* Master page:-
* A site benefits from a consistent look and feel. We rarely find sites on the internet that having generic site layout, which generally includes the following:

1. A common header and menu system for the entire site.
2. A bar on the left side of the page offering some page navigation options.
3. A footer providing copyright information and a secondary menu for contacting the webmaster.

* These elements will be present on every page and they not only provide essential features, but the consistent layout of these elements signal to users that they are still in the same site.
* Master pages allow you to create a consistent look and behavior for all the pages (or group of pages) in your web application.

A master page provides a template for other pages, with shared layout and functionality.

* A master page defines the layout to be used by all pages based on the Master. It’s the overall parent that controls your layout, specifying how big you header will be on every page, where your navigation features will be placed etc.
* The master page contains some of the content available to each page on the site, so standard copyright footer text can be defined here, along with positioning the main site logo at the top of the page.
* The content pages contain the content you want to display.
* The content page is based on master, in content page we can add the content for each page on a site that is different from page to page.
* When the content page is requested, its content is combined with a copy of the master page.
* We can create a master page by right click on the root of the site, selecting add item and select the type as a Master page. By default, the name for a new Master page is ***MasterPage.master*** and located in the root of the site.
* The master page has three parts:-

1. First are some basic page tags and designation that are required by any rendered page. This content is entered only once in the Master page.
2. The second part of the Master page is a place for scripts that will be run on all pages (Such as Page\_Load).
3. Third, the master page includes some HTML for layout, and the start/end tags.

* Navigation Control:-

There are three navigation control.

1. SiteMapPath
2. TreeView
3. Menu

* The three out-of-the-box controls available for navigating a site.

1. SiteMapPath control:-

* SitemapPath control display navigation path.
* This Navigation path is often called as breadcrumb.
* Remind you where you are in the site hierarchy.
* As you click through pages on a site, you many pass through to different sections and subsections of the site.
* A breadcrumbs helps you to go back to a specific point in your navigation path without having to depend on the back button on the browser.
* You may have encountered situations where you can’t hit the back button of the browser at that time SiteMapPath control is useful.

Important Property of siteMapPath – Pathseprator <asp: SiteMapPath ID="SiteMapPath1" runat="server"></asp: SiteMapPath>

TreeView

* The TreeView Web control is useful to display hierarchical data in a tree structure.
* TreeView control three ways bound

1. XML file
2. web.sitemap file
3. Database table

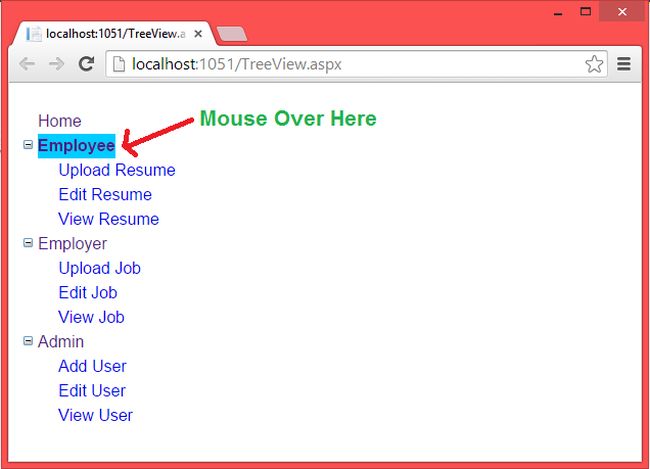
* TreeView control available in Navigation part on VisualStudio.
* Put TreeView control on our website just like drag and drop

Syntax:

<asp:TreeView ID="TreeView1" runat="server> </asp:TreeView>

Some property on TreeView

Pathseparator – it is used to separate the path.



Menu

It was the very difficult task to maintain the menu of a large website and time consuming.

It is used to display the Menus. You can use it as easy as other Navigation controls.

Menu can be stored in a file to make it easier to maintain.

Menu control available in Navigation part on VisualStudio.

Put Menu control on our website just like drag and drop

Syntax:

<asp:Menu ID="Menu1" runat="server"> </asp:Menu>

Some Property of menu

Orientaion Horizontali,vertically

Items –add to menu items

And items has two property text and value

Validation Control

**RequiredFieldValidator Control**

RequirefieldVaidator control Checks that the validated control must contains a value. It cannot be empty.

