ASP.Net
Lab Book

Document Revision History

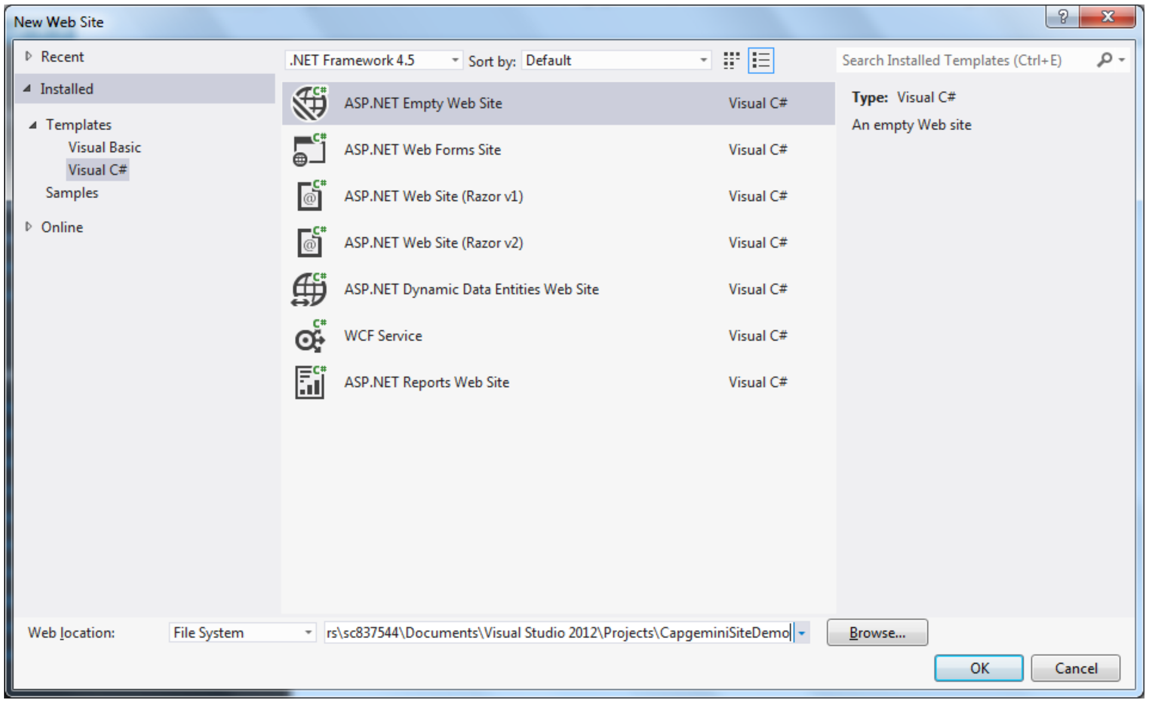
Date	Revision No.	Author	Summary of Changes
22-June-2011	1	Ajit Jog	Content Creation as per integration process
16-May-2016	2	Sangeetha	Revamping as per the Integration process

Table of Contents

Lab 1.Simple Event Handling using asp.net controls

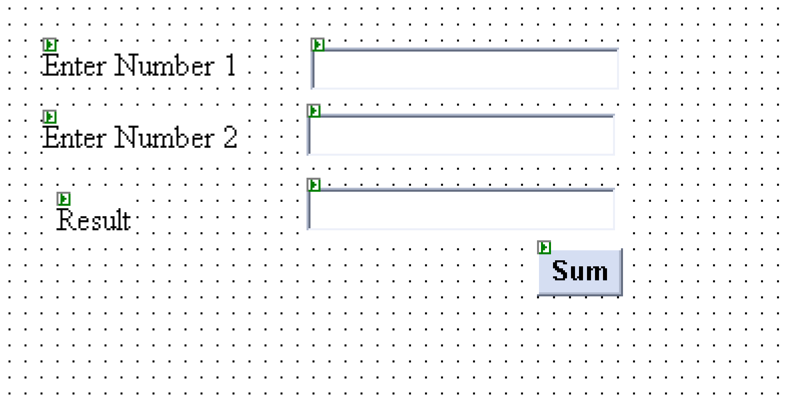
Description	• In this lab we will be displaying arithmetic calculation result using asp.net controls
Objective	To learn: • How to design a page with asp.net controls • Handling server side events for asp.net controls
Time	30 Min

1. Start a New Web Site Application
    a. File => New Website, select asp.net website template, location as Filesystem, language Visual C# and specify physical location for website. Give Name "**CapgeminiSiteDemo**" to the site.


New Web Site
Recent
Installed
Templates
Visual Basic
Visual C#
Samples
Online
.NET Framework 4.5
Sort by: Default
Search Installed Templates (Ctrl+E)
ASP.NET Empty Web Site Visual C#
ASP.NET Web Forms Site Visual C#
ASP.NET Web Site (Razor v1) Visual C#
ASP.NET Web Site (Razor v2) Visual C#
ASP.NET Dynamic Data Entities Web Site Visual C#
WCF Service Visual C#
ASP.NET Reports Web Site Visual C#
Type: Visual C#
An empty Web site
Browse...
OK
Cancel

Web location: File System — rs\sc837544\Documents\Visual Studio 2012\Projects\CapgeminiSiteDemo

2. Design the aspx page as below:
   Asp.net controls: Labels (3), TextBoxes (3) and Button (1)
   Name the controls appropriately:
   TextBoxes: txtnumber1, txtnumber2, txtresult
   Button: btnsum



3. Code for Sum Button:

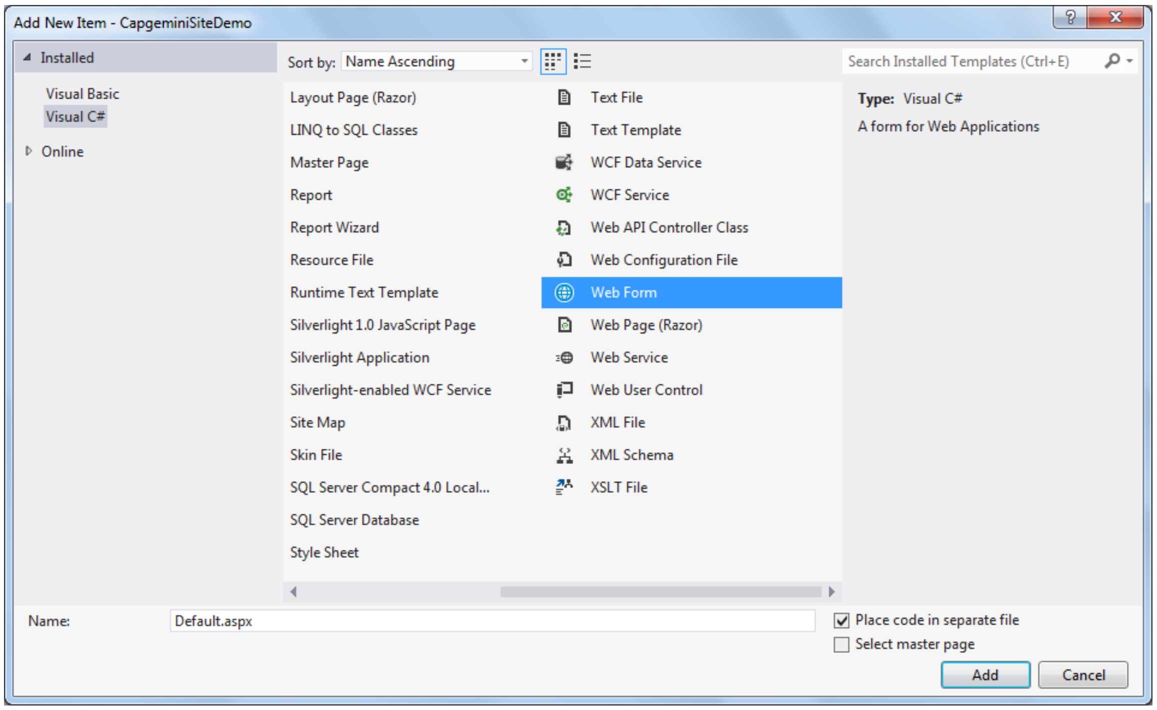
```
private void btnsum_Click(object sender, System.EventArgs e)
 {
        txtresult.Text = System.Convert.ToInt32(txtnumber1.Text) +
                                                System.Convert.ToInt32(txtnumber2.Text);
 }
```

Note: That every time you click button, the entire page is submitted to server and a new page output is sent with certain changes due to event code. The earlier page is visible if you use “back” button of your browser.

Lab 2.Working with Post Back Concept and a HTML control

Description	• In this lab we will be making use of post back related properties ie IsPostback and AutoPostback. We will also see about using an client side plain HTML control.
Objective	To learn: • Post back concept of asp.net • Using an plain HTML control and client side event handling
Time	90 Mins

1. Add a new Web Page
   a. Website => Add New Item
      i. Select Web Form Template, Name it: PostBackDemo.aspx, click add

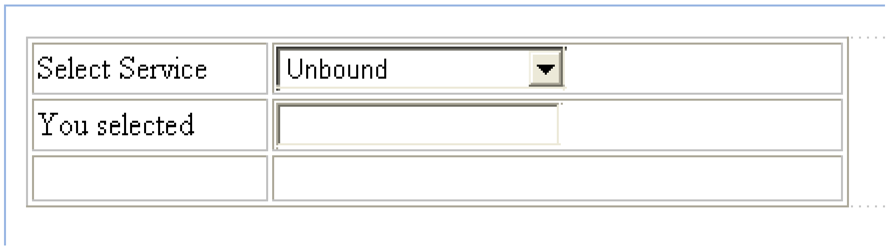

Add New Item - CapgeminiSiteDemo

Type: Visual C#
A form for Web Applications

Name: Default.aspx

☑ Place code in separate file
☐ Select master page

2. Add a HTML Table ot 3 rows 2 columns
   a. Table => Insert Table
      i. specify 3 rows and 2 columns
3. Drag Labels (2), TextBox (1) (txtservice), Dropdown (1) (dnservicelist)



Select Service	Unbound
You selected	

4. In the code behind add the following page load code:

```
protected void Page_Load(object sender, EventArgs e)
  {
                //Add Items to Dropdown
                dnservicelist.Items.Add("Airtel");
                dnservicelist.Items.Add("Vodafone");
               //Different Text and Value
                dnservicelist.Items.Add(new ListItem("BPL", "Loop"));
  }
```

5. Make this Page as Default
   a. In Solution Explorer right click PostBackDemo web page and select “Set as Startup”
6. Run the web page (CTRL+F5 OR simply F5), if dialog box for debugging comes up say ok with default selection.
7. if you change drop down selection there will be no response.
8. close the browser , set autopostback property of dropdown to true and double click drop down and write the following event code:

```
 protected void dnservicelist_SelectedIndexChanged(object sender, EventArgs e)
{
        txtservice.Text = dnservicelist.SelectedValue;
}
```

9. Run the web page and change drop down selection the current selection will be displayed in textbox

There will be one problem; that every time we change the drop down selection, the page submits and the page load event fires and re adds the options to drop down. Moreover the earlier entries of drop down are remembered by the page due to “**view state**” feature.

To solve it we will make use of property IsPostBack in page load as follows

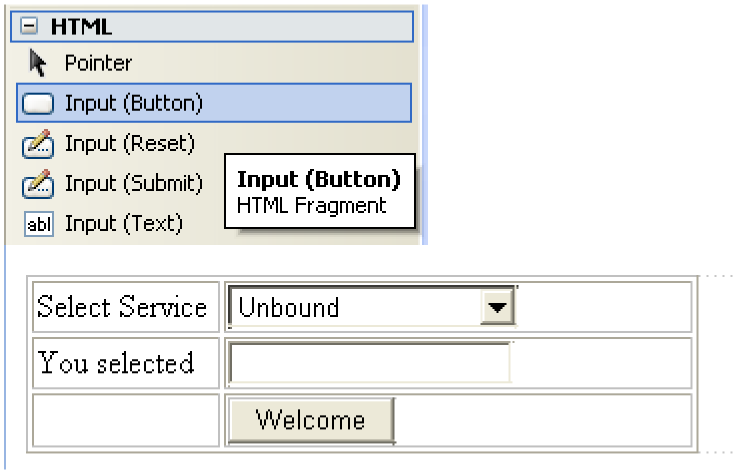
10. Change the page load code:

```
  //ie: when page executes for the first time, or there is no postback ...
  if (this.IsPostBack == false)
  {
        //Add Items to Dropdown
        dnservicelist.Items.Add("Airtel");
        dnservicelist.Items.Add("Vodafone");
        //Different Text and Value
        dnservicelist.Items.Add(new ListItem("BPL", "Loop"));
    }
```

11. Run the web page and now the drop down will be populated only once even if you change the selection.
12. Let us check how the view state (visual state of page) is maintained by asp.net page
    a. Run the page and go to view source view => source OR right click web page in free space and view source.
13. In the actual page response find a hidden variable "__viewstate" that is the storage are of entire view state.

<u>Working with a HTML Button Control</u>

14. From HTML tab in toolbox drag a HTML Button on the web page in the 3 rd row in the HTML table.



15. Set the following properties of the HTML Button
    a. id Property : btnwelcome
    b. value Property: Welcome
16. double click the btnwelcome button to generate client side JavaScript event handler and write the event code:

```
<script language="javascript" type="text/javascript">
// <!CDATA[

    function btnwelcome_onclick() {
        window.alert('Welcome to Asp.Net  !!!');
        }

    // ]]>
           </script>
```

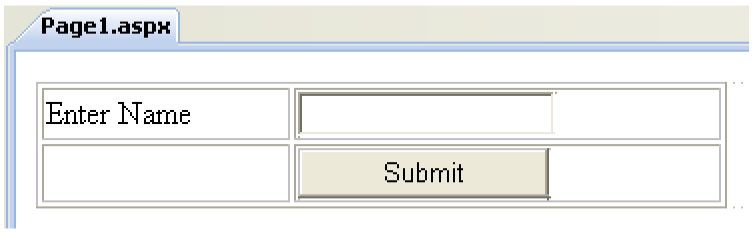
17. Run the page and click welcome button.

Note: The Post back (page submit) does not occur in this case. i.e.: There is no server trip.

Lab 3.Working with Cross Page Post Back

Description	• In this lab we will use CrossPagePostback feature to submit the data of one page to another.
Objective	To learn: • How to use CrossPagePostBack feature to submit the data to another page • How to identify whether page is executing due to self post back or cross post back
Time	30 Mins

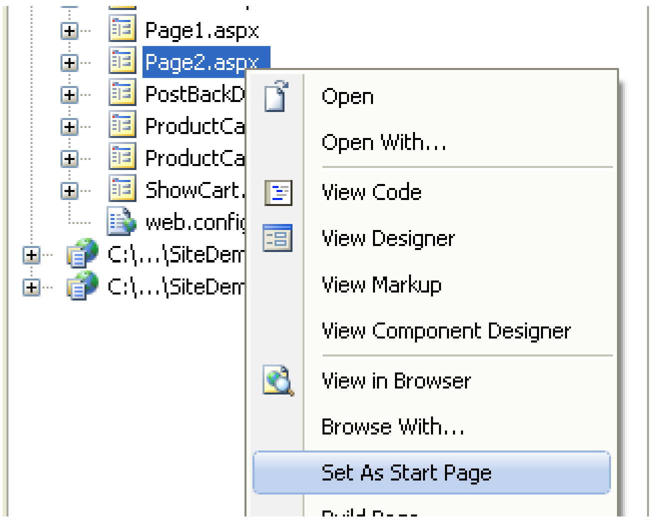
1. Add a new web Page Page1.aspx
2. Add a HTML Table of 2 rows and 2 columns
    a. Table => Insert Table.
3. Drag a Label, TextBox (txtname) and Button (btnsubmit)
    a. Set Text property of label to “Enter Name”



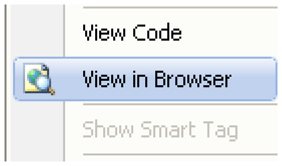
4. Add one more page Page2.aspx and drag just one label on to page.
5. In Page2.aspx.cs code behind write the following page load code:

```
protected void Page_Load(object sender, EventArgs e)
   {
      if (PreviousPage != null && PreviousPage.IsCrossPagePostBack)
      {
         TextBox txtbox = PreviousPage.FindControl("txtname") as TextBox;
         if (txtbox != null)
         {
            Label1.Text = "Welcome User " + txtbox.Text;
         }
         else
            Label1.Text = "No txtname Text Box found in earlier page";
      }
      else
         Label1.Text = "You directly came to this Page2.aspx and not through Page1.aspx";
   }
```

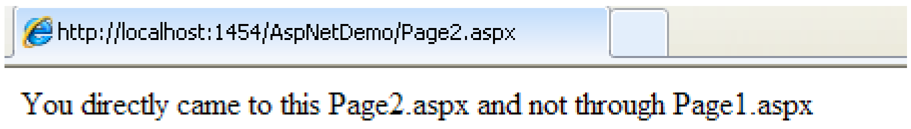
6. In Page1.aspx go to Button properties
    a. locate postbackurl property and from ellipsis option select Page2.aspx
7. Make Page2.aspx



8. Run the application Ctrl+F5 or F5. Alternatively you can right click the Page2.aspx in free are and select View in Browser
9.



