



AMIT ACADEMY
for Computer Education



Subject Title: WEB DEVELOPMENT USING ASP.NET

Subject Code: BCA 501

Unit 3: Various Controls

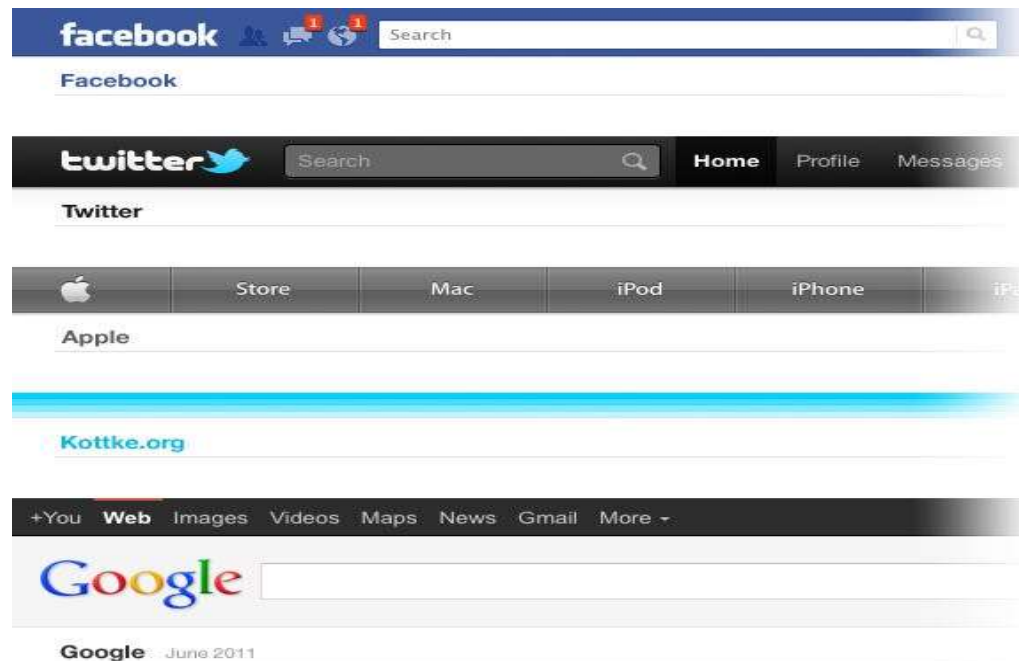
Prepared by Ankit Rami(AR) Any Query

Contact – +91 8460467193

Email – ankitramiblog@gmail.com

Navigation Controls in ASP.NET

- ✓ Navigation control in ASP.NET manages the data passing between ASPX pages.
- ✓ Web applications are having multiple pages interconnected with each other. So proper navigation system must be there which can help the end user to successfully work through an application. There are standard methods are available in ASP.NET 1.x that offers well defined navigation system for the web application.
- ✓ Only the method which is present for building navigation within web application is to fill the pages with hyperlinks.
- ✓ When the site grows and new pages are added and at that time it is very difficult for the developer to manage all links in the application.
- ✓ ASP.NET 2.0 and above eliminate problems of links with a built in site navigation features. It provides consistent way for the user to navigate the website. It enables defining all the links at central location file as xml file and display those links in list or navigation menus in each required page using navigation controls.



📢 Different Navigation Controls in ASP.NET

The screenshot displays the Visual Studio IDE with the 'Navigation' group selected in the toolbox. The design view shows three navigation controls: 'Menu Control', 'Site Map Path', and 'TreeView'. Red arrows point from the following text labels to these controls:

- Menu Control** (points to the 'Menu Control' control)
- Site Map Path** (points to the 'Site Map Path' control)
- Tree View** (points to the 'TreeView' control)

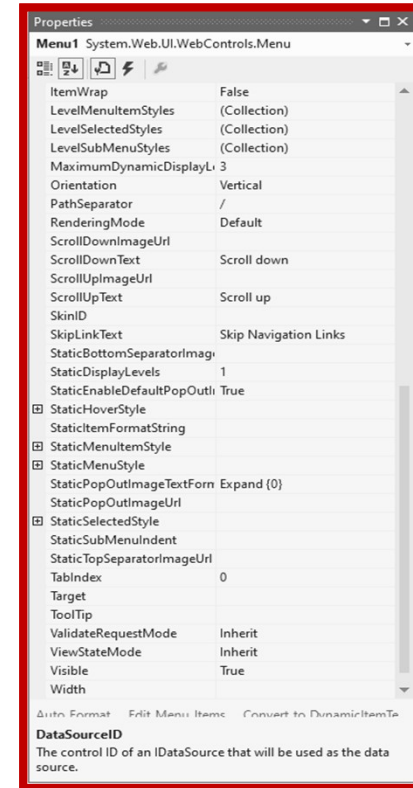
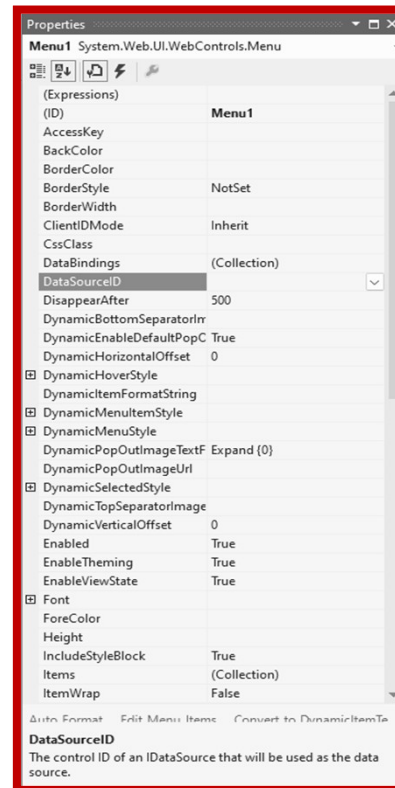
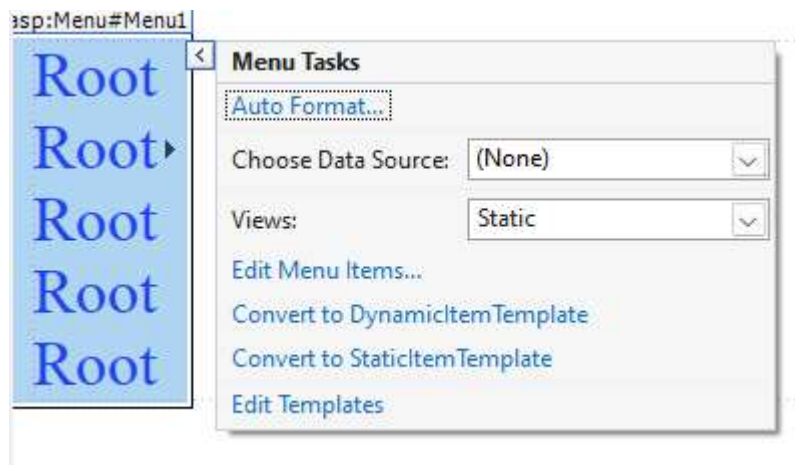
The 'TreeView' control is expanded, showing a hierarchical structure:

- Root
 - Parent 1
 - Leaf 1
 - Leaf 2
 - Parent 2
 - Leaf 1
 - Leaf 2

📢 ASP.NET Navigation Controls : Menu Control

- ✓ Menu is a navigation control in ASP.NET which is used to display menu in web page.
- ✓ This control is used in combination with SiteMapDataSource control for navigating the web Site.
- ✓ It displays two types of menu (1) static menu and (2) dynamic menu.
- ✓ Static menu is always displayed in menu Control, by default only menu items at the root levels are displayed.
- ✓ Dynamic menu is displayed only when the use moves the mouse pointer over the parent menu that contains a dynamic sub menu.

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



ASP.NET Navigation Controls : Menu Control

The screenshot shows the Visual Studio IDE with the following components:

- Toolbox:** Standard and Data controls are visible on the left.
- Design View:** A menu control is placed on the page with the text "NPCCSM KADI KSV UNIVERSITY BPCCS GANDHINAGAR".
- Menu Tasks:** A panel on the right of the design view provides options like "Auto Format...", "Choose Data Source:", "Views:", "Edit Menu Items...", "Convert to DynamicItemTemplate", "Convert to StaticItemTemplate", and "Edit Templates".
- Menu Item Editor:** A dialog box for editing the selected menu item.

Property	Value
Enabled	True
ImageUrl	
NavigateUrl	https://npccsm.in/
PopOutImageUrl	
Selectable	True
Selected	False
SeparatorImageUrl	
Target	
Text	NPCCSM KADI
ToolTip	
Value	NPCCSM KADI
- Properties Window:** Shows the properties for the selected menu item, including (Expressions), (ID), AccessKey, BackColor, BorderColor, BorderStyle, BorderWidth, ClientIDMode, CssClass, DataBindings, DataSourceID, DisappearAfter, DynamicBottomSeparatorIn, DynamicEnableDefaultPopC, DynamicHorizontalOffset, DynamicHoverStyle, DynamicItemFormatString, DynamicMenuItemStyle, DynamicMenuStyle, DynamicPopOutImageTextF, DynamicPopOutImageUrl, and DynamicSelectedStyle.
- Output Window:** Shows the debug output from the application, including assembly loading and program exit messages.

📢 ASP.NET Navigation Controls : Menu Control

The screenshot displays the Visual Studio IDE with the source code of an ASP.NET page, ASP_MENU.aspx. The code defines a menu control with three items: NPCCSM KADI, KSV UNIVERSITY, and BPCCS GANDHINAGAR. The rendered output in the browser shows these items as a horizontal menu with a blue background and white text. A red arrow points from the 'OUTPUT' label to the browser window.

```

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

<!DOCTYPE html>

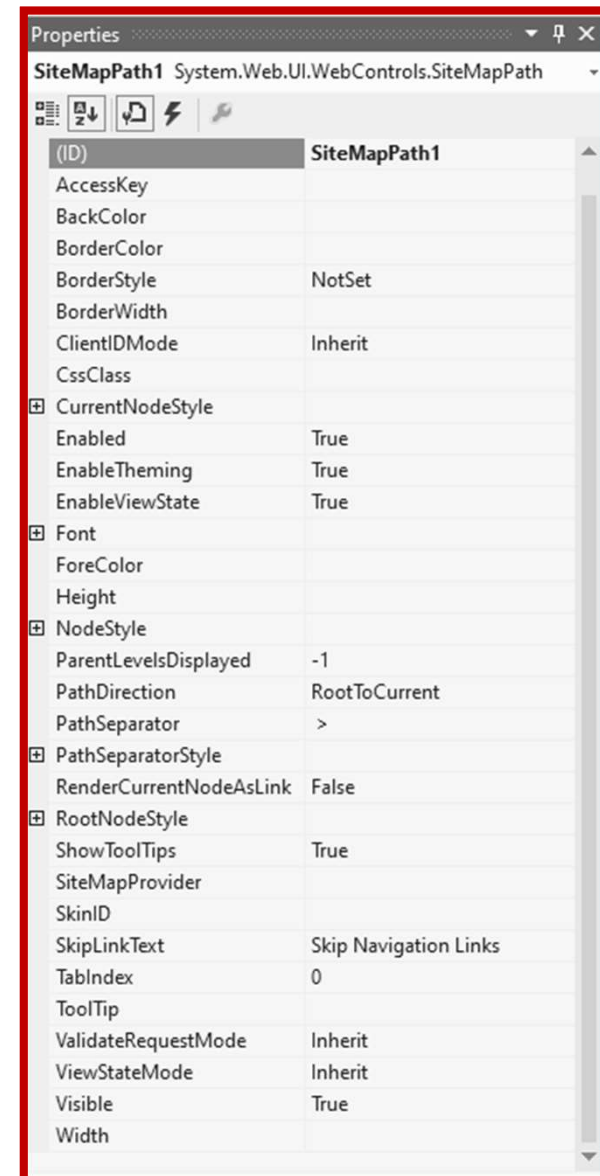
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<asp:Menu ID="Menu1" runat="server" Font-Size="X-Large" Orientation="Horizontal">
<Items>
<asp:MenuItem NavigateUrl="https://npccsm.in/" Text="NPCCSM KADI" Value="NPCCSM KADI"></asp:MenuItem>
<asp:MenuItem NavigateUrl="https://ksv.ac.in/" Text="KSV UNIVERSITY" Value="KSV University"></asp:MenuItem>
<asp:MenuItem NavigateUrl="https://bpccs.org/" Text="BPCCS GANDHINAGAR" Value="BPCCS GANDHINAGAR"></asp:MenuItem>
</Items>
<StaticHoverStyle BackColor="#FFFF99" BorderStyle="Dotted" BorderWidth="2px" />
<StaticMenuStyle BackColor="#CCCCFF" BorderStyle="Solid" HorizontalPadding="10px" VerticalPadding="10px" />
<StaticSelectedStyle ItemSpacing="2px" />
</asp:Menu>
</form>
</body>
</html>
    
```

Browser Output: NPCCSM KADI KSV UNIVERSITY BPCCS GANDHINAGAR

Browser URL: localhost:49981/ASP_MENU.aspx

ASP.NET Navigation Controls : SiteMapPath Control

- ✓ Site maps are XML files which are mainly used to describe the logical structure of the web application.
- ✓ It defines the layout of all pages in web application and how they relate to each other. Whenever you want you can add or remove pages to your site map there by managing navigation of website efficiently.
- ✓ Site map files are defined with .sitemap extension. <sitemap> element is the root node of the sitemap file.
- ✓ It has three attributes:
 - Title:** It provides textual description of the link.
 - URL:** It provides the location of the valid physical file.
 - Description:** It is used for tooltip of the link.
- ✓ SiteMapPath control displays the navigation path of the current page. The path acts as click able links to previous page. The sitemap path control uses the web.
- ✓ This control creates the navigation mechanism which is linear path defining where the user is currently located in navigation arrangement. It helps end user to know his location in relation to the rest of the site.
- ✓ Ex- `<asp:SiteMapPath ID="SiteMapPath1" runat="server"></asp:SiteMapPath>`

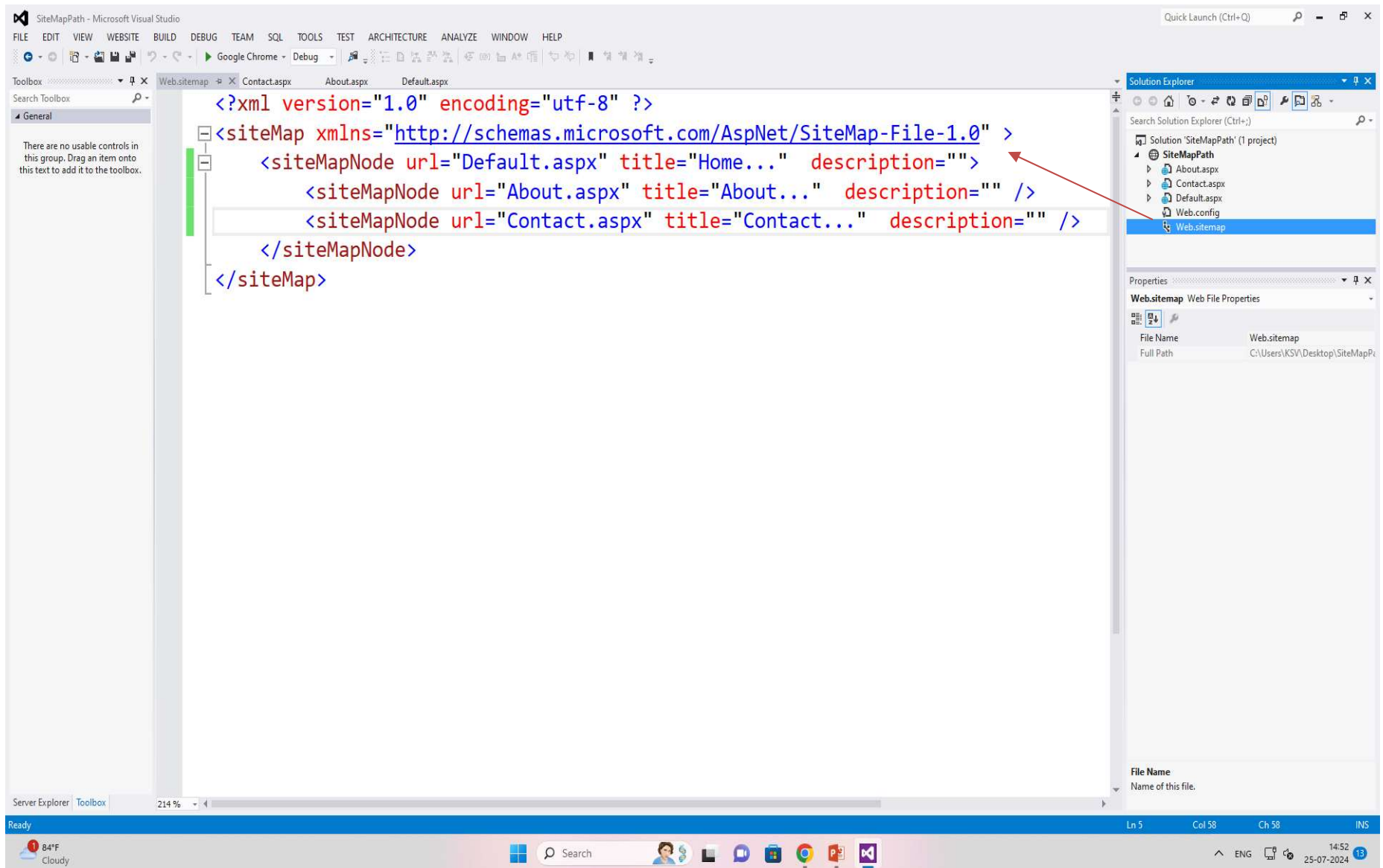


📢 ASP.NET Navigation Controls : SiteMapPath Control

The screenshot displays the Microsoft Visual Studio environment with the following components:

- Source View:** Shows the HTML code for the page. The `<asp:SiteMapPath ID="SiteMapPath1" runat="server"></asp:SiteMapPath>` control is placed within a `<div>` inside a `<form id="form1" runat="server">`.
- Design View:** Shows the rendered output of the page, displaying "Home Page" in a large font and "Home..." in a smaller font below it.
- Properties Window:** Lists various accessibility properties for the `<DIV>` element, such as `accesskey`, `aria-activedescendant`, `aria-atomic`, etc.

