Experiment 11 – ASP.net Updation/Deletion via GridView Control and Insertion

Objective

- Familiarize you with Delete through Grid View and Insert via ASP.net web form.

Delete via GridView

As in the last lab we used the Grid Views to display the data from an SQL table , The Grid View can also be used to delete data from that table , following exercise will demonstrate how to:

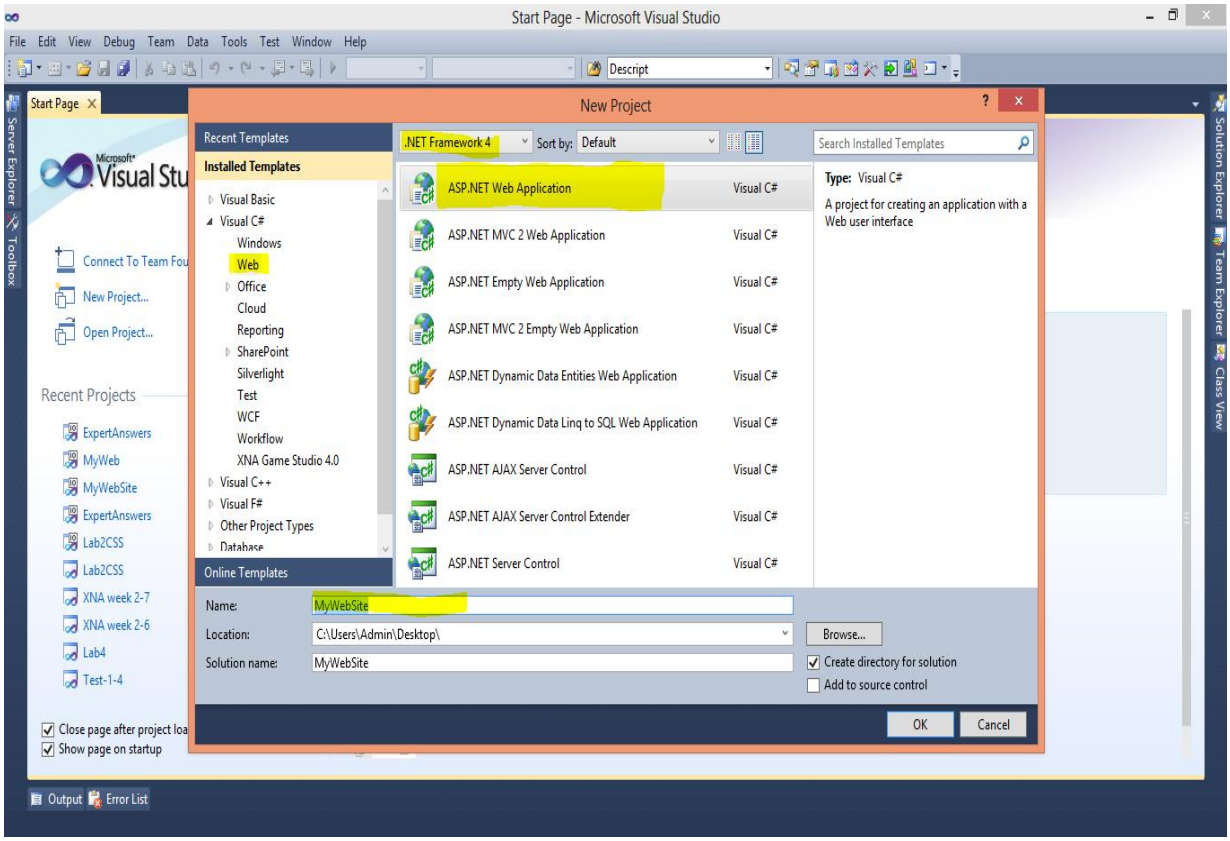
1) (If not done in last lab) Create a Data base named <your rollnumber>
2) (If not done in last lab)Create a table named Items and insert values in it using following queries

```
--Create table
Create table items
(ItemNo int,
ItemName varchar(15),
TotalUnits int
)
go

--insert values
Insert into items
values
(1,'Soap', 10 )
,(2,'Handwash', 20)
,(3,'Shampoo',5)
go
```

3) Now Create the following procedure to Delete the Item from Items table , we will call this procedure from asp.net to delete tuples from Items table

```
Create PROCEDURE [dbo].[deleteItem]
      @ID int
AS
BEGIN
      SET NOCOUNT ON;

DELETE FROM Items WHERE ItemNo = @ID
END
```

4) Now in Visual Studio Create a New web project using settings as show in Figure

