UNIT-I

1. Explain about .Net Framework? Write an example program to display a table dynamically

ANS:
The .NET Framework is really a cluster of several technologies:

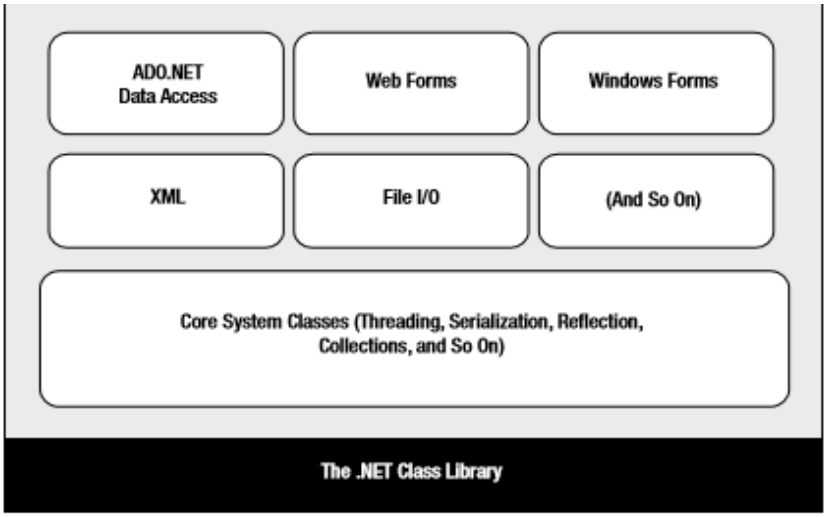
(a)The .NET languages: These include Visual Basic, C#, F#, and C++.

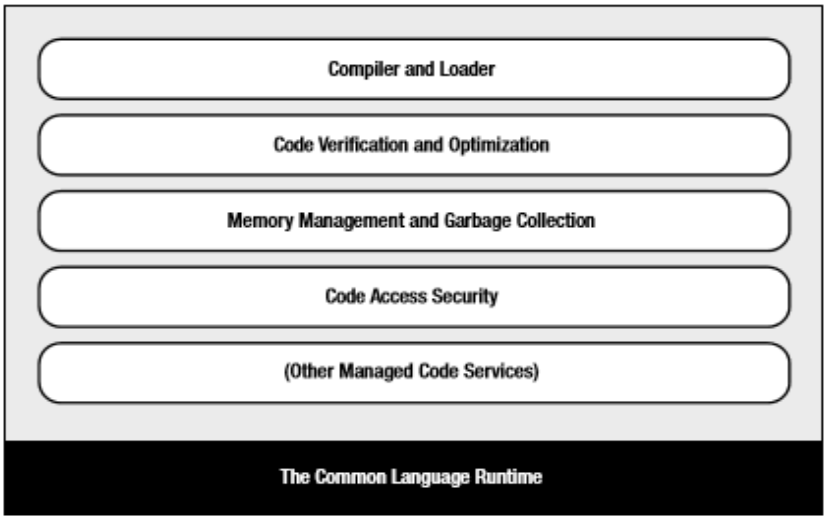
(b)The Common Language Runtime (CLR): This is the engine that executes all .NET programs and provides automatic services for these applications, such as security checking, memory management, and optimization.

(c)The .NET Framework class library: The class library collects thousands of pieces of prebuilt functionality that you can "snap in" to your applications. These features are sometimes organized into technology sets, such as ADO.NET (the technology for creating database applications) and Windows Presentation Foundation (WPF, the technology for creating desktop user interfaces).

(d)ASP.NET: This is the engine that hosts the web applications you create with .NET, and supports almost any feature from the .NET Framework class library. ASP.NET also includes a set of web specific services, such as secure authentication and data storage.

(e)Visual Studio: This optional development tool contains a rich set of productivity and debugging features. Visual Studio includes the complete .NET Framework, so you won't need to download it separately.



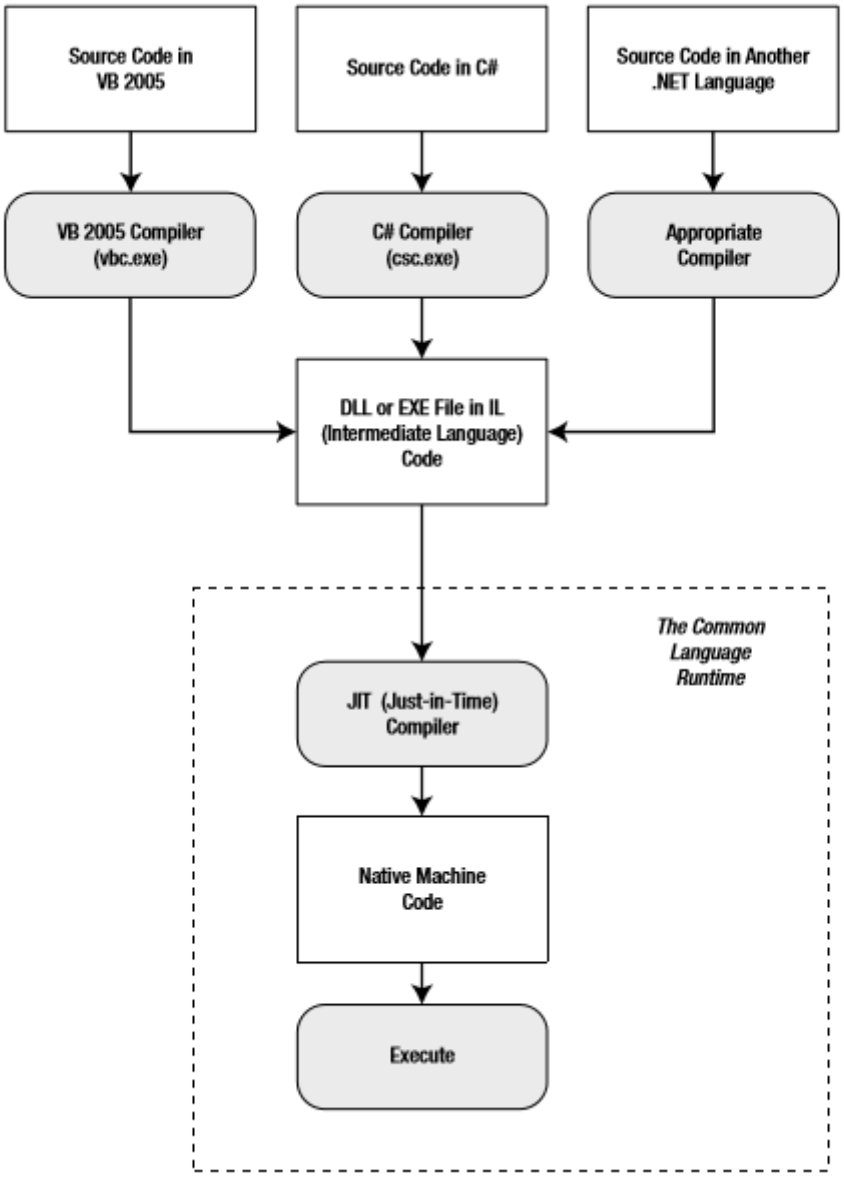


Intermediate Language

All the .NET languages are compiled into another lower-level language before the code is executed. This lower level language is the Common Intermediate Language (CIL, or just IL). The CLR, the engine of .NET, uses only IL code. Because all .NET languages are based on IL, they all have profound similarities. This is the reason that the VB and C# languages provide essentially the same features and performance. In fact, the languages are so compatible that a web page written with C# can use a VB component in the same way it uses a C# component, and vice versa.

The .NET Framework formalizes this compatibility with something called the Common Language Specification (CLS). Essentially, the CLS is a contract that, if respected, guarantees that a component written in one .NET language can be used in all the others. One part of the CLS is the common type system (CTS), which defines the rules for data types such as strings, numbers, and arrays that are shared in all .NET languages. The CLS also defines object-oriented ingredients such as classes, methods, events, and quite a bit more. For the most part, .NET developers don't need to think about how the CLS works, even though they rely on it every day. Figure 1-5 shows how the .NET languages are compiled to

IL. Every EXE or DLL file that you build with a .NET language contains IL code. This is the file you deploy to other computers. In the case of a web application, you deploy your compiled code to a live web server.



The CLR runs only IL code, which means it has no idea which .NET language you originally used. Notice, however, that the CLR performs another compilation step—it takes the IL code and transforms it to native machine language code that's appropriate for the current platform. This step occurs when the application is launched, just before the code is executed. In an ASP.NET application, these machine-specific files are cached while the web application is running so they can be reused, ensuring optimum performance.

Visual Studio:

Visual Studio is available in several editions:

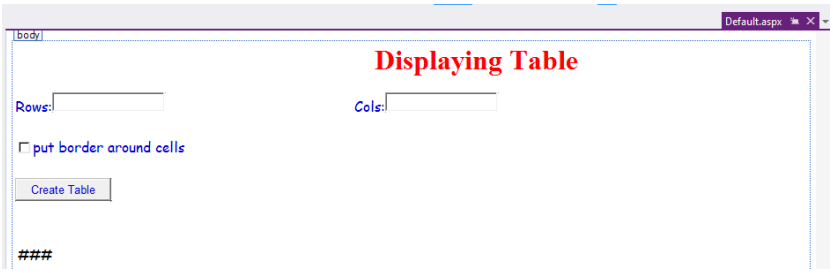
Visual Web Developer Express: This is a completely free version of Visual Studio that's surprising capable; it even includes SQL Server Express, a free database engine that you can use to power small websites. Its main limitation is that it allows you to build web applications and components only, not other types of .NET programs (like Windows applications).

Visual Studio Professional: This is the leanest full version of Visual Studio. It has all the features you need to build any type of .NET application (Windows or web).

Visual Studio Premium, Ultimate, or Test Professional: These versions increase the cost and pile on more tools and frills (which aren't discussed in this book). For example, they incorporate features for automated testing and version control, which helps team members coordinate their work on large projects.

Table Creation Example:

Design View:



Source View:

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

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <h1 style="font-family: 'Lao UI'; color: #FF0000 Displaying Table</h1>
    <form id="form1" runat="server">
    <div style="font-family: 'Comic Sans MS'; background-color: #FFFFFF">

        <asp:Label ID="Label1" runat="server" ForeColor="#0000CC"
Text="Rows:"></asp:Label>
        <asp:TextBox ID="TextBox1" runat="server" ForeColor="#0000CC"></asp:TextBox>
        <asp:Label ID="Label2" runat="server" ForeColor="#0000CC"
Text="Cols:"></asp:Label>
        <asp:TextBox ID="TextBox2" runat="server" ForeColor="#0000CC"></asp:TextBox>
        <br />
        <br />
        <asp:CheckBox ID="CheckBox1" runat="server" ForeColor="#0000CC" Text="put
border around cells" />
        <br />
        <br />
```

```
        <asp:Button ID="Button1" runat="server" ForeColor="#0000CC"
OnClick="Button1_Click" Text="Create Table" />
        <br />
        <br />
        <br />
        <asp:Table ID="Table1" runat="server">
        </asp:Table>

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

Default.aspx.cs:

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

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        TextBox1.Focus();
        TextBox2.Focus();
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        Table1.BorderStyle = BorderStyle.Dotted;
        Table1.BorderWidth = 10;
        int nr = int.Parse(TextBox1.Text);
        int nc = int.Parse(TextBox2.Text);
        for(int i=0;i<nr;i++){
            TableRow row=new TableRow();
            for(int j=0;j<nc;j++){
                TableCell cell =new TableCell();
                if(CheckBox1.Checked){
                    cell.BorderStyle=BorderStyle.Solid;
                    cell.BorderWidth=3;
                }

                cell.Text = "prakash";
                row.Controls.Add(cell);
            }
            Table1.Controls.Add(row);
        }
    }
}
```

2. Explain about ASP.Net web control classes using Registration form example.

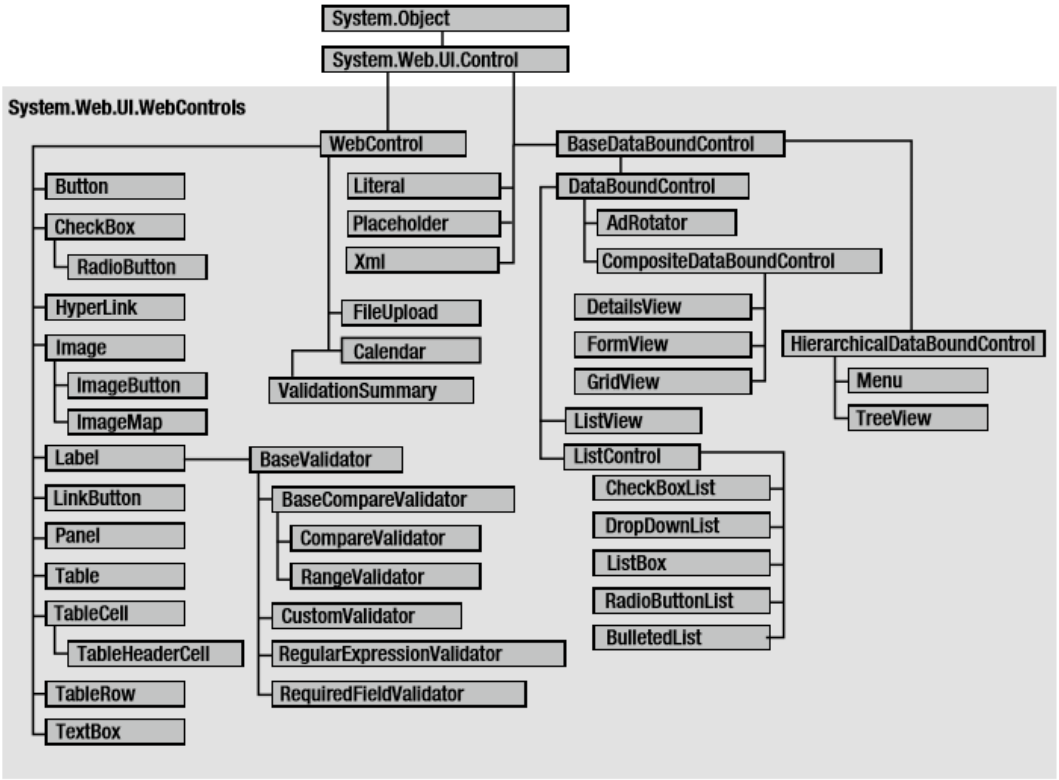


Figure 6-2. *The web control hierarchy*

The WebControl Base Class

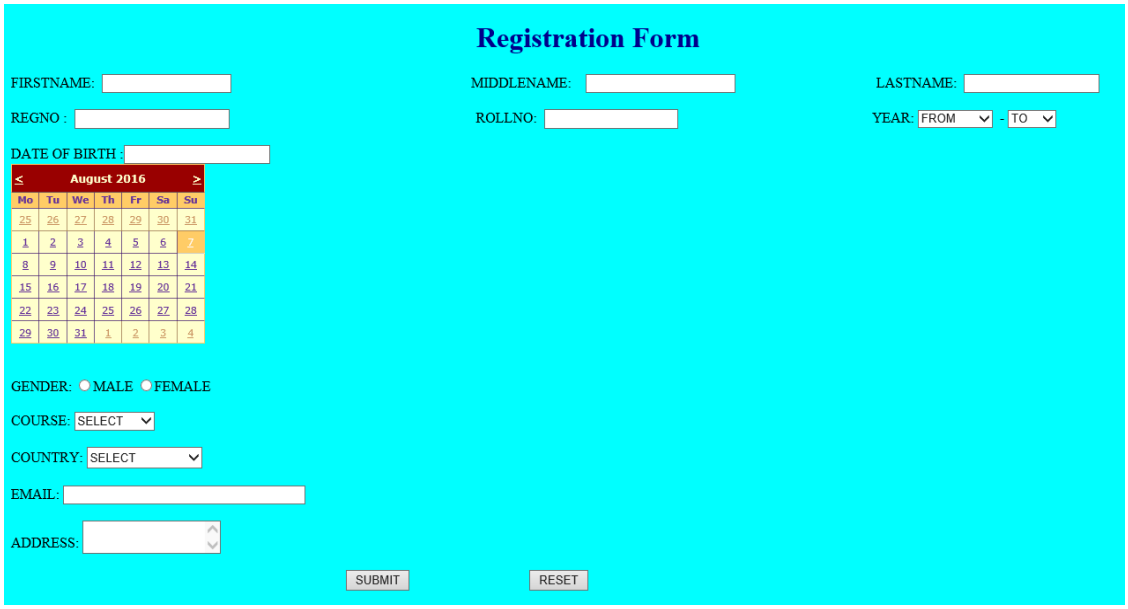
Most web controls begin by inheriting from the WebControl base class. This class defines the essential functionality for tasks such as data binding and includes some basic properties that you can use with almost any web control, as described in Table 6-2.

Table 6-2. *WebControl Properties*

Property	Description
AccessKey	Specifies the keyboard shortcut as one letter. For example, if you set this to Y, the Alt+Y keyboard combination will automatically change focus to this web control (assuming the browser supports this feature).
BackColor, ForeColor, and BorderColor	Sets the colors used for the background, foreground, and border of the control. In most controls, the foreground color sets the text color.
BorderWidth	Specifies the size of the control border.
BorderStyle	One of the values from the BorderStyle enumeration, including Dashed, Dotted, Double, Groove, Ridge, Inset, Outset, Solid, and None.
Controls	Provides a collection of all the controls contained inside the current control. Each object is provided as a generic System.Web.UI.Control object, so you will need to cast the reference to access control-specific properties.
Enabled	When set to false, the control will be visible, but it will not be able to receive user input or focus.
EnableViewState	Set this to false to disable the automatic state management for this control. In this case, the control will be reset to the properties and formatting specified in the control tag (in the .aspx page) every time the page is posted back. If this is set to true (the default), the control uses the hidden input field to store information about its properties, ensuring that any changes you make in code are remembered.
Font	Specifies the font used to render any text in the control as a System.Web. UI.WebControls.FontInfo object.
Height and Width	Specifies the width and height of the control. For some controls, these properties will be ignored when used with older browsers.
ID	Specifies the name that you use to interact with the control in your code (and also serves as the basis for the ID that's used to name the top-level element in the rendered HTML).
Page	Provides a reference to the web page that contains this control as a System.Web. UI.Page object.
Parent	Provides a reference to the control that contains this control. If the control is placed directly on the page (rather than inside another control), it will return a reference to the page object.
TabIndex	A number that allows you to control the tab order. The control with a TabIndex of 0 has the focus when the page first loads. Pressing Tab moves the user to the control with the next lowest TabIndex, provided it is enabled. This property is supported only in Internet Explorer.
ToolTip	Displays a text message when the user hovers the mouse above the control. Many older browsers don't support this property.
Visible	When set to false, the control will be hidden and will not be rendered to the final HTML page that is sent to the client.

Registration form:

Design View:



Source View:

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

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="background-color:aqua;" >
      <h1 style="color:darkblue;background-color:aqua;"> Registration Form</h1>
    <form id="form1" runat="server" >
    <div >

        <asp:Label ID="Label1" runat="server" Text="FIRSTNAME:"></asp:Label>
         <asp:TextBox ID="TextBox1" runat="server" Width="143px"></asp:TextBox>
                <asp:Label ID="Label2" runat="server" Text="MIDDLENAME:"></asp:Label>
   <asp:TextBox ID="TextBox2" runat="server"
Width="166px"></asp:TextBox>
        <asp:Label ID="Label13" runat="server" Text="LASTNAME:"></asp:Label>
         <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
                <asp:Label ID="Label4" runat="server" Text="REGNO    :"></asp:Label>
 <asp:TextBox ID="TextBox4" runat="server" Width="172px"></asp:TextBox>
        <asp:Label ID="Label14" runat="server" Text="ROLLNO:"></asp:Label>
 <asp:TextBox ID="TextBox5" runat="server" Width="148px"></asp:TextBox>
                <asp:Label ID="Label6" runat="server" Text="YEAR:"></asp:Label>

        <asp:DropDownList ID="DropDownList3" runat="server" Width="85px">
            <asp:ListItem>FROM</asp:ListItem>
            <asp:ListItem>2010</asp:ListItem>
            <asp:ListItem>2011</asp:ListItem>
            <asp:ListItem>2012</asp:ListItem>
            <asp:ListItem>2013</asp:ListItem>
            <asp:ListItem>2014</asp:ListItem>
            <asp:ListItem>2015</asp:ListItem>
            <asp:ListItem>2016</asp:ListItem>
            <asp:ListItem>2017</asp:ListItem>
            <asp:ListItem>2018</asp:ListItem>
        </asp:DropDownList>
```

```
 -
        <asp:DropDownList ID="DropDownList4" runat="server">
            <asp:ListItem>TO</asp:ListItem>
            <asp:ListItem>2014</asp:ListItem>
            <asp:ListItem>2015</asp:ListItem>
            <asp:ListItem>2016</asp:ListItem>
            <asp:ListItem>2017</asp:ListItem>
            <asp:ListItem>2018</asp:ListItem>
            <asp:ListItem>2019</asp:ListItem>
            <asp:ListItem>2020</asp:ListItem>
            <asp:ListItem>2021</asp:ListItem>
        </asp:DropDownList>

        <br />
        <br />

    </div>

        <asp:Label ID="Label7" runat="server" Text="DATE OF BIRTH"></asp:Label>
        :<asp:TextBox ID="TextBox7" runat="server" Width="162px" ></asp:TextBox>
        <asp:Calendar ID="Calendar1" runat="server" BackColor="#FFFFCC"
BorderColor="#FFCC66" BorderWidth="1px" DayNameFormat="Shortest" Font-Names="Verdana"
Font-Size="8pt" ForeColor="#663399" Height="200px"
OnSelectionChanged="Calendar1_SelectionChanged" Width="220px" ShowGridLines="True">
            <DayHeaderStyle BackColor="#FFCC66" Height="1px" Font-Bold="True" />
            <NextPrevStyle Font-Size="9pt" ForeColor="#FFFFCC" />
            <OtherMonthDayStyle ForeColor="#CC9966" />
            <SelectedDayStyle BackColor="#CCCCFF" Font-Bold="True" />
            <SelectorStyle BackColor="#FFCC66" />
            <TitleStyle BackColor="#990000" Font-Bold="True" Font-Size="9pt"
ForeColor="#FFFFCC" />
            <TodayDayStyle BackColor="#FFCC66" ForeColor="White" />
        </asp:Calendar>
        <br />
        <br />
        <asp:Label ID="Label8" runat="server" Text="GENDER:"></asp:Label>
        <asp:RadioButton ID="RadioButton1" runat="server" Text="MALE" />
        <asp:RadioButton ID="RadioButton2" runat="server" Text="FEMALE" />
        <br />
        <br />
        <asp:Label ID="Label9" runat="server" Text="COURSE:"></asp:Label>
        <asp:DropDownList ID="DropDownList1" runat="server" Height="21px"
Width="91px">
            <asp:ListItem>SELECT</asp:ListItem>
            <asp:ListItem>CSE</asp:ListItem>
            <asp:ListItem>ECE</asp:ListItem>
            <asp:ListItem>EEE</asp:ListItem>
            <asp:ListItem>IT</asp:ListItem>
            <asp:ListItem>CIVIL</asp:ListItem>
            <asp:ListItem>MECH</asp:ListItem>
        </asp:DropDownList>
        <br />
        <br />
        <asp:Label ID="Label10" runat="server" Text="COUNTRY:"></asp:Label>
        <asp:DropDownList ID="DropDownList2" runat="server" Height="24px"
Width="131px">
            <asp:ListItem>SELECT</asp:ListItem>
            <asp:ListItem>INDIA</asp:ListItem>
            <asp:ListItem>AMERICA</asp:ListItem>
            <asp:ListItem>LONDON</asp:ListItem>
            <asp:ListItem>SINGAPUR</asp:ListItem>
        </asp:DropDownList>
```

```
        <br />
        <br />
        <asp:Label ID="Label11" runat="server" Text="EMAIL:"></asp:Label>
        <asp:TextBox ID="TextBox8" runat="server" Width="271px"></asp:TextBox>
        <br />
        <br />
        <asp:Label ID="Label12" runat="server" Text="ADDRESS:"></asp:Label>
        <asp:TextBox ID="TextBox9" runat="server" Width="151px"
TextMode="MultiLine"></asp:TextBox>
        <br />
        <br />
        <asp:Button ID="Button1" runat="server" Text="SUBMIT"
OnClick="Button1_Click"/>
        <asp:Button ID="Button2" runat="server" Text="RESET" />
        </form>
    <p id="para" runat="server" style="color:red"></p>
     </body>
</html>
```