10. You should get the output as shown:



http://localhost:1454/AspNetDemo/Page2.aspx

You directly came to this Page2.aspx and not through Page1.aspx

11. Now make Page1.aspx as startup and run the web application
12. Enter Name and click submit
    a. You will be navigated to Page2.aspx with a welcome message there.

Lab 4.Understanding Page Life Cycle

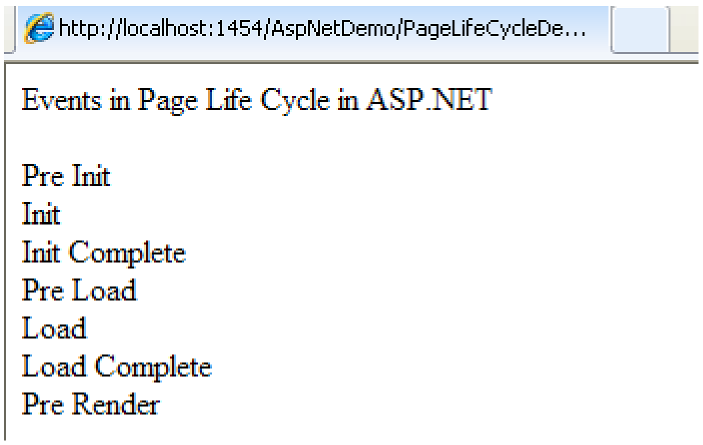
Description	• In this lab we will handle the major page lifecycle events to know their order of execution.
Objective	To learn: • Some of the major page lifecycle events available and • To know their order of execution
Time	30 Mins

1. Add a new Web Page, Name it PageLifeCycleDemo.aspx
2. Drag a single Label on to it.
3. Go to Code behind and write the code for the following page events as shown below:

```
public partial class PageLifeCycleDemo : System.Web.UI.Page
{
   String msg = "Events in Page Life Cycle in ASP.NET" + "<BR>";

   protected void Page_PreInit(object sender, EventArgs e)
   {
      msg = msg + "<BR>" + "Pre Init";
   }

   protected void Page_Init(object sender, EventArgs e)
   {
      msg = msg + "<BR>" + "Init";
   }

   protected void Page_InitComplete(object sender, EventArgs e)
   {
      msg = msg + "<BR>" + "Init Complete";
   }

   protected void Page_PreLoad(object sender, EventArgs e)
   {
      msg = msg + "<BR>" + "Pre Load";
   }

   protected void Page_LoadComplete(object sender, EventArgs e)
   {
      msg = msg + "<BR>" + "Load Complete";
   }

   protected void Page_Load(object sender, EventArgs e)
   {
      msg = msg + "<BR>" + "Load";
   }

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

```
    {
        msg = msg + "<BR>" + "Pre Render";
        Label1.Text = msg;
    }

}
```

4. Note that we displaying the final message in pre-render event. since that is the last event that is called before rendering the page.
5. Run this page. You should get output as below:

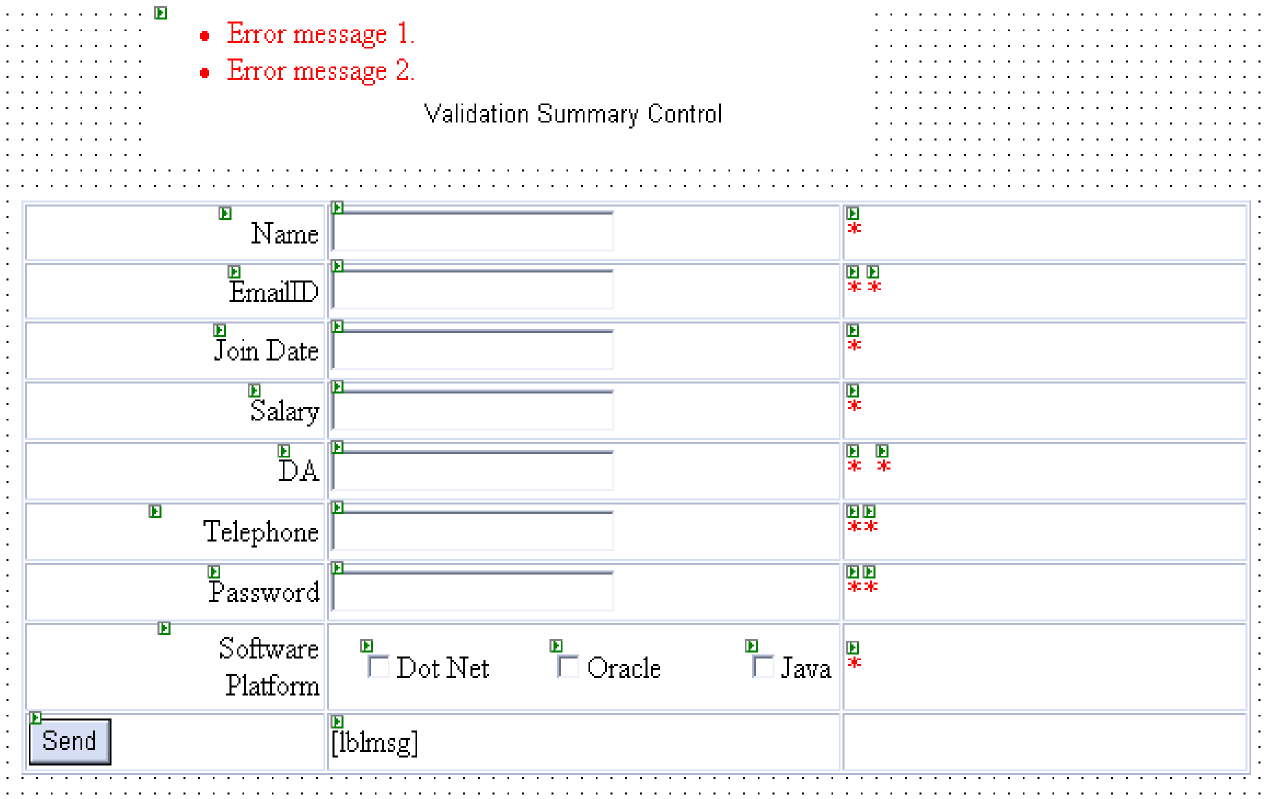


```
Events in Page Life Cycle in ASP.NET

Pre Init
Init
Init Complete
Pre Load
Load
Load Complete
Pre Render
```

Lab 5.Working with built-in asp.net validation controls

Description	• In this lab we will be having a data entry asp.net page, where we will accept details from user and validate it at client side using various built-in asp.net validation controls before it is submitted to the web server.
Objective	To learn: • Various built-in asp.net validation controls available • Using built-in asp.net validation controls as per requirement
Time	60 Mins

1. Add a new aspx web page, name it validationdemopage.aspx
2. Design the page as shown below: (all are asp.net server controls)
    a. There is an HTML table with 9 rows & 3 columns
    b. In the 1st column there are labels, 2nd column has textboxes & 3rd column has validation controls dragged.
    c. For accepting software platform there are 3 checkboxes
    d. The last row has a send button and a additional label called lblmsg.
    e. additionally add a validation summary control (above or below HTML table)

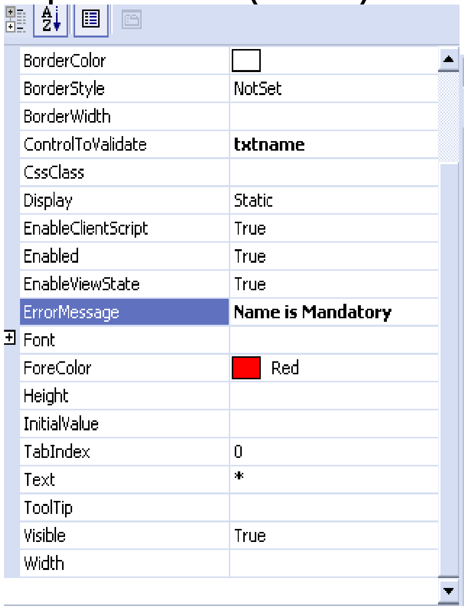


3. Drag the following validation controls as given below in the corresponding row in the 3rd column.

Note:
Every validation control has certain properties specific to itself. for e.g.: validationexpression property is available to only regularexpression validation control.

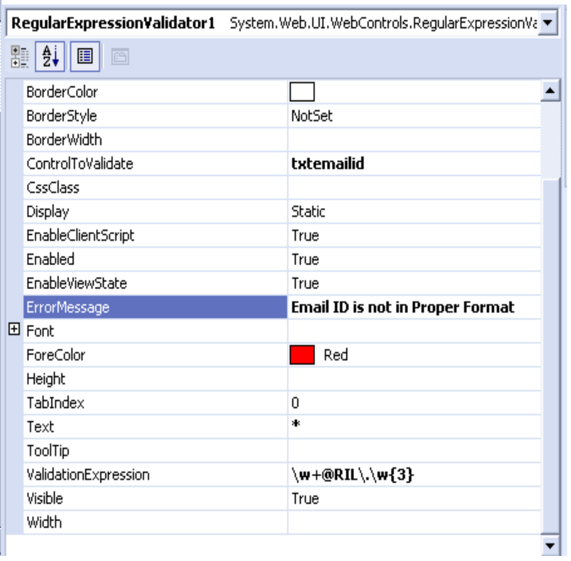
There are some common properties all validation controls i.e.: ControlToValidate, ErrorMessage, Text which are set.

Required Validator (txtname)



BorderColor	□
BorderStyle	NotSet
BorderWidth	
ControlToValidate	**txtname**
CssClass	
Display	Static
EnableClientScript	True
Enabled	True
EnableViewState	True
ErrorMessage	**Name is Mandatory**
Font	
ForeColor	🟥 Red
Height	
InitialValue	
TabIndex	0
Text	*
ToolTip	
Visible	True
Width	

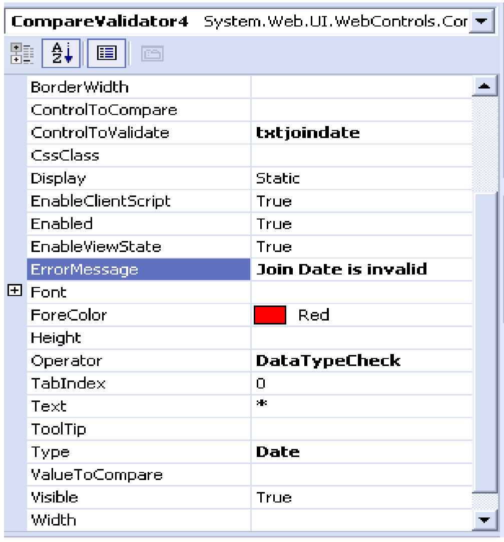
Reg Expr Validator (txtemailid)



RegularExpressionValidator1 System.Web.UI.WebControls.RegularExpressionVa

BorderColor	□
BorderStyle	NotSet
BorderWidth	
ControlToValidate	**txtemailid**
CssClass	
Display	Static
EnableClientScript	True
Enabled	True
EnableViewState	True
ErrorMessage	**Email ID is not in Proper Format**
Font	
ForeColor	🟥 Red
Height	
TabIndex	0
Text	*
ToolTip	
ValidationExpression	**\w+@RIL\.\w{3}**
Visible	True
Width	

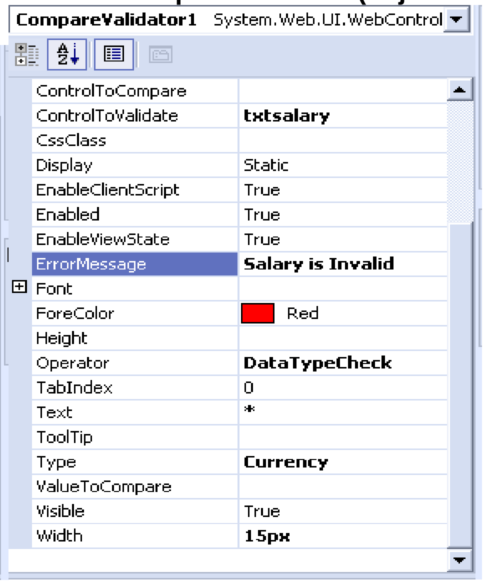
compare validator (txtsalary)



CompareValidator4 System.Web.UI.WebControls.Cor

BorderWidth	
ControlToCompare	
ControlToValidate	**txtjoindate**
CssClass	
Display	Static
EnableClientScript	True
Enabled	True
EnableViewState	True
ErrorMessage	**Join Date is invalid**
Font	
ForeColor	🟥 Red
Height	
Operator	**DataTypeCheck**
TabIndex	0
Text	*
ToolTip	
Type	**Date**
ValueToCompare	
Visible	True
Width	

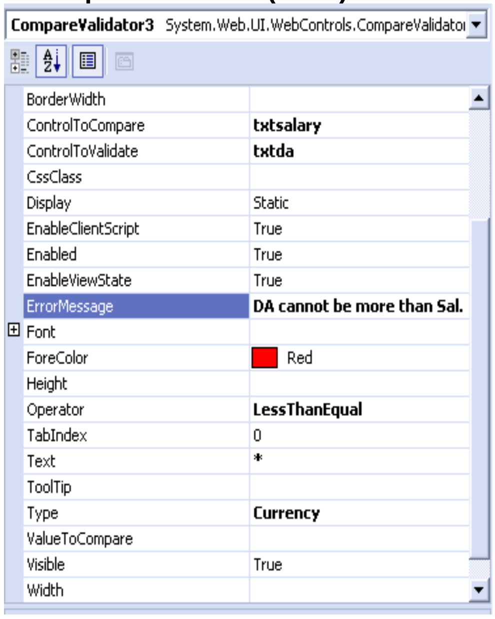
compare validator (txtjoindate)



CompareValidator1 System.Web.UI.WebControl

ControlToCompare	
ControlToValidate	**txtsalary**
CssClass	
Display	Static
EnableClientScript	True
Enabled	True
EnableViewState	True
ErrorMessage	**Salary is Invalid**
Font	
ForeColor	🟥 Red
Height	
Operator	**DataTypeCheck**
TabIndex	0
Text	*
ToolTip	
Type	**Currency**
ValueToCompare	
Visible	True
Width	**15px**

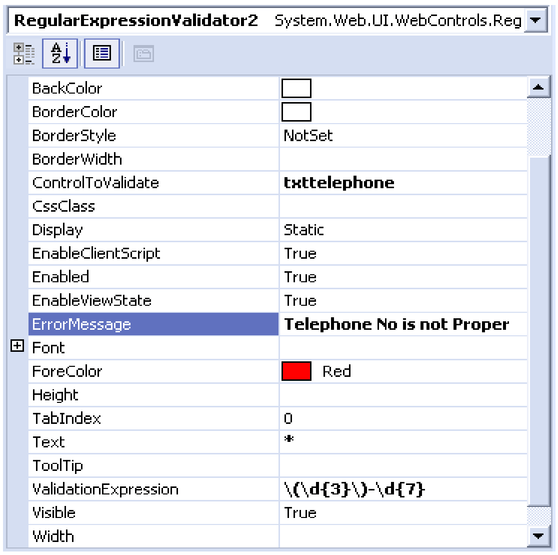
compare validator (txtda)



CompareValidator3 System.Web.UI.WebControls.CompareValidato

BorderWidth	
ControlToCompare	txtsalary
ControlToValidate	txtda
CssClass	
Display	Static
EnableClientScript	True
Enabled	True
EnableViewState	True
ErrorMessage	DA cannot be more than Sal.
⊞ Font	
ForeColor	🟥 Red
Height	
Operator	LessThanEqual
TabIndex	0
Text	*
ToolTip	
Type	Currency
ValueToCompare	
Visible	True
Width	

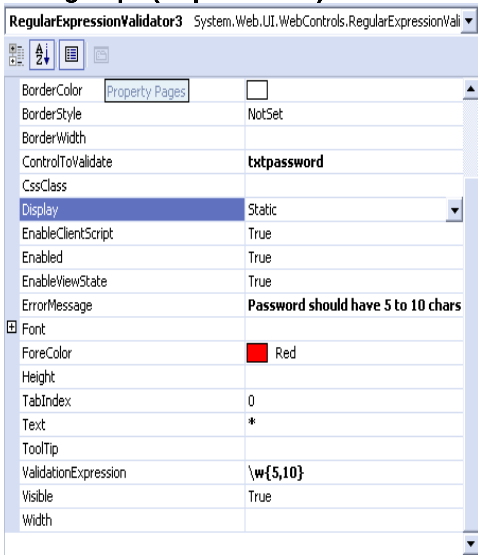
Reg expr (txttelehone)



RegularExpressionValidator2 System.Web.UI.WebControls.Reg

BackColor	□
BorderColor	□
BorderStyle	NotSet
BorderWidth	
ControlToValidate	txttelephone
CssClass	
Display	Static
EnableClientScript	True
Enabled	True
EnableViewState	True
ErrorMessage	Telephone No is not Proper
⊞ Font	
ForeColor	🟥 Red
Height	
TabIndex	0
Text	*
ToolTip	
ValidationExpression	\(\d{3}\)-\d{7}
Visible	True
Width	

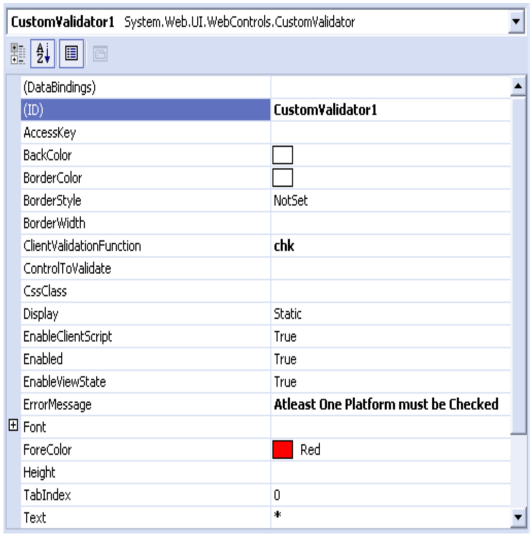
Reg expr (txtpassword)



RegularExpressionValidator3 System.Web.UI.WebControls.RegularExpressionVali

BorderColor	□
BorderStyle	NotSet
BorderWidth	
ControlToValidate	txtpassword
CssClass	
Display	Static
EnableClientScript	True
Enabled	True
EnableViewState	True
ErrorMessage	Password should have 5 to 10 chars
⊞ Font	
ForeColor	🟥 Red
Height	
TabIndex	0
Text	*
ToolTip	
ValidationExpression	\w{5,10}
Visible	True
Width	

Custom Validator (for checkboxes)



CustomValidator1 System.Web.UI.WebControls.CustomValidator

(DataBindings)	
(ID)	CustomValidator1
AccessKey	
BackColor	□
BorderColor	□
BorderStyle	NotSet
BorderWidth	
ClientValidationFunction	chk
ControlToValidate	
CssClass	
Display	Static
EnableClientScript	True
Enabled	True
EnableViewState	True
ErrorMessage	Atleast One Platform must be Checked
⊞ Font	
ForeColor	🟥 Red
Height	
TabIndex	0
Text	*

4. Go to the HTML Tab of the aspx design page and add this javascript code just under <body> tag :

