School of Information Technology

Practical 08: Database Part 2 (Insert, Update, Delete)

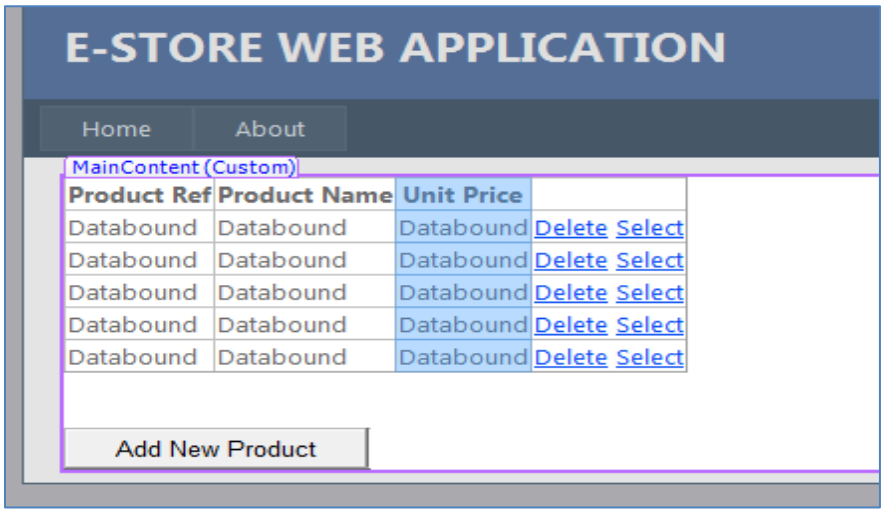
OBJECTIVES:

By the end of this Practical students should be able to:

- Edit the data in the database
- Add Update and Delete link button command in Grid View

Add “Delete and Update” command fields to the ProductView.aspx Gridview.

- Insert
- Delete – RowDeleting event
- Update – RowUpdating event



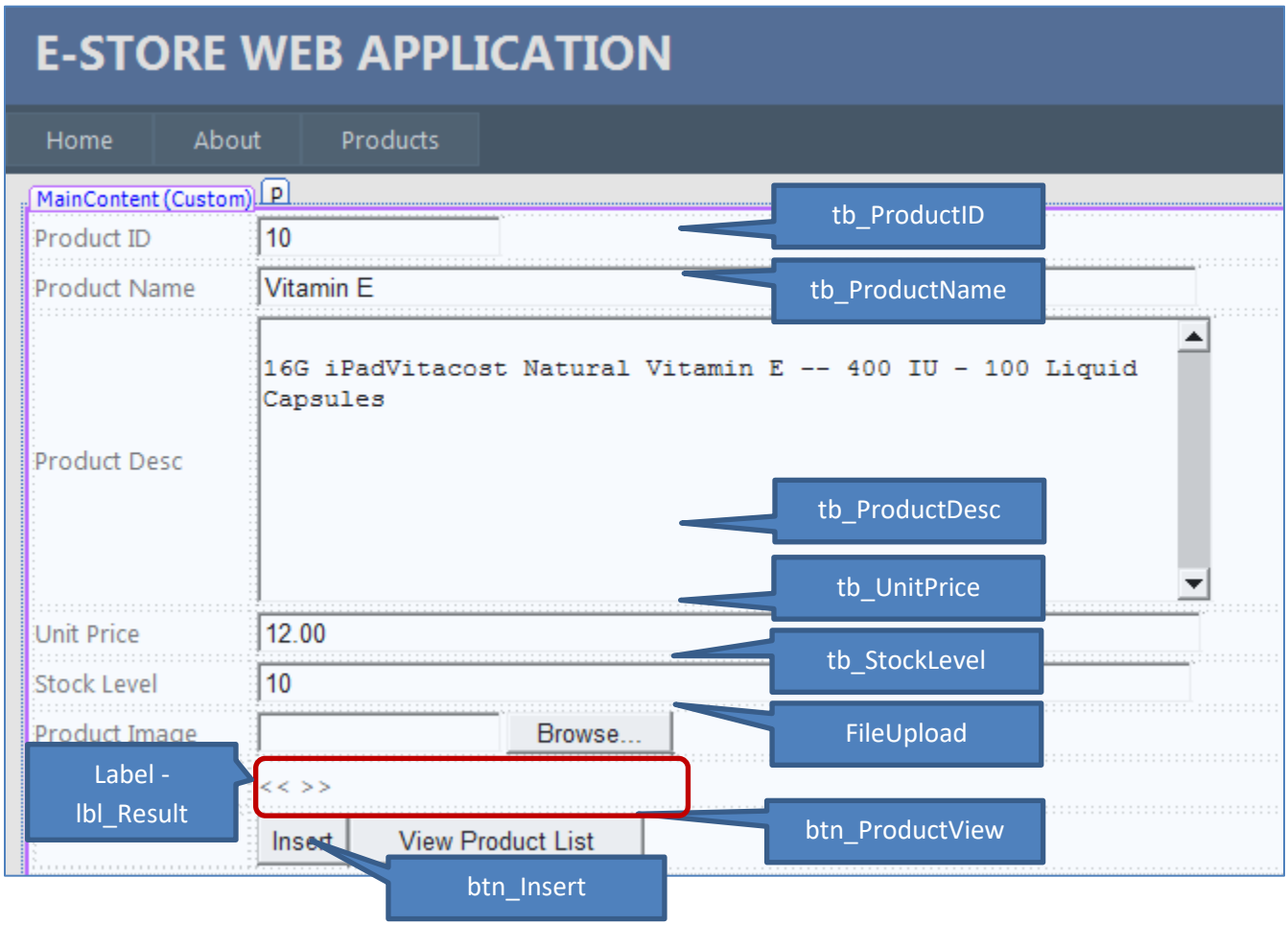
When user clicks the ‘Select’ button SelectedIndexChanged event will be fired and when user click ‘Delete’ button, RowDeleting event will be fired –both events are handled by different handlers.

