the server until the client validation is executed successfully.

For developer point of view server side is preferable because it will not fail, it is not dependent on browser and scripting language.

You can use ASP.NET validation, which will ensure client, and server validation. It work on both end; first it will work on client validation and then on server validation. At any cost server validation will work always whether client validation is executed or not. So you have a safety of validation check.

For client script .NET used JavaScript. WebUIValidation.js file is used for client validation by .NET

## Validation Controls in ASP.NET

An important aspect of creating ASP.NET Web pages for user input is to be able to check that the information users enter is valid. ASP.NET provides a set of validation controls that provide an easy-to-use but powerful way to check for errors and, if necessary, display messages to the user.

There are six types of validation controls in ASP.NET

1. RequiredFieldValidation Control
2. CompareValidator Control
3. RangeValidator Control
4. RegularExpressionValidator Control
5. CustomValidator Control
6. ValidationSummary

The below table describes the controls and their work:

Validation Control	Description
RequiredFieldValidation	Makes an input control a required field
CompareValidator	Compares the value of one input control to the value of another input control or to a fixed value
RangeValidator	Checks that the user enters a value that falls between two values



RegularExpressionValidator	Ensures that the value of an input control matches a specified pattern
CustomValidator	Allows you to write a method to handle the validation of the value entered
ValidationSummary	Displays a report of all validation errors occurred in a Web page

All validation controls are rendered in form as <span> (label are referred as <span> on client by server)

### Important points for validation controls

- ControlToValidate property is mandatory to all validate controls.
- One validation control will validate only one input control but multiple validate control can be assigned to a input control.

### Validation Properties

Usually, Validation is invoked in response to user actions like clicking submit button or entering data. Suppose, you wish to perform validation on page when user clicks submit button.

Server validation will only performed when CauseValidation is set to true.

When the value of the CausesValidation property is set to true, you can also use the ValidationGroup property to specify the name of the validation group for which the Button control causes validation.

Page has a Validate() method. If it is true this methods is executed. Validate() executes each validation control.

To make this happen, simply set the CauseValidation property to true for submit button as shown below:

```
<asp:Button ID="Button2" runat="server" Text="Submit" CausesValidation=true />
```

Lets understand validation controls one by one with practical demonstration:

### RequiredFieldValidation Control

The RequiredFieldValidator control is simple validation control, which checks to see if the data is entered for the input control. You can have a RequiredFieldValidator control for each form element on which you wish to enforce Mandatory Field rule.

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server" Style="top: 98px;
left: 367px; position: absolute; height: 26px; width: 162px" ErrorMessage="password required"
ControlToValidate="TextBox2"></asp:RequiredFieldValidator>
```

### CompareValidator Control

The CompareValidator control allows you to make comparison to compare data entered in an input control with a constant value or a value in a different control.

It can most commonly be used when you need to confirm password entered by the user at the registration time. The data is always case sensitive.

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" Style="top: 145px;
left: 367px; position: absolute; height: 26px; width: 162px" ErrorMessage="password required"
ControlToValidate="TextBox3"></asp:RequiredFieldValidator>
```

### RangeValidator Control

The RangeValidator Server Control is another validator control, which checks to see if a control value is within a valid range. The attributes that are necessary to this control are: MaximumValue, MinimumValue, and Type.

```
<asp:RangeValidator ID="RangeValidator1" runat="server" Style="top: 194px; left: 365px;
    position: absolute; height: 22px; width: 105px"
    ErrorMessage="Range Validator" ControlToValidate="TextBox4" MaximumValue="100"
    MinimumValue="18" Type="Integer"></asp:RangeValidator>
```

### RegularExpressionValidator Control

A regular expression is a powerful pattern matching language that can be used to identify simple and complex characters sequence that would otherwise require writing code to perform.

Using RegularExpressionValidator server control, you can check a user's input based on a pattern that you define using a regular expression.

It is used to validate complex expressions. These expressions can be phone number, email address, zip code and many more. Using Regular Expression Validator is very simple. Simply set the ValidationExpression property to any type of expression you want and it will validate it.

If you don't find your desired regular expression, you can create your custom one.

In the example I have checked the email id format:

```
<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server" Style="top: 234px;
    left: 366px; position: absolute; height: 22px; width: 177px"
    ErrorMessage="RegularExpressionValidator" ControlToValidate="TextBox5"
    ValidationExpression="\w+([-+.] \w+)*@\w+([-.] \w+)*\.\w+([-
    .] \w+)*"></asp:RegularExpressionValidator>
```

The complete code for the above 4 controls is as:

### Default.aspx Design

Enter your name:	<input type="text"/>	name is mandatory	
Password	<input type="password"/>	password required	
Confirm Password	<input type="password"/>	password required	CompareValidator
Enter your age:	<input type="text"/>	RangeValidator	
Enter your email id:	<input type="text"/>	RegularExpressionValidator	

### Default.aspx Source code

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
```

```

</head>
<body>
  <form id="form1" runat="server">
    <div>
      <asp:Label ID="Label3" runat="server" Style="top: 241px; left: 70px; position: absolute;
        height: 22px; width: 128px; bottom: 282px;" Text="Enter your email id:"></asp:Label>
      <asp:Label ID="Label1" runat="server" Style="top: 54px; left: 74px; position: absolute;
        height: 22px; width: 128px" Text="Enter your name:"></asp:Label>
      <asp:TextBox ID="TextBox1" runat="server" Style="top: 54px; left: 221px; position: absolute;
        height: 22px; width: 128px; right: 396px;"></asp:TextBox>
      <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" Style="top: 56px;
        left: 378px; position: absolute; height: 22px; width: 128px" ErrorMessage="RequiredFieldValidator"
        ControlToValidate="TextBox1">name is
        mandatory </asp:RequiredFieldValidator>
    </div>
    <p>
      <asp:Button ID="Button1" runat="server" Style="top: 311px; left: 267px; position: absolute;
        height: 26px; width: 61px" Text="Submit" />
    </p>
    <asp:TextBox ID="TextBox3" runat="server" Style="top: 145px; left: 217px; position: absolute;
      height: 22px; width: 131px" TextMode="Password"></asp:TextBox>
    <p>
      <asp:TextBox ID="TextBox2" runat="server" Style="top: 101px; left: 218px; position: absolute;
        height: 22px; width: 131px" TextMode="Password"></asp:TextBox>
      <asp:Label ID="Label4" runat="server" Style="top: 105px; left: 74px; position: absolute;
        height: 22px; width: 128px" Text="Password"></asp:Label>
      <asp:TextBox ID="TextBox5" runat="server" Style="top: 239px; left: 210px; position: absolute;
        height: 22px; width: 134px"></asp:TextBox>
    </p>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server" Style="top: 98px;
      left: 367px; position: absolute; height: 26px; width: 162px" ErrorMessage="password required"
      ControlToValidate="TextBox2"></asp:RequiredFieldValidator>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" Style="top: 145px;
      left: 367px; position: absolute; height: 26px; width: 162px" ErrorMessage="password required"
      ControlToValidate="TextBox3"></asp:RequiredFieldValidator>
    <asp:CompareValidator ID="CompareValidator1" runat="server" Style="top: 149px; left: 512px;
      position: absolute; height: 26px; width: 162px" ErrorMessage="CompareValidator"
      ControlToValidate="TextBox3" ValueToCompare="hello"></asp:CompareValidator>
    <p>
      <asp:Label ID="Label5" runat="server" Style="top: 148px; left: 71px; position: absolute;
        height: 22px; width: 128px; bottom: 375px;" Text="Confirm Password"></asp:Label>
      <asp:TextBox ID="TextBox4" runat="server" Style="top: 194px; left: 212px; position: absolute;
        height: 22px; width: 140px"></asp:TextBox>
      <asp:Label ID="Label6" runat="server" Style="top: 194px; left: 71px; position: absolute;
        height: 22px; width: 128px; bottom: 329px;" Text="Enter your age:"></asp:Label>
    </p>
    <asp:RangeValidator ID="RangeValidator1" runat="server" Style="top: 194px; left: 365px;
      position: absolute; height: 22px; width: 105px" ErrorMessage="RangeValidator"
      ControlToValidate="TextBox4" MaximumValue="100" MinimumValue="18"
      Type="Integer"></asp:RangeValidator>
    <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server" Style="top: 234px;
      left: 366px; position: absolute; height: 22px; width: 177px"
      ErrorMessage="RegularExpressionValidator" ControlToValidate="TextBox5"
      ValidationExpression="\w+([-+.] \w+)*@\w+([-.] \w+)*\.\w+([-
      .] \w+)*"></asp:RegularExpressionValidator>
  </form>
</body>
</html>

```

### CustomValidator Control

You can solve your purpose with ASP.NET validation control. But if you still don't find solution you can create

your own custom validator control.

The CustomValidator Control can be used on client side and server side. JavaScript is used to do client validation and you can use any .NET language to do server side validation.

I will explain you CustomValidator using server side. You should rely more on server side validation.

To write CustomValidator on server side you override ServerValidate event.

User ID:  User id required    User ID should have atleast a capital, small and digit and should be greater than 5 and less than 26 letters

### Source Code

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server" Text="User ID:"></asp:Label>
            &nbsp;<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            &nbsp;<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
                ControlToValidate="TextBox1" ErrorMessage="User id
required"></asp:RequiredFieldValidator>&nbsp;
            &nbsp;&nbsp;<asp:CustomValidator ID="CustomValidator1" runat="server" OnServerValidate="UserCustomValidate"
                ControlToValidate="TextBox1"
                ErrorMessage="User ID should have atleast a capital, small and digit and should be greater than 5 and
less
than 26 letters"
                SetFocusOnError="True"></asp:CustomValidator>
        </div>
        <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Submit" />
    </form>
</body>
</html>
```

### Code behind file

```
using System;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

public partial class _Default : System.Web.UI.Page
```

```

{
protected void UserCustomValidate(object source, ServerValidateEventArgs args)
{
    string str = args.Value;
    args.IsValid = false;
    //checking for input length greater than 6 and less than 25 characters
    if (str.Length < 6 || str.Length > 25)
    {
        return;
    }
    //checking for a atleast a single capital letter
    bool capital = false;
    foreach (char ch in str)
    {
        if (ch >= 'A' && ch <= 'Z')
        {
            capital = true;
            break;
        }
    }
    if (!capital)
    {
        return;
    }
    //checking for a atleast a single lower letter
    bool lower = false;
    foreach (char ch in str)
    {
        if (ch >= 'a' && ch <= 'z')
        {
            lower = true;
            break;
        }
    }
    if (!lower)
    {
        return;
    }
    bool digit = false;
    foreach (char ch in str)
    {
        if (ch >= '0' && ch <= '9')
        {
            digit = true;
            break;
        }
    }
    if (!digit)
    {
        return;
    }
    args.IsValid = true;
}
protected void Page_Load(object sender, EventArgs e)
{
}
protected void Button1_Click(object sender, EventArgs e)
{
}
}

```

User ID:

User ID should have atleast a capital, small and digit and sh

The server side validation you write does not need to provide the exact same validation as that of the client side validation. The client side validation can check for the user input data for range and type and server side validation can check for matching of data with database. Both server side and client side validation can be used for total solution.

### ValidationSummary

ASP.NET has provided an additional control that complements the validator controls.

The ValidationSummary control is reporting control, which is used by the other validation controls on a page.

You can use this validation control to consolidate errors reporting for all the validation errors that occur on a page instead of leaving this up to each and every individual validation control.

The validation summary control will collect all the error messages of all the non-valid controls and put them in a tidy list.

```
<asp:ValidationSummary ID="ValidationSummary1" runat="server" style="top: 390px; left: 44px; position: absolute; height: 38px; width: 625px" />
```

Both ErrorMessage and Text properties are used to display error messages. Text error message have precedence.

If you are using ValidationSummary than only ErrorMessage and Text property is used.

The complete code for the above ValidationSummary is as:

### Default.aspx Design

Enter your name:  \*

---

Enter Password:  \*

Confirm Password:  \* \*

Enter your Age:  \*

- Error message 1.
- Error message 2.