Default.aspx.cs:

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

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

    }
    protected void Calendar1_SelectionChanged(object sender, EventArgs e)
    {
        DateTime d = Calendar1.SelectedDate;
        string a = d.ToString();
        TextBox7.Text = a;
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        string a = TextBox1.Text;
        string b = TextBox2.Text;
        string c = TextBox3.Text;
        string d = TextBox4.Text;
        string f = TextBox5.Text;
        string g = DropDownList3.Text;
        string h = TextBox7.Text;
        string i = TextBox8.Text;
        string j = TextBox9.Text;
        string k = DropDownList1.Text;
        string l = DropDownList2.Text;
        string m = DropDownList4.Text;
        if (RadioButton1.Checked == true)
        {
            Response.Write("u selected:" + RadioButton1.Text + "<br>");
        }
        else
        {
            Response.Write("u selected:" + RadioButton2.Text + "<br>");
        }
```

```
        para.InnerHtml = a + "<br>" + b + "<br>" + c + "<br>" + d + "<br>" + e +
"<br>" + f + "<br>" + g + "-" + m + "<br>" + h + "<br>" + k + "<br>" + l + "<br>" + i
+ "<br>" + j + "<br>";
    }
}
```

3. Explain about Html server control classes using currency converter example.

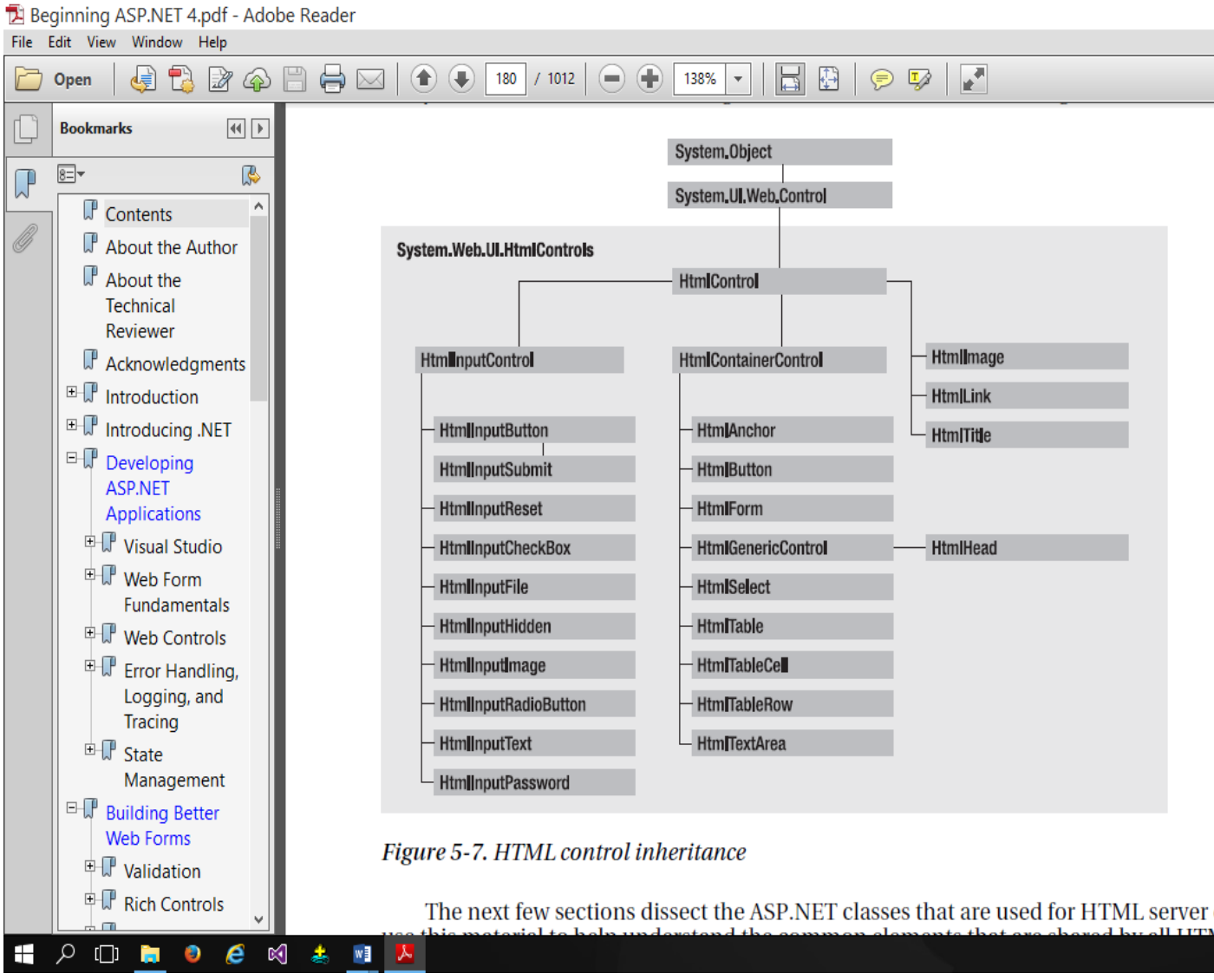


Figure 5-7. HTML control inheritance

The next few sections dissect the ASP.NET classes that are used for HTML server

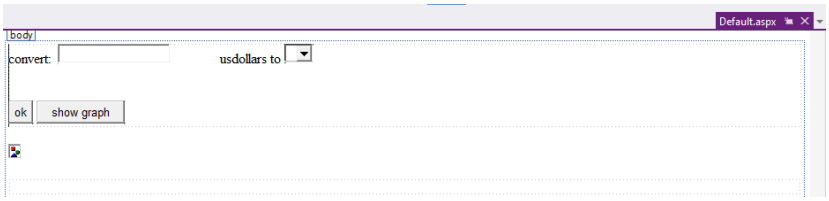
Table 5-3. *The HTML Server Control Classes*

Class Name	HTML Element	Description
HtmlForm	<form>	Wraps all the controls on a web page. All ASP.NET server controls must be placed inside an HtmlForm control so that they can send their data to the server when the page is submitted. Visual Studio adds the <form> element to all new pages. When designing a web page, you need to ensure that every other control you add is placed inside the <form> section.
HtmlAnchor	<a>	A hyperlink that the user clicks to jump to another page.
HtmlImage	<img>	A link that points to an image, which will be inserted into the web page at the current location.
HtmlTable, HtmlTableRow, and HtmlTableCell	<table>, <tr>, <th>, and <td>	A table that displays multiple rows and columns of static text.
HtmlInputButton, HtmlInputSubmit, and HtmlInputReset	<input type="button">, <input type="submit">, and <input type="reset">	A button that the user clicks to perform an action (HtmlInputButton), submit the page (HtmlInputSubmit), or clear all the user-supplied values in all the controls (HtmlInputReset).
HtmlButton	<button>	A button that the user clicks to perform an action. This is not supported by all browsers, so HtmlInputButton is usually used instead. The key difference is that the HtmlButton is a container element. As a result, you can insert just about anything inside it, including text and pictures. The HtmlInputButton, on the other hand, is strictly text-only.
HtmlInputCheckBox	<input type="checkbox">	A check box that the user can select or clear. Doesn't include any text of its own.
HtmlInputRadioButton	<input type="radio">	A radio button that can be selected in a group. Doesn't include any text of its own.
HtmlInputText and HtmlInputPassword	<input type="text"> and <input type="password">	A single-line text box enabling the user to enter information. Can also be displayed as a password field (which displays bullets instead of characters to hide the user input).
HtmlTextArea	<textarea>	A large text box for typing multiple lines of text.
HtmlInputImage	<input type="image">	Similar to the <img> tag, but inserts a "clickable" image that submits the page. Using server-side code, you can determine exactly where the user clicked in the image—a technique you'll consider later in this chapter.
HtmlInputFile	<input type="file">	A Browse button and text box that can be used to upload a file to your web server, as described in

Class Name	HTML Element	Description
HtmlInputHidden	<input type="hidden">	Contains text information that will be sent to the server when the page is posted back but won't be visible in the browser.
HtmlSelect	<select>	A drop-down or regular list box enabling the user to select an item.
HtmlHead and HtmlTitle	<head> and <title>	Represents the header information for the page, which includes information about the page that isn't actually displayed in the page, such as search keywords and the web page title. These are the only HTML server controls that aren't placed in the <form> section.
HtmlGenericControl	Any other HTML element.	This control can represent a variety of HTML elements that don't have dedicated control classes. For example, if you add the `runat="server"` attribute to a element, it's provided to your code as an HtmlGenericControl object. You can identify the type of element by reading the TagName property, which stores a string (for example, "div").

Currency Convertor:

Design View:



Source View:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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">
    <div>
     convert:
 <input type="text" id="us"
runat="server"/>           &nbs
p;  usdollars to         <select id="currency" runat="server"></select>
        <br />
        <br />
        <br />
    <input type="submit" value="ok" id="convert" runat="server"
onserverclick="convert_serveclick"/>
    <input type="submit" value="show graph" id="show" runat="server"
onserverclick="showgraph_serverclick"/>
    </div>

    </form>
    <img src="" id="graph" runat="server"/>
    <p id="result" runat="server"></p>
</body>
</html>
```

Default.aspx.cs:

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

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            currency.Items.Add(new ListItem("EURO","55"));
```

```
            currency.Items.Add(new ListItem("japanese yen"," 75"));
            currency.Items.Add(new ListItem("china"," 95"));
            graph.Visible = false;
        }
    }
    protected void showgraph_serverclick(object sender, EventArgs e)
    {
        graph.Src = "pic" + currency.SelectedIndex.ToString() + ".jpg";
        graph.Visible = true;
    }
    protected void convert_serveclick(object sender, EventArgs e)
    {
        Decimal oldamount;
        bool sucess = Decimal.TryParse(us.Value, out oldamount);
        if (oldamount<=0||sucess==false){
            result.Style["Color"]="Red";
            result.InnerText="specify correct formate";
        }
        else{
            result.Style["Color"]="Green";
        }

            ListItem li = currency.Items[currency.SelectedIndex];
            Decimal newamount = oldamount * Decimal.Parse(li.Value);
            result.InnerText += oldamount.ToString() + "us dollars=";
            result.InnerText += newamount.ToString() + li.Text;

    }
}
```

4. Explain List control classes available in ASP.Net with an example program.

The list controls include the ListBox, DropDownList, CheckBoxList, RadioButtonList, and BulletedList. They all work in essentially the same way but are rendered differently in the browser. The ListBox, for example, is a rectangular list that displays several entries, while the DropDownList shows only the selected item. The CheckBoxList and RadioButtonList are similar to the ListBox, but every item is rendered as a check box or option button, respectively. Finally, the BulletedList is the odd one out—it's the only list control that isn't selectable. Instead, it renders itself as a sequence of numbered or bulleted items.

Single-select list Controls:

Most of the list controls allow single selection. The list controls that come under this are RadioButtonList,DropDownList .By using the Text and the selectedvalue properties we are going to retrieve the information from the control what we have been selected.

Multiple-Select List Controls

Some list controls can allow multiple selections. This isn't allowed for the DropDownList or RadioButtonList, but it is supported for a ListBox, provided you have set the SelectionMode property to the enumerated value List SelectionMode to Multiple. The user can then select multiple items by holding down the Ctrl key while clicking the items in the list. With the CheckBoxList, multiple selections are always possible.

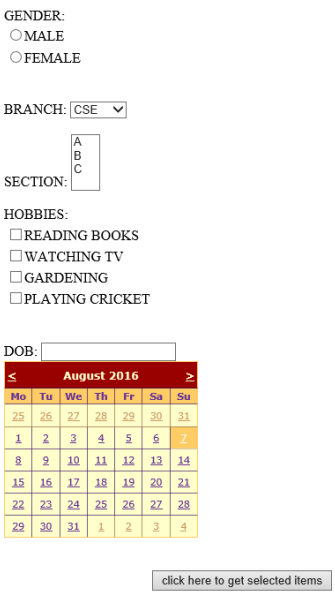
The SelectedIndex and SelectedItem properties aren't much help with a list that supports multiple selection, because they will simply return the first selected item. Instead, you can find all the selected items by iterating through the Items collection of the list control and checking the ListItem.Selected property of each item. (If it's true, that item is one of the selected items.) Figure 6-5 shows a simple web page example that provides a list of computer languages. After the user clicks the OK button, the page indicates which selections the user made.

The BulletedList Control

The BulletedList control is a server-side equivalent of the <ul> (unordered list) and <ol> (ordered list) elements. As with all list controls, you set the collection of items that should be displayed through the Items property.

Example:

Design View:



Source View:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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">
    <div>

        <asp:Label ID="Label1" runat="server" Text="GENDER:"></asp:Label>
        <asp:RadioButtonList ID="RadioButtonList1" runat="server">
```

```
            <asp:ListItem>MALE</asp:ListItem>
            <asp:ListItem>FEMALE</asp:ListItem>
        </asp:RadioButtonList>
        <br />
        <br />
        <asp:Label ID="Label2" runat="server" Text="BRANCH:"></asp:Label>
        <asp:DropDownList ID="DropDownList1" runat="server">
            <asp:ListItem>CSE</asp:ListItem>
            <asp:ListItem>ECE</asp:ListItem>
            <asp:ListItem>EEE</asp:ListItem>
            <asp:ListItem>MECH</asp:ListItem>
        </asp:DropDownList>
        <br />
        <br />
        <asp:Label ID="Label3" runat="server" Text="SECTION:"></asp:Label>
        <asp:ListBox ID="ListBox1" runat="server" SelectionMode="Multiple">
            <asp:ListItem>A</asp:ListItem>
            <asp:ListItem>B</asp:ListItem>
            <asp:ListItem>C</asp:ListItem>
        </asp:ListBox>
        <br />
        <br />
        <asp:Label ID="Label4" runat="server" Text="HOBBIES:"></asp:Label>
        <asp:CheckBoxList ID="CheckBoxList1" runat="server">
            <asp:ListItem>READING BOOKS</asp:ListItem>
            <asp:ListItem>WATCHING TV</asp:ListItem>
            <asp:ListItem>GARDENING</asp:ListItem>
            <asp:ListItem>PLAYING CRICKET</asp:ListItem>
        </asp:CheckBoxList>
        <br />
        <br />
        <asp:Label ID="Label5" runat="server" Text="DOB:"></asp:Label>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        <asp:Calendar ID="Calendar1" runat="server" BackColor="#FFFFCC"
BorderColor="#FFCC66" BorderWidth="1px" DayNameFormat="Shortest" Font-Names="Verdana"
Font-Size="8pt" ForeColor="#663399" Height="200px" ShowGridLines="True" Width="220px"
OnSelectionChanged="Calendar1_SelectionChanged">
            <DayHeaderStyle BackColor="#FFCC66" Font-Bold="True" Height="1px" />
            <NextPrevStyle Font-Size="9pt" ForeColor="#FFFFCC" />
            <OtherMonthDayStyle ForeColor="#CC9966" />
            <SelectedDayStyle BackColor="#CCCCFF" Font-Bold="True" />
            <SelectorStyle BackColor="#FFCC66" />
            <TitleStyle BackColor="#990000" Font-Bold="True" Font-Size="9pt"
ForeColor="#FFFFCC" />
            <TodayDayStyle BackColor="#FFCC66" ForeColor="White" />
        </asp:Calendar>
        <br />
        <asp:BulletedList ID="BulletedList1" runat="server">
        </asp:BulletedList>
        <br />

        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="click
here to get selected items" />
        <br />

    </div>
    </form>
    <P id="para" runat="server"></p>
</body>
</html>
```