```
<script language="javascript" type="text/javascript">
   function chk(src, args)
   {
      if (window.document.Form1.chkdotnet.checked == false &&
                window.document.Form1.chkoracle.checked == false &&
                window.document.Form1.chkjava.checked == false)
        args.IsValid = false;
     else
        args.IsValid = true;
  }
    </script>
```

5. Set this function "**chk**" as a **clientvalidationfunction** property value for the customvalidator

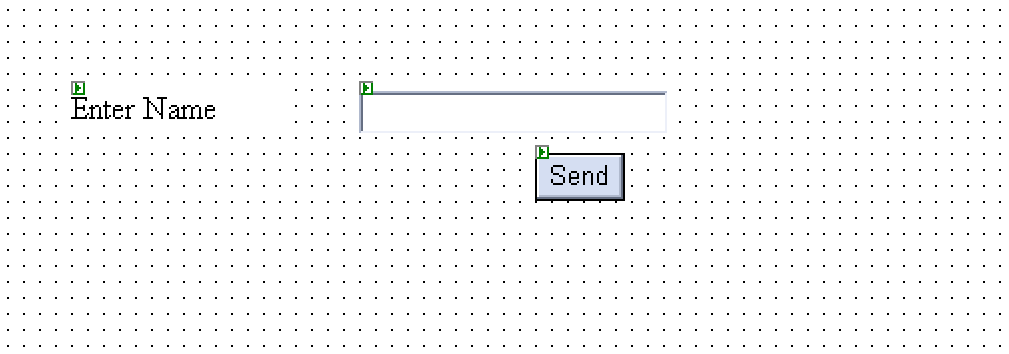
6. Set ShowMessageBox and ShowSummary properties of Validation Summary to true/false so that messages can be alerted (messageboxed) or printed on the webpage.

7. Run the web page and check the validation.

Lab 6.Working with Redirection

Description	• In this lab we will use redirect method available in response built-in object to take the user to different page.
Objective	To learn: • How to redirect user to a different page
Time	30 Mins

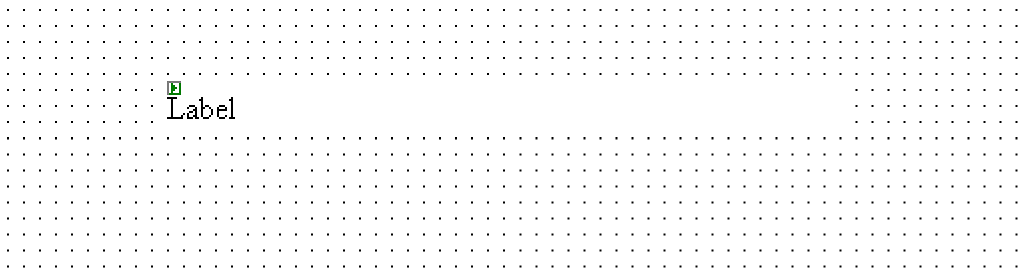
1. Create a Page RedirectFromPage.aspx



2. The send button click code

```
private void btnsend_Click(object sender, System.EventArgs e)
{
    Response.Redirect("NextPage_R.aspx?name=" +
                                              Server.UrlEncode(txtname.Text));
}
```

3. Create a RedirectToPage.aspx page as below (with only one label)



4. The page load code for RedirectToPage.aspx

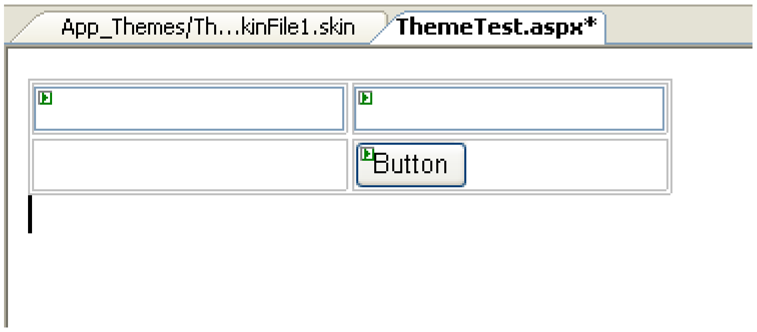
```
private void Page_Load(object sender, System.EventArgs e)
{
    Label1.Text = "Welcome " + Request.QueryString.Get("name");
    }
```

8. Make RedirectFromPage.aspx as start page and Run. Type in a name and click send button. Take a look at Browser Url.

Lab 7.Working with Asp.net Themes

Description	• In this lab we will be having 2 pages and applying theme at page and application level.
Objective	To learn: • How to create a theme • Applying theme at page and application level
Time	60 Mins

1. Add a new Page Name it as ThemeTest.aspx
2. Add a HTML table of 2rows and 2 columns
3. Drag a 2 Textboxes and Button into the table



4. Create a Theme
   a. In Solution Explorer Right Click Web Site => Add ASP.NET Folder => Theme
   b. An "**App_Themes**" folder will be added with **Theme1** folder in it.
   c. Right Click **Theme1** Folder; Select Add New Item; Select Skin File. Name it **SkinFile1.Skin**

5. Add this code below the existing commented block or you can delete all contents of skin file(Ctrl+A, Delete) and put the lines given below

```
 <asp:TextBox runat="server" BorderColor="#0000C0" BorderStyle="Solid"
        BorderWidth="1px" Font-Bold="True" Font-Size="Smaller" ForeColor="Maroon">
</asp:TextBox>
<asp:Button  runat="server" BackColor="Tan" BorderStyle="Ridge" BorderWidth="1px"
                  Font-Bold="True" Font-Size="Small" ForeColor="Maroon" Text="Button" />
```

6. We have created a Theme but not applied. So Run your ThemeTest.aspx and the controls will be seen as default
7. Lets apply the Theme first at page level.
8. For the page ThemeTest.aspx in design view. Goto Property window of the page(documet should be seen in the property window dropdown at top OR select it there).
9. Locate a property Theme, and from elipsis select Theme1.

10. Run the Page the theme will be applied
11. Add One More Page ThemeTest2.aspx with same GUI as ThemeTest.aspx

12. Make this page as startup and run. Controls will not be according to theme.
13. Now let’s apply theme at the Application Level.
14. Remove Theme Property Setting of ThemeTest.aspx.
15. Open web.config add <pages theme="Theme1"/> below <system.web>
16. Now run and check both pages ThemeTest.aspx and ThemeTest2.aspx

Lab 8.Working with Master Page Concept

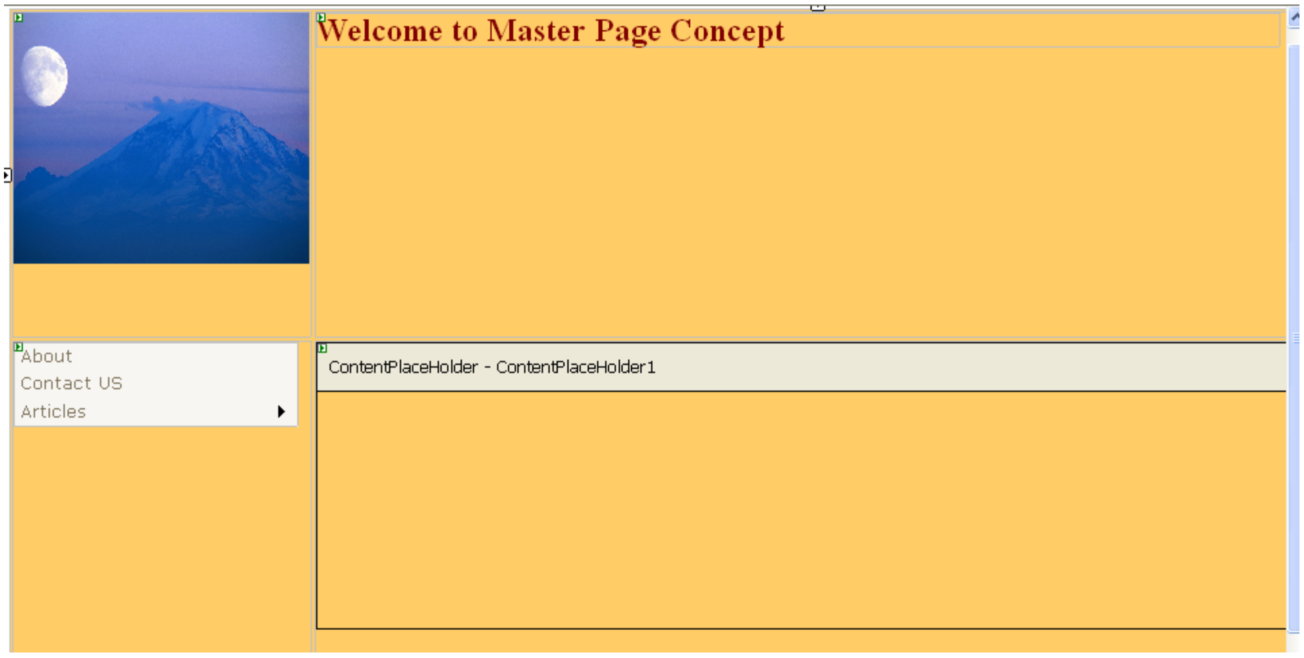
Description	• In this lab we will be creating a master page and a content page based on the master.
Objective	To learn: • How to create a theme How to create a master page • How to create a content page based on master page • Accessing contents of master page from content page
Time	90 Mins

1. Add a Master Page
   a. Project => Add New Item => Master Page
   b. Name it MyMaster1.master
2. A Design Page Similar to aspx will come.
3. Select the Content Pace Holder Control and delete it.
4. Goto Source Tab of the Master Page and paste the following between <form>…/form>

```
    <div>
      <table style="width: 100%; height: 100%; background-color: #ffcc66;">
          <tr>
             <td  valign="top" >
                <asp:Image style="position:relative" ID="Image1" runat="server" Height="186px"
Width="219px" ImageUrl="~/Images/Ascent.jpg" />
              </td>
              <td valign="top">
                <asp:Label ID="Label1" runat="server" Text="Welcome to Master Page Concept"
Width="714px" Font-Bold="True" Font-Size="X-Large" ForeColor="Maroon"></asp:Label>
               </td>
          </tr>
          <tr>
             <td  valign="top" style="width: 176px">
                <asp:Menu ID="Menu1" runat="server" Width="211px" BackColor="#F7F6F3"
DynamicHorizontalOffset="2" Font-Names="Verdana" Font-Size="0.8em" ForeColor="#7C6F57"
StaticSubMenuIndent="10px">
                   <StaticMenuItemStyle HorizontalPadding="5px" VerticalPadding="2px" />
                   <DynamicHoverStyle BackColor="#7C6F57" ForeColor="White" />
                   <DynamicMenuStyle BackColor="#F7F6F3" />
                   <StaticSelectedStyle BackColor="#5D7B9D" />
                   <DynamicSelectedStyle BackColor="#5D7B9D" />
                   <DynamicMenuItemStyle HorizontalPadding="5px" VerticalPadding="2px" />
                   <Items>
                      <asp:MenuItem NavigateUrl="About.aspx" Text="About"
Value="About"></asp:MenuItem>
                      <asp:MenuItem NavigateUrl="ContactUs.aspx" Text="Contact US" Value="Contact
US"></asp:MenuItem>
                      <asp:MenuItem Text="Articles" Value="Articles">
```

```
                                <asp:MenuItem NavigateUrl="aspnet.aspx" Text="ASP.NET"
Value="ASP.NET"></asp:MenuItem>
                                <asp:MenuItem NavigateUrl="adonet.aspx" Text="ADO.NET"
Value="ADO.NET"></asp:MenuItem>
                                <asp:MenuItem NavigateUrl="remoting.aspx" Text="Remoting"
Value="Remoting"></asp:MenuItem>
                            </asp:MenuItem>
                        </Items>
                        <StaticHoverStyle BackColor="#7C6F57" ForeColor="White" />
                    </asp:Menu>
                </td>
                <td  valign="top">
                    <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
                    </asp:ContentPlaceHolder>
                </td>
            </tr>
        </table>
    </div>
```

5. Switch over to to the Design TAB View.
   It should look like below:


Welcome to Master Page Concept
About
Contact US
Articles
ContentPlaceHolder - ContentPlaceHolder1

6. Lets create a web page based on the master called as Content Page.
   a. Website => Add Content Page (if MyMaster1.master is already open and active in Studio)
   b. OR Web Site => Add New Item
   c. Select Web Form as template and check checkbox below.
   d. Name the page as MasterTest1.aspx. Click Add
   e. Select MyMaster1.master from the next dialog and click OK
   f. So now the content page has logcially inherited the looks of its master

g. Goto the Source Tab of the content page and observe the code
   Lets understand few things:
   1. The Master page has ContentPlaceHolder and this contents page has Conetnt control.
   2. The Content Page does not have a html form.
   3. The Content Control of Content Page and ContentPlaceholder of Master is linked through attribute.

   Content Control is a place which can be customized for the very actual objective of the page.

7. Goto Source tab of Content Page (MasterTest1.aspx)
   a. The Souce Code should like this:

```
        <asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <table>
        <tr>
            <td style="width: 100px">
                <asp:Label ID="Label1" runat="server" Text="Enter No1"></asp:Label></td>
            <td style="width: 100px">
                <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox></td>
        </tr>
        <tr>
            <td style="width: 100px">
                <asp:Label ID="Label2" runat="server" Text="Enter No2"></asp:Label></td>
            <td style="width: 100px">
                <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox></td>
        </tr>
        <tr>
            <td style="width: 100px">
         <asp:Label ID="Label3" runat="server" EnableViewState="False" ></asp:Label></td>
            <td style="width: 100px">
                <asp:Button ID=" btnmultiply " runat="server" OnClick="Button1_Click" Text="Multiply"
/></td>
        </tr>
    </table>
                          </asp:Content>
```

8. Come to the design TAB of Content Page. and check
   What have we done simply is in HTML table we have some GUI for Obvious Mulitply Operation.

9. Double Click Multiply Button and write as below:

```
    protected void btnmultiply_Click(object sender, EventArgs e)
    {
        try{
            int res = int.Parse(TextBox1.Text) * int.Parse(TextBox2.Text);
            Label3.Text = res.ToString();
        }
        catch(Exception ex)
        {
            Label3.Text = ex.Message;
        }
```

```
}
```

10. Make this page as startup and run

Accessing the parts of master page from content page.

11. Goto the Code Behind of MyMaster1.master
12. Add the property **PageHeading** as below

```
    public string PageHeading
  {
         set
     {
            Label1.Text = value;
     }
  }
```

its writeonly property

13. Come to Page Load Event of Content Page MasterTest1.aspx
14. Write the code as below:

```
protected void Page_Load(object sender, EventArgs e)
  {
            MyMaster1 m = (MyMaster1)this.Master;
            m.PageHeading = "A New Page Heading by Child Content Page";

  }
```

15. Run the Content Page.

Lab 9.Working with Asp.net Tracing feature

Description	• In this lab we will enable tracing for web pages at application level and see the trace output.
Objective	To learn: • How to enable tracing at application level. • How to view trace information • How to write custom additional messages to trace
Time	60 Mins

1. Open the PostBackDemo.aspx page created in earlier lab.
2. In the selection change event of drop down change the event code as below:

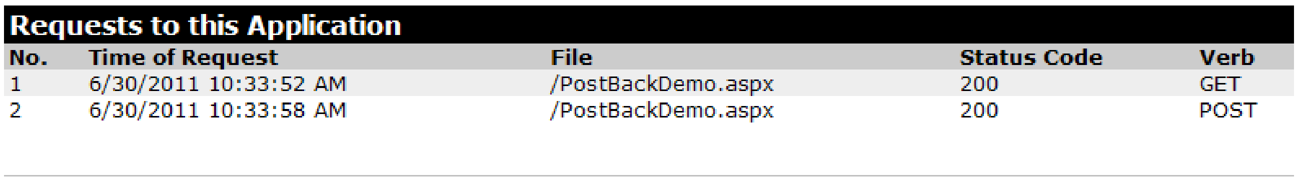
```
protected void dnservicelist_SelectedIndexChanged(object sender, EventArgs e)
{
        txtservice.Text = dnservicelist.SelectedValue;
       //Write custom message text to trace output
        Trace.Write("user selected " + dnservicelist.SelectedValue);
       Trace.Warn("user selected " + dnservicelist.SelectedValue);
 }
```

3. Open web.config and put the following xml entry within <system.web> …</system.web>

```
<trace enabled="true" pageOutput="false"/>
```

4. Run the PostBackDemo.aspx, change the selection in the dropdown you will get the output as well as trace for the page execution will be generated.
5. To see the trace replace the url by changing "**PostBackDemo.aspx**" in the url to "**trace.axd**"
   For eg: http://localhost:1454/AspNetDemo/PostBackDemo.aspx to http://localhost:1454/AspNetDemo/Trace.axd

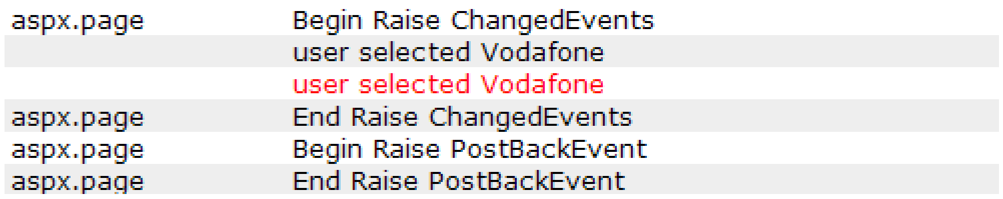
6. 2 Trace entries will be listed



Requests to this Application

No.	Time of Request	File	Status Code	Verb
1	6/30/2011 10:33:52 AM	/PostBackDemo.aspx	200	GET
2	6/30/2011 10:33:58 AM	/PostBackDemo.aspx	200	POST

7. View the trace for page execution in post method (verb).
8. The trace will show lot information about page execution
   a. sessionid, cookies, application, session details, page lifecycle, time taken for evey lifecycle event. The render size in bytes of each control
   b. notice the custom messages written to trace also will be seen as shown below



```
aspx.page      Begin Raise ChangedEvents
               user selected Vodafone
               user selected Vodafone
aspx.page      End Raise ChangedEvents
aspx.page      Begin Raise PostBackEvent
aspx.page      End Raise PostBackEvent
```

This trace information is useful for debugging and error tracking not just during development but post deployment.

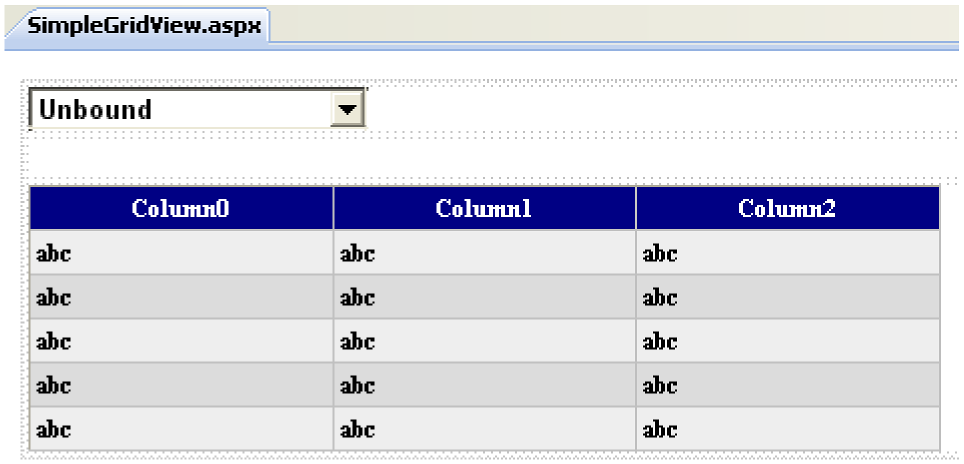
9. Open web.config and disable tracing

   ```
   <trace enabled="false" pageOutput="false"/>
   ```

Lab 10. Working with GridView

Description	• In this lab we will be having a DropDownList which will have product categories listed, and on the selection of a category the GridView control will show the corresponding category products.
Objective	To learn: • How to data bind a Dropdownlist • How to data bind a GridView
Time	2 Hrs

1. Add a new web page. Name it simplegridview.aspx
2. Add 3 rows, 1 column table.
   a. table => insert table
3. From toolbox drag a dropdown (dncategory) in 1st row and GridView (grdproduct) in 3rd row
4. Select GridView from smart tag go to auto format and select a format
5. Set the autopostback property of dropdown to true.



Column0	Column1	Column2
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc

6. Open web.config from solution explorer
   If web.config file is missing add it via website => add new item; and selecting “web configuration file” template.

7. Add following Connection String related xml entry in web.config. just above <system.web>

```
<connectionStrings>
   <add name="labdemoconnectstring"
              connectionString="server=ndamssql\sqlilearn;database=labdemos;user
                                                    id=sqluser;password=sqluser"/>
</connectionStrings>
```

8. Go to code behind file **SimpleGridView.aspx.cs**
9. Add the following namespaces

```
using System.Configuration;
using System.Data.SqlClient;
using System.Data;
```

10. Define following 2 functions

```
 private void PopulateDropDown()
  {
     string connectsring = ConfigurationManager.ConnectionStrings
                                            ["labdemoconnectstring"].ConnectionString;
     SqlConnection con = new SqlConnection(connectsring);
     SqlDataAdapter da = new SqlDataAdapter
                                  ("select * from category order by categoryname", con);

     DataSet ds = new DataSet();
     da.Fill(ds, "cat");
     dncategory.DataSource = ds.Tables["cat"];
     dncategory.DataTextField = "categoryname";
     dncategory.DataValueField = "categoryid";
     dncategory.DataBind();

  }

  private void PopulateGridView()
  {
     string connectsring = ConfigurationManager.ConnectionStrings
                                            ["labdemoconnectstring"].ConnectionString;
     SqlConnection con = new SqlConnection(connectsring);
     SqlDataAdapter da = new SqlDataAdapter
         ("select productid,productname,unitprice,unitsinstock from product " +
                  " where categoryid = @catid  order by productname", con);
     da.SelectCommand.Parameters.Add("@catid", SqlDbType.Int);
     da.SelectCommand.Parameters["@catid"].Value =
                                                 dncategory.SelectedValue;

     DataSet ds = new DataSet();
     da.Fill(ds, "pro");
     grdproduct.DataSource = ds.Tables["pro"];
     grdproduct.DataBind();

  }
```

11. The page load event code

```
protected void Page_Load(object sender, EventArgs e)
{
     if (!IsPostBack)
     {
        PopulateDropDown();
        PopulateGridView();
```

```
            }
    }
```

12. Double click the drop down to generate selection changed event and write this code:

```
protected void dncategory_SelectedIndexChanged(object sender, EventArgs e)
   {
      PopulateGridView();
   }
```

13. Make this page as startup and run
14. If you change the drop down category the products for the category will be immediately listed in the grid view.

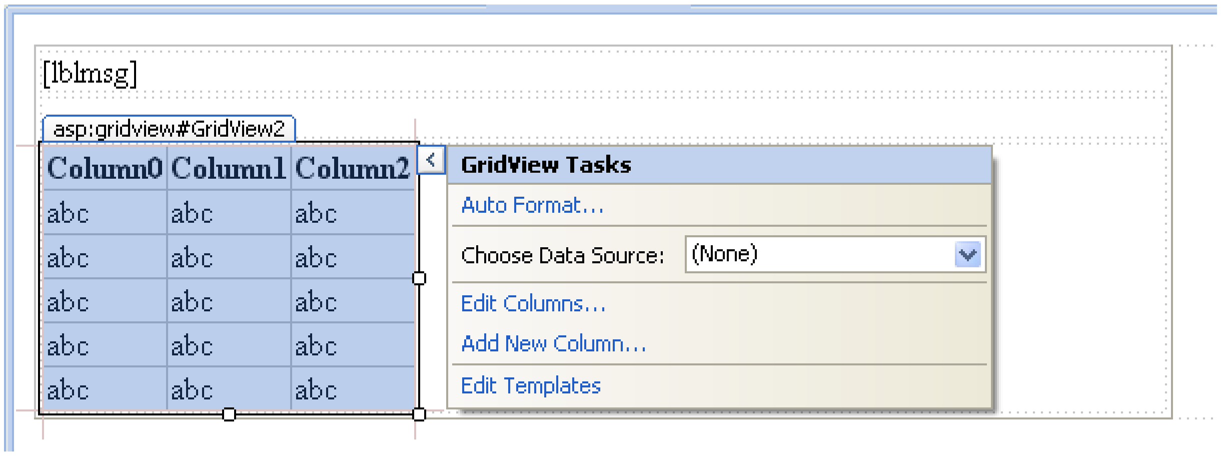
Assignment 1 To Do::

Create one more page SimpleGridViewWithRadioButtonList.aspx functionally similar to the above lab. Instead of dropdown use RadioButtonList control from the toolbox. With this control every category will be shown as a separate radio button.

Lab 11. Working with GridView Templates

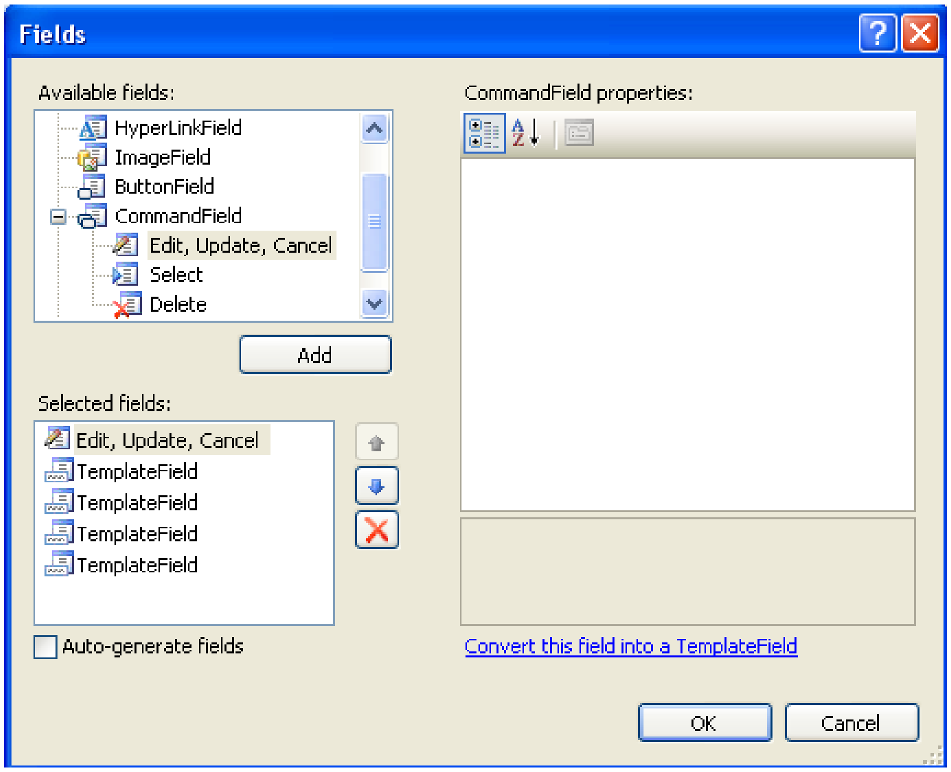
Description	• In this lab we will use a GridView control to display multiple customers information in tabular format. Page will also allow user to modify the customer details.
Objective	To learn: • Using GridView Control and it's features a. Templates b. Paging c. Editing
Time	4 Hrs

1. Add a blank asp.net web page, Name it GridViewDemoPage
2. Add 4 rows, 1 column HTML table. Table => Insert Table
3. Drag a Label (lblmsg) and a GridView (grdcustomer) in the HTML Table as below:
    a. From the smart tag select Auto Format, and select on of the formatting template
    b. Go to Property window for GridView and change the following properties:
        i. AllowPaging:= True
        ii. BackColor:= White
        iii. FontSize:= 10pt (under Font property ellipse), FontBold:= True
        iv. PageSize:= 3



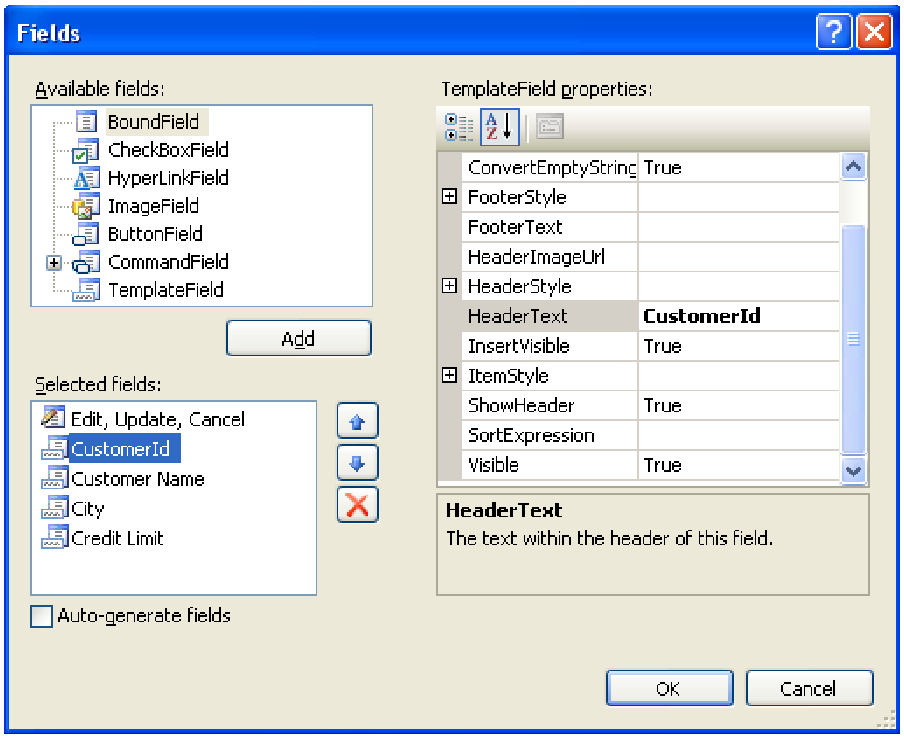
Column0	Column1	Column2
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc

4. From the smart tag go to "Edit Columns"
    a. Uncheck Auto generate fields
    b. From top listbox expand CommandField node and select "Edit, Update and Cancel" and click add.
    c. Select Templatefield from top listbox & click add to add such 4 templated columns



5. Select the TemplateField one-by-one from “Selected Fields” list and set the HeaderText Property as shown below:

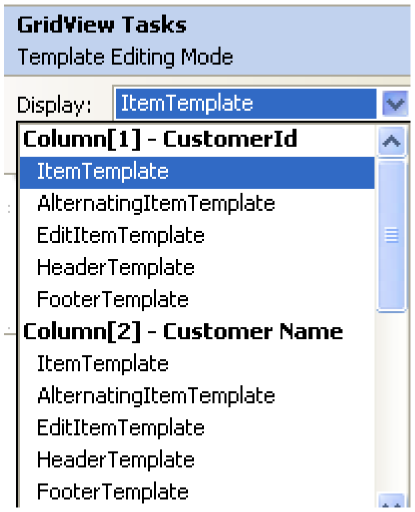
   You can click A↓Z button to sort properties in alphabetical order as shown here.



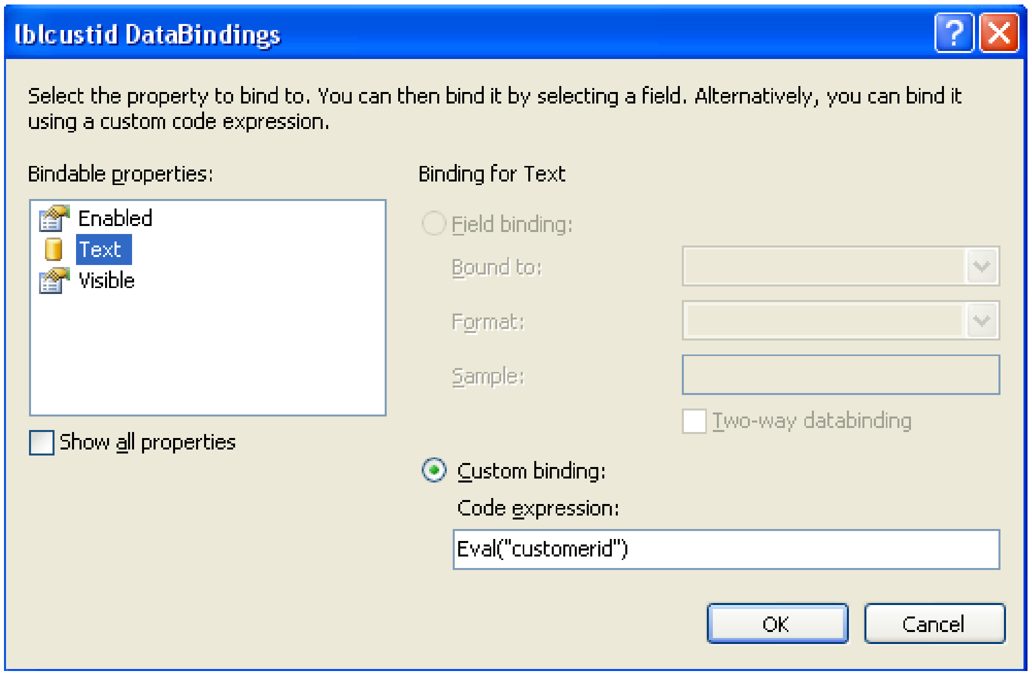
ConvertEmptyString	True
FooterStyle	
FooterText	
HeaderImageUrl	
HeaderStyle	
HeaderText	CustomerId
InsertVisible	True
ItemStyle	
ShowHeader	True
SortExpression	
Visible	True

HeaderText
The text within the header of this field.

6. Click Ok to close the dialog box.
7. From GridView smart tag select "edit templates"
8. There will be 4 columns as per added in step 4 – c
   We will be adding controls (labels in itemtemplate and textboxes in edititemtemplate) for each one of the columns in the GridView.



9. In the item template for column 0 (i.e.: CustomerId) drag a label rename the label as lblcustid.
10. Go to smart tag of this label, click Edit Bindings…
    a. For Text property set the value for code expression as shown below:



lblcustid DataBindings

Select the property to bind to. You can then bind it by selecting a field. Alternatively, you can bind it using a custom code expression.

Bindable properties: Enabled, Text, Visible

Binding for Text

Custom binding:

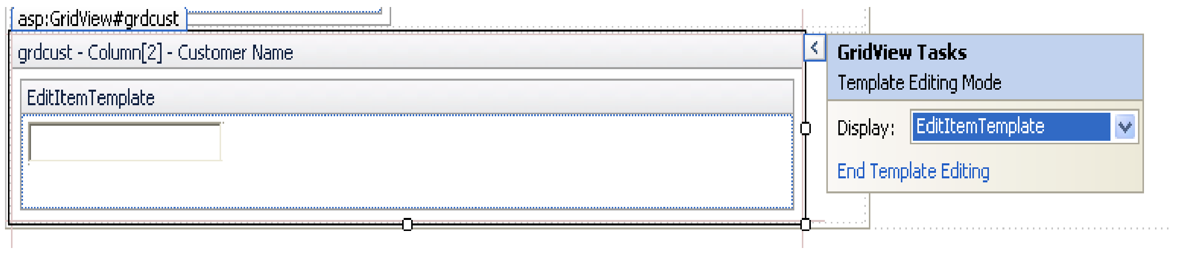
Code expression: Eval("customerid")

11. Repeat these steps to drag and bind labels in itemtemplate for every column.

Configuring Edit Item Template for every column in GridView:

12. Now in the edititemtemplate of every column drag a textbox and bind them with the corresponding code expression
13. Exception is for edititemtemplate of CustomerId. For edititemtemplate of CustomerId drag a label, since we will not allow modification of primary key i.e. CustomerId while modifying customer details.

Images below show textbox dragged in edititemtemplate for customername column in GridView and it's data bindings



asp:GridView#grdcust

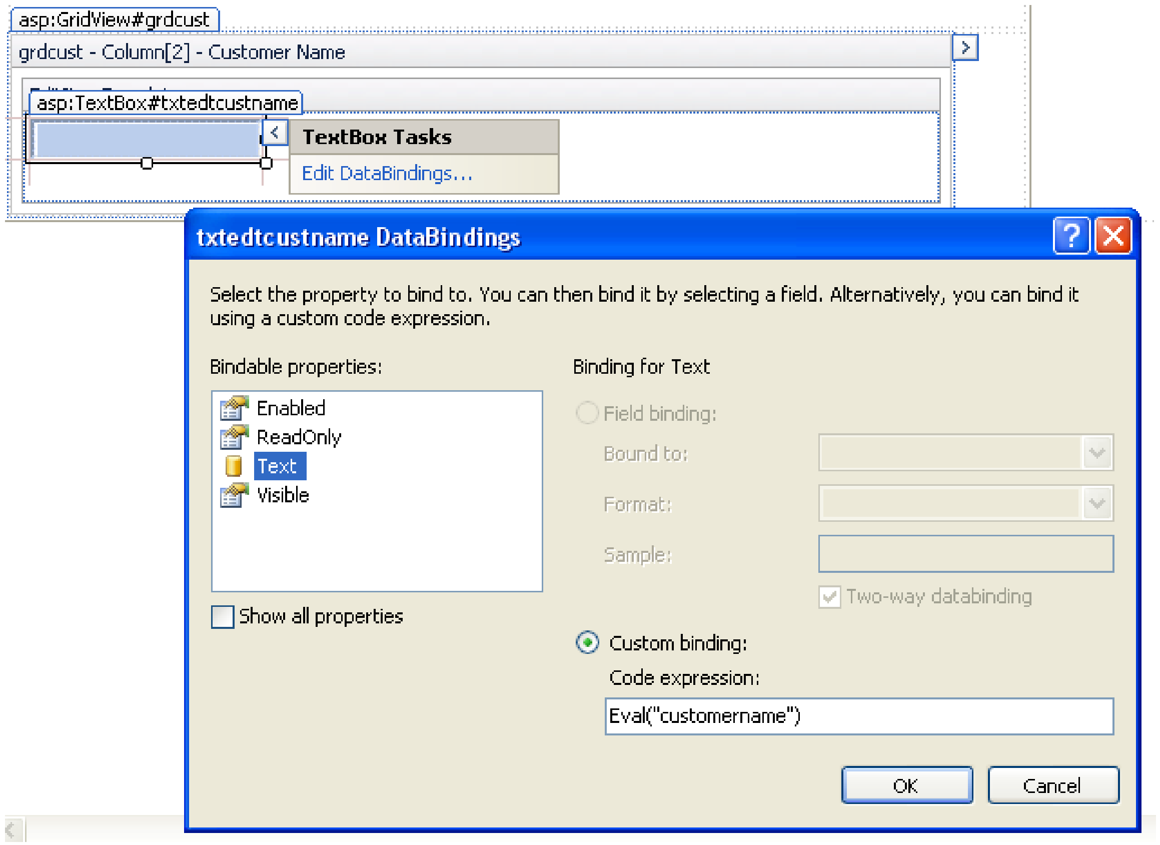
grdcust - Column[2] - Customer Name

EditItemTemplate

GridView Tasks

Template Editing Mode

Display: EditItemTemplate


asp:GridView#grdcust
grdcust - Column[2] - Customer Name
asp:TextBox#txtedtcustname
TextBox Tasks
Edit DataBindings...

txtedtcustname DataBindings

Select the property to bind to. You can then bind it by selecting a field. Alternatively, you can bind it using a custom code expression.

Bindable properties: Enabled, ReadOnly, Text, Visible

[ ] Show all properties

Binding for Text

( ) Field binding:
Bound to:
Format:
Sample:
[✓] Two-way databinding

(●) Custom binding:
Code expression: Eval("customername")

OK | Cancel

14. Click Ed Edit Templates
15. Go to the "Source View " tab of the aspx page and check the html code for the columns definition of GridView is as given below:
    Note: There is label for customerid in itemtemplate and edit item template

```
<Columns>
            <asp:CommandField ShowEditButton="True" />

            <asp:TemplateField HeaderText="CustomerId">
               <EditItemTemplate>
                  <asp:Label ID="lbledtcustid" runat="server"
                              Text='<%# Eval("customerid") %>'></asp:Label>
               </EditItemTemplate>
               <ItemTemplate>
                  <asp:Label ID="lblcustid" runat="server"
                              Text='<%# Eval("customerid") %>'></asp:Label>
               </ItemTemplate>
            </asp:TemplateField>


            <asp:TemplateField HeaderText="Customer Name">
               <EditItemTemplate>
                  <asp:TextBox ID="txtedtcustname" runat="server"
```

```
                Text='<%# Eval("customername") %>'></asp:TextBox>
            </EditItemTemplate>
            <ItemTemplate>
                <asp:Label ID="lblcustname" runat="server"
                        Text='<%# Eval("customername") %>'></asp:Label>
            </ItemTemplate>
        </asp:TemplateField>


        <asp:TemplateField HeaderText="City">
            <EditItemTemplate>
                <asp:TextBox ID="txtedtcity" runat="server"
                        Text='<%# Eval("city") %>'></asp:TextBox>
            </EditItemTemplate>
            <ItemTemplate>
                <asp:Label ID="lblcity" runat="server"
                                    Text='<%# Eval("city") %>'></asp:Label>
            </ItemTemplate>
        </asp:TemplateField>


        <asp:TemplateField HeaderText="Credit Limit">
            <EditItemTemplate>
                <asp:TextBox ID="txtedtcreditlimit" runat="server"
                        Text='<%# Eval("creditlimit") %>'></asp:TextBox>
            </EditItemTemplate>
            <ItemTemplate>
                <asp:Label ID="lblcreditlimit" runat="server"
                            Text='<%# Eval("creditlimit") %>'></asp:Label>
            </ItemTemplate>
        </asp:TemplateField>

    </Columns>
```

16. In the web page code behind add the following namespaces

```
using System.Data.SqlClient;
using System.Data;
using System.Configuration;
```

17. Define a private method in the page class

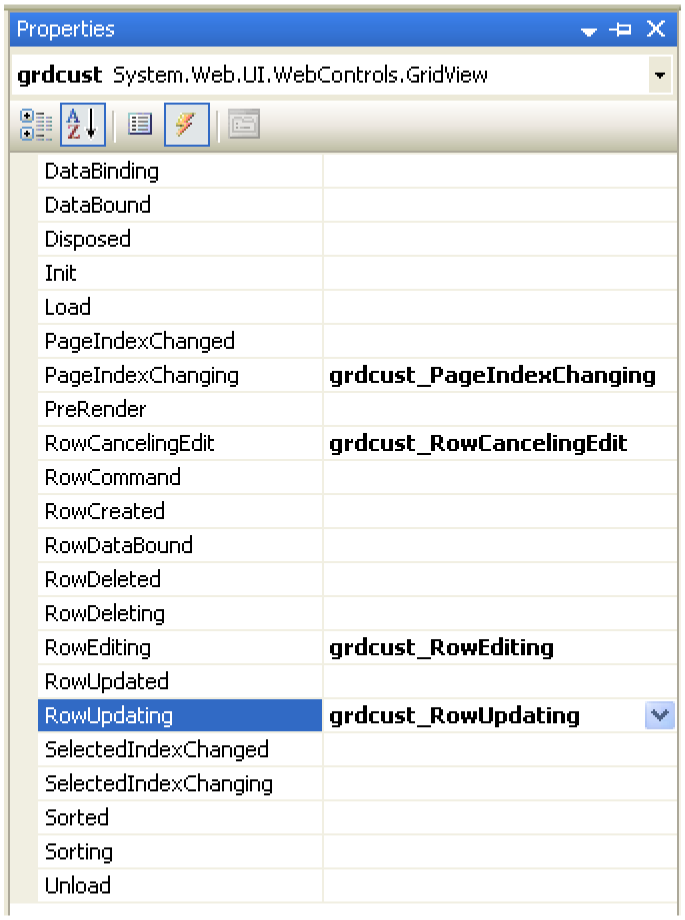
```
private void PopulateGrid()
{
    string connectsring = ConfigurationManager.ConnectionStrings
                                ["labdemoconnectstring"].ConnectionString;
    SqlConnection con = new SqlConnection(connectsring);
    SqlDataAdapter da = new SqlDataAdapter("select * from customer", con);

    DataSet ds = new DataSet();
    da.Fill(ds, "cust");
```

```
        grdcust.DataSource = ds.Tables["cust"];
        grdcust.DataBind();
    }
```

18. The page load event code

```
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
            PopulateGrid();
    }
```

19. Go to property window of GridView (click ⚡) and generate the following event handlers for the GridView by double clicking the events



DataBinding	
DataBound	
Disposed	
Init	
Load	
PageIndexChanged	
PageIndexChanging	grdcust_PageIndexChanging
PreRender	
RowCancelingEdit	grdcust_RowCancelingEdit
RowCommand	
RowCreated	
RowDataBound	
RowDeleted	
RowDeleting	
RowEditing	grdcust_RowEditing
RowUpdated	
RowUpdating	grdcust_RowUpdating
SelectedIndexChanged	
SelectedIndexChanging	
Sorted	
Sorting	
Unload	

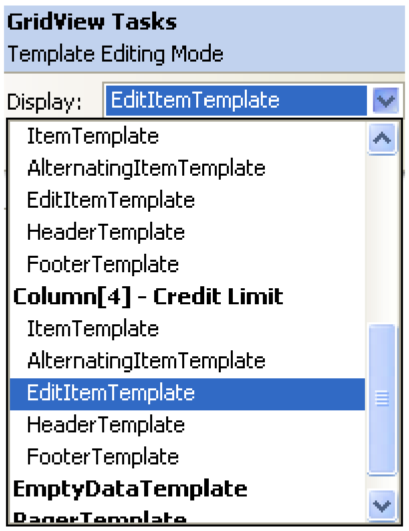
20. The GridView event handler code given below:

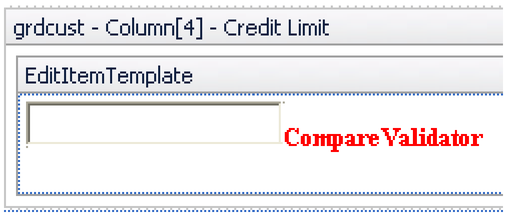
```
    protected void grdcust_PageIndexChanging(object sender, GridViewPageEventArgs e)
    {
        grdcust.PageIndex = e.NewPageIndex;
        PopulateGrid();
    }

    protected void grdcust_RowEditing(object sender, GridViewEditEventArgs e)
    {
        grdcust.EditIndex = e.NewEditIndex;
        PopulateGrid();
    }

  protected void grdcust_RowUpdating(object sender, GridViewUpdateEventArgs e)
  {
            string custid,custname,city,crlimit;