### Default.aspx Source code

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server" Style="top: 239px; left: 75px; position: absolute;
                height: 22px; width: 128px" Text="Enter your Age:"></asp:Label>
            &nbsp;
            <asp:Label ID="Label2" runat="server" Style="top: 94px; left: 81px; position: absolute;
                height: 22px; width: 128px" Text="Enter your name:"></asp:Label>
        </div>
        <asp:TextBox ID="TextBox1" runat="server" Style="top: 95px; left: 250px; position: absolute;
            height: 22px; width: 128px"></asp:TextBox>
        <p>
            <asp:TextBox ID="TextBox4" runat="server" Style="top: 195px; left: 249px; position: absolute;
                height: 22px; width: 127px"></asp:TextBox>
        </p>
        <p>
            <asp:Label ID="Label3" runat="server" Style="top: 148px; left: 76px; position: absolute;
                height: 22px; width: 128px" Text="Enter Password:"></asp:Label>
        </p>
        <p>
            <asp:TextBox ID="TextBox3" runat="server" Style="top: 146px; left: 249px; position: absolute;
                height: 22px; width: 127px" TextMode="Password"></asp:TextBox>
        </p>
        <p>
            <asp:Label ID="Label4" runat="server" Style="top: 197px; left: 75px; position: absolute;
                height: 22px; width: 128px" Text="Confirm Password:"></asp:Label>
        </p>
        <asp:TextBox ID="TextBox2" runat="server" Style="top: 236px; left: 250px; position: absolute;
            height: 22px; width: 127px" TextMode="Password"></asp:TextBox>
        <asp:CompareValidator ID="CompareValidator1" runat="server" Style="top: 197px; left: 522px;
            position: absolute; height: 22px; width: 17px" ErrorMessage="CompareValidator"
            ControlToCompare="TextBox2" ControlToValidate="TextBox3">*</asp:CompareValidator>
        <p>
            <asp:Button ID="Button1" runat="server" Style="top: 333px; left: 248px; position: absolute;
                height: 26px; width: 56px" Text="Submit" />
            <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" Style="top: 196px;
                left: 393px; position: absolute; height: 22px; width: 22px" ErrorMessage="Confirm Password mandatory
            &amp; should match password"
                ControlToValidate="TextBox3">*</asp:RequiredFieldValidator>
            <asp:RangeValidator ID="RangeValidator1" runat="server" Style="top: 235px; left: 388px;
                position: absolute; height: 22px; width: 156px; bottom: 288px;" ErrorMessage="age between 18-100"
                ControlToValidate="TextBox4" MaximumValue="100" MinimumValue="18"
                Type="Integer">*</asp:RangeValidator>
            <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" Style="top: 92px;
                left: 393px; position: absolute; height: 22px; width: 156px" ErrorMessage="Name is required"
                ControlToValidate="TextBox1">*</asp:RequiredFieldValidator>
            <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server" Style="top: 146px;
                left: 391px; position: absolute; height: 22px; width: 156px" ErrorMessage="Password mandatory"
                ControlToValidate="TextBox2">*</asp:RequiredFieldValidator>
        </p>
        <asp:ValidationSummary ID="ValidationSummary1" runat="server" Style="top: 390px;
            left: 44px; position: absolute; height: 38px; width: 625px" />
    </form>
</body>
</html>

```

## Output of ValidationSummary program

Enter your name:	<input type="text"/>	*
Enter Password:	<input type="password"/>	*
Confirm Password:	<input type="password"/>	*
Enter your Age:	<input type="text"/>	

- Confirm Password mandatory & should match password
- Name is required
- Password mandatory

## Login Control in Asp.net

Login Controls are used to create features like user can login, change the password, create new user, password recovery etc. Such controls are very useful in web applications.

- **Pointer:** It is just a pointer. If we drag any other control on form it causes to create that control on form but pointer does not create any control on form. In other word we can say, we select it for to ignore any other selected control.
- **ChangePassword:** This control is used to provide the user a function to change the existing password. In the process of changing password user have to provide his old password and then new password. We can use functionality to this control to send the user password change confirmation by email.
- **CreateUserWizard:** This control is used to create new user. We can add additional steps to collect more information from user. We also can add functionality to this control to send the user email containing his username and password.
- **Login:** This control is used to provide the user functionality to login in website. This control contains two textboxes for username and password; one checkbox control for to remember password and one button.
- **LoginName:** This control is used to display the current logged-in/authenticated user name on web page. We can add the functionality to this control to display "Guest" when no one is authenticated to website.



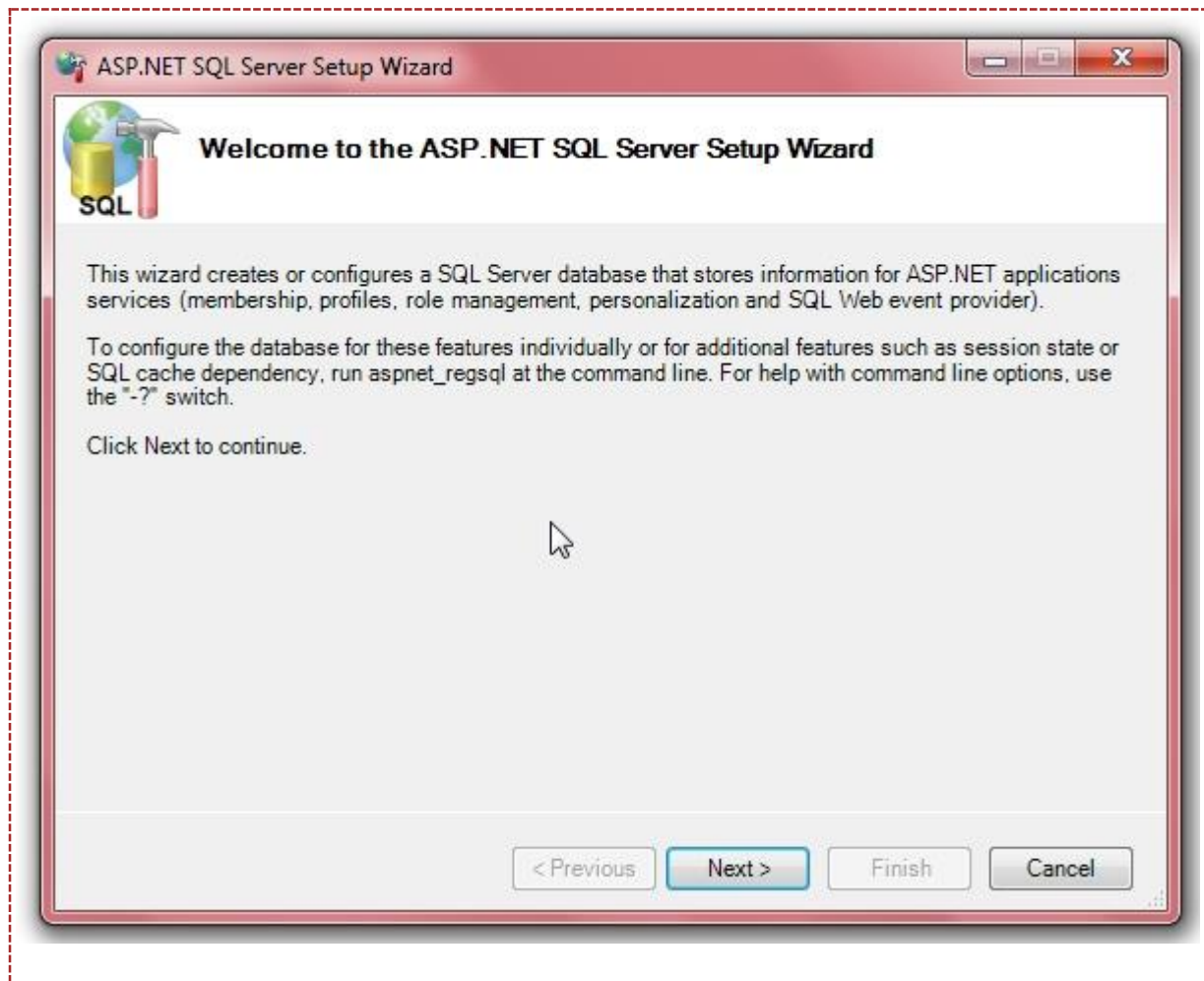
- **LoginStatus:** This control is used to display the information that user is currently authenticated or not. If user is authenticated then it will display a link like "Logout" and if not then "Login".
- **LoginView:** This control is used to display the user's authentication status and roles.
- **PasswordRecovery:** This control provides the functionality to recover the forgotten password. We can also add function to send the email containing password to user when user request for password.

In one of my project I got requirement like using asp.net membership concept with custom database. First of all we learn what is asp.net membership? And why we need to use this one?

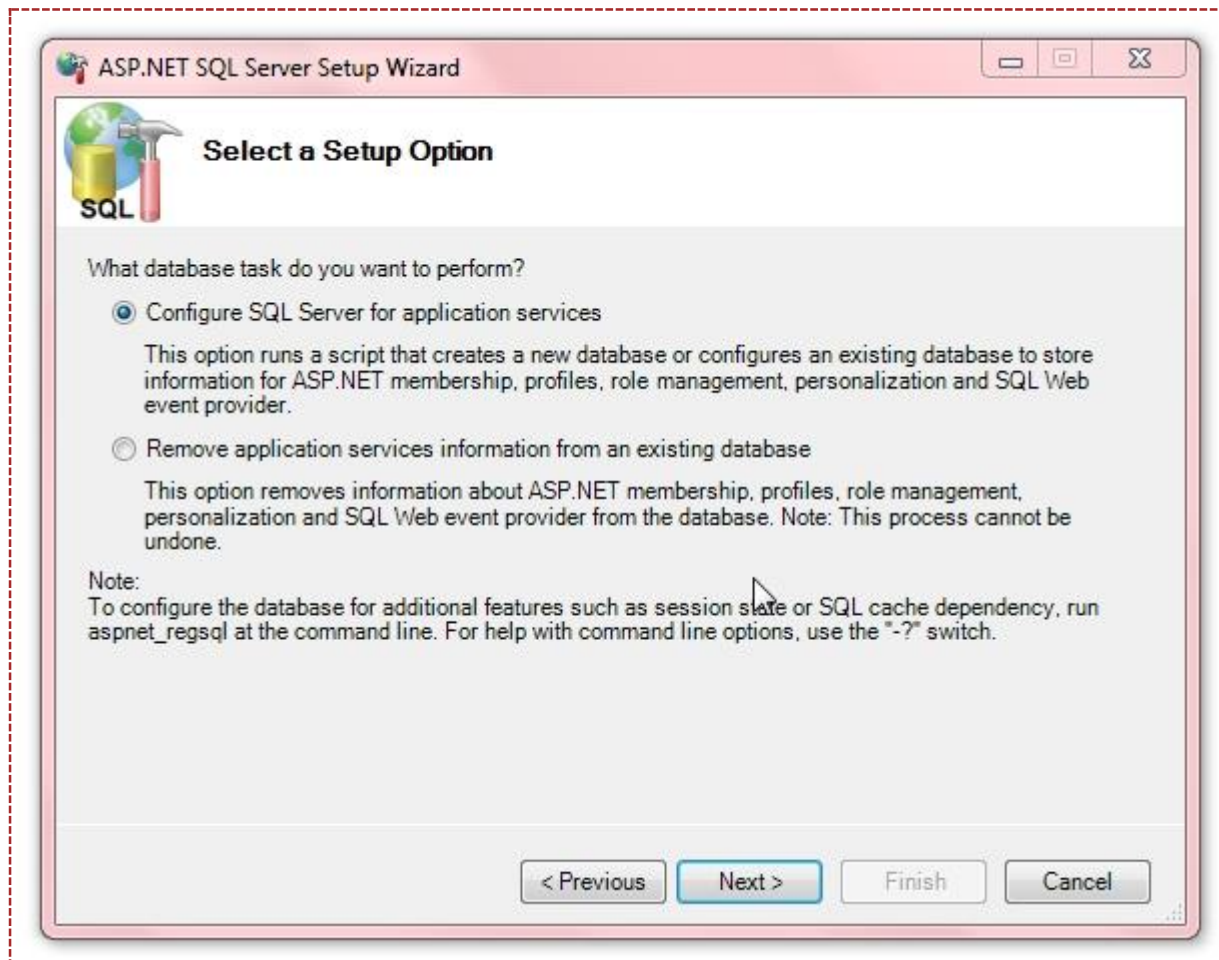
ASP.NET membership gives you a built-in way to validate and store user credentials. ASP.NET membership therefore helps you manage user authentication in your Web sites. You can use ASP.NET membership with ASP.NET Forms authentication or with the ASP.NET login controls to create a complete system for authenticating users. By Using ASP.NET membership we can create new users and passwords and we can authenticate users who visit your site. You can authenticate users programmatically, or you can use the ASP.NET login controls to create a complete authentication system that requires little or no code and we can manage passwords, which includes creating, changing, and resetting them.

Now we will do the step by step process how to use Custom membership database.

1. Go to the Start menu and open Run and enter **%WINDIR%\Microsoft.Net\Framework\v2.0.50727\aspnet\_regsql.exe** and click ok it will open the Asp.net SQL Server setup wizard like this



We can get this in alternatively go to  
**C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\aspnet\_regsql.exe**  
 After that click next you will get window like this



In this wizard we have two options

1. Configure SQL Server for application Services
2. Remove application services information from an existing database.

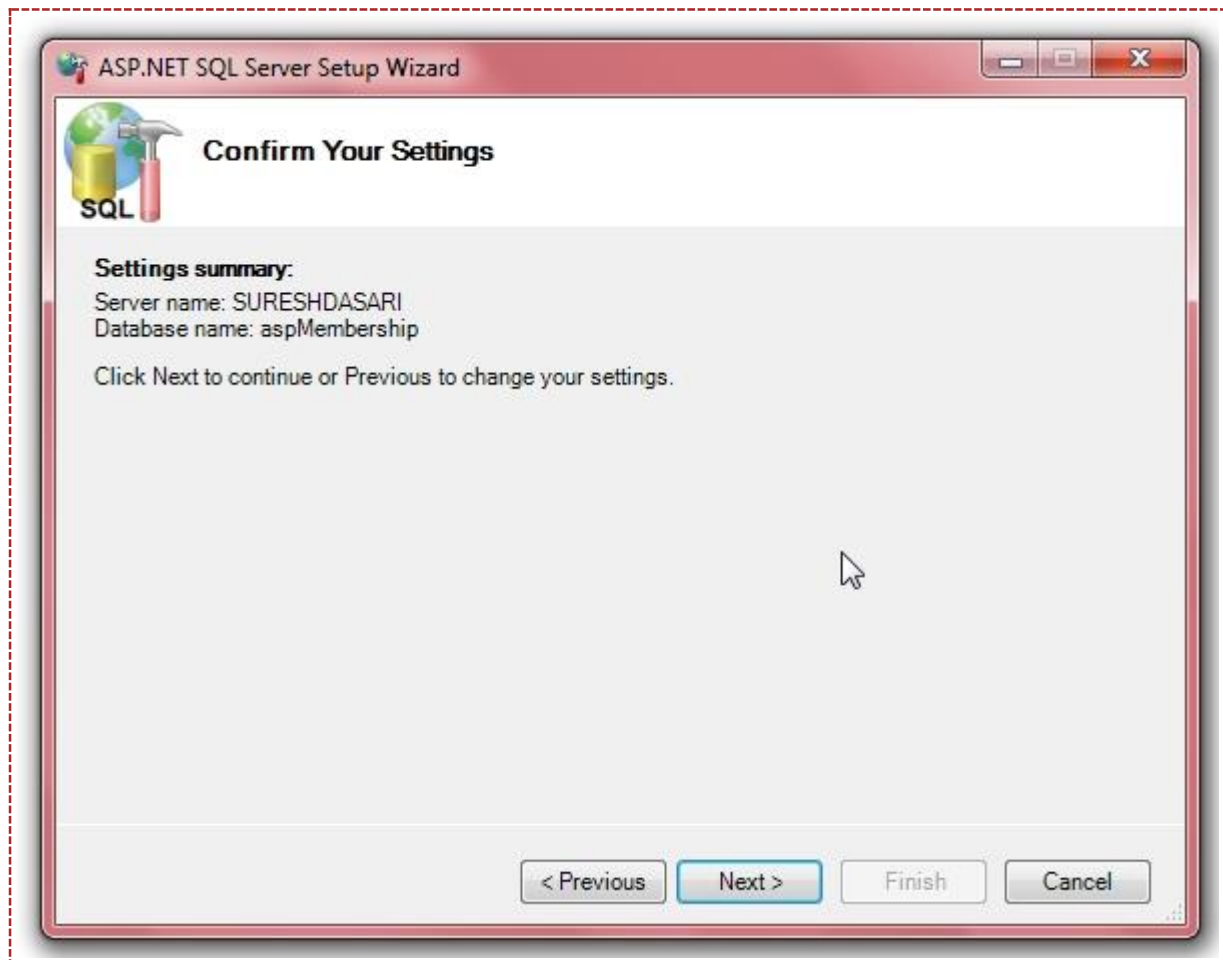
Now select first option Configure SQL Server for application services because first time we are creating database. If you want to delete already created database select second option

After selecting the first option click next now the window like this

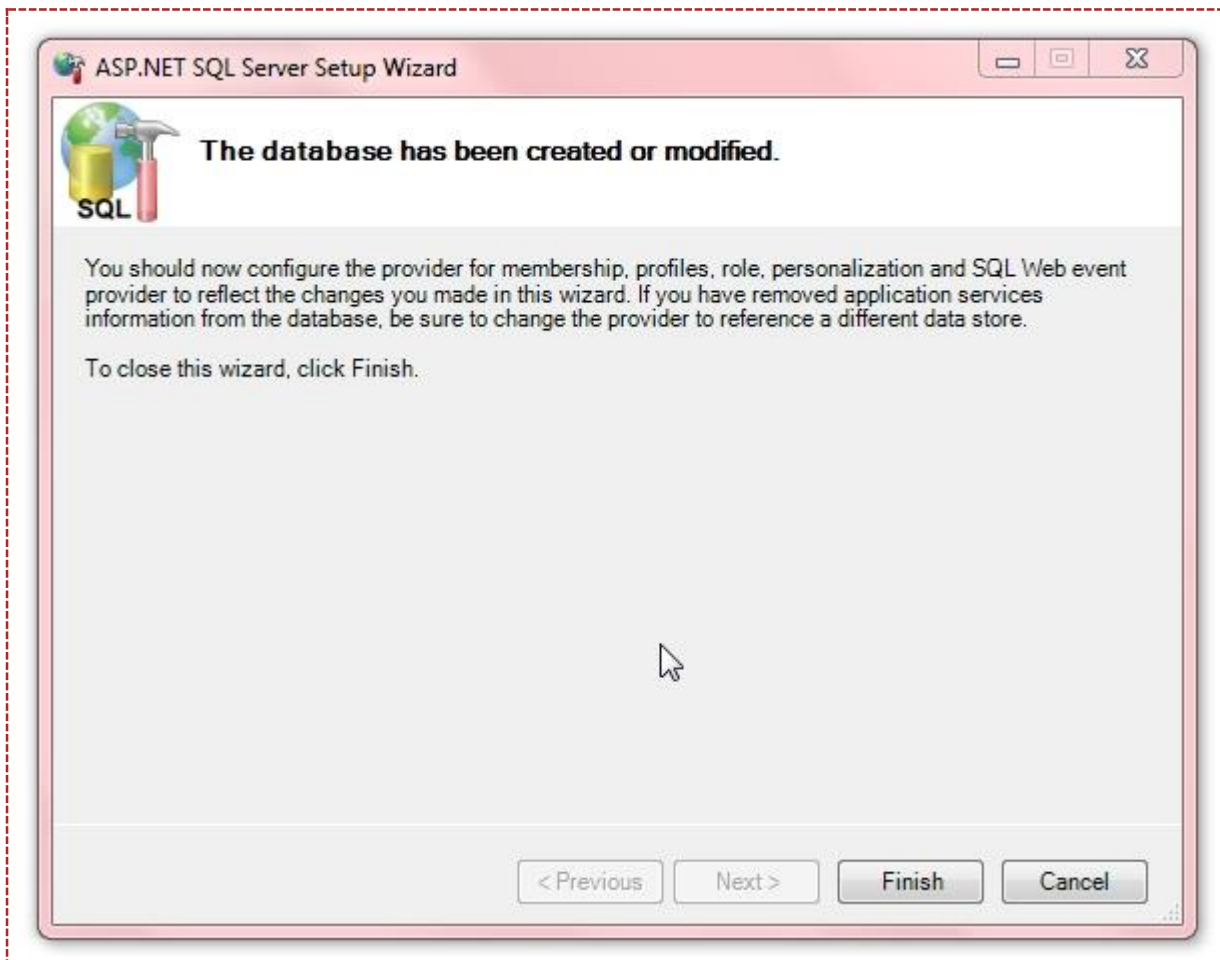


The image shows a screenshot of the 'ASP.NET SQL Server Setup Wizard' window, specifically the 'Select the Server and Database' step. The window has a title bar with the text 'ASP.NET SQL Server Setup Wizard' and standard Windows window controls. Below the title bar is a header area with an SQL logo (a globe and a hammer) and the text 'SQL'. The main content area contains instructions: 'Specify the SQL Server name, database name to create or remove, and the credentials to use when connecting to the database.' A 'Note' states: 'The credentials must identify a user account that has permissions to create or remove a database.' The form includes a 'Server:' text box with 'SURESHDASARI' entered. Below it are two radio buttons: 'Windows authentication' (selected) and 'SQL Server authentication'. Under 'SQL Server authentication' are 'User name:' and 'Password:' text boxes. At the bottom of the form is a 'Database:' dropdown menu with 'aspMembership' selected. At the bottom of the window are four buttons: '< Previous', 'Next >' (highlighted in blue), 'Finish', and 'Cancel'.

After that enter your database server name and select database from existing list or give new name and click next you will see the window like this



After that click next button you now your aspnet membership database installation is completed and now that is ready to use with our application final window like this



Now check your database everything installed and ready to use in our application.

Everything is ready how to use that custom database in our application for that create one new web application and name it as Registration.aspx now drag and drop the one new control **CreateUserWizard** from Login Controls list that aspx page like this

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
<title>Untitled Page</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:CreateUserWizard ID="CreateUserWizard1" runat="server">
<WizardSteps>
<asp:CreateUserWizardStep ID="CreateUserWizardStep1" runat="server">
</asp:CreateUserWizardStep>
<asp:CompleteWizardStep ID="CompleteWizardStep1" runat="server">
</asp:CompleteWizardStep>
</WizardSteps>
</asp:CreateUserWizard>
</div>
</form>
</body>
</html>
```