📢 ASP.NET Navigation Controls : SiteMapPath Control



The screenshot displays the Microsoft Visual Studio IDE with the following components:

- Code Editor:** Shows the XML content of the `Web.sitemap` file:


```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="Default.aspx" title="Home..." description="" >
    <siteMapNode url="About.aspx" title="About..." description="" />
    <siteMapNode url="Contact.aspx" title="Contact..." description="" />
  </siteMapNode>
</siteMap>
```
- Solution Explorer:** Shows the project structure for 'SiteMapPath' (1 project) with files: `About.aspx`, `Contact.aspx`, `Default.aspx`, `Web.config`, and `Web.sitemap`. A red arrow points from the `Web.sitemap` file to the corresponding XML code in the editor.
- Properties Window:** Shows the 'Web.sitemap' file properties, including the file name and full path: `C:\Users\KSV\Desktop\SiteMapP...`

ASP.NET Navigation Controls : SiteMapPath Control

The screenshot displays the Visual Studio IDE with the following components:

- Code Editor:** Shows the source code for `Default.aspx`. The code includes a page directive, HTML boilerplate, and an `asp:SiteMapPath` control. The control is configured with `ID="SiteMapPath1"` and `runat="server"`.
- Solution Explorer:** Shows the project structure with files: `About.aspx`, `Contact.aspx`, `Default.aspx`, `Web.config`, and `Web.sitemap`.
- Browser Windows:** Three browser windows are open, showing the rendered output:
 - `localhost:14863/Default.aspx`: Displays "Home Page" and "Home..."
 - `localhost:14863/about.aspx`: Displays "About Us" and a breadcrumb "[Home...](#) > About..."
 - `localhost:14863/contact.aspx`: Displays "Contact Us" and a breadcrumb "[Home...](#) > Contact..."
- Properties Window:** Shows the `id` property of the `SiteMapPath` control.
- Output Window:** A red box labeled "OUTPUT" is positioned at the bottom center of the browser windows.

Validation Controls in ASP.NET

- ✓ ASP.NET enables programmers to create dynamic web pages and design robust and quality websites.
- ✓ ASP.NET validation controls validate the user input data to ensure it is not useless, unauthenticated, or contradictory.
- ✓ It provides a way to check the accuracy and validity of user input before the web application processes it.
- ✓ It is used to validate user input value.
- ✓ It is used to help in implementing presentation logic Which is a layer of the web application.
- ✓ Validation Controls are used in data formatting, data ranging, and data type for validation.
- ✓ ASP.NET provides the following validation controls:

1. RequiredFieldValidator

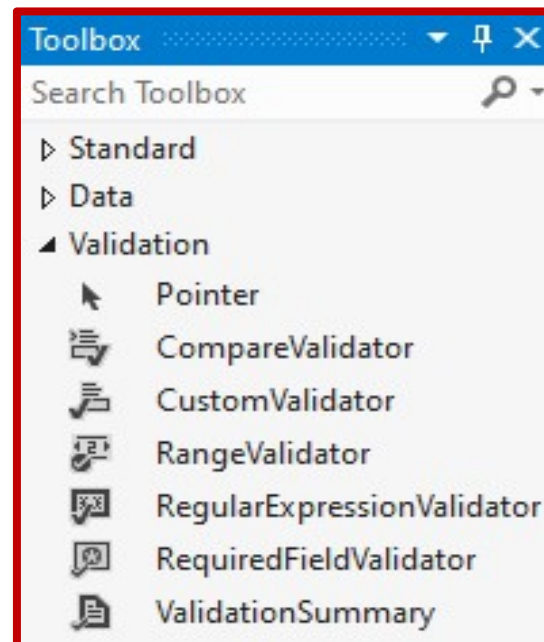
2. RangeValidator

3. CompareValidator

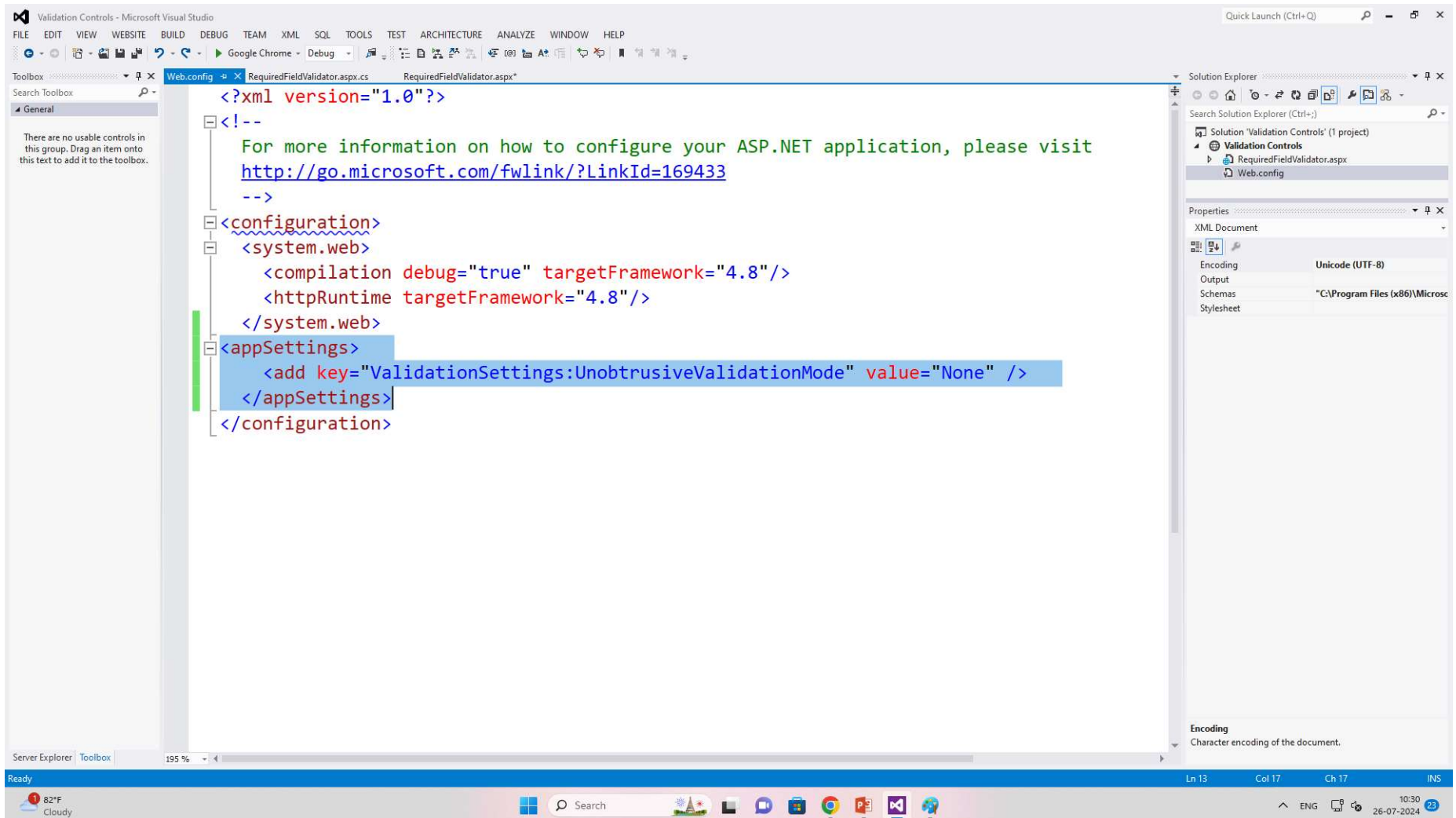
4. RegularExpressionValidator

5. CustomValidator

6. ValidationSummary



📣 Validation Controls in ASP.NET : Setting for Error Handling in Web.Config File



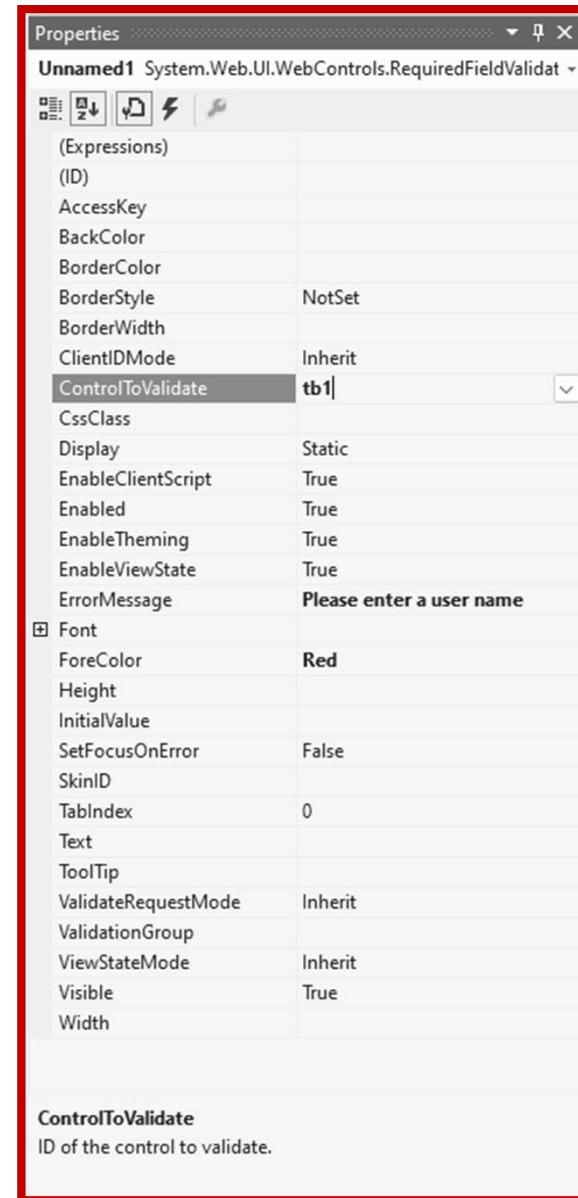
The screenshot shows the Visual Studio IDE with the Web.config file open. The configuration is as follows:

```
<?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>
    <compilation debug="true" targetFramework="4.8"/>
    <httpRuntime targetFramework="4.8"/>
  </system.web>
  <appSettings>
    <add key="ValidationSettings:UnobtrusiveValidationMode" value="None" />
  </appSettings>
</configuration>
```

The `<appSettings>` section is highlighted in blue in the image, indicating the setting for unobtrusive validation mode.

📢 Validation Controls in ASP.NET : RequiredFieldValidator

- ✓ It is also known as an elementary validation control.
- ✓ The form consists of mandatory fields to be filled.
- ✓ If the users or clients want to proceed with the form, they will have to mandatorily fill these fields, RequiredFieldValidator is used when such fields are **not left empty**.
- ✓ It checks that there must be some value added within the control.
- ✓ Important Properties of RequiredFieldValidators such as Initial value, ControlToValidate, and Text.
- ✓ Initial value is shown by default to guide the users or client on how the value must be added, hence it is especially used for a drop-down list.
- ✓ The second property is the ControlToValidate property, It is used to set the field of the text box for validation purposes.
- ✓ And the last property is "Text", It is used to set the text value for the validation.
- ✓ Ex- `<asp:RequiredFieldValidator ID:RequiredFieldValidatorID="user" runat="server" ControlToValidate="tb1" ErrorMessage="Please enter a user name" ForeColor="Red">`
`</asp:RequiredFieldValidator>`



📢 Validation Controls in ASP.NET : RequiredFieldValidator

The screenshot displays the Visual Studio IDE with the following components:

- Code File (RequiredFieldValidator.aspx.cs):**

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="RequiredFieldValidator.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
<script type="text/javascript" src="js/JScript.js" ></script>
</head>
<body>
<form id="form1" runat="server">
<div>
Enter Name <asp:TextBox ID="tb1" runat="server"></asp:TextBox>
<asp:RequiredFieldValidator RequiredFieldValidatorID="user" runat="server" ControlToValidate="tb1"
ErrorMessage="Please enter a user name" ForeColor="Red"></asp:RequiredFieldValidator>
<br/>
<asp:Button ID="Button1" runat="server" Text="Check"/>
</div>
</form>
</body>
</html>

```
- Design View:** Shows a rendered form with a text box labeled "Enter Name" and a "Check" button. A red error message "Please enter a user name" is displayed next to the text box.
- Properties Window:** Shows the properties of the document, including Language (C#), MasterPageFile, Style, StyleSheetTheme, Theme, Title, Trace, TraceMode, and UICulture.

Validation Controls in ASP.NET : RequiredFieldValidator

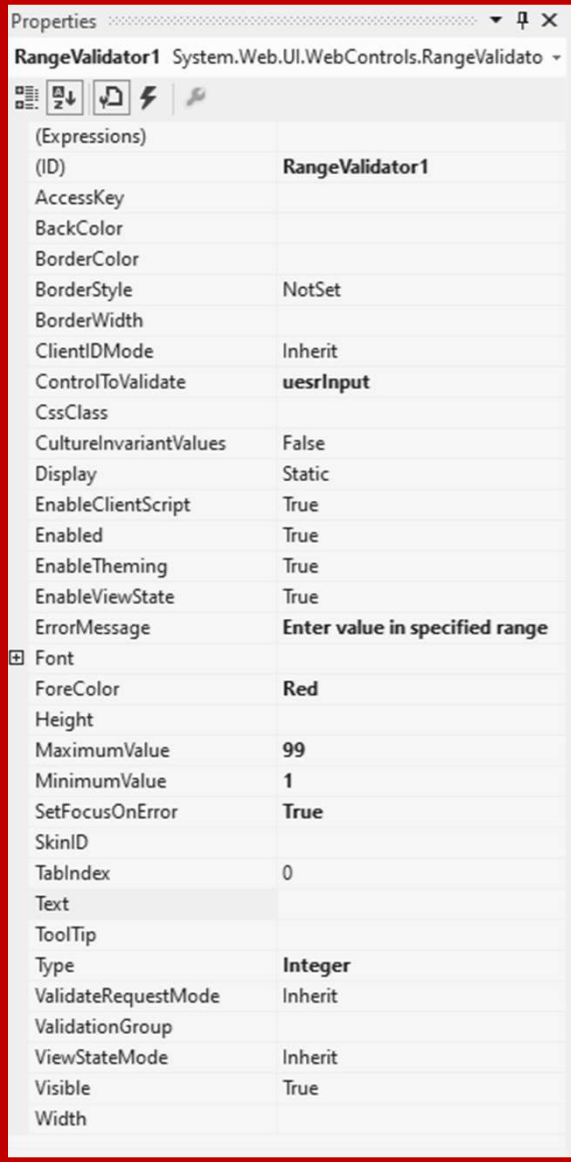
The screenshot displays the Visual Studio IDE with a C# code file named `RequiredFieldValidator.aspx.cs`. The code defines a partial class `_Default` that inherits from `System.Web.UI.Page`. It includes several using statements for `System`, `System.Collections.Generic`, `System.Linq`, `System.Web`, `System.Web.UI`, and `System.Web.UI.WebControls`. The `Page_Load` method is empty. The `Button1_Click` method contains a validation check: `if (Page.IsValid)`. If the page is valid, it writes `"Data Check Successfully."` to the response. A red arrow points from the `Page.IsValid` property access in the code to the browser output below.

The browser output shows three states of the application:

- Top:** The page is rendered with the text "Enter Name" and an empty text input field, followed by a "Check" button.
- Middle:** After clicking the "Check" button, the page displays "Please enter a user name" in red text next to the empty input field, indicating a validation error.
- Bottom:** After clicking the "Check" button with the name "Ankit Rami" entered in the input field, the page displays "Data Check Successfully." in red text, indicating a successful validation.

📢 Validation Controls in ASP.NET : RangeValidator

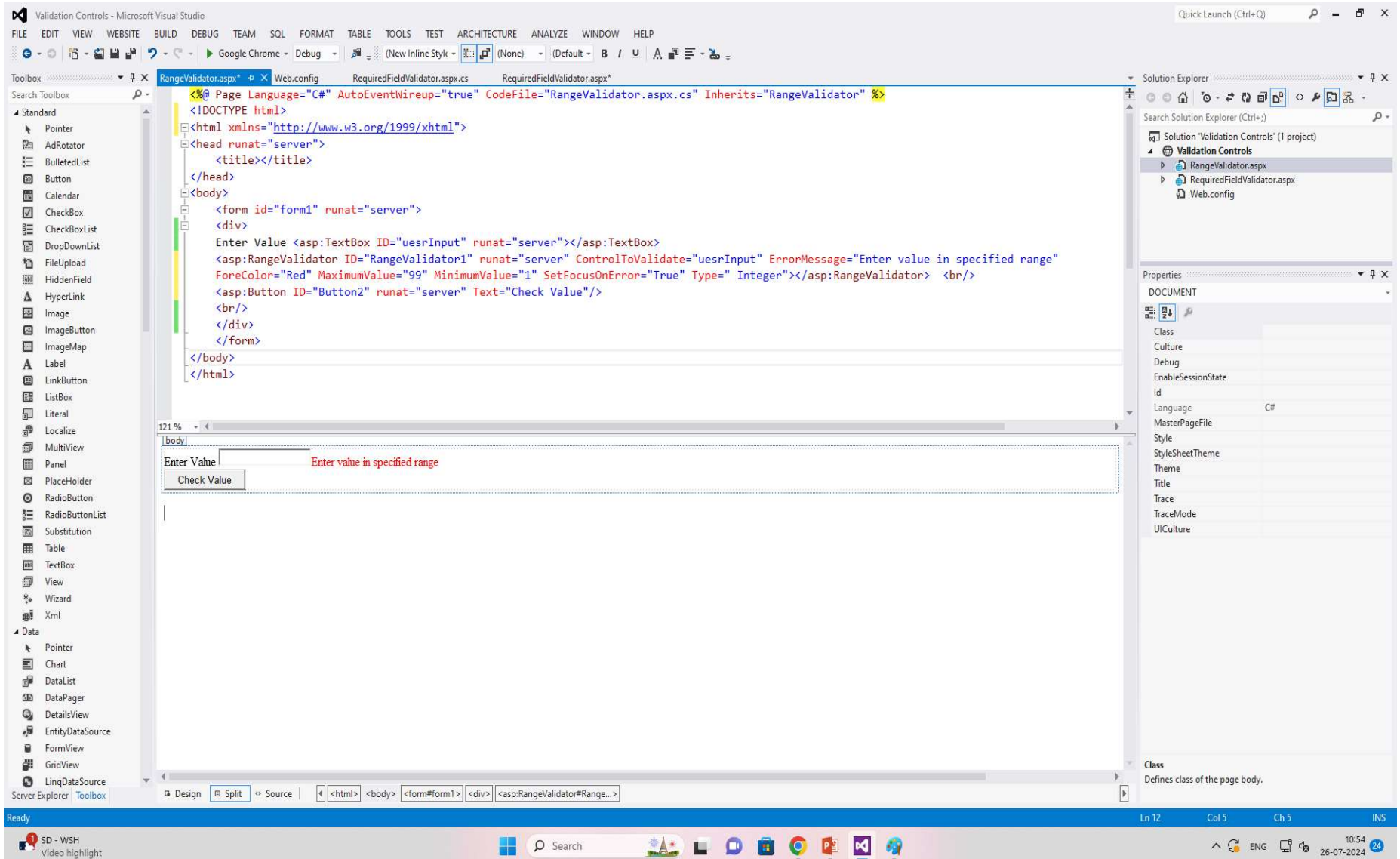
- ✓ It is used to check whether the value of the input control is inside a particular range or not.
- ✓ RangeValidator control is used when it comes to getting inputs such as Age, Date of Birth, or mobile numbers from the user.
- ✓ Important properties of RangeValidator control such as Minimum Value, Maximum Value, ControlToValidate, and Type.
- ✓ Minimum Value is used when the programmer holds the valid range's minimum value.
- ✓ The maximum value is used when the programmer holds a valid range's maximum value.
- ✓ ControlToValidate allows the programmer to set the specific control that needs to be validated.
- ✓ Type is used to set after the above properties if needed.
- ✓ This validator can compare the following data types : **String, Integer, Double, Date, Currency**
- ✓ Ex- `<asp:RangeValidator ID="RangeValidator1" runat="server" ControlToValidate="uesrInput" ErrorMessage="Enter value in specified range" ForeColor="Red" MaximumValue="99" MinimumValue="1" SetFocusOnError="True" Type="Integer">`
`</asp:RangeValidator>`



Properties: RangeValidator1 System.Web.UI.WebControls.RangeValidato

(Expressions)	
(ID)	RangeValidator1
AccessKey	
BackColor	
BorderColor	
BorderStyle	NotSet
BorderWidth	
ClientIDMode	Inherit
ControlToValidate	uesrInput
CssClass	
CultureInvariantValues	False
Display	Static
EnableClientScript	True
Enabled	True
EnableTheming	True
EnableViewState	True
ErrorMessage	Enter value in specified range
Font	
ForeColor	Red
Height	
MaximumValue	99
MinimumValue	1
SetFocusOnError	True
SkinID	
TabIndex	0
Text	
ToolTip	
Type	Integer
ValidateRequestMode	Inherit
ValidationGroup	
ViewStateMode	Inherit
Visible	True
Width	

Validation Controls in ASP.NET : RangeValidator



The screenshot displays the Microsoft Visual Studio IDE with a project named "Validation Controls". The main window shows the source code for "RangeValidator.aspx.cs", which inherits from "RangeValidator". The code defines a page with a form containing a text box and a range validator.

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="RangeValidator.aspx.cs" Inherits="RangeValidator" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
Enter Value <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="99" MinimumValue="1" SetFocusOnError="True" Type=" Integer"></asp:RangeValidator> <br/>
<asp:Button ID="Button2" runat="server" Text="Check Value"/>
</div>
</form>
</body>
</html>
    
```

The Design view below the code shows a visual representation of the form. It features a text box labeled "Enter Value" and a "Check Value" button. A red error message, "Enter value in specified range", is displayed above the text box, indicating that the current input is outside the specified range.

The Solution Explorer on the right shows the project structure, including "RangeValidator.aspx" and "RequiredFieldValidator.aspx". The Properties window on the right shows the document properties, such as Language (C#) and Class (Defines class of the page body).

Validation Controls in ASP.NET : RangeValidator

The screenshot displays the Visual Studio IDE with the RangeValidator.aspx.cs file open. The code defines a partial class `RangeValidator` that inherits from `System.Web.UI.Page`. It includes the following code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

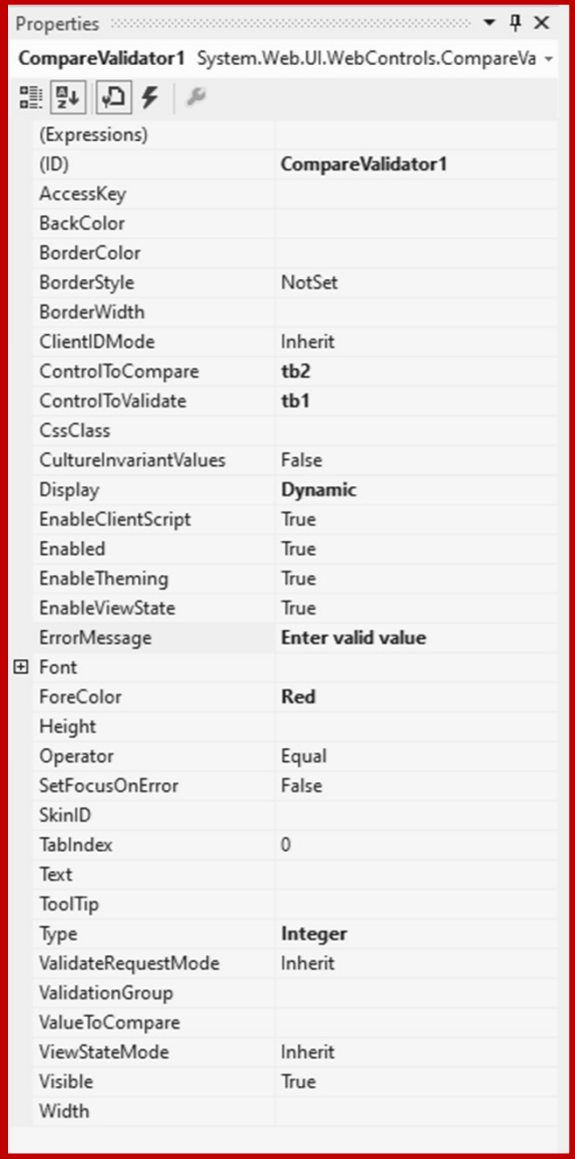
public partial class RangeValidator : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        if (Page.IsValid)
        {
            Response.Write("Data Check Successfully.");
        }
    }
}
```

A red box labeled "OUTPUT" is positioned above three browser window screenshots. The first screenshot shows the initial state with an empty text box and a "Save" button. The second screenshot shows the text box containing the value "199" and a red message "Enter value in specified range" next to it. The third screenshot shows the text box containing the value "44".

Validation Controls in ASP.NET : CompareValidator

- ✓ It compares the value of one control with either a fixed value or a value in another control.
- ✓ This validator evaluates the value of an input control against another input control on the basis of specified operator.
- ✓ There are three properties included in this validator such as ControlToCompare, ValueToCompare, and Operator.
- ✓ ControlToCompare holds the ControlToValidate ID of another form.
- ✓ Then, The value of both the form fields is then compared.
- ✓ ValueToCompare is a fixed value with which the comparison has to be made.
- ✓ Operator is the type of comparison and its attributes are: **Data Type Check, Equal, Not Equal, Greater Than, Greater Than Equal, Less Than, Less Than Equal**
- ✓ Ex- `<asp:CompareValidator ID="CompareValidator1" runat="server" ControlToCompare="tb2" ControlToValidate="tb1" Display="Dynamic" ErrorMessage="Enter valid value" ForeColor="Red" Type="Integer"> </asp:CompareValidator>`



Properties window for **CompareValidator1** (System.Web.UI.WebControls.CompareValidator).

Property	Value
(Expressions)	
(ID)	CompareValidator1
AccessKey	
BackColor	
BorderColor	
BorderStyle	NotSet
BorderWidth	
ClientIDMode	Inherit
ControlToCompare	tb2
ControlToValidate	tb1
CssClass	
CultureInvariantValues	False
Display	Dynamic
EnableClientScript	True
Enabled	True
EnableTheming	True
EnableViewState	True
ErrorMessage	Enter valid value
Font	
ForeColor	Red
Height	
Operator	Equal
SetFocusOnError	False
SkinID	
TabIndex	0
Text	
ToolTip	
Type	Integer
ValidateRequestMode	Inherit
ValidationGroup	
ValueToCompare	
ViewStateMode	Inherit
Visible	True
Width	

Validation Controls in ASP.NET : CompareValidator

The screenshot displays the Visual Studio IDE with the following components:

- Code View:** Shows the source code for `CompareValidator.aspx`. The code includes:


```

      <!DOCTYPE html>
      <html xmlns="http://www.w3.org/1999/xhtml">
      <head runat="server">
      <title></title>
      </head>
      <body>
      <form id="form1" runat="server">
      <div>
      Value-1 <asp:TextBox ID="tb1" runat="server" required="true"></asp:TextBox>
      Value-2 <asp:TextBox ID="tb2" runat="server"></asp:TextBox>
      <br />
      <asp:Button ID="Button1" runat="server" Text="Check Value"/>
      <br />
      <asp:CompareValidator ID="CompareValidator1" runat="server" ControlToCompare="tb2"
      ControlToValidate="tb1" Display="Dynamic" ErrorMessage="Enter valid value" ForeColor="Red" Type="Integer"></asp:CompareValidator>
      </div>
      </form>
      </body>
      </html>
      
```
- Design View:** Shows a preview of the web form with two text boxes labeled 'Value-1' and 'Value-2', and a 'Check Value' button. A red error message 'Enter valid value' is visible below the text boxes.
- Toolbox:** Lists various ASP.NET controls such as Pointer, AdRotator, BulletedList, Button, Calendar, CheckBox, etc.
- Properties Window:** Shows the properties for the selected 'DIV' element, including accessibility and ARIA attributes.

Validation Controls in ASP.NET : CompareValidator

OUTPUT

The screenshot displays the Visual Studio IDE with the following code in the CompareValidator.aspx.cs file:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class CompareValidator : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

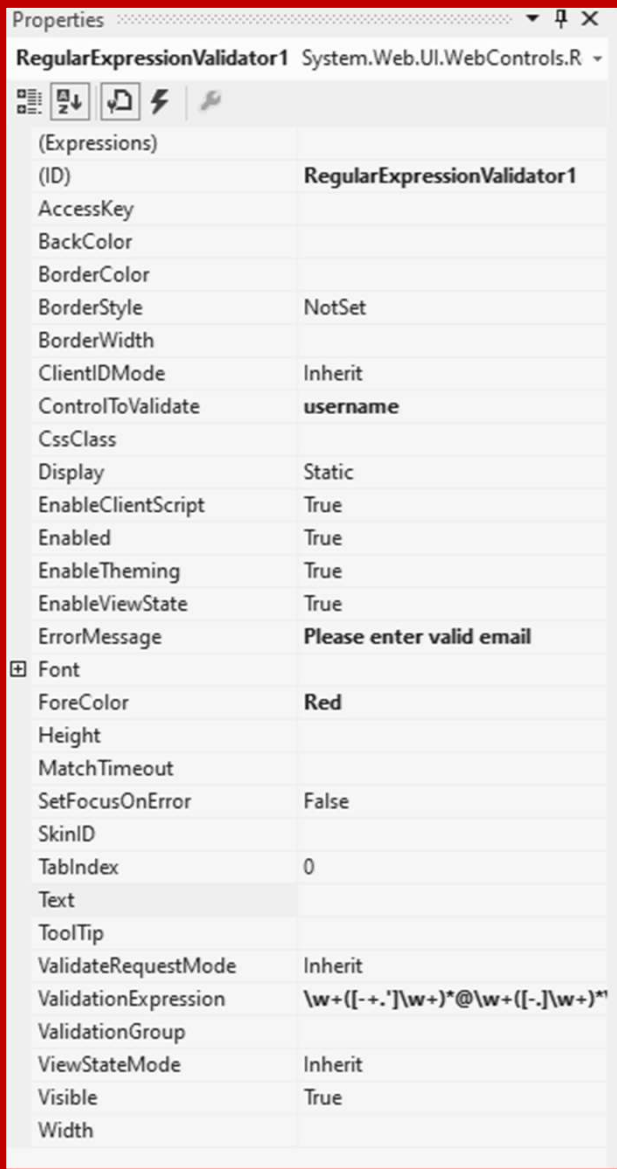
    protected void Button1_Click(object sender, EventArgs e)
    {
        if (Page.IsValid)
        {
            Response.Write("Data Check Successfully.");
        }
    }
}
    
```

Three browser screenshots illustrate the output:

- Top Screenshot:** Shows the initial state with empty input fields for "Value-1" and "Value-2" and a "save" button.
- Middle Screenshot:** Shows the state after clicking "save" with "11" entered in both fields. The text "Data Check Successfully." is displayed in red.
- Bottom Screenshot:** Shows the state after clicking "save" with "11" entered in both fields. The text "Enter valid value" is displayed in red, indicating a validation error.

Validation Controls in ASP.NET : RegularExpressionValidator

- ✓ RegularExpressionValidator also known as Regex.
- ✓ It is a control that manages various patterns which clearly define the format of the text.
- ✓ The RegularExpressionValidator allows validating the input text by matching against a pattern of a regular expression.
- ✓ The regular expression is set in the ValidationExpression property.
- ✓ If the text that has been added is in the same format then the RegularExpressionValidator will return true or else false.
- ✓ It is used to validate input fields such as email, phone, Zip code, etc...
- ✓ Ex- `<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server" ControlToValidate="username" ErrorMessage="Please enter valid email" ForeColor="Red" ValidationExpression="\w+([-+.']\w+)*@\w+([-.\w+)*\.\w+([-.\w+)*">`



The screenshot shows the Properties window for a `RegularExpressionValidator1` control. The control is of type `System.Web.UI.WebControls.RegularExpressionValidator`. The `ControlToValidate` property is set to `username`, and the `ValidationExpression` property is set to `\w+([-+.']\w+)*@\w+([-.\w+)*`. The `ErrorMessage` property is set to `Please enter valid email`, and the `ForeColor` property is set to `Red`.

Properties	
RegularExpressionValidator1 System.Web.UI.WebControls.RegularExpressionValidator	
(Expressions)	
(ID)	RegularExpressionValidator1
AccessKey	
BackColor	
BorderColor	
BorderStyle	NotSet
BorderWidth	
ClientIDMode	Inherit
ControlToValidate	username
CssClass	
Display	Static
EnableClientScript	True
Enabled	True
EnableTheming	True
EnableViewState	True
ErrorMessage	Please enter valid email
Font	
ForeColor	Red
Height	
MatchTimeout	
SetFocusOnError	False
SkinID	
TabIndex	0
Text	
ToolTip	
ValidateRequestMode	Inherit
ValidationExpression	\w+([-+.']\w+)*@\w+([-.\w+)*
ValidationGroup	
ViewStateMode	Inherit
Visible	True
Width	

Validation Controls in ASP.NET : RegularExpressionValidator

Character Escapes	Description	Quantifier	Description
\b	Matches a backspace.	*	Zero or more matches.
\t	Matches a tab.	+	One or more matches.
\r	Matches a carriage return.	?	Zero or one matches.
\v	Matches a vertical tab.	{N}	N matches.
\f	Matches a form feed.	{N,}	N or more matches.
\n	Matches a new line.	{N,M}	Between N and M matches.
\	Escape character.		