Default.aspx.cs:

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

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

    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        string a = RadioButtonList1.Text;
        string b = DropDownList1.SelectedValue;
        string c = ListBox1.SelectedValue;
        string d = TextBox1.Text;
        BulletedList1.Items.Add(a);
        BulletedList1.Items.Add(b);
        BulletedList1.Items.Add(c);


        foreach (ListItem li in CheckBoxList1.Items)
        {
            if (li.Selected)
            {
                BulletedList1.Items.Add(li.Text);
                para.InnerHtml+=li.Text+"<br>";
            }
        }
        BulletedList1.Items.Add(d);
        para.InnerHtml += a + "<br>" + b + "<br>" + c + "<br>" + d;
    }
    protected void Calendar1_SelectionChanged(object sender, EventArgs e)
    {
        TextBox1.Text = Calendar1.SelectedDate.ToString();
    }
}
```

5. Explain about Tracing in ASP.Net with an example program.

Page Tracing:

Visual Studio's debugging tools and ASP.NET's detailed error pages are extremely helpful when you're testing an application. However, sometimes you need a way to identify problems after you've deployed an application, when you don't have Visual Studio to rely on.

Fortunately, there's an easier way to solve the problem without resorting to a home grown solution. ASP.NET provides a feature called *tracing* that gives you a far more convenient and flexible way to report diagnostic information.

Enabling Tracing:

To use tracing, you need to explicitly enable it. There are several ways to switch on tracing. One of the easiest ways is by adding an attribute to the Page directive in the .aspx file:
<%@ Page **Trace="true"** ... %>
You can also enable tracing using the built-in Trace object . Here's an example of how you might turn tracing on in the Page.Load event handler:

```
protected void Page_Load(Object sender, EventArgs e)
{
Trace.IsEnabled = true;
}
```

This technique is useful because it allows you to enable or disable tracing for a page under specific circumstances that you test for in your code.

Request Details

This section includes some basic information such as the current session ID, the time the web request was made, and the type of web request and encoding (see Figure 7-8). Most of these details are fairly uninteresting, and you won't spend much time looking at them. The exception is the session ID. By watching for when the session ID changes, you'll know when a new session is created. (Sessions are used to store information for a specific user in between page requests. You'll learn about them in Chapter 8.)

Figure 7-8. *Request details*

Trace Information

Trace information shows the stages of processing that the page went through before being sent to the client (see Figure 7-9). Each section has additional information about how long the processing took to complete, as a measure from the start of the first stage (From First) and as a measure from the start of the previous stage (From Last). If you add your own trace messages (a technique described shortly), they will also appear in this section.

Trace Information

Category	Message	From First(s)	From Last(s)
aspx.page	End Raise PostBackEvent		
aspx.page	Begin LoadComplete	4.83301648673225E-05	0.000048
aspx.page	End LoadComplete	8.85587414042846E-05	0.000040
aspx.page	Begin PreRender	0.000119847634266366	0.000031
aspx.page	End PreRender	0.000163987322411089	0.000044
aspx.page	Begin PreRenderComplete	0.00019695240596221	0.000033
aspx.page	End PreRenderComplete	0.000227682568594612	0.000031
aspx.page	Begin SaveState	0.000686400087161916	0.000459
aspx.page	End SaveState	0.000843403281702004	0.000157
aspx.page	Begin SaveStateComplete	0.000880279476860886	0.000037
aspx.page	End SaveStateComplete	0.000917155672019768	0.000037
aspx.page	Begin Render	0.00094760646953733	0.000030
aspx.page	End Render	0.00136246366507475	0.000415

Figure 7-9. *Trace information*

Writing Trace Information

In addition to the standard trace information, you'll often want to generate your own tracing messages. For example, you might want to output the value of a variable at various points in execution so you can compare it with an expected value. Similarly, you might want to output messages when the code reaches certain points in execution so you can verify that various procedures are being used (and are used in the order you expect). Once again, these are tasks you can also achieve using Visual Studio debugging, but tracing is an invaluable technique when you're working with a web application that's been deployed to a web server.

To write a custom trace message, you use the Write() method or the Warn() method of the built-in Trace object. These methods are equivalent. The only difference is that Warn() displays the message in red lettering, which makes it easier to distinguish from other messages in the list. Here's a code snippet that writes a trace message when the user clicks a button:

```
protected void cmdWrite_Click(Object sender, EventArgs e)
{
    Trace.Write("About to place an item in session state.");
    Session["Test"] = "Contents";
    Trace.Write("Placed item in session state.");
}
```

These messages appear in the Trace Information section of the page, along with the default messages that ASP.NET generates automatically (see Figure 7-16).

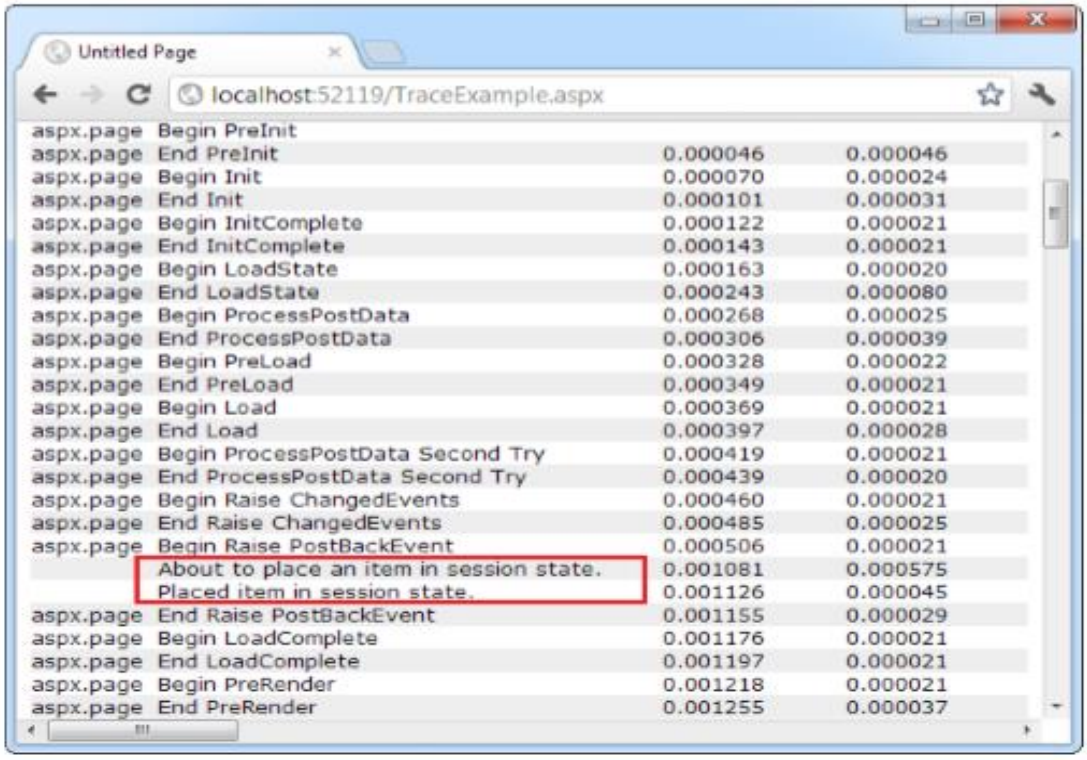
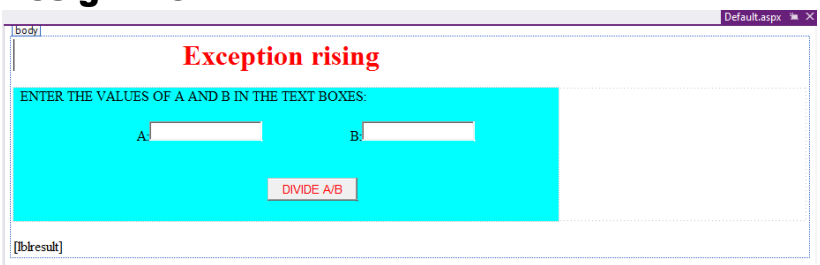


Figure 7-16. Custom trace messages

Tracing Example:

Design View:



Source View:

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

<!DOCTYPE html>

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

```
<head runat="server">
    <title></title>
</head>
<body >
    <h1 style="color:red;"> Exception rising </h1>
    <form id="form1" runat="server">

    <div style="background-color:aqua; width: 618px;">

          ENTER THE VALUES OF A AND B IN THE TEXT BOXES:<br />
        <br />
A:<asp:TextBox ID="TextBox1" runat="server" ForeColor="Blue"></asp:TextBox>
                B:<asp:TextBox ID="TextBox2" runat="server"
ForeColor="Blue"></asp:TextBox>
        <br />
        <br />
        <br />

        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="DIVIDE
A/B" ForeColor="Red" />
        <br />
        <br />

    </div>

    </form>
    <asp:Label ID="lblresult" runat="server" Text=""></asp:Label>
</body>
</html>
```

Default.aspx.cs:

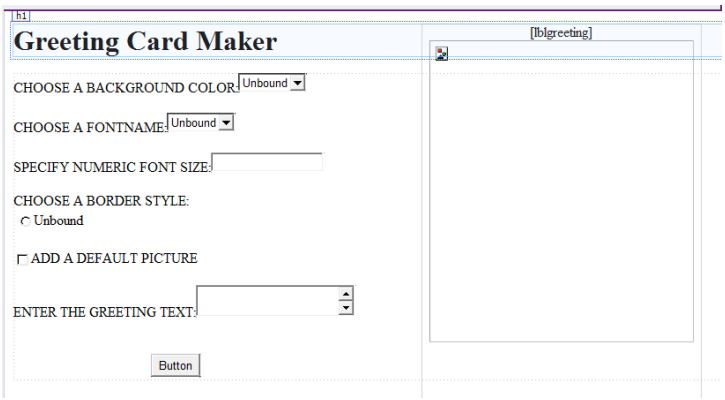
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Drawing;

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

    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        try
        {
            decimal a, b, result;
            a = decimal.Parse(TextBox1.Text);
            b = decimal.Parse(TextBox2.Text);
            result = a / b;
            lblresult.Text += "result=" + result.ToString();
            lblresult.ForeColor = Color.Green;
        }
        catch (DivideByZeroException er)
        {

            lblresult.Text += "message:" + er.Message + "<br>";
            lblresult.Text += "source:" + er.Source + "<br>";
            lblresult.Text += "trace:" + er.StackTrace + "<br>";
```

```
            lblresult.ForeColor = Color.Red;
            Trace.Warn("divide by zero exception");
        }
        catch (FormatException er)
        {
            lblresult.Text += "message:" + er.Message + "<br>";
            lblresult.Text += "source:" + er.Source + "<br>";
            lblresult.Text += "trace:" + er.StackTrace + "<br>";
            lblresult.ForeColor = Color.Red;
            Trace.Write("formate exception");
        }
        catch (OverflowException er)
        {
            lblresult.Text += "message:" + er.Message + "<br>";
            lblresult.Text += "source:" + er.Source + "<br>";
            lblresult.Text += "trace:" + er.StackTrace + "<br>";
            lblresult.Text += "numbers must be between " + Int32.MinValue.ToString() +
"to" + Int32.MaxValue.ToString();
            lblresult.ForeColor = Color.Red;
            Trace.Warn("overflow exception");
        }
        catch (Exception er) {
            lblresult.Text += "message:" + er.Message + "<br>";
            lblresult.Text += "source:" + er.Source + "<br>";
            lblresult.Text += "trace:" + er.StackTrace + "<br>";
        }
    }
}
```

6. Explain about try-catch- finally blocks with an example program. Write an example program to generate a Greeting card maker.

Ans:

Greeting card Example:



Source View:

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

<!DOCTYPE html>

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

```
    <h1>Greeting Card Maker</h1>
    <form id="form1" runat="server">
    <div>

        CHOOSE A BACKGROUND COLOR:<asp:DropDownList ID="DropDownList1" runat="server">
        </asp:DropDownList>
        <br />
        <br />
        CHOOSE A FONTNAME:<asp:DropDownList ID="DropDownList2" runat="server">
        </asp:DropDownList>
        <br />
        <br />
        SPECIFY NUMERIC FONT SIZE:<asp:TextBox ID="TextBox1"
runat="server"></asp:TextBox>
        <br />
        <br />
        CHOOSE A BORDER STYLE:<asp:RadioButtonList ID="RadioButtonList1"
runat="server">
        </asp:RadioButtonList>
        <br />
        <asp:CheckBox ID="CheckBox1" runat="server" Text="ADD A DEFAULT PICTURE" />
        <br />
        <br />
        ENTER THE GREETING TEXT:<asp:TextBox ID="TextBox2" runat="server"
TextMode="MultiLine"></asp:TextBox>
        <br />
        <br />
        <br />
        <asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click"
/>
     <asp:Panel runat="server" ID="panel1" style="position:absolute;top:14px;
left:473px; width: 318px; height: 498px;" HorizontalAlign="Center">
         <asp:Label ID="lblgreeting" runat="server" ></asp:Label>
         <br />
         <asp:Image ID="Image1" Visible="false" width="300px" runat="server" />
        </asp:Panel>
    </div>
    </form>
</body>
</html>
```

Default.aspx.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Drawing;
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        DropDownList1.Items.Add("Red");
        DropDownList1.Items.Add("Yellow");
        DropDownList1.Items.Add("Pink");
        DropDownList1.Items.Add("Blue");
        DropDownList2.Items.Add("TimesNewRoman");
        DropDownList2.Items.Add("Gabriola");
        DropDownList2.Items.Add("Arial");
```

```
        DropDownList2.Items.Add("Impact");
        ListItem li = new ListItem();
        li.Text = BorderStyle.None.ToString();
        li.Value = ((int)BorderStyle.None).ToString();
        RadioButtonList1.Items.Add(li);
        li = new ListItem();
        li.Text = BorderStyle.Dashed.ToString();
        li.Value = ((int)BorderStyle.Dashed).ToString();
        RadioButtonList1.Items.Add(li);
         li = new ListItem();
        li.Text = BorderStyle.Dotted.ToString();
        li.Value = ((int)BorderStyle.Dotted).ToString();
        RadioButtonList1.Items.Add(li);

        Image1.ImageUrl = "happy.jpg";
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        panel1.BackColor = Color.FromName(DropDownList1.SelectedItem.Text);
        lblgreeting.Font.Name = DropDownList2.SelectedItem.Text;
        if (int.Parse(TextBox1.Text) > 0)
        {
            lblgreeting.Font.Size = FontUnit.Point(int.Parse(TextBox1.Text));
        }
        if (RadioButtonList1.SelectedIndex == 0)
        {
            int value = int.Parse(RadioButtonList1.SelectedItem.Value);
            panel1.BorderStyle = (BorderStyle)value;
        }
        if(RadioButtonList1.SelectedIndex == 1)
        {
            int value = int.Parse(RadioButtonList1.SelectedItem.Value);
            panel1.BorderStyle = (BorderStyle)value;
        }
        if (RadioButtonList1.SelectedIndex == 2)
        {
            int value = int.Parse(RadioButtonList1.SelectedItem.Value);
            panel1.BorderStyle = (BorderStyle)value;
        }
        if (CheckBox1.Checked)
            Image1.Visible = true;
        else
            Image1.Visible = false;
        lblgreeting.Text = TextBox2.Text;
    }
}
```

UNIT-II

View State:

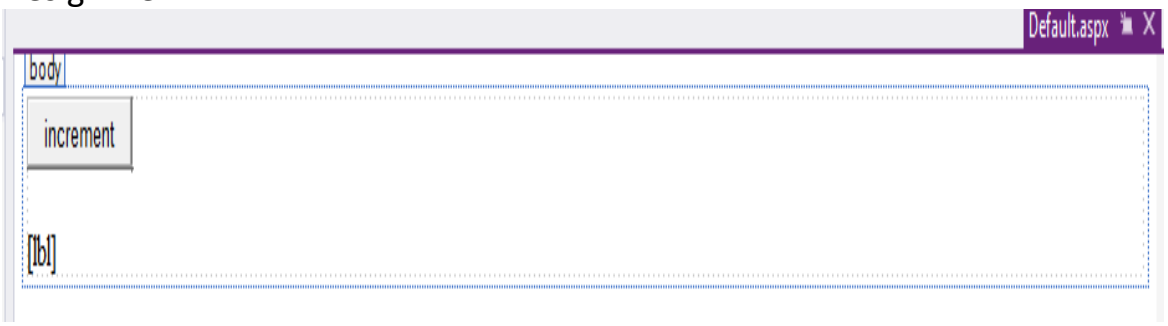
One of the most common ways to store information is in *view state*. View state uses a hidden field that ASP.NET automatically inserts in the final, rendered HTML of a web page. It's a perfect place to store information that's used for multiple postbacks in a single web page.

Web controls use view state to keep track of certain details. For example, if you change the text of a label, the Label control automatically stores its new text in view state. That way, the text remains in place the next time the page is posted back. Web controls store most of their property values in view state, provided you haven't switched View State.

View state isn't limited to web controls. Your web page code can add bits of information directly to the view state of the containing page and retrieve it later after the page is posted back. The type of information you can store includes simple data types and your own custom objects.

Example of the View state:

Design view:



Default.aspx page:

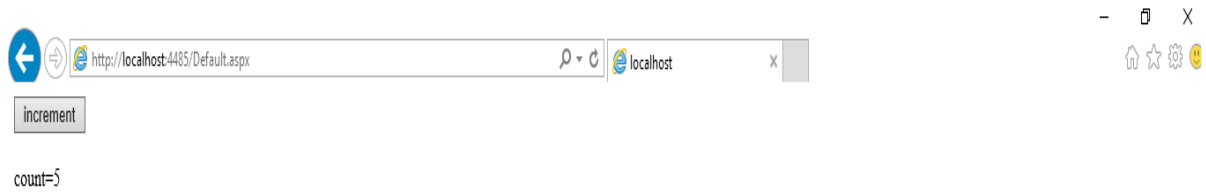
```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html>

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

        <asp:Button ID="Button1" runat="server" Text="increment"
OnClick="Button1_Click" />
        <br />
        <br />
        <asp:Label ID="lbl" runat="server"></asp:Label>

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

Default.aspx.cs page:

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

```
using System.Web.UI;
using System.Web.UI.WebControls;

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

    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        int c;
        if (ViewState["Counter"] == null)
            c = 1;
        else
            c = (int)(ViewState["Counter"]) + 1;
        ViewState["Counter"] = c;
        lbl.Text = "count=" + c.ToString();
    }
}
```

Output:



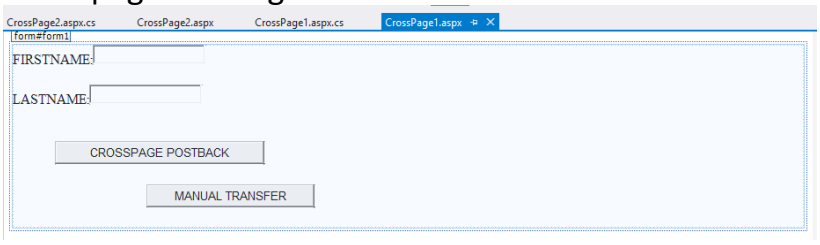
Cross-Page Posting

A cross-page postback is a technique that extends the postback mechanism you've already learned about so that one page can send the user to another page, complete with all the information for that page. This technique sounds conceptually straight forward, but it's a potential minefield. If you're not careful, it can lead you to create pages that are tightly coupled to others and difficult to enhance and debug.