Now open Web.config file and write the following code

Write the **connectionstring** like this

```
<connectionStrings>
<add name="Connection" connectionString="Data Source=SureshDasari;Integrated
Security=true;Initial Catalog=AspMembership"
providerName="System.Data.SqlClient"/>
</connectionStrings>
```

After that write the following code in **system.web** section

```
<membership>
<providers>
<clear/>
<add name="AspNetSqlMembershipProvider"
type="System.Web.Security.SqlMembershipProvider"
connectionStringName="Connection" applicationName="SampleApplication"/>
</providers>
</membership>
<profile>
<providers>
<clear/>
<add name="AspNetSqlProfileProvider" type="System.Web.Profile.SqlProfileProvider"
connectionStringName="Connection" applicationName="SampleApplication"/>
</providers>
</profile>
<roleManager enabled="false">
<providers>
<clear/>
<add name="AspNetSqlRoleProvider" type="System.Web.Security.SqlRoleProvider"
connectionStringName="Connection" applicationName="SampleApplication"/>
</providers>
</roleManager>
```

After that run your application and create user by using **CreateUserWizard**. Here functionality of inserting the user details and everything will take care by **CreateUserWizard**. No need to write any coding just set the database and set the conditions in web.config file that is enough. Now your program will work perfectly.

In previous post I explained clearly [how to install asp.net membership database schema in SQL Server](#) and [create users using CreateUserWizard in asp.net membership](#). Here I am using these two concepts to validate user details with Login Control using asp.net membership in our login page.

In visual studio toolbox we have Login control in Login list section. This control contains built-in functionalities like validate username, password and check the user details against asp.net membership. We can use this control directly in login page instead of writing huge code in your login page with username, password textbox controls etc with asp.net membership database.

To install asp.net membership database schema in sql server check this post [install membership database in SQL Server](#) and to create user accounts check this post [user accounts with CreateUserWizard](#).

After completion of database setup and user accounts creation open visual studio and create new web application. After that right-click on your application and Add new item, select WebForm and give name as **Login.aspx** (If you want another name you can give another name also that is your wish). Now drag and drop the new control **Login** from Login Controls section into **Login.aspx** page. After that check your.aspx page code that would be like this:

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>Login Control to validate User Details</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:Login ID="Login1" runat="server" DestinationPageUrl="~/Default.aspx">
</asp:Login>
</div>
</form>
</body>
</html>

```

Now open **Default.aspx** page drag and drop the Login status control and one label to our **Default.aspx** page will be like this

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>Login Control to validate User Details</title>
</head>
<body>
<form id="form1" runat="server">
<table align="center" width="200px">
<tr>
<td>
</td>
<td>
<asp:LoginStatus ID="LoginStatus1" runat="server" />
</td>
</tr>

```



```

<tr>
<td><asp:Label ID="lblResult" runat="server" Font-Bold="true"/></td>
<td></td>
</tr>
</table>
</form>
</body>
</html>

```

Now write the following code in **Default.aspx** code behind to get login username

```

protected void Page_Load(object sender, EventArgs e)
{
    lblResult.Text = "Welcome " + Page.User.Identity.Name;
}

```

Now open **Web.config** file and change authentication mode to **Forms** and deny permissions for anonymous access

```

<authentication mode="Forms"/>
<authorization>
<deny users="?"/>
</authorization>

```

After that set database connection settings in web.config

First set the **connectionstring** like this

```

<connectionStrings>
<add name="dbconnection" connectionString="Data Source=SureshDasari;Initial
Catalog=AspMembership;Integrated Security=true"
providerName="System.Data.SqlClient"/>
</connectionStrings>

```

After that write the following code in **system.web** section

```

<membership>
<providers>

```

```

<clear/>

<add name="AspNetSqlMembershipProvider"
type="System.Web.Security.SqlMembershipProvider"
connectionStringName="dbconnection" applicationName="SampleApplication"/>

</providers>

</membership>

<profile>

<providers>

<clear/>

<add name="AspNetSqlProfileProvider" type="System.Web.Profile.SqlProfileProvider"
connectionStringName="dbconnection" applicationName="SampleApplication"/>

</providers>

</profile>

<roleManager enabled="false">

<providers>

<clear/>

<add name="AspNetSqlRoleProvider" type="System.Web.Security.SqlRoleProvider"
connectionStringName="dbconnection" applicationName="SampleApplication"/>

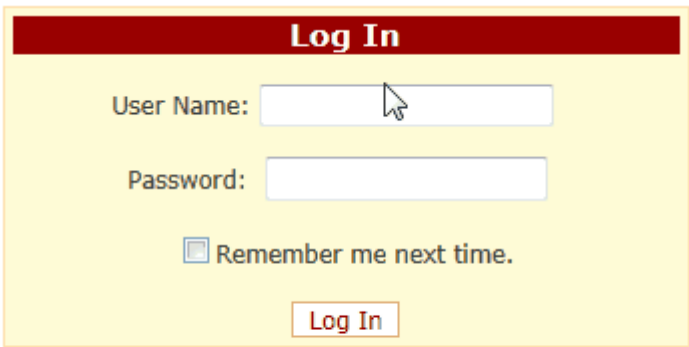
</providers>

</roleManager>

```

Now run your application and check user credentials with asp.net membership database

### **Demo**



In previous post I explained clearly [how to install asp.net membership database schema in SQL Server](#) . After install asp.net membership database check your database everything installed and ready to use in our application or not.