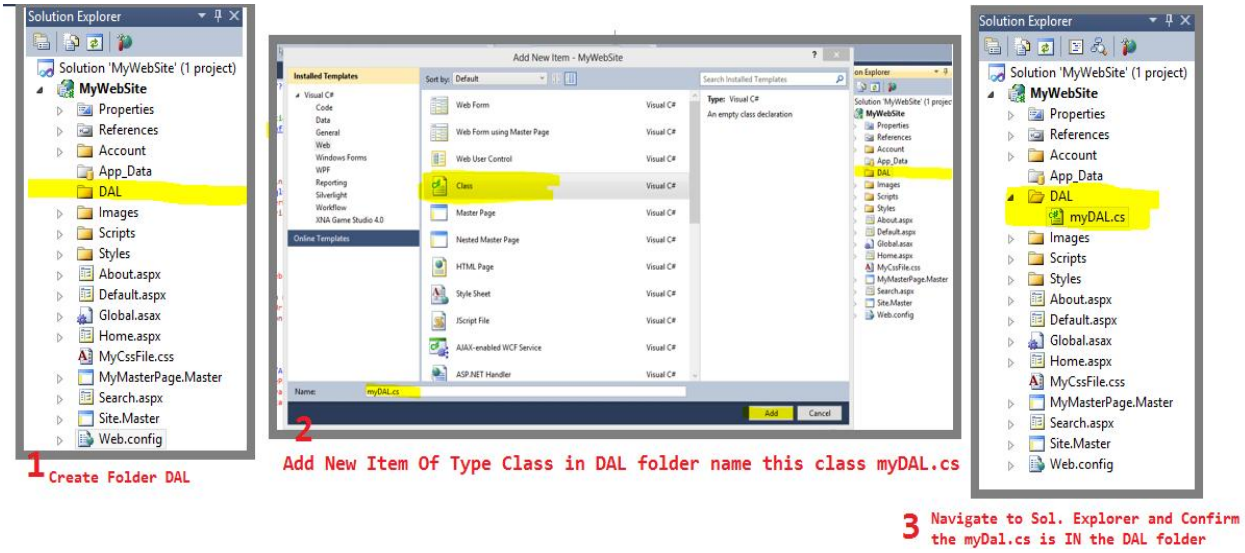


5) Follow the Steps of last manual to create a Master Page (briefed as follows)

- Add The CSS files (MyCSSFile.CSS given in Resource Folder along with Manual) in your Project using Add Existing Item
- Create new Folder named Images and add all the images given in Resource/Images
- Add Master Page in you Project , using Add New Item Option from Solution Explorer Name this Master Page as MyMasterPage.Master
- Drag and Drop CSS file in header tag of Master Page, After that Open the MasterPage_Body.txt file given in Resources and Copy All the contents , Replace everything inside the Body tags of MyMasterPage with this content

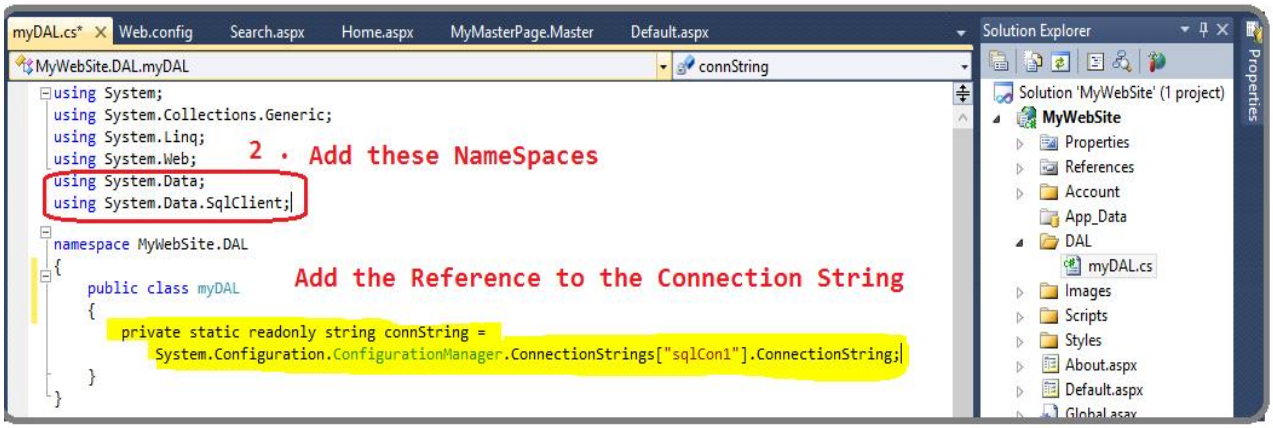
6) Open your Web Config file and add connectivity String in it (as done in last lab, consult lab11 Manual for details). Change the Initial Catalog to <your rollnumber> (your current DB) for this exercise

7) Create DAL folder and add new myDAL.cs file in it



8) Open the myDal.cs file and Add the Reference to Connection String plus, Name Spaces for SQL and DataSets

```
Using System.Data;
Using System.Data.SqlClient;

Private static readonly string connString =
System.Configuration.ConfigurationManager.ConnectionStrings["sqlCon1"].Connecti
onString;
```