The infrastructure that supports cross-page postbacks is a property named PostBackUrl, which is defined by the IButtonControl interface and turns up in button controls such as ImageButton,LinkButton, and Button. To use cross-posting, you simply set PostBackUrl to the name of another webform. When the user clicks the button, the page will be posted to that new URL with the values from all the input controls on the current page.

Example:

Crosspage1:Design view



Crosspage1.aspx:

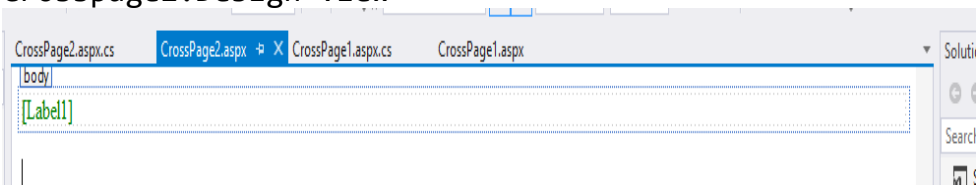
```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="CrossPage1.aspx.cs"
Inherits="CrossPage1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>CrossPage1</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

        FIRSTNAME:<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        <br />
        LASTNAME:<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
        <br />
        <br />
        <br />
        <asp:Button ID="Button1" runat="server"  PostBackUrl="~/CrossPage2.aspx"
Text="CROSSPAGE POSTBACK" />
        <br />
        <br />
        <asp:Button ID="Button2" runat="server" OnClick="Button2_Click1" Text="MANUAL
TRANSFER" />

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

Crosspage1.aspx.cs:

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

public partial class CrossPage1 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void Button2_Click1(object sender, EventArgs e)
    {
        Server.Transfer("~/CrossPage2.aspx");
    }
   public string fullName
    {
        get
        {
            return "<br>" + TextBox1.Text + "<br>" + TextBox2.Text;
        }
    }
```

```
}
```

Crosspage2:Design view



Crosspage2.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="CrossPage2.aspx.cs"
Inherits="CrossPage2" %>
<%@ PreviousPageType VirtualPath="~/CrossPage1.aspx" %>
<!DOCTYPE html>

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

        <asp:Label ID="Label1" runat="server" Text=""
style="color:green;"></asp:Label>

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

CrossPage2.aspx.cs:

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

public partial class CrossPage2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        try
        {
            Label1.Text = "you come from " + PreviousPage.Title + "<br>";
            CrossPage1 prev = (CrossPage1)PreviousPage;
            Label1.Text += "the text u entered in the all controls are " +
prev.fullName + "<br>";
            Label1.Text += "the text u entered in textbox1 is " +
((TextBox)prev.FindControl("TextBox1")).Text + "<br>";
        }
        catch (Exception er) {
            Response.Write("First go to crosspage1 otherwise it shows :" +
er.Message);
        }

    }
}
```

The Query String

Another common approach is to pass information using a query string in the URL. This approach is commonly found in search engines. For example, if you perform a search on the Google website, you'll be redirected to a new URL that incorporates your search parameters.

Here's an example: http://www.google.ca/search?q=organic+gardening

The query string is the portion of the URL after the question mark. In this case, it defines a single variable named *q*, which contains the string *organic+gardening*.

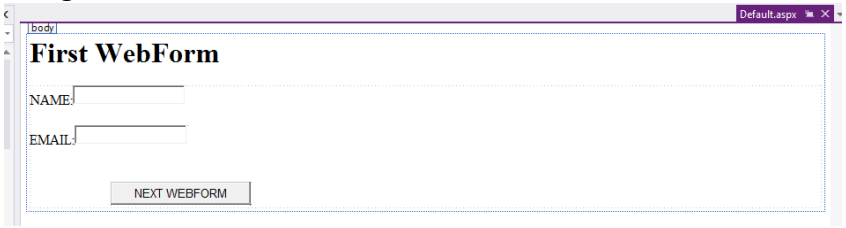
The advantage of the query string is that it's lightweight and doesn't exert any kind of burden on the server. However, it also has several limitations:

- Information is limited to simple strings, which must contain URL-legal characters.
- Information is clearly visible to the user and to anyone else who cares to eavesdrop on the Internet.
- The enterprising user might decide to modify the query string and supply new values, which your program won't expect and can't protect against.
- Many browsers impose a limit on the length of a URL (usually from 1KB to 2KB). For that reason, you can't place a large amount of information in the query string and still be assured of compatibility with most browsers.

Example:

Default page:

Design View:



Source View:

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

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <h1>First WebForm</h1>
    <form id="form1" runat="server">
    <div>

        NAME:<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        <br />
        EMAIL:<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
        <br/>
        <br />
```

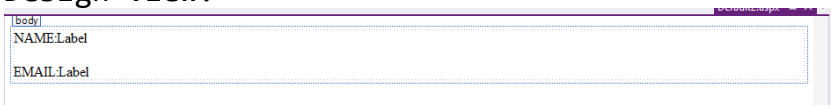
```
        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="NEXT
WEBFORM" />
 </div>
    </form>
</body>
</html>
```

Default.aspx.cs:

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

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

    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        Response.Redirect("~/Default2.aspx?Name=" + Server.UrlEncode(TextBox1.Text) +
"&Email=" +Server.UrlEncode( TextBox2.Text));
    }
}
```

Default2 Page:

Design View:



Source View:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default2.aspx.cs"
Inherits="Default2" %>

<!DOCTYPE html>

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

        NAME:<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
        <br />
        <br />
        EMAIL:<asp:Label ID="Label2" runat="server" Text="Label"></asp:Label>

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

Default2.aspx.cs:

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

public partial class Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Label1.Text = Request.QueryString["Name"];
        Label2.Text = Request.QueryString["Email"];
    }
}
```

Cookies:

Cookies provide another way that you can store information for later use. Cookies are small files that are created in the web browser's memory (if they're temporary) or on the client's hard drive (if they're permanent). One advantage of cookies is that they work transparently without the user being aware that information needs to be stored. They also can be easily used by any page in your application and even be retained between visits, which allows for truly long-term storage. They suffer from some of the same drawbacks that affect query strings—namely, they're limited to simple string information, and they're easily accessible and readable if the user finds and opens the corresponding file. These factors make them a poor choice for complex or private information or large amounts of data.

Some users disable cookies on their browsers, which will cause problems for web applications that require them. Also, users might manually delete the cookie files stored on their hard drives. But for the most part, cookies are widely adopted and used extensively on many websites.

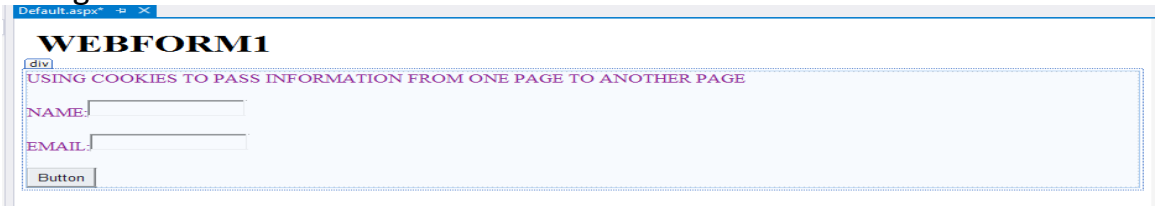
To set a cookie, just create a new HttpCookie object.

```
// Create the cookie object.
    HttpCookie cookie = new HttpCookie("Preferences");
```

Example:

Default Page:

Design View:



Source View:

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

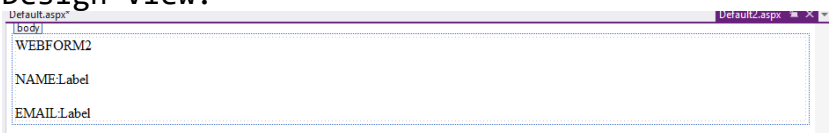
```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <h1>  WEBFORM1</h1>
    <form id="form1" runat="server">
    <div style="color: #800080">

        USING COOKIES TO PASS INFORMATION FROM ONE PAGE TO ANOTHER PAGE<br />
        <br />
        NAME:<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        <br />
        EMAIL:<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
        <br />
        <br />
        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Button"
/>
        <br />

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

Default.aspx.cs:

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

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

    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        HttpCookie c = new HttpCookie("user Info");
        c["name"] = TextBox1.Text;
        c["email"] = TextBox2.Text;
        Response.Cookies.Add(c);
        c.Expires = DateTime.Now.AddYears(1);
        Response.Redirect("~/Default2.aspx");
    }
}
```

Default2 Page:

Design View:



Source View:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default2.aspx.cs"
Inherits="Default2" %>

<!DOCTYPE html>

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

        WEBFORM2<br />
        <br />
        NAME:<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
        <br />
        <br />
        EMAIL:<asp:Label ID="Label2" runat="server" Text="Label"></asp:Label>

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

Deafault2.aspx.cs:

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

public partial class Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        HttpCookie C = Request.Cookies["USER INFO"];
        Label1.Text = C["name"];
        Label2.Text = C["email"];
    }
}
```

Session State:

ASP.NET session state enables you to store and retrieve values for a user as the user navigates ASP.NET pages in a Web application. HTTP is a stateless protocol. This means that a Web server treats each HTTP request for a page as an independent request. The server retains no knowledge of variable values that were used during previous requests. ASP.NET session state identifies requests from the same browser during a limited time window as a session, and provides a way to persist variable values for the duration of that session. By default, ASP.NET session state is enabled for all ASP.NET applications.

Session Variables:

Session variables are stored in a SessionStateItemCollection object that is exposed through the HttpContext.Session property. In an ASP.NET page, the current session variables are exposed through the **Session** property of the **Page** object.

Session Identifiers

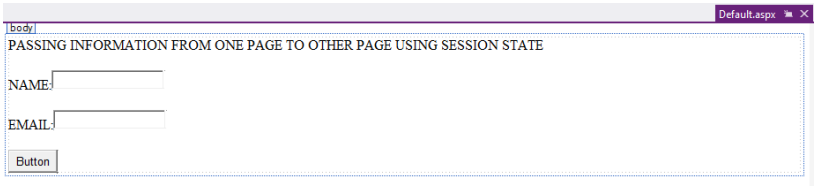
Sessions are identified by a unique identifier that can be read by using the SessionID property. When session state is enabled for an ASP.NET application, each request for a page in the application is examined for a SessionID value sent from the browser. If no SessionID value is supplied, ASP.NET starts a new session and the SessionID value for that session is sent to the browser with the response.

By default, SessionID values are stored in a cookie. However, you can also configure the application to store SessionID values in the URL for a "cookieless" session.

Example:

Default Page:

Design View:



Source View:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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">
    <div>

        PASSING INFORMATION FROM ONE PAGE TO OTHER PAGE USING SESSION STATE<br />
        <br />
        NAME:<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        <br />
        EMAIL:<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
        <br />
```

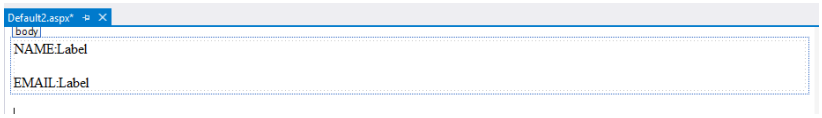
```
        <br />
        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Button"
/>

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

Default.aspx.cs:

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

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

    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        Session["Name"] = TextBox1.Text;
        Session["Email"] = TextBox2.Text;
        Response.Redirect("~/Default2.aspx");
    }

}
```

Default2 Page:

Design View:



Source View:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default2.aspx.cs"
Inherits="Default2" %>

<!DOCTYPE html>

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

        NAME:<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
        <br />
        <br />
```

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

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

Default2.aspx.cs:

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

public partial class Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["Name"] != null) {
            Label1.Text = (Session["Name"]).ToString();
        }
        if (Session["Email"] != null) {
            Label2.Text = Session["Email"].ToString();
        }
    }

}
```

Validation Controls:

ASP.NET validation controls validate the user input data to ensure that useless, unauthenticated, or contradictory data don't get stored.

ASP.NET provides the following validation controls:

- RequiredFieldValidator
- RangeValidator
- CompareValidator
- RegularExpressionValidator
- CustomValidator
- ValidationSummary

RequiredFieldValidator:

The RequiredFieldValidator control ensures that the required field is not empty. It is generally tied to a text box to force input into the textbox.

RangeValidator control:

The RangeValidator control verifies that the input value falls within a predetermined range.

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.

CompareValidator Control

The CompareValidator control compares a value in one control with a fixed value or a value in another 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.

RegularExpressionValidator Control:

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.

Custom Validator Control:

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.

Validation Summary:

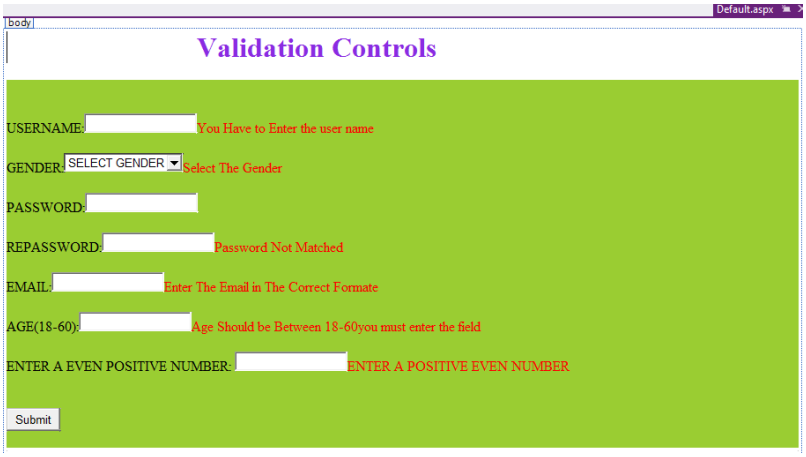
The ValidationSummary control does not perform any validation but shows a summary of all errors in the page. The summary displays the values of the ErrorMessage property of all validation controls that failed validation.

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.

Default page:

Design View:



Source View:

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

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script type="text/javascript" lang="javascript">
        function Even(source, args) {
            if (args.value == "") {
                args.IsValid = false;
            }
            if (args.value % 2 == 0) {
                args.IsValid = true;
            }
            else {
                args.IsValid = false;
            }
        }
    </script>
</head>
<body>
    <h1 style="color:
blueviolet">            &n
bsp;            &nbs
p; Validation Controls</h1>
    <form id="form1" runat="server">
        <div style="background-color: yellowgreen">


            <br />
            <br />
```

```
            USERNAME:<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
ControlToValidate="TextBox1" ErrorMessage="You Have to Enter the user name"
ForeColor="Red"></asp:RequiredFieldValidator>
            <br />
            <br />
            GENDER:<asp:DropDownList ID="DropDownList1" runat="server">
                <asp:ListItem Value="-1">SELECT GENDER</asp:ListItem>
                <asp:ListItem>MALE</asp:ListItem>
                <asp:ListItem>FEMALE</asp:ListItem>
            </asp:DropDownList>
            <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
ControlToValidate="DropDownList1" ErrorMessage="Select The Gender" ForeColor="Red"
InitialValue="-1"></asp:RequiredFieldValidator>
            <br />
            <br />
            PASSWORD:<asp:TextBox ID="TextBox2" runat="server"
TextMode="Password"></asp:TextBox>
            <br />
            <br />
            REPASSWORD:<asp:TextBox ID="TextBox3" runat="server"
TextMode="Password"></asp:TextBox>
            <asp:CompareValidator ID="CompareValidator1" runat="server"
ControlToCompare="TextBox2" ControlToValidate="TextBox3" ErrorMessage="Password Not
Matched" ForeColor="Red"></asp:CompareValidator>
            <br />
            <br />
            EMAIL:<asp:TextBox ID="TextBox4" runat="server"></asp:TextBox>
            <asp:RegularExpressionValidator ID="RegularExpressionValidator1"
runat="server" ControlToValidate="TextBox4" ErrorMessage="Enter The Email in The
Correct Formate" ForeColor="Red" ValidationExpression="\w+([-+.']\w+)*@\w+([-
.]\w+)*\.\w+([-.]\w+)*"></asp:RegularExpressionValidator>
            <br />
            <br />
            AGE(18-60):<asp:TextBox ID="TextBox5" runat="server"></asp:TextBox>
            <asp:RangeValidator ID="RangeValidator1" runat="server" Display="Dynamic"
ErrorMessage="Age Should be Between 18-60" ForeColor="Red" MaximumValue="60"
MinimumValue="18" Type="Integer" ControlToValidate="TextBox5"></asp:RangeValidator>
            <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
Display="Dynamic" ErrorMessage="you must enter the field" ForeColor="Red"
ControlToValidate="TextBox5"></asp:RequiredFieldValidator>
            <br />
            <br />

            ENTER A EVEN POSITIVE NUMBER:
        <asp:TextBox ID="TextBox6" runat="server"></asp:TextBox>
            <asp:CustomValidator ID="CustomValidator1" runat="server"
ControlToValidate="TextBox6" ErrorMessage="ENTER A POSITIVE EVEN NUMBER"
ForeColor="Red" ClientValidationFunction="Even"
OnServerValidate="CustomValidator1_ServerValidate"
ValidateEmptyText="True"></asp:CustomValidator>
            <br />
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" Text="Submit"
OnClick="Button1_Click" />
            <br />
            <br />



        </div>
```

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



        </p>
    </form>
</body>
</html>
```

Default.aspx.cs:

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

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

    }
    protected void CustomValidator1_ServerValidate(object source,
ServerValidateEventArgs args)
    {
        if (args.Value == "") {
            args.IsValid = false;
        }
        int number;
        bool a = int.TryParse(args.Value, out number);
        if (a && number >= 0 && number % 2 == 0)
        {
            args.IsValid = true;
        }
        else {
            args.IsValid = false;
        }
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        if (Page.IsValid) {
            Label1.ForeColor=System.Drawing.Color.Green;
            Label1.Text = "Data Saved Succesfully";
        }
        else
        {
            Label1.ForeColor=System.Drawing.Color.Red;
            Label1.Text="Data Not saved Succesfully";
        }
    }