Exercise 1 – Insert Data

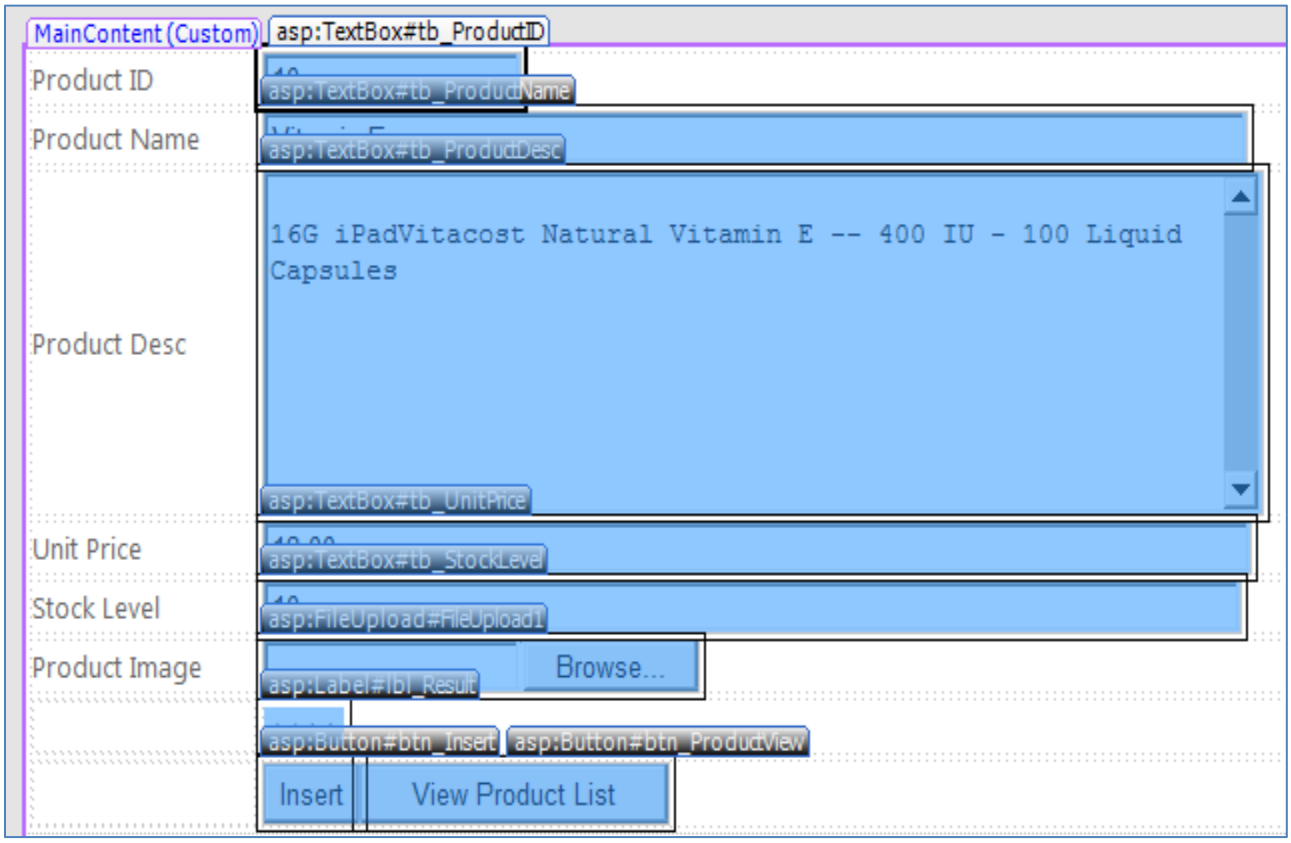
1. Open Practical 07 Solution.

   Start the Visual Studio.NET 2015. Select File → Open→ Web Site → locate the root directory of your website.

2. Create a ProductInsert.aspx web form to allow users to insert products into the Product table. See sample screen below.