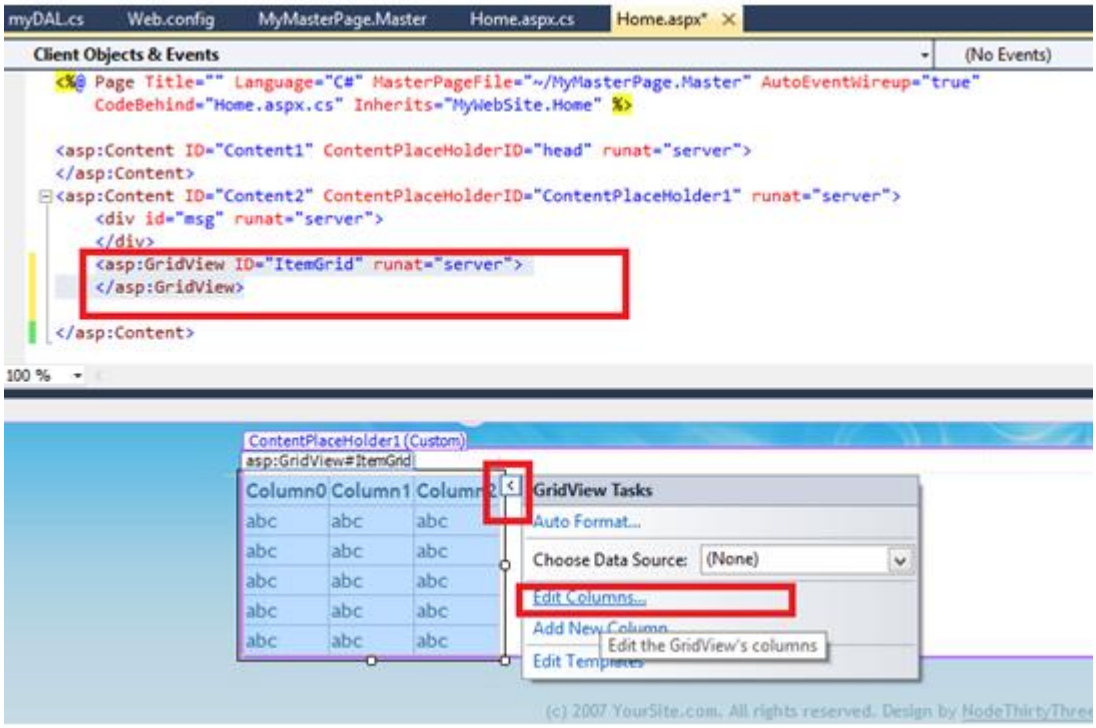


9) Copy Paste following 2 functions SelectItem() and DeleteItem() in myDal.cs file

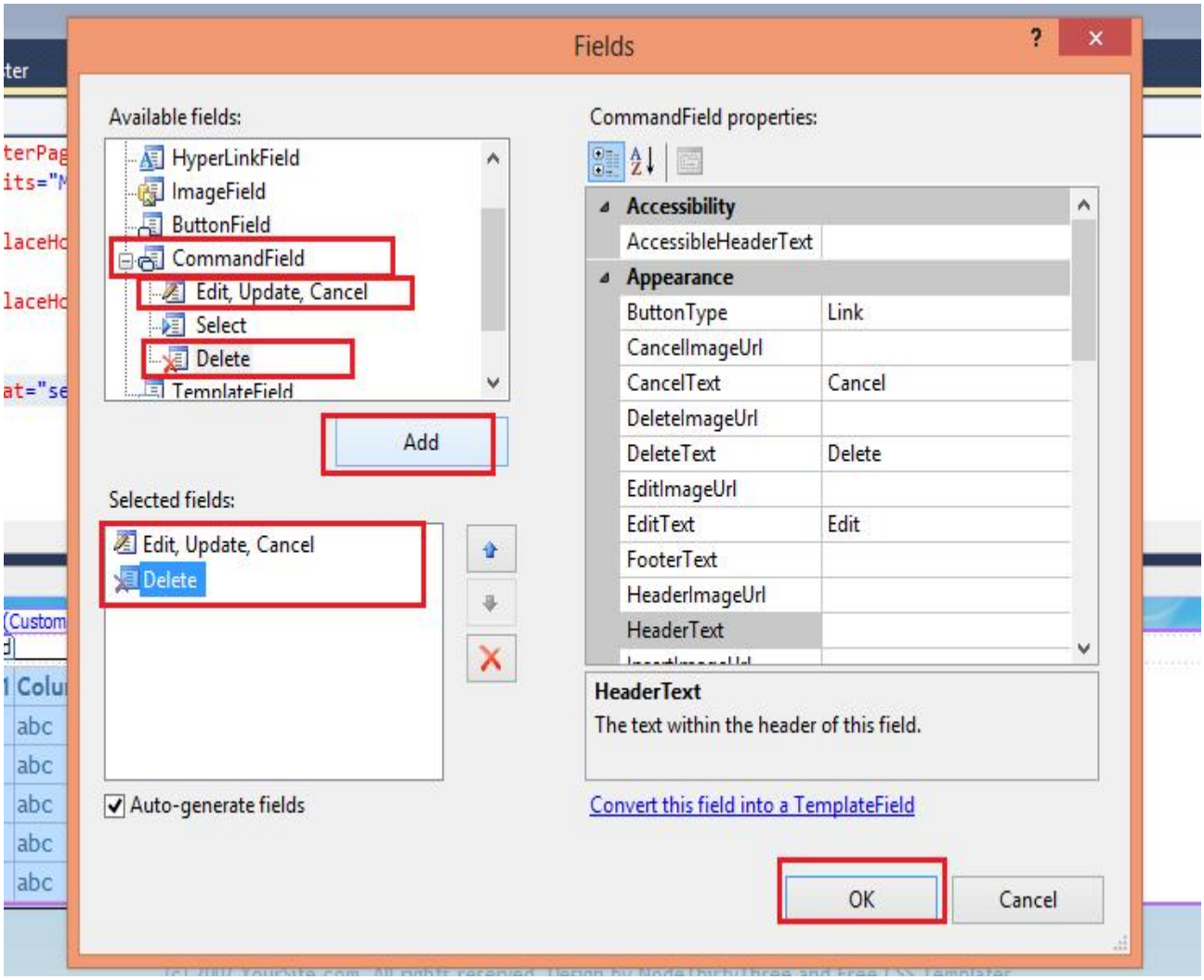
```
public DataSet SelectItem() //to get the values of all the items from table Items and return
the Dataset
        {

DataSet ds = new DataSet(); //declare and instantiate new dataset
SqlConnection con = new SqlConnection(connString); //declare and instantiate new SQL
connection
con.Open(); // open sql Connection
SqlCommand cmd;
try
            {
cmd = new SqlCommand("Select * from Items", con);  //instantiate SQL command
cmd.CommandType = CommandType.Text; //set type of sqL Command
using (SqlDataAdapter da = newSqlDataAdapter(cmd))
                {
da.Fill(ds); //Add the result  set  returned from SQLCommand to ds
                }
            }
catch (SqlException ex)
            {Console.WriteLine("SQL Error" + ex.Message.ToString());            }
finally
            {con.Close();    }

return ds; //return the dataset

        }
//////////////////////////////////////////////////
public int DeleteItem(int id)
        {
            SqlConnection con = new SqlConnection(connString);
            con.Open();
            SqlCommand cmd;
            int result = 0;
            try
            {
                cmd = new SqlCommand("deleteItem", con);
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.Parameters.Add("@ID", SqlDbType.Int).Value = id;
                result = cmd.ExecuteNonQuery();
            }
            catch (SqlException ex)
            {    Console.WriteLine("SQL Error" + ex.Message.ToString());
            }
            finally
            {     con.Close();

            }

              return result;
        }
```