}
```

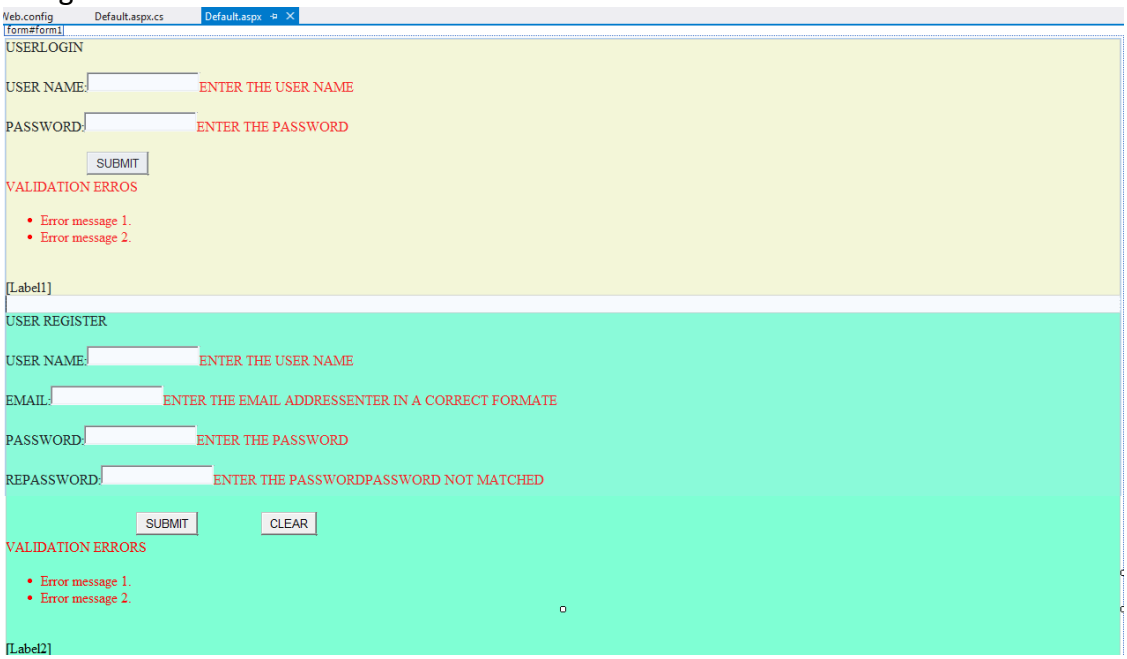
Validation Groups:

Complex pages have different groups of information provided in different panels. In such situation ,a need might arise for performing validation separately for separate group. This kind of situation is handled using validation groups.

To create a validation group, you should put the input controls and the validation controls into the same logical group by setting their *ValidationGroup* property.

EXAMPLE:

Design View:



SourceView:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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">
    <div style="background-color:lightgoldenrodyellow">
        <asp:Panel ID="Panel1" runat="server">
            USERLOGIN<br />
            <br />
            USER NAME:<asp:TextBox ID="TextBox1" runat="server"
ValidationGroup="GROUP1"></asp:TextBox>
            <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
ControlToValidate="TextBox1" ErrorMessage="ENTER THE USER NAME" ForeColor="Red"
ValidationGroup="GROUP1"></asp:RequiredFieldValidator>
            <br />
            <br />
            PASSWORD:<asp:TextBox ID="TextBox2" runat="server" TextMode="Password"
ValidationGroup="GROUP1"></asp:TextBox>
```

```
            <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
ControlToValidate="TextBox2" ErrorMessage="ENTER THE PASSWORD" ForeColor="Red"
ValidationGroup="GROUP1"></asp:RequiredFieldValidator>
            <br />
            <br />

             &n
bsp;       
            <asp:Button ID="Button2" runat="server" Text="SUBMIT"
ValidationGroup="GROUP1" OnClick="Button2_Click" />
            <asp:ValidationSummary ID="ValidationSummary1" runat="server"
ForeColor="Red" HeaderText="VALIDATION ERROS" ShowMessageBox="True"
ValidationGroup="GROUP1" />
            <br />
            <asp:Label ID="Label1" runat="server" Text=""></asp:Label>

        </asp:Panel>
    </div>

        <br/>
        <div style="background-color:aquamarine">
            <asp:Panel ID="Panel2" runat="server">
                USER REGISTER<br />
                <br />
                USER NAME:<asp:TextBox ID="TextBox4" runat="server"
ValidationGroup="GROUP2"></asp:TextBox>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator3"
runat="server" ControlToValidate="TextBox4" ErrorMessage="ENTER THE USER NAME"
ForeColor="Red" ValidationGroup="GROUP2"></asp:RequiredFieldValidator>
                <br />
                <br />
                EMAIL:<asp:TextBox ID="TextBox5" runat="server"
ValidationGroup="GROUP2"></asp:TextBox>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator4"
runat="server" ControlToValidate="TextBox5" ErrorMessage="ENTER THE EMAIL ADDRESS"
ForeColor="Red" ValidationGroup="GROUP2"></asp:RequiredFieldValidator>
                <asp:RegularExpressionValidator ID="RegularExpressionValidator1"
runat="server" ControlToValidate="TextBox5" ErrorMessage="ENTER IN A CORRECT FORMATE"
ForeColor="Red" ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*"
ValidationGroup="GROUP2"></asp:RegularExpressionValidator>
                <br />
                <br />
                PASSWORD:<asp:TextBox ID="TextBox6" runat="server" TextMode="Password"
ValidationGroup="GROUP2"></asp:TextBox>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator5"
runat="server" ControlToValidate="TextBox6" Display="Dynamic" ErrorMessage="ENTER THE
PASSWORD" ForeColor="Red" ValidationGroup="GROUP2"></asp:RequiredFieldValidator>
                <br />
                <br />
                REPASSWORD:<asp:TextBox ID="TextBox7" runat="server"
TextMode="Password" ValidationGroup="GROUP2"></asp:TextBox>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator6"
runat="server" ControlToValidate="TextBox7" ErrorMessage="ENTER THE PASSWORD"
ForeColor="Red" ValidationGroup="GROUP2"></asp:RequiredFieldValidator>
                <asp:CompareValidator ID="CompareValidator1" runat="server"
ControlToCompare="TextBox6" ControlToValidate="TextBox7" Display="Dynamic"
ErrorMessage="PASSWORD NOT MATCHED" ForeColor="Red"
ValidationGroup="GROUP2"></asp:CompareValidator>
                <br />
                <br />

             &n
```

```
bsp;              &nbs
p;       
                <asp:Button ID="Button1" runat="server" Text="SUBMIT"
ValidationGroup="GROUP2" OnClick="Button1_Click" />

              &n
bsp;  
                <asp:Button ID="Button3" runat="server" Text="CLEAR"
ValidationGroup="GROUP2" OnClick="Button3_Click" />
                <asp:ValidationSummary ID="ValidationSummary2" runat="server"
ForeColor="Red" HeaderText="VALIDATION ERRORS" ShowMessageBox="True"
ValidationGroup="GROUP2" />
                <br />
                <asp:Label ID="Label2" runat="server" Text=""></asp:Label>
            </asp:Panel>
            </div>
    </form>
</body>
</html>
```

Default.aspx.cs:

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

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

    }
    protected void Button3_Click(object sender, EventArgs e)
    {
        TextBox4.Text = "";
        TextBox5.Text = "";
        TextBox6.Text = "";
        TextBox7.Text = "";
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        if (Page.IsValid)
        {
            Label2.ForeColor = System.Drawing.Color.Green;
            Label2.Text = "Data Saved Succesfully";
        }
        else
        {
            Label2.ForeColor = System.Drawing.Color.Red;
            Label2.Text = "Data Not saved Succesfully";
        }
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        if (Page.IsValid)
        {
            Label1.ForeColor = System.Drawing.Color.Green;
            Label1.Text = "Data Saved Succesfully";
        }
```

```
        else
        {
            Label1.ForeColor = System.Drawing.Color.Red;
            Label1.Text = "Data Not saved Succesfully";
        }
    }
}
```

Web.config:

```
<?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>
  <appSettings>
    <add key="ValidationSettings:unobtrusivevalidationmode" value="None"/>
  </appSettings>
  <system.web>
      <compilation debug="true" targetFramework="4.5" />
      <httpRuntime targetFramework="4.5" />
    </system.web>

</configuration>
```

calendar control:

The calendar control is a functionally rich web control, which provides the following capabilities:

- Displaying one month at a time
- Selecting a day, a week or a month
- Selecting a range of days
- Moving from month to month
- Controlling the display of the days programmatically

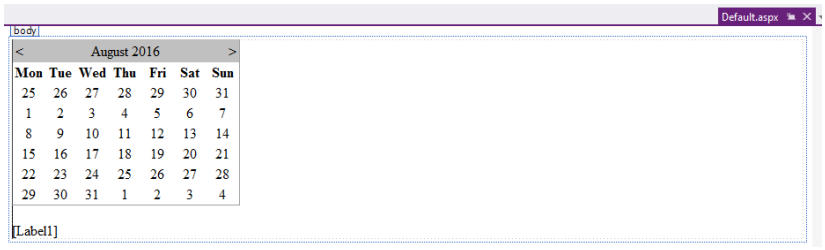
The basic syntax of a calendar control is:

```
<asp:Calender ID = "Calendar1" runat = "server">
</asp:Calender>
```

Example:

Default Page:

Design View:



Source View:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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">
    <div>

        <asp:Calendar ID="Calendar1" runat="server" OnDayRender="Calendar1_DayRender"
OnSelectionChanged="Calendar1_SelectionChanged"></asp:Calendar>
        <br />
        <asp:Label ID="Label1" runat="server" Text=""></asp:Label>

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

Default.aspx.cs:

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

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

    }
    protected void Calendar1_SelectionChanged(object sender, EventArgs e)
    {
        Label1.Text = "you selected these dates:<br>";
        foreach (DateTime dt in Calendar1.SelectedDates) {
            Label1.Text += dt.ToLongDateString() + "<br>";
        }
    }
    protected void Calendar1_DayRender(object sender, DayRenderEventArgs e)
    {
        if (e.Day.Date.Day == 21 && e.Day.Date.Month == 10) {
            e.Cell.BackColor = System.Drawing.Color.Yellow;
            Label l1 = new Label();
```

```
            l1.Text = "<br>my birthday";
            e.Cell.Controls.Add(l1);
        }
    }

}
```

MultiView:

MultiView and View controls allow you to divide the content of a page into different groups, displaying only one group at a time. Each View control manages one group of content and all the View controls are held together in a MultiView control.

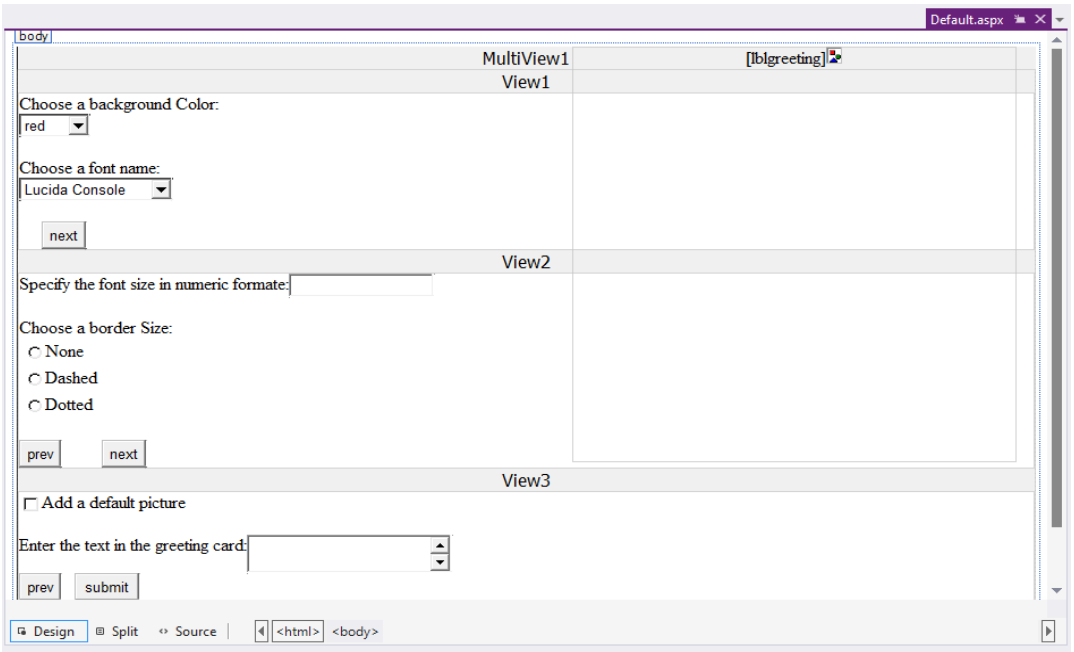
The MultiView control is responsible for displaying one View control at a time. The View displayed is called the active view.

The syntax of MultiView control is:

```
<asp:MultView ID= "MultiView1" runat= "server">
</asp:MultiView>
```

Example:

DESIGN VIEW:



SOURCE VIEW:

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

<!DOCTYPE html>
```

```
<html xbmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:MultiView ID="MultiView1" runat="server">
            <asp:View ID="View1" runat="server">Choose a background Color:<br />
                <asp:DropDownList ID="DropDownList1" runat="server">
                    <asp:ListItem>red</asp:ListItem>
                    <asp:ListItem>yellow</asp:ListItem>
                    <asp:ListItem>green</asp:ListItem>
                    <asp:ListItem>pink</asp:ListItem>
                </asp:DropDownList>
                <br />
                <br />
                Choose a font name:<br />
                <asp:DropDownList ID="DropDownList2" runat="server">
                    <asp:ListItem>Lucida Console</asp:ListItem>
                    <asp:ListItem>Times New Roman</asp:ListItem>
                    <asp:ListItem>Vivaldi</asp:ListItem>
                    <asp:ListItem>Viner Hand ITC</asp:ListItem>
                </asp:DropDownList>
                <br />
                <br />
                    
                <asp:Button ID="Button1" runat="server" Text="next"
OnClick="Button1_Click" />
            </asp:View>
            <asp:View ID="View2" runat="server">Specify the font size in numeric
formate:<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                <br />
                <br />
                Choose a border Size:<asp:RadioButtonList ID="RadioButtonList1"
runat="server">
                    <asp:ListItem>None</asp:ListItem>
                    <asp:ListItem>Dashed</asp:ListItem>
                    <asp:ListItem>Dotted</asp:ListItem>
                </asp:RadioButtonList>
                <br />
                <asp:Button ID="Button2" runat="server" Text="prev"
OnClick="Button2_Click" />
                        
                <asp:Button ID="Button3" runat="server" Text="next"
OnClick="Button3_Click" />
            </asp:View>
            <asp:View ID="View3" runat="server">
                <asp:CheckBox ID="CheckBox1" runat="server" Text="Add a default
picture" />
                <br />
                <br />
                Enter the text in the greeting card:<asp:TextBox ID="TextBox2"
runat="server" TextMode="MultiLine"></asp:TextBox>
                <br />
                <asp:Button ID="Button4" runat="server" Text="prev"
OnClick="Button4_Click" />
                  
                <asp:Button ID="Button5" runat="server" Text="submit"
OnClick="Button5_Click" />
            </asp:View>
```

```
        </asp:MultiView>
    </div>
    </form>
    <asp:Panel ID="panel1" HorizontalAlign="Center"  runat ="server"
style="position:absolute;top:15px;left:500px; height: 390px; width: 392px; ">
          <asp:Label ID="lblgreeting" runat="server" Text=""></asp:Label>
      <asp:Image ID="Image1" runat="server" Visible="false" />
    </asp:Panel>
</body>
</html>
```

DEFAULT.ASPX.CS:

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

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            MultiView1.ActiveViewIndex = 0;
        }


        Image1.ImageUrl = "happy.jpg";
    }

    protected void Button5_Click(object sender, EventArgs e)
    {
        panel1.BackColor = Color.FromName(DropDownList1.SelectedItem.Text);
        lblgreeting.Font.Name = DropDownList2.SelectedItem.Text;
        if (int.Parse(TextBox1.Text) > 0)
        {
            lblgreeting.Font.Size = FontUnit.Point(int.Parse(TextBox1.Text));
        }
        if (RadioButtonList1.SelectedIndex == 0)
        {

            panel1.BorderStyle = BorderStyle.None;
        }
        if (RadioButtonList1.SelectedIndex == 1)
        {
            panel1.BorderStyle = BorderStyle.Dashed;
        }
        if (RadioButtonList1.SelectedIndex == 2)
        {
            panel1.BorderStyle = BorderStyle.Dotted;
        }
        if (CheckBox1.Checked)
            Image1.Visible = true;
        else
            Image1.Visible = false;
        lblgreeting.Text = TextBox2.Text;
```

```
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        MultiView1.ActiveViewIndex = 1;
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        MultiView1.ActiveViewIndex = 0;
    }
    protected void Button3_Click(object sender, EventArgs e)
    {
        MultiView1.ActiveViewIndex = 2;
    }
    protected void Button4_Click(object sender, EventArgs e)
    {
        MultiView1.ActiveViewIndex = 1;
    }

}
```

Adrotator control:

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" Advertisem entFile = "adfile.xm l" Target = "_blank" />
```

Example:

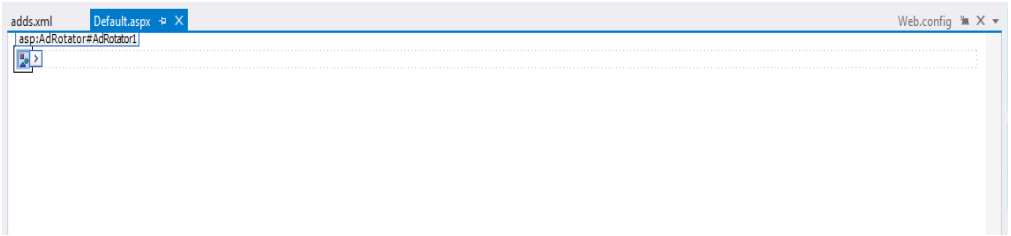
Add.xml:

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
  <Ad>
    <ImageUrl>~/1.jpg</ImageUrl>
    <height>60</height>
    <width>190</width>
    <NavigateUrl>https://thewaltdisneycompany.com/</NavigateUrl>
    <AlternateText>Microsoft Main Site</AlternateText>
    <Impressions>80</Impressions>
    <Keyword>1</Keyword>
  </Ad>
  <Ad>
    <ImageUrl>~/2.jpg</ImageUrl>
    <height>90</height>
    <width>90</width>
    <NavigateUrl>https://www.airtelindia.in/</NavigateUrl>
    <AlternateText>Wingtip Toys</AlternateText>
```

```
    <Impressions>80</Impressions>
    <Keyword>2</Keyword>
  </Ad>
  <Ad>
    <ImageUrl>~/3.jpg</ImageUrl>
    <height>90</height>
    <width>90</width>
    <NavigateUrl>https://www.foodpanda.in/</NavigateUrl>
    <AlternateText>food panda</AlternateText>
    <Impressions>80</Impressions>
    <Keyword>3</Keyword>
  </Ad>

</Advertisements>
```

Default.aspx:

Design View:



Source View:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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">
    <div>
        <asp:AdRotator id="AdRotator1" runat="server"
            Target="_blank"
            AdvertisementFile="~/adds.xml"/>

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

Wizard control:

The Wizard control provides navigation through a series of steps that collect information incrementally from a user. Many websites include functionality that collects information from the end user (e.g. checking out on an ecommerce website). The Wizard consists of:

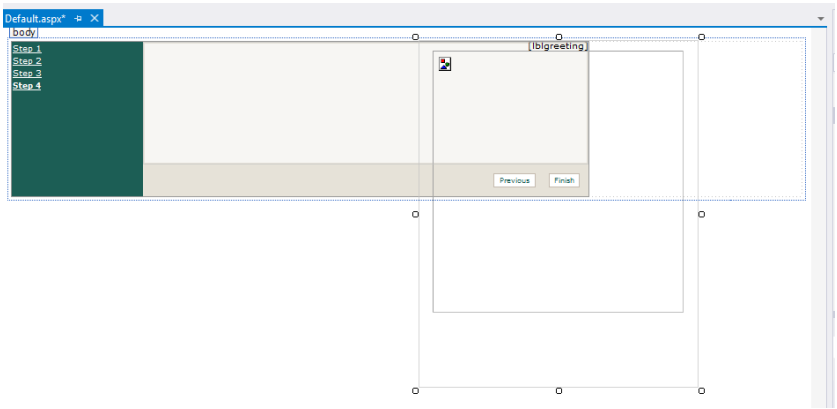
- **Collection of WizardSteps:** Each WizardStep contains a discrete piece of content to be displayed to the user. Only one WizardStep will be displayed at a time.
- **Navigation Area:** The navigation area below each WizardStep that contains the navigation buttons to go the next and pervious steps in the wizard.
- **SideBar:** An optional element that contains a list of all WizardSteps and provides a means to skip around the WizardSteps in a random order.
- **Header:** An optional element to provide consistent information at the top of the WizardStep.

The StepType associated with each WizardStep determines the type of navigation buttons that will be displayed for that step. The StepTypes are:

- **Start:** Displays a Next button.
- **Step:** Displays Next and Previous buttons.
- **Finish:** Displays a Finish button.
- **Complete:** Displays no navigation buttons and hides the SideBar if it is displayed.
- **Auto:**One of the step types listed above is selected based on the order of the step in the collection (e.g. the first step will have a Next button).