Now open visual studio and create new web application after that drag and drop the new control **CreateUserWizard** from Login Controls section into aspx page and check the aspx page code that would be like this

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>Create Users Using CreateUserWizard Control</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:CreateUserWizard ID="CreateUserWizard1" runat="server">
<WizardSteps>
<asp:CreateUserWizardStep ID="CreateUserWizardStep1" runat="server">
</asp:CreateUserWizardStep>
<asp:CompleteWizardStep ID="CompleteWizardStep1" runat="server">
</asp:CompleteWizardStep>
</WizardSteps>
</asp:CreateUserWizard>
</div>
</form>
</body>
</html>
```

Now open Web.config file and set database connection settings in web.config

First set the **connectionstring** like this

```
<connectionStrings>
<add name="dbconnection" connectionString="Data Source=SureshDasari;Initial
Catalog=AspMembership;Integrated Security=true"
providerName="System.Data.SqlClient"/>
</connectionStrings>
```

After that write the following code in **system.web** section

```

<membership>
<providers>
<clear/>
<add name="AspNetSqlMembershipProvider"
type="System.Web.Security.SqlMembershipProvider"
connectionStringName="dbconnection" applicationName="SampleApplication"/>
</providers>
</membership>
<profile>
<providers>
<clear/>
<add name="AspNetSqlProfileProvider" type="System.Web.Profile.SqlProfileProvider"
connectionStringName="dbconnection" applicationName="SampleApplication"/>
</providers>
</profile>
<roleManager enabled="false">
<providers>
<clear/>
<add name="AspNetSqlRoleProvider" type="System.Web.Security.SqlRoleProvider"
connectionStringName="dbconnection" applicationName="SampleApplication"/>
</providers>
</roleManager>

```

After that run your application and create users by using **CreateUserWizard** control.

Here the functionality of insert user details and everything will take care by **CreatUserWizard** no need to write any coding just set the database and set the conditions in web.config file that is enough it will work perfectly.

During the time of create user if you get error like "Password length minimum: 7. Non-alphanumeric characters required: 1." check this post [solve password length minimum: 7 problem](#). This problem because of our password basically asp.net membership concept will accept the password only if it contains mixed of alphanumeric and special characters.

## Demo





If you observe above demo in that we need to enter different options like UserName, password (strong password like mixed of alphanumeric and special characters), Email (Unique Email), Security Question and answer to register new user.

If we don't want security question and answer options and we need to allow users to enter his own password without any restrictions for that we need to configure some properties in web.config file like `requiresQuestionAndAnswer="false"`, `minRequiredPasswordLength="6"` and `minRequiredNonalphanumericCharacters="0"` to set those properties check this post [solve password length problem](#).

## Web User control in Asp.net

- ASP.NET allows you to create new controls according to your requirements. The controls that you create are called ASP.NET User Controls. ASP.NET User Controls have .ascx extension. The biggest advantage of ASP.NET User Controls is that it allows you to save a part of the web form and reuse it many other web forms. This drastically reduces the developers' time.
- ASP.NET User Controls are self-contained entities that are saved in an independent file. You can relate a user control with a "black box". ASP.NET User Controls are very helpful when you want to use functionality on multiple pages of a web-application.
- As ASP.NET User Controls are self-contained the developers working on the same project need not worry about transgressing on other's code. ASP.NET User Controls offer a great deal of functionality than Server-side Includes (SSIs) . Using SSIs in encapsulating the site functionality is a quite tedious task when compared to ASP.NET User Controls. You can create a user control by using the Register directive. This process is called registering the user control. For instance, you can use this code to register a user control:  

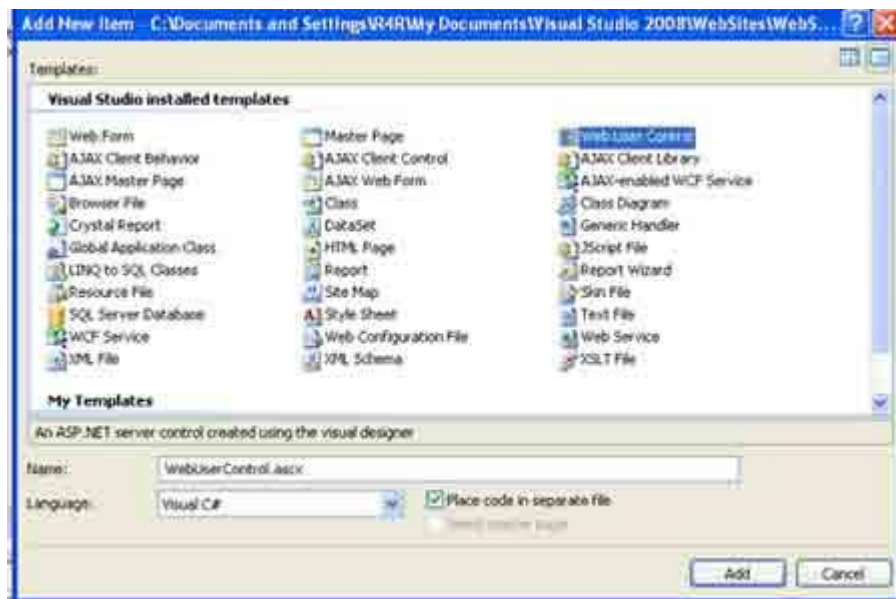
```
<%@ Register TagPrefix="sampNamespace" TagName="Myname"
Src="mypage.ascx" %>
```
- The TagPrefix is the exclusive namespace provided to the user control so that multiple ASP.NET User Controls with similar name can be discriminated from each other. The TagName is the name of the user control and the Src is the virtual path to the user control. After registering the user control you can place the tag of the user control in an appropriate location on the web form. You have to include the `runat="server"` attribute with the tag.

### User controls have the following similarities with normal ASPX pages:

- They have a markup section where you can add standard markup and server controls.
- They can be created and designed with Visual Web Developer in Markup, Design, and Split View.
- They can contain programming logic, either inline or with a Code Behind file.
- You have access to page-based information like `Request.QueryString`.
- They raise some (but not all) of the events that the Page class raises, including `Init`, `Load`, and `PreRender`.

### Creating User Controls:

User controls are added to the site like any other content type: through the Add New Item dialog box. Similar to pages, you get the option to choose the programming language and whether you want to place the code in a separate Code Behind file



**Now Add following code to file .ascx**

```
<%@ Page Language="C#" AutoEventWireup="true"
Codevirtual="windows_level_authntication.aspx.cs"
Inherits="windows_level_authntication" %>

<%@ Register src="WebUserControl.ascx" tagname="WebUserControl"
tagprefix="uc1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Untitled Page</title>
  <style type="text/css">
    .style1
    {
      width: 100%;
      border: 4px solid #00FFFF;
      background-color: #FFCC99;
    }
    .style2
    {
      height: 242px;
    }
    .style3
    {
      height: 231px;
    }
    .style8
    {
      width: 213px;
```

```

    }
    .style9
    {
        height: 664px;
        width: 213px;
    }
</style>
</head>
<body>
    <form id="form1" runat="server">
    <div>

        <table class="style1">
            <tr>
                <td class="style8">
                    <uc1:WebUserControl ID="WebUserControl1"
runat="server" />
                </td>
                <td class="style3">
                    <asp:Login ID="Login1" runat="server" BackColor="#FFFBD6"
BorderColor="#FFDFAD"
                    BorderPadding="4" BorderStyle="Solid"
                    BorderWidth="1px" Font-Names="Verdana"
                    Font-Size="0.8em" ForeColor="#333333"
                    OnAuthenticate="Login1_Authenticate"
                    TextLayout="TextOnTop">
                        <TextBoxStyle Font-Size="0.8em" />
                        <LoginButtonStyle BackColor="White"
                    BorderColor="#CC9966" BorderStyle="Solid"
                    BorderWidth="1px" Font-Names="Verdana" Font-
                    Size="0.8em" ForeColor="#990000" />
                        <InstructionTextStyle Font-Italic="True"
                    ForeColor="Black" />
                        <TitleTextStyle BackColor="#990000" Font-Bold="True"
                    Font-Size="0.9em"
                        ForeColor="White" />
                    </asp:Login>
                </td>
            </tr>
            <tr>
                <td class="style3">
                </td>
            </tr>
            <tr>
                <td class="style9">
                    &nbsp;   </td>

                <td class="style2">
                </td>
            </tr>
        </table>

    </div>
</form>
</body>
</html>

```

## Design Of User Control



## .CS CODE OF WEB USER CONTROL

```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
    TextBox2.Text = Calendar1.SelectedDate.ToLongDateString();
}
protected void TextBox3_TextChanged(object sender, EventArgs e)
{
    Literal1.Text = "<h1><marquee>Welcome to Mr..'" + TextBox3.Text +
    "' </marquee> </h1> ";
}
```

## How to Call User Control in .aspx Page:

```
<%@ Register src="WebUserControl.ascx" tagname="WebUserControl"
tagprefix="uc1" %>
```

Output of User Control:





## Master Page

### Introduction Of Master Page

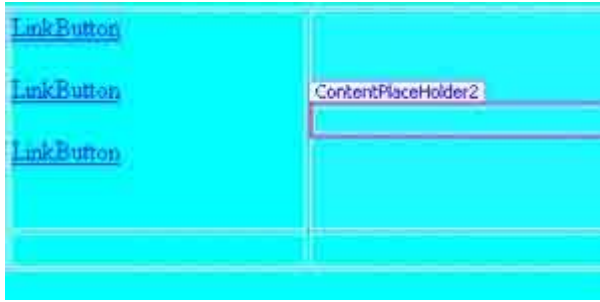
Microsoft ASP.NET 2.0 introduces a new feature .Master Pages. that permits you to define common user interface (UI) elements and common structure in a template that can be applied to multiple content pages in a Web application. Standard elements are, for example, navigational menus, logos, and other stock graphical elements. This allows you to have a common appearance and functionality across your Web site, avoids duplication of code, and makes the site easier to maintain. If the content pages are based on the master page, all the elements defined in the master page would automatically be defined in the content pages.

The page layout for Master Pages is easy to create, simple to maintain, and simple to assign to a Web application. The Master page provides a single point of reference for all pages to display standardized Web content. Master pages are completely transparent to end users and permit developers to create Web sites where pages share a common layout. Defining a Master page is similar to defining a normal page, but saving a Master page is different from saving a normal page. You must save the Master pages by using the .master file extension.

### How to Add Master Page in Your Application

Adding a master page to your web application is straight forward. Just right click on the project and select "Add New Item" and then select "Master Page". This will add the master page to your project.

The master page already has the content place holder control which is used to hold and display your contents. Let's delete that content placeholder and add it by our self. In this case we will create two content place holders. One will be on the left and other one on the right .After you insert the content placeholder control inside your table your master page will look something like this:



### Using the master page in your aspx pages

Just add a new aspx page and name it as "first.aspx". Now you want to use the Sample1.master file in your aspx page. Just go to the html view of your page and add a Master Page File attribute in the page directive and delete all the other html that is written in the aspx page. The Master Page File attribute denotes that the Page is inheriting from the master page.