Deletion in GridView

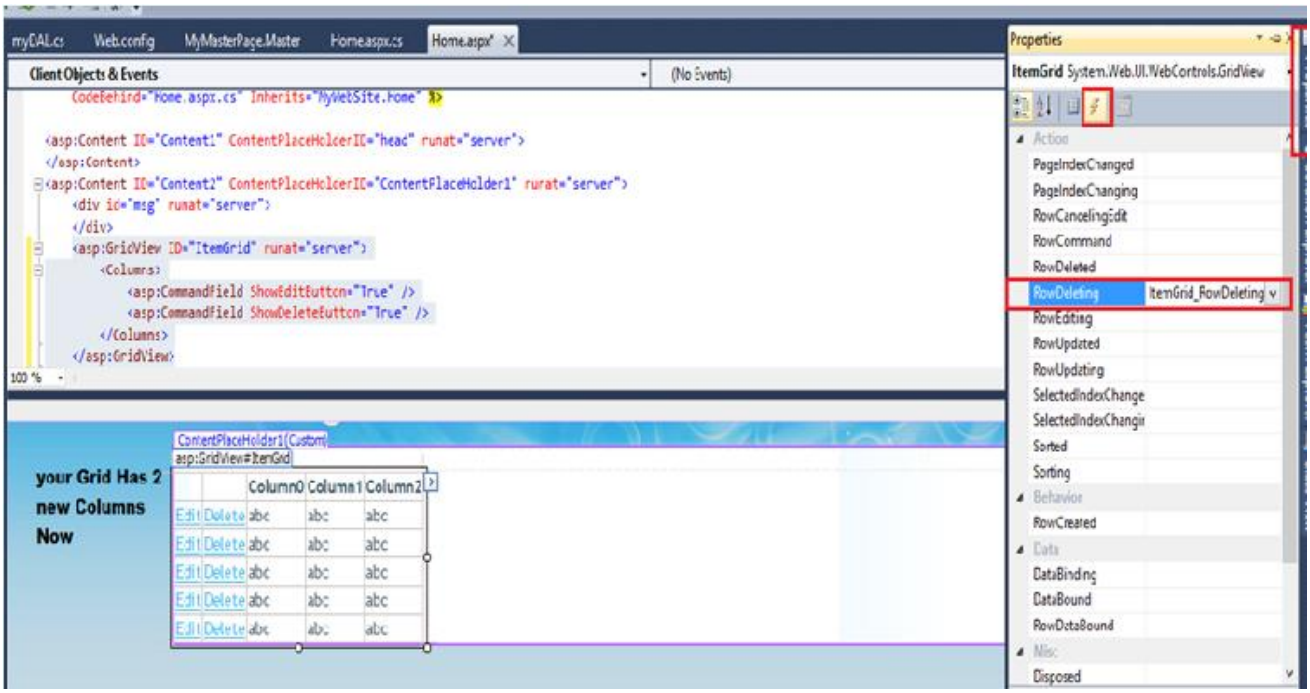
10) Add New Grid in your home.aspx page with ID ItemGrid , Click the Forward Arrow button and Select Edit Columns



The following popup will appear, Expand *CommandField* and Add*Edit Update, Cancel* and *Delete* and click OK



11) Your Grid will have 2 new columns now
Keep the Grid selected and go to its properties, click on Small Lightening Icon, and Change the Value of Column *Row Deleting* to *ItemGrid_RowDeleting*



