Introduction :- You already Know ,Internet is emerging as the most widely used medium for performing various tasks,such as online shopping ,Data exchanging and bank transactions etc. All this Information and data need to be secured against unauthorized access from malicious and illegal sources . For this purpose, we use authentication and authorization process. You can learn more from below about authentication and authorization in asp.net application.

- Windows Authentication
- Form Based Authentication

You already know, we need to write large piece of code to create forms and user interfaces for authenticating the user and displaying the desired page based on the roles or rights given to the user.But It is very time consuming so that Microsoft developed a new series of server controls ,called login controls.

To use login controls in your website. You just need to drag and drop them on the web page.

There is no need to write more codes in the codes-behind file.The Login controls have built in functionality for authentication and authorization of users.Microsoft were introduced (started) this services with ASP.NET 2.0. This control help to build the registration and login application without writing any code-behind codes and database.

The Membership Service

This membership services is an important feature of ASP.NET that helps you validatng and storing user credentials.

The ASP.NET Membership services helps to implement the folllowing functionalities in an application.

- To create new user and password
- To store the membership information ,such as username ,password .address,email and supporting data
- To authenticate and authorization the visitors of the websites.
- It allows the user to create and reset the password
- It allows to create a unique Identification system for authenticated users

The Login Controls:-

There are following Login controls developed by the Microsoft which are used in ASP.NET Website as given below:-

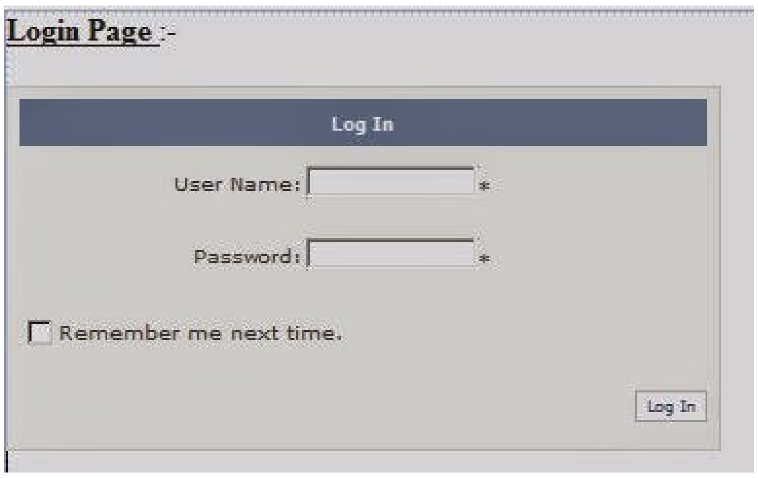
1. Login
2. LoginView
3. LoginStatus
4. Loginname
5. PasswordRecovery
6. ChangePassword

7. CreateUserWizard

1.) The Login Control:-

The Login control provides a user interface which contains username and password, that authenticate the usernaMe and password and grant the access to the desired services on the the basis of the credentials.

There are used some methods ,properties and events in this Login control,You can check manually after drag and drop this control on your web form as given below:-

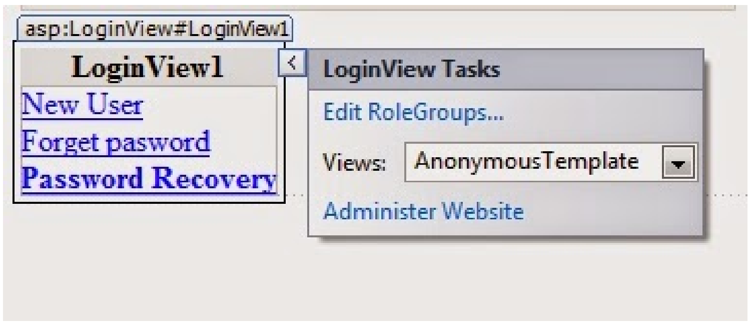


2.) The LoginView Control:-

The LoginView Control is a web server control ,Which is used to display two different views of a web page of any website , dependending on whether the any user has logged on to a web page as anonymous user or registered user .If the user is authenticated,the control displays the appropriate to the person with the help of the following views template.

- **Anonymous Template** :- This template (default of all) will be displayed when any user just open the web page but not logged in.
- **LoggedInTemplate**:- This Template (page)will be displayed when the user in logged in.
- **RoleGroups**:- This template will be displayed when user logged in, that is the member of the specific role (defined role group).

You can drag and drop Loginview Control on the web page from toolbox as shown below:-



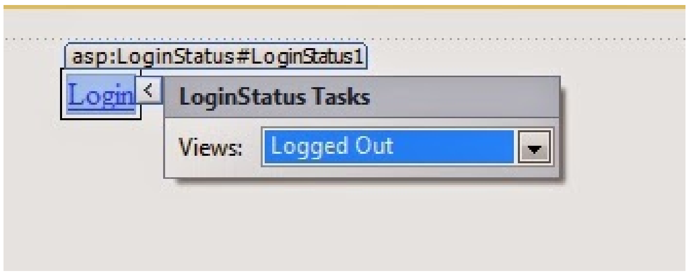
Note:- You can do so, by adding any server controls such as Label ,Hyperlink and TextBox, to the empty region on the loginview control.

3.) The LoginStatus Control :-

The LoginStatus control specifies whether a particular user has logged on the website or not . When the user is not logged in,this control display the Login text as a hyperlink.When the user is logged in ,this control display the logout text as a hyperlink.
To do this ,Login Status control uses the authentication section of the web.config file. This control provides the following two views:-

1. **LoggedIn** --> It will be displayed,when the user is not Logged In.
2. **Logout** --> It will be displayed when the user is Logged In.

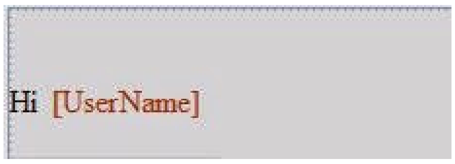
You can drag and drop this control on the web page as shown below:-



4.) The LoginName Control :-

The LoginName Control is responsible for display the names of all the authenticated users.If no users id logged in ,then this control is not displayed on the page.
This control uses the page .User.Identity.Name namespace to return the value for the user's name.

You can drag and drop Login Name control on the page from the toolbox as given below:-



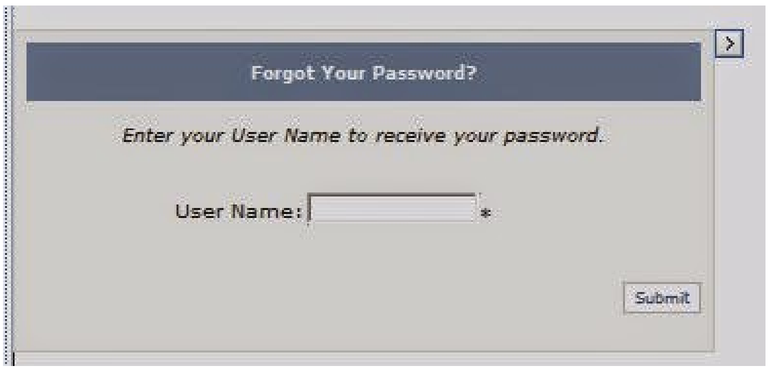
5.) Passwordrecovery Control:-

The Passwordrecovery control is used to recover or reset the forgotten password of a user . This control does not display the password on the browser,but it sends the password to the respective email address whatever you have specified at the time of registration.

This control has included three views as given below:-

1. **UserName** :- It refers to the view that accepts the username of a user.
2. **Question** :- It accepts the security questions asked from the users.
3. **Success** :- It display a message to the user that retrieved password has been set to the user.

You can easily drag and drop this control on the web a page as shown below.



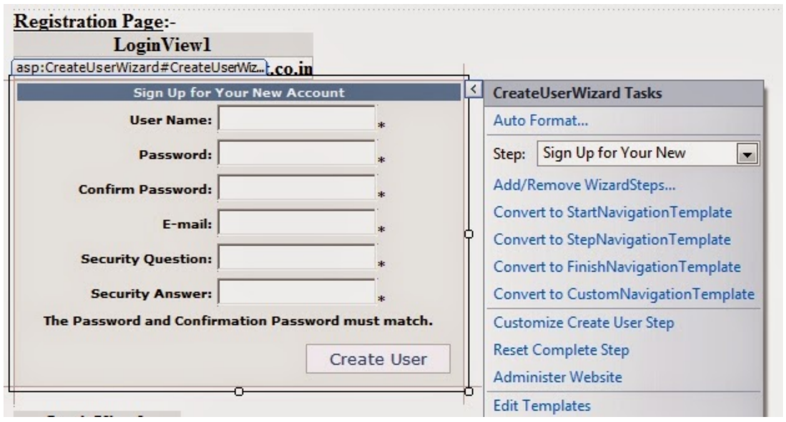
Note :-

- To retrieve and reset password you must set some properties inside the asp.net Membership services.
- You can can learn its methods ,properties and events from PasswordRecovery class yourself.

6.) CreateUserWizard control:-

This control uses the membership service to create a new user in the membership data store.The CreateUserWizard control is provides by the CreateUserWizard class and can be customized by using template and style properties .Using this control any user can easily create an account and login to the web page.

You can drag and drop CreateUserWizared control on the web page as shown below:-


Registration Page:-

LoginView1

asp:CreateUserWizard#CreateUserWiz...

Sign Up for Your New Account

User Name: *

Password: *

Confirm Password: *

E-mail: *

Security Question: *

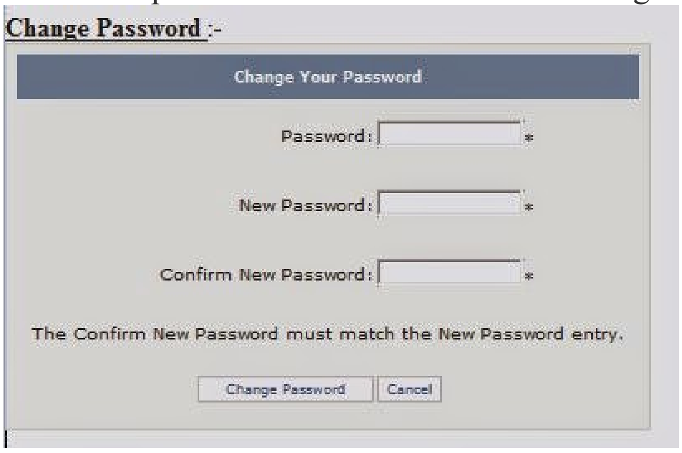
Security Answer: *

The Password and Confirmation Password must match.

Create User
CreateUserWizard Tasks
Auto Format...
Step: Sign Up for Your New
Add/Remove WizardSteps...
Convert to StartNavigationTemplate
Convert to StepNavigationTemplate
Convert to FinishNavigationTemplate
Convert to CustomNavigationTemplate
Customize Create User Step
Reset Complete Step
Administer Website
Edit Templates

7.) The ChangePassword Control:-

Using this control ,user can easily change your existing password (old password) on the ASP.NET Website.This control prompts uses to provide the current password first and then set the new password first and then set the new password.If the old password is not correct then new password can't be set. This is also helps to send the email to the respective users about the new password.This control is used ChangePassword class.


Change Password :-

Change Your Password

Password: *

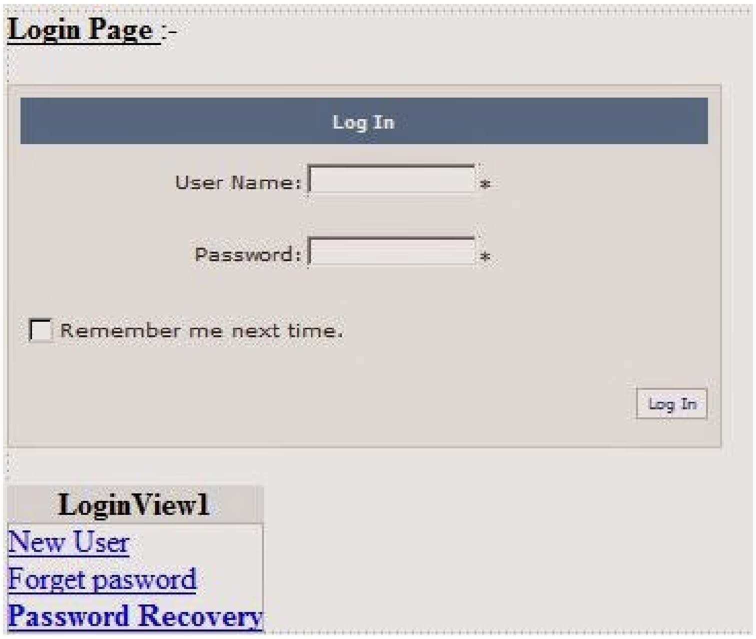
New Password: *

Confirm New Password: *

The Confirm New Password must match the New Password entry.