Important syntax constructs for regular expressions

Metacharacters	Description
.	Matches any character except \n.
[abcd]	Matches any character in the set.
[^abcd]	Excludes any character in the set.
[2-7a-mA-M]	Matches any character specified in the range.
\w	Matches any alphanumeric character and underscore.
\W	Matches any non-word character.
\s	Matches whitespace characters like, space, tab, new line etc.
\S	Matches any non-whitespace character.
\d	Matches any decimal character.
\D	Matches any non-decimal character.

Validation Controls in ASP.NET : RegularExpressionValidator

- ✓ Different validationexpression pattern for RegularExpressionValidator
- ✓ **Email Patten** - `[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,4}$`
 Ex- arinfoy@gmail.com
- ✓ **Mobile No** - `[7-9]{1}[0-9]{9}`
 Ex- **8460467193**
- ✓ **Full Name** - `^[A-Z][a-zA-Z]{3,}(: [A-Z][a-zA-Z]*){0,2}$`
 Ex- **Rami Ankit Rajendrabhai**
- ✓ **Username** - `^[a-zA-Z0-9]+([\._]?[a-zA-Z0-9]+)*$`
 Ex - **ankitrami_ar**
- ✓ **Password** - `/^[a-zA-Z0-9!@#\$%\^&* _+=-]{8,12}$/g`
 Ex- **Abc@1234**
 At least 1 Uppercase
 At least 1 Lowercase
 At least 1 Number
 At least 1 Symbol, symbol allowed --> !@#%&* _+=-
 Min 8 chars and Max 12 chars
- ✓ **Date** - `^(0?[1-9]|[12][0-9]|3[01])([\/-](0?[1-9]|1[012])([\/-]\d{4})$`
 Ex - **21/07/1995**
- ✓ **Time** - `^(0[1-9]|1[0-2]):([0-5][0-9]) ((a|p)m|(A|P)M)$`
 Ex - **11:45 AM (12 Hour Format)**
- ✓ **Aadhaar Number** - `^[2-9][0-9]{3}\s[0-9]{4}\s[0-9]{4}$`
 Ex- **4412 4585 8578**
- ✓ **Pancard Number** - `[A-Z]{5}[0-9]{4}[A-Z]{1}`
 Ex- **ABCPR4124L**

**Important
Regular
Expression
Pattern**



Validation Controls in ASP.NET : RegularExpressionValidator

The screenshot displays the Visual Studio IDE with the following components:

- Code File (RangeValidator.aspx.cs):**

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="RegularExpressionValidator.aspx.cs" Inherits="RegularExpressionValidator" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
Enter E-Mail ID - <asp:TextBox ID="username" runat="server"></asp:TextBox>
<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server" ControlToValidate="username"
ErrorMessage="Please enter valid email" ForeColor="Red" ValidationExpression="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,4}$">
</asp:RegularExpressionValidator>
<br />
<asp:Button ID="Button1" runat="server" Text="Check Value" OnClick="Button1_Click"/>
</div>
</form>
</body>
</html>

```
- Design View:** Shows the rendered HTML with a text box containing "Enter E-Mail ID -" and a "Check Value" button. A red error message "Please enter valid email" is displayed next to the text box.
- Properties Window:** Shows the document properties for the class, including Class, Culture, Debug, EnableSessionState, Id, Language (C#), MasterPageFile, Style, StyleSheetTheme, Theme, Title, Trace, and UICulture.

Validation Controls in ASP.NET : RegularExpressionValidator

OUTPUT

The screenshot displays the Visual Studio IDE with the following code in the `RegularExpressionValidator.aspx.cs` file:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

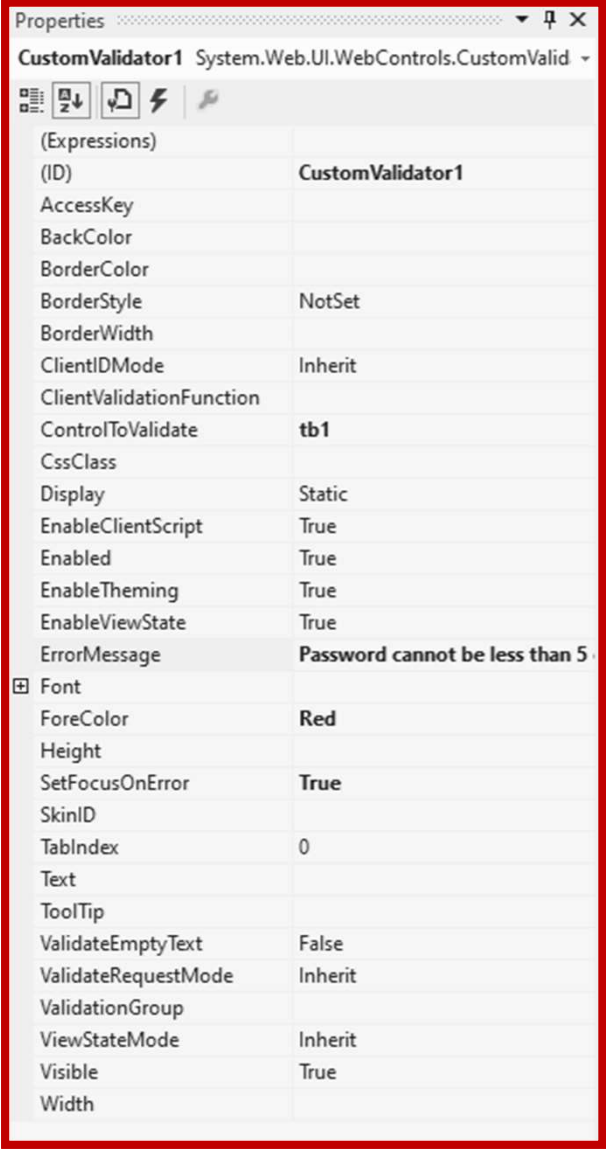
public partial class RegularExpressionValidator : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        if (Page.IsValid)
        {
            Response.Write("Data Check Successfully.");
        }
    }
}
```

Three browser screenshots illustrate the control's output:

- Top Screenshot:** Shows the initial state with an empty text input field and a "Check Value" button.
- Middle Screenshot:** Shows the input field containing "ankitrami@gmail.com". A red error message, "Please enter valid email", is displayed to the right of the input field.
- Bottom Screenshot:** Shows the input field containing "ankitramiblog@gmail.com". No error message is present, indicating a successful validation.

Validation Controls in ASP.NET : CustomValidator Control

- ✓ It is used to customize and implement data validation as per the requirements and conditions that occur.
- ✓ The developers use CustomValidationFor instance when they want to check whether the entered number is even or odd.
- ✓ Some of the major properties of custom validators are ClientValidationFunction and ValidateEmptyText.
- ✓ ClientValidationFunction is used to set the name of the function in the custom client-side scripting languages such as javascript or VBScript for validation purposes.
- ✓ ValidateEmptyText is used to set a Boolean value that checks whether the empty text is valid or not.
- ✓ Ex- `<asp:Customvalidator id="CustomValidator1" forecolor="Red" onservervalidate="CustomValidator1_ServerValidate" errormessage="Password cannot be less than 5 characters." controtovalidate="tb1" runat="server" SetFocusOnError="True"> </asp:Customvalidator>`



Properties	
CustomValidator1 System.Web.UI.WebControls.CustomValid	
(Expressions)	
(ID)	CustomValidator1
AccessKey	
BackColor	
BorderColor	
BorderStyle	NotSet
BorderWidth	
ClientIDMode	Inherit
ClientValidationFunction	
ControlToValidate	tb1
CssClass	
Display	Static
EnableClientScript	True
Enabled	True
EnableTheming	True
EnableViewState	True
ErrorMessage	Password cannot be less than 5
Font	
ForeColor	Red
Height	
SetFocusOnError	True
SkinID	
TabIndex	0
Text	
ToolTip	
ValidateEmptyText	False
ValidateRequestMode	Inherit
ValidationGroup	
ViewStateMode	Inherit
Visible	True
Width	

Validation Controls in ASP.NET : CustomValidator Control

The screenshot displays the Visual Studio IDE with the following components:

- Code View:** Shows the ASP.NET page markup for `CustomValidator.aspx.cs`. The page inherits from `CustomValidator`. The markup includes a form with a text box (`tb1`) and a button (`Button1`). A `CustomValidator` control is associated with the text box, using the `CustomValidator1_ServerValidate` method and displaying the error message "Password cannot be less than 5 characters." in red.
- Design View:** Shows the rendered web page. The text box contains the text "Enter Password" and the button is labeled "Validate". A red error message "Password cannot be less than 5 characters." is displayed next to the text box.
- Toolbox:** Lists various ASP.NET controls such as `Button`, `TextBox`, and `CustomValidator`.
- Properties Window:** Shows the properties of the selected control, including `Class`, `Culture`, `Language`, and `Style`.

Validation Controls in ASP.NET : CustomValidator Control

The screenshot displays the Visual Studio IDE with the following code in the `CustomValidator1_ServerValidate` method:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class CustomValidator : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args)
    {
        if (args.Value.Length < 5)
        {
            args.IsValid = false;
        }
        else
        {
            args.IsValid = true;
        }
    }
}

```

Three browser screenshots illustrate the validation process:

- Top Screenshot:** Shows the initial state with an empty password field and a "Validate" button.
- Middle Screenshot:** Shows the password field containing "Rami". A red error message is displayed: "Password cannot be less than 5 characters." The "Validate" button is disabled.
- Bottom Screenshot:** Shows the password field containing "Ankit". The "Validate" button is enabled, indicating a successful validation.

Validation Controls in ASP.NET : ValidationSummary Control

- ✓ It is a control that is used to show error messages.
- ✓ It also collects every type of validation control error message and is used by the other validation controls on a web page.
- ✓ It then shows the error message on the screen.
- ✓ The properties of validation summary control are Forecolor, DisplayMode, and HeaderText
- ✓ Forecolor is used to set the foreground color.
- ✓ DisplayMode is used to set the display mode of the control.
- ✓ HeaderText is used to set at the top of the summary.
- ✓ Ex- `<asp:ValidationSummary ID="ValidationSummary1" runat="server" />`



Properties ValidationSummary1 System.Web.UI.WebControls.Validation

(Expressions)	
(ID)	ValidationSummary1
AccessKey	
BackColor	
BorderColor	
BorderStyle	NotSet
BorderWidth	
ClientIDMode	Inherit
CssClass	
DisplayMode	BulletList
EnableClientScript	True
Enabled	True
EnableTheming	True
ableViewState	True
Font	
ForeColor	
HeaderText	
Height	
ShowMessageBox	False
ShowModelStateErrors	True
ShowSummary	True
ShowValidationErrors	True
SkinID	
TabIndex	0
ToolTip	
ValidateRequestMode	Inherit
ValidationGroup	
ViewStateMode	Inherit
Visible	True
Width	

Validation Controls in ASP.NET : ValidationSummary Control

The screenshot displays the Visual Studio IDE with the following components:

- Code Editor:** Shows the ASP.NET page code for `ValidationSummary.aspx`. The code includes:


```

      <!DOCTYPE html>
      <html xmlns="http://www.w3.org/1999/xhtml">
      <head runat="server">
      <title></title>
      </head>
      <body>
      <form id="form1" runat="server">
      <div>
      Enter Username - <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>
      <br />
      Enter Password - <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>
      <br />
      <asp:Button ID="Button1" runat="server" Text="Check Value"/>
      <asp:ValidationSummary ID="ValidationSummary1" runat="server" ForeColor="Red"/>
      </div>
      </form>
      </body>
      </html>
      
```
- Properties Window:** Shows the properties for the selected `ValidationSummary` control, including `Class`, `Culture`, `Debug`, `EnableSessionState`, `Id`, `Language` (set to C#), `MasterPageFile`, `Style`, `StyleSheetTheme`, `Theme`, `Title`, `Trace`, `TraceMode`, and `UICulture`.
- Design View:** Shows the rendered web page with two text boxes labeled "Enter Username" and "Enter Password", each with a red asterisk indicating a validation error. Below the text boxes is a "Check Value" button and a red error message box containing:
 - Error message 1.
 - Error message 2.

Validation Controls in ASP.NET : ValidationSummary Control

OUTPUT

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class ValidationSummary : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        if (Page.IsValid)
        {
            Response.Write("Data Check Successfully.");
        }
    }
}
    
```

The output shows three browser screenshots:

- Initial State:** A form with "Enter Username" and "Enter Password" fields and a "Check Value" button.
- Validation Failure:** After clicking "Check Value", the form displays two red error messages: "Please enter a user name" and "Please enter a password".
- Validation Success:** After clicking "Check Value" with valid input, the page displays "Data Check Successfully." and the form fields contain "ankit" and "ankitrami".

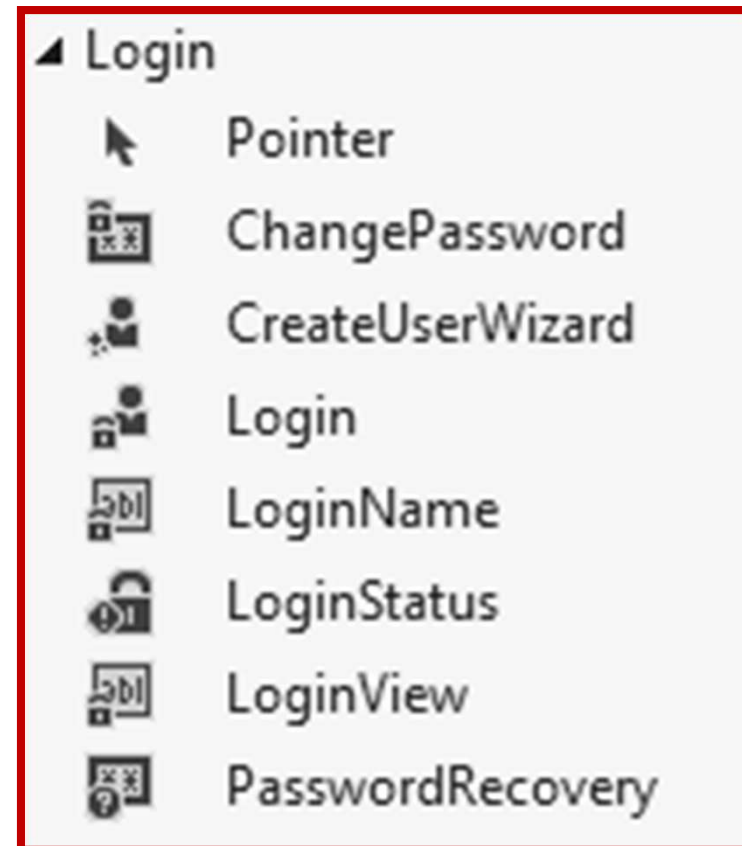
Login Controls in ASP.NET

- ✓ The ASP.NET login controls provide a robust login solution for ASP.NET Web applications without requiring programming.
- ✓ By default, login controls integrate with ASP.NET membership and forms authentication to help automate user authentication for a Web site.
- ✓ It provides you with a ready-to-use user interface that queries the user name and password from the user and offers a Log In button for login.
- ✓ It validates user credentials against the membership API and encapsulates the basic forms authentication functionality like redirecting back to the original requested page in a restricted area of your application after the successful login.
- ✓ The Login control displays a user interface for user authentication. The Login control contains text boxes for the user name and password and a check box that allows users to indicate whether they want the server to store their identity using ASP.NET membership and automatically be authenticated the next time they visit the site.
- ✓ The Login control has properties for customized display, customized messages, and links to other pages where users can change their password or recover a forgotten password.
- ✓ The Login control can be used as a standalone control on a main or home page, or you can use it on a dedicated login page. If you use the Login control with ASP.NET membership, you do not need to write code to perform authentication.
- ✓ Reference Link - <http://www.shotdev.com/aspnet/aspnet-vbnet-login-control/>

Login Controls in ASP.NET

✓ ASP.NET provides the following Login Controls:

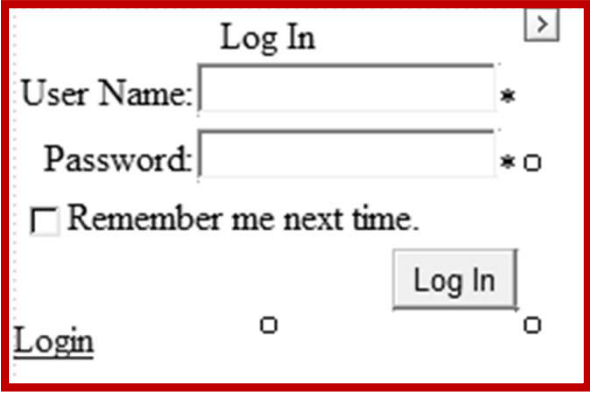
1. Login
2. LoginView
3. LoginStatus
4. Loginname
5. PasswordRecovery
6. ChangePassword
7. CreateUserWizard



Login Controls in ASP.NET

❑ 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 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