12) Open the Home.aspx.cs file and replace pageload() function with this

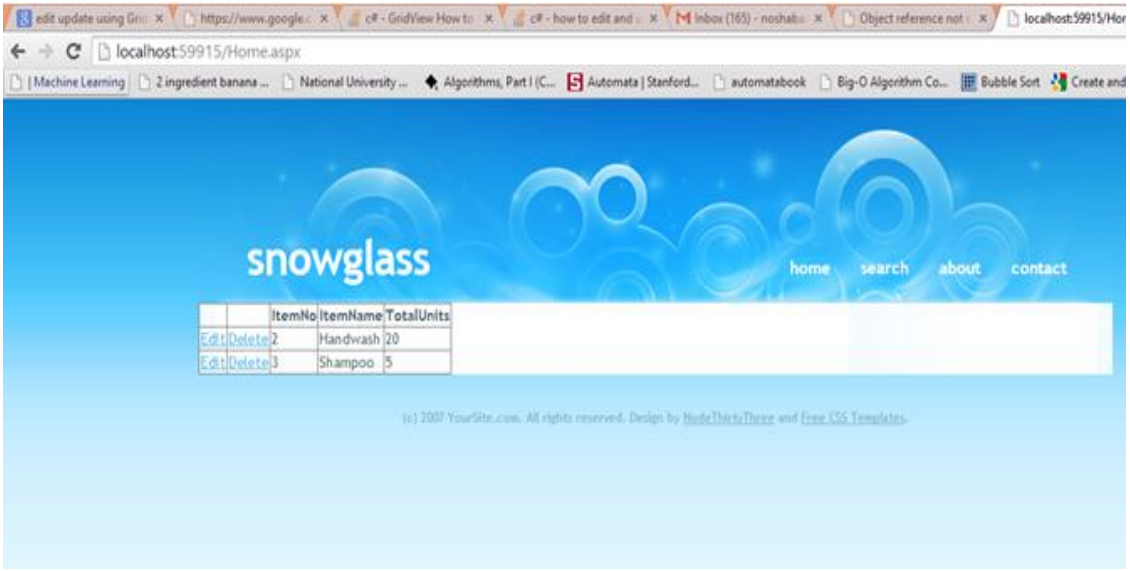
```
protectedvoidPage_Load(object sender, EventArgs e)
        {
LoadGrid(); // fill the grid every time page loads

        }
```

And add the following two functions LoadGrid() and `ItemGrid_RowDeleting`

```
public void LoadGrid()
        {
        myDAL objMyDal = new myDAL();
   ItemGrid.DataSource = objMyDal.SelectItem();//seting data source for this Grid
   ItemGrid.DataBind(); //bind the data source to this grid
        }// end of loadgrid

Protected void ItemGrid_RowDeleting(object sender, GridViewDeleteEventArgs e)
        {
myDAL objMyDal = new myDAL();
GridViewRow row = ItemGrid.Rows[e.RowIndex];
int ItemID = Convert.ToInt32(row.Cells[2].Text.ToString());
int result = objMyDal.DeleteItem(ItemID);
if (result == -1)
            {
ItemGrid.DataSource = objMyDal.SelectItem();
ItemGrid.DataBind();
            }
else
            {
string message = "No row deleted";
ClientScript.RegisterOnSubmitStatement(this.GetType(), "alert",

message.ToString());
            }

        }
```

Now execute your project, following results should appear in browser:

On Clicking Delete, the corresponding row will delete (check from SQL server as well)



Perform Updation via GridView Control

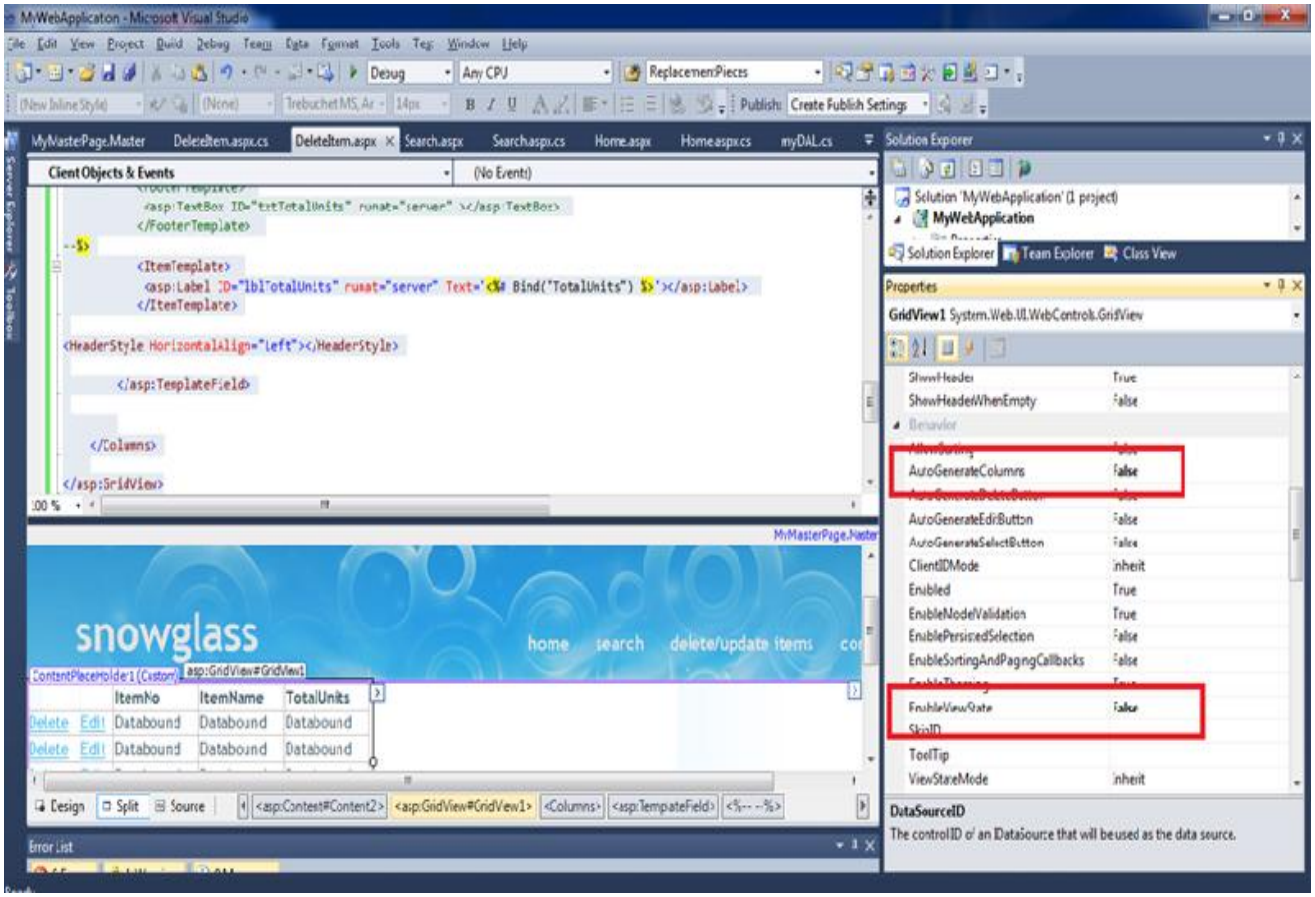
Step1:

First we will add the following code in Home.aspx page under the "<Columns>" tag of GridView we created earlier.

```
  <Columns>

<asp:CommandField ShowEditButton="True"/>
<asp:CommandField ShowDeleteButton="True"/>

<asp:TemplateField HeaderText="ItemNo" HeaderStyle-HorizontalAlign="Left">
<EditItemTemplate>
<asp:Label ID="txtItemNo" runat="server" Text='<%#
Bind("ItemNo") %>'></asp:Label>
</EditItemTemplate>

<ItemTemplate>
<asp:Label ID="lblItemNo" runat="server" Text='<%#
Bind("ItemNo") %>'></asp:Label>
</ItemTemplate>

<HeaderStyle HorizontalAlign="Left"></HeaderStyle>

</asp:TemplateField>
<asp:TemplateField HeaderText="ItemName" HeaderStyle-HorizontalAlign="Left">
       <EditItemTemplate>
       <asp:TextBox ID="txtItemName" runat="server" Text='<%#
Bind("ItemName") %>'></asp:TextBox>
       </EditItemTemplate>
       <ItemTemplate>
       <asp:Label ID="lblItemName" runat="server" Text='<%#
Bind("ItemName") %>'></asp:Label>
       </ItemTemplate>
       <HeaderStyle HorizontalAlign="Left"></HeaderStyle>
</asp:TemplateField>
```

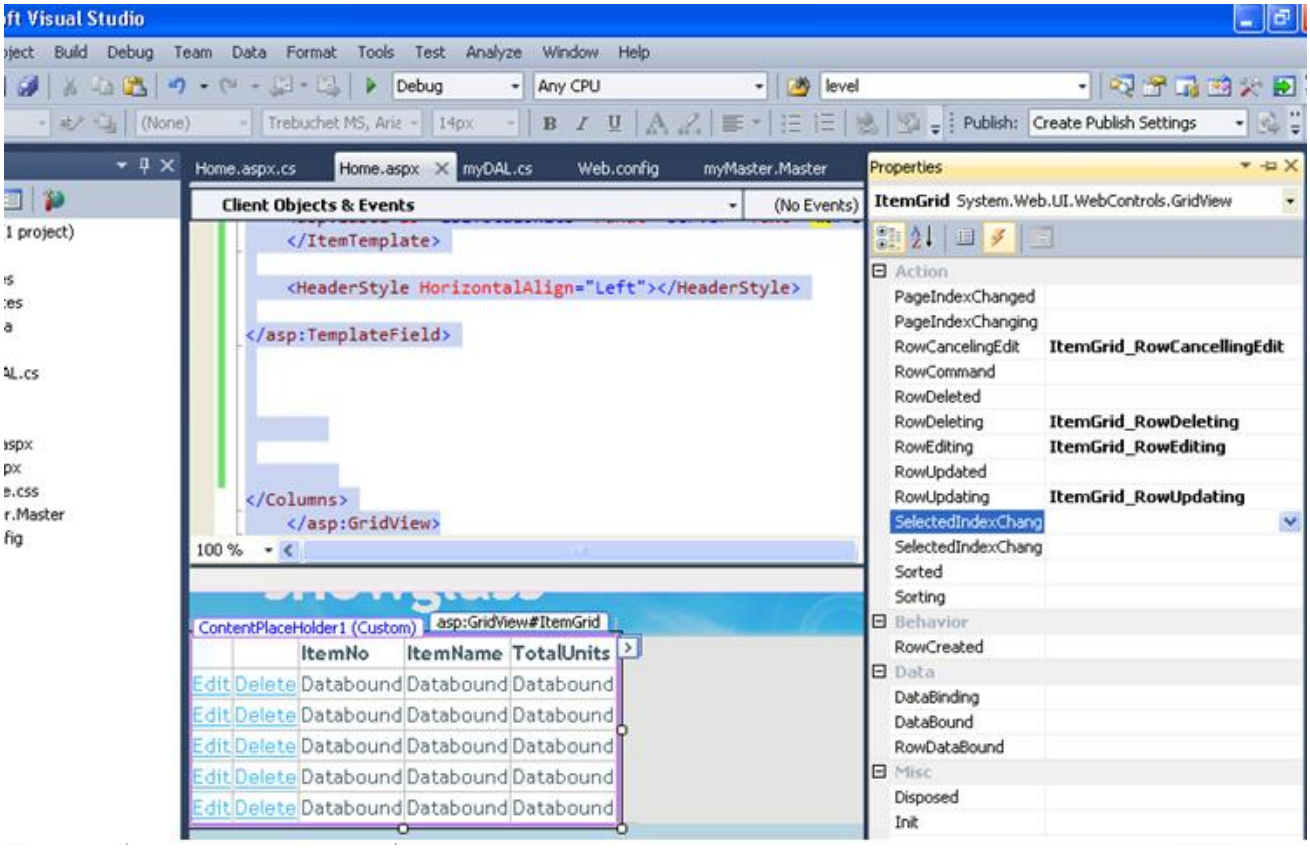
```
<asp:TemplateField HeaderText="TotalUnits" HeaderStyle-HorizontalAlign="Left">
       <EditItemTemplate>
       <asp:TextBox ID="txtTotalUnits"  runat="server" Text='<%#
Bind("TotalUnits") %>'></asp:TextBox>
       </EditItemTemplate>
       <ItemTemplate>
       <asp:Label ID="lblTotalUnits" runat="server" Text='<%#
Bind("TotalUnits") %>'></asp:Label>
       </ItemTemplate>
       <HeaderStyle HorizontalAlign="Left"></HeaderStyle>
</asp:TemplateField>

</Columns>
```