There are some steps to use Login controls concepts in ASP.NET Application as given below:-

Step 1 :- First open your visual studio -->File -->New -->Select ASP.NET Empty website --> OK -->Open Solution Explorer -->Add a New web form (login.aspx) -->Now drag and Drop Login control and and LoginView control on the page from toolbox --> Add a Hyperlink control in LoginView 's blank space as shown below:-


Login Page :-

Log In

User Name: *

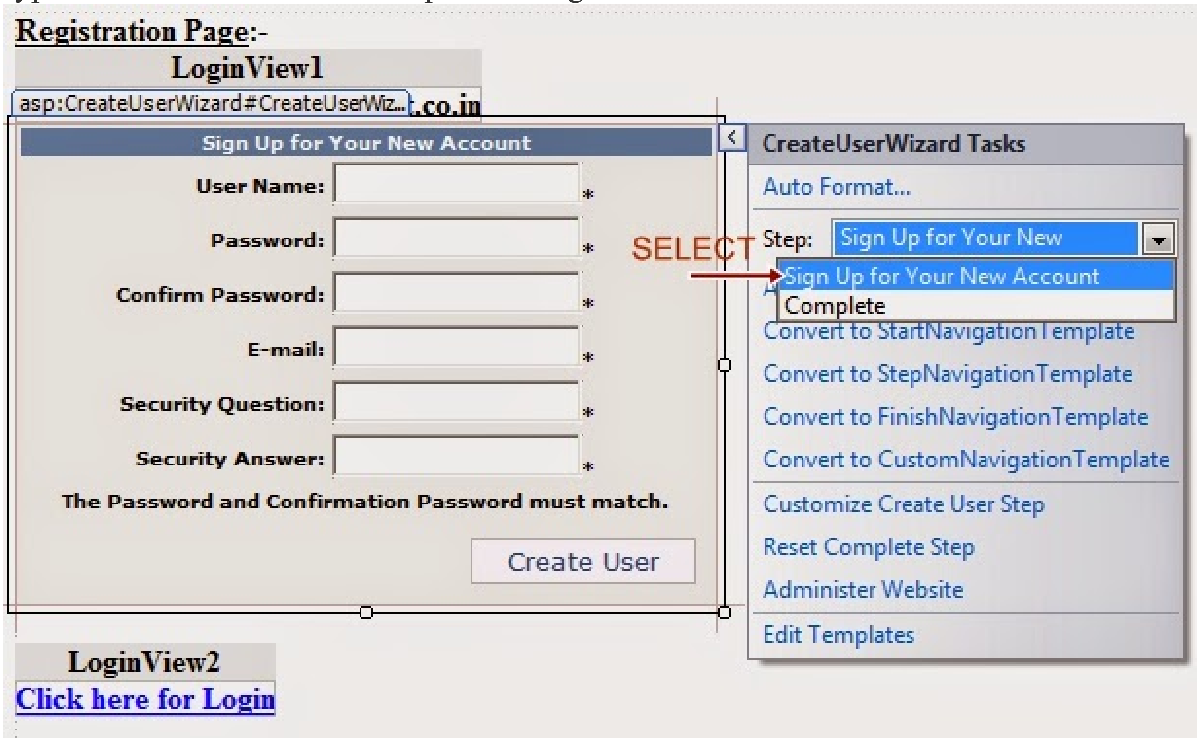
Password: *

Remember me next time.

Log In

LoginView1
New User
Forget pasword
Password Recovery

Step 2 :- Now open Solution Explorer --> Add a New web Form (Registrationpage.aspx) --> Drag and drop CreateUserWizard and LoginView controls on the page --> Put a Hyperlink control inside blank space in LoginView control as shown below:-


Registration Page:-

LoginView1
asp:CreateUserWizard#CreateUserWiz... .co.in

Sign Up for Your New Account

User Name: *

Password: *

Confirm Password: *

E-mail: *

Security Question: *

Security Answer: *

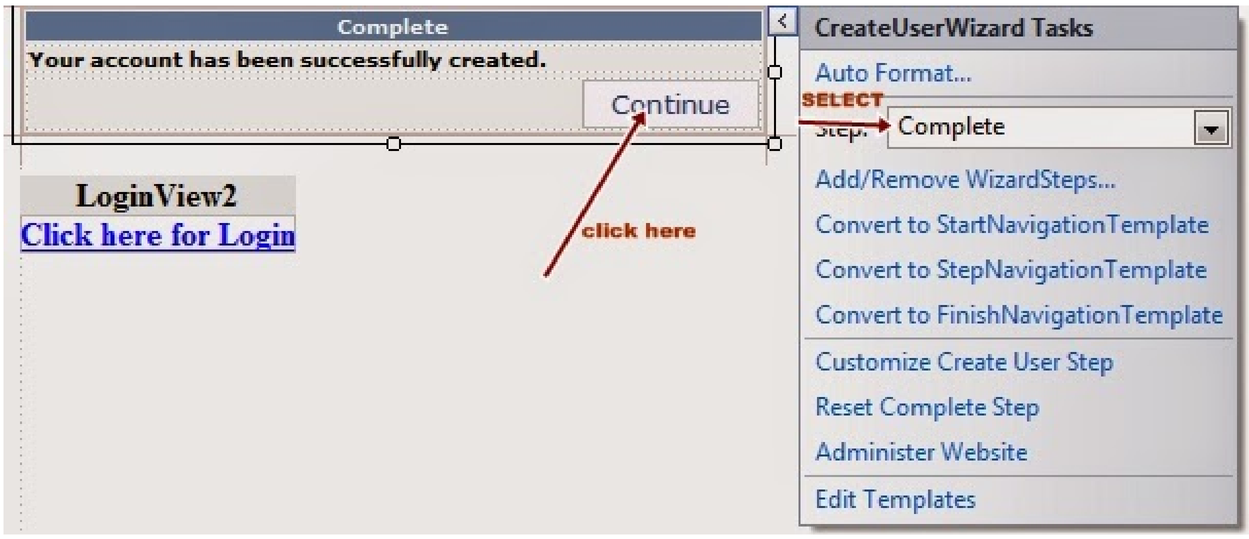
The Password and Confirmation Password must match.

Create User

SELECT
CreateUserWizard Tasks
Auto Format...
Step: Sign Up for Your New
Sign Up for Your New Account
Complete
Convert to StartNavigationTemplate
Convert to StepNavigationTemplate
Convert to FinishNavigationTemplate
Convert to CustomNavigationTemplate
Customize Create User Step
Reset Complete Step
Administer Website
Edit Templates

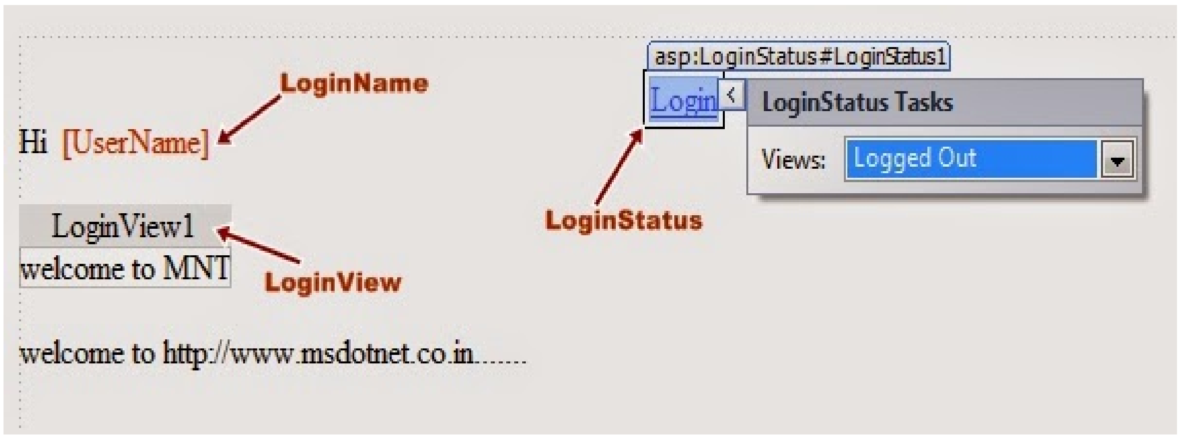
LoginView2
Click here for Login

- Now select **Complete** from createUserWizard Tasks as shown above --> Now double click on Continue button and write the following c# codes for Navigation as given below:-

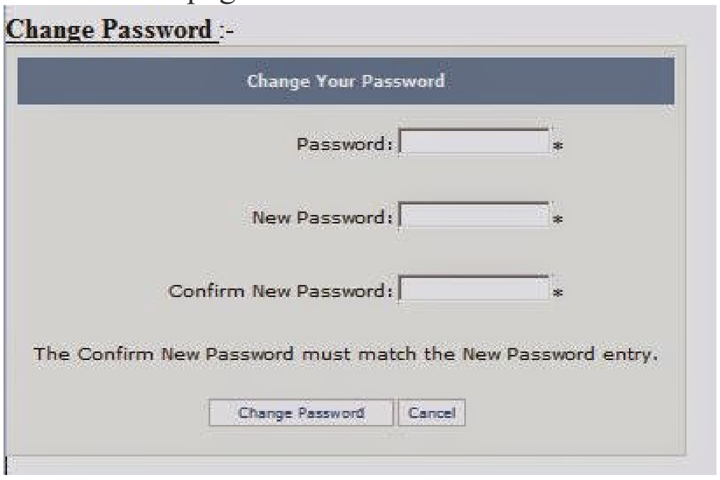

Complete
Your account has been successfully created.
Continue
click here
LoginView2
Click here for Login
CreateUserWizard Tasks
Auto Format...
SELECT
Step: Complete
Add/Remove WizardSteps...
Convert to StartNavigationTemplate
Convert to StepNavigationTemplate
Convert to FinishNavigationTemplate
Customize Create User Step
Reset Complete Step
Administer Website
Edit Templates

NOTE :- You can set this Navigation URL from the properties of **Continue** button also instead of this above codes.

Step 3 :- Now Add a New Web Form (welcomwpage.aspx) --> drag and drop LoginName,LoginStatus and LoginView Controls on the page from toolbox as shown below:-



Step 4 :- Now Add again a New Web Form (changepassword.aspx)-->drag and drop ChangePassword control on the page from toolbox as shown below:-


Change Password :-

Change Your Password

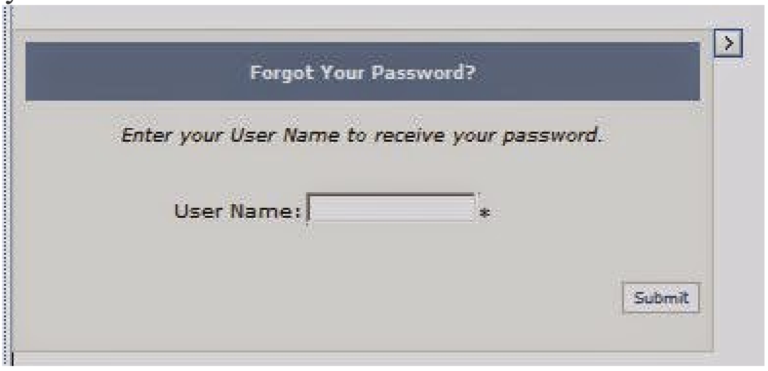
Password: *

New Password: *

Confirm New Password: *

The Confirm New Password must match the New Password entry.

Step 5 :- Now Add a New web form (PasswordRecovery.aspx) -->drag and drop Passwordrecovery control from toolbox as shown below:-


Forgot Your Password?

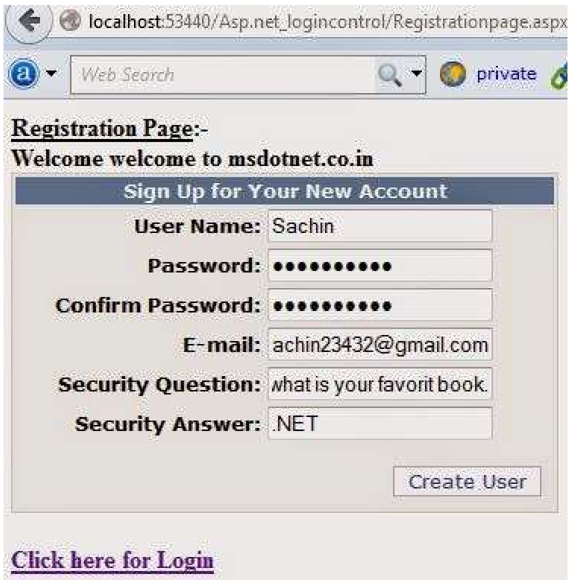
Enter your User Name to receive your password.

User Name: *

Step 6 :- Now open web.config file and write the following codes as given below:-

```
<?xml version="1.0"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
 <system.web>
   <authentication mode="Forms">
  </authentication>
    <compilation debug="true" targetFramework="4.0"/>
 </system.web>
</configuration>
```

Step 7 :- Now Run the Application (Press F5) --> Now create a account first for access the website --> Press Create User button --> After that Press Continue button as shown below:-

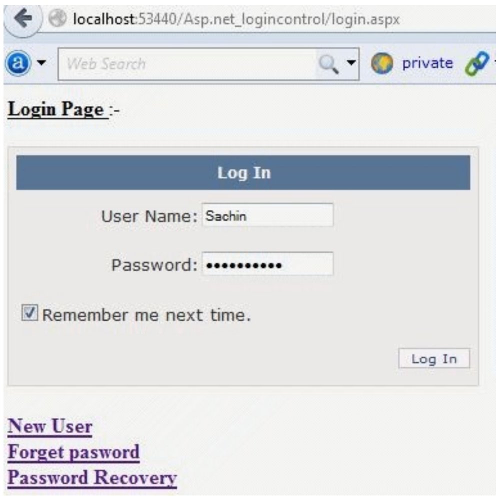

localhost:53440/Asp.net_logincontrol/Registrationpage.aspx

Registration Page:-
Welcome welcome to msdotnet.co.in

Sign Up for Your New Account
User Name: Sachin
Password: ••••••••••
Confirm Password: ••••••••••
E-mail: achin23432@gmail.com
Security Question: what is your favorit book.
Security Answer: .NET
Create User

Click here for Login

Step 8 :- After step 7, login.aspx page will be opened --> Now put login credentials such as username and password --> Press login button --> You will see the following output as shown below:-


localhost:53440/Asp.net_logincontrol/login.aspx

Login Page :-

Log In
User Name: Sachin
Password: ••••••••••
☑ Remember me next time.
Log In

New User
Forget pasword
Password Recovery

Introduction to Master Pages

Master Pages are used when user needs a consistent look and behavior over all web pages in an application. When a user needs to attach header and footer for all the web pages in an application, the master pages are used. Master pages provide a template for all other pages in an application.