            //when update button is clicked the row is in edit item template
            //Get the reference of the controls in edit item template to get the entered values
               Label lb = (Label)grdcust.Rows
                              [e.RowIndex].FindControl("lbledtcustid");
               custid = lb.Text;

               TextBox tb=(TextBox)grdcust.Rows
                              [e.RowIndex].FindControl("txtedtcustname");
               custname = tb.Text;

               tb = (TextBox)grdcust.Rows
                                    [e.RowIndex].FindControl("txtedtcity");
               city = tb.Text;

               tb = (TextBox)grdcust.Rows
                                       [e.RowIndex].FindControl("txtedtcreditlimit");
               crlimit = tb.Text;
               string connectsring = ConfigurationManager.ConnectionStrings
                                                        ["labdemoconnectstring"].ConnectionString;
               SqlConnection con = new SqlConnection(connectsring);
               SqlCommand cmd = new SqlCommand("update customer set “  +
                                             “customername=@cname,city=@city,creditlimit=@credit " +
                                   " where customerid= @cid", con);

               cmd.Parameters.AddWithValue("@cname", custname);
               cmd.Parameters.AddWithValue("@city", city);
               cmd.Parameters.AddWithValue("@credit", crlimit);
               cmd.Parameters.AddWithValue("@cid", custid);

               con.Open();
               cmd.ExecuteNonQuery();
               con.Close();

               grdcust.EditIndex = -1;
               PopulateGrid();
    }

  protected void grdcust_RowCancelingEdit(object sender,
                                                            GridViewCancelEditEventArgs e)
```

```
 {
     grdcust.EditIndex = -1;
     PopulateGrid();
}
```

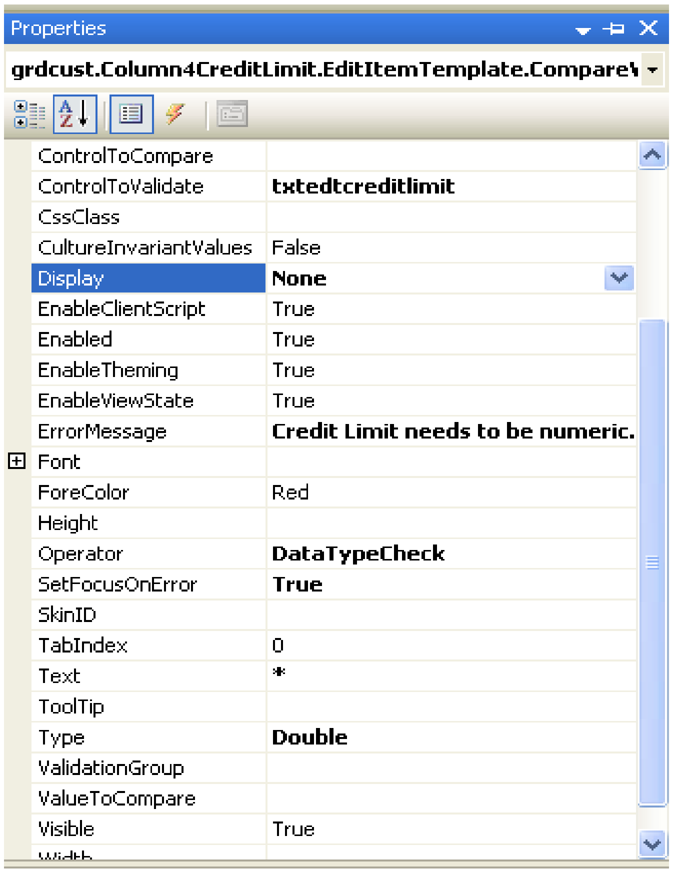
21. Make this page as startup and run.
    a. Click on page nos to navigate to different pages.
    b. Click edit make changes to customer name and click update to check that changes are updated.
    c. Again Click edit, change credit limit to a non numeric value and click update. There will be an error. Close the browser

22. Lets add validation; go to edit item template of creditlimit, drag a comparevalidator besides the textbox for creditlimit.





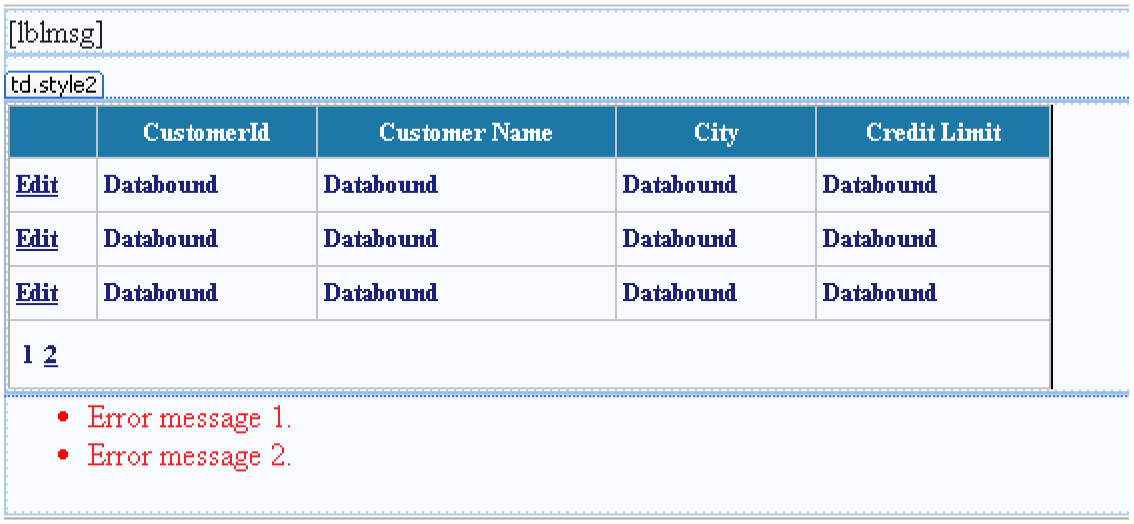
23. Set the following properties of comparevalidator control


Properties

grdcust.Column4CreditLimit.EditItemTemplate.CompareV

ControlToCompare	
ControlToValidate	txtedtcreditlimit
CssClass	
CultureInvariantValues	False
Display	None
EnableClientScript	True
Enabled	True
EnableTheming	True
EnableViewState	True
ErrorMessage	Credit Limit needs to be numeric.
Font	
ForeColor	Red
Height	
Operator	DataTypeCheck
SetFocusOnError	True
SkinID	
TabIndex	0
Text	*
ToolTip	
Type	Double
ValidationGroup	
ValueToCompare	
Visible	True

24. Drag a validation summary control on page below the GridView Row. Set the ShowMessageBox to true and ShowSummary property to false.


[lblmsg]

td.style2

	CustomerId	Customer Name	City	Credit Limit
Edit	Databound	Databound	Databound	Databound
Edit	Databound	Databound	Databound	Databound
Edit	Databound	Databound	Databound	Databound
1 2				

- Error message 1.
- Error message 2.

25. Run the page click edit, change the creditlimit to non numeric and click update. The page will respond with a proper message. Correct the creditlimit and update

Assignment 2 To Do::

Create a Employee Table in SQL Server which will have EmployeeId (primary key), EmployeeName, Salary, DA, PF as columns.Similar to this lab, create a web page to show all the employee details in a grid. Only 5 employees should be shown at a time. End User should be able to edit and update all the details except employeeid and PF. Make use of templates.

Lab 12. Working with Data Bound Controls GridView and DataList

Description	• In this lab we will be creating 3 web pages for small shopping demo. The first page will use a GridView control to display product categories information and second page will use DataList control to show product details and third page will use a GridView control to display cart.
Objective	To learn: • Using DataList Control • Linking informational pages • Adding hyperlink in GridView • Adding a custom button in DataList control • Use of user defined complex data type and collection for cart • Use of server side state management (Session/Application/Cache)
Time	5 Hrs

1. Add a new web page, Name it CategoryList.aspx.
2. Add a html table of 2 rows and 1 column
3. Drag a gridview (grdcategory) to 2nd row, Auto format the grid.
4. From smart tag go to edit columns. uncheck auto generate fields
5. Add 2 bound fields and a hyperlink field
6. Set the properties of as given below:

BoundField1	**DataTextField**: Categoryid **HeaderText**: Categoryid
HyperLinkField1	**DataTextField**: CategoryName **HeaderText**: CategoryName **DataNavigateUrlFormatString**:ProductCatalogue.aspx?catid={0} **DatNavigateUrlFields**: Categoryid **Target**: _blank
BoundField2	**DataTextField**: Description **HeaderText**: Description

7. Click Ok
8. Go to code behind add the namespaces

```
using System.Data.SqlClient;
using System.Data;
   using System.Configuration;
```

9. Define this function PopulateGridView() in page

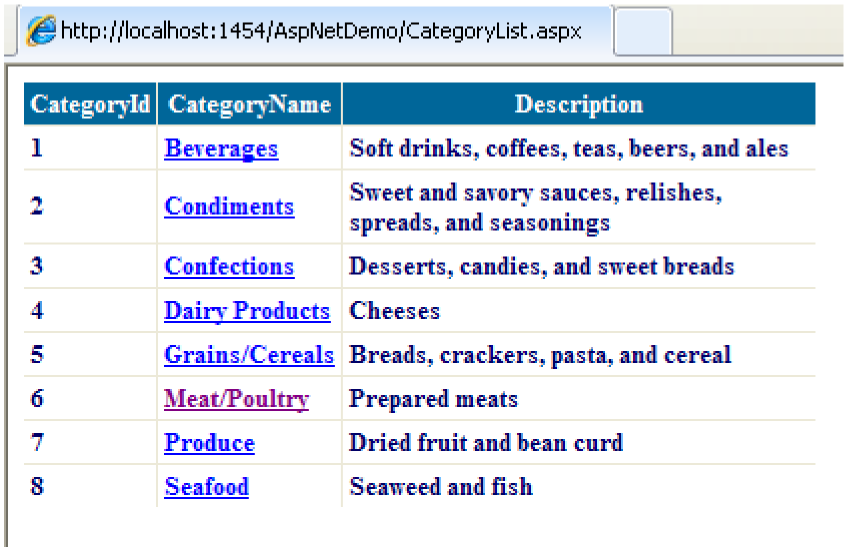
```
private void PopulateGridView()
 {
    DataSet ds = null;
```

```
         string connectsring =
ConfigurationManager.ConnectionStrings["labdemoconnectstring"].ConnectionString;
 SqlConnection con = new SqlConnection(connectsring);
 SqlDataAdapter da = new SqlDataAdapter("select * from category order by categoryname",
con);

            ds = new DataSet();
            da.Fill(ds, "procat");


        grdcategory.DataSource = ds.Tables["procat"];
        grdcategory.DataBind();
    }
```

10. The page load code:

```
  protected void Page_Load(object sender, EventArgs e)
    {
                  if (!IsPostBack)
                      PopulateGridView();
    }
```

11. Run this page, the product categories will be listed with categoryname as a hyperlink.

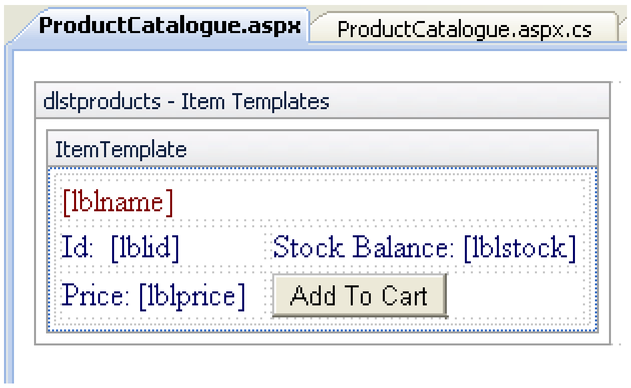


http://localhost:1454/AspNetDemo/CategoryList.aspx

CategoryId	CategoryName	Description
1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
3	Confections	Desserts, candies, and sweet breads
4	Dairy Products	Cheeses
5	Grains/Cereals	Breads, crackers, pasta, and cereal
6	Meat/Poultry	Prepared meats
7	Produce	Dried fruit and bean curd
8	Seafood	Seaweed and fish

12. Add one more page, ProductCatalogue.aspx
13. Drag a DataList control (dlstproducts)
14. From smart tag go to edit templates…
15. Put the text cursor in the Item Template
    a. Add a html table of 3 rows and 2 columns
    b. Merge the 2 cells of 1st row.

c. Drag a label in the merged cell. Go to edit Bindings put code expression as Eval("ProductName") to bind it to productname column
d. Similarly drag 3 more labels and Bind them to ProductId, Unitprice and UnitsInStock columns. Name the labels appropriately
e. Drag a Button in the 2nd cell 3rd row.
    i. Set Button Properties: Name=btnadd; Text=Add To Cart; CommandName=Add
f. Arrange the controls as shown below:



16. From smart tag select end template editing
17. Go to DataList properties and set Repeat columns: 3, Repeat Direction: Horizontal

18. Add a class definition
    a. website => add new item, select class template, Name it Product.cs, click add
    b. Click “yes” if asked to add to app_code folder.
19. The class definition is below:

```
public class Product
{
        public int ProductId{get;set;}
        public string ProductName{get;set;}
        public double UnitPrice{get;set;}
        public int UnitsInStock{get;set;}
        public int QtyOrdered { get; set; }
}
```

20. In the ProductCatalogue.aspx.cs code add the following namespaces

```
        using System.Data.SqlClient;
        using System.Data;
    using System.Configuration;
```

21. In the ProductCatalogue.aspx.cs code behind file add the following function in the page class

```
  private void PopulateDataList()
 {
            DataSet ds = null;
            //check whether information is in Application object
            //if not retrieve details and fill  Application object
            if (Application["productcategories"] == null)
            {
                string connectsring = ConfigurationManager.ConnectionStrings
                                        ["labdemoconnectstring"].ConnectionString;
                SqlConnection con = new SqlConnection(connectsring);
                SqlDataAdapter da = new SqlDataAdapter
             ("select * from product where categoryid = @catid  order by productname", con);
                da.SelectCommand.Parameters.Add("@catid", SqlDbType.Int);
                da.SelectCommand.Parameters["@catid"].Value = Request["catid"];

                ds = new DataSet();
                da.Fill(ds, "prod");

                Application["products"] = ds;
            }
            else
                ds = Application["products"] as DataSet;

            dlstproducts.DataSource = ds.Tables["prod"];
            dlstproducts.DataBind();
        }
```

22. The page load code:

```
    protected void Page_Load(object sender, EventArgs e)
     {
        if (!IsPostBack)
            PopulateDataList();

     }
```

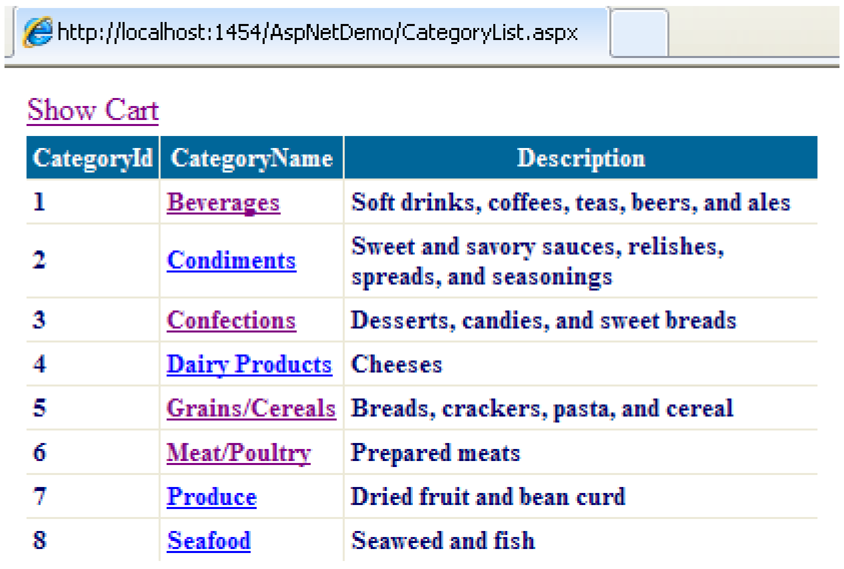
23. Generate a Item Command event handler and write the following code:

```
protected void dlstproducts_ItemCommand
                            (object source, DataListCommandEventArgs e)
  {
      if (e.CommandName == "Add")
      {
                    Label l;

                     List<Product> cart;
                     //find if cart is created for session
                    if (Session["cart"] == null)
                    {
                        cart = new List<Product>();
                        Session["cart"] = cart;
                    }
                    else
                        cart = Session["cart"] as List<Product>;
                     //Get the product details from the controls placed in Data List Item
                    Product prod = new Product();
                    l = e.Item.FindControl("lblname") as Label;
                    prod.ProductId = l.Text;

                    l = e.Item.FindControl("lblid") as Label;
                    prod.ProductName = l.Text;

                    l = e.Item.FindControl("lblstock") as Label;
                    prod.UnitsInStock = l.Text;

                    l = e.Item.FindControl("lblprice") as Label;
                    prod.UnitPrice = l.Text;

                    prod.QtyOrdered = 1;
                     //Add it to cart
                    cart.Add(prod);
     }
   }
```

24. Add a new page ShowCart.aspx
25. Drag a GridView (grdcart), auto format it.
26. From smart tag go to edit templates
    a. In the empty data template type “Cart Empty”
    b. click end template editing
27. The page load code for ShowCart.aspx

```
protected void Page_Load(object sender, EventArgs e)
{
   grdcart.DataSource = Session["cart"];
   grdcart.DataBind();
 }
```

28. In the categorylist.aspx drag a hyperlink in the 1st row
    a. Set it’s Text to Show Cart
    b. Go to navigateurl property, from ellipse button option select showcart.aspx
29. Make categorylist.aspx as startup page and run.

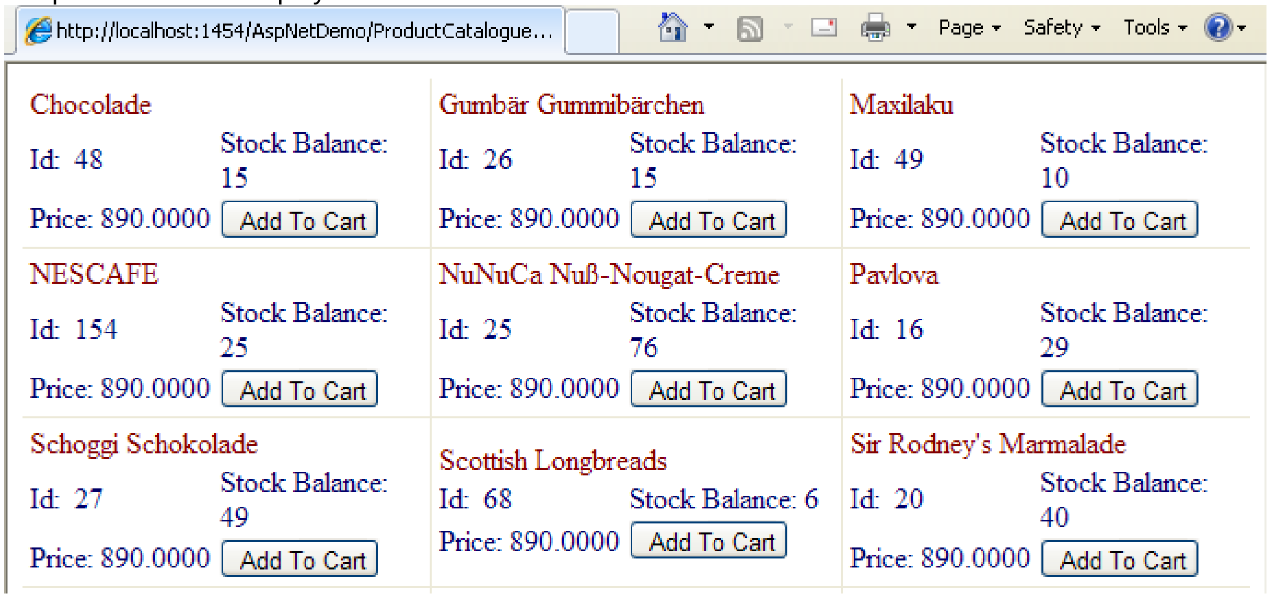


http://localhost:1454/AspNetDemo/CategoryList.aspx

Show Cart

CategoryId	CategoryName	Description
1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
3	Confections	Desserts, candies, and sweet breads
4	Dairy Products	Cheeses
5	Grains/Cereals	Breads, crackers, pasta, and cereal
6	Meat/Poultry	Prepared meats
7	Produce	Dried fruit and bean curd
8	Seafood	Seaweed and fish

30. Click show cart, it will be empty. Click browser back button and then click a category the products will be displayed as shown below:

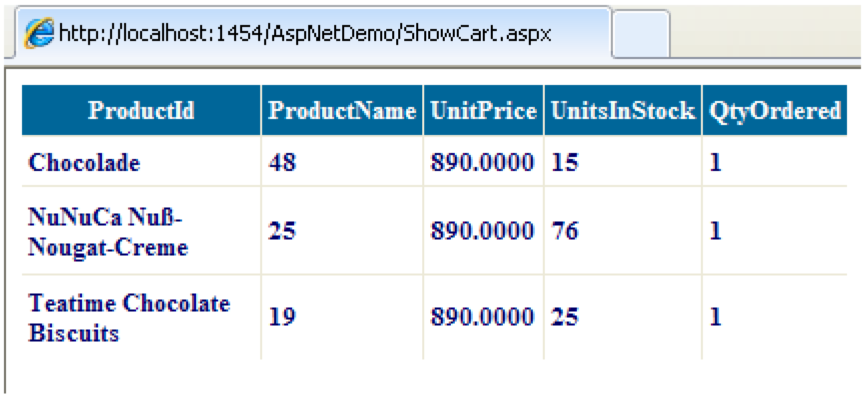


http://localhost:1454/AspNetDemo/ProductCatalogue...

Chocolade Id: 48 Stock Balance: 15 Price: 890.0000 Add To Cart	Gumbär Gummibärchen Id: 26 Stock Balance: 15 Price: 890.0000 Add To Cart	Maxilaku Id: 49 Stock Balance: 10 Price: 890.0000 Add To Cart
NESCAFE Id: 154 Stock Balance: 25 Price: 890.0000 Add To Cart	NuNuCa Nuß-Nougat-Creme Id: 25 Stock Balance: 76 Price: 890.0000 Add To Cart	Pavlova Id: 16 Stock Balance: 29 Price: 890.0000 Add To Cart
Schoggi Schokolade Id: 27 Stock Balance: 49 Price: 890.0000 Add To Cart	Scottish Longbreads Id: 68 Stock Balance: 6 Price: 890.0000 Add To Cart	Sir Rodney's Marmalade Id: 20 Stock Balance: 40 Price: 890.0000 Add To Cart

Note: For 5 minutes the products will be shown from cache. To check this, connect to back end sql server using SQL Management Studio and change the stock balance of a product. Refresh this page couple of times the page will still show the old stock due to cache. Wait for around 5 minutes to elapse and refresh the page the updated stock for the product will be visible because of cache being refilled due to expiration.

31. Add few products to cart and then click show cart on the CategoryList.aspx page. The purchases will be shown.



http://localhost:1454/AspNetDemo/ShowCart.aspx

ProductId	ProductName	UnitPrice	UnitsInStock	QtyOrdered
Chocolade	48	890.0000	15	1
NuNuCa Nuß-Nougat-Creme	25	890.0000	76	1
Teatime Chocolate Biscuits	19	890.0000	25	1

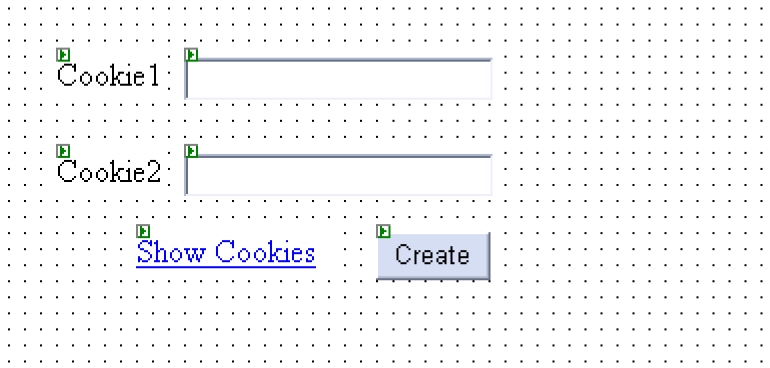
Assignment 3 To Do::

In the ShowCart.aspx add a Button with text “Confirm Purchase” On Click save the Cart details to the database and inform user by displaying appropriate message. Create a separate database table to store the cart details Also change the code of “Add to Cart” in such a way that if end user adds the same product to the cart it wont be added twice into the cart.

Lab 13. Working with Cookies Concept

Description	• In this lab we will have 2 pages. One page will create cookies and the other page will consume those cookies to print it on the page.
Objective	To learn: • How to create cookies • How to access cookies that are already created • To know whether a cookie exists • About Cookie expiration
Time	30 Mins

1. Create a page called CreateCookies.aspx as below
   a. 2 Labels, 2 Textboxes, 1 Button and 1 Hyperlink



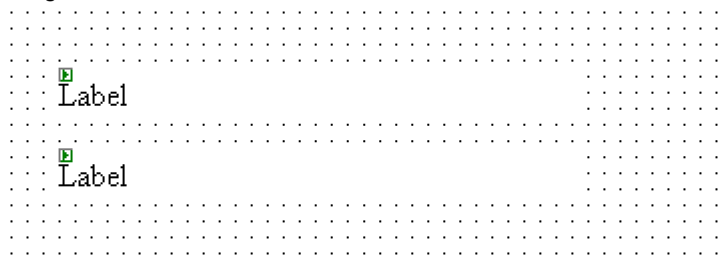
2. Set Hyperlink Properties as
   a. Text : Show Cookies
   b. NavigateUrl: ShowCookies.aspx

3. The Create button click code

```
private void btncreate_Click(object sender, System.EventArgs e)
  {

      HttpCookie c1 = new HttpCookie("c1", TextBox1.Text);
      HttpCookie c2 = new HttpCookie("c2");

      c2["name"] = TextBox1.Text;
      c2["lastname"] = TextBox2.Text;

      c1.Expires = System.DateTime.Now.AddMinutes(3);

      Response.Cookies.Add(c1);
      Response.Cookies.Add(c2);
```

```
        }
```

4. The ShowCookies.aspx page is designed below
   a. Drag 2 Labels on it.



5. ShowCookies.aspx page Load Code

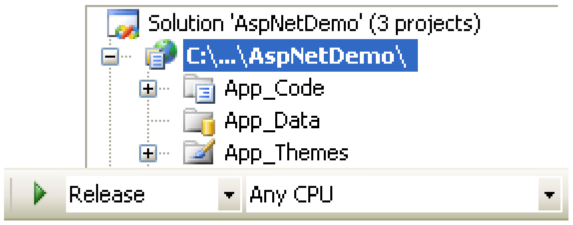
```
        protected void Page_Load(object sender, EventArgs e)
    {
            HttpCookie hc;
            hc = Request.Cookies["c1"];
            if (hc != null)
            {
                Label1.Text = "Cookie 1 : " + hc.Value;
            }
            else
            {
                Label1.Text = "Cookie not created";
            }