- Create the UI shown below
- A table with 2 column : toolbox → HTML → Table.
- Tb_ProductID, tb_ProdName, tb_UnitPrice, tb_ProductDesc, tb_StockLevel, FileUpload1, lbl_Result



3. Add codes to ProductInsert.aspx.cs and Product.cs

ProductInsert.aspx.cs (Insert Button)

```
protected void btn_Insert_Click(object sender, EventArgs e)
{
   int result = 0;
   string image = "";

   if (FileUpload1.HasFile == true)
   {
      image = "Images\\" + FileUpload1.FileName;
   }

   Product prod = new Product(tb_ProductID.Text, tb_ProductName.Text, tb_ProductDesc.Text,
      decimal.Parse(tb_UnitPrice.Text), FileUpload1.FileName, int.Parse(tb_StockLevel.Text));
      result = prod.ProductInsert();

   if (result > 0)
   {
      string saveimg = Server.MapPath(" ") + "\\" + image;
      lbl_Result.Text = saveimg.ToString();
      FileUpload1.SaveAs(saveimg);
      //loadProductInfo();
      //loadProduct();
      //clear1();
      Response.Write("<script>alert('Insert successful');</script>");
   }
   else { Response.Write("<script>alert('Insert NOT successful');</script>"); }
}
```

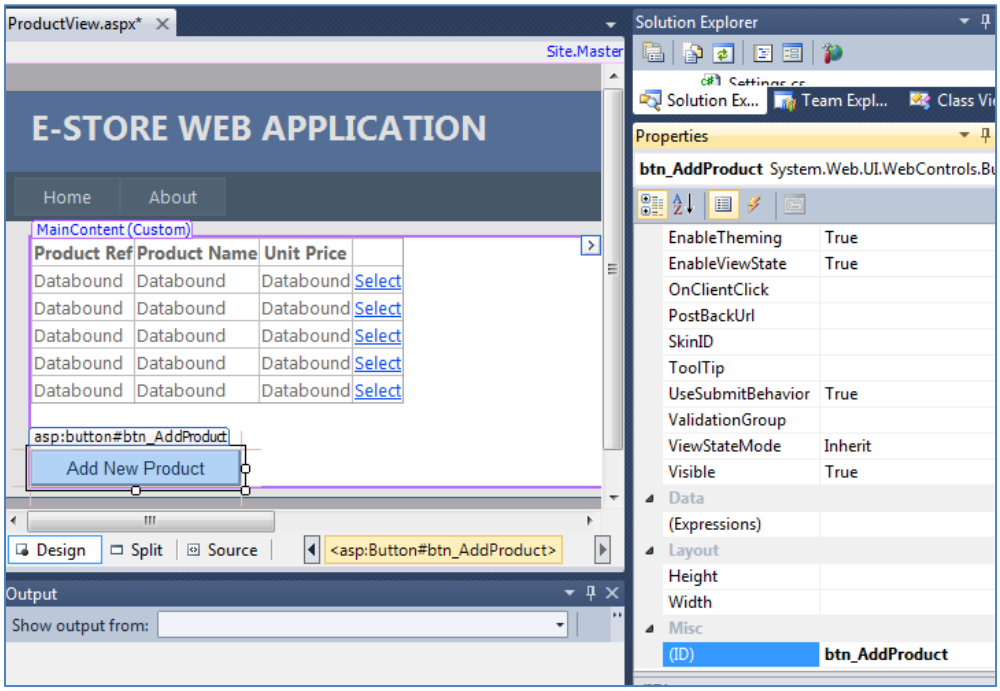
ProductInsert.aspx.cs (View Product List Button)

Re-direct page to “ProductView.aspx”

Product.cs (ProductInsert method)

```
public int ProductInsert()
{

  string msg = null;
  int result = 0;

  string queryStr = "INSERT INTO Products(Product_ID,Product_Name, Product_Desc, Unit_Price,
     Product_Image,Stock_Level)"
     + "values (@Product_ID,@Product_Name, @Product_Desc, @Unit_Price,
     @Product_Image,@Stock_Level)";
        //+ "values (@Product_ID, @Product_Name, @Product_Desc, @Unit_Price,
        @Product_Image,@Stock_Level)";

  SqlConnection conn = new SqlConnection(_connStr);
  SqlCommand cmd = new SqlCommand(queryStr, conn);
  cmd.Parameters.AddWithValue("@Product_ID", this.Product_ID);
  cmd.Parameters.AddWithValue("@Product_Name", this.Product_Name);
  cmd.Parameters.AddWithValue("@Product_Desc", this.Product_Desc);
  cmd.Parameters.AddWithValue("@Unit_Price", this.Unit_Price);
  cmd.Parameters.AddWithValue("@Product_Image", this.Product_Image);
  cmd.Parameters.AddWithValue("@Stock_Level", this.Stock_Level);

  conn.Open();
  result += cmd.ExecuteNonQuery(); // Returns no. of rows affected. Must be > 0
  conn.Close();

  return result;
 }//end Insert
```

