**EXPERIMENT 10**

#### ASP.net Insertion and Updation/Deletion via GridView Control

**Objective**

* Familiarize you with Insert via ASP.net web form and Delete/update through Grid View.

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

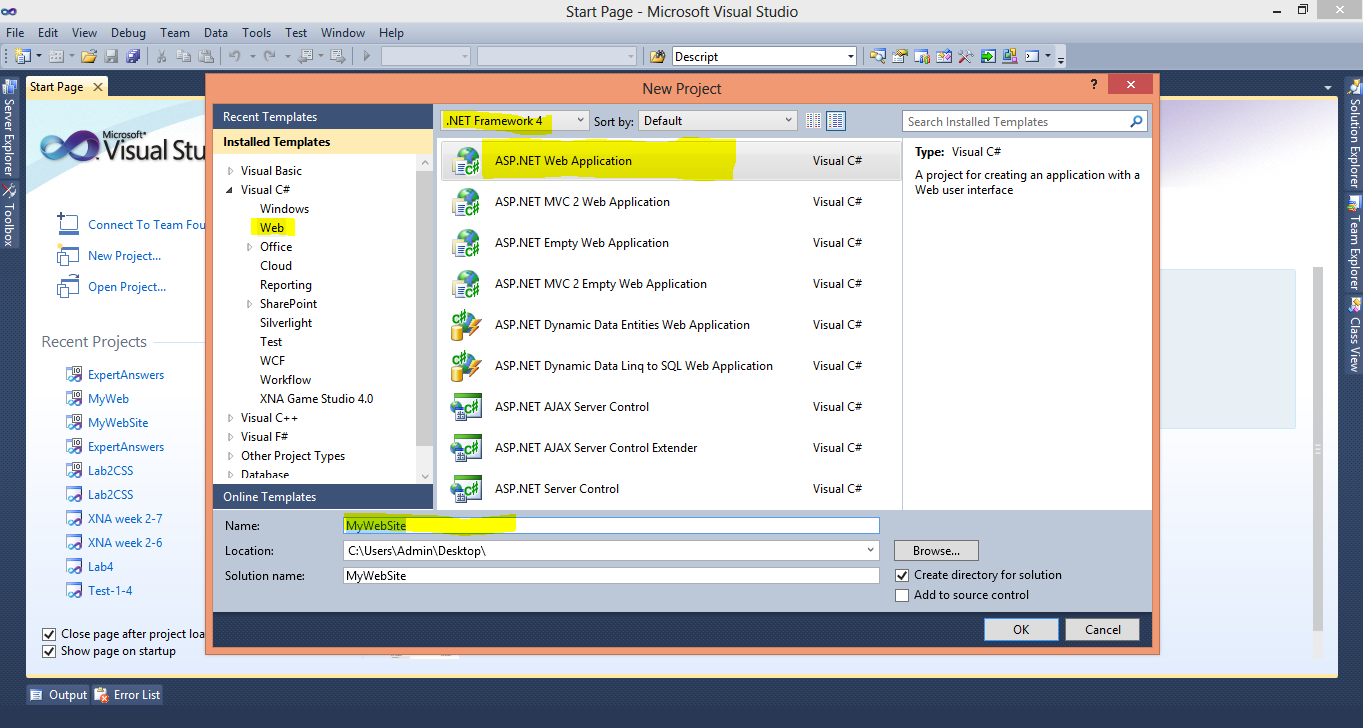
(1,'Soap', 10 )

,(2,'Handwash', 20)