First drag the RequiredFielValidator control on asp.net web form

|  |  |
| --- | --- |
| **Members** | **Description** |
| ControlToValidate | Indicates the input control to validate. |
|  |  |
| EnableClientScript | Indicates whether client side validation will take. |
| Enabled | Enables or disables the validator. |
| ErrorMessage | Display the error message |
| Text | Error text to be shown if validation fails. |
| IsValid | Indicates whether the value of the control is valid. |
| SetFocusOnError | It indicates whether in case of an invalid control, the focus should switch to the related input control.  True and False |
| ValidationGroup | The logical group of multiple validators, where this control belongs. |
| Validate() | This method revalidates the control and updates the IsValid property. |

Syntax:

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"

ErrorMessage="RequiredFieldValidator"></asp:RequiredFieldValidator>

## RangeValidator Control

RangeValidator control is used to checks the input control’s value is within a specified range, Means input values must be between two values,  It has minimum and maximum value.

Generally we use RangeValidator control for Mobile no and Pincode Number.

The most important properties of the RangeValidator control are **MaximumValue**, **MinimumValue**, and **Type**.

We must specify the Range from Minimum values to Maximum Values and T

It has three specific properties:

|  |  |
| --- | --- |
| **Properties** | **Description** |
| Type | It defines the type of the data. The available values are: Currency, Date, Double, Integer, and String. |
| MinimumValue | It specifies the minimum value of the range. |
| MaximumValue | It specifies the maximum value of the range. |

|  |  |
| --- | --- |
| **Members** | **Description** |
| ControlToValidate | Indicates the input control to validate. |
| Display | Indicates how the error message is shown. |
| EnableClientScript | Indicates whether client side validation will take. |
| Enabled | Enables or disables the validator. |
| ErrorMessage | Print Error msg |
| Text | Error text to be shown if validation fails. |
| IsValid | Indicates whether the value of the control is valid. |
| SetFocusOnError | It indicates whether in case of an invalid control, the focus should switch to the related input control.  True and False |
| ValidationGroup | The logical group of multiple validators, where this control belongs. |
| Validate() | This method revalidates the control and updates the IsValid property. |

## RegularExpressionValidator

The RegularExpressionValidator allows validating the input text by matching against a pattern of a regular expression.

RegularExpression calidator control mostly use for validate **Internet E-mail address** and **Internet URL.**

The Internet Email address must contain @ after some text and dot(.) before some text, then the address is right other wise the email address is invalid format.

|  |  |
| --- | --- |
| **Members** | **Description** |
| ControlToValidate | Indicates the input control to validate. |
| Display | Indicates how the error message is shown. |
| EnableClientScript | Indicates whether client side validation will take. |
| Enabled | Enables or disables the validator. |
| ErrorMessage | Indicates error string. |
| Text | Error text to be shown if validation fails. |
| IsValid | Indicates whether the value of the control is valid. |
| SetFocusOnError | It indicates whether in case of an invalid control, the focus should switch to the related input control. |
| ValidationGroup | The logical group of multiple validators, where this control belongs. |
| Validate() | This method revalidates the control and updates the IsValid property. |
|  |  |

## CompareValidator Control

* CompareValidator control is used to compare/match the value of one input control to the value of another input control.
* Mostly this control is used to Entering the **Password detail** in Registration  form.
* For this control we must specify the input Datatype for comparison.
* **Input Type may be:**

– String

– Date

– Currency

– Integer

– Double

* Here we have to set two property for validation.
* **ControlToValidate**– The main input control means**Original first password.**
* **ControlToComapre**– The second input control which is compare with main control, means **confim -password control.**

It has the following specific properties:

|  |  |
| --- | --- |
| **Properties** | **Description** |
| Type | It specifies the data type. |
| ControlToCompare | It specifies the value of the input control to compare with. |
| ValueToCompare | It specifies the constant value to compare with. |
| Operator | It specifies the comparison operator, the available values are: Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, and DataTypeCheck. |
| Members | Description |
| ControlToValidate | Indicates the input control to validate. |
| Display | Indicates how the error message is shown. |
| EnableClientScript | Indicates whether client side validation will take. |
| Enabled | Enables or disables the validator. |
| ErrorMessage | Indicates error string. |
| Text | Error text to be shown if validation fails. |
| IsValid | Indicates whether the value of the control is valid. |
| SetFocusOnError | It indicates whether in case of an invalid control, the focus should switch to the related input control. |
| ValidationGroup | The logical group of multiple validators, where this control belongs. |
| Validate() | This method revalidates the control and updates the IsValid property. |

CustomValidator

The CustomValidator control allows writing application specific custom validation routines for both the client side and the server side validation.