The master pages define placeholders for the content, which are overridden for the content. The result is combination of master and content page. Every master page has one or more content pages in an application.

The advantages of the master page are as mentioned below:

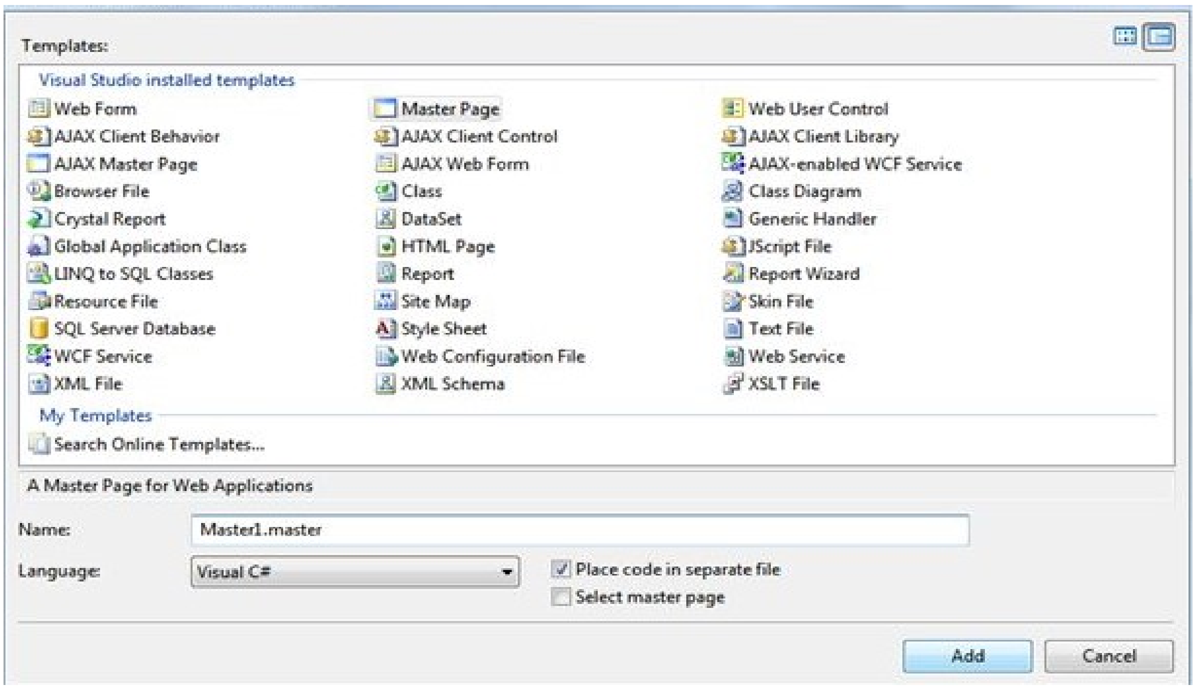
1. They provide an object model allowing users to customize the master page from the individual content pages.
2. They allows user design the rendering of the controls in the placeholder
3. It is centralized with common functionality of all pages to makes updates in one place
4. Code can be applied on one set of controls and the results to the set of pages in the application

The **@Master** directive is defines in the master page. The master page contains one or more **<asp:ContentPlaceHolder>** for an individual content. The id attribute identifies the placeholder from all present in a web page. The master page has **.master** extension. The syntax of the master directive is as shown below:

<%@ Master Language=”C#” CodeFile=”MasterPage.master.cs” Inherits=”MasterPage” %

To create a master page, create an ASP.NET website by clicking ‘File’ > ‘New’ > ‘Website’. Right click on the Project in the solution explorer and click ‘Add New Item’. In the dialog box, choose the ‘Master Page’ and click ‘Add’.
The **‘MasterPage.master’** appears in the solution explorer.

The master page will be displayed as shown below:



Templates:

Visual Studio installed templates

Web Form	Master Page	Web User Control
AJAX Client Behavior	AJAX Client Control	AJAX Client Library
AJAX Master Page	AJAX Web Form	AJAX-enabled WCF Service
Browser File	Class	Class Diagram
Crystal Report	DataSet	Generic Handler
Global Application Class	HTML Page	JScript File
LINQ to SQL Classes	Report	Report Wizard
Resource File	Site Map	Skin File
SQL Server Database	Style Sheet	Text File
WCF Service	Web Configuration File	Web Service
XML File	XML Schema	XSLT File

My Templates

Search Online Templates...

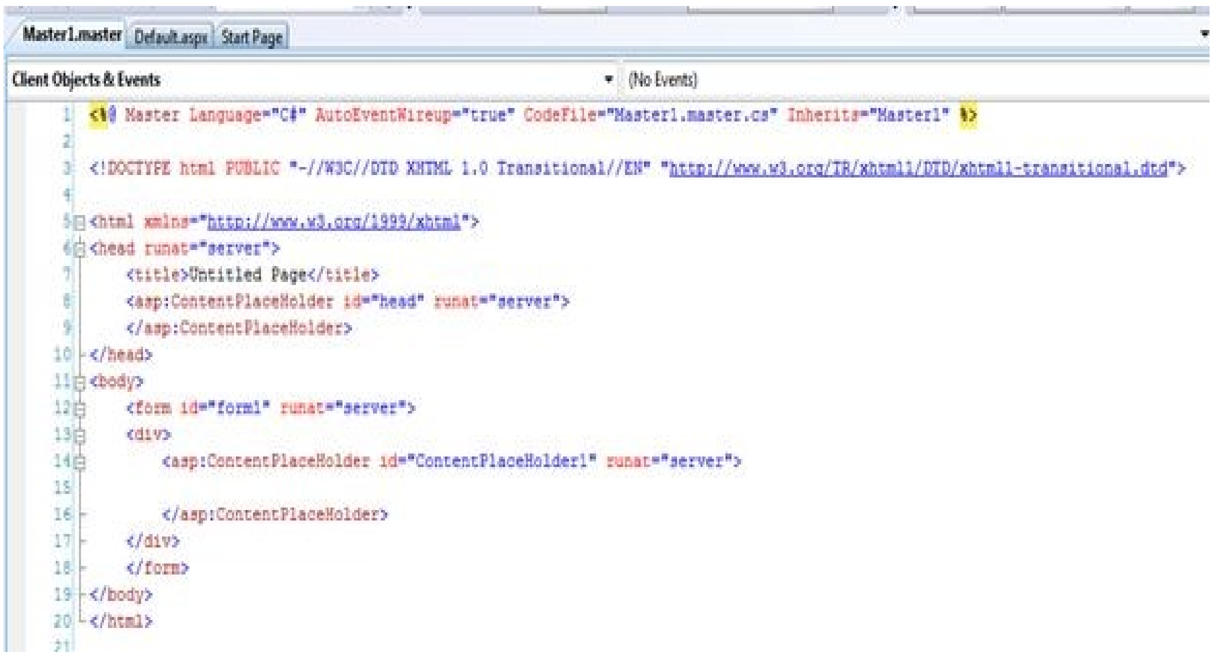
A Master Page for Web Applications

Name: Master1.master

Language: Visual C#

☑ Place code in separate file

☐ Select master page

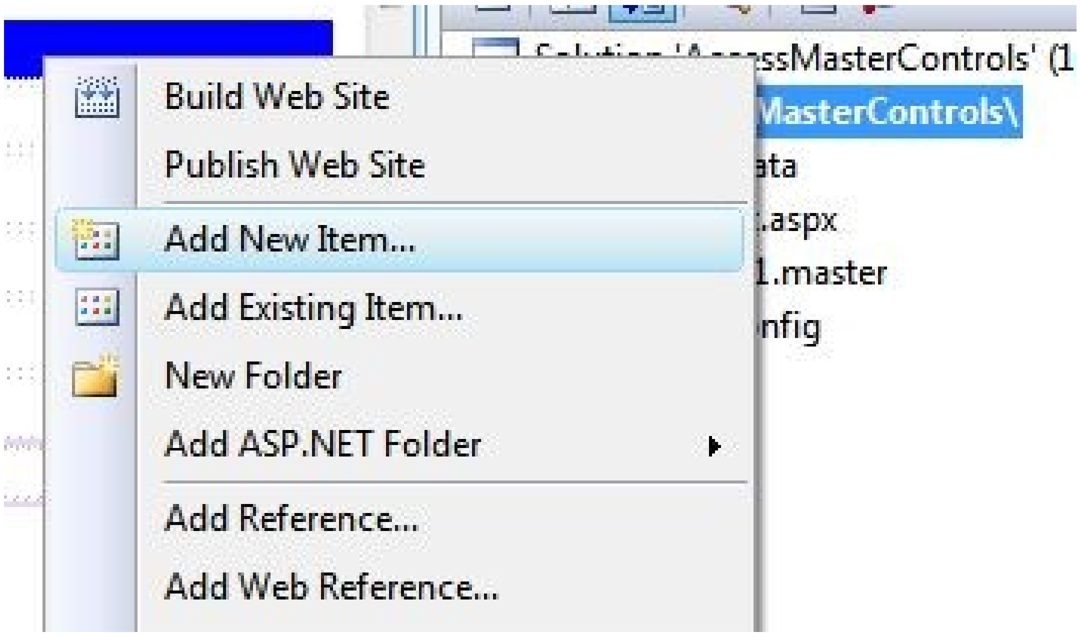


```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="Master1.master.cs" Inherits="Master1" %>

<!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>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

        </asp:ContentPlaceHolder>
    </div>
    </form>
</body>
</html>
```

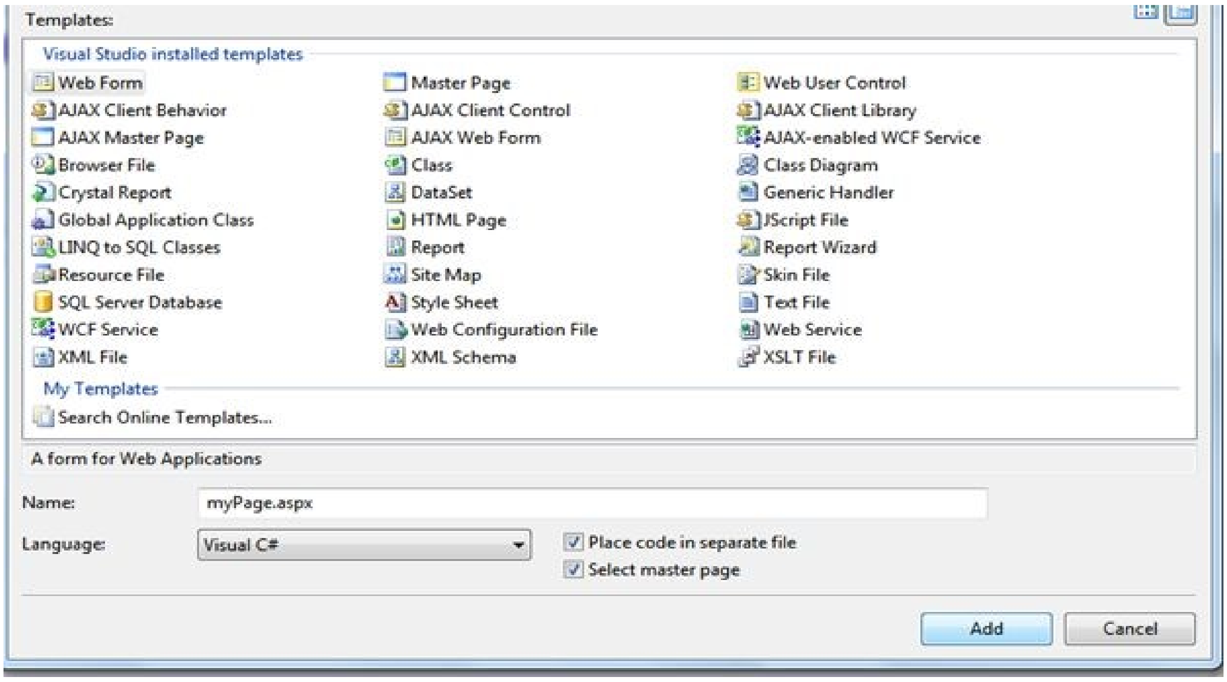
11.2 ContentPlaceHolder and content tags

The steps to add a content page to the application is as shown below:

1. Select the 'Add New Item' from the list.



2) Add the web form page to the application.