,(3,'Shampoo',5)

go

1. Now in Visual Studio Create a New web project using settings as show in Figure 1

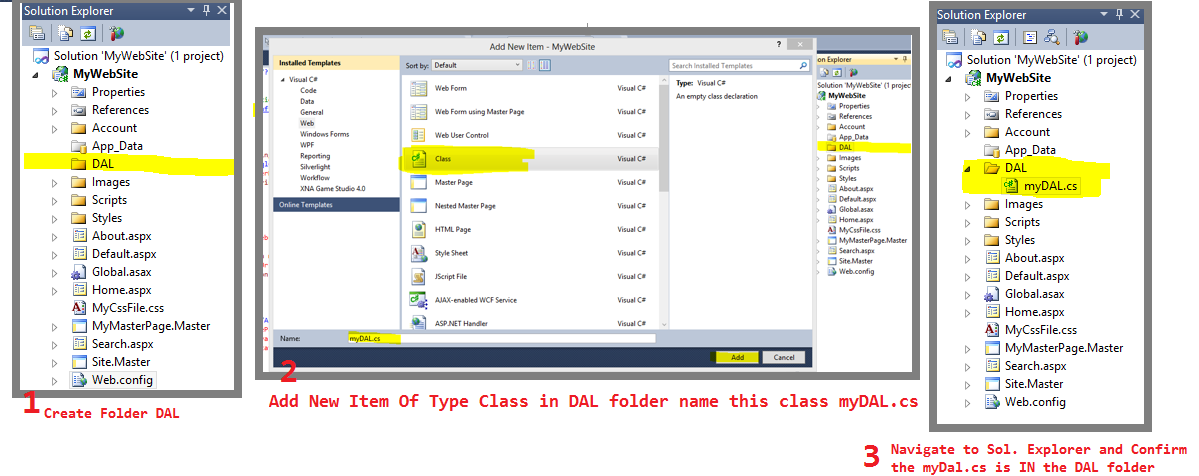


**Figure 1**

1. 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

1. 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
2. Create DAL folder and add new myDAL.cs file in it (as shown in figure 2)



**Figure 2**

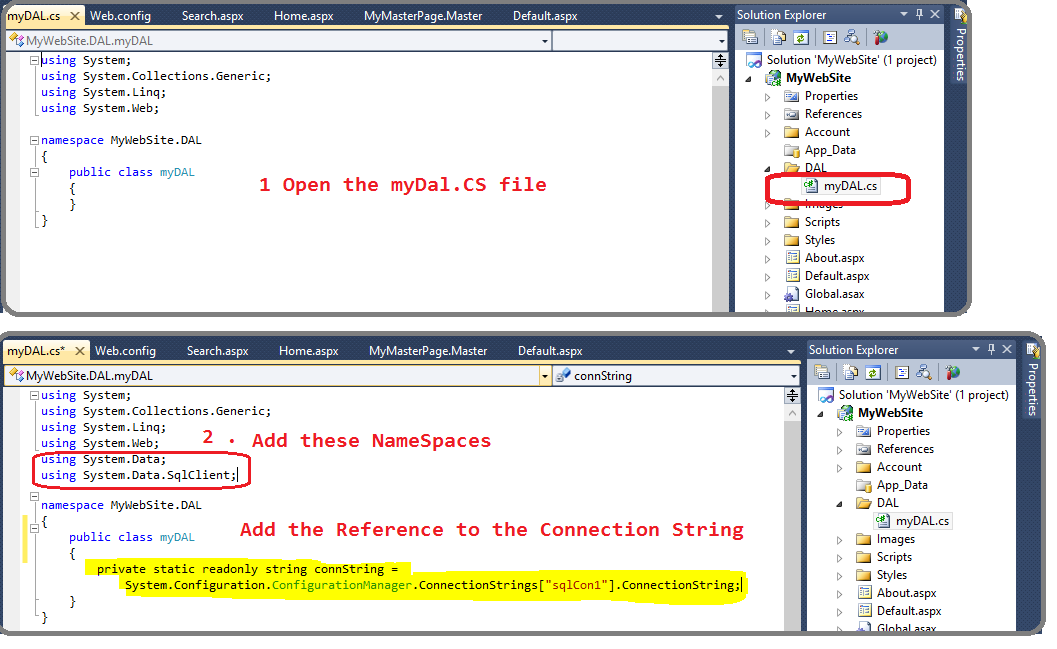
1. Open the myDal.cs file and Add the Reference to Connection String plus, Name Spaces for SQL and DataSets as shown in figure 3

using System.Data;

using System.Data.SqlClient;

private static readonly string connString =

System.Configuration.ConfigurationManager.ConnectionStrings["sqlCon1"].ConnectionString;