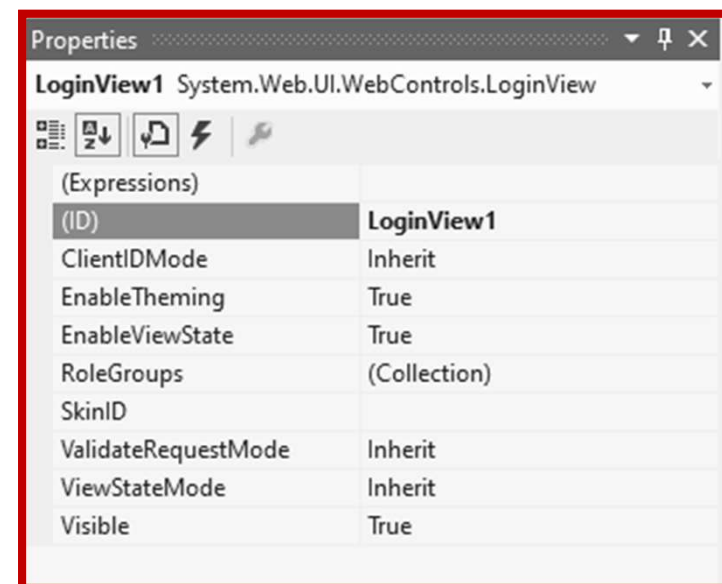
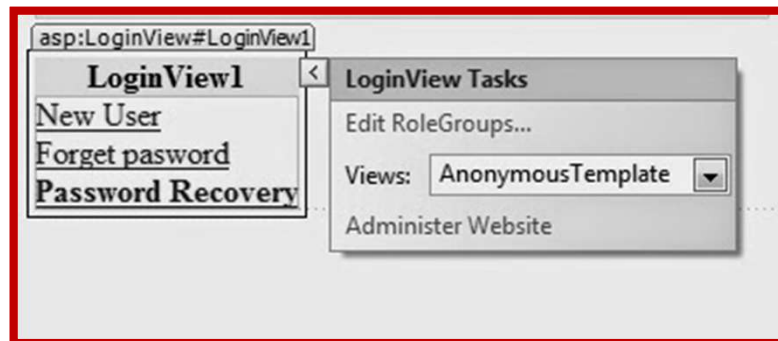
Properties	
Login1 System.Web.UI.WebControls.Login	
(Expressions)	
(ID)	Login1
AccessKey	
BackColor	
BorderColor	
BorderPadding	1
BorderStyle	NotSet
BorderWidth	
CheckBoxStyle	
ClientIDMode	Inherit
CreateUserIconUrl	
CreateUserText	
CreateUserUrl	
CssClass	
DestinationPageUrl	
DisplayRememberMe	True
Enabled	True
EnableTheming	True
ableViewState	True
FailureAction	Refresh
FailureText	Your login attempt was not su
FailureTextStyle	
Font	
ForeColor	
Height	
HelpPageIconUrl	
HelpPageText	
HelpPageUrl	
HyperLinkStyle	

Properties	
Login1 System.Web.UI.WebControls.Login	
HyperLinkStyle	
InstructionText	
InstructionTextStyle	
LabelStyle	
LoginButtonImageUrl	
LoginButtonStyle	
LoginButtonText	Log In
LoginButtonType	Button
MembershipProvider	
Orientation	Vertical
PasswordLabelText	Password:
PasswordRecoveryIconUrl	
PasswordRecoveryText	
PasswordRecoveryUrl	
PasswordRequiredErrorMessage	Password is required.
RememberMeSet	False
RememberMeText	Remember me next time.
RenderOuterTable	True
SkinID	
TabIndex	0
TextBoxStyle	
TextLayout	TextOnLeft
TitleText	Log In
TitleTextStyle	
ToolTip	
UserName	
UserNameLabelText	User Name:
UserNameRequiredErrorMessage	User Name is required.
ValidateRequestMode	Inherit
ValidatorTextStyle	
ViewStateMode	Inherit
Visible	True
VisibleWhenLoggedIn	True
Width	

Login Controls in ASP.NET

❑ LoginView Control

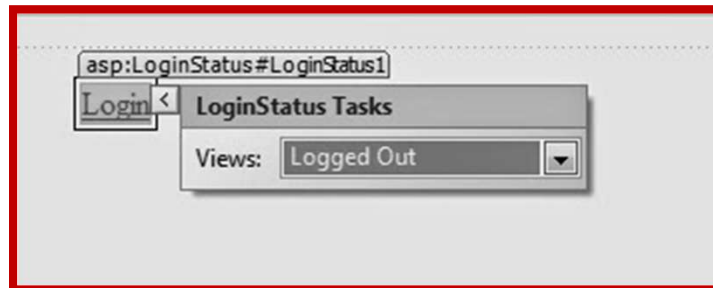
- ✓ The LoginView Control is a web server control ,Which is used to display two different views of a web page of any website , depending on whether the any user has logged on to a web page as anonymous user or registered user .If the user is authenticated,the control displays the appropriate to the person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



Login Controls in ASP.NET

❑ LoginStatus Control

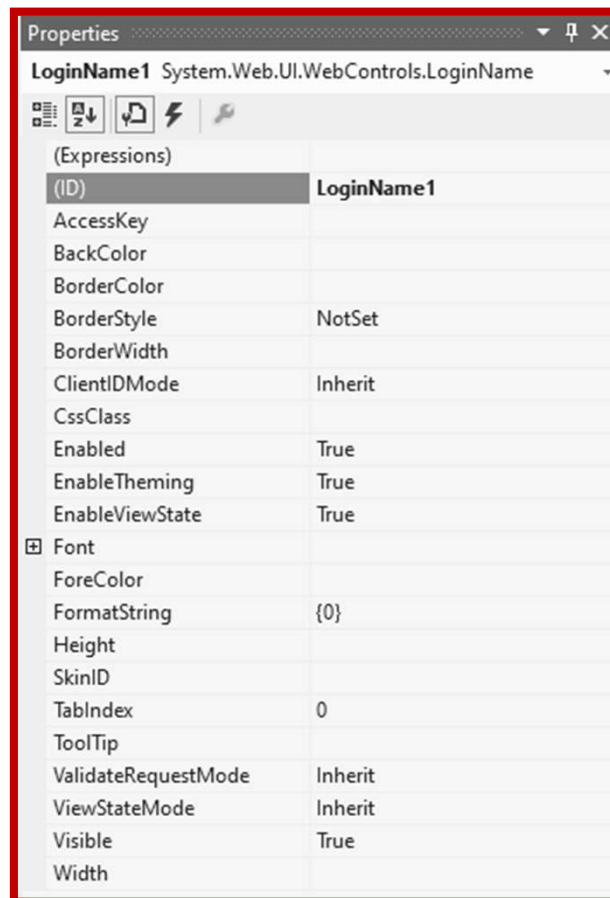
- ✓ The LoginStatus control specifies whether a particular user has logged on the website or not . When the user is not logged in, this control display the Login text as a hyperlink. When the user is logged in ,this control display the logout text as a hyperlink.
- ✓ To do this ,Login Status control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



Login Controls in ASP.NET

❑ Loginname Control

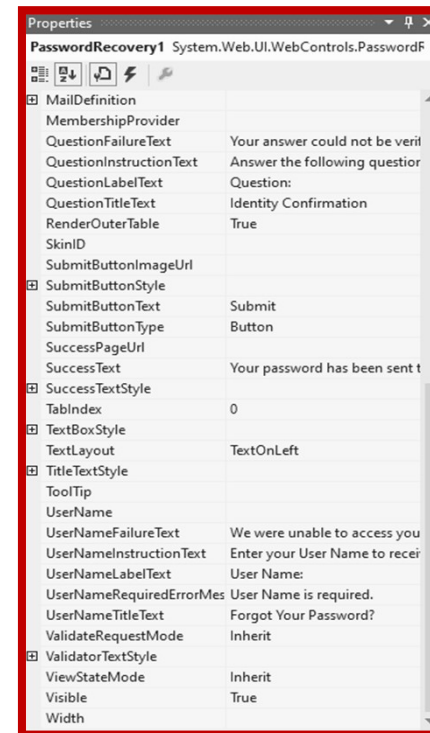
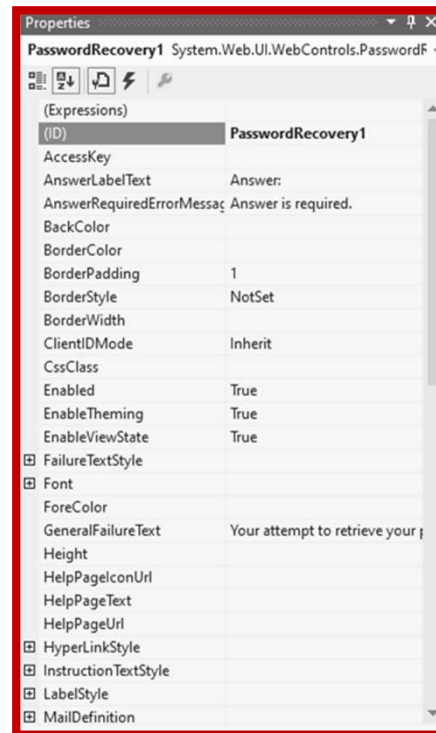
- ✓ The LoginName Control is responsible for display the names of all the authenticated users.If no users id logged in ,then this control is not displayed on the page.
- ✓ This control uses the page User Identity.Name namespace to return the value for the user's name.



Login Controls in ASP.NET

Passwordrecovery Control

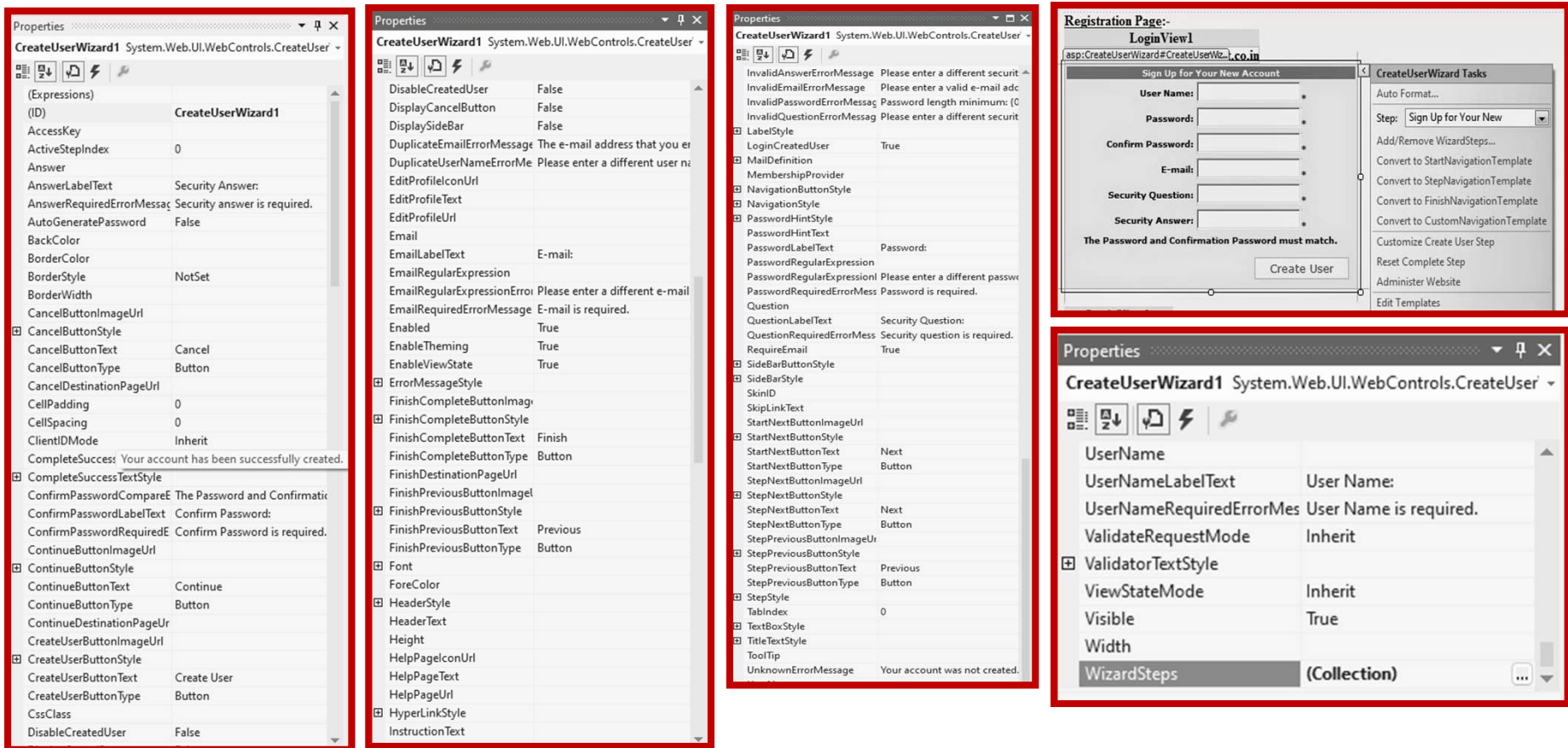
- ✓ The Passwordrecovery control is used to recover or reset the forgotten password of a user . This control does not display the password on the browser, but it sends the password to the respective email address whatever you have specified at the time of registration.
- ✓ This control has included three views as given below:-
- ✓ UserName :- It refers to the view that accepts the username of a user.
- ✓ Question :- It accepts the security questions asked from the users.
- ✓ Success :- It display a message to the user that retrieved password has been set to the user.



📢 Login Controls in ASP.NET

❑ CreateUserWizard Control

- ✓ This control uses the membership service to create a new user in the membership data store. The CreateUserWizard control is provided 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.



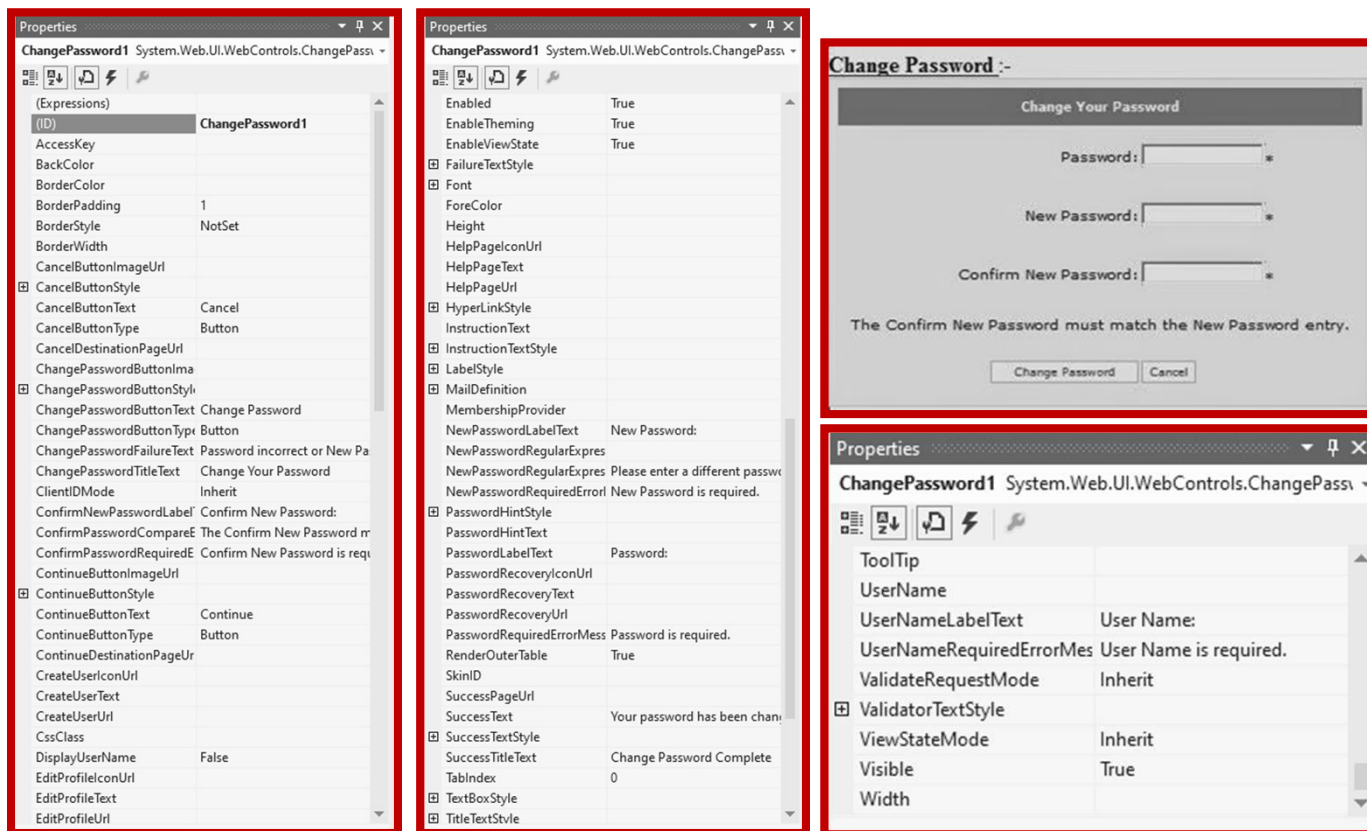
The image displays four screenshots related to the CreateUserWizard control in ASP.NET:

- Properties Window (Left):** Shows the properties of the CreateUserWizard control. Key properties include:
 - AccessKey:** (ID)
 - ActiveStepIndex:** 0
 - Answer:** Security Answer:
 - AnswerRequiredErrorMessage:** Security answer is required.
 - AutoGeneratePassword:** False
 - BackColor:**
 - BorderColor:**
 - BorderStyle:** NotSet
 - BorderWidth:**
 - CancelButtonImageUri:**
 - CancelButtonStyle:**
 - CancelButtonText:** Cancel
 - CancelButtonType:** Button
 - CancelDestinationPageUri:**
 - CellPadding:** 0
 - CellSpacing:** 0
 - ClientIDMode:** Inherit
 - CompleteSuccessText:** Your account has been successfully created.
 - CompleteSuccessTextStyle:**
 - ConfirmPasswordCompareErrorText:** The Password and Confirmation Password must match.
 - ConfirmPasswordLabelText:** Confirm Password:
 - ConfirmPasswordRequiredErrorMessage:** Confirm Password is required.
 - ContinueButtonImageUri:**
 - ContinueButtonStyle:**
 - ContinueButtonText:** Continue
 - ContinueButtonType:** Button
 - ContinueDestinationPageUri:**
 - CreateUserButtonImageUri:**
 - CreateUserButtonStyle:**
 - CreateUserButtonText:** Create User
 - CreateUserButtonType:** Button
 - CssClass:**
 - DisableCreatedUser:** False
- Properties Window (Middle-Left):** Shows the properties of the CreateUserWizard control. Key properties include:
 - DisableCreatedUser:** False
 - DisplayCancelButton:** False
 - DisplaySideBar:** False
 - DuplicateEmailErrorMessage:** The e-mail address that you entered is already in use.
 - DuplicateUserNameErrorMessage:** Please enter a different user name.
 - EditProfileIconUri:**
 - EditProfileText:**
 - EditProfileUri:**
 - Email:**
 - EmailLabelText:** E-mail:
 - EmailRegularExpression:**
 - EmailRegularExpressionErrorMessage:** Please enter a different e-mail address.
 - EmailRequiredErrorMessage:** E-mail is required.
 - Enabled:** True
 - EnableTheming:** True
 - EnableViewState:** True
 - ErrorMessageStyle:**
 - FinishCompleteButtonImageUri:**
 - FinishCompleteButtonText:** Finish
 - FinishCompleteButtonType:** Button
 - FinishDestinationPageUri:**
 - FinishPreviousButtonImageUri:**
 - FinishPreviousButtonStyle:**
 - FinishPreviousButtonText:** Previous
 - FinishPreviousButtonType:** Button
 - Font:**
 - ForeColor:**
 - HeaderStyle:**
 - HeaderText:**
 - Height:**
 - HelpPageIconUri:**
 - HelpPageText:**
 - HelpPageUri:**
 - HyperLinkStyle:**
 - InstructionText:**
- Properties Window (Middle-Right):** Shows the properties of the CreateUserWizard control. Key properties include:
 - InvalidAnswerErrorMessage:** Please enter a different security question.
 - InvalidEmailErrorMessage:** Please enter a valid e-mail address.
 - InvalidPasswordErrorMessage:** Password length minimum: [0].
 - InvalidQuestionErrorMessage:** Please enter a different security question.
 - LabelStyle:**
 - LoginCreatedUser:** True
 - MailDefinition:**
 - MembershipProvider:**
 - NavigationButtonStyle:**
 - NavigationStyle:**
 - PasswordHintStyle:**
 - PasswordHintText:** Password:
 - PasswordLabelText:** Password:
 - PasswordRegularExpression:**
 - PasswordRegularExpressionErrorMessage:** Please enter a different password.
 - PasswordRequiredErrorMessage:** Password is required.
 - Question:**
 - QuestionLabelText:** Security Question:
 - QuestionRequiredErrorMessage:** Security question is required.
 - RequireEmail:** True
 - SideBarButtonStyle:**
 - SideBarStyle:**
 - SkinID:**
 - SkipLinkText:**
 - StartNextButtonImageUri:**
 - StartNextButtonStyle:**
 - StartNextButtonText:** Next
 - StartNextButtonType:** Button
 - StepNextButtonImageUri:**
 - StepNextButtonStyle:**
 - StepNextButtonText:** Next
 - StepNextButtonType:** Button
 - StepPreviousButtonImageUri:**
 - StepPreviousButtonStyle:**
 - StepPreviousButtonText:** Previous
 - StepPreviousButtonType:** Button
 - StepStyle:**
 - TabIndex:** 0
 - TextBoxStyle:**
 - TitleTextStyle:**
 - ToolTip:**
 - UnknownErrorMessage:** Your account was not created.
- Registration Page (Right):** Shows a registration page titled "Sign Up for Your New Account" with the following fields:
 - User Name:** [Text Box]
 - Password:** [Text Box]
 - Confirm Password:** [Text Box]
 - E-mail:** [Text Box]
 - Security Question:** [Text Box]
 - Security Answer:** [Text Box]
 A "Create User" button is present at the bottom. A message below the fields states: "The Password and Confirmation Password must match." To the right, there is a "CreateUserWizard Tasks" panel with options like "Auto Format...", "Add/Remove WizardSteps...", "Convert to StartNavigationTemplate", etc.
- Properties Window (Bottom-Right):** Shows the properties of the CreateUserWizard control. Key properties include:
 - UserName:**
 - UserNameLabelText:** User Name:
 - UserNameRequiredErrorMessage:** User Name is required.
 - ValidateRequestMode:** Inherit
 - ValidatorTextStyle:**
 - ViewStateMode:** Inherit
 - Visible:** True
 - Width:**
 - WizardSteps:** (Collection)

Login Controls in ASP.NET

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.

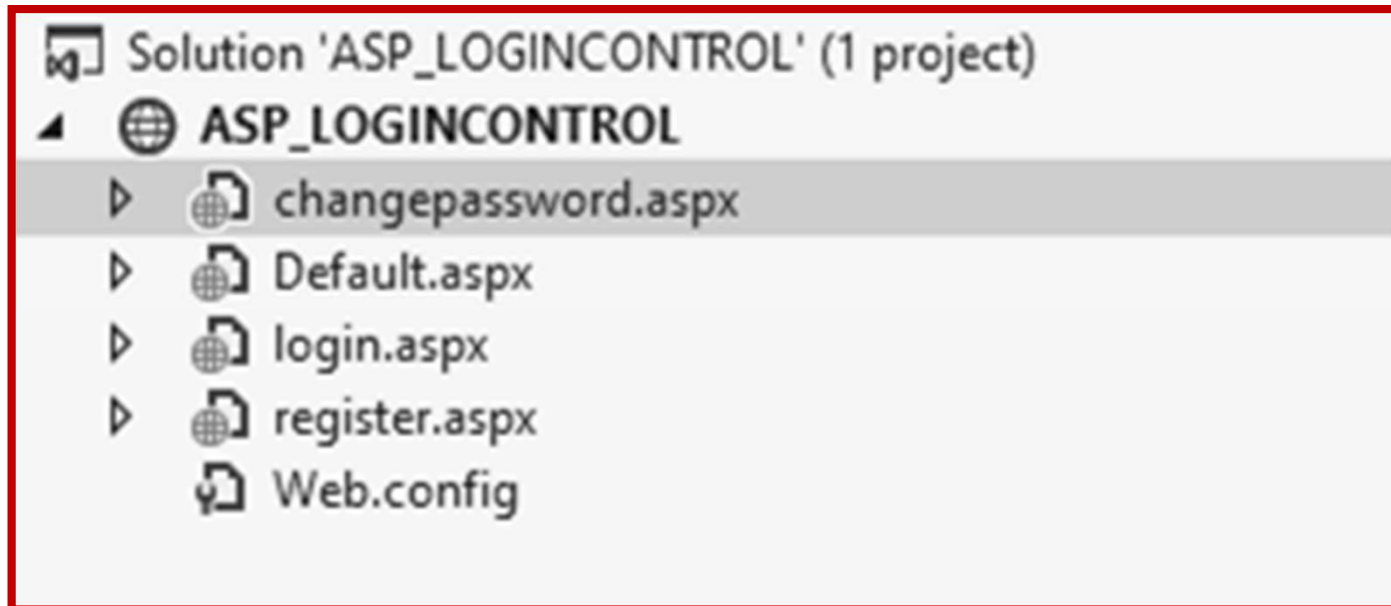


The image displays three screenshots related to the ASP.NET ChangePassword control:

- Left Screenshot:** Shows the Properties window for the ChangePassword1 control. The (ID) is ChangePassword1. Key properties include:
 - Enabled: True
 - EnableTheming: True
 - EnableViewState: True
 - FailureTextStyle: (expanded)
 - Font: (expanded)
 - HyperLinkStyle: (expanded)
 - InstructionTextStyle: (expanded)
 - LabelStyle: (expanded)
 - MailDefinition: (expanded)
 - MembershipProvider: (expanded)
 - MemberShipProvider: New Password:
 - NewPasswordLabelText: Password:
 - NewPasswordRegularExpression: Please enter a different password.
 - NewPasswordRequiredError: New Password is required.
 - PasswordHintStyle: (expanded)
 - PasswordHintText: Password:
 - PasswordRecoveryIconUrl: (expanded)
 - PasswordRecoveryText: Password is required.
 - PasswordRecoveryUrl: (expanded)
 - RenderOuterTable: True
 - SkinID: (expanded)
 - SuccessPageUrl: (expanded)
 - SuccessText: Your password has been changed.
 - SuccessTextStyle: (expanded)
 - SuccessTitleText: Change Password Complete
 - TabIndex: 0
 - TextBoxStyle: (expanded)
 - TitleTextStyle: (expanded)
- Middle Screenshot:** Shows the rendered output of the ChangePassword control. The title is "Change Password :-". The form contains three input fields: "Password:", "New Password:", and "Confirm New Password:". Below the fields is a message: "The Confirm New Password must match the New Password entry." At the bottom are two buttons: "Change Password" and "Cancel".
- Right Screenshot:** Shows the Properties window for the ChangePassword1 control with the following visible properties:
 - ToolTip: (empty)
 - UserName: (empty)
 - UserNameLabelText: User Name:
 - UserNameRequiredErrorMessage: User Name is required.
 - ValidateRequestMode: Inherit
 - ValidatorTextStyle: (expanded)
 - ViewStateMode: Inherit
 - Visible: True
 - Width: (empty)

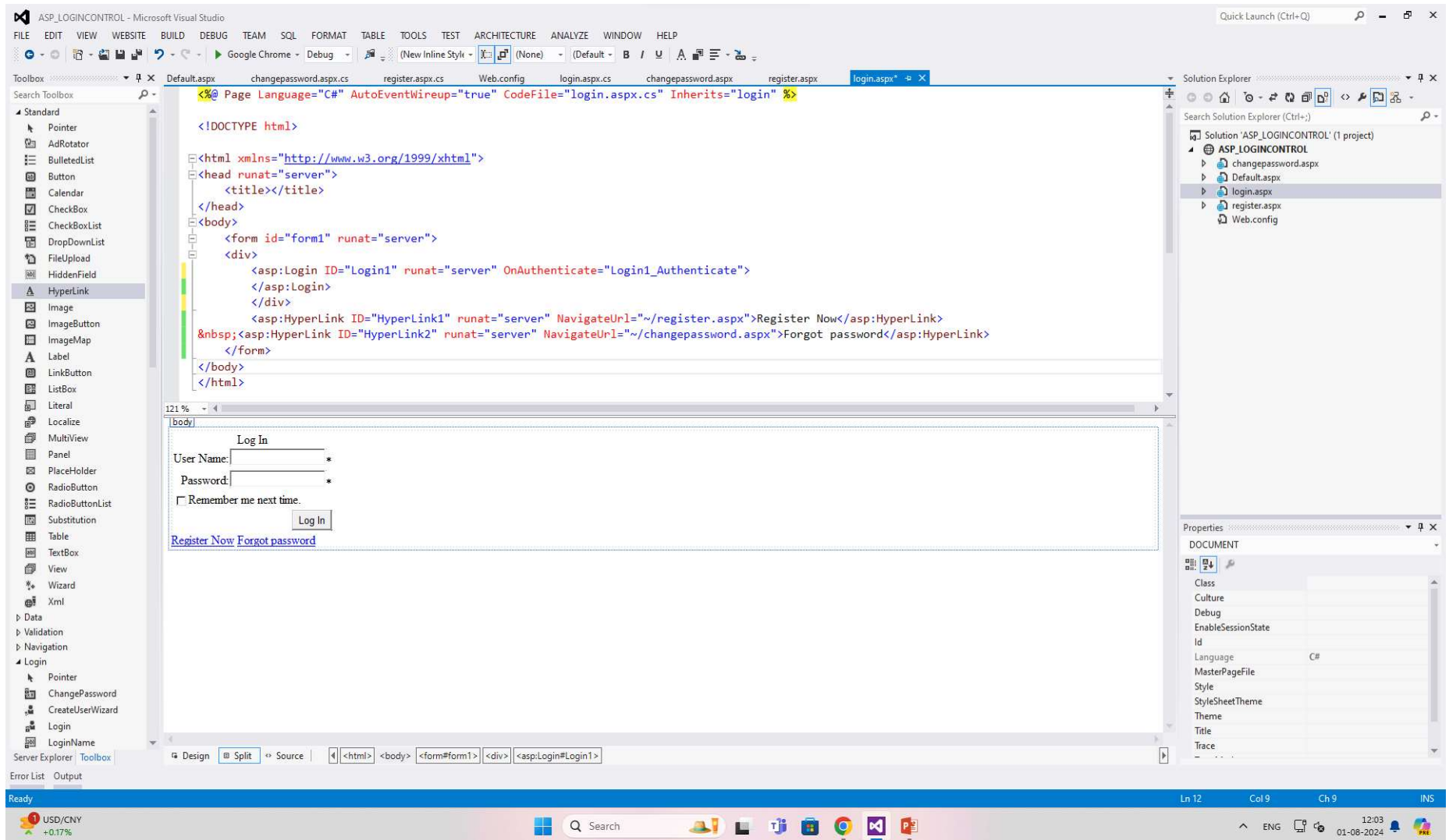
Login Controls Example in ASP.NET

□ Total Pages for Website



📢 Login Controls Example in ASP.NET

📄 Login.aspx Page Design



The screenshot displays the Visual Studio IDE for an ASP.NET project named 'ASP_LOGINCONTROL'. The main editor shows the source code for 'login.aspx', which includes a form with a login control and two hyperlinks for registration and password recovery.

```

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

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:Login ID="Login1" runat="server" OnAuthenticate="Login1_Authenticate">
</asp:Login>
</div>
<asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="~/register.aspx">Register Now</asp:HyperLink>
&nbsp;&nbsp;&nbsp;<asp:HyperLink ID="HyperLink2" runat="server" NavigateUrl="~/changepassword.aspx">Forgot password</asp:HyperLink>
</form>
</body>
</html>
    
```

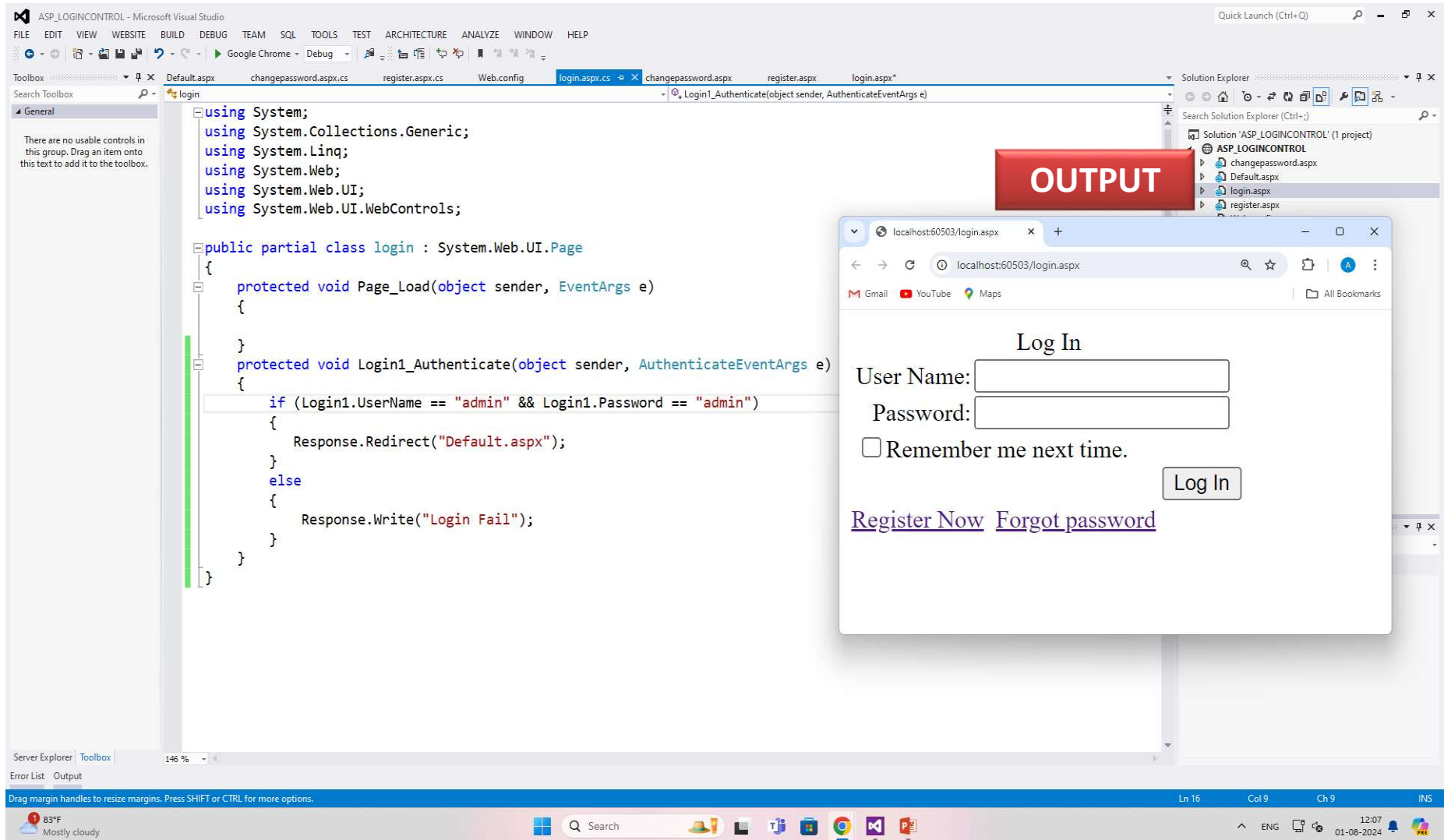
The design view below the code shows a 'Log In' form with the following elements:

- User Name:
- Password:
- Remember me next time.
- Log In button
- Register Now [Forgot password](#)

The Solution Explorer on the right shows the project structure, and the Properties window at the bottom right shows the document properties.