A form for Web Applications

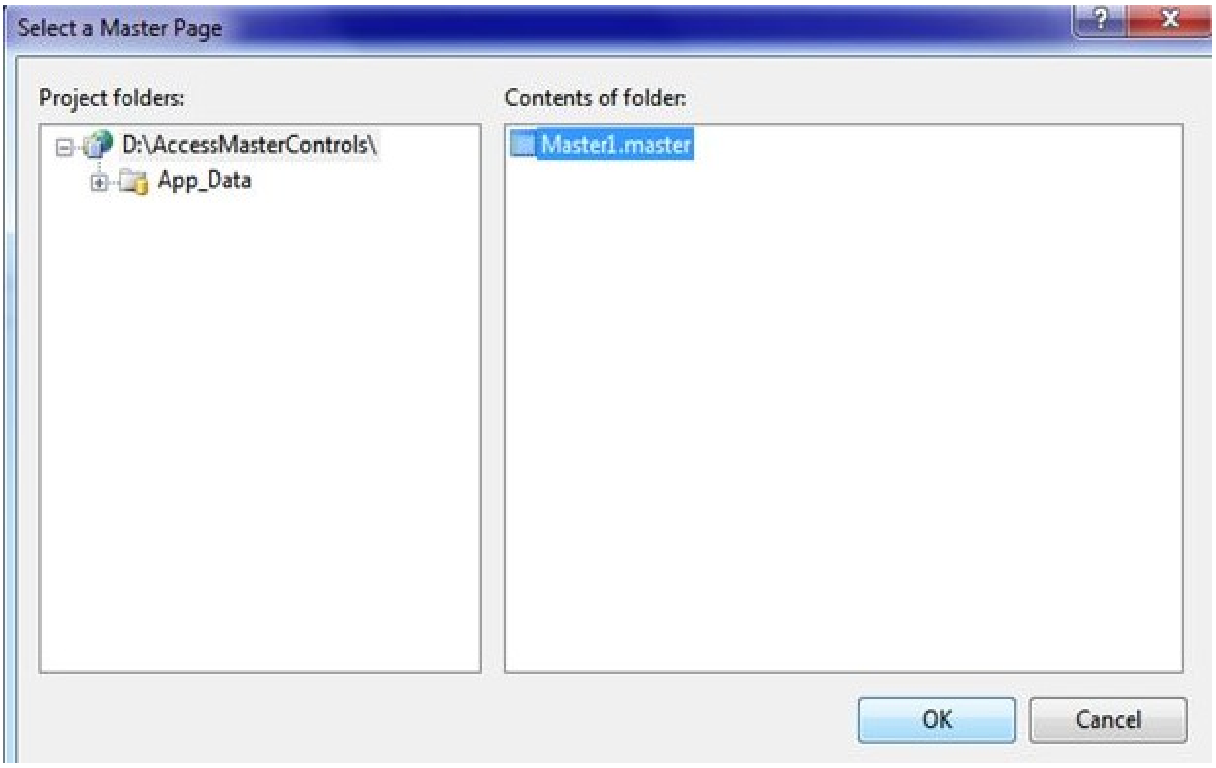
Name: myPage.aspx

Language: Visual C#

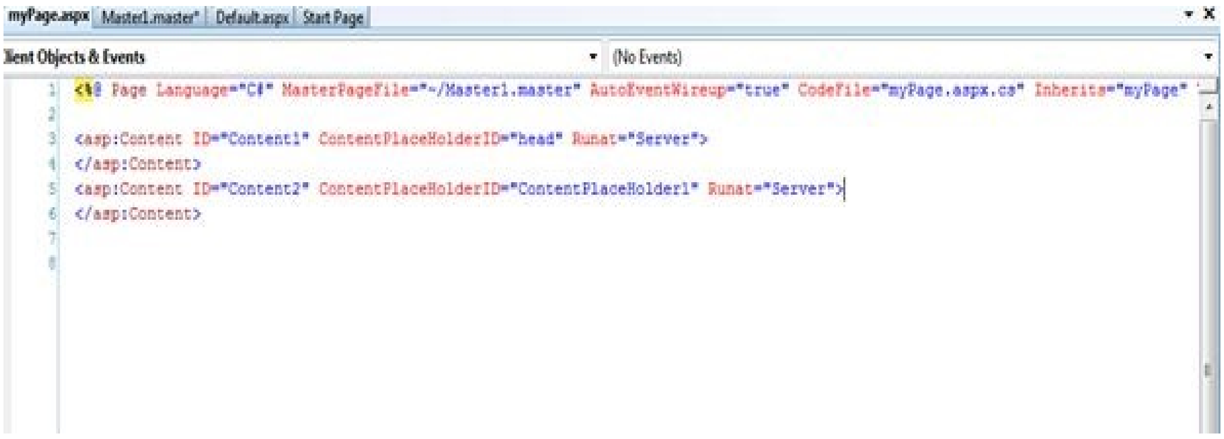
☑ Place code in separate file

☑ Select master page

3) Select a master page for the content page from all the available master pages in an application.



4) The content page in an application is as shown below:



```
<%@ Page Language="C#" MasterPageFile="~/Master1.master" AutoEventWireup="true" CodeFile="myPage.aspx.cs" Inherits="myPage"

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
</asp:Content>
```

The content are used for holding values of the master page placeholder control. The syntax for the content page is as shown below:

```
<%Page Language=”vb” MasterPageFile=”~/MasterPages/Master1.master”
Title=”Content Page %>
```

The **@Page** directive defines as a standard page. The content pages are saved with **page1.aspx** extension. The **MasterPageFile** is the master file which is applied to the content page. The content page binds with the respective master page. The content page looks as shown below:

```
<%Page Language=”vb” MasterPageFile=”~/MasterPages/Master1.master”
Title=”Content Page1 %>
    <asp:Content ID=”Content1” ContentPlaceHolderID=”Main”
runat=”server”>
      Main Content
    </asp:Content>
```

The @Page directive page binds the content page to a specific master page. It defines a title for the page to be merged with the master page. User can create content by adding content controls and mapping them to the contentplaceholder controls on the master page.

ASP.NET - Ad Rotator

The AdRotator control randomly selects banner graphics from a list, which is specified in an external XML schedule file. This external XML schedule file is called the advertisement file.

The AdRotator control allows you to specify the advertisement file and the type of window that the link should follow in the AdvertisementFile and the Target property respectively.

The basic syntax of adding an AdRotator is as follows:

```
<asp:AdRotator  runat = "server" AdvertisementFile = "adfile.xml"  Target =  "_blank" />
```

Before going into the details of the AdRotator control and its properties, let us look into the construction of the advertisement file.

The Advertisement File

The advertisement file is an XML file, which contains the information about the advertisements to be displayed.

Extensible Markup Language (XML) is a W3C standard for text document markup. It is a text-based markup language that enables you to store data in a structured format by using meaningful tags. The term 'extensible' implies that you can extend your ability to describe a document by defining meaningful tags for the application.

XML is not a language in itself, like HTML, but a set of rules for creating new markup languages. It is a meta-markup language. It allows developers to create custom tag sets for special uses. It structures, stores, and transports the information.

Following is an example of XML file:

```
<BOOK>
  <NAME> Learn XML </NAME>
  <AUTHOR> Samuel Peterson </AUTHOR>
  <PUBLISHER> NSS Publications </PUBLISHER>
  <PRICE> $30.00</PRICE>
</BOOK>
```

Like all XML files, the advertisement file needs to be a structured text file with well-defined tags delineating the data. There are the following standard XML elements that are commonly used in the advertisement file:

Element	Description
Advertisements	Encloses the advertisement file.
Ad	Delineates separate ad.
ImageUrl	The path of image that will be displayed.
NavigateUrl	The link that will be followed when the user clicks the ad.
AlternateText	The text that will be displayed instead of the picture if it cannot be displayed.
Keyword	Keyword identifying a group of advertisements. This is used for filtering.
Impressions	The number indicating how often an advertisement will appear.
Height	Height of the image to be displayed.
Width	Width of the image to be displayed.

Apart from these tags, customs tags with custom attributes could also be included. The following code illustrates an advertisement file ads.xml:

```
<Advertisements>
  <Ad>
    <ImageUrl>rose1.jpg</ImageUrl>
    <NavigateUrl>http://www.1800flowers.com</NavigateUrl>
    <AlternateText>
      Order flowers, roses, gifts and more
    </AlternateText>
    <Impressions>20</Impressions>
    <Keyword>flowers</Keyword>
  </Ad>
  <Ad>
    <ImageUrl>rose2.jpg</ImageUrl>
    <NavigateUrl>http://www.babybouquets.com.au</NavigateUrl>
    <AlternateText>Order roses and flowers</AlternateText>
    <Impressions>20</Impressions>
    <Keyword>gifts</Keyword>
  </Ad>
  <Ad>
    <ImageUrl>rose3.jpg</ImageUrl>
    <NavigateUrl>http://www.flowers2moscow.com</NavigateUrl>
    <AlternateText>Send flowers to Russia</AlternateText>
    <Impressions>20</Impressions>
    <Keyword>russia</Keyword>
  </Ad>
  <Ad>
    <ImageUrl>rose4.jpg</ImageUrl>
    <NavigateUrl>http://www.edibleblooms.com</NavigateUrl>
    <AlternateText>Edible Blooms</AlternateText>
    <Impressions>20</Impressions>
    <Keyword>gifts</Keyword>
  </Ad>
</Advertisements>
```

Properties and Events of the AdRotator Class
The AdRotator class is derived from the WebControl class and inherits its properties. Apart from those, the AdRotator class has the following properties:

Properties	Description
AdvertisementFile	The path to the advertisement file.
AlternateTextFeild	The element name of the field where alternate text is provided. The default value is AlternateText.
DataMember	The name of the specific list of data to be bound when advertisement file is not used.
DataSource	Control from where it would retrieve data.
DataSourceID	Id of the control from where it would retrieve data.
Font	Specifies the font properties associated with the advertisement banner control.
ImageUrlField	The element name of the field where the URL for the image is provided. The default value is ImageUrl.
KeywordFilter	For displaying the keyword based ads only.

NavigateUrlField	The element name of the field where the URL to navigate to is provided. The default value is NavigateUrl.
Target	The browser window or frame that displays the content of the page linked.
UniqueID	Obtains the unique, hierarchically qualified identifier for the AdRotator control.

Following are the important events of the AdRotator class:

Events	Description
AdCreated	It is raised once per round trip to the server after creation of the control, but before the page is rendered
DataBinding	Occurs when the server control binds to a data source.
DataBound	Occurs after the server control binds to a data source.
Load	Occurs when the server control is loaded into the Page object.
PreRender	Occurs after the Control object is loaded but prior to rendering.
Unload	Occurs when the server control is unloaded from memory.

Working with AdRotator Control Create a new web page and place an AdRotator control on it.

```
<form id="form1" runat="server">
  <div>
    <asp:AdRotator ID="AdRotator1" runat="server" AdvertisementFile  ="~/ads.xml"
onadcreated="AdRotator1_AdCreated" />
  </div>
</form>
```

The ads.xml file and the image files should be located in the root directory of the web site.
Try to execute the above application and observe that each time the page is reloaded, the ad is changed.

ASP.NET Validation

In this chapter, we will discuss about the data validation in the Web Forms. To perform validation, ASP.NET provides controls that automatically check user input and require no code. We can also create custom validation for our application.

ASP.NET validation controls

Following are the validation controls

Validator	**Description**
CompareValidator	It is used to compare the value of an input control against a value of another input control.
RangeValidator	It evaluates the value of an input control to check the specified range.
RegularExpressionValidator	It evaluates the value of an input control to determine whether it matches a pattern defined by a regular expression.
RequiredFieldValidator	It is used to make a control required.
ValidationSummary	It displays a list of all validation errors on the Web page.

ASP.NET CompareValidator Control

This validator evaluates the value of an input control against another input control on the basis of specified operator.

We can use comparison operators like: less than, equal to, greater than etc.

Note: If the input filed is empty, no validation will be performed.

CompareValidator Properties

Property	Description
AccessKey	It is used to set keyboard shortcut for the control.
TabIndex	The tab order of the control.
BackColor	It is used to set background color of the control.
BorderColor	It is used to set border color of the control.
BorderWidth	It is used to set width of border of the control.
Font	It is used to set font for the control text.
ForeColor	It is used to set color of the control text.
Text	It is used to set text to be shown for the control.
ToolTip	It displays the text when mouse is over the control.
Visible	To set visibility of control on the form.
Height	It is used to set height of the control.
Width	It is used to set width of the control.
ControlToCompare	It takes ID of control to compare with.

ControlToValidate	It takes ID of control to validate.
ErrorMessage	It is used to display error message when validation failed.
Operator	It is used set comparison operator.

Example

Here, in the following example, we are validating user input by using CompareValidator controller. Source code of the example is given below.

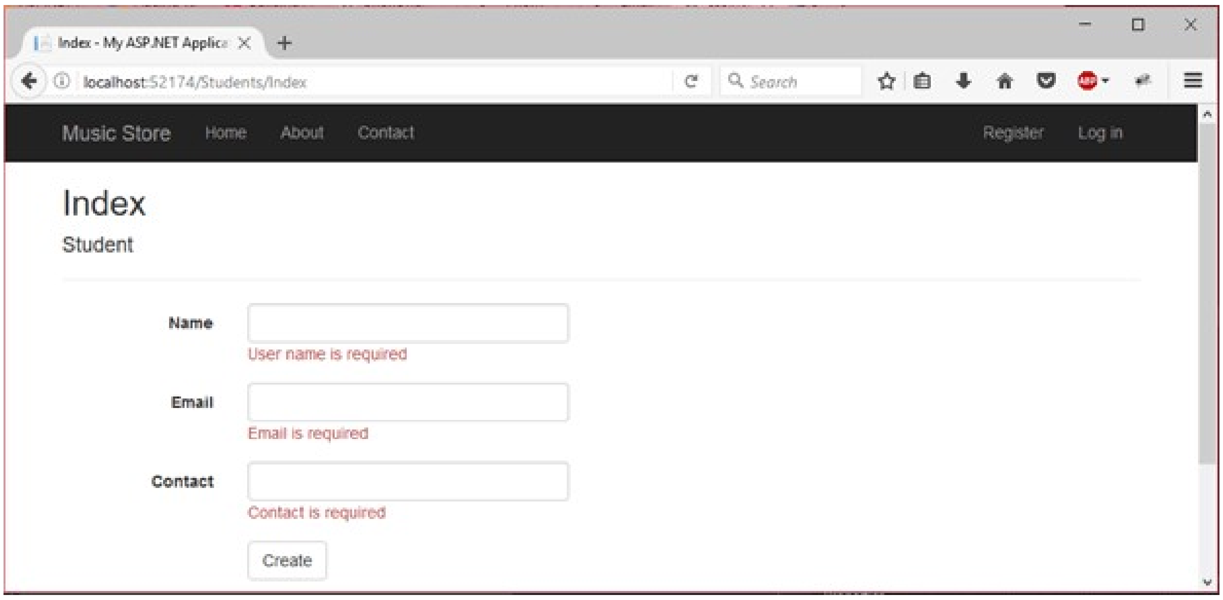
// compare_validator_demo.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="compare_validator_demo.aspx.cs"
Inherits="asp.netexample.compare_validator_demo" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
<style type="text/css">
.auto-style1 {
width: 100%;
        }
.auto-style2 {
height: 26px;
        }
.auto-style3 {
height: 26px;
width: 93px;
        }
.auto-style4 {
width: 93px;
```