4. Add button “Add New Product” to ProductView.aspx. Set ID to “btn_AddProduct”.



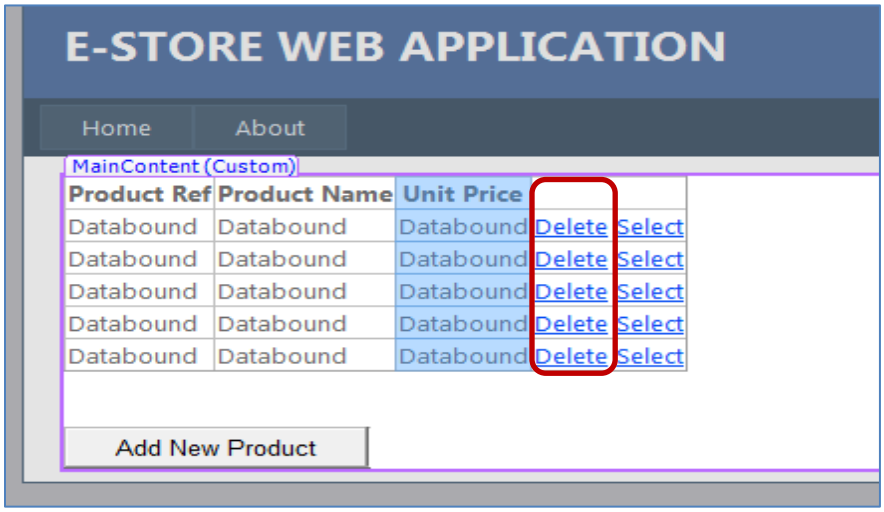
5. Add code-behind to “Add New Product”

```
protected void btn_AddProduct_Click(object sender, EventArgs e){
        Response.Redirect("ProductInsert.aspx");
}
```

6. Test out your application. Set ProductView.aspx as startup page. Try to insert some new data into the database. Since Product_ID is a primary key in that database, do ensure that the Product_ID remains unique before hitting on the Insert button.

Exercise 2 – Delete Data

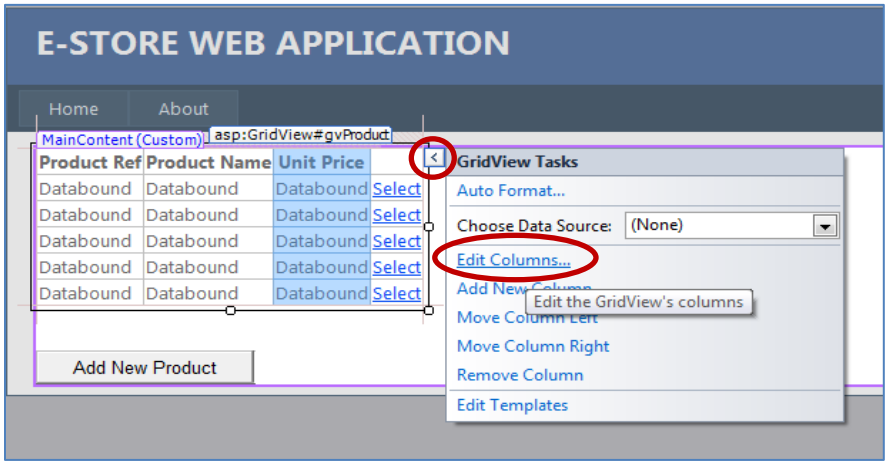
7. In this section, student will add an additional "Delete" link button.