```
<%@ Page Master Page virtual="~/Sample1.master" %>
```

## Query String In ASP.NET

**Query string is used to Pass the values or information form one page to another page.**

The **query string** is a holdover from the ASP days of web programming. You will see this a lot when you are surfing around the internet. Basically, it is passing information to the next page with the URL. You can use a query string to submit data back to your page or to another page through the URL. Query strings provide a simple but limited way of maintaining some state information.

### The advantages of using query strings are

- No server resources required. The query string is contained in the HTTP request for a specific URL.
- Broad support. Almost all browsers and client devices support passing values in a query string
- it is very easy

### The disadvantages of using query strings are

- Security. The information in the query string is directly visible to the user via the browser user interface. The query values are exposed to the Internet via the URL so in some cases security may be an issue.
- Limited capacity. Most browsers and client devices impose a 255-character limit on URL length

### .CS CODE

```
private void btnSubmit_Click(object sender, EventArgs e)
{
    Response.Redirect("Default.aspx?Name=" +
    this.txtName.Text + "&LastName=" +
    this.txtLastName.Text);
}
```

```
}
```

Our first code part builds a query string for your application and send contents of your textboxes to second page. Now how to retrieve this values from second page.

Put this code to second page page\_load.

```
private void Page_Load(object sender, EventArgs e)
{
    this.txtBox1.Text = Request.QueryString["Name"];
    this.txtBox2.Text = Request.QueryString["LastName"];
}
```

## State Management in ASP.NET

State management is the process by which you maintain state and page information over multiple requests for the same or different pages.

### Types of State Management

#### Client – Side State Management

ASP.NET provides various client side state management options like Cookies, Query Strings (URL), Hidden fields, View State and Control state (ASP.NET 2.0). This stores information on the client's computer by embedding the information into a Web page, a uniform resource locator (url), or a cookie. The techniques available to store the state information at the client computer.

a. View State – Asp .Net uses View State to track the values in the Controls. You can add custom values to the view state. It is used by the Asp .net page framework to automatically save the values of the page and of each control just prior to rendering to the page. When the page is posted, one of the first tasks performed by page processing is to restore view state.

### Example of View state

#### Write code in .aspx page

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeVirtual="Default.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>ViewState</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:TextBox runat="server" id="NameField" />
        <asp:Button runat="server" id="SubmitForm"
onclick="SubmitForm_Click"
```

```

text="Submit & set name" />
<asp:Button runat="server" id="RefreshPage" text="Just
submit" />
<br /><br />
Name retrieved from ViewState: <asp:Label runat="server"
id="NameLabel" />
</form>
</body>
</html>

```

## .Cs Code

```

protected void Page_Load(object sender, EventArgs e)
{
    if (ViewState["NameOfUser"] != null)
        NameLabel.Text = ViewState["NameOfUser"].ToString();
    else
        NameLabel.Text = "Not set yet...";
}
protected void SubmitForm_Click(object sender, EventArgs e)
{
    ViewState["NameOfUser"] = NameField.Text;
    NameLabel.Text = NameField.Text;
}

```

## Advantages

The main advantage of view state is persisting controls properties without any programming and memory will not be occupied on the client system or on the server system.

## Disadvantage

The disadvantage can be considered as a lot amount of data transmission between client and server.

1. **Control State** – If you create a custom control that requires view state to work properly, you should use control state to ensure other developers don't break your control by disabling view state.
2. **Hidden fields** – Like view state, hidden fields store data in an HTML form without displaying it in the user's browser. The data is available only when the form is processed.
3. **Cookies** – Cookies store a value in the user's browser that the browser sends with every page request to the same server. Cookies are the best way to store state data that must be available for multiple Web pages on a web site.
4. **Query Strings** - Query strings store values in the URL that are visible to the user. Use query strings when you want a user to be able to e-mail or instant message state data with a URL.

## Server Side State Management

When users visit Web sites it becomes necessary to maintain session related and controls related information. In an HTTP exchange between a browser and a remote host, session related information which identifies state, such as a unique session ID,

information about the user's preferences or authorization level is preserved. Note that sessions are maintained in the data being exchanged.

State Management is the process by which we maintain session related information and additional information about the controls and its state. The necessity of state management arises when multiple users request for the same or different Web Pages of a Web Site. State management can be accomplished using Client Side options or the Server side options.

**Session State:** Its nothing but defined as a period of time shared between the web application and user. Every user has individual session. Items/Objects can be placed into the Session which would only define these object for that user. Session contains key variables which help to identify the related values. This can be thought of as a hash table. Each user would represent a different key node in the hash identifying unique values. The Session variables will be clear by the application which can clear it, as well as through the timeout property in the web config file. Usually the timeout is 20 minutes by default. ASP.NET allows you to save values using session state, a storage mechanism that is accessible from all pages requested by a single Web browser session. Therefore, you can use session state to store user-specific information. Session state is similar to application state, except that it is scoped to the current browser session. If different users are using your application, each user session has a different session state. In addition, if a user leaves your application and then returns later after the session timeout period, session state information is lost and a new session is created for the user. Session state is stored in the Session key/value dictionary.

Session Variables are stored on the server, can hold any type of data including references, they are similar to global variables in a windows application and use HTTP cookies to store a key with which to locate user's session variables. The collection of session variables is indexed by the name of the variable or by an integer index. Session variables are created by referring to the session variable by name. You do not have to declare a session variable or explicitly add it to the collection.

**Example:**

```
//Storing informaton in session state
Session["username"] = "Ashish";
//Retrieving information from session state
string str = Session["username"];
```

### **Application State:**

Application State is used to store information which is shared among users of the ASP.Net web application. Application state is stored in the memory of the windows process which is processing user requests on the web server. Application state is useful in storing small amount of often-used data. If application state is used for such data instead of frequent trips to database, then it increases the response time/performance of the web application.

In ASP.Net, application state is an instance of `HttpApplicationState` class and it exposes key-value pairs to store information. Its instance is automatically created when a first request is made to web application by any user and same state object is being shared across all subsequent users.

Application state can be used in similar manner as session state but it should be noted that many user might be accessing application state simultaneously so any call to application state object needs to be thread safe. This can be easily achieved in ASP.Net by using lock keyword on the statements which are accessing application state object.

This lock keyword places a mutually exclusive lock on the statements and only allows a single thread to access the application state at a time.

#### Code Sample

```
//Storing information in application state
lock (this)
{
    Application["username"] = "william";
}
//Retrieving value from application state
lock (this)
{
    string str = Application["username"].ToString();
}
```

## Cookies in ASP.NET

A cookie is a small bit of text file that browser creates and stores on your machine (hard drive). Cookie is a small piece of information stored as a string. Web server sends the cookie and browser stores it, next time server returns that cookie. Cookies are mostly used to store the information about the user. Cookies are stored on the client side. To store data in a cookie is not secure due to its location at client end. Cookie was introduced with first version of Netscape navigator (that was 1.0).

### Advantages

1. Cookies do not require any server resources since they are stored on the client.
2. Cookies are easy to implement.
3. You can configure cookies to expire when the browser session ends (session cookies) or they can exist for a specified length of time on the client computer (persistent cookies).

### Disadvantages

1. Users can delete a cookies.
2. Users browser can refuse cookies ,so your code has to anticipate that possibility.
3. Cookies exist as plain text on the client machine and they may pose a possible security risk as anyone can open and tamper with cookies.

#### .ASPX CODE

```
<%@ Page Language="C#" AutoEventWireup="true" CodeVirtual="Default3.aspx.cs"
Inherits="Default3" %>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
<title></title>
</head>

<body>
<form id="form1" runat="server">
<div>
<fieldset>
```

```

<legend>rdfdf</legend>
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:Button ID="Button1" runat="server" Text="Add" Width="70px"
onclick="Button1_Click" />
<asp:Button ID="Button2" runat="server" Text="View" Width="84px"
onclick="Button2_Click" />
<br />
<br />
<asp:Label ID="Label1" runat="server" Text="" Width="138px"></asp:Label>
</fieldset>
</div>
</form>
</body>
</html>

```

## .CS CODE

```

(e EventArgs , sender object) Button1_Click protected void
{
    Cookies To Create // ; Text.TextBox1 = ["Data"] ["MyCookie"] Response.Cookies
    ; ("G") DateTime.Now.ToString = ["Time"] ["MyCookie"] Response.Cookies
    For Expire // ; (1) Expires = DateTime.Now.AddMonths. ["MyCookie"] Response.Cookies
    + "<font color=red>:Your cookie contains" + "<p>!Cookie created" = Text.Label1
    ["Data"] ["MyCookie"] Request.Cookies
    ; "<font/>" + ["Time"] ["MyCookie"] Request.Cookies + "<br>" +
    ; ("1-10-2006") Response.Cookie("MyCookie").Expires = DateTime.FromString//
    {
        (e EventArgs , sender object) Button2_Click protected void
        {
            (null == ["MyCookie"] Request.Cookies) if
            ; ":There is no cookie" = Text.Label1
            else
            + "<font color=red>" + ":Your cookie contains" = Text.Label1
            "<br>" + ["Data"] ["MyCookie"] Request.Cookies
            ; "<font/>" + ["Time"] ["MyCookie"] Request.Cookies +
            {

```

## Gridview in ASP.NET

The GridView control displays data as a table and provides the capability to sort columns, page through data, and edit or delete a single record. the GridView control offers improvements such as the ability to define multiple primary key fields, improved user interface customization using bound fields and templates, and a new model for handling or canceling events.

Displays the values of a data source in a table where each column represents a field and each row represents a record. The GridView control enables you to select, sort, and edit these items.

**Namespace:** System.Web.UI.WebControls .  
**Assembly:** System.Web (in System.Web.dll) .

## GridView Control Features

- Enhanced data source binding capabilities (Direct interaction with Data Source with any writing any ADO.NET code)
- Built-in support for sorting and paging functionalities
- Improved Design time features(Smart Panel Tag)
- Customized pager user interface with PagerTemplate property.
- Additional Column types(ImageField).
- New Event model with support for pre event and post event operations

## Difference between Datagrid,DataList and Data Repeater

- Datagrid has paging while Datalist doesn't.
- Datalist has a property called repeat. Direction = vertical/horizontal. (This is of great help in designing layouts). This is not there in Datagrid.
- A repeater is used when more intimate control over html generation is required.
- When only checkboxes/radiobuttons are repeatedly served then a checkboxlist or radio button list are used as they involve fewer overheads than a Datagrid.

## Design of Gridview:

purchase1	tax	issue_date
sugar	13	12-05-2010
tea	4	12-05-2010
oil	5	19-05-2010
cream	5	19-05-2010
books	2	
oil	324	14-05-2010

## .CS Code

```
protected void Page_Load(object sender, EventArgs e)
{
    OleDbConnection con;

    OleDbDataAdapter da;

    string conn =
System.Configuration.ConfigurationManager.AppSettings["x"].ToString();
    con = new OleDbConnection(conn);

    da = new OleDbDataAdapter("select purchase1,tax,issue_date from purchase",
```



```

con);
    DataSet ds=new DataSet();
    da.Fill( ds, "purchase");
    GridView1.DataSource = ds.Tables["purchase"];
    GridView1.DataBind();

}

```

## .ASPX Code

```

<%@ Page Language="C#" AutoEventWireup="true" CodeVirtual="Default.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/
TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
    <style type="text/css">
        .style1 {
            width: 100%;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
        <div>

            <asp:GridView ID="GridView1" runat="server" AllowPaging="True"
                BackColor="#DEBA84" BorderColor="#DEBA84" BorderStyle="None"
                BorderWidth="1px"
                CellPadding="3" CellSpacing="2" Height="341px"
                onselectedindexchanged="GridView1_SelectedIndexChanged" Width="192px">
                <FooterStyle BackColor="#F7DFB5" ForeColor="#8C4510" />
                <RowStyle BackColor="#FFF7E7" ForeColor="#8C4510" />
                <PagerStyle ForeColor="#8C4510" HorizontalAlign="Center" />
                <SelectedRowStyle BackColor="#738A9C" Font-Bold="True" ForeColor="White"
            />
                <HeaderStyle BackColor="#A55129" Font-Bold="True" ForeColor="White" />
            </asp:GridView>
            <table class="style1">
                <tr>
                    <td>
                        <marquee direction="up" behavior="scroll" onmouseover="this.stop()"
onmouseout="this.start()" scrollamount="2">
                            <a href="#">Retail Express Distribution</a><br /><br />
                            <a href="#">Global Express Consolidation Service</a><br /><br />
                            <a href="#">Import Express </a><br /><br />
                            <a href="#">Collect charges and cash on delivery</a><br /><br />
                        </marquee>&nbsp;</td>
                    <td>
                        &nbsp;</td>
                </tr>
            </table>

```

[illegible]

## Simple Code For Edit Update And Delete In Gridview

## Design of EDIT Gridview

ITEM	Name	Tax			
0	Hydrogen Peroxide	0	<a href="#">Update</a> <a href="#">Cancel</a>		
1	Sugar	5	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
2	Toothpaste	5	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
3	Rice	5	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
4	Salt	8	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
5	Soap	8	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
6	Oil	7	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>