The client side validation is accomplished through the ClientValidationFunction property. The client side validation routine should be written in a scripting language, such as JavaScript or VBScript, which the browser can understand.

The server side validation routine must be called from the control's ServerValidate event handler. The server side validation routine should be written in any .Net language, like C# or VB.Net.

The basic syntax for the control is as given:

<asp:CustomValidator ID="CustomValidator1" runat="server"

ClientValidationFunction=.cvf\_func. ErrorMessage="CustomValidator">

</asp:CustomValidator>

## ValidationSummary

The ValidationSummary control does not perform any validation but shows a summary of all errors in the page. The summary displays the values of the ErrorMessage property of all validation controls that failed validation.

The following two mutually inclusive properties list out the error message:

* **ShowSummary** : shows the error messages in specified format.
* **ShowMessageBox** : shows the error messages in a separate window.

The syntax for the control is as given:

<asp:ValidationSummary ID="ValidationSummary1" runat="server"

DisplayMode = "BulletList" ShowSummary = "true" HeaderText="Errors:" />

The ValidationSummary control is used to display a summary of all the validation control error messages  in a single control on web forms in asp.net. We can also cay the valiudationsummary control is the summary of all the validation control error messages display on web forms.

The Validationsummary control display all the validation control error message at single place in asp.net.

* Login Controls :-

1. Login

* The Login control, which provides text boxes, buttons, and built-in validation to enable you to add login functionality to a page with a single drag and drop operation.
* This control is quite powerful. This control is new addition to edition of ASP.NET. Previously, we have had to add text boxes and buttons, and write lines of C# code to handle login process.
* Whenever user hits the Log In button, the control automatically validates the user name and password with the membership API function Membership.ValidateUser() and then redirected to login page if the validation was successful.

1. LoginView control:

* The LoginView control, which provides a way of changing the appearance of the page dependent on wheather a user is logged in or not.

1. LoginStatus control:

* The LoginStatus control, which gives a simple bit of feedback to users so that they know whether they have remembered to login to the site.
* Methods of LoginStatus control:

1. LoggedOut:

It raises the event after user is logged out.

1. LoggingOut:

It raises the event before user is logged out.

1. LoginName control:

* The LoginName control is a quick and easy way to display a user’s currently logged-in identity on a site.
* The control is not displayed if it does not contain any logged in user.
* LoginName class is used for the control.
* The control does not cotain any method, property or events associated with it.

1. PasswordRecovery Control:

* It is used to recover or reset the password for the user. The new password is sent through an email as a message.
* The membership is used for creating and resetting the password.
* This control has 3 views:-

1. UserName
2. Question
3. Success

* PasswordRecovery Control has 4 methods

1. SendingMail():-

It raises the SendingMail event when the user is verified and the password is sent to the user.

1. UserLookupError():

It raised when the username is invalid, if user is not found in the database at that time this even is raised.

1. SendMailError():-

Raised when there is an error sending mail.

1. VerifyingUser():-

It raises the event once the username is submitted, and the membership provider verification is pending.

1. CreateUserWizard control:-

* The control uses the membership service for creation of a new user.
* The control can be customized through templates and properties.
* The control has two steps:

1. Sign up for your new account
2. Complete

* Some methods of CreqteUserWizard control are as below:

1. CreatedUser() :

It is initiated after the membership service provider has created a new user account.

1. CreatingUser():

It is initiated before the membership service provider is called for creating user account.

1. SendingMail():

It is initiated before sending the conformation email on the successful creation of the account.

1. SendMailError() :

It is initiated when the SMTP error occurs during the mail sent to the user.