Example:

DESIGN VIEW:



SOURCE VIEW:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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">
    <div>
```

```
        <asp:Wizard ID="Wizard1" runat="server" ActiveStepIndex="3"
BackColor="#E6E2D8" BorderColor="#999999" BorderStyle="Solid" BorderWidth="1px" Font-
Names="Verdana" Font-Size="0.8em" Height="177px"
OnFinishButtonClick="Wizard1_FinishButtonClick" Width="657px">
            <HeaderStyle BackColor="#666666" BorderColor="#E6E2D8" BorderStyle="Solid"
BorderWidth="2px" Font-Bold="True" Font-Size="0.9em" ForeColor="White"
HorizontalAlign="Center" />
            <NavigationButtonStyle BackColor="White" BorderColor="#C5BBAF"
BorderStyle="Solid" BorderWidth="1px" Font-Names="Verdana" Font-Size="0.8em"
ForeColor="#1C5E55" />
            <SideBarButtonStyle ForeColor="White" />
            <SideBarStyle BackColor="#1C5E55" Font-Size="0.9em" VerticalAlign="Top" />
            <StepStyle BackColor="#F7F6F3" BorderColor="#E6E2D8" BorderStyle="Solid"
BorderWidth="2px" />
            <WizardSteps>
                <asp:WizardStep runat="server" title="Step 1">
                    CHOOSE A BACKGROUND COLOR:<asp:DropDownList ID="DropDownList1"
runat="server">
        </asp:DropDownList>
        <br />
        <br />
        CHOOSE A FONTNAME:<asp:DropDownList ID="DropDownList2" runat="server">
        </asp:DropDownList>
        <br />
        <br />
                </asp:WizardStep>
                <asp:WizardStep runat="server" title="Step 2">
                     SPECIFY NUMERIC FONT SIZE:<asp:TextBox ID="TextBox1"
runat="server"></asp:TextBox>
        <br />
        <br />
        CHOOSE A BORDER STYLE:<asp:RadioButtonList ID="RadioButtonList1"
runat="server">
        </asp:RadioButtonList>
        <br />
                </asp:WizardStep>
                <asp:WizardStep ID="WizardStep1" runat="server" title="Step 3">
                    <asp:CheckBox ID="CheckBox1" runat="server" Text="ADD A DEFAULT
PICTURE" />
        <br />
        <br />
        ENTER THE GREETING TEXT:<asp:TextBox ID="TextBox2" runat="server"
TextMode="MultiLine"></asp:TextBox>
        <br />
        <br />
        <br />
                </asp:WizardStep>
                <asp:WizardStep ID="WizardStep2" runat="server" title="  4"><asp:Panel
runat="server" ID="panel1" style="position:absolute;top:14px; left:473px; width:
318px; height: 394px;" HorizontalAlign="Center">
         <asp:Label ID="lblgreeting" runat="server" ></asp:Label>
         <br />
         <asp:Image ID="Image1" Visible="false" width="285px" runat="server"
Height="297px" />
        </asp:Panel>
                </asp:WizardStep>
            </WizardSteps>
        </asp:Wizard>

    </div>
    </form>
```

```
</body>
</html>
```

DEFAULT2.ASPX.CS:

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

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        DropDownList1.Items.Add("Red");
        DropDownList1.Items.Add("Yellow");
        DropDownList1.Items.Add("Pink");
        DropDownList1.Items.Add("Blue");
        DropDownList2.Items.Add("TimesNewRoman");
        DropDownList2.Items.Add("Gabriola");
        DropDownList2.Items.Add("Arial");
        DropDownList2.Items.Add("Impact");
        ListItem li = new ListItem();
        li.Text = BorderStyle.None.ToString();
        li.Value = ((int)BorderStyle.None).ToString();
        RadioButtonList1.Items.Add(li);
        li = new ListItem();
        li.Text = BorderStyle.Dashed.ToString();
        li.Value = ((int)BorderStyle.Dashed).ToString();
        RadioButtonList1.Items.Add(li);
        li = new ListItem();
        li.Text = BorderStyle.Dotted.ToString();
        li.Value = ((int)BorderStyle.Dotted).ToString();
        RadioButtonList1.Items.Add(li);

        Image1.ImageUrl = "happy.jpg";
    }
    protected void Wizard1_FinishButtonClick(object sender, WizardNavigationEventArgs
e)
    {
        panel1.BackColor = Color.FromName(DropDownList1.SelectedItem.Text);
        lblgreeting.Font.Name = DropDownList2.SelectedItem.Text;
        if (int.Parse(TextBox1.Text) > 0)
        {
            lblgreeting.Font.Size = FontUnit.Point(int.Parse(TextBox1.Text));
        }
        if (RadioButtonList1.SelectedIndex == 0)
        {
            int value = int.Parse(RadioButtonList1.SelectedItem.Value);
            panel1.BorderStyle = (BorderStyle)value;
        }
        if (RadioButtonList1.SelectedIndex == 1)
        {
            int value = int.Parse(RadioButtonList1.SelectedItem.Value);
            panel1.BorderStyle = (BorderStyle)value;
        }
        if (RadioButtonList1.SelectedIndex == 2)
        {
```

```
            int value = int.Parse(RadioButtonList1.SelectedItem.Value);
            panel1.BorderStyle = (BorderStyle)value;
        }
        if (CheckBox1.Checked)
            Image1.Visible = true;
        else
            Image1.Visible = false;
        lblgreeting.Text = TextBox2.Text;
    }

}
```

What are the advantages of master pages ?How master page and content pages are connected to each other . (6M)

i. Master pages enable consistent and standardized layout of the website.
ii. You can make layout changes of the site in master page instead of making changes in the pages.
iii. It provide an object model which allows you to customize the master page from individual content pages.
iv. It allows you to centralize the common functionality of your pages so that you can make updates in just one place.

Master Pages and Content Pages Are Connected:

The ContentPlaceHolder is the portion of the master page that a content page can change

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

<%@ Page Language = "C#" MasterPageFile = "~/SiteTemplate.master" AutoEventWireup = "true" CodeFile = "SimpleContentPage.aspx.cs" Inherits = "SimpleContentPage" Title = "Untitled Page" %>

<asp:Content ID = "Content1" ContentPlaceHolderID = "MainContent" runat = "Server">

UNIT-III

Explain about styles, themes

Web pages can use styles in three different ways:

• *Inline style*: An inline style is a style that's placed directly inside an HTML tag. This can get messy, but it's a reasonable approach for one-time formatting. You can remove the style and put it in a style sheet later.

<p **style="background: Blue"**>This text has a blue background.</p>

• *Internal style sheet*: An internal style sheet is a collection of styles that are placed in the <head> section of your web page markup. You can then use the styles from this style sheet to format the web controls on that page. By using an internal style sheet, you get a clear separation between formatting (your styles) and your content (the rest of your HTML markup). You can also reuse the same style for multiple elements.

```
<style type="text/css">
.red{
   color: red;
}
</style>

<body>
<p class="red">This is red</p>
</body>
```

• *External style sheet*: An external style sheet is similar to an internal style sheet, except it's placed in a completely separate file. This is the most powerful approach, because it gives you a way to apply the same style rules to many pages.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>...</title>
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
...
</body>
</html>
```

What is ADO.Net ? Write a program to perform insert ,update ,delete using ADO.Net

ADO.NET

1. ADO.NET is known as database technology which is used to connect with the database. That means some objects will work for interacting with database
2. Basically why we require database is because the front end application itself cannot store data directly, so that we require a storage mechanism. That storage mechanism is known as database
3. It can be used for database connection and offers to perform database manipulations like inserting data, deleting unnecessary data, retrieving the required data from tables.
4. It can be used in any type of applications like console applications, windows form application, website
5. It can be used in any dot net language like c#, VB, etc..
6. It was developed based on its previous version called ADO.
7. It offers much efficient features to easily handle with the database tables & especially when you are dealing with multiple

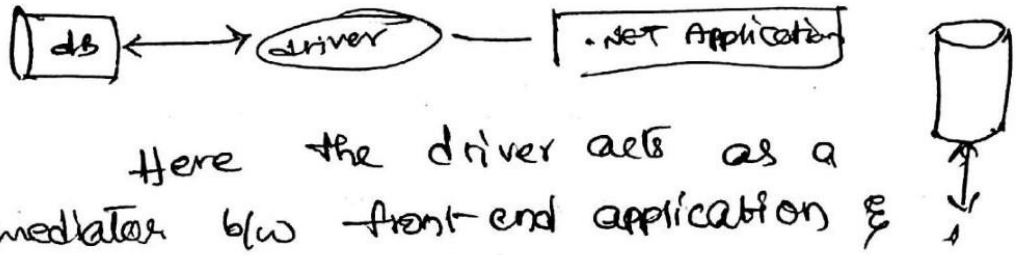
Types of tables we can connect

- file database - firefox, Ms access
- server databases - Sql, Oracle

what we can do using ADO.Net

Insert some data into database table
delete some data from database table
update the data of the table
Retrieve some data from the table

ADO.net connection database Architecture



Here the driver acts as a mediator b/w front-end application & backend databases

The driver can also be called as a provider.

Understanding the connection string:

The connection string provides the details about connections string i.e if you want to connect with database you have to specify some details like server, userid, password, provider

server

Specify the name of the server system which you want to connect

The server name can also be called

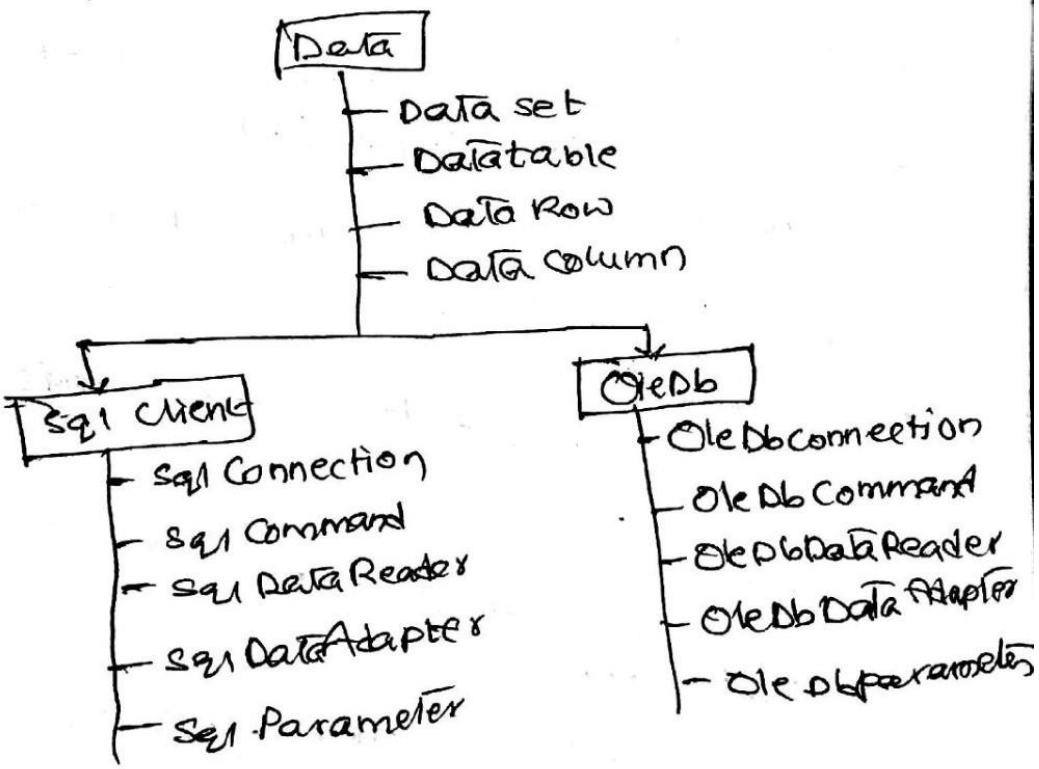
user-id: Specify the user name for logging with the database. specify the password for logging in with the database

provider: specify the name of the driver which you want to use with the connection.

ADO.NET Library:

To perform above mentioned database operations. ADO.NET Technology offers some predefined classes organized in the form of name structure:

library: System.data



ADO.NET NAMESpaces:

1) System.Data

contains necessary classes & namespaces to manipulate the databases

eg System.Data.SqlClient

contains necessary classes, used

ADO.NET Classes:

1) **Connection:**
Maintains connection with database

2) **Command:**
executes a query statement, non query statement / insert / delete / update function at backend

3) **Data Reader:**
It acts as buffer, which holds data, after execution of query statement at backend

4) **Data Adapter:**
Executes a query statement at backend

5) **Parameter:**
Sends a parameter value to backend stored procedure / function

6) **Dataset**
Acts as buffer which holds multiple tables at a time

7) **Data Table:**
Acts as buffer, which holds a single table

8) **Data row:**
Acts as buffer, which holds single row

9) **Data Column:**
Acts as buffer, which holds Single column

Sending commands to database

1. Insertion
2. Deletion
3. Updation
4. Selection

Insert : To insert a row

Delete : To delete or more rows from table

Updation : To update the table data

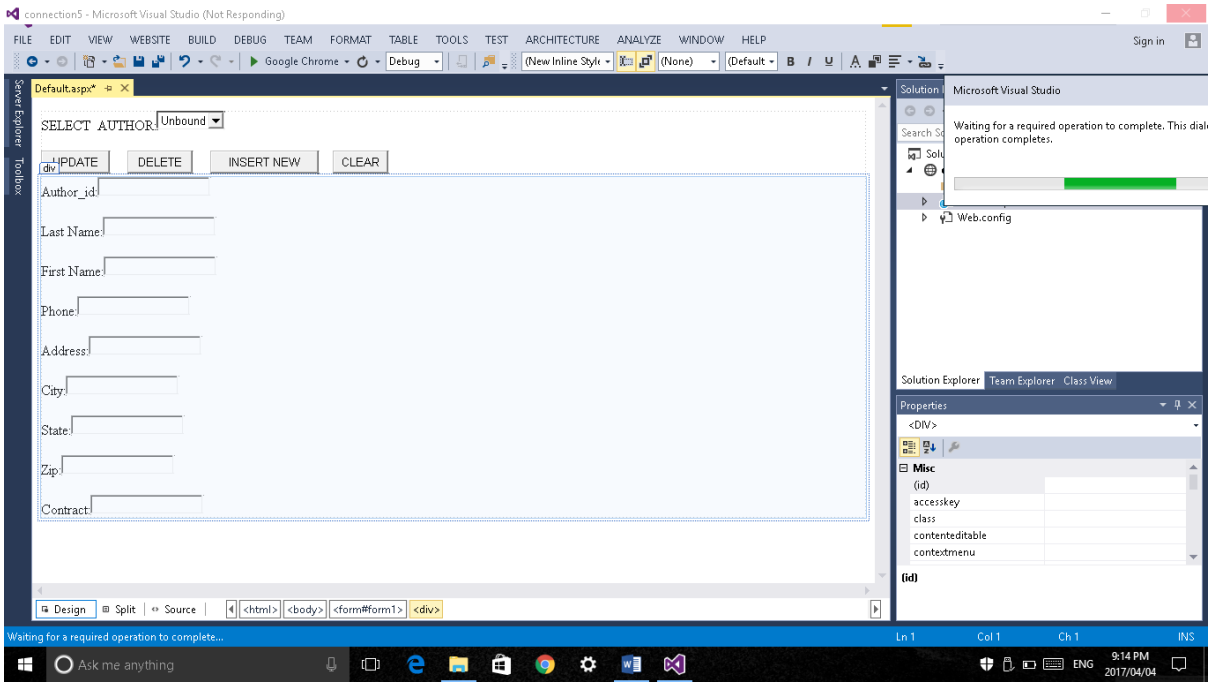
Select : To retrieve the data from database table into frontend application

Function : To call stored procedure from frontend application, that is already created at backend.

ExecuteNonQuery() : This method is used to execute any SQL statement. This method moves the execution flow to backend database, execute the command there and then come back with some result. This method returns "no. of rows affected" which represents the count of rows, which are affected by executing this command. Suppose after executing a delete statement, 2 rows are evoked.

ExecuteReader() : To execute 'select' statement only. used to retrieve some data from database based on given select statement. This method moves the execution flow to backend database, execute the, execute the command and then come back.

Design:



Default.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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">
    <div>

        SELECT  AUTHOR:<asp:DropDownList ID="DropDownList1" runat="server"
OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged" AutoPostBack="True">
        </asp:DropDownList>
        <br />
        <br />
        <asp:Button ID="Button1" runat="server" Text="UPDATE" OnClick="Button1_Click"
/>
    
        <asp:Button ID="Button2" runat="server" Text="DELETE" OnClick="Button2_Click"
/>
    
        <asp:Button ID="Button3" runat="server" Text="INSERT NEW"
OnClick="Button3_Click" />
   
        <asp:Button ID="Button4" runat="server" Text="CLEAR" OnClick="Button4_Click"
/>

    </div>
```

```
        <div>

            Author_id:<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <br />
            <br />
            Last Name:<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
            <br />
            <br />
            First Name:<asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
            <br />
            <br />
            Phone:<asp:TextBox ID="TextBox4" runat="server"></asp:TextBox>
            <br />
            <br />
            Address:<asp:TextBox ID="TextBox5" runat="server"></asp:TextBox>
            <br />
            <br />
            City:<asp:TextBox ID="TextBox6" runat="server"></asp:TextBox>
            <br />
            <br />
            State:<asp:TextBox ID="TextBox7" runat="server"></asp:TextBox>
            <br />
            <br />
            Zip:<asp:TextBox ID="TextBox8" runat="server"></asp:TextBox>
            <br />
            <br />
            Contract:<asp:TextBox ID="TextBox9" runat="server"></asp:TextBox>

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

Default.aspx.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

public partial class _Default : System.Web.UI.Page
{
    SqlConnection con;
    SqlCommand cmd;
    SqlDataReader sdr;
    protected void Page_Load(object sender, EventArgs e)
    {
        con = new SqlConnection("data source=l223\\sqlexpress;database=pubs;User
Id=sa;password=sqlserver");
        cmd = new SqlCommand("select * from authors", con);
        con.Open();
        sdr = cmd.ExecuteReader();
            while (sdr.Read())
                {
                    ListItem li = new ListItem();
                    li.Text = sdr["au_lname"] + "," + sdr["au_fname"];
```

```
                    li.Value = sdr["au_id"].ToString();
                    DropDownList1.Items.Add(li);
                }
                con.Close();
    }
    protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
    {
        cmd.Connection = con;
        cmd.CommandText = "select * from authors where au_id='" +
DropDownList1.SelectedItem.Value + "'";
        con.Open();
        sdr = cmd.ExecuteReader();
        while (sdr.Read()) {
            TextBox1.Text = sdr[0].ToString();
            TextBox2.Text = sdr[1].ToString();
            TextBox3.Text = sdr[2].ToString();
            TextBox4.Text = sdr[3].ToString();
            TextBox5.Text = sdr[4].ToString();
            TextBox6.Text = sdr[5].ToString();
            TextBox7.Text = sdr[6].ToString();
            TextBox8.Text = sdr[7].ToString();
            TextBox9.Text = sdr[8].ToString();

        }
        con.Close();
    }
    protected void Button4_Click(object sender, EventArgs e)
    {
        TextBox1.Text = "";
        TextBox2.Text = "";
        TextBox3.Text = "";
        TextBox4.Text = "";
        TextBox5.Text = "";
        TextBox6.Text = "";
        TextBox7.Text = "";
        TextBox8.Text = "";
        TextBox9.Text = "";

    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        TextBox1.Enabled = false;
        cmd.CommandText = "update authors set
au_lname='"+TextBox2.Text+"',au_fname='"+TextBox3.Text+"'where
au_id='"+TextBox1.Text+"';";
        try
        {
            con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
            Response.Redirect("~/Default.aspx");
        }
        catch (Exception err) {

        }
    }

    protected void Button3_Click(object sender, EventArgs e)
    {
         try{
        string insertSQL;
```

```
        insertSQL = "insert into authors values(@au_id,@au_fname,
@au_lname,@phone,@address,@city,@state, @zip, @contract);";
        SqlCommand cmd = new SqlCommand(insertSQL, con);
        con.Open();
        cmd.Parameters.AddWithValue("@au_id", TextBox1.Text);
        cmd.Parameters.AddWithValue("@au_fname", TextBox3.Text);
        cmd.Parameters.AddWithValue("@au_lname", TextBox2.Text);
        cmd.Parameters.AddWithValue("@phone", TextBox4.Text);
        cmd.Parameters.AddWithValue("@address", TextBox5.Text);
        cmd.Parameters.AddWithValue("@city", TextBox6.Text);
        cmd.Parameters.AddWithValue("@state", TextBox7.Text);
        cmd.Parameters.AddWithValue("@zip", TextBox8.Text);
        cmd.Parameters.AddWithValue("@contract",int.Parse(TextBox9.Text));
            cmd.ExecuteNonQuery();
            con.Close();
        }
        catch (Exception err) {
            Label1.Text = err.Message;
        }


    }
    protected void Button2_Click(object sender, EventArgs e)
    {
            SqlCommand cmd = new SqlCommand();
            cmd.CommandText= "delete from authors where au_id='"+TextBox1.Text+"';";
            cmd.Connection = con;
            try
            {
             con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
            Response.Redirect("~/Default.aspx");
        }
        catch (Exception err) {
            Label1.Text = err.Message;
        }
    }
}
```

Write about single valued and repeated valued data binding with examples.