```
    }
</style>
</head>
<body>
<form id="form1" runat="server">
<table class="auto-style1">
<tr>
<td class="auto-style3">
            First value</td>
<td class="auto-style2">
<asp:TextBox ID="firstval" runat="server" required="true"></asp:TextBox>
</td>
</tr>
<tr>
<td class="auto-style4">
    Second value</td>
<td>
<asp:TextBox ID="secondval" runat="server"></asp:TextBox>
     It should be greater than first value</td>
</tr>
<tr>
<td class="auto-style4"></td>
<td>
<asp:Button ID="Button1" runat="server" Text="save"/>
</td>
</tr>
</table>
< asp:CompareValidator ID="CompareValidator1" runat="server" ControlToCompare ="secondval"
ControlToValidate="firstval" Display="Dynamic" ErrorMessage="Enter valid value" ForeColor="Red"
Operator="LessThan" Type="Integer"></asp:CompareValidator>
</form>
```

```
51. </body>
52. </html>
```

Output:



Index

Student

Name

User name is required

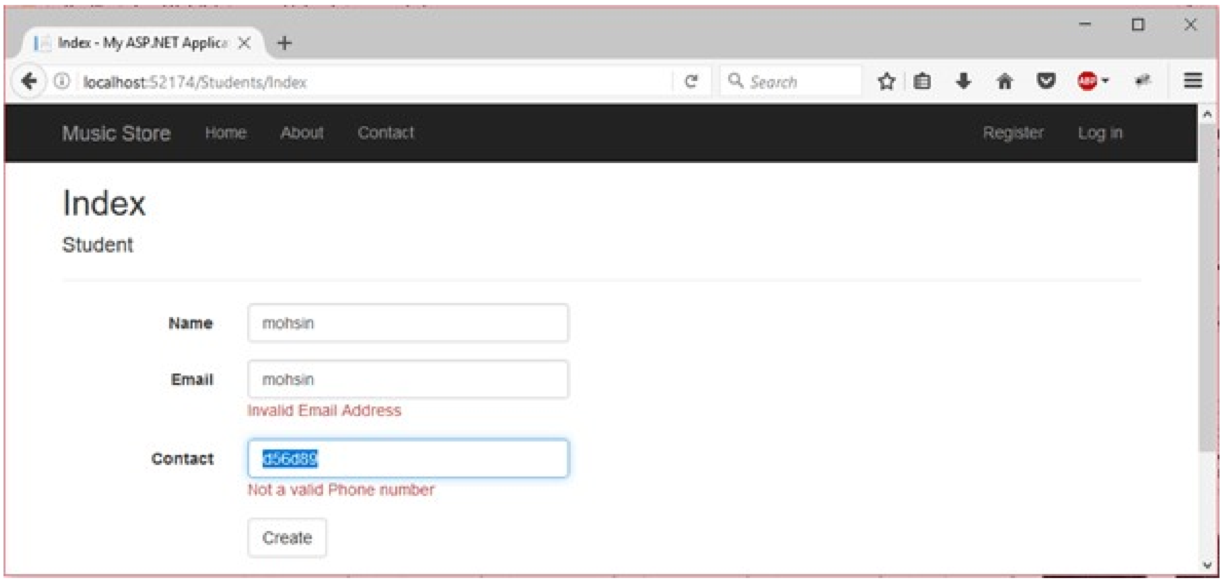
Email

Email is required

Contact

Contact is required

Create



Index

Student

Name: mohsin

Email: mohsin

Invalid Email Address

Contact: d56d89

Not a valid Phone number

Create

ASP.NET RangeValidator Control

This validator evaluates the value of an input control to check that the value lies between specified ranges.

It allows us to check whether the user input is between a specified upper and lower boundary. This range can be numbers, alphabetic characters and dates.

Note: if the input control is empty, no validation will be performed.

The **ControlToValidate**property is used to specify the control to validate. The **MinimumValue** and **MaximumValue** properties are used to set minimum and maximum boundaries for the control.

RangeValidator Properties

Property	**Description**
AccessKey	It is used to set keyboard shortcut for the control.
TabIndex	The tab order of the control.
BackColor	It is used to set background color of the control.
BorderColor	It is used to set border color of the control.
BorderWidth	It is used to set width of border of the control.
Font	It is used to set font for the control text.
ForeColor	It is used to set color of the control text.
Text	It is used to set text to be shown for the control.

ToolTip	It displays the text when mouse is over the control.
Visible	To set visibility of control on the form.
Height	It is used to set height of the control.
Width	It is used to set width of the control.
ControlToValidate	It takes ID of control to validate.
ErrorMessage	It is used to display error message when validation failed.
Type	It is used to set datatype of the control value.
MaximumValue	It is used to set upper boundary of the range.
MinimumValue	It is used to set lower boundary of the range.

Example

In the following example, we are using **RangeValidator** to validate user input in specified range.

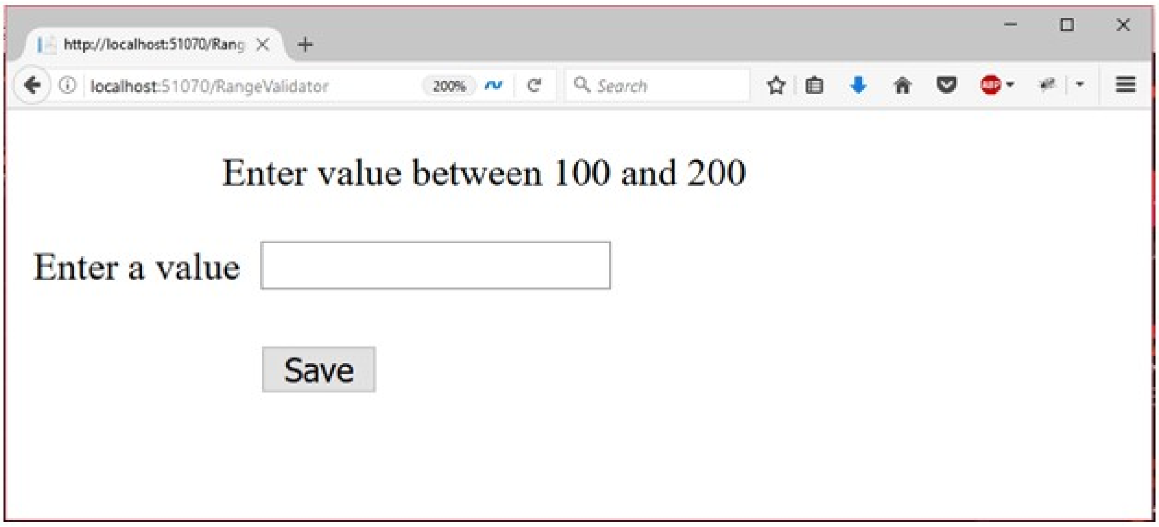
// RangeValidator.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="RangeValidator.aspx.cs"
Inherits="asp.netexample.RangeValidator" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
```

```
<title></title>
<style type="text/css">
.auto-style1 {
height: 82px;
    }
.auto-style2 {
width: 100%;
    }
.auto-style3 {
width: 89px;
    }
.auto-style4 {
margin-left: 80px;
    }
</style>
</head>
<body>
<form id="form1" runat="server">
<div class="auto-style1">
<p class="auto-style4">
        Enter value between 100 and 200<br/>
</p>
<table class="auto-style2">
<tr>
<td class="auto-style3">
<asp:Label ID="Label2" runat="server" Text="Enter a value"></asp:Label>
</td>
<td>
<asp:TextBox ID="uesrInput"runat="server"></asp:TextBox>
<asp:RangeValidator ID="RangeValidator1" runat="server" ControlToValidate="uesrInput"
ErrorMessage="Enter value in specified range" ForeColor="Red" MaximumValue="199"
 MinimumValue="101"
```

```
SetFocusOnError="True"Type=" Integer"></asp:RangeValidator>
</td>
</tr>
<tr>
<td class="auto-style3"> </td>
<td>
<br/>
<asp:Button ID="Button2" runat="server" Text="Save"/>
</td>
</tr>
</table>
<br/>
<br/>
</div>
</form>
</body>
</html>
```

Output:

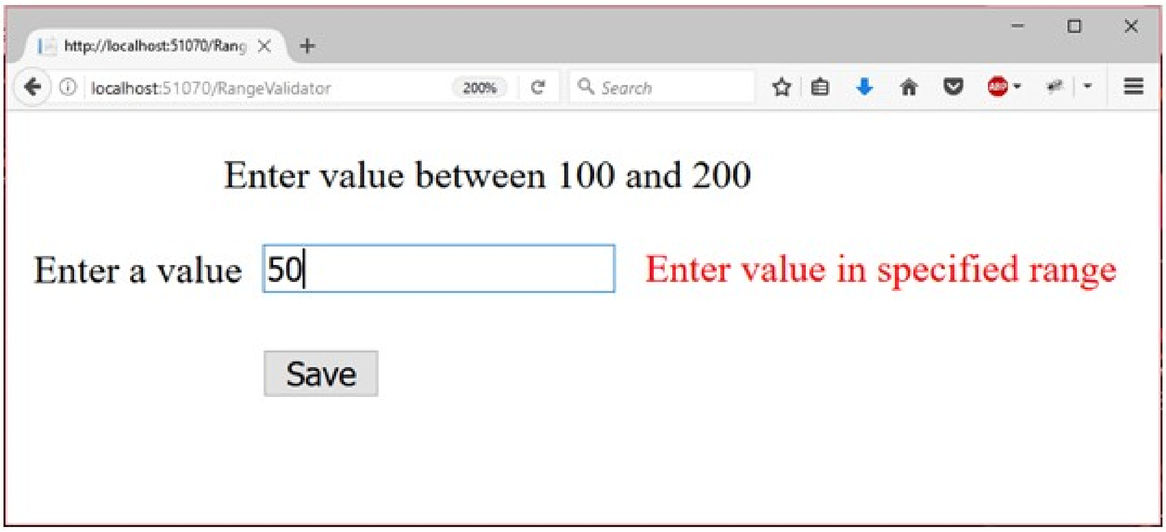


Enter value between 100 and 200

Enter a value

Save

It throws an error message when the input is not in range.



Enter value between 100 and 200

Enter a value 50 Enter value in specified range

Save

RegularExpressionValidator Control

This validator is used to validate the value of an input control against the pattern defined by a regular expression.

It allows us to check and validate predictable sequences of characters like: e-mail address, telephone number etc.

The **ValidationExpression** property is used to specify the regular expression, this expression is used to validate input control.

RegularExpression Properties

Property	Description
AccessKey	It is used to set keyboard shortcut for the control.
BackColor	It is used to set background color of the control.
BorderColor	It is used to set border color of the control.

Font	It is used to set font for the control text.
ForeColor	It is used to set color of the control text.
Text	It is used to set text to be shown for the control.
ToolTip	It displays the text when mouse is over the control.
Visible	To set visibility of control on the form.
Height	It is used to set height of the control.
Width	It is used to set width of the control.
ErrorMessage	It is used to set error message that display when validation fails.
ControlToValidate	It takes ID of control to validate.
ValidationExpression	It is used to set regular expression to determine validity.

Example

Here, in the following example, we are explaining how to use RegularExpressionValidator control to validate the user input against the given pattern.