Now set the "**EnableViewState**" and "**AutoGenerateColumns**" properties of the Grid View as follows:



Step2:

Now we will set up all the events in our .aspx page which we will require for implementing Edit functionality as shown in the following screen shot:



Step3:

Now we are set to move to the BLL of this page which is the Home.aspx.cs. There we will implement all the above create event handling functions.

Replace the **Page_Load()** function with the following function:

```
Protected void Page_Load(object sender, EventArgs e)
        {            LoadGrid();
        }
```

Following are the functions which you have to copy/paste as well against all the event handling definitions created above:

```
Public void LoadGrid()
        {         myDAL md = new myDAL();
                ItemGrid.DataSource = md.SelectItem();
                ItemGrid.DataBind();
        }
protectedvoid ItemGrid_RowUpdating(object sender, GridViewUpdateEventArgs e)
        {
GridViewRow row = (GridViewRow)ItemGrid.Rows[e.RowIndex];
//==== getting the value from the respective controls=====
Label itemNo = (Label)ItemGrid.Rows[e.RowIndex].FindControl("txtItemNo");
TextBox ItemName =
(TextBox)ItemGrid.Rows[e.RowIndex].FindControl("txtItemName");
TextBox TotalUnits =
(TextBox)ItemGrid.Rows[e.RowIndex].FindControl("txtTotalUnits");
//=======================================================
int itemNoValue = Convert.ToInt32(itemNo.Text.ToString());
```

```
string itemNameValue = ItemName.Text.ToString();
int totalUnits = Convert.ToInt32(TotalUnits.Text.ToString());

//=====updating the newly entered values in database====
myDAL objMyDal = new myDAL();
            objMyDal.UpdateItem(itemNoValue, totalUnits, itemNameValue);
//=====================================================
            ItemGrid.EditIndex = -1;
            LoadGrid();
      }


protectedvoid ItemGrid_RowCancellingEdit(object sender,
GridViewCancelEditEventArgs e)
{
    ItemGrid.EditIndex = -1;
    LoadGrid();
}

protectedvoid ItemGrid_RowEditing(object sender, GridViewEditEventArgs e)
        {
            ItemGrid.EditIndex = e.NewEditIndex;
            LoadGrid();
        }
```

Step 4:

Now moving on to the DAL layer of our application, we will have to create function for Update/Edit functionality. For that copy paste the following function in to you **myDAL.cs** file.

```
public int UpdateItem(int id, int totalUnits, string itemName)
        {
SqlConnection con = new SqlConnection(connString);
con.Open();
SqlCommand cmd;
int result = 0;
try
            {
cmd = new SqlCommand("UpdateItem", con);
cmd.CommandType = CommandType.StoredProcedure;
cmd.Parameters.Add("@itemNo", SqlDbType.Int).Value = id;
cmd.Parameters.Add("@itemname", SqlDbType.VarChar).Value = itemName;
cmd.Parameters.Add("@totalUnits", SqlDbType.Int).Value = totalUnits;
                result = cmd.ExecuteNonQuery();
            }
catch (SqlException ex)
            {
            Console.WriteLine("SQL Error" + ex.Message.ToString());
            }
finally   {
            con.Close();
            }
return result;
        }
```

Step 5:

Now before executing the application, execute the below script on your database so that the required procedure for editing functionality is created:

```
CREATE PROCEDURE [dbo].[UpdateItem]
      @itemNo int,      @itemname varchar(100), @totalUnits int
AS
BEGIN SET NOCOUNT ON;
Update items set ItemName=@itemname, TotalUnits=@totalUnits
where ItemNo=@itemno;
END
```

Perform insertion

1) Change the Home.aspx page to look as follows:

Please ensure that the ID property of the three text fields are set as `TxtItmNo` , `TxtItmName and TxtUnits` respectively.