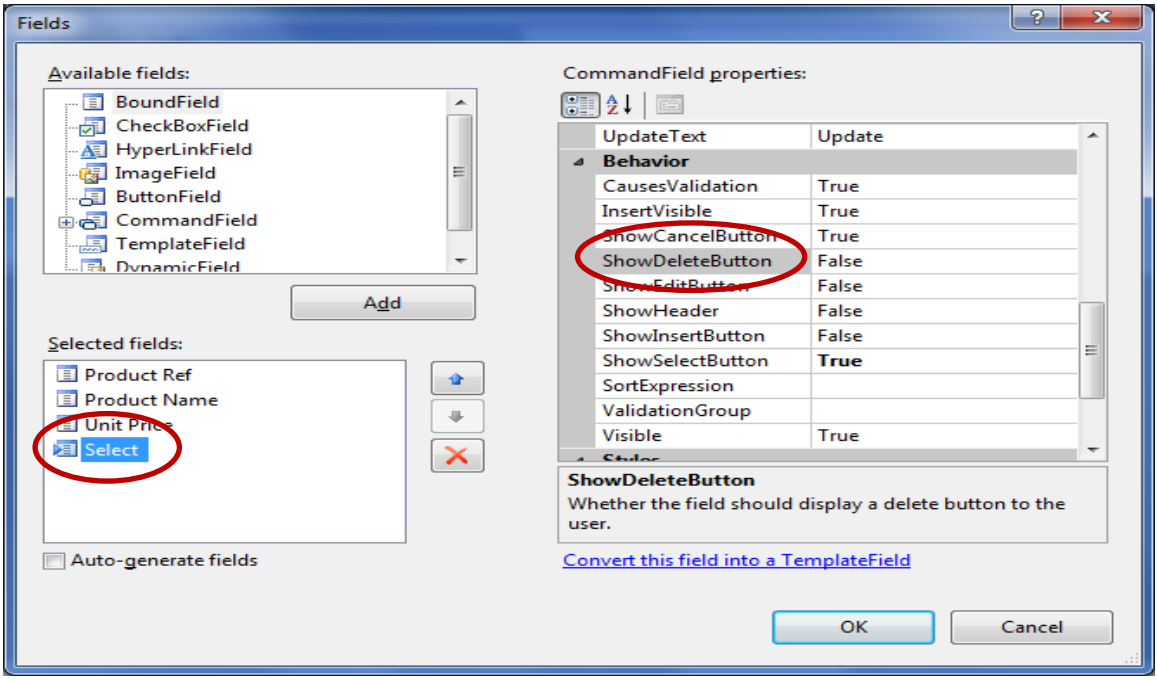


8. Modify Product.cs to include a ProductDelete method.

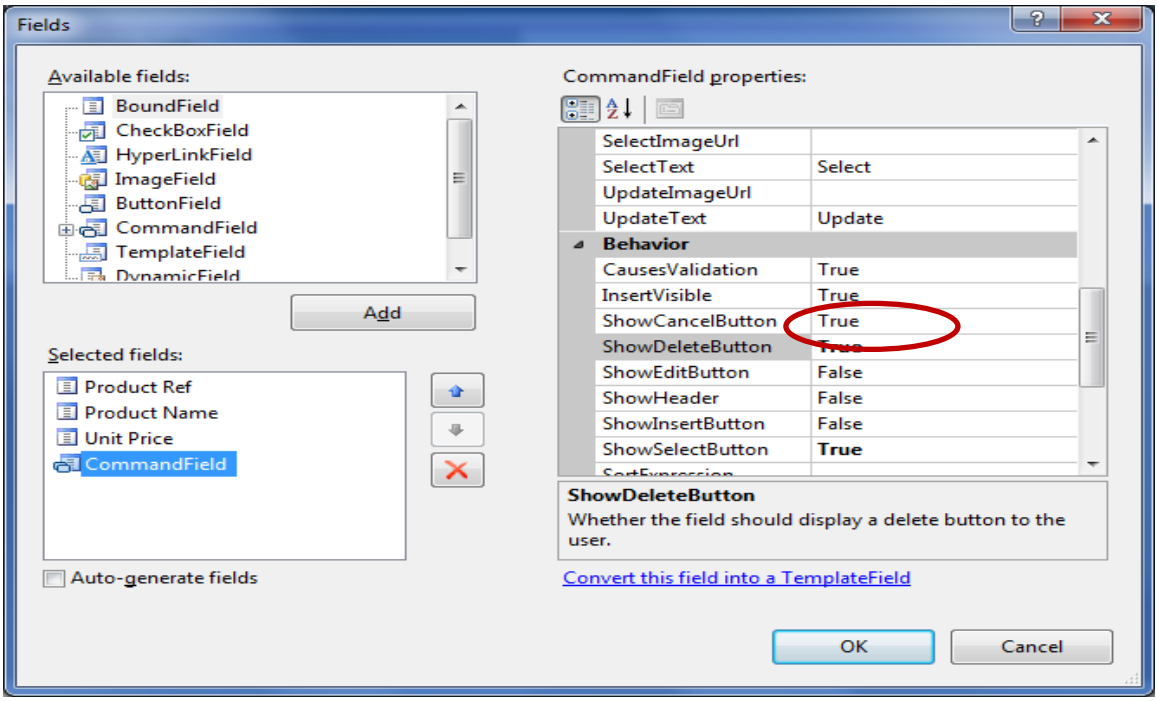
```
public int ProductDelete(string ID)
{
   string queryStr = "DELETE FROM Product WHERE Product_ID=@ID";
   SqlConnection conn = new SqlConnection(_connStr);
   SqlCommand cmd = new SqlCommand(queryStr, conn);
   cmd.Parameters.AddWithValue("@ID", ID);
   conn.Open();
   int nofRow = 0;
   nofRow = cmd.ExecuteNonQuery();
   conn.Close();
   return nofRow;

 }//end Delete
```