Types of ASP.NET Data Binding:

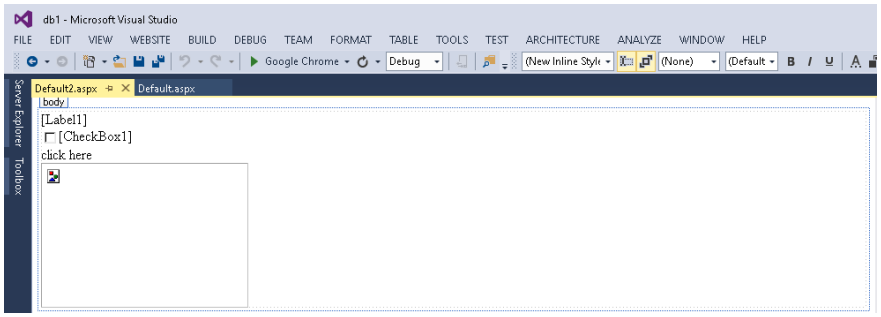
Two types of ASP.NET data binding exist: single-value binding and repeated-value binding. Single-value data binding is by far the simpler of the two, whereas repeated-value binding provides the foundation for the most advanced ASP.NET data controls.

Single-Value, or "Simple," Data Binding

You can use single-value data binding to add information anywhere on an ASP.NET page. You can even place information into a control property or as plain text inside an HTML tag. Single-value data binding doesn't necessarily have anything to do with ADO.NET. Instead, single-value data binding allows you to take a variable, a property, or an expression and insert it dynamically into a page. Single-value binding also helps you create templates for the rich data controls .

Example:

Design:



Default.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default2.aspx.cs"
Inherits="Default2" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:Label ID="Label1" runat="server" Text="">&lt;%#url %></asp:Label>
        <br />
        <asp:CheckBox ID="CheckBox1" runat="server" Text="<%#url %>" />
        <br />
        <asp:HyperLink ID="HyperLink1" runat="server" Text="click here"
NavigateUrl="<%# url %>"></asp:HyperLink>
        <br />
        <asp:Image ID="Image1" runat="server" ImageUrl="<%#url %>" Height="164px"
Width="235px" />
    </div>
    </form>
</body>
</html>
```

Default.aspx.cs:

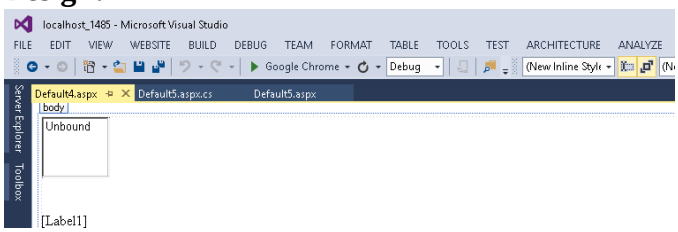
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Default2 : System.Web.UI.Page
{
    protected string url;
    protected void Page_Load(object sender, EventArgs e)
    {
        url = "picture.jpg";
        this.DataBind();
    }
}
```

Repeated-Value, or "List," Binding:

Repeated-value data binding allows you to display an entire table (or just a single field from a table). Unlike single-value data binding, this type of data binding requires a special control

that supports it. Typically, this will be a list control such as CheckBoxList or ListBox, but it can also be a much more sophisticated control such as the GridView (which is described in Chapter 16). You'll know that a control supports repeated-value data binding if it provides a DataSource property. As with single-value binding, repeated-value binding doesn't necessarily need to use data from a database, and it doesn't have to use the ADO.NET objects. For example, you can use repeated-value binding to bind data from a collection or an array.

Design:



Default.aspx:

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

<!DOCTYPE html>

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

        <asp:ListBox ID="ListBox1" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="ListBox1_SelectedIndexChanged"></asp:ListBox>
        <br />
        <br />
        <br />
        <asp:Label ID="Label1" runat="server"></asp:Label>

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

Default.aspx.cs:

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

public partial class Default4 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!this.IsPostBack)
        {
            Dictionary<int, string> fruit = new Dictionary<int, string>();
            fruit.Add(1, "kiwi");
            fruit.Add(2, "Apple");
```

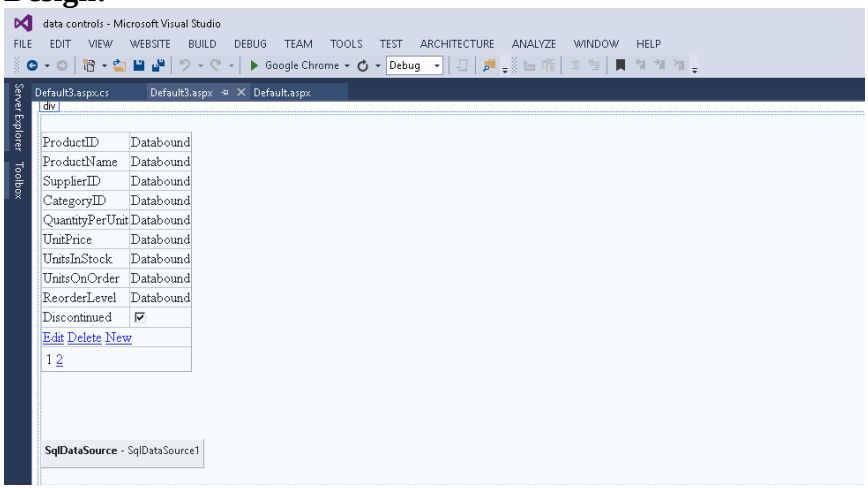
```
            fruit.Add(3, "Banana");
            fruit.Add(4, "Mango");
            fruit.Add(5, "blue berry");
            fruit.Add(6, "pine apple");
            fruit.Add(7, "Apriot");
            fruit.Add(8, "pear");
            fruit.Add(9, "peach");
            ListBox1.DataSource = fruit;
            ListBox1.DataTextField = "value";
            ListBox1.DataValueField = "key";
            this.DataBind();
        }
    }
    protected void ListBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        Label1.Text = "you picked" + ListBox1.SelectedItem.Text+"<br>";
        Label1.Text += "which as the key value :" + ListBox1.SelectedValue;
    }
}
```

Explain about Details View and FormView with example program

DetailsView: The DetailsView is ideal for showing a single record at a time, in a table that has one row per field. The DetailsView also supports editing.

Example:

Design:



Default.aspx.cs:

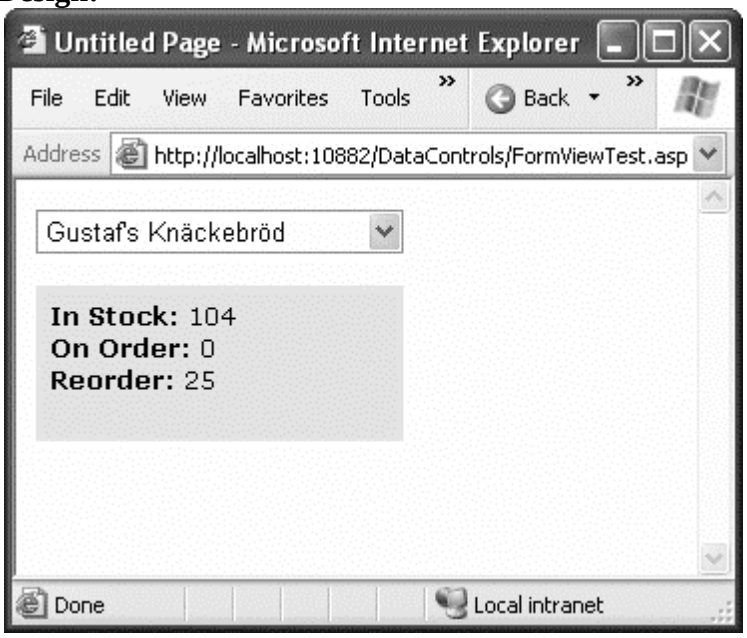
```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default3.aspx.cs"
Inherits="Default3" %>

<!DOCTYPE html>

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

        <br />
        <asp:DetailsView ID="DetailsView1" runat="server" AllowPaging="true"
AutoGenerateRows="false" DataKeyNames="ProductID" DataSourceID="SqlDataSource1"
Height="50px" Width="125px">
            <Fields>
```

```
                <asp:BoundField DataField="ProductID" HeaderText="ProductID"
InsertVisible="False" ReadOnly="True" SortExpression="ProductID" />
                <asp:BoundField DataField="ProductName" HeaderText="ProductName"
SortExpression="ProductName" />
                <asp:BoundField DataField="SupplierID" HeaderText="SupplierID"
SortExpression="SupplierID" />
                <asp:BoundField DataField="CategoryID" HeaderText="CategoryID"
SortExpression="CategoryID" />
                <asp:BoundField DataField="QuantityPerUnit"
HeaderText="QuantityPerUnit" SortExpression="QuantityPerUnit" />
                <asp:BoundField DataField="UnitPrice" HeaderText="UnitPrice"
SortExpression="UnitPrice" />
                <asp:BoundField DataField="UnitsInStock" HeaderText="UnitsInStock"
SortExpression="UnitsInStock" />
                <asp:BoundField DataField="UnitsOnOrder" HeaderText="UnitsOnOrder"
SortExpression="UnitsOnOrder" />
                <asp:BoundField DataField="ReorderLevel" HeaderText="ReorderLevel"
SortExpression="ReorderLevel" />
                <asp:CheckBoxField DataField="Discontinued" HeaderText="Discontinued"
SortExpression="Discontinued" />
                <asp:CommandField ShowDeleteButton="True" ShowEditButton="True"
ShowInsertButton="True" />
            </Fields>
        </asp:DetailsView>
        <br />
        <br />
        <br />
        <br />
        <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:NorthwindConnectionString2 %>" DeleteCommand="DELETE FROM [Products]
WHERE [ProductID] = @ProductID" InsertCommand="INSERT INTO [Products] ([ProductName],
[SupplierID], [CategoryID], [QuantityPerUnit], [UnitPrice], [UnitsInStock],
[UnitsOnOrder], [ReorderLevel], [Discontinued]) VALUES (@ProductName, @SupplierID,
@CategoryID, @QuantityPerUnit, @UnitPrice, @UnitsInStock, @UnitsOnOrder,
@ReorderLevel, @Discontinued)" SelectCommand="SELECT * FROM [Products]"
UpdateCommand="UPDATE [Products] SET [ProductName] = @ProductName, [SupplierID] =
@SupplierID, [CategoryID] = @CategoryID, [QuantityPerUnit] = @QuantityPerUnit,
[UnitPrice] = @UnitPrice, [UnitsInStock] = @UnitsInStock, [UnitsOnOrder] =
@UnitsOnOrder, [ReorderLevel] = @ReorderLevel, [Discontinued] = @Discontinued WHERE
[ProductID] = @ProductID">
            <DeleteParameters>
                <asp:Parameter Name="ProductID" Type="Int32" />
            </DeleteParameters>
            <InsertParameters>
                <asp:Parameter Name="ProductName" Type="String" />
                <asp:Parameter Name="SupplierID" Type="Int32" />
                <asp:Parameter Name="CategoryID" Type="Int32" />
                <asp:Parameter Name="QuantityPerUnit" Type="String" />
                <asp:Parameter Name="UnitPrice" Type="Decimal" />
                <asp:Parameter Name="UnitsInStock" Type="Int16" />
                <asp:Parameter Name="UnitsOnOrder" Type="Int16" />
                <asp:Parameter Name="ReorderLevel" Type="Int16" />
                <asp:Parameter Name="Discontinued" Type="Boolean" />
            </InsertParameters>
            <UpdateParameters>
                <asp:Parameter Name="ProductName" Type="String" />
                <asp:Parameter Name="SupplierID" Type="Int32" />
                <asp:Parameter Name="CategoryID" Type="Int32" />
                <asp:Parameter Name="QuantityPerUnit" Type="String" />
                <asp:Parameter Name="UnitPrice" Type="Decimal" />
                <asp:Parameter Name="UnitsInStock" Type="Int16" />
                <asp:Parameter Name="UnitsOnOrder" Type="Int16" />
```

```
                <asp:Parameter Name="ReorderLevel" Type="Int16" />
                <asp:Parameter Name="Discontinued" Type="Boolean" />
                <asp:Parameter Name="ProductID" Type="Int32" />
            </UpdateParameters>
        </asp:SqlDataSource>
        <br />

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

FormView: Like the DetailsView, the FormView shows a single record at a time and supports editing. The difference is that the FormView is based on templates, which allow you to combine fields in a flexible layout that doesn't need to be table-based.

Design:




```
<asp:SqlDataSource ID="sourceProducts" runat="server"
ConnectionString="<%$ ConnectionStrings:Northwind %>"
SelectCommand="SELECT ProductID, ProductName FROM Products">
</asp:SqlDataSource>
<asp:DropDownList ID="lstProducts" runat="server"
AutoPostBack="True" DataSourceID="sourceProducts"
DataTextField="ProductName" DataValueField="ProductID" Width="184px">
</asp:DropDownList>

<asp:SqlDataSource ID="sourceProductFull" runat="server"
ConnectionString="<%$ ConnectionStrings:Northwind %>"
SelectCommand="SELECT * FROM Products WHERE ProductID=@ProductID">
<SelectParameters>
<asp:ControlParameter Name="ProductID"
ControlID="lstProducts" PropertyName="SelectedValue" />
</SelectParameters>
</asp:SqlDataSource>
<asp:FormView ID="formProductDetails" runat="server"
DataSourceID="sourceProductFull"
```

BackColor="#FFE0C0" CellPadding="5">
<ItemTemplate>
...
</ItemTemplate>
</asp:FormView>

Explain about different data source controls and Write a program to perform insert ,delete and Update using data source control

The data source controls include any control that implements the IDataSource interface. The .NET Framework includes the following data source controls:

• **SqlDataSource**: This data source allows you to connect to any data source that has an ADO.NET data provider. This includes SQL Server, Oracle, and OLE DB or ODBC data sources. When using this data source, you don't need to write the data access code.

• **AccessDataSource**: This data source allows you to read and write the data in an Access database file (.mdb). However, its use is discouraged, because Access doesn't scale well to large numbers of users (unlike SQL Server Express).

• **ObjectDataSource**: This data source allows you to connect to a custom data access class. This is the preferred approach for large-scale professional web applications, but it forces you to write much more code. You'll study the ObjectDataSource in Chapter 22.

• **XmlDataSource**: This data source allows you to connect to an XML file. You'll learn more about XML in Chapter 18.

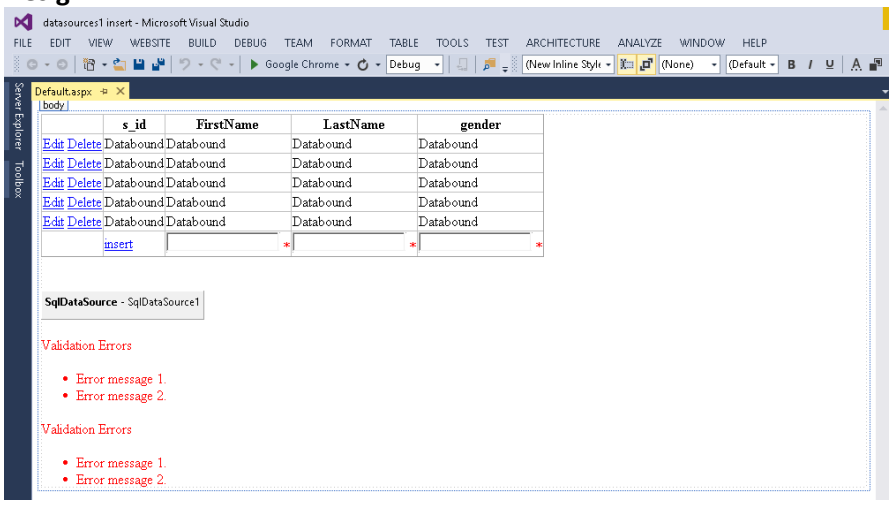
• **SiteMapDataSource**: This data source allows you to connect to a .sitemap file that describes the navigational structure of your website. You saw this data source in Chapter 13.

• **EntityDataSource**: This data source allows you to query a database using the LINQ to Entities feature, which you'll tackle in Chapter 24.

• **LinqDataSource**: This data source allows you to query a database using the LINQ to SQL feature, which is a similar (but somewhat less powerful) predecessor to LINQ to Entities.