Login Controls Example in ASP.NET

❑ Login.aspx Page C# Code



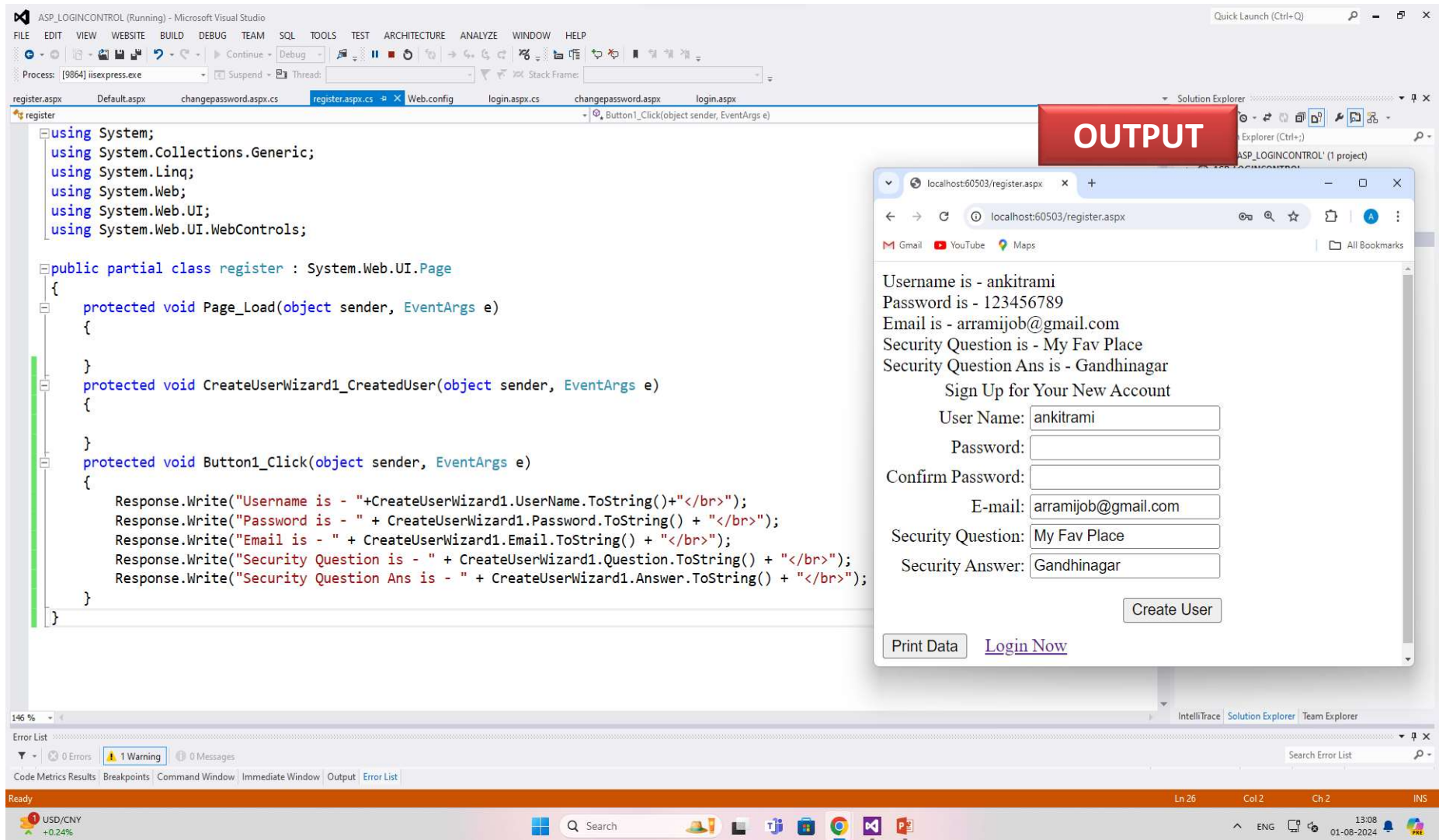
The image shows a screenshot of Microsoft Visual Studio with the C# code for the login.aspx page. The code includes using statements for System, System.Collections.Generic, System.Linq, System.Web, System.Web.UI, and System.Web.UI.WebControls. The main class is `login` which inherits from `System.Web.UI.Page`. It contains two methods: `Page_Load` and `Login1_Authenticate`. The `Login1_Authenticate` method checks if the username is "admin" and the password is "admin". If true, it redirects to "Default.aspx". Otherwise, it writes "Login Fail".

Overlaid on the code is a red box with the word "OUTPUT". To the right, a browser window shows the rendered login page. The page has a title "Log In", a "User Name:" label with a text input field, a "Password:" label with a password input field, a "Remember me next time." checkbox, a "Log In" button, and two links: "Register Now" and "Forgot password".

Login Controls Example in ASP.NET

Register.aspx Page C# Code

OUTPUT



The screenshot displays the Visual Studio IDE with the C# code for the Register.aspx page. The code includes necessary namespace imports and implements the Page_Load, CreateUserWizard1_CreatedUser, and Button1_Click methods. The Button1_Click method uses Response.Write to output the user details.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class register : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void CreateUserWizard1_CreatedUser(object sender, EventArgs e)
    {
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        Response.Write("Username is - " + CreateUserWizard1.UserName.ToString() + "<br>");
        Response.Write("Password is - " + CreateUserWizard1.Password.ToString() + "<br>");
        Response.Write("Email is - " + CreateUserWizard1.Email.ToString() + "<br>");
        Response.Write("Security Question is - " + CreateUserWizard1.Question.ToString() + "<br>");
        Response.Write("Security Question Ans is - " + CreateUserWizard1.Answer.ToString() + "<br>");
    }
}
    
```

The browser output shows the following information:

```

Username is - ankitrami
Password is - 123456789
Email is - arramijob@gmail.com
Security Question is - My Fav Place
Security Question Ans is - Gandhinagar
    
```

Below the output, there is a "Sign Up for Your New Account" section with input fields for User Name (filled with "ankitrami"), Password, Confirm Password, E-mail (filled with "arramijob@gmail.com"), Security Question (filled with "My Fav Place"), and Security Answer (filled with "Gandhinagar"). A "Create User" button is present at the bottom of this section. There are also "Print Data" and "Login Now" links at the bottom of the browser window.

📢 Login Controls Example in ASP.NET

📄 changepassword.aspx Page Design

The screenshot displays the Visual Studio IDE with the ASP.NET project 'ASP_LOGINCONTROL'. The 'changepassword.aspx' file is open in Design view. The page contains two main sections:

- Change Your Password:** A form with three text boxes labeled 'Password:', 'New Password:', and 'Confirm New Password:'. Below the 'Confirm New Password' field is a validation message: 'The Confirm New Password must match the New Password entry.' There are 'Change Password' and 'Cancel' buttons.
- Forgot Your Password?:** A section with the text 'Enter your User Name to receive your password.' and a 'User Name:' text box with a '*' validation symbol. A 'Submit' button is located to the right. Below this is a blue link labeled 'Login now'.

A red box labeled 'OUTPUT' points to a browser window showing the rendered page at localhost:60503/changepasswo... The browser view shows the rendered HTML, including the form fields and buttons.

📢 Login Controls Example in ASP.NET

📄 Default.aspx Home Page Design

The screenshot displays the Visual Studio IDE for an ASP.NET project named 'ASP_LOGINCONTROL'. The main window shows the source code for 'Default.aspx', which includes the following HTML and ASP.NET controls:

```

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
Admin<asp:LoginName ID="LoginName1" runat="server" />
<br />
<asp:LoginStatus ID="LoginStatus1" runat="server" />
</div>
</form>
</body>
</html>
    
```

A red box labeled 'OUTPUT' points to a browser window showing the rendered page. The browser displays the text 'Admin' followed by a blue underlined link 'Logout'. Below the browser window, the Visual Studio Properties window shows the 'ID' property of the selected 'DIV' element.

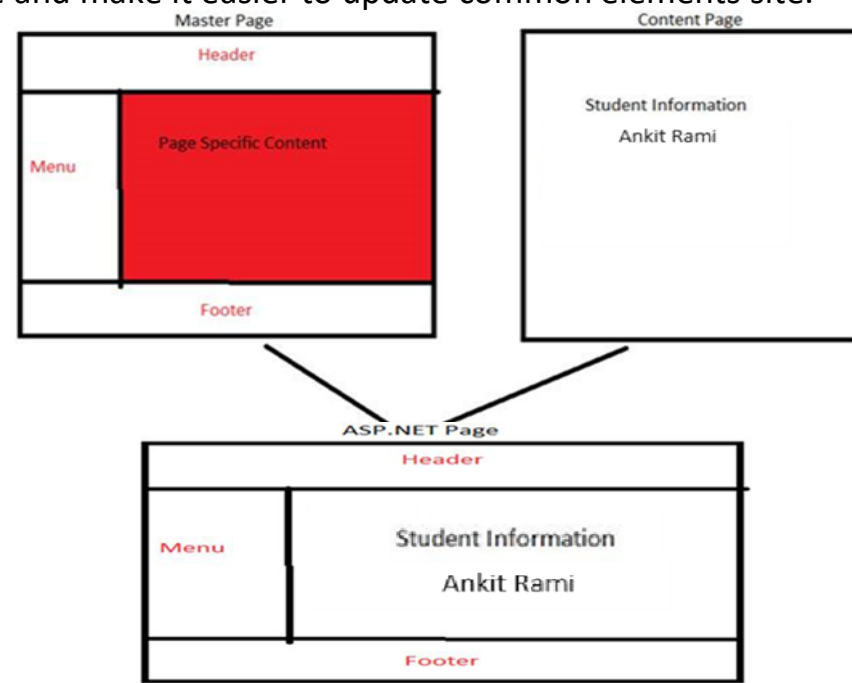
Property	Value
(id)	
accesskey	
aria-activedescendant	
aria-atomic	False
aria-autocomplete	none
aria-busy	False
aria-checked	undefined
aria-controls	
aria-describedby	
aria-disabled	False
aria-dropeffect	none
aria-expanded	undefined

Master Page in ASP.NET

- ✓ 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.
- ✓ The master page defines placeholders for the content, which can be overridden by content pages. The output result is a combination of the master page and the content page.
- ✓ Master page allows you to create a consistent look and behavior across all pages in a web application.
- ✓ Master Page Design is common for all the pages.
- ✓ Master page actually consists of two pieces, the master page itself and one or more content pages.
- ✓ A master page is an ASP.NET file with the extension .master. ContentPlaceholder control, which defines a region on the master page, is where the content pages can plug in page specific content.
- ✓ Master pages help create a uniform user experience and make it easier to update common elements site.

Important points about master pages

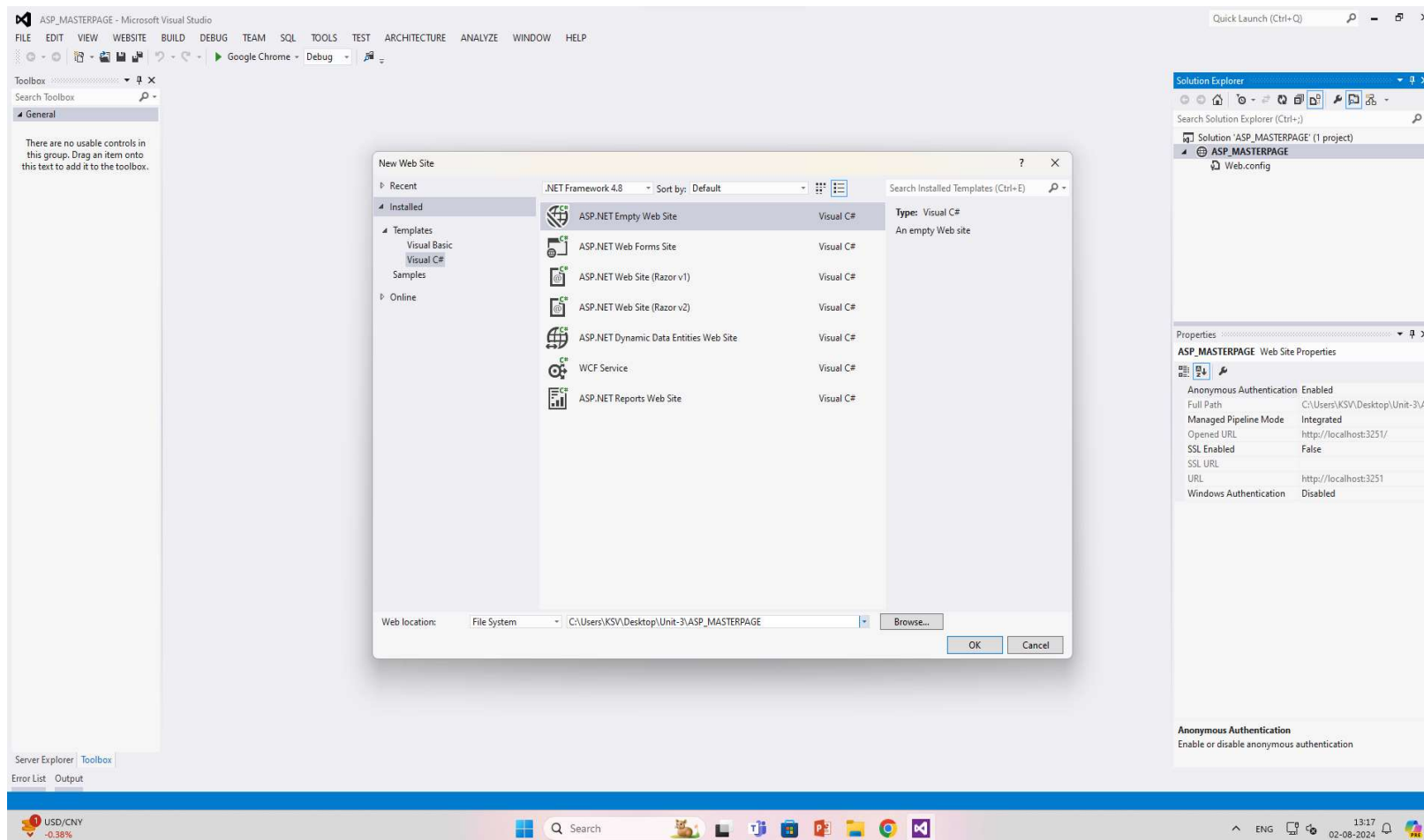
- ✓ The @ Master directive defines it as a master page.
- ✓ ContentPlaceHolder control is only available for master pages.
- ✓ ASP.NET page that uses master pages is called content pages.
- ✓ Master page - Content page relationship, let's look at the picture given below.



Master Page in ASP.NET

Step 1

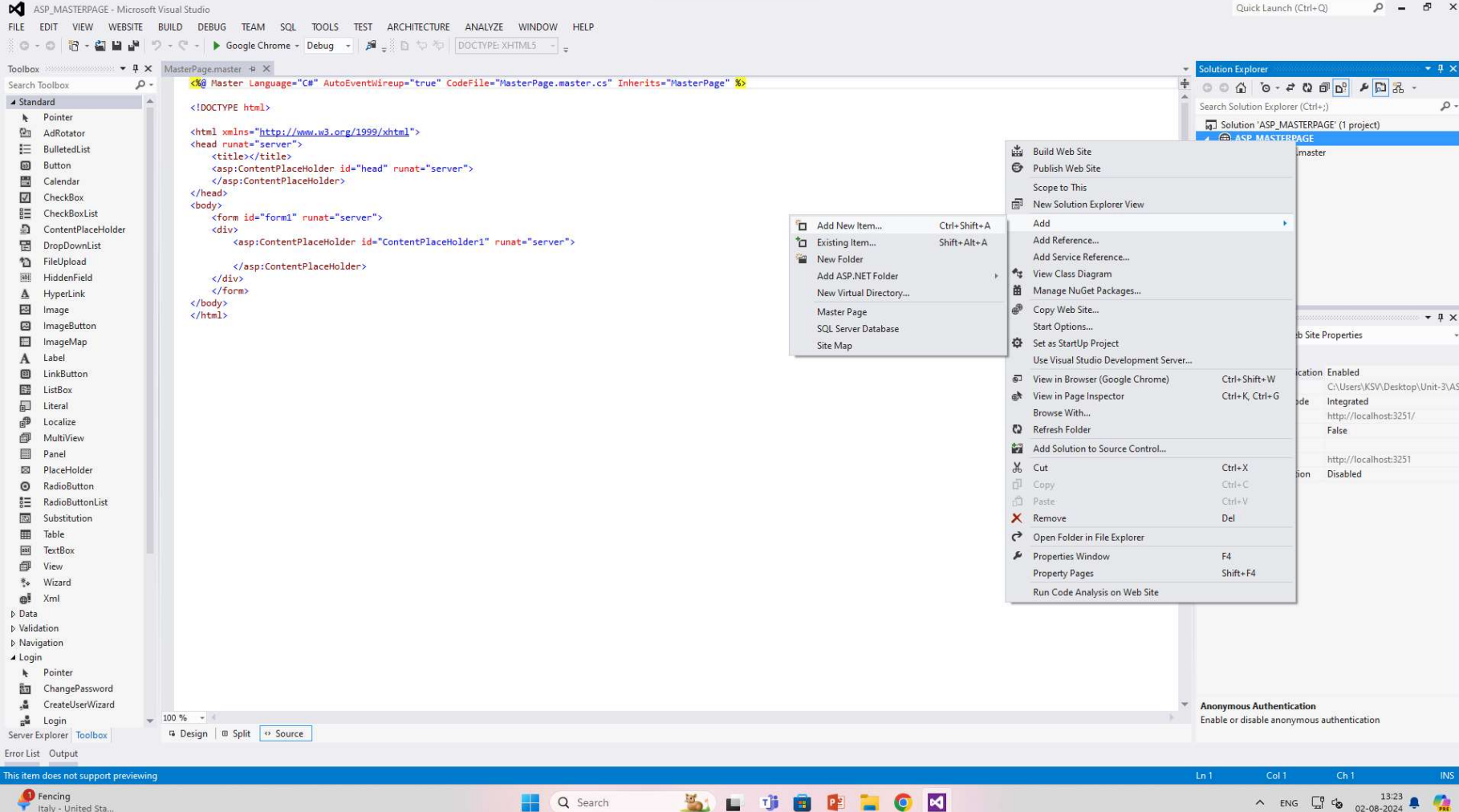
- ✓ Create new project in Visual Studio 2012
- ✓ Go to File-> New-> Web Site-> Visual C#->ASP.NET Empty



Master Page in ASP.NET

Step 2

- ✓ Create Master Page
- ✓ Project name-> Add-> Add New Item->Visual C#-> Master Page->Write Master Page Name-> Add

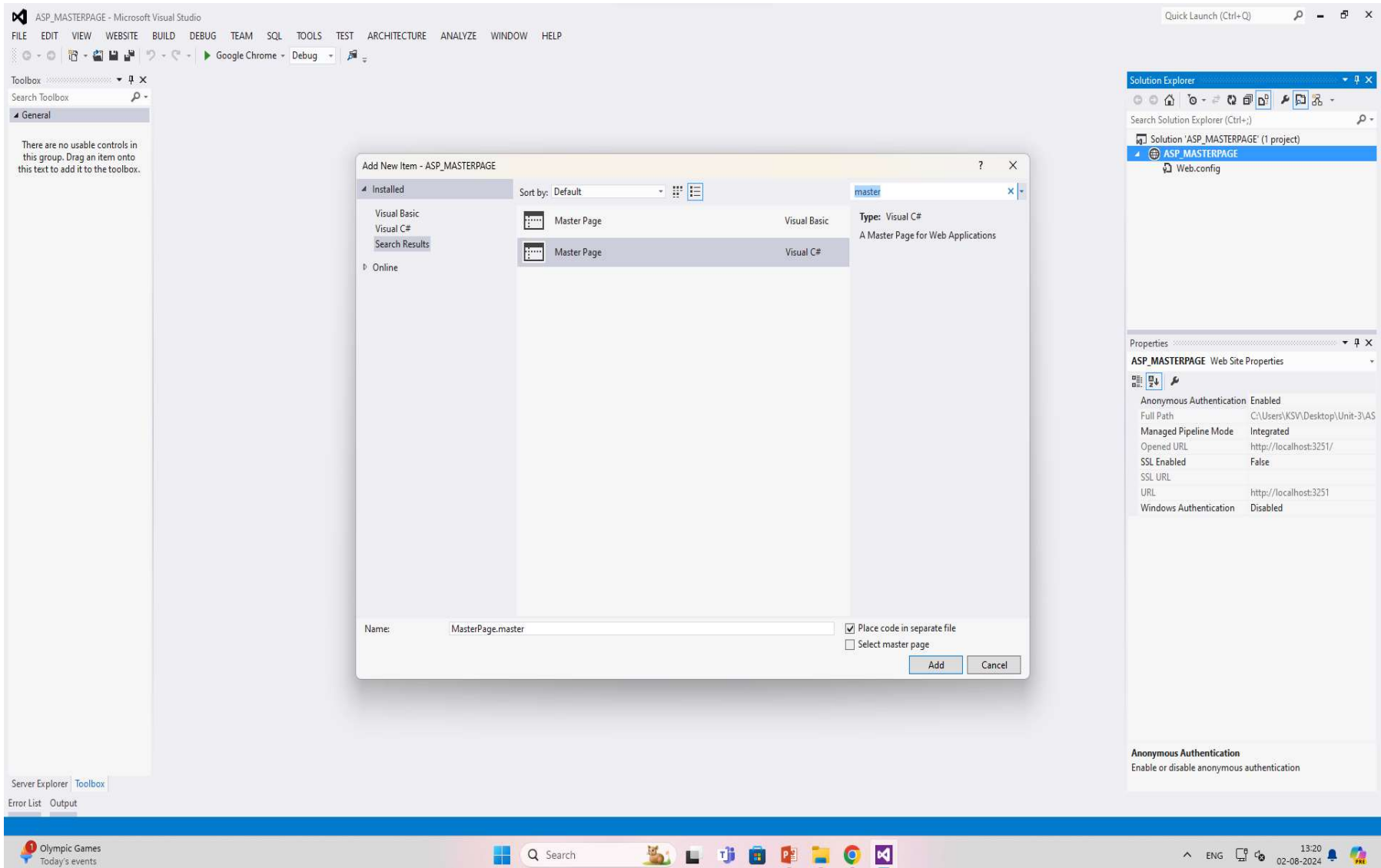


The screenshot shows the Visual Studio IDE with the following elements:

- Code Editor:** Displays the code for `MasterPage.master`. The code includes a form with a content placeholder:


```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
<asp:ContentPlaceHolder id="head" runat="server">
</asp:ContentPlaceHolder>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
</div>
</asp:ContentPlaceHolder>
</div>
</form>
</body>
</html>
```
- Solution Explorer:** Shows the project structure for `ASP_MASTERPAGE`. A context menu is open over the project, with the `Add` option selected, opening a sub-menu where `Add New Item...` is highlighted.
- Toolbox:** Shows various ASP.NET controls like Pointer, AdRotator, BulletedList, Button, Calendar, etc.
- Properties Window:** Shows the properties for the selected item, including `Anonymous Authentication`.

Master Page in ASP.NET



The screenshot displays the Microsoft Visual Studio IDE with the 'Add New Item' dialog box open for the project 'ASP_MASTERPAGE'. The dialog shows the 'Installed' category selected, with 'Master Page' items listed under both 'Visual Basic' and 'Visual C#' types. The 'Visual C#' item is selected, and its description reads: 'Type: Visual C# A Master Page for Web Applications'. The 'Name' field is set to 'MasterPage.master', and the checkbox 'Place code in separate file' is checked. The 'Add' button is highlighted.

The background shows the Solution Explorer with the project structure, the Properties window displaying 'ASP_MASTERPAGE Web Site Properties' (e.g., Anonymous Authentication: Enabled, Managed Pipeline Mode: Integrated), and the Windows taskbar at the bottom.

📢 Master Page in ASP.NET

The screenshot displays the Visual Studio IDE with the following components:

- Source Code (MasterPage.master):**

```

<% Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
<asp:ContentPlaceHolder id="head" runat="server">
</asp:ContentPlaceHolder>
</head>
<body>
<form id="form1" runat="server">
<div>
<h1><asp:Image ID="Image1" runat="server" Height="129px" ImageUrl="~/IMG_20240517_145717.jpg" Width="129px" BorderColor="#CC0000" BorderStyle="Solid" BorderWidth="5px" />
<asp:Label ID="Label1" runat="server" style="z-index: 1; left: 154px; top: 66px; position: absolute" Text="ASP.NET LEARN WITH ANKIT RAMI"></asp:Label>
</h1><asp:Menu ID="Menu1" runat="server" Font-Size="X-Large" ForeColor="white" Orientation="Horizontal">
<Items>
<asp:MenuItem NavigateUrl="https://ksv.ac.in" Text="KSV University" Value="KSV University"></asp:MenuItem>
<asp:MenuItem NavigateUrl="https://npccsm.in" Text="NPCCSM BCA" Value="NPCCSM BCA"></asp:MenuItem>
<asp:MenuItem NavigateUrl="https://bpccs.org" Text="BPCCS BCA" Value="BPCCS BCA"></asp:MenuItem>
</Items>
<StaticHoverStyle BackColor="#669900" BorderColor="#FF9966" BorderStyle="Dotted" BorderWidth="5px" ForeColor="white" />
<StaticMenuStyle BackColor="#6699FF" BorderColor="#FF3300" BorderStyle="Solid" BorderWidth="5px" Height="20px" HorizontalPadding="15px" VerticalPadding="15px" />
</asp:Menu>
<asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
</asp:ContentPlaceHolder>
</div>
</form>
</body>
</html>

```
- Design View:** Shows a rendered page with a profile picture of Ankit Rami, the text "ASP.NET LEARN WITH ANKIT RAMI", and a navigation menu with items for "KSV University", "NPCCSM BCA", and "BPCCS BCA". A content placeholder is visible at the bottom.
- Properties Window:** Shows the properties for the selected `ContentPlaceHolder1` control, including `ClientIDMode` (Inherit), `EnableViewState` (True), `ValidateRequestMode` (Inherit), `ViewStateMode` (Inherit), and `Visible` (True).

📢 Master Page in ASP.NET

Step 3

- ✓ Adding the Content Pages to Master Page
- ✓ Master Page -> Add Content Page.

The screenshot displays the Visual Studio IDE with the following components:

- Toolbox:** Shows various ASP.NET controls like Pointer, AdRotator, Button, Calendar, etc.
- Code View:** Shows the source code of `MasterPage.master` with the following structure:


```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title></title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
  </head>
  <body>
    <form id="form1" runat="server">
      <div>
        <h1><asp:Image ID="Image1" runat="server" Height="129px" ImageUrl="~/IMG_20240517_145717.jpg" Width="129px" BorderColor="#CC0000" BorderStyle=
        <asp:Label ID="Label1" runat="server" style="z-index: 1; left: 154px; top: 66px; position: absolute" Text="ASP.NET LEARN WITH ANKIT RAMI"
        </h1><asp:Menu ID="Menu1" runat="server" Font-Size="X-Large" ForeColor="white" Orientation="Horizontal">
          <Items>
            <asp:MenuItem NavigateUrl="https://ksv.ac.in" Text="KSV University" Value="KSV University"></asp:MenuItem>
            <asp:MenuItem NavigateUrl="https://npccsm.in" Text="NPCCSM BCA" Value="NPCCSM BCA"></asp:MenuItem>
            <asp:MenuItem NavigateUrl="https://bpccs.org" Text="BPCCS BCA" Value="BPCCS BCA"></asp:MenuItem>
          </Items>
          <StaticHoverStyle BackColor="#669900" BorderColor="#FF9966" BorderStyle="Dotted" BorderWidth="5px" ForeColor="white" />
          <StaticStyle BackColor="#6699FF" BorderColor="#FF3300" BorderStyle="Solid" BorderWidth="5px" Height="20px" HorizontalPadding="15px" Ve
        </asp:Menu>
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
        </asp:ContentPlaceHolder>
      </div>
    </form>
  </body>
</html>
```
- Solution Explorer:** Shows the project structure with `MasterPage.master` selected. A context menu is open with the following options:
 - Open
 - Open With...
 - Scope to This
 - New Solution Explorer View
 - View Code
 - View Designer
 - View Markup
 - View Component Designer
 - View in Page Inspector (Ctrl+K, Ctrl+G)
 - Add Content Page** (highlighted)
 - Build Page
 - Check Accessibility...
 - Exclude From Project
 - Cut (Ctrl+X)
 - Copy (Ctrl+C)
 - Delete (Del)
 - Rename
- Design View:** Shows a visual representation of the master page. It features a profile picture of Ankit Rami, the text "ASP.NET LEARN WITH ANKIT RAMI", and a navigation menu with links for "KSV University", "NPCCSM BCA", and "BPCCS BCA". Below the menu is a content area with a placeholder labeled "ContentPlaceHolder1".

Master Page in ASP.NET

The screenshot displays the Visual Studio IDE with the following components:

- Code Editor:** Shows the source code for `MasterPage.master` and `Default.aspx`. The master page contains two content placeholders: `Content1` (with ID `head`) and `Content2` (with ID `ContentPlaceholder1`). The content page (`Default.aspx`) includes a header image and the text "ASP.NET LEARN WITH ANKIT RAMI" within the `Content1` placeholder, and a navigation menu with "KSV University NPCCSM BCA BPPCS BCA" and "Home Page" within the `Content2` placeholder.
- Solution Explorer:** Shows the project structure for `ASP_MASTERPAGE`, including `Default.aspx`, `IMG_20240517_145717.jpg`, `MasterPage.master`, and `Web.config`. A red arrow points to `MasterPage.master`.
- Properties Window:** Shows the properties for the selected `<P>` element, including `accesskey`, `aria-atomic`, `aria-autocomplete`, etc.
- Design View:** Shows a visual representation of the master page layout with the content page applied.

Create Home Page Using Master Page

Master Page in ASP.NET

Create Contact Page Using Master Page

```

<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true" CodeFile="Contact.aspx.cs" Inherits="Default2" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
<p>
Mobile No - 8460467193</p>
<p>
Email - arramijob@gmail.com</p>
</asp:Content>

```

ASP.NET LEARN WITH ANKIT RAMI

KSV University NPCCSM BCA BPCCS BCA

Mobile No - 8460467193

Email - arramijob@gmail.com


📢 Master Page in ASP.NET

OUTPUT

localhost:3251/Default.aspx

localhost:3251/Default.aspx

Gmail YouTube Maps All Bookmarks

 **ASP.NET LEARN WITH ANKIT RAMI**

KSV University NPCCSM BCA BPCCS BCA

Home Page

Content Placeholder Design

URL with Redirect

https://npccsm.in

Master Page Design

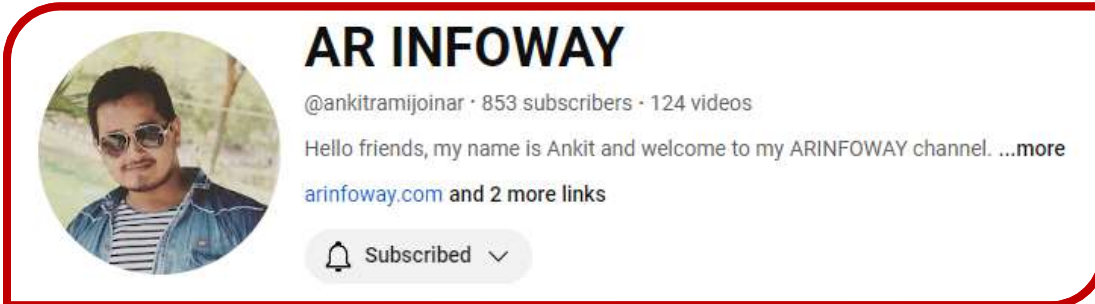
📢 Master Page in ASP.NET

OUTPUT

The screenshot shows a web browser window with the URL `localhost:3251/contact.aspx`. The page content includes a profile picture of a man, the text **ASP.NET LEARN WITH ANKIT RAMI**, a blue banner with **KSV University NPCCSM BCA** and a green banner with **BPCCS BCA**. Below the banners, the text **Mobile No - 8460467193** and **Email - arramijob@gmail.com** is displayed. A red box labeled **ContentPlaceholder Design** points to the email text. A red box labeled **Master Page Design** encompasses the profile picture and the main title. A red box labeled **URL with Redirect** points to the browser's address bar, which shows `https://bpccs.org`.

Reference Link for E-Learning

- ❖ <https://www.javatpoint.com/asp-net-tutorial>
- ❖ <https://www.javatpoint.com/c-sharp-tutorial>
- ❖ https://www.geeksforgeeks.org/csharp-programming-language/?ref=header_search
- ❖ <https://www.ankitweblogic.com/asp/index.php>
- ❖ <https://www.tutorialspoint.com/asp.net/>
- ❖ <https://amit.arinfoway.com>
- ❖ <https://www.youtube.com/channel/UCWbJh2iQ8w-8nrU0Xpjpw7g>



AR INFOWAY
@ankitramijoinar · 853 subscribers · 124 videos
Hello friends, my name is Ankit and welcome to my ARINFOWAY channel. ...more
[arinfoway.com](https://amit.arinfoway.com) and 2 more links
Subscribed



**More Information Join
YouTube Channel**