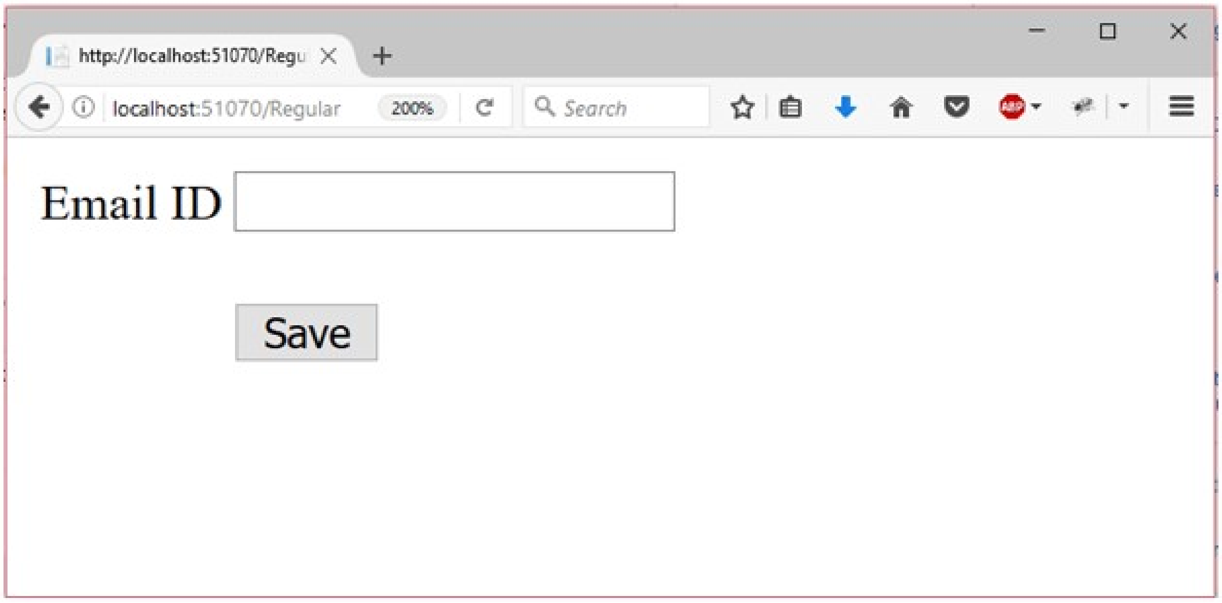
// RegularExpressionDemo.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="RegularExpression
Demo.aspx.cs"
Inherits="asp.netexample.RegularExpressionDemo" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
```

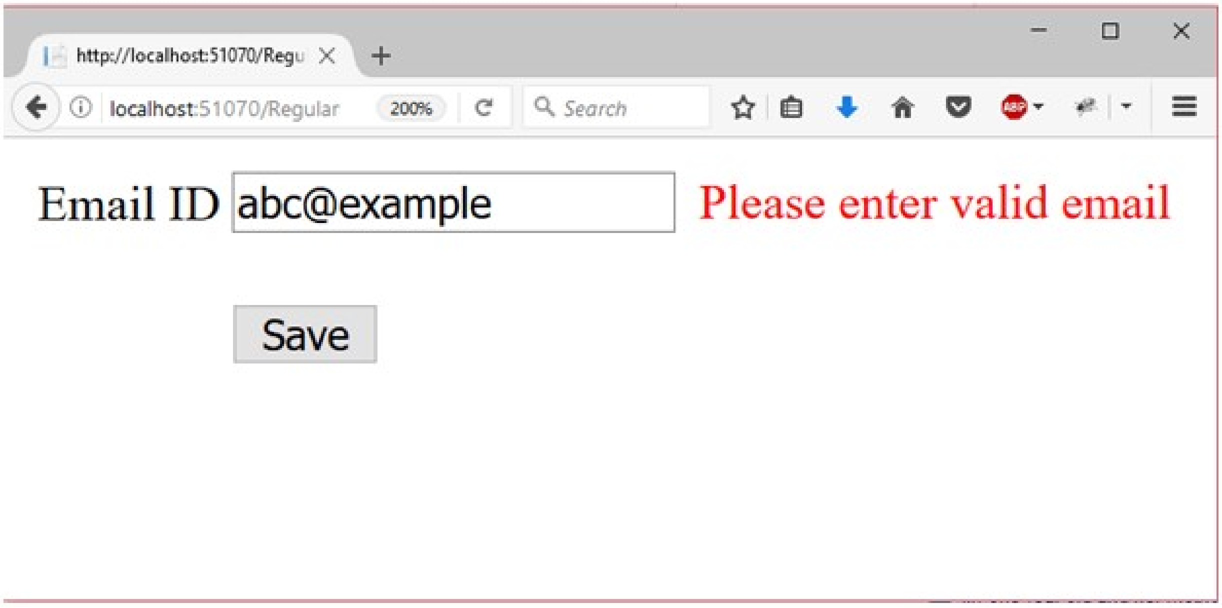
```
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
<table class="auto-style1">
<tr>
<td class="auto-style2">Email ID</td>
<td>
<asp:TextBox ID="username" runat="server"></asp:TextBox>
<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"ControlToValidate="username"
ErrorMessage="Please enter valid email" ForeColor="Red"ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*">
</asp:RegularExpressionValidator>
</td>
</tr>
<tr>
<td class="auto-style2"></td>
<td>
<br/>
<asp:Button ID="Button1" runat="server" Text="Save"/>
</td>
</tr>
</table>
</div>
</form>
</body>
</html>
```

Output:

It produces the following output when view in the browser.



It will validate email format as we specified in regular expression. If validation fails, it throws an error message.



Email ID abc@example Please enter valid email

RequiredFieldValidator Control

This validator is used to make an input control required. It will throw an error if user leaves input control empty.

It is used to mandate form control required and restrict the user to provide data.

Note: It removes extra spaces from the beginning and end of the input value before validation is performed.

The ControlToValidateproperty should be set with the ID of control to validate.

RequiredFieldValidator Properties

Property	Description
AccessKey	It is used to set keyboard shortcut for the control.
BackColor	It is used to set background color of the control.
BorderColor	It is used to set border color of the control.
Font	It is used to set font for the control text.
ForeColor	It is used to set color of the control text.
Text	It is used to set text to be shown for the control.
ToolTip	It displays the text when mouse is over the control.
Visible	To set visibility of control on the form.
Height	It is used to set height of the control.
Width	It is used to set width of the control.
ErrorMessage	It is used to set error message that display when validation fails.

ControlToValidate	It takes ID of control to validate.

Example

Here, in the following example, we are explaining **RequiredFieldValidator** control and creating to mandatory TextBox controls.

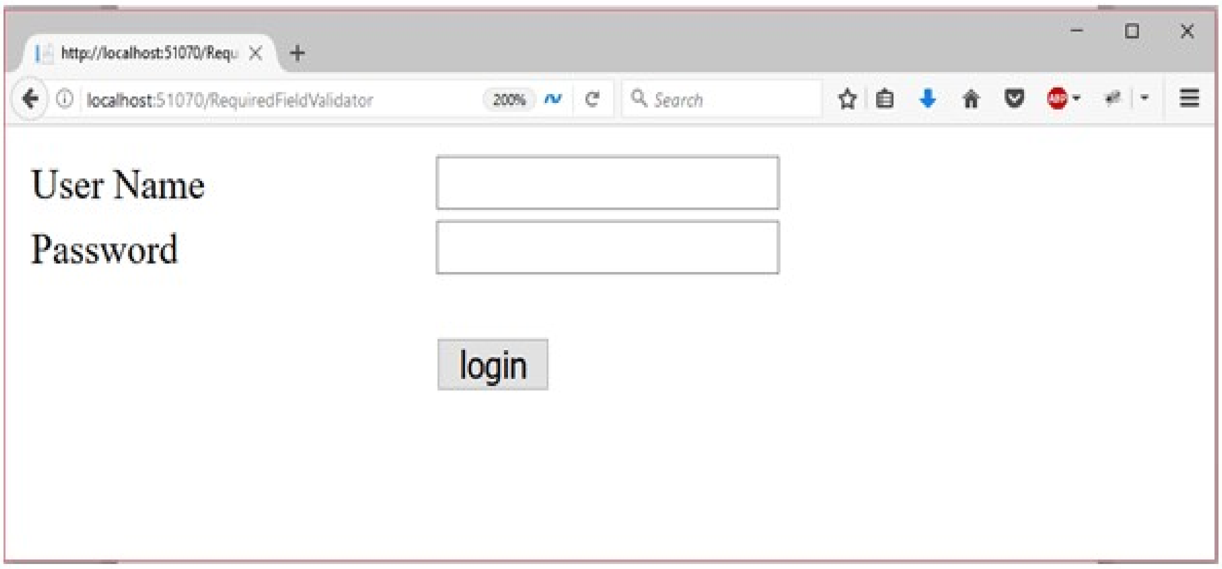
// RequiredFieldValidator.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="RequiredFieldValid
ator.aspx.cs"
Inherits="asp.netexample.RequiredFieldValidator" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
<style type="text/css">
.auto-style1 {
width: 100%;
        }
.auto-style2 {
width: 165px;
        }
</style>
</head>
<body>
<form id="form1" runat="server">
<div>
</div>
<table class="auto-style1">
<tr>
<td class="auto-style2">User Name</td>
<td>
```

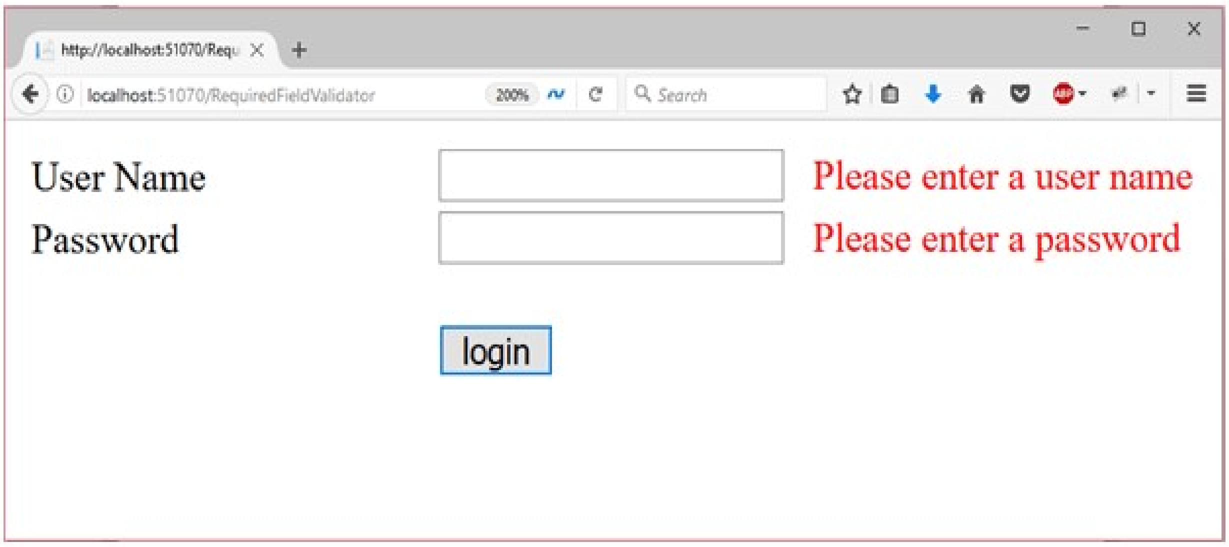
```
<asp:TextBox ID="username" runat="server"></asp:TextBox>
<asp:RequiredFieldValidatorIDasp:RequiredFieldValidatorID="user" runat="server" ControlToValidate="username"
ErrorMessage="Please enter a user name" ForeColor="Red"></asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td class="auto-style2">Password</td>
<td>
<asp:TextBox ID="password" runat="server"></asp:TextBox>
<asp:RequiredFieldValidator ID="pass" runat="server" ControlToValidate="password" ErrorMessage="Please enter a password"
ForeColor="Red"></asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td class="auto-style2"> </td>
<td>
<br/>
<asp:Button ID="Button1" runat="server" Text="login"/>
</td>
</tr>
</table>
</form>
</body>
</html>
```

Output:

It produces the following output when view in the browser.


http://localhost:51070/Requ
localhost:51070/RequiredFieldValidator
200%
Search
User Name
Password
login

It throws error messages when user login with empty controls.


http://localhost:51070/Requ
localhost:51070/RequiredFieldValidator
200%
Search
User Name
Password
login

Please enter a user name

Please enter a password

ASP.NET ValidationSummary Control

This validator is used to display list of all validation errors in the web form.

It allows us to summarize the error messages at a single location.

We can set **DisplayMode** property to display error messages as a list, bullet list or single paragraph.

ValidationSummary Properties

This control has following properties.

Property	**Description**
AccessKey	It is used to set keyboard shortcut for the control.
BackColor	It is used to set background color of the control.
BorderColor	It is used to set border color of the control.
Font	It is used to set font for the control text.
ForeColor	It is used to set color of the control text.
Text	It is used to set text to be shown for the control.
ToolTip	It displays the text when mouse is over the control.
Visible	To set visibility of control on the form.
Height	It is used to set height of the control.
Width	It is used to set width of the control.
ShowMessageBox	It displays a message box on error in up-level browsers.

ShowSummary	It is used to show summary text on the form page.
ShowValidationErrors	It is used to set whether the validation summary should be shown or not.

Example

The following example explains how to use **ValidationSummery** control in the application.