9. In ProductView.aspx, add a new command button “Delete” to the gridview. Choose “Edit Columns”.

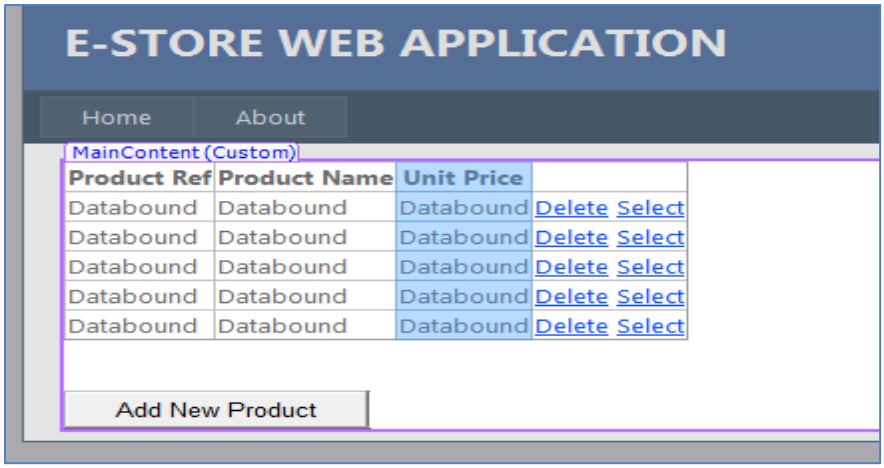




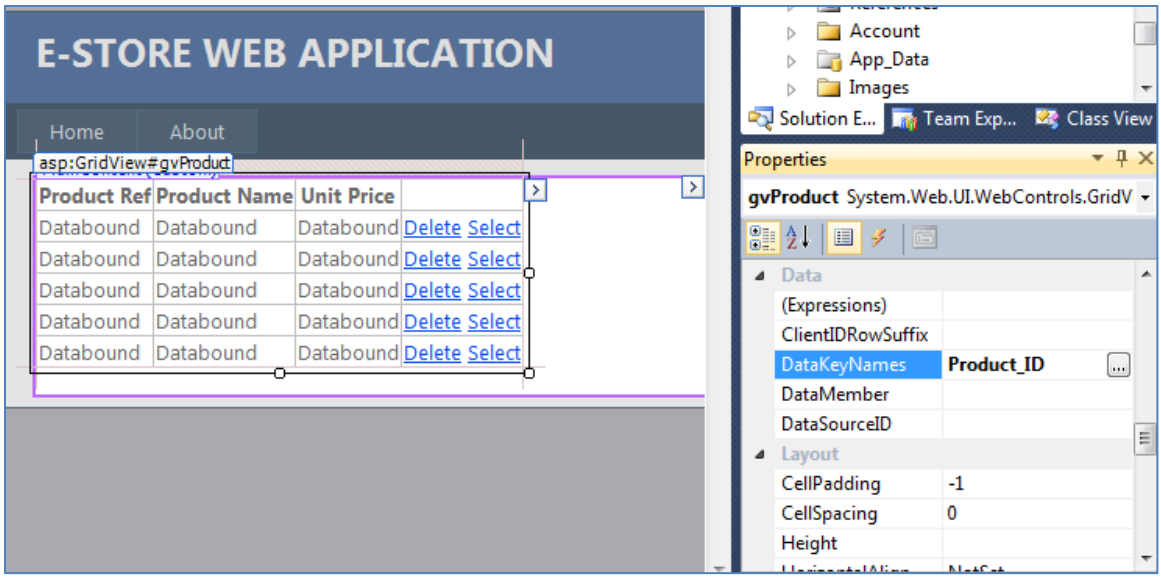
10. Set **ShowDeleteButton = True**. After setting ShowDeleteButton to True, the "Select" field will automatically change to "CommandField".



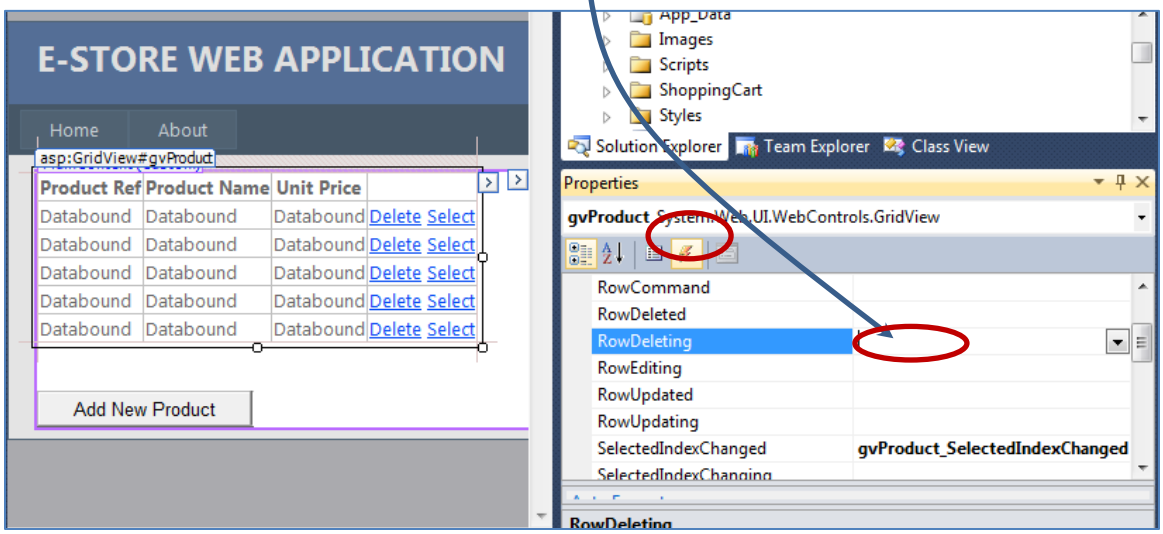
11. The 'Delete' link button will appear beside the "Select" link button.