Program:

Design:



Default.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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">
    <div>

        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
DataKeyNames="s_id" DataSourceID="SqlDataSource1"  ShowFooter="True" >
            <Columns>
                <asp:CommandField ShowDeleteButton="True" ShowEditButton="True" />
                <asp:TemplateField HeaderText="s_id" SortExpression="s_id">
                    <EditItemTemplate>
                        <asp:Label ID="Label1" runat="server" Text='<%# Eval("s_id")
%>'></asp:Label>
                    </EditItemTemplate>
                    <FooterTemplate>
                        <asp:LinkButton ID="LinkButton1" runat="server"
OnClick="LinkButton1_Click" ValidationGroup="group2">insert</asp:LinkButton>
                    </FooterTemplate>
                    <ItemTemplate>
                        <asp:Label ID="Label1" runat="server" Text='<%# Bind("s_id")
%>'></asp:Label>
                    </ItemTemplate>
                </asp:TemplateField>
                <asp:TemplateField HeaderText="FirstName" SortExpression="FirstName">
                    <EditItemTemplate>
                        <asp:TextBox ID="TextBox1" runat="server" Height="25px"
Text='<%# Bind("FirstName") %>' Width="128px" ValidationGroup="group1"></asp:TextBox>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server" ControlToValidate="TextBox1" ErrorMessage="f_name should be entered"
ForeColor="Red" ValidationGroup="group1">*</asp:RequiredFieldValidator>
                    </EditItemTemplate>
                    <FooterTemplate>
                        <asp:TextBox ID="TextBox4" runat="server" Height="16px"
ValidationGroup="group2"></asp:TextBox>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator2"
runat="server" ControlToValidate="TextBox4" ErrorMessage="F_name must be entered"
ForeColor="Red" ValidationGroup="group2">*</asp:RequiredFieldValidator>
                    </FooterTemplate>
                    <ItemTemplate>
                        <asp:Label ID="Label2" runat="server" Text='<%#
Bind("FirstName") %>'></asp:Label>
                    </ItemTemplate>
                </asp:TemplateField>
                <asp:TemplateField HeaderText="LastName" SortExpression="LastName">
                    <EditItemTemplate>
                        <asp:TextBox ID="TextBox2" runat="server" Text='<%#
Bind("LastName") %>' ValidationGroup="group1"></asp:TextBox>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator3"
runat="server" ControlToValidate="TextBox2" ErrorMessage="L_name Must Be Entered"
ForeColor="Red" ValidationGroup="group1">*</asp:RequiredFieldValidator>
                    </EditItemTemplate>
                    <FooterTemplate>
                        <asp:TextBox ID="TextBox5" runat="server"
ValidationGroup="group2"></asp:TextBox>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator4"
runat="server" ControlToValidate="TextBox5" ErrorMessage="L_name Must be entered"
ForeColor="Red" ValidationGroup="group2">*</asp:RequiredFieldValidator>
                    </FooterTemplate>
                    <ItemTemplate>
```

```
                        <asp:Label ID="Label3" runat="server" Text='<%#
Bind("LastName") %>'></asp:Label>
                    </ItemTemplate>
                </asp:TemplateField>
                <asp:TemplateField HeaderText="gender" SortExpression="gender">
                    <EditItemTemplate>
                        <asp:TextBox ID="TextBox3" runat="server" Text='<%#
Bind("gender") %>' ValidationGroup="group1"></asp:TextBox>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator4"
runat="server" ControlToValidate="TextBox3" ErrorMessage="Gender Must Be Entered"
ForeColor="Red" ValidationGroup="group1">*</asp:RequiredFieldValidator>
                    </EditItemTemplate>
                    <FooterTemplate>
                        <asp:TextBox ID="TextBox6" runat="server"
ValidationGroup="group2"></asp:TextBox>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator6"
runat="server" ControlToValidate="TextBox6" ErrorMessage="Gender Must Be Entered"
ForeColor="Red" ValidationGroup="group2">*</asp:RequiredFieldValidator>
                    </FooterTemplate>
                    <ItemTemplate>
                        <asp:Label ID="Label4" runat="server" Text='<%# Bind("gender")
%>'></asp:Label>
                    </ItemTemplate>
                </asp:TemplateField>
            </Columns>
        </asp:GridView>
        <br />
        <br />
        <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:masterConnectionString %>" InsertCommand="INSERT INTO [prakash]
([FirstName], [LastName], [gender]) VALUES (@FirstName, @LastName, @gender)"
SelectCommand="SELECT * FROM [prakash]" DeleteCommand="DELETE FROM [prakash] WHERE
[s_id] = @s_id" UpdateCommand="UPDATE [prakash] SET [FirstName] = @FirstName,
[LastName] = @LastName, [gender] = @gender WHERE [s_id] = @s_id">
            <DeleteParameters>
                <asp:Parameter Name="s_id" Type="Int32" />
            </DeleteParameters>
            <InsertParameters>
                <asp:Parameter Name="FirstName" Type="String" />
                <asp:Parameter Name="LastName" Type="String" />
                <asp:Parameter Name="gender" Type="String" />
            </InsertParameters>
            <UpdateParameters>
                <asp:Parameter Name="FirstName" Type="String" />
                <asp:Parameter Name="LastName" Type="String" />
                <asp:Parameter Name="gender" Type="String" />
                <asp:Parameter Name="s_id" Type="Int32" />
            </UpdateParameters>
        </asp:SqlDataSource>

        <br />
        <asp:ValidationSummary ID="ValidationSummary1" runat="server" ForeColor="Red"
HeaderText="Validation Errors" ShowMessageBox="True" ValidationGroup="group1" />

    </div>
        <asp:ValidationSummary ID="ValidationSummary2" runat="server" ForeColor="Red"
HeaderText="Validation Errors" ShowMessageBox="True" ValidationGroup="group2" />
    </form>
</body>
</html>
```

Default.aspx.cs:

```
using System;
```

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

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

    }
    protected void LinkButton1_Click(object sender, EventArgs e)
    {
        SqlDataSource1.InsertParameters["FirstName"].DefaultValue =
((TextBox)GridView1.FooterRow.FindControl("TextBox4")).Text;
        SqlDataSource1.InsertParameters["LastName"].DefaultValue =
((TextBox)GridView1.FooterRow.FindControl("TextBox5")).Text;
        SqlDataSource1.InsertParameters["gender"].DefaultValue =
((TextBox)GridView1.FooterRow.FindControl("TextBox6")).Text;
        SqlDataSource1.Insert();

    }

}
```

Expalin about LINQ object and LINQ to SQL with examples.

LINQ to Objects: This is the simplest form of LINQ. It allows you to query collections of inmemory objects. LINQ return an usual type of object, called an iterator object. Although the iterator object looks like an ordinary collection to your code, it doesn't hold any information.LINQ evaluates your expression and quickly grabs the information you need. This trick is called deferred execution using System;

```
using
System.Collections.Ge
neric; using
System.Linq; using
System.Web; using
System.Web.UI;

using System.Web.UI.WebControls;


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

{

   protected void Page_Load(object sender, EventArgs e)

   {

      List<Employee> employees = new List<Employee>();
employees.Add(new Employee(1, "Nancy", "Davolio", "Ms."));
employees.Add(new Employee(2, "Andrew", "Fuller", "Dr."));
employees.Add(new Employee(3, "Janet", "Leverling", "Ms."));
```

```
    var matches = from employee in
employees                where
employee.LastName.StartsWith("D")
select employee;

    gridEmployees.DataSource = matches;
gridEmployees.DataBind();



  }

  public class Employee

  {

    public int EmployeeID { get;
set; }      public string FirstName {
get; set; }      public string
LastName { get; set; }      public
string TitleOfCourtesy { get; set; }

    public Employee(int employeeID, string firstName, string
lastName,      string titleOfCourtesy)

    {

      EmployeeID = employeeID;

      FirstName = firstName;

      LastName = lastName;

      TitleOfCourtesy = titleOfCourtesy;

    }
```

Linq to sql:

LINQ (Language Integrated Query)

what is LINQ

LINQ enables us to query any type of data store (SQLServer, XML documents, objects in memory etc)

why should we use LINQ and benifits:

If the .NET application that is being developed

a) Requires data from SQLserver - Then the developer has to understand ADO.NET code and SQL specific to SQL Server Database

b) Requires data from xml document - Then the developer has to understand XSLT & XPATH queries

c) need to query objects in memory (List<customer>, List<order> etc) - then the developer has to understand how to work with objects in memory

LINQ enables us to work with these different data sources using a similar coding style without having the need to know syntax specific to data source.

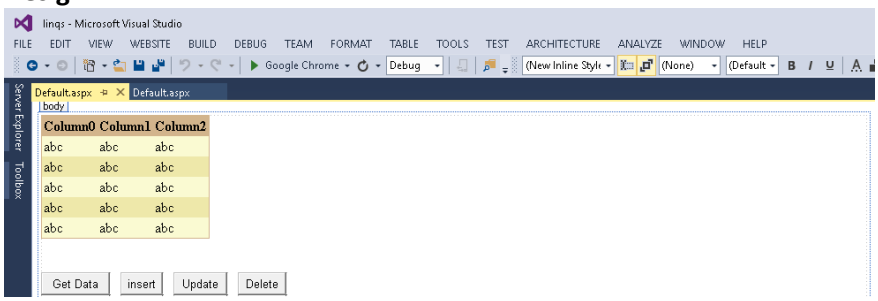
Another benefit of using LINQ is that it provides intellisense and compile time error checking.

LINQ Architecture & LINQ providers

1. LINQ Query can be written ~~between~~ using any .NET supported programming language
2. LINQ provider is a component between the LINQ query and actual data source which converts the LINQ query into a format that underlying data source can understand.

for example LINQ to SQL provider converts a LINQ query to T-SQL that Server database can understand

Design:



Default.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.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">
    <div>

        <asp:GridView ID="GridView1" runat="server" BackColor="LightGoldenrodYellow"
BorderColor="Tan" BorderWidth="1px" CellPadding="2" ForeColor="Black"
GridLines="None">
            <AlternatingRowStyle BackColor="PaleGoldenrod" />
            <FooterStyle BackColor="Tan" />
            <HeaderStyle BackColor="Tan" Font-Bold="True" />
            <PagerStyle BackColor="PaleGoldenrod" ForeColor="DarkSlateBlue"
HorizontalAlign="Center" />
            <SelectedRowStyle BackColor="DarkSlateBlue" ForeColor="GhostWhite" />
            <SortedAscendingCellStyle BackColor="#FAFAE7" />
            <SortedAscendingHeaderStyle BackColor="#DAC09E" />
            <SortedDescendingCellStyle BackColor="#E1DB9C" />
            <SortedDescendingHeaderStyle BackColor="#C2A47B" />
        </asp:GridView>
        <br />
        <br />
        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Get
Data" />
  
        <asp:Button ID="Button2" runat="server" OnClick="Button2_Click" Text="insert"
/>
  
        <asp:Button ID="Button3" runat="server" OnClick="Button3_Click" Text="Update"
/>
  
        <asp:Button ID="Button4" runat="server" OnClick="Button4_Click" Text="Delete"
/>
```

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

Default.aspx.cs:

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

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

    }
    protected void getdata()
    {
        DataClassesDataContext datacontext = new DataClassesDataContext();
        GridView1.DataSource = from student in datacontext.students select student;
        GridView1.DataBind();
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        getdata();
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        DataClassesDataContext dc = new DataClassesDataContext();
        prakash s = new prakash
        {
            FirstName = "tim",
            LastName="t",
            gender = "Male"
        };
        dc.prakashes.InsertOnSubmit(s);
        dc.SubmitChanges();
        getdata();
    }
    protected void Button3_Click(object sender, EventArgs e)
    {
        DataClassesDataContext dc = new DataClassesDataContext();
        var m = from s in dc.students where s.s_id == 2 select s;
        prakash stu = m.Single();
        stu.gender = "male";
        dc.SubmitChanges();
        getdata();
    }
    protected void Button4_Click(object sender, EventArgs e)
    {
        DataClassesDataContext dc = new DataClassesDataContext();
        var m = from s in dc.students where s.s_id == 10 select s;
        prakash stu = m.Single();
        dc.prakashes.DeleteOnSubmit(stu);
        dc.SubmitChanges();
        getdata();
    }
}
```

UNIT-IV

8a) Explain in detail about MVC. (6M)

In high –level terms ,the MVC pattern means that an MVC application will be split into atleast three pieces:

Models, which contain or represent data that user works with. These can be simple view models, which just represent data being transferred between views and controllers; or they can be domain models, which contain the data in a business domain as well as the operations, transformations and rules from manipulating that data.

Views, which are used to render some part of the model as a user interface.

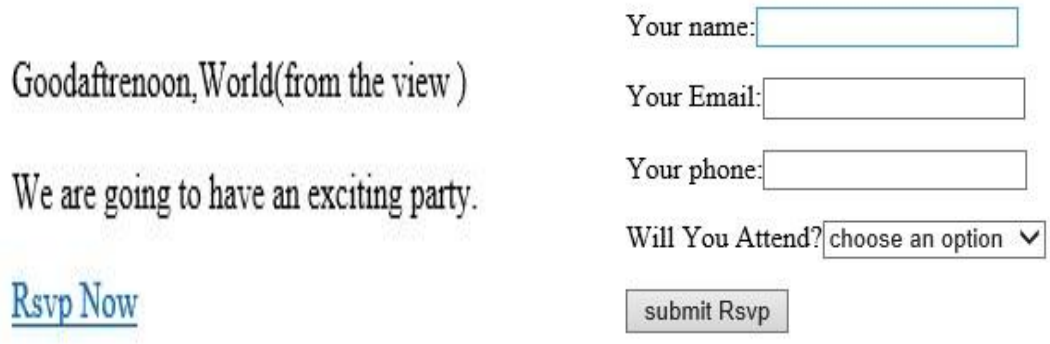
Controllers, which process incoming requests, perform operations on the model, and select views to render to the user .

Models are the definition of the universe your application works in. controllers. Views contain the logic required to display elements of the models to the user and nothing more. They have no direct awareness of the model and do not directly communicate with the model in any way.

Controllers, are the bridge between views and the model requests come in from the client and are serviced by the controller, which selects an appropriate view to show the user and ,if required ,an appropriate operation to perform on the model.

Each piece of the MVC architecture is well defined and self contained—this is referred to as the *separation of concerns.* The logic that manipulates the data in the model is contained only in the model; the logic that displays the data is only in the view, and the code that handles user requests and input is contained only in controller. With a clear division between each of the pieces , your application will be easier to maintain and extend over its lifetime, no matter how large it becomes.

8b) Create a simple application using MVC architecture. (6M)



HOMECONTROLLER.CS

```
using
partyinvites.Models;
using System; using
System.Collections.Ge
neric; using
System.Linq; using
System.Web; using
System.Web.Mvc;

namespace partyinvites.Controllers

{

   public class HomeController : Controller

   {

      public ViewResult Index()

      {

         int hour = DateTime.Now.Hour;

         ViewBag.Greeting = hour < 12 ? "Goodmorning" : "Goodaftrenoon";
return View();

      }

      [HttpGet]
public ViewResult
Rsvpform()

      {

         return View();

      }

      [HttpPost]

      public ViewResult Rsvpform(GuestResponse guestResponse)

      {

         if(ModelState.IsValid)

         {

            return View("thanks", guestResponse);

 }

else

         {

            return View();
```

```
        }
      }
   }
}
```

MODELS

GUESTRESPONSE.CS

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

namespace PartyInvites.Models
{
   public class GuestResponse
   {
      [Required(ErrorMessage="please enter your naame")]        public String name { get; set;}
      [Required(ErrorMessage="please enter mail id")]
      [Regular Expression("+||.+||..+") ErrorMessage="please enter valid email id"]
public String email { get; set; }
      [Required(ErrorMessage = "please enter your number")]
      public String phone { get; set; }
      [Required(ErrorMessage="please specify whether you will attend")]
public bool willattend { get; set; }
   }
}
```

INDEX:

```
@{
   Layout = null;
}
<!DOCTYPE html>
<html>
   <head>
```

```
    <meta name="viewport" content="width=device-width"/>
    <title>Index</title>
  </head>
  <body>
    <div>
      @ViewBag.Greeting,World(from the view )
      <p>We are going to have an exciting party.<br/>
      </p>
      @Html.ActionLink("Rsvp Now","RsvpForm")
    </div>
  </body>
</html>
```

RSVPFORM

```
@model partyinvites.Models.GuestResponse
@{
  Layout = null;
}
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>RsvpForm</title>
</head>
<body>
  @using (Html.BeginForm())
  {
    @Html.ValidationSummary()
    <p>Your name:@Html.TextBoxFor(x => x.name)</p>
    <p>Your Email:@Html.TextBoxFor(x => x.Email)</p>
    <p>Your phone:@Html.TextBoxFor(x => x.phone)</p>
```

```
    <p>
        Will You Attend?@Html.DropDownListFor(x => x.WillAttend, new[]
            {
                new SelectListItem(){Text="Yes i'll be
there",Value=bool.TrueString},          new SelectListItem(){Text="no,i can
not come",Value=bool.FalseString}          }, "choose an option")
    </p>
    <input type="submit" value="submit Rsvp" />
  }
</body>
</html>
```

THANKS:

```
@model partyinvites.Models.GuestResponse
@{
  Layout = null;

}
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>Thanks</title>
</head>
<body>
  <div>
    <h1>Thankyou,@Model.name!</h1>
    @if (Model.WillAttend == true)
    {
      @:Its great that you are coming.The drinks are already in the fridge.
  }
else    {
      @:sorry to hear that you can not make,but thanks for letting us know.
```

```
    }
  </div>
</body>
</html>
```

9a) Design an application to demonstrate creating and consuming a WCF web service. (6M)

HELLOSERVICE.CS

```
using System;

using
System.Collections.Gene
ric; using System.Linq;
using
System.Runtime.Serializa
tion; using
System.ServiceModel;
using System.Text;


namespace HelloServices

{
  // NOTE: You can use the "Rename" command on the "Refactor" menu to change the
class name "HelloService" in both code and config file together.
  public class HelloService : IHelloService
  {
    public String GetMessage(String name)
    {
      return "hello" + name;
    }
  }
}
```

APP.CONFIG FILE:

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

```
<configuration>
   <startup>
      <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.2" />
</startup>
  <system.serviceModel>
   <behaviors>
    <serviceBehaviors>
     <behavior name="mexBehaviour">
      <serviceMetadata httpGetEnabled="true"/>
     </behavior>
    </serviceBehaviors>
   </behaviors>
   <services>
    <service name="HelloService.HelloService" behaviorConfiguration="mexBehaviour">
     <endpoint address="HelloService" binding="basicHttpBinding"
contract="HelloService.IHelloService">
     </endpoint>
     <endpoint address="HelloService" binding="netTcpBinding"
contract="HelloService.IHelloService">
     </endpoint>
     <endpoint address="mex" binding="mexHttpBinding"
contract="IMetadataxchange"/>
     <host>
      <baseAddresses>
       <add baseAddress="http://localhost:8080/"/>
       <add baseAddress="http://localhost:8090/"/>
      </baseAddresses>
     </host>
    </service>
   </services>
  </system.serviceModel>
</configuration>
```

PROGRAM.CS

```
using System;

using
System.Collections.Ge
neric; using
System.Linq; using
System.Text; using
System.Threading.Task
s;

using System.Security;


namespace HelloServiceHost

{

   class Program

   {

      static void Main(string[] args)

      {

         using (ServiceHost host = new ServiceHost(typeof(HelloServiceHost.HelloService))

         {

            host.Open();

         Console.WriteLine("host started @" + DateTime.Now.ToString());
Console.ReadLine();

         }

      }

}
```

CLIENT SIDE PROGRAMMMING

HELLOSERVICECLIENT.ASPX

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

<!DOCTYPE html>


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

<head runat="server">

   <title></title>
```

```
</head>
<body>
   <form id="form1" runat="server">
   <div style="font-weight: 700">

      &amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
&amp;nbsp;&amp;nbsp;&amp;nbsp;
      <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
      <br />
      <br />
&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;
      <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Button" />

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

HELLOSERVICEHOST.ASPX.CS

```
using System;
using
System.Collections.Ge
neric; using
System.Linq; using
System.Web; using
System.Web.UI;
using System.Web.UI.WebControls;

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

   }
```

```
    protected void Button1_Click(object sender, EventArgs e)
    {
            HelloService.HelloServiceClient client = new
            HelloService.HelloServiceClient("BasicHttpBinding_IHelloService");
Label1.Text = client.GetMessage(TextBox1.Text);
    }
}
```

9b) Explain deployment of an ASP.NET application with simple example . (6M)

Visual Studio has several deployment-related features:

- You can create a virtual directory when you create a new project.
- You can use the Copy Web Site feature to transfer an existing website to a virtual directory.
- You can use the Publish Web Site feature to compile your website and transfer it to another location.
- If you use web projects (not projectless websites), you can use the web package feature to bundle IIS settings, security certificates, and SQL scripts, with the actual files of your application, into a single convenient package.

Deploying with Visual Studio:

i) You can create a virtual directory when you create a new project

ii) You can use the Copy Web Site feature to transfer an existing website to a virtual directory

iii) You can use the Publish Web Site feature to compile your website and transfer it to another location.

iv) If you use web projects (not projectless websites), you can use the web package feature to bundle IIS settings, security certificates, and SQL scripts, with the actual files of your application, into a single convenient package.