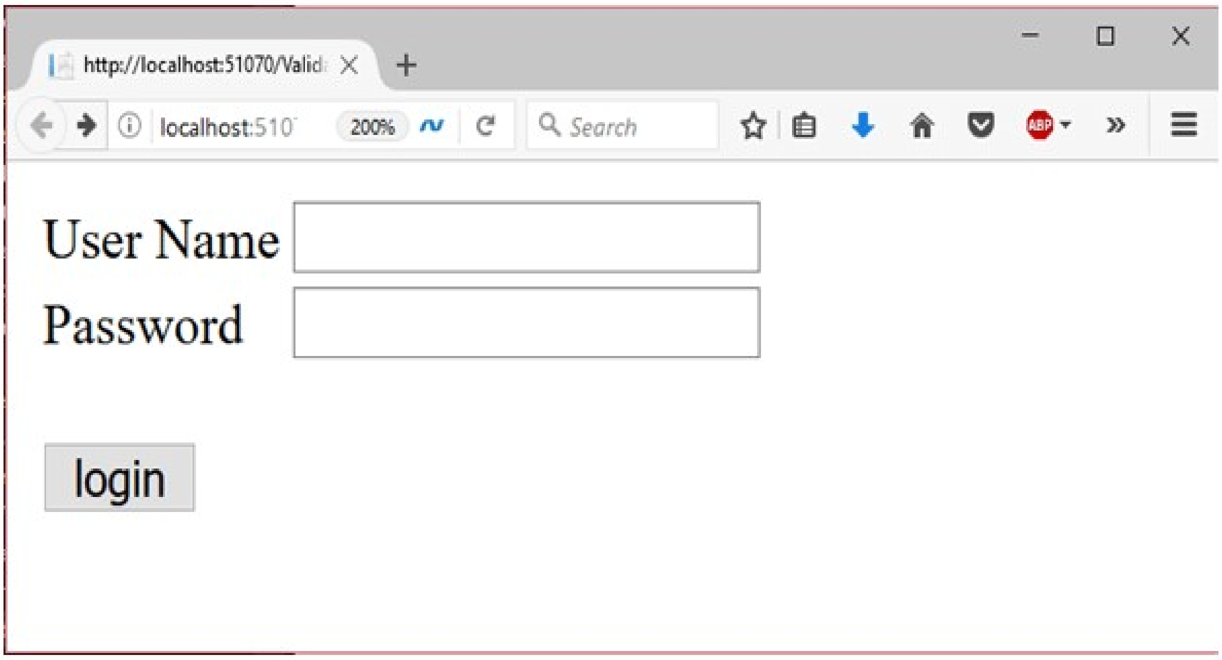
// ValidationSummeryDemo.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="ValidationSummeryDemo.aspx.cs"
Inherits="asp.netexample.ValidationSummeryDemo" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
</div>
<table class="auto-style1">
<tr>
<td class="auto-style2">User Name</td>
<td>
<asp:TextBox ID="username" runat="server"></asp:TextBox>
<asp:RequiredFieldValidator ID="user" runat="server" ControlToValidate="username"
ErrorMessage="Please enter a user name" ForeColor="Red">*</asp:RequiredFieldValidator>
```

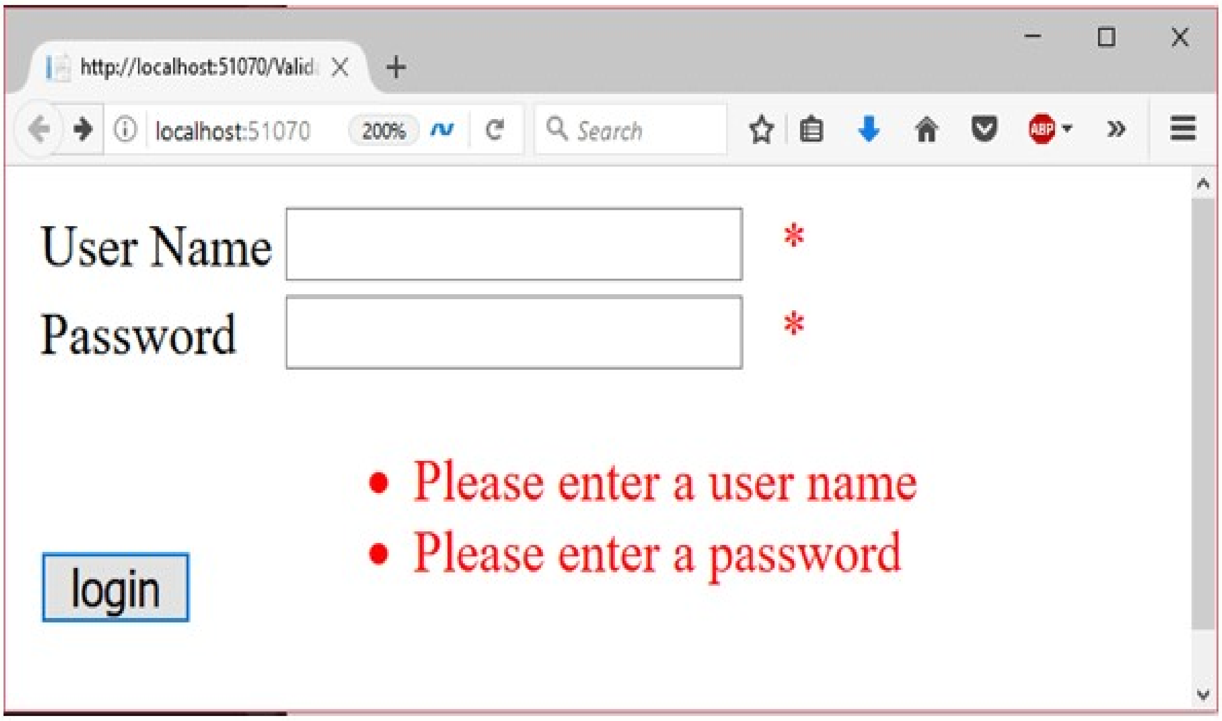
```
</td>
</tr>
<tr>
<td class="auto-style2">Password</td>
<td>
<asp:TextBox ID="password" runat="server"></asp:TextBox>
<asp:RequiredFieldValidator ID="pass" runat="server" ControlToValidate="password"
ErrorMessage="Please enter a password" ForeColor="Red">*</asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td class="auto-style2">
<br/>
<asp:Button ID="Button1" runat="server"Text="login"/>
</td>
<td>
<asp:ValidationSummary ID="ValidationSummary1" runat="server" ForeColor="Red" />
<br/>
</td>
</tr>
</table>
</form>
</body>
</html>
```

Output:

It produces the following output when view in the browser.



It throws error summary when user login without credentials.



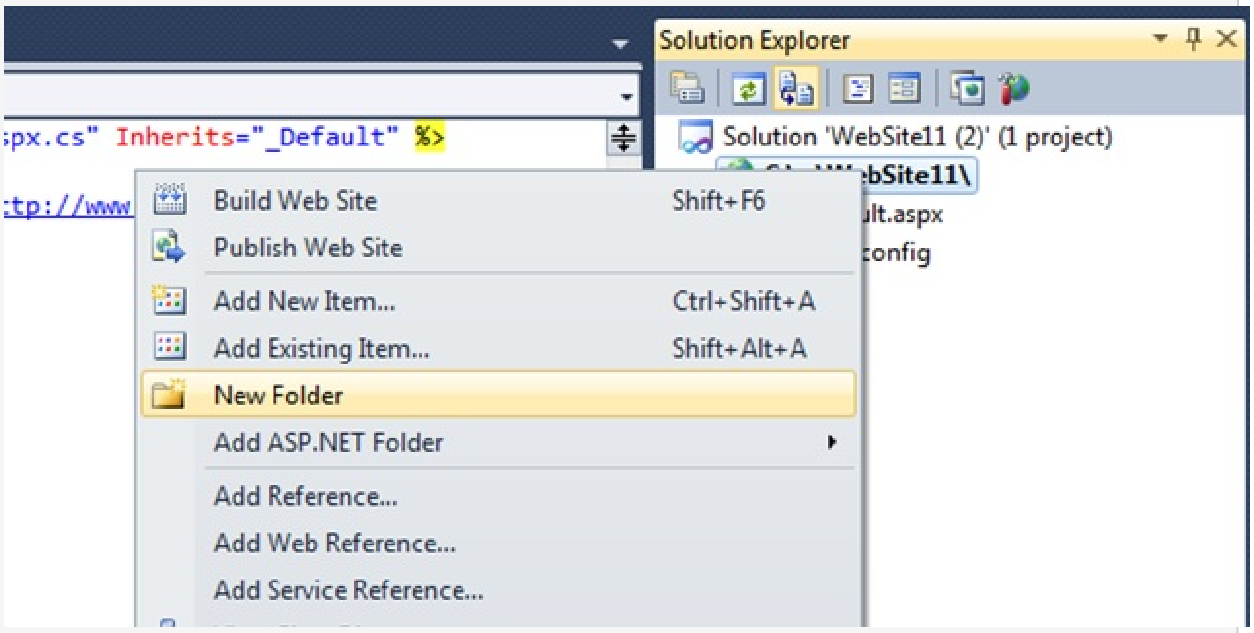
- Please enter a user name
- Please enter a password

What is Themes

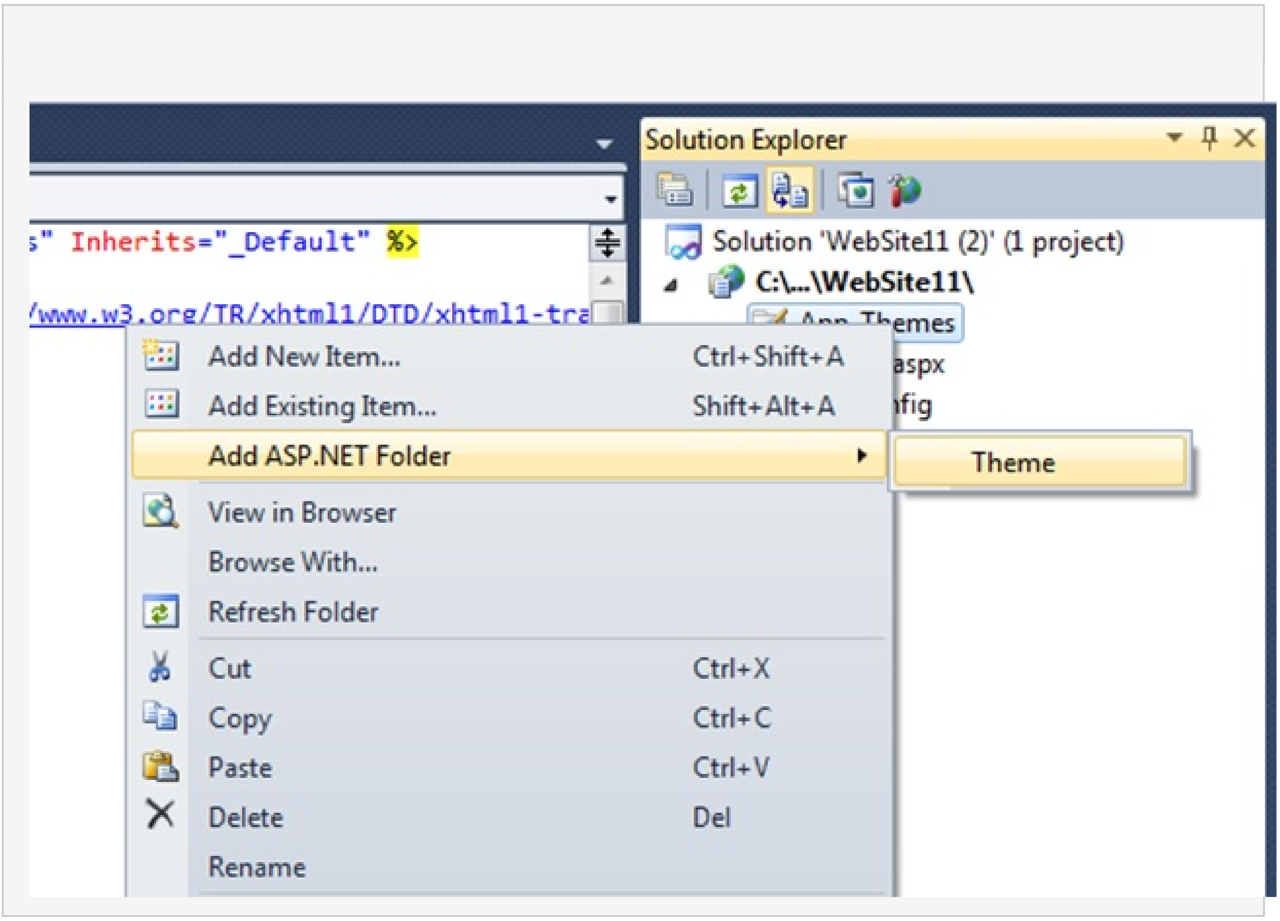
Sometimes we need to change the layout of application for different-different user. For example if a single application is used by different-different client and every client has his own color schema, logo, font etc. requirements then we use themes **A theme is a collection of skin files, css, graphics, images etc**.

How to Apply Themes

To add theme we have to first create the theme folder using name App_Themes.



Now add theme in the App_Themes folder by clicking on right on this folder and go to Add Asp.net folder and select theme. And give it proper name. I created two theme named Client1 and Client2



Skin files in Themes

Skin files are those files in which we define the common property setting for asp.net controls like button, texbox, label etc. The extension of skin files is .skin. It comes under the themes.

Type of Skin

There are two type of skin in asp.net and these are as follows:-

Default Skin

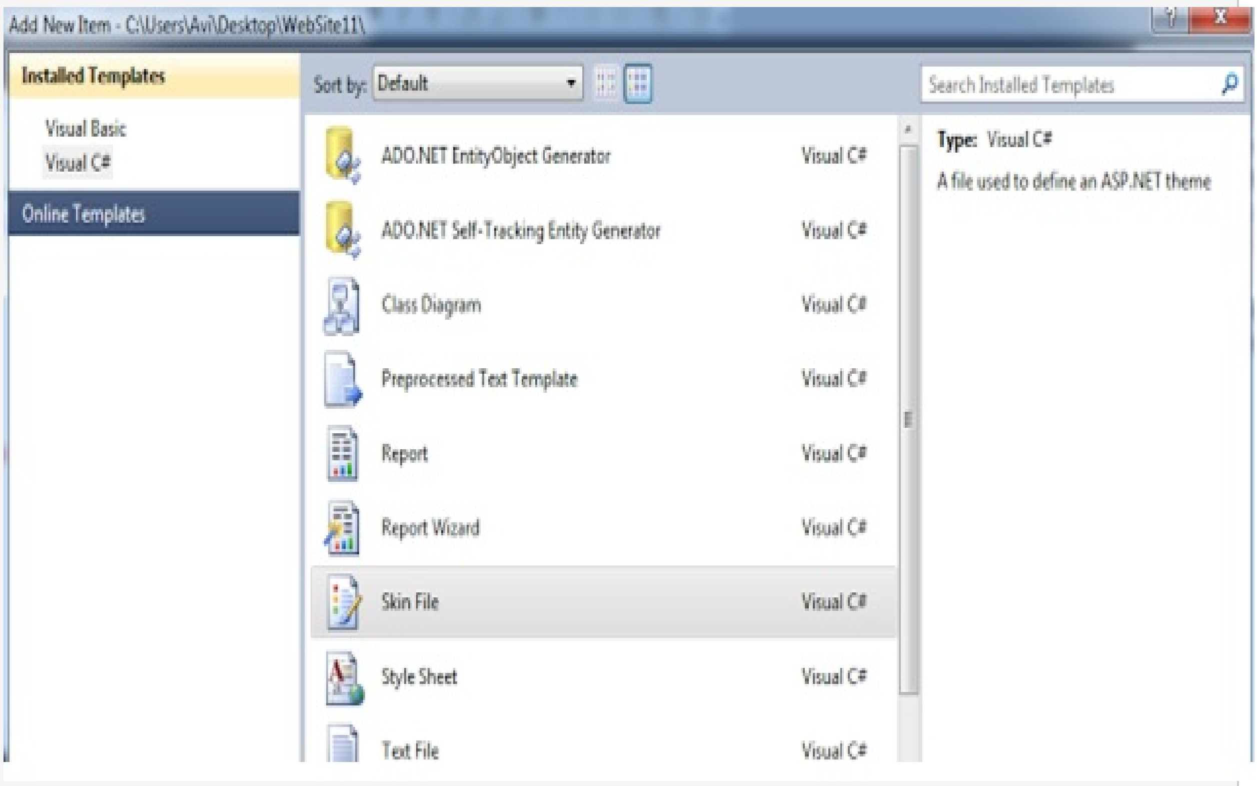
When we want to apply same property for all controls in the page then we use default skin. This skin automatically applies to the control which defined in the skin. This type of skin does not contain the SkinId property.

Named skin

A named skin is applied to those controls in which we pass skin id. A named skin contains the SkinId.

Example of Skin:-

Right click on the theme in which you want to add skin file and give it proper name.


Add New Item - C:\Users\Avi\Desktop\WebSite11\
Installed Templates
Visual Basic
Visual C#
Online Templates
Sort by: Default
Search Installed Templates

ADO.NET EntityObject Generator	Visual C#
ADO.NET Self-Tracking Entity Generator	Visual C#
Class Diagram	Visual C#
Preprocessed Text Template	Visual C#
Report	Visual C#
Report Wizard	Visual C#
Skin File	Visual C#
Style Sheet	Visual C#
Text File	Visual C#

Type: Visual C#

A file used to define an ASP.NET theme

mpty the skin file and add your own code in this file. For example I added skin for button where I add the back color property which is as follows:-