12. Set the DataKeyNames of the GridView to **Product_ID** – according to the property of the Product Entity class.



13. Add codes to the Delete command event. Double-click on the space provided.



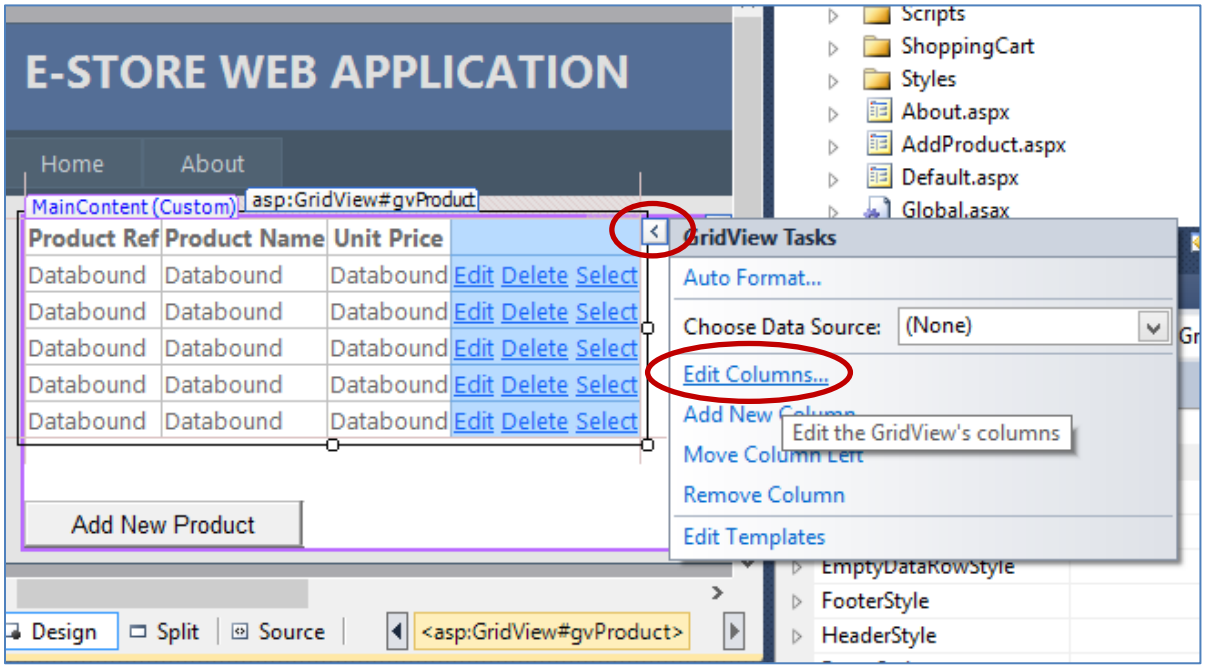

```
protected void gvProduct_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
   int result = 0;
   Product prod = new Product();
   string categoryID = gvProduct.DataKeys[e.RowIndex].Value.ToString();
   result = prod.ProductDelete(categoryID);

   if (result > 0)
   {
      Response.Write("<script>alert('Product Remove successfully');</script>");
   }
   else
   {
       Response.Write("<script>alert('Product Removal NOT successfully');</script>");
   }

   Response.Redirect("ProductView.aspx");

}
```

14. Test run your web application by first adding a new product. Proceed to delete the newly created product from ProductView.aspx.