On click of the “Add New Item” button copy the following code

```
protected void insrtItem_Click(object sender, EventArgs e)
        {
int itemNoValue = Convert.ToInt32(TxtItmNo.Text.ToString());
string itemNameValue = TxtItmName.Text.ToString();
int totalUnits = Convert.ToInt32(TxtUnits.Text.ToString());

//=====updating the newly entered values in database====
myDAL objMyDal = new myDAL();
            objMyDal.insertItem(itemNoValue, totalUnits,
itemNameValue);

            LoadGrid();
        }
```

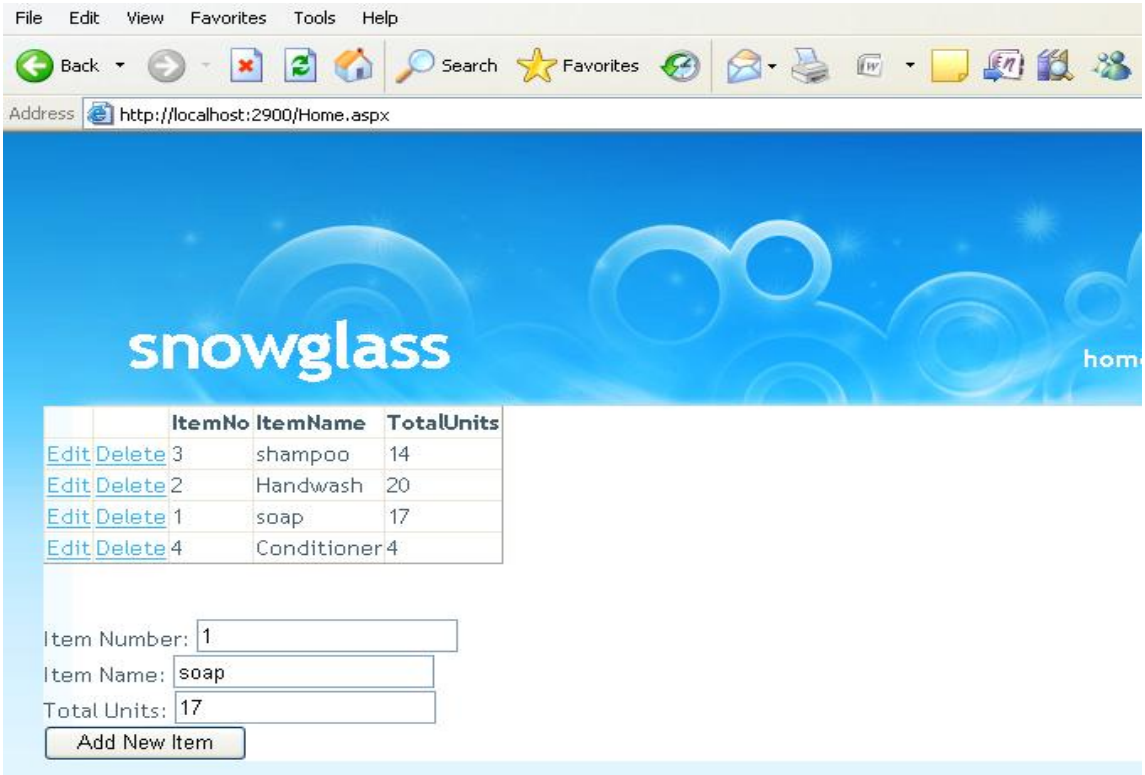
In myDAL.cs file Add the following function:

```
public int insertItem(int id, int totalUnits, string itemName)
        {
SqlConnection con = new SqlConnection(connString);
            con.Open();
SqlCommand cmd;
int result = 0;
try
            {
                cmd = new SqlCommand("InsertItem", con);
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.Parameters.Add("@itemNo", SqlDbType.Int).Value = id;
                cmd.Parameters.Add("@itemname", SqlDbType.VarChar).Value =
itemName;
                cmd.Parameters.Add("@totalUnits", SqlDbType.Int).Value =
totalUnits;

                result = cmd.ExecuteNonQuery();
            }
catch (SqlException ex)
            {
Console.WriteLine("SQL Error" + ex.Message.ToString());
            }
finally
            {
                con.Close();
            }

return result;
        }//end of insert function
```

Create the following procedure in SQL Server 2008

```
CREATE PROCEDURE [dbo].[InsertItem]
        @itemNo int,
        @itemname varchar(100),
        @totalUnits int
AS
BEGIN
        SET NOCOUNT ON;
insert into items values(@itemno, @itemname, @totalUnits );
END
```

Build and run your project. On giving the values and clicking the “Add New Item” button a new item should get
inserted.



Post Lab

Make another page which has a search bar in it. The searched item can be updated/deleted.