            hc = Request.Cookies["c2"];
            if (hc.HasKeys())
            {
         foreach (string str in hc.Values)
                {
           Label2.Text += "<li>" + str + " is " + Request.Cookies["c2"][str];
                }
            }
    }
```

6. Make CreateCookies.aspx page as start page and Run
   a. Enter Some Text and Click “Create” Button
   b. Then click Show Cookies Link. We will see that the data is carried to next page through cookies.
   c. Do not close Browser wait for 4-5 minutes and Press Refresh you will see that details of cookie c1 is lost.

Lab 14. Creating a Setup for Asp.net Web Application

Description	• In this lab we will be creating a setup for the asp.net site created.
Objective	To learn: • How to create a setup
Time	60 Mins

1. Select the Asp.net site from solution explorer and select Release as the build configuration in the toolbar above



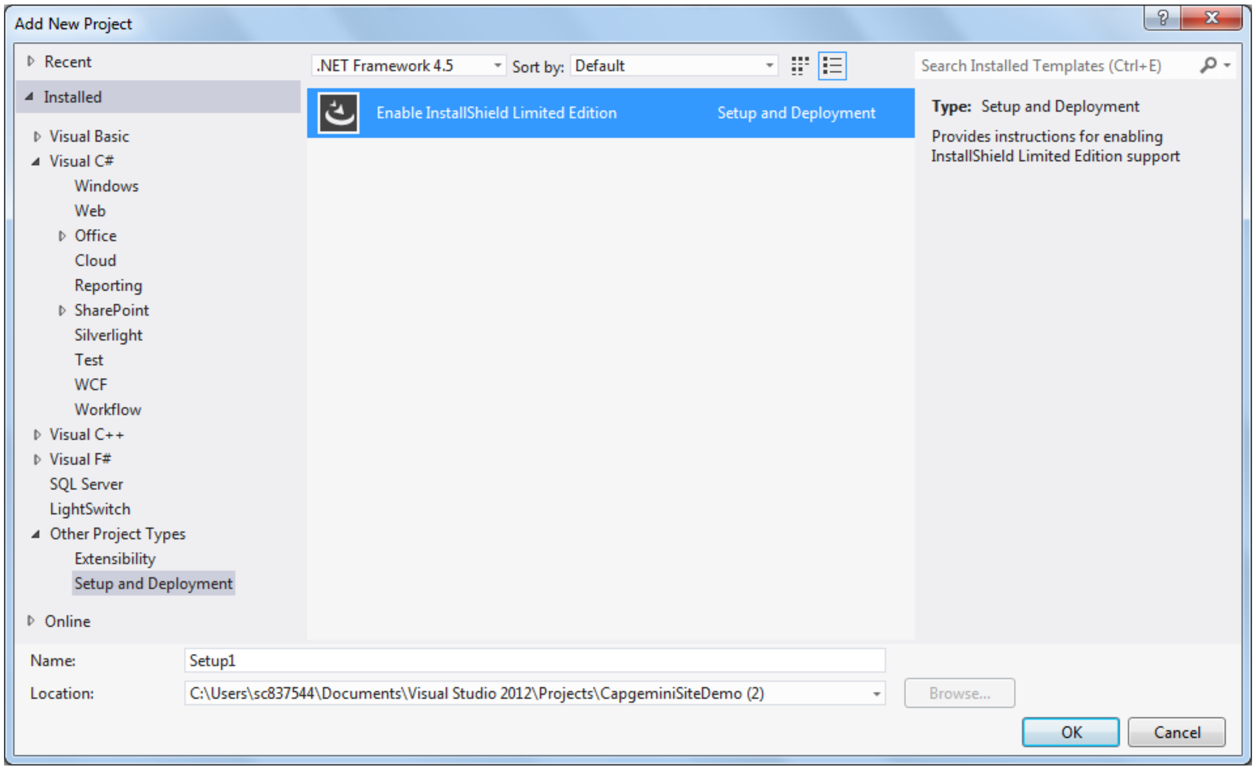
Solution 'AspNetDemo' (3 projects)
C:\...\AspNetDemo\
App_Code
App_Data
App_Themes

Release | Any CPU

2. Open web.config and change debug = “false” in compilation begin tag

   ```
   <compilation debug="false">
   ```

3. Build the Asp.Net web site once
4. Add a Web Setup project File => Add New Project; select “setup and deployment” node from left side and web setup template from right side pane. Name project as IGatePatni.WebSiteSetup (Specify the proper physical folder location where you want your web setup to generated)


Add New Project

Recent
Installed
Visual Basic
Visual C#
Windows
Web
Office
Cloud
Reporting
SharePoint
Silverlight
Test
WCF
Workflow
Visual C++
Visual F#
SQL Server
LightSwitch
Other Project Types
Extensibility
Setup and Deployment
Online

.NET Framework 4.5 Sort by: Default

Search Installed Templates (Ctrl+E)

Enable InstallShield Limited Edition — Setup and Deployment

Type: Setup and Deployment

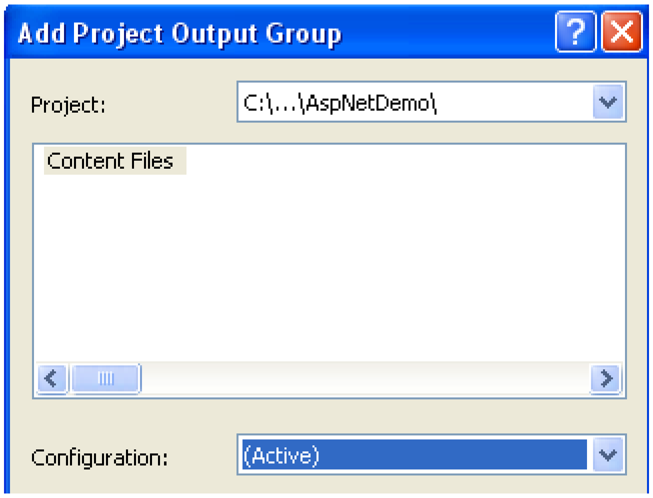
Provides instructions for enabling InstallShield Limited Edition support

Name: Setup1

Location: C:\Users\sc837544\Documents\Visual Studio 2012\Projects\CapgeminiSiteDemo (2)

Browse... OK Cancel

5. In the Setup “Filesystem” window, right click Web Application Folder; select Add => Project Ouput, Select Asp.net web site from drop down and select content files. click ok.


Add Project Output Group

Project: C:\...\AspNetDemo\

Content Files

Configuration: (Active)

6. Content Files entry will be seen added in file system window in right pane.
7. To see what is “Content Files” right click the node and select “outputs”.

8. Right Click the Web SetUp Project in solution explorer and click Rebuild the setup file will be generated.Go to the build output directory of this project to see the Setup MSI generated.

This Setup can has to be executed on the IIS web server to get the application installed.

Note: This Setup creates a “updateable” deployment of Web Site. That is the setup copies not only “.aspx” design files but “.aspx.cs” code files which can be modified.

Lab 15. Creating a non updateable deployment files for Asp.net Web Application using aspnet_compiler utility

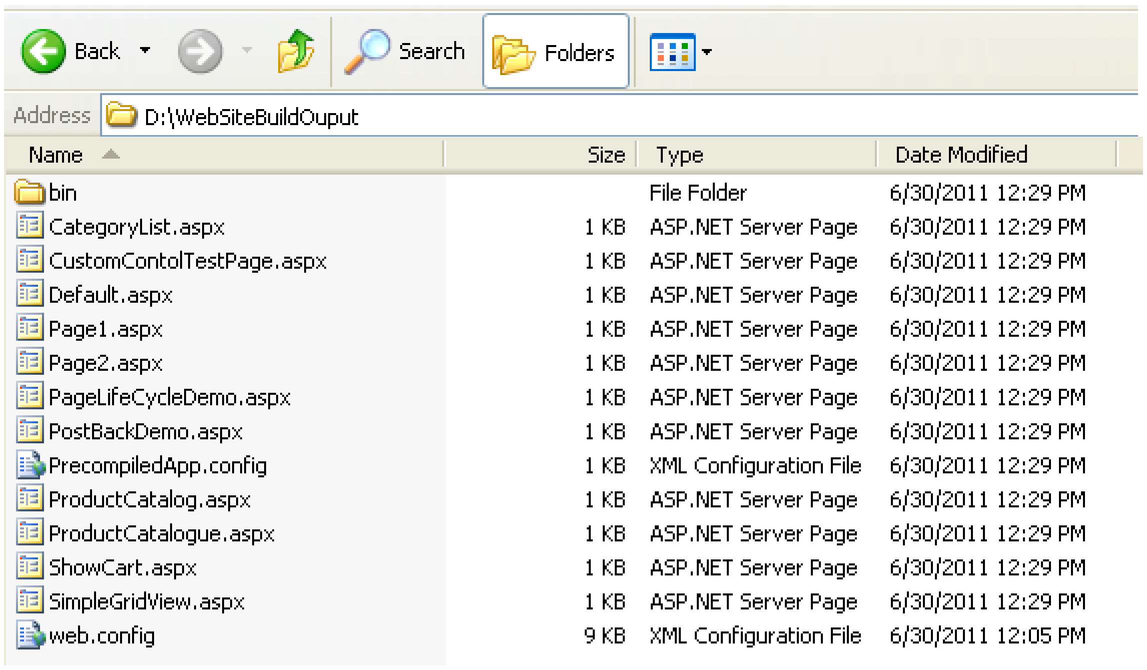
Description	• In this lab we will be generating non-updateable ready to deploy build output.
Objective	To learn: • How to use aspnet_compiler utility
Time	30 Mins

1. Create a separate folder to store the build output of the asp.net web site. (for eg D:\WebSiteBuildOuput)
2. Go to Visual Studio 2008 command prompt. Execute the compiler utility as shown:

   aspnet_compiler -v /AspNetDemo -p C:\AspNetDemo D:\WebSiteBuildOuput

   where "C:\AspNetDemo" is the folder where asp.net web site source files exist.
   and “D:\WebSiteBuildOuput” is where build output is to be generated

3. Move to this above output folder to see the build output to be deployed on the Web Server.



Name	Size	Type	Date Modified
bin		File Folder	6/30/2011 12:29 PM
CategoryList.aspx	1 KB	ASP.NET Server Page	6/30/2011 12:29 PM
CustomContolTestPage.aspx	1 KB	ASP.NET Server Page	6/30/2011 12:29 PM
Default.aspx	1 KB	ASP.NET Server Page	6/30/2011 12:29 PM
Page1.aspx	1 KB	ASP.NET Server Page	6/30/2011 12:29 PM
Page2.aspx	1 KB	ASP.NET Server Page	6/30/2011 12:29 PM
PageLifeCycleDemo.aspx	1 KB	ASP.NET Server Page	6/30/2011 12:29 PM
PostBackDemo.aspx	1 KB	ASP.NET Server Page	6/30/2011 12:29 PM
PrecompiledApp.config	1 KB	XML Configuration File	6/30/2011 12:29 PM
ProductCatalog.aspx	1 KB	ASP.NET Server Page	6/30/2011 12:29 PM
ProductCatalogue.aspx	1 KB	ASP.NET Server Page	6/30/2011 12:29 PM
ShowCart.aspx	1 KB	ASP.NET Server Page	6/30/2011 12:29 PM
SimpleGridView.aspx	1 KB	ASP.NET Server Page	6/30/2011 12:29 PM
web.config	9 KB	XML Configuration File	6/30/2011 12:05 PM

Note: The “.aspx.cs” files are not there. In this build output all “.aspx.cs” as well as “.aspx” files are also compiled so that application is completely prepcompiled and there is not additionally delay at runtime.

Lab 16. Working with Forms Authentication

Description	• In this lab we will secure a asp.net web page from anonymous access using forms authentication.
Objective	To learn: • Using Forms Authentication a. Configuring SQL Server b. Forms Authentication configuration in web.config c. Using Signup and Signin built-in login controls
Time	90 Mins

1. Configure the SQL Server Database for the Forms Authentication
   a. Launch Windows Explorer, goto C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727. (*Considering C: is your primary partition or OS Install Drive*)
   b. Locate and execute **aspnet_regsql.exe** utility.
      i. Click next to screen 1
      ii. Select Configure SQL Server …
      iii. Over here specify the SQL Server Name (*Machine Name where SQL is installed your local system or some on the network*)
      iv. Specify Win Authentication if your current windows user has rights OR select SQL Authentication and give userid and password.
      v. Let the database be default. Wizard creates aspnetdb by default.
      vi. Click Next and again Click Next
      vii. You should get the success message. Click Finish
2. Now, Copy the instruction below in the web.config just below <system.web>

```
        <membership>
    <providers>
        <clear/>
      <add name="AspNetSqlMembershipProvider"
type="System.Web.Security.SqlMembershipProvider, System.Web, Version=4.5, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a" connectionStringName="MyServer"
enablePasswordRetrieval="false" enablePasswordReset="true" requiresQuestionAndAnswer="true"
applicationName="/" requiresUniqueEmail="false" passwordFormat="Hashed"
maxInvalidPasswordAttempts="5" minRequiredPasswordLength="7"
minRequiredNonalphanumericCharacters="1" passwordAttemptWindow="10"
passwordStrengthRegularExpression="" />
    </providers>
        </membership>
```

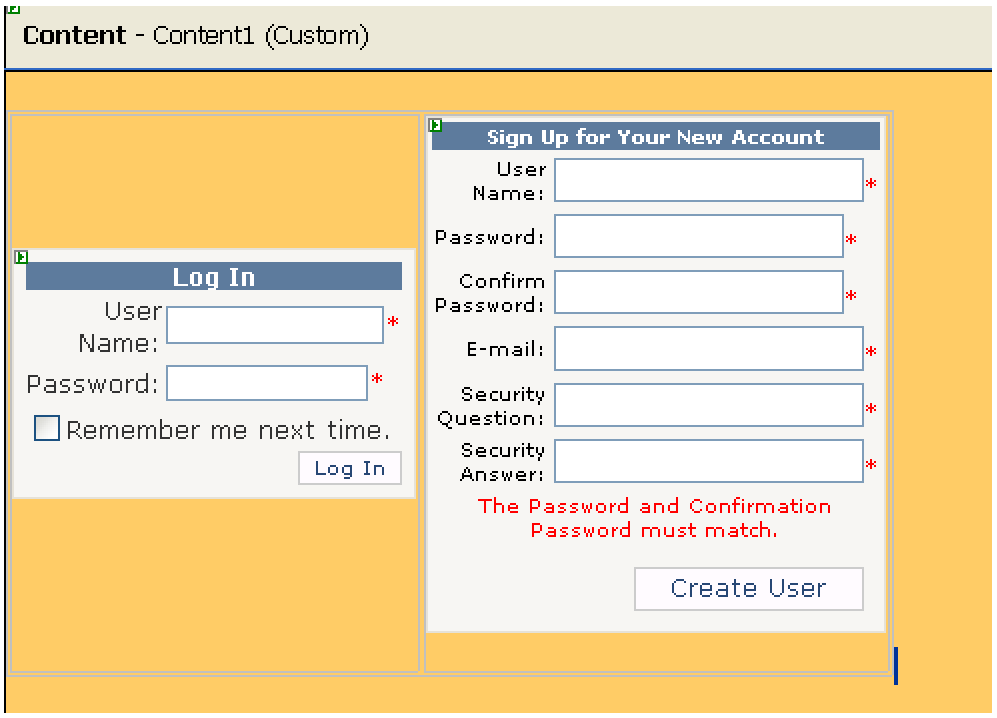
3. Now copy the instruction below in the web.config just above <system.web>

```
<connectionStrings>
   <add name="MyServer"
        connectionString="server=ndamssql\sqlilearn;database=aspnetdb;user
        id=sqluser;password=sqluser;" providerName="System.Data.SqlClient" />
 </connectionStrings>
```

Note: You may have to substitute your **server name** and **user id** and **password** that are valid.

What have we done till far in this labe is that we have configured the Membership Provider (*see type Attribute* of <add …>) to connect to proper SQL Server which has the database configured to store user credentials information and there roles membership details.

4. Lets add a Login Page
   a. Website => Add New Item;Select Web Form, Name it as LoginPage.aspx. Base it on MyMaster1.master.
   b. In the Content Control add a 1 Row 2 Column Table; Table => Insert Table
   c. In the Content Control in the first cell Drag Login Control from Toolbox
5. From the smart tag of login control select Auto format => professional click OK



6. Next to the Login Control, Drag CreatUserWizard Control from Toolbox onto Content Control in the 2nd cell . Goto Smart Tag, in Autoformat Select Professional

7. In the web.config add the instruction below (*if* <authentication …> tag *already* exists change it as below)

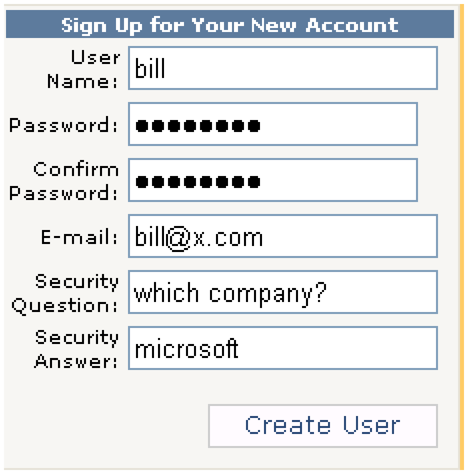
```
        <authentication mode="Forms">
    <forms loginUrl="LoginPage.aspx" cookieless="AutoDetect" name="usercookie"
         defaultUrl ="Main.aspx"/>
    </authentication>
    <authorization>
      <deny users="?"/>
      <allow users="*"/>
    </authorization>
```

Add few more Pages:

8. Website => Add New Item Select Web Form Name it as Main.aspx.Base it on MyMaster1.master.
   a. Drag a label in Content Control of the Main.aspx and set the text as "This is Main Page" increase the fontsize and make it bold.
9. Lets add one more page AnotherPage.aspx
   a. Drag a label in Content Control of the AnotherPage.aspx and put the text as "This is Another Page" increase the fontsize and make it bold.

.

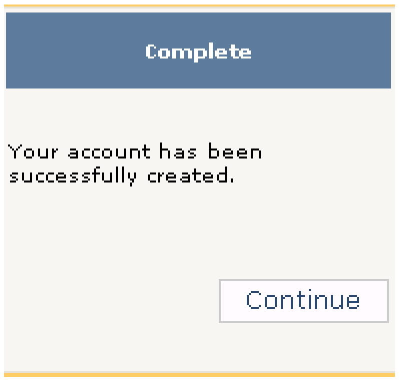
10. Create an Account first: Make login.aspx as startup and run website. Enter details to signup similar to as below:



Sign Up for Your New Account

User Name:	bill
Password:	●●●●●●●●
Confirm Password:	●●●●●●●●
E-mail:	bill@x.com
Security Question:	which company?
Security Answer:	microsoft

11. Click Create User . It should succeed



Complete

Your account has been successfully created.

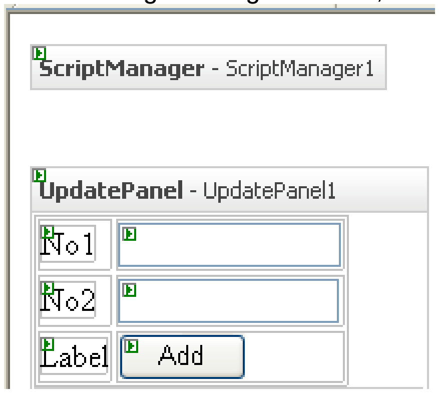
12. Close Browser. And run the Login.aspx and login with credentials. we will be redirected to Main.aspx (The default redirect in web.config). Close the browser
13. Make AnotherPage.aspx as startup and run
14. We will be redirected to Login.aspx. We cant access this page until we get logged in

15. So again login with credentials. we will be redirected to AnotherPage.aspx
16. So we can’t enter site unless you login !

Lab 17. Using Update Panel and Script Manager for Partial Updates

Description	• In this lab we use built-in core AJAX controls Update Panel and Script Manager to achieve partial page update.
Objective	To learn: • How to use Update Panel and ScriptManager Control to achieve Partial Page Update
Time	2 Hr

1. Add a new ASPX Web Page
2. On the Page
   a. Drag an UpdatePanel Control from Toolbox (AJAX Extensions Section)
   b. Put a HTML table of 2 rows and 2 cols inside UpdatePanel
      i. Click inside UpdatePanel , Table => Insert Table
   c. Drag an Drag 3 Labels , 2 TextBoxes (txtno1, txtno2) and 1 Button (btnadd)



3. Goto btnadd click event, write the following code

```
protected void btnadd_Click(object sender, EventArgs e)
{
     int r = int.Parse(txtno1.Text) + int.Parse(txtno2.Text);
     Label3.Text = r.ToString();
}
```

4. Make the page as startup and Run the page
   a. Enter nos and click Add
   Notice that the entire page is not submitted and partial rendering is done.

5. Now from toolbox (AJAX Extensions Section)
   a. Drag UpdateProgress Control inside UpdatePanel below the HTML Table.
   b. Inside updateprogress control drag a label
      i. Set Text property to “Calculating …”
      ii. Change ForeColor Property to Orange
6. Change btnadd click code to

```
    protected void btnadd_Click(object sender, EventArgs e)
    {
        System.Threading.Thread.Sleep(3000);
        int r = int.Parse(txtno1.Text) + int.Parse(txtno2.Text);
        Label3.Text = r.ToString();
  }
```

7. Make the page as startup and Run the page
   a. Enter nos and click Add

   Notice that till the response comes the status is shown to user through the UpateProgess Content.

Assignment 4 To Do::

AjAX enable the web page GridViewDemoPage.aspx from the above labs.
Hint: Drag scriptmanager control and update panel. And Embed the entire Grid (grdcustomer) in the update panel for partial updates.