****

**Figure 3**

1. Copy Paste the following function SelectItem() 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 order by itemno", con); //instantiate SQL command

cmd.CommandType = CommandType.Text; //set type of sqL Command

using (SqlDataAdapter da = new SqlDataAdapter(cmd))

{

da.Fill(ds); //Add the result set returned from SQLCommand to ds

}

}

catch (SqlException ex)

{Console.WriteLine("SQL Error" + ex.Message.ToString()); }

finally

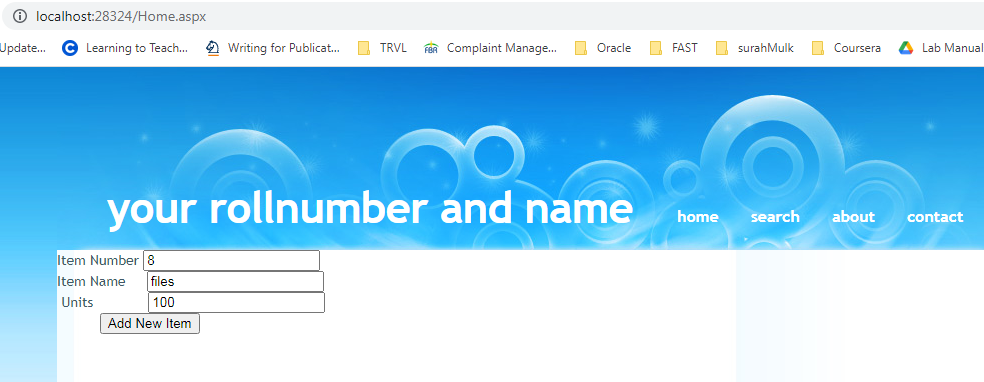
{con.Close(); }

return ds; //return the dataset

}

## Perform insertion

1. Change the Home.aspx page to look as shown in figure 1.



**Figure 1**

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

1. On click of the “Add New Item” button copy the following code in [Home.aspx.cs](http://Home.aspx.cs) as given below (shows error on insert)

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();

}

1. 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

1. Create the following procedure in MS SQL Server as shown .

CREATE PROCEDURE [dbo].[InsertItem]

@itemNo int,

@itemname varchar(100),

@totalUnits int

AS

BEGIN

SET NOCOUNT ON;

insert into items values(@itemno, @itemname, @totalUnits );

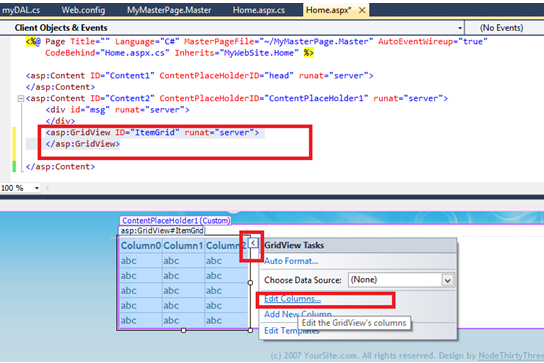
END

Build and run your project. On giving the values and clicking the “Add New Item” button a new item should get inserted. Please give a screenshot of the output in your lab report

**Deletion in GridView**

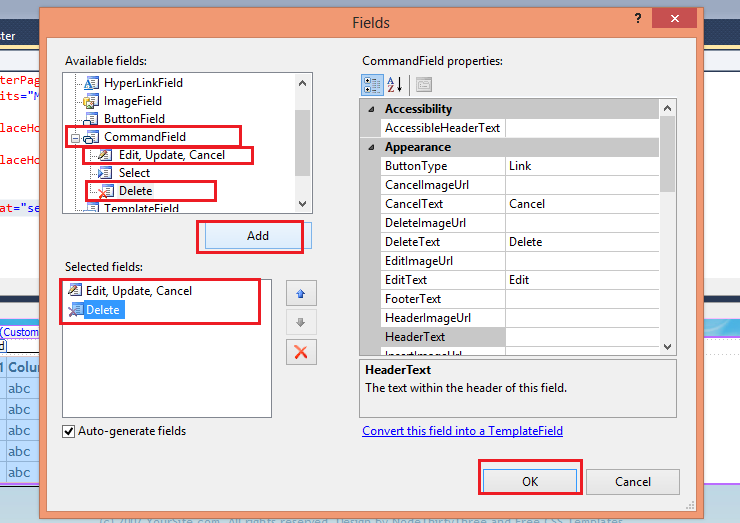
As in the last lab we used the Grid Views to display the data from an SQL table , The Grid View can also be used to delete data from that table , following exercise will demonstrate how to:

Add New Grid in your home.aspx page with ID ItemGrid , Click the Forward Arrow button and Select Edit Columns as shown in figure 2



**Figure 2**

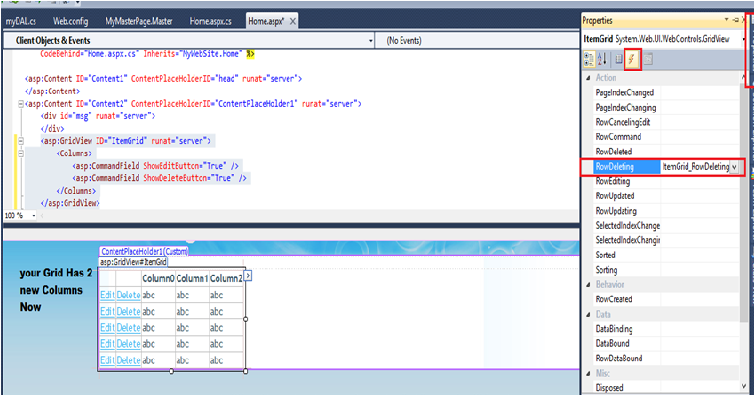
The following popup will appear, Expand *CommandField* and Add *Edit Update, Cancel* and *Delete* and click OK as shown in figure 3 below



**Figure 3**

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* as shown in figure 4



**Figure 4**

Open the [Home.aspx.cs](http://Home.aspx.cs) file and replace pageload() function with the code given below

Protected void Page\_Load(object sender, EventArgs e)

{

LoadGrid(); // fill the grid every time page loads

}

And add two functions LoadGrid() and ItemGrid\_RowDeleting and add the code given below.

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());

Label itemNo = (Label)ItemGrid.Rows[e.RowIndex].FindControl("**lblItemNo**");  
            int ItemID = Convert.ToInt32(itemNo.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());

}

}

Add the following function in your myDAL.cs file

//////////////////////////////////////////////////////

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;

}

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

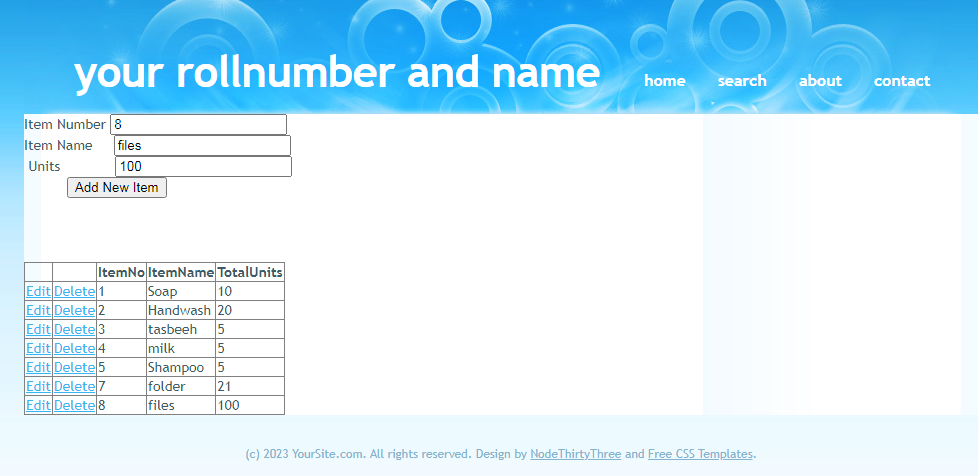
BEGIN

SET NOCOUNT ON;

DELETE FROM Items WHERE ItemNo = @ID

END

Now execute your project, following results should appear in browser as shown below in figure5



**Figure 5**

On Clicking Delete, the corresponding row will delete (check from SQL server as well). Please paste screen shots demonstrating deletion in your lab report.