```
<asp:Button runat="server"  BackColor="Red"  />
```

The output of this code as follows:-

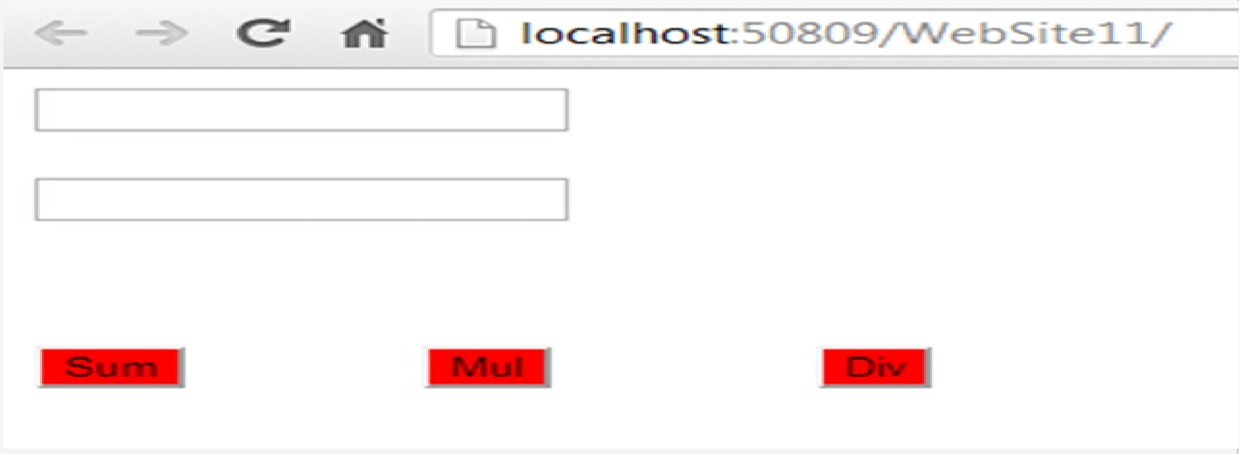


Figure 4

This is the example of Default skin file.

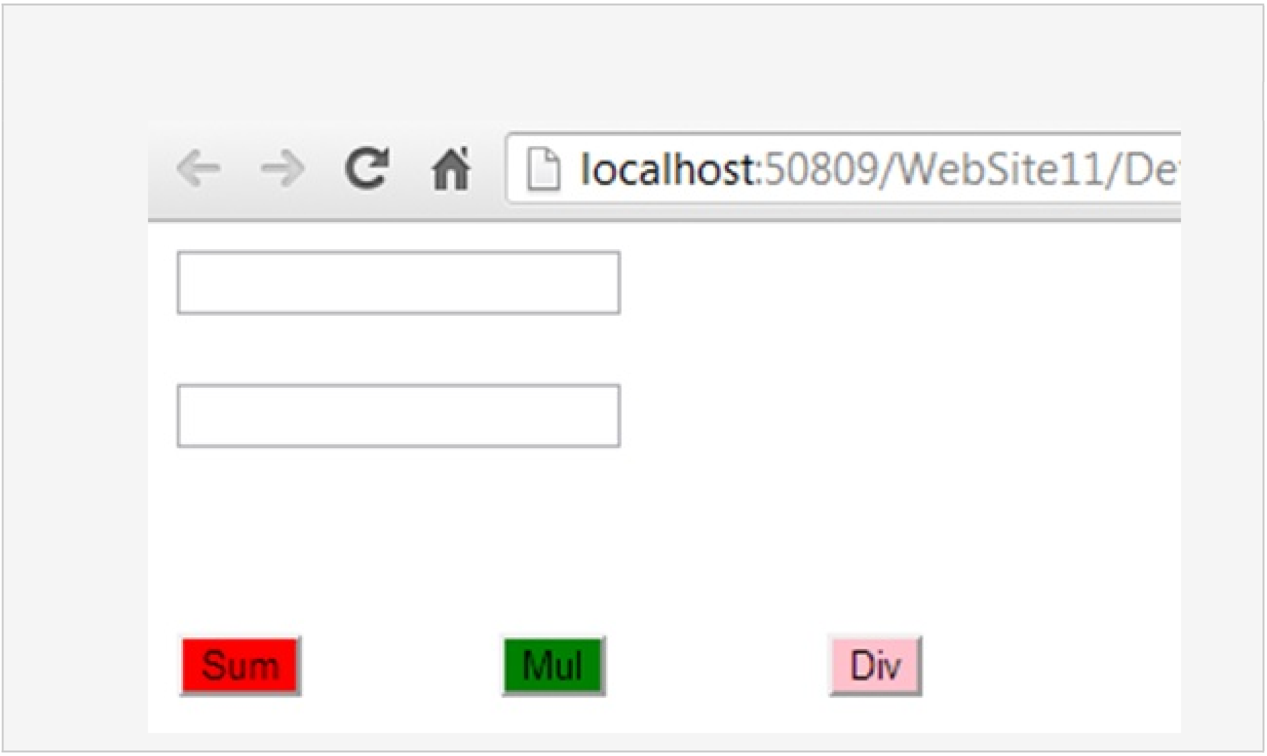
Now add the control property using skinId which is as follows:-

```
<asp:Button runat="server"  BackColor="Red"  skinid="sk1"  />
<asp:Button runat="server"  BackColor="Green"  skinid="sk2"  />
<asp:Button runat="server"  BackColor="Pink"  skinid="sk3"  />
```

Now apply these skin ids to the buttons in the form which is as follows:-

```
<asp:Button ID="Button1" runat="server" Text="Sum" SkinID="sk1" />
<asp:Button ID="Button2" runat="server" Text="Mul" SkinID="sk2" />
<asp:Button ID="Button3" runat="server" Text="Div" SkinID="sk3" />
```

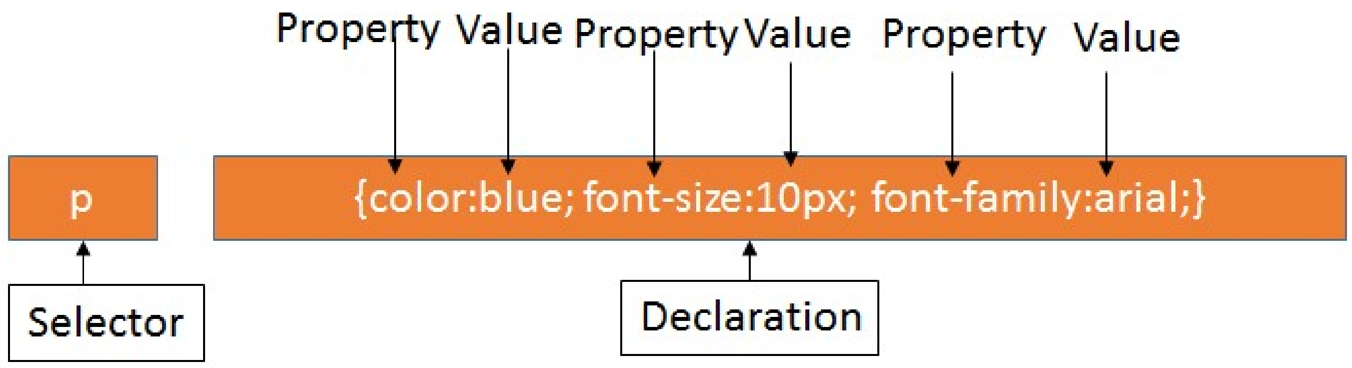
The output of this code as follows:-



CSS Introduction

CSS (Cascading Style Sheets) is used for defining styles to display the elements written in a markup language. It helps user separate the HTML or XHTML content from it style. The separation provides flexibility, faster accessibility to content, reduces complexity and repetition of data.

The syntax for CSS has two parts, a selector and one or more declarations. The selector contains HTML element for which the style is to added. The declaration contains property and a value associated with it. The pictorial representation of the syntax is shown below:



CSS comments: Comments are used for explaining codes. Comments are ignored by the browser for execution. The comments begin with '/*' and ends with '*/'. A sample code after adding comments is shown below:

?

```
h1
{
   /* Set the font with the specified style*/
   text-font : arial;
   /*Set the background color*/
   color:red;
}
```

There are two types of style sheets declaration used for designing. The list of details are explained below:

- **Internal Style Sheet**
- **External Style Sheet**
- **Inline Style Sheet**

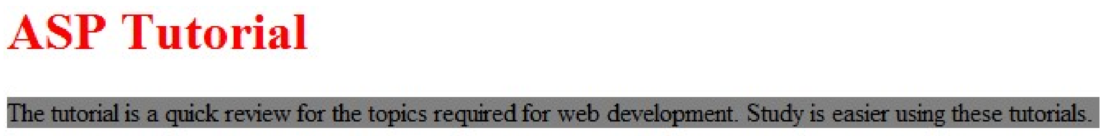
***Internal Style Sheet*:** The style is defined in the <head> section of the web page using the <style> tag. The styles defined are limited to the particular web page in which it is declared. User must create a new style for every new page added in the document. Internal Styles are also known as 'Embedded' styles.

A code sample of internal style sheet is mentioned below:

?

```
<html xmlns=”http://www.w3.org/1999/xhtml”>
 <head runat=”server”>
    <title>First Web Page</title>
     <style type=”text/css” >
    h1 { color:Red}
    p{ background:gray}
     </style>
 </head>
 <body>
    <form id=”form1” runat=”server”>
      <div>
        <h1>ASP Tutorial</h1>
    <p>The tutorial is a quick review for the topics required for web development.</p>
      </div>
    </form>
 </body>
</html>
```

The output after executing the code is shown below:



ASP Tutorial

The tutorial is a quick review for the topics required for web development. Study is easier using these tutorials.

External Style Sheet: The styles created in an external style sheets can be reused by many applications. There is an external style page created and it can be linked to the web page. The external style sheet is a text file created with **.css** extension. The syntax to link the external style sheet to the web page is as shown below:

?

```
<head>
 <title>Web Creation</title>
 <link href=”name1.css” rel=”stylesheet” type=”text/css” >
</head>
```

The code added in the CSS file is as shown below:

?

```
 Body
 {
   Background-color:Aqua;
 }
 h1
 {
   border :10pt, 5pt, 4pt, 4pt;
   background-color:Yellow;
 }
 p
 {
    background-color:Green;
    border-style:dotted;
```

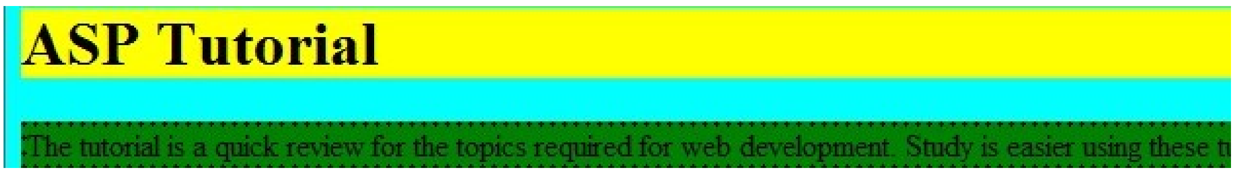
```
}
```

The code to link the CSS file with the web page is as shown below:

?

```
<html xmlns=”http://www.w3.org/1999/xhtml”>
 <head runat=”server”>
   <title>First Web Page</title>
   <link rel=”Stylesheet” type=”text/css” href=”StyleSheet1.css” />
 </head>
 <body>
    <form id=”form1” runat=”server” >
     <div>
      <h1>ASP Tutorial</h1>
       <p> The tutorial is a quick review for the topics required for web development. Study is easi
     </div>
 </body>
</html>
```

The output of the file when executed on the server is as shown below:



ASP Tutorial

The tutorial is a quick review for the topics required for web development. Study is easier using these t

Inline Style Sheet: The Inline style sheets are used for adding style to the particular element in the web page. The inline style element is embedded directly in the html elements. The selector element is not required in Inline styles. An example of adding an Inline style in an html element is as mentioned below:

It is not efficient to use the Inline style for large codes as the style is limited for an individual element.

<p style=”font-family : arial; color:red; font-size:16px;”></p>