## How to Edit Gridview

```
protected void GridView1_RowEditing(object sender,
GridViewEditEventArgs e)
{
    GridView1.EditIndex = e.NewEditIndex;

    bindgrid();
}
```

## Design of Update in Gridview

ITEM	Name	Tax			
6	Dalda	6	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
1	Sugar	5	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
2	Toothpaste	5	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
3	Rice	5	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
4	Salt	8	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
7	Soap	8	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
5	Oil	7	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>

## How to Update Gridview

This event will update information in database.

```
protected void GridView1_RowUpdating(object sender,
GridViewUpdateEventArgs e)
{
    string UserID =
GridView1.DataKeys[e.RowIndex].Value.ToString();
    string l =
((TextBox)GridView1.Rows[e.RowIndex].Cells[1].Controls[0]).Text;
    string k =
((TextBox)GridView1.Rows[e.RowIndex].Cells[2].Controls[0]).Text;
    OleDbConnection can = new
OleDbConnection("provider=microsoft.jet.oledb.4.0;Data
source=D:\\invent.mdb");
    can.Open();
    OleDbCommand com1 = new OleDbCommand("update
purchase set purchase1='" + l + "',
tax='" + k + "' where item_id='" + UserID + "'", can);

    com1.ExecuteNonQuery();
    can.Close();
    //to go back to the previous position
    GridView1.EditIndex = -1;
    // // fetch and rebind the data.
    bindgrid();
}
```

Note-- item\_id must be primary key  
And right click on gridview go to property datakey  
names=item\_id(primary key)

## How to Delete Gridview

### Design of Delete Row from Gridview

ITEM	Name	Price	Edit	Select	Delete
1	Sugar	5	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
2	Toothpaste	5	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
3	Rice	15	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
4	Salt	2	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
7	Soap	2	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
5	Oil	7	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>

## How to Delete Particular Row From Gridview

This event shows how to delete a row from database

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    string UserID = GridView1.DataKeys[e.RowIndex].Value.ToString();

    OleDbConnection can = new OleDbConnection("provider=microsoft.jet.
oledb.4.0;Data source=D:\\invent.mdb");
    can.Open();
    OleDbCommand com1 = new OleDbCommand("delete from purchase
where item_id='"+UserID+"' , can);
    com1.ExecuteNonQuery();
    can.Close();
    //to go back to the previous position
    GridView1.EditIndex = -1;
    // // fetch and rebind the data.
    bindgrid();
}
```

## How to cancel editable mode.

```
protected void GridView1_RowCancelingEdit(object sender,
GridViewCancelEventArgs e)
{
    GridView1.EditIndex = -1;
    bindgrid();
}
```

## Simple Select, Edit, Update and Delete in Asp.Net GridView

### Design Of Gridview

ITEM	Name	Tax			
6	Rajdhani Aata	6	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
1	Sugar	5	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
2	Toothpaste	5	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
3	Rice	5	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
4	Salt	3	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
7	Soap	3	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>
5	Oil	7	<a href="#">Edit</a>	<a href="#">Select</a>	<a href="#">Delete</a>

## .ASPX CODE

```

<%@ Page Language="C#" AutoEventWireup="true" CodeVirtual="Default.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>

            <asp:GridView ID="GridView1" runat="server" AllowPaging="True"
                AutoGenerateColumns="False" BackColor="#DEBA84" BorderColor="#DEBA84"
                BorderStyle="None" BorderWidth="1px" CellPadding="3" CellSpacing="2"
                DataKeyNames="item_id" onrowcancelingedit="GridView1_RowCancelingEdit"
onrowdeleting="GridView1_RowDeleting"
onrowediting="GridView1_RowEditing"
onrowupdating="GridView1_RowUpdating"
onselectedindexchanged="GridView1_SelectedIndexChanged">
                <FooterStyle BackColor="#F7DFB5" ForeColor="#8C4510" />
                <RowStyle BackColor="#FFF7E7" ForeColor="#8C4510" />
                <Columns>
                    <asp:BoundField DataField="item_id" HeaderText="ITEM" />
                    <asp:BoundField DataField="purchase1" HeaderText="name" />
                    <asp:BoundField DataField="tax" HeaderText="tax" />
                    <asp:CommandField ShowEditButton="True" />
                    <asp:CommandField ShowSelectButton="True" />
                    <asp:CommandField ShowDeleteButton="True" />
                </Columns>
                <PagerStyle ForeColor="#8C4510" HorizontalAlign="Center" />
                <SelectedRowStyle BackColor="#738A9C" Font-Bold="True" ForeColor="White"
/>
                <HeaderStyle BackColor="#A55129" Font-Bold="True" ForeColor="White" />
                <AlternatingRowStyle BorderColor="#FF99FF" />
            </asp:GridView>

        </div>
    </form>
</body>
</html>

```

## .CS CODE

```

using System;
using System.Configuration;
using System.Data;
using System.Linq;

```

```

using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Data.OleDb;

public partial class _Default : System.Web.UI.Page
{
    public void bindgrid()
    {
        OleDbConnection con;

        OleDbDataAdapter da;

        con = new OleDbConnection("provider=microsoft.jet.oledb.4.0;Data
source=D:\\invent.mdb");

        da = new OleDbDataAdapter("select item_id, purchase1,tax from purchase", con);
        DataSet ds = new DataSet();
        da.Fill(ds, "purchase");
        GridView1.DataSource = ds.Tables["purchase"];
        GridView1.DataBind();
    }
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            bindgrid();
        }
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
    }
    protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
    {
    }
    protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
    {
        GridView1.EditIndex = e.NewEditIndex;

        bindgrid();
    }
    protected void GridView1_RowUpdating(object sender, GridViewUpdateEventArgs e)
    {
        string UserID = GridView1.DataKeys[e.RowIndex].Value.ToString();
        string l = ((TextBox)GridView1.Rows[e.RowIndex].Cells[1].Controls[0]).Text;
        string k = ((TextBox)GridView1.Rows[e.RowIndex].Cells[2].Controls[0]).Text;
        OleDbConnection can = new
OleDbConnection("provider=microsoft.jet.oledb.4.0;Data
source=D:\\invent.mdb");
        can.Open();
    }
}

```

```

        OleDbCommand com1 = new OleDbCommand("update purchase set purchase1="
+ l + " ,tax="
+ k + " where item_id=" + UserID + """, can);

        com1.ExecuteNonQuery();
        can.Close();
        //to go back to the previous position
        GridView1.EditIndex = -1;
        // // fetch and rebind the data.
        bindgrid();

    }
    protected void GridView1_RowCancelingEdit(object sender,
GridViewCancelEventArgs e)
    {
        GridView1.EditIndex = -1;
        bindgrid();
    }
    protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs
e)
    {
    }
    protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
    {
        string UserID = GridView1.DataKeys[e.RowIndex].Value.ToString();

        OleDbConnection can = new
OleDbConnection("provider=microsoft.jet.oledb.4.0;Data
source=D:\\invent.mdb");
        can.Open();
        OleDbCommand com1 = new OleDbCommand("delete from purchase where
item_id =" + UserID + """, can);

        com1.ExecuteNonQuery();
        can.Close();
        //to go back to the previous position
        GridView1.EditIndex = -1;
        // // fetch and rebind the data.
        bindgrid();

    }
}

```

## The ASP.NET DataList Control

The DataList control like the Repeater control is a template driven, light weight control, and acts as a container of repeated data items. The templates in this control are used to define the data that it will contain. It is flexible in the sense that you can easily customize the display of one or more records that are displayed in the control. You have a property in the DataList control called **RepeatDirection** that can be used to customize the layout of the control.



The RepeatDirection property can accept one of two values, that is, Vertical or Horizontal. The RepeatDirection is Vertical by default. However, if you change it to Horizontal, rather than displaying the data as rows and columns, the DataList control will display them as a list of records with the columns in the data rendered displayed as rows.

the DataList control as a combination of the DataGrid and the Repeater controls. You can use templates with it much as you did with a Repeater control and you can also edit the records displayed in the control, much like the DataGrid control of ASP.NET. The next section compares the features of the three controls that we have mentioned so far, that is, the Repeater, the DataList, and the DataGrid control of ASP.NET.

When the web page is in execution with the data bound to it using the Page\_Load event, the data in the DataList control is rendered as DataListItem objects, that is, each item displayed is actually a DataListItem. Similar to the Repeater control, the DataList control does not have Paging and Sorting functionalities build into it.

### Using the DataList Control:

To use this control, drag and drop the control in the design view of the web form onto a web form from the toolbox.

Refer to the following screenshot, which displays a **DataList** control on a web form:



### DataList Control Contain Following Templates:

1. ItemTemplate
2. AlternatingItemTemplate
3. EditItemTemplate
4. FooterTemplate
5. HeaderTemplate
6. SelectedItemTemplate
7. SeparatorTemplate

Here is a sample of how your DataList control's templates are arranged:

```
<asp:DataList id="dlEmployee" runat="server">
<HeaderTemplate>
...
</HeaderTemplate>
<ItemTemplate>
...
</ItemTemplate>
<AlternatingItemTemplate>
...
</AlternatingItemTemplate>
<FooterTemplate>
```

```
...
</FooterTemplate>
</asp:DataList>
```

### Output:

Item Name	Tax	Location	Issue Date
Sugar	8	Kanpur	12-05-2010
Toothpaste	5	Ghaziabad	17-05-2010
Rice	5	Lucknow	19-05-2010
Soap	8	Mumbai	22-05-2010
Kanpur	9	Banaras	27-06-2010
Kanpur	10	Jalaun	1-07-2010
Oil	7	Orai	14-05-2010

### .aspx Code:

```
<asp:DataList ID="DataList1"
runat="server">
  <HeaderTemplate>
    <table border="1">
      <tr>
        <th>
          Item Name
        </th>
        <th>
          Tax
        </th>
        <th>
          Location
        </th>
        <th>
          Issue Date
        </th>
      </tr>
    </HeaderTemplate>
    <ItemTemplate>
      <tr bgcolor="#0xbbbb">
        <td>
          <%#
            DataBinder.Eval(Container.DataItem,
              "purchase1")%>
        </td>
        <td>
          <%#
            DataBinder.Eval(Container.DataItem,
```

```

        "tax")%>
    </td>
    <td>
        <%#
        DataBinder.Eval(Container.DataItem,
        "ship_location")%>
    </td>
    <td>
        <%#
        DataBinder.Eval(Container.DataItem,
        "issue_date")%>
    </td>
</tr>
</ItemTemplate>
<FooterTemplate>
</FooterTemplate>
</asp:DataList>

```

#### .CS CODE:

```

public void bindgrid()
{
    OleDbConnection con;

    OleDbDataAdapter da;
    con = new OleDbConnection("provider=microsoft.jet.oledb.4.0;Data
source=D:\\invent.mdb");
    da = new OleDbDataAdapter("select * from purchase", con);
    //DataTable dt = new DataTable();
    DataSet ds = new DataSet();
    da.Fill(ds);
    DataList1.DataSource = ds.Tables[0];
    DataList1.DataBind();
}

```

## Detailview in ASP.NET

The DetailsView is a control that is complementary to the GridView control. It has been introduced in ASP.NET 2.0 to provide an ability to work with a single record or row from an associated data source. The default view of this control is vertical with each column of the record displayed on a line of its own. It is used for updating and inserting new records and used often in the Master-details scenario where the details of the record selected in the master is displayed in detail.

The DetailsView, like the GridView rides on the data source capabilities to update, insert or delete rows. It renders a single data item at a time even when the data source exposes more items. If the data object represents the ICollection interface or the underlying data source exposes the paging operation, the DetailsView can page over the data, provided the AllowPaging property is set to true. Users can also navigate between rows of data using this control.

The process of binding the DetailsView control to a data source is very simple. The DataSourceID property can be set declaratively or programmatically.

The `AutoGenerateEditButton` property, if set to `True`, enables editing operations and the `DetailsView` renders the edit button in addition to the other columns in the control. The Edit mode can be invoked pressing this button. The Edit user interface can be customized using `Styles`, `DataControlField` objects and templates.

The `DetailsView` control can be configured to display a Delete and insert button also. The `AutoGenerateInsertButton`, when set to `true`, generates a New button. The new button invokes the insert mode and appropriate user interfaces are rendered including the fields in the `DataKeyNames` collection. Using this collection the data source control can locate the unique row to be updated.

The `DetailsView` control user interface can be customized using `Style` properties such as `HeaderRowStyle`, `RowStyle`, `AlternatingRowStyle` and so on. Templates also can be used to customize this control. `EmptyDataTemplate`, `HeaderTemplate`, `FooterTemplate` and `PagerTemplate` are some of the templates available for this control.

A number of events can be used to customize the code including pre and post insert, update and delete data source events. This includes `ItemCreated` and `ItemCommand` events. There is no selected event in the `DetailsView` control as the current event is treated as the selected event.

### **DetailsView Fields in ASP.Net**

<b>The DetailView control supports the following Fields</b>	
<b>BoundField</b>	Displays the value of a data item as text
<b>CheckBoxField</b>	Displays the value of the data item as a check box
<b>CommandField</b>	Displays links for editing, deleting and selecting rows
<b>ButtonField</b>	Displays the value of a data item as a button, imagebutton, linkbutton
<b>HyperlinkField</b>	Displays the value of a data item as a link
<b>ImageField</b>	Displays the value of a data item as an image
<b>TemplateField</b>	Customize the appearance of a data item

### **Example:**

Item name	Sugar
Tax	8
Location	kanpur
Date	12-05-2010
1 2 3 4 5 6 7	

#### .aspx Code:

```

<asp:DetailsView ID="DetailsView1" runat="server" BackColor="White"
    BorderColor="#DEDFDE" BorderStyle="None" BorderWidth="1px"
    CellPadding="4"
    ForeColor="Black" GridLines="Vertical" Height="229px"
    onpageindexchanging="DetailsView1_PageIndexChanging"
    Width="399px"
    AllowPaging="True" AutoGenerateRows="False">
    <FooterStyle BackColor="#CCCC99" />
    <RowStyle BackColor="#F7F7DE" />
    <PagerStyle BackColor="#F7F7DE" ForeColor="Black"
    HorizontalAlign="Right" />
    <Fields> <asp:BoundField DataField="tax" HeaderText="Tax" />
    <asp:BoundField DataField="ship_location" HeaderText="Location" />
    <asp:BoundField DataField="issue_date" HeaderText="Date" />
    </Fields><HeaderStyle BackColor="#6B696B" Font-Bold="True"
    ForeColor="White" />
    <EditRowStyle BackColor="#CE5D5A" Font-Bold="True"
    ForeColor="White" />
    <AlternatingRowStyle BackColor="White" />
    </asp:DetailsView>

```

#### .CS CODE:

```

bindgrid public void ()
{
    ;OleDbConnection con
    ;OleDbDataAdapter da

    provider=microsoft.jet.oledb.4.0;Data ")OleDbConnection new = con
    ;("source=D:\\invent.mdb

    select purchase1,tax,issue_date,ship_location from ")OleDbDataAdapter new = da
    ;(con ,"purchase
    ;()DataTable dt = new DataTable//
    ;()DataSet new = ds DataSet
    ;da.Fill(ds)

    ;[0]DetailsView1.DataSource= ds.Tables

    ;()DetailsView1.DataBind

    (++DropDownList1.Items.Count; i > for (int i = 0; i//
    }//
    .[DropDownList1.Items[i].Attributes.Add("style", "color:" + ds.Tables[0 //
    ;()Rows[i]["categorycolor"].ToString
    }//

    {
    (e EventArgs ,sender object)Page_Load protected void
    }
    ;()bindgrid
    {

```

## Formview Control in ASP.NET

FormView is the new feature in the ASP.NET 2.0, for display the data in the FormView control; we use the Eval method in the ItemTemplate. You must create a Template for the mode in which the FormView control is configured. For example, a FormView control that supports updating records must have an EditItemTemplate defined.

### Types of Templates:

- HeaderTemplate
- FooterTemplate
- ItemTemplate
- InsertItemTemplate
- EditItemTemplate
- EmptyDataTemplate
- PagerTemplate

### HeaderTemplate:

Header Template defines the contents which we want to display in the header row of the FormView Control.

### FooterTemplate:

Footer Template defines the contents which we want to display in the footer row of the FormView Control.

### ItemTemplate:

ItemTemplates defines the contents for the FormView row when it is in read only mode. This template basically used for display the existing data from the data source.

### InsertItemTemplate:

InsertItemTemplates defines the contents for the FormView row when it is in insert mode. This template contains the input controls and commands for inserting the new record.

### EditItemTemplate:

EditItemTemplates defines the contents for the FormView row when it is in edit mode. This template contains the input controls and commands for editing the existing record.

### EmptyDataTemplate:

EmptyDataTemplate basically used for showing the alert to the user if there is no record in the bound column.

### PagerTemplate:

PagerTemplate defines the content for the pager row when the "AllowPaging" property is true. It also contains the controls by using that, user can navigate to another record.

### Output:

#### .aspx Code:

```
                "FormViewStudent"=ID asp:FormView>
                "White"=BackColor "server"=runat
                "None"=BorderStyle "DEDFDE#"=BorderColor
                "4"=CellPadding "px1"=BorderWidth
                "px229"=Height "Vertical"=GridLines "Black"=ForeColor
                <"FormViewStudent_PageIndexChanging"=onpageindexchanging
                <ItemTemplate>
                <table>
                <tr>
                <"right"=align td>
```

```

        <td/><b/>:ID <b>
            <td>
                <% ("ID") Eval #%>
            <td/>
        <tr/>
    <tr>
        <"right"=align td>
        <td/><b/>:Supplier Name<b>
            <td>
                <% ("supplier_name") Eval #%>
            <td/>
        <tr/>
    <tr>
        <"right"=align td>
        <td/><b/>:contact Person<b>
            <td>
                <% ("contact_person") Eval #%>
            <td/>
        <tr/>
    <tr>
        <"right"=align td>
        <td/><b/>:Address<b>
            <td>
                <% ("address") Eval #%>
            <td/>
        <tr/>
    <table/>
</ItemTemplate/>

```



**.CS CODE:**

```

public void bindformview()
{
    OleDbConnection con;
    OleDbDataAdapter da;

    con = new OleDbConnection("provider=microsoft.jet.oledb.4.0;Data
source=D:\\invent.mdb");

    da = new OleDbDataAdapter("select ID,supplier_name,contact_person,address
from supplier", con);
    //DataTable dt = new DataTable();
    DataSet ds = new DataSet();
    da.Fill(ds);

    FormViewStudent.DataSource = ds.Tables[0];

    FormViewStudent.DataBind();
}
protected void Page_Load(object sender, EventArgs e)
{
    bindformview();
}

```

**Repeater Control in ASP.NET**

Repeater control is a template based container control. you define layout for the Repeater control by creating different templates based on your needs. The Repeater control may be bound to a database table, an XML file, or another list of items. Here we will show how to bind data to a Repeater control.

A Repeater control is a light weight control which can be used for simple reporting purposes. It supports basic event-handling like Init, Load, Unload etc., This also supports some basic formatting of data and can be presented to the user. A Repeater control offers limited level of data editing or selecting capabilities. For such editing and updates ASP .Net offers DataList and DataGrid controls.

**Repeater control supports five templates which are as follows**

- ItemTemplate
- AlternatingItemTemplate

- HeaderTemplate
- FooterTemplate
- SeparatorTemplate

**ItemTemplate:** ItemTemplate defines how the each item is rendered from data source collection.

**AlternatingItemTemplate:** AlternatingItemTemplates define the markup for each Item but for AlternatingItems in DataSource collection like different background color and styles.

**HeaderTemplate:** HeaderTemplate will emit markup for Header element for DataSource collection

**FooterTemplate:** FooterTemplate will emit markup for footer element for DataSource collection

**SeparatorTemplate:** SeparatorTemplate will determine separator element which separates each Item in Item collection. Usually, SeparatorTemplate will be <br> html element or <hr> html element.

Example:

Supplier Name	ContactPerson	Address
Amit	Rakesh	GZB
Vipin	Rituraj	Delhi
Rakesh	Rohan	kanpur
Anil	Ramprasad	orai
vinay	Arjun	Banaras
Puneet	Rahul	jalaun
Raghav	Aslam	Lucknow
Aarif	prabhakar	Allahabad
Aditya	Gyanendra	Noida
kamal		

**.aspx Code:**

```
<asp:Repeater ID="Repeater1" runat="server"
    onitemcommand="Repeater1_ItemCommand">
    <HeaderTemplate>
    <table border="1" cellpadding="5" cellspacing="2">
    <tr bgcolor="gray">
    <td><b>Supplier Name</b></td>
    <td><b>ContactPerson</b></td>
    <td><b>Address</b></td>
```

```

</tr>
</HeaderTemplate>
<ItemTemplate>
<tr>
<td>
<%#DataBinder.Eval(Container.DataItem, "supplier_name")%>
</td>
<td>
<%#DataBinder.Eval(Container.DataItem, "contact_person")%>
</td>

<td>
<%#DataBinder.Eval(Container.DataItem, "address")%>
</td>
</tr>
</ItemTemplate>
<AlternatingItemTemplate >
<tr bgcolor="aqua" >
<td>
<%#DataBinder.Eval(Container.DataItem, "supplier_name")%>
</td>
<td>
<%#DataBinder.Eval(Container.DataItem, "contact_person")%>
</td>
<td>
<%#DataBinder.Eval(Container.DataItem, "address")%>
</td>
</tr>
</AlternatingItemTemplate>
<FooterTemplate>
</table>
</FooterTemplate>
</asp:Repeater>

```

### .CS CODE:

```

public void repeaterbinddata()
{

    OleDbConnection con;
    OleDbDataAdapter da;
    con = new
OleDbConnection("provider=microsoft.jet.oledb.4.0;Data
source=D:\\invent.mdb");
    da = new OleDbDataAdapter("select
supplier_name,contact_person,address from supplier", con);

    //DataTable dt = new DataTable();
    DataSet ds = new DataSet();
    da.Fill(ds);
}

```

```

        Repeater1.DataSource= ds.Tables[0];

        Repeater1.DataBind();
    }

    protected void Page_Load(object sender, EventArgs e)
    {

        repeaterbinddata();
    }

```

## How to Rotate Image in ASP.NET

### Design:



When You Click On Button Image Will Rotate:

Like this



#### .ASPX CODE:

```
<asp:Image ID="Image1" runat="server" Height="297px"
    ImageUrl="~/images/image.bmp" Width="343px" />

<asp:Button ID="Button1" runat="server"
onclick="Button1_Click"
    Text="Rotate Image" />
```

#### .CS CODE:

```
protected void Button1_Click(object sender, EventArgs e)
{
    // get the full path of image url
    string path = Server.MapPath(Image1.ImageUrl);

    // creating image from the image url
    System.Drawing.Image i =
    System.Drawing.Image.FromFile(path);

    // rotate Image 90' Degree
    i.RotateFlip(RotateFlipType.Rotate90FlipXY);

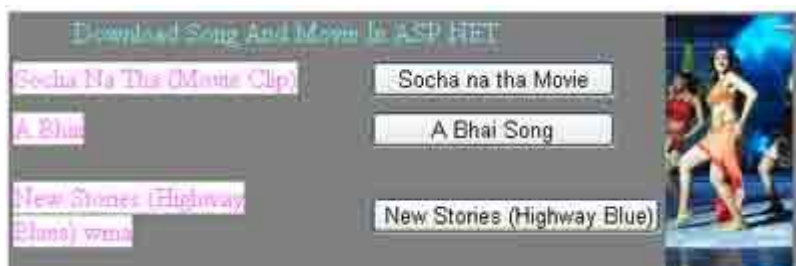
    // save it to its actual path
    i.Save(path);

    // release Image File
    i.Dispose();

    // Set Image Control Attribute property to new image(but its
old path)
    Image1.Attributes.Add("ImageUrl", path);
}
```

## How to Download Any File in ASP.NET

### Design:



### Write Code in .CS File:

```
public void Download()
{
    string filename = "Socha Na Tha ";
    if (filename != "")
    {
        string path = Server.MapPath(filename);
        System.IO.FileInfo file = new System.IO.FileInfo("D://Socha Na
Tha.avi");
        if (file.Exists)
        {
            Response.Clear();
            Response.AddHeader("Content-disposition", "attachment;
filename=" + filename);
            Response.AddHeader("content-Length", file.Length.ToString());
            Response.ContentType = "application/octet-stream";//
            Response.WriteFile(file.FullName);
            Response.End();
        }
        else
        {
            Response.Write("this file does not exist");
        }
    }
}

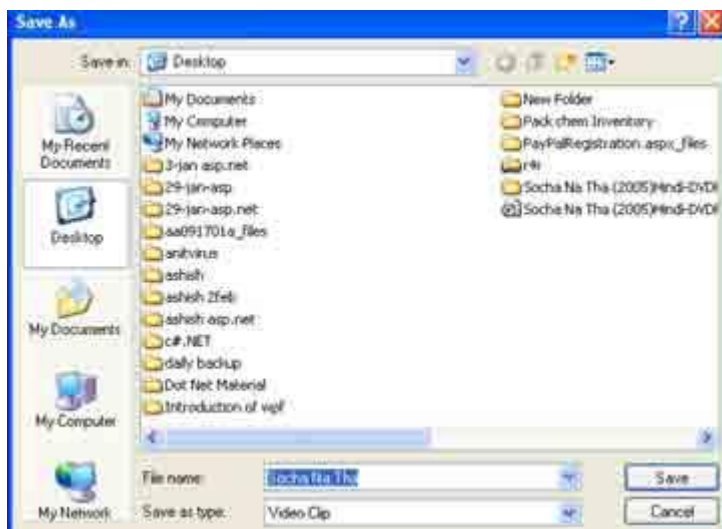
protected void Button1_Click(object sender, EventArgs e)
{
    Download();
}
```

### Output:



When You Click Soch Na Tha Movie Button This Dialog Box Will Appear

When You Click On Save This File Any where in Computer



After This Download process Start:



## How to Save Image in Database

Create a table in a SQL Server 2005 database which has at least one field of type IMAGE.

	picture	comment
	<Binary data>	water
*	NULL	NULL

### Design :



### Namespaces used in this application:

```
using System.Data.SqlClient;
using System.Drawing;
using System.Data;
using System.IO;
using System.Drawing.Imaging;
```

### Source Code for Save the image file into the database

```
protected void Button1_Click(object sender, EventArgs e)
{
    int length = FileUpload1.PostedFile.ContentLength;
    byte[] pic = new byte[length];

    FileUpload1.PostedFile.InputStream.Read(pic, 0, length);
    // Insert the image and comment into the database

    SqlConnection con = new SqlConnection(@"data source=R4R-
01\SQLEXPRESS;initial
catalog=saveimage;integrated security=true");
    try
```



```

    {
        con.Open();
        SqlCommand cmd = new SqlCommand("insert into
saveimage1 "
        + "(Picture, Comment) values (@a, @b)", con);
        cmd.Parameters.Add("@a", pic);
        cmd.Parameters.Add("@b", TextBox1.Text);
        cmd.ExecuteNonQuery();
    }
    finally
    {
        con.Close();
    }
}

```

### Read image from a database using ADO.NET and display it in a Web Page

```

protected void Button2_Click(object sender, EventArgs e)
{
    // Put user code to initialize the page here

    MemoryStream stream = new MemoryStream();
    SqlConnection con = new SqlConnection(@"data source=R4R-
01\SQLEXPRESS;initial
catalog=saveimage;integrated security=true");
    con.Open();
    SqlCommand command = new SqlCommand("select picture
from saveimage1",con);
    byte[] image = (byte[])command.ExecuteScalar();

    stream.Write(image, 0, image.Length);
    Bitmap bitmap = new Bitmap(stream);
    Response.ContentType = "image/gif";

    bitmap.Save(Response.OutputStream, ImageFormat.Gif);

}

```

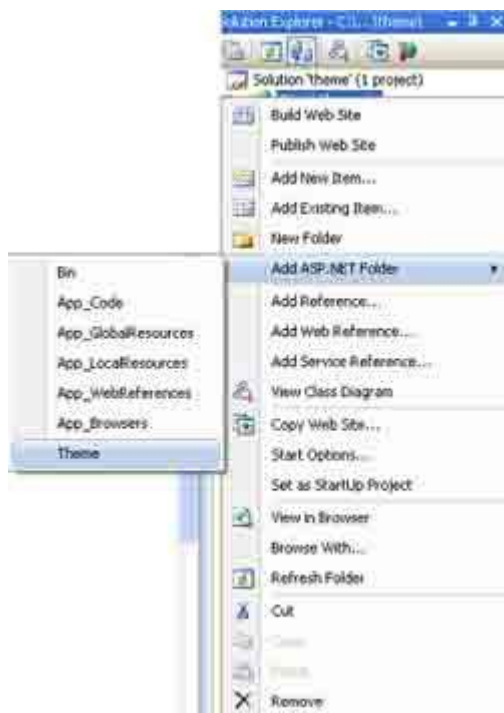
### Output:



## Skin Style And Theme in ASP.NET

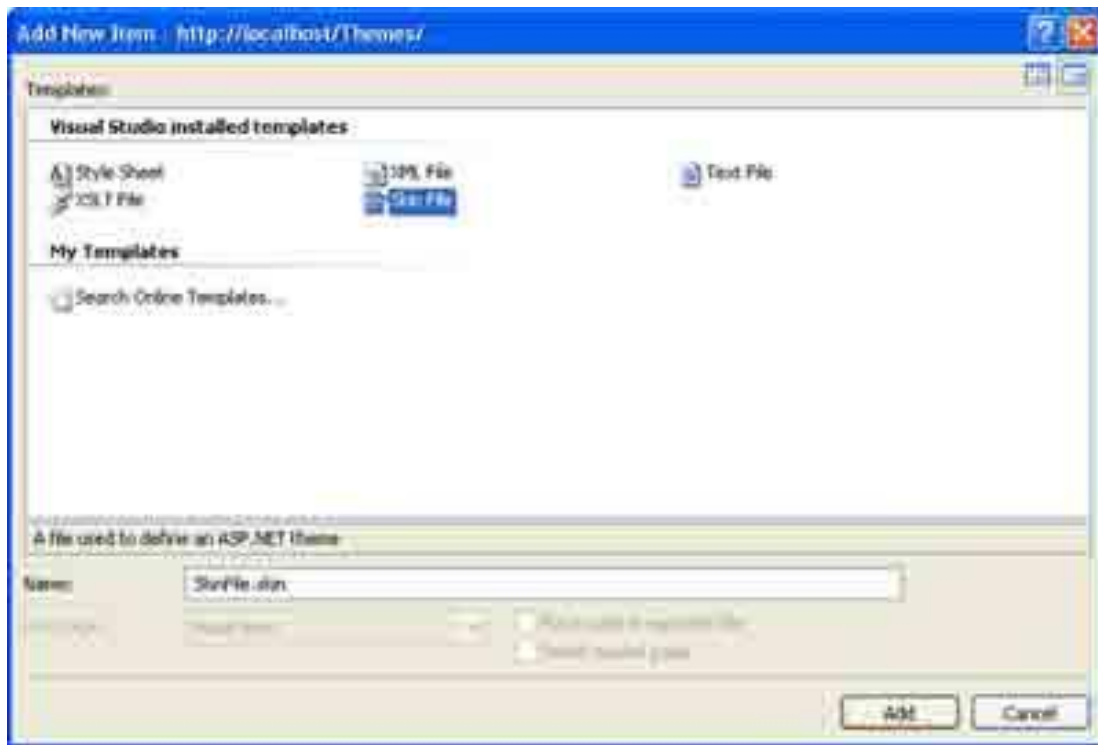
Themes are the great way to customize user-experience in the web application. The main purpose of Themes are used to define the look and feel of the web application, similar to the use of Cascading Style Sheets (CSS). OR we can say Themes are used to define visual styles for web pages. Themes are made up of a set of elements: skins, Cascading Style Sheets (CSS), images, and other resources. Themes can be applied at the application, page, or server control level.

Step1..Once you've created your site, the next step is to create a theme. Right click in Solution Explorer and select **Add ASP.NET Folder -> Theme**.



In order to add a skin file to an existing theme, you just right click on the theme's folder in Solution Explorer and select Add New Item....

In the resulting dialog box, select "Skin File" as the type of file and choose a name for the file. For this example, I'll just use the default value of "SkinFile.skin".



Right Code And Set Color Size and Border of Control in .Skin File

```
<asp:button runat="server" BackColor="lightblue" BorderColor="AliceBlue"
Font-Bold
="true" ForeColor="black"/>

<asp:Calendar runat="server" BackColor="White"

        BorderColor="#3366CC" BorderWidth="1px" CellPadding="1"

        DayNameFormat="Shortest" Font-Names="Verdana" Font-Size="8pt"

        ForeColor="#003399" Height="200px" Width="220px">

    <SelectedDayStyle BackColor="#009999" Font-Bold="True"

        ForeColor="#CCFF99" />

    <SelectorStyle BackColor="#99CCCC" ForeColor="#336666" />

    <WeekendDayStyle BackColor="#CCCCFF" />

    <OtherMonthDayStyle ForeColor="#999999" />

    <TodayDayStyle BackColor="#99CCCC" ForeColor="White" />

    <NextPrevStyle Font-Size="8pt" ForeColor="#CCCCFF" />

    <DayHeaderStyle BackColor="#99CCCC" ForeColor="#336666" Height="1px"
/>
```

```

<TitleStyle BackColor="#003399" BorderColor="#3366CC" BorderWidth="1px"
    Font-Bold="True"
    Font-Size="10pt" ForeColor="#CCCCFF" Height="25px" />
</asp:Calendar>
<asp:Image runat="server"
    ImageUrl = "~/App_Themes/p_0010.jpg"
    BorderWidth = "4"
    BorderStyle = "Dashed"
    BorderColor = "Navy"
/>
<asp:TextBox runat="server" BorderColor="#3366CC" BorderWidth="1px"
    ForeColor="#003399" Font-Names="Verdana" Font-
    Size="8pt"></asp:TextBox>

```

Write This line In .ASPX Page:

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Theme="Theme1" Inherits="_Default" %>

```

When you view the page without skin in a browser, you should see:

When you set your theme to the ".aspx" page and view the page in a browser, you should see:



### Differences between Themes and CSS:

- o CSS deals with HTML code. You cannot apply CSS to certain ASP.NET specific server controls which are not present in HTML.
  - o You can apply Themes and skins to all type of ASP.NET controls with less effort. Themes and skins can be uniformly applied on both windows and asp.net applications.
  - o You can apply single theme to each page where as multiple style sheets can be applied to each page.
  - o Themes don't override or cascade style definitions by default the way CSS generally do. But you can selectively override local property settings for a control using StyleSheetTheme attribute in Themes.
  - o You can include CSS files in Themes which is applied as part of Theme structure but not vice-versa.
- You can apply theming for only those properties that have ThemeableAttribute attribute set to true in their control class.
- Skins

## How to Write Gridview data to XML file in .NET

In this Example you have to write the file name with the extension .xml and the code will create the xml file with that name and populate it.

### CS Code:

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Xml;

public partial class Default6 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        DataSet ds = new DataSet();
        ds.ReadXml(MapPath("Regis.xml"));
        GridView1.DataSource = ds;
        //DataGrid1.DataSource = ds;
        GridView1.DataBind();
    }

    protected void Button3_Click(object sender,
    EventArgs e)
    {
        DataSet ds = new DataSet("TestDataset");
        DataTable dt = new DataTable("TestDataTable");
        dt.Columns.Add("Id",
        System.Type.GetType("System.String"));
        dt.Columns.Add("Loc",
        System.Type.GetType("System.String"));
        dt.Columns.Add("email",
```

```

System.Type.GetType("System.String"));
dt.Columns.Add("Disc",
System.Type.GetType("System.String"));
dt.Columns.Add("Date",
System.Type.GetType("System.String"));
ds.Tables.Add(dt);
// ds.Tables.Add(dt);
//Add any number columns here
for (int count = 0; count < GridView1.Rows.Count;
count++)
{
    DataRow dr = dt.NewRow();
    dr[0] = GridView1.Rows[count].Cells[0].Text;
    dr[1] = GridView1.Rows[count].Cells[1].Text;
    dr[2] = GridView1.Rows[count].Cells[2].Text;
    dr[3] = GridView1.Rows[count].Cells[3].Text;
    dr[4] = GridView1.Rows[count].Cells[4].Text;

    //you can add values to all columns in datatable
    dt.Rows.Add(dr);
}
ds.Tables[0].WriteXml("E:\\test2.xml");
}

```

```

<?xml version="1.0" encoding="utf-8"?>
<Information>
  <Comments>
    <Name>Ashish Kumar Gupta</Name>
    <location>Ghaziabad</location>
    <Email>Ashish07akg@gmail.com</Email>
    <Description>very Nice Article</Description>
    <Date>1/12/2011 11:22:13 AM</Date>
  </Comments>
  <Comments>
    <Name>Anil</Name>
    <location>Delhi</location>
    <Email>anil.rkgit@gmailcom</Email>
    <Description>Very nice Article Thanks</Description>
    <Date>1/12/2011 11:29:56 AM</Date>
  </Comments>
  <Comments>
    <Name>Anil</Name>
    <location>Delhi</location>
    <Email>anil.rkgit@gmailcom</Email>
    <Description>Very nice Article Thanks</Description>
    <Date>1/12/2011 11:30:09 AM</Date>
  </Comments>
  <Comments>
    <Name>aditya</Name>
    <location>anil</location>
    <Email>ash</Email>
  </Comments>

```

```

<Description>sferterte</Description>
<Date>4/1/2011 8:46:51 AM</Date>
</Comments>
</Information>

```

### Output:

Name	location	Email	Description	Date
Ashish Kumar Gupta	Ghaziabad	Ashish07akg@gmail.com	very Nice Article	1/12/2011 11:22:13 AM
Anil	Delhi	anil.rkgit@gmail.com	Very nice Article Thanks	1/12/2011 11:29:56 AM
Anil	Delhi	anil.rkgit@gmail.com	Very nice Article Thanks	1/12/2011 11:30:09 AM
aditya	anil	ash	sferterte	4/1/2011 8:46:51 AM

generate xml file

When You Click Generate xml file Button XML file is generated in E Drive with Name test2.xml:

[test2.xml](#)