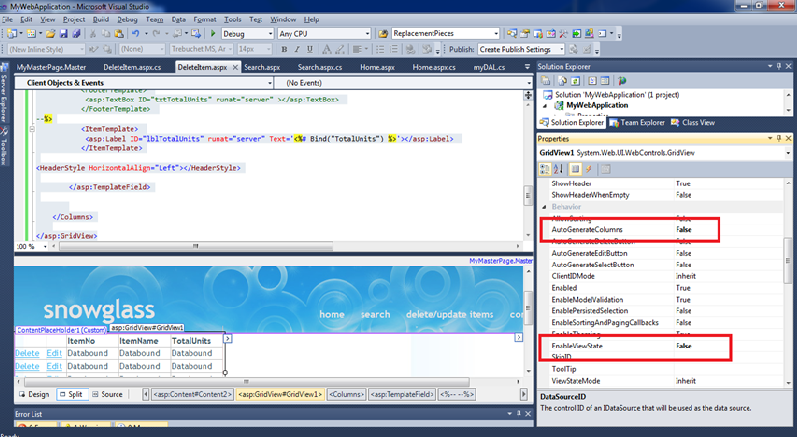
## Perform Updation via GridView Control

**Step1:**

First we will add the following code in Home.aspx page under the “<Columns>” tag of GridView we created earlier as shown below

|  |
| --- |
| <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> |

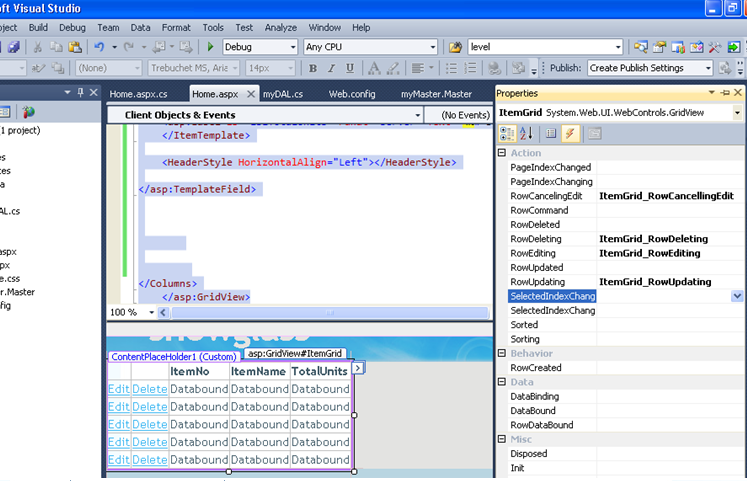
Now set the “**EnableViewState**” and “**AutoGenerateColumns**” properties of the Grid View to false as follows in the figure 6



**Figure 6**

**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 figure 7



**Figure 7**

**Step3:**

Now we are set to move to the BLL (Business Logic layer) of this page which is the [Home.aspx.cs](http://Home.aspx.cs). There we will implement all the above create event handling functions.

Replace the **Page\_Load()** function with the following function shown below :

|  |
| --- |
| 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 a shown below

|  |
| --- |
| Public void LoadGrid()  {myDAL md = new myDAL();  ItemGrid.DataSource = md.SelectItem();  ItemGrid.DataBind();  }  protected void 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(); }  protected void ItemGrid\_RowCancellingEdit(object sender, GridViewCancelEditEventArgs e)  {  ItemGrid.EditIndex = -1;  LoadGrid();  }  Protected void 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 as shown below.

|  |
| --- |
| 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 |

Please paste screen shots demonstrating updation in your lab report.