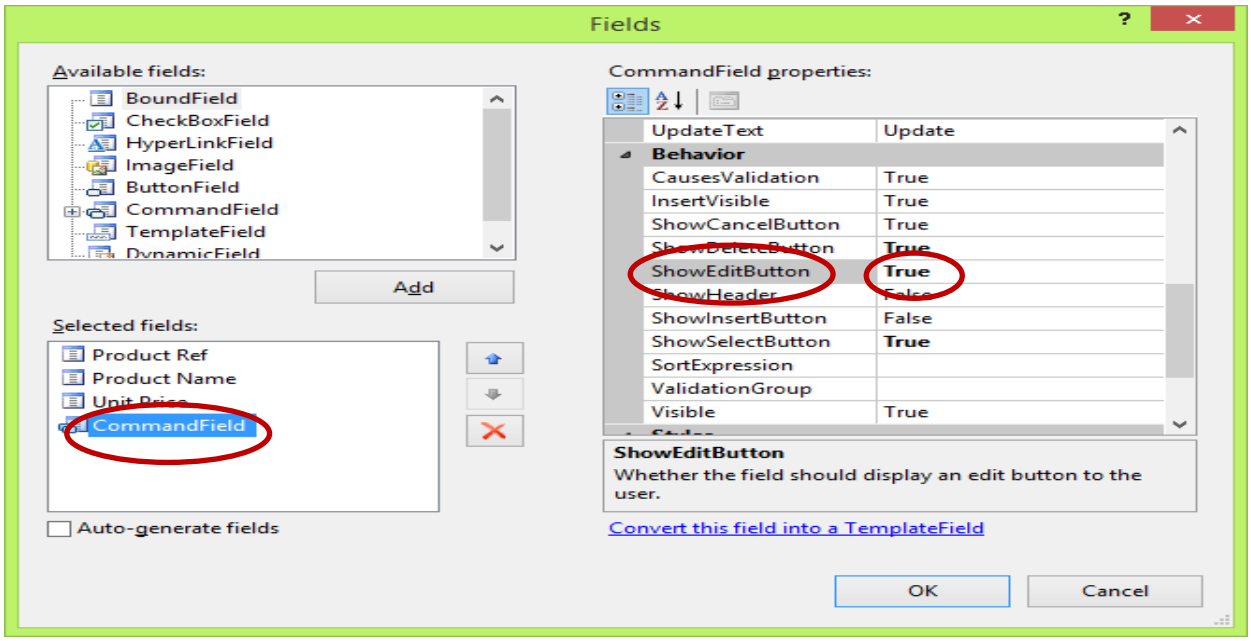
Exercise 3 – Update Data

15. In this section, student will add an additional "Edit" link button. The "Edit" feature will allow users to edit a row of data - *RowEditing*

16. Activate the "Edit" link button inside the CommandField. Select the GridView and Choose "Edit Columns".

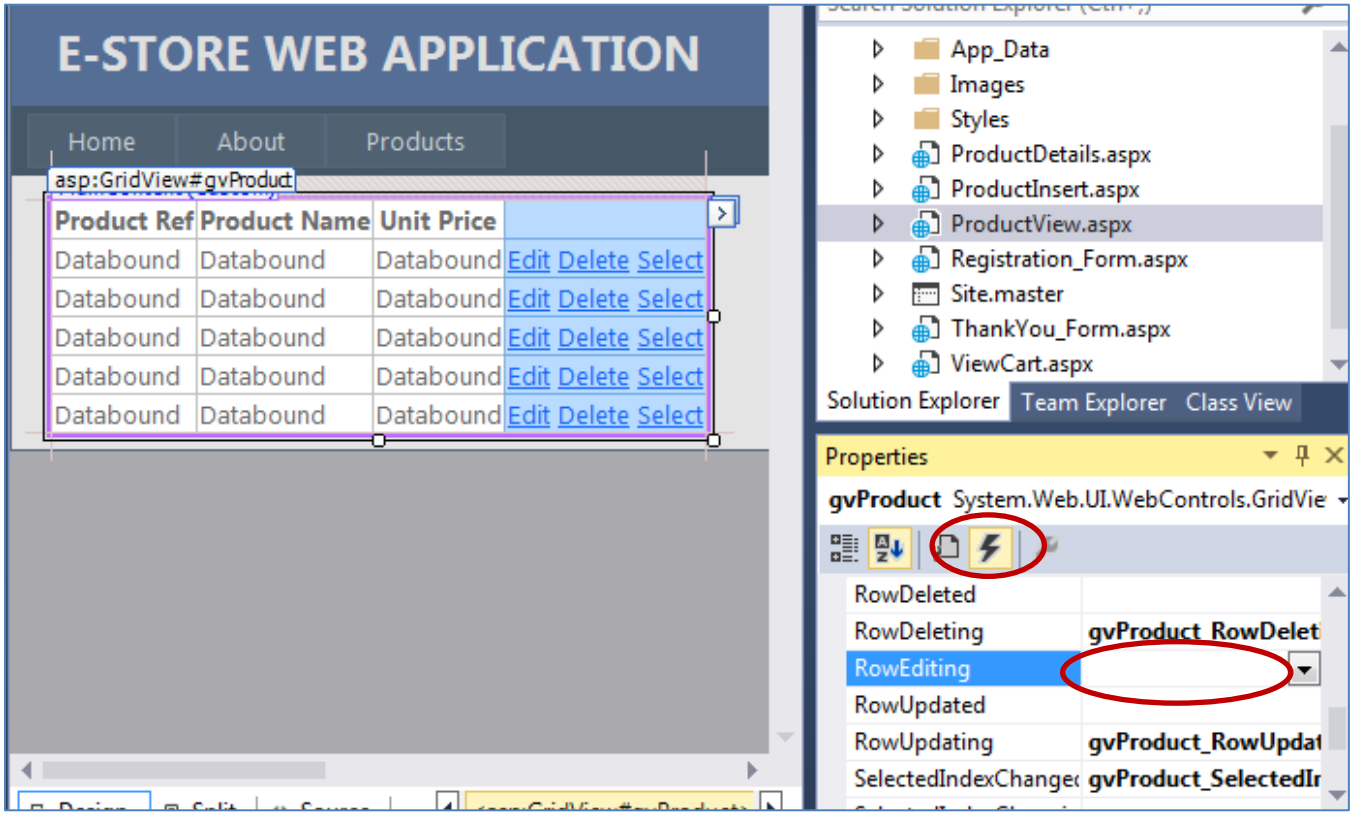


17. Set the value of "ShowEditButton" to True.



18. Implement the event handler for the “Edit” link button.

19. Double-click on empty value for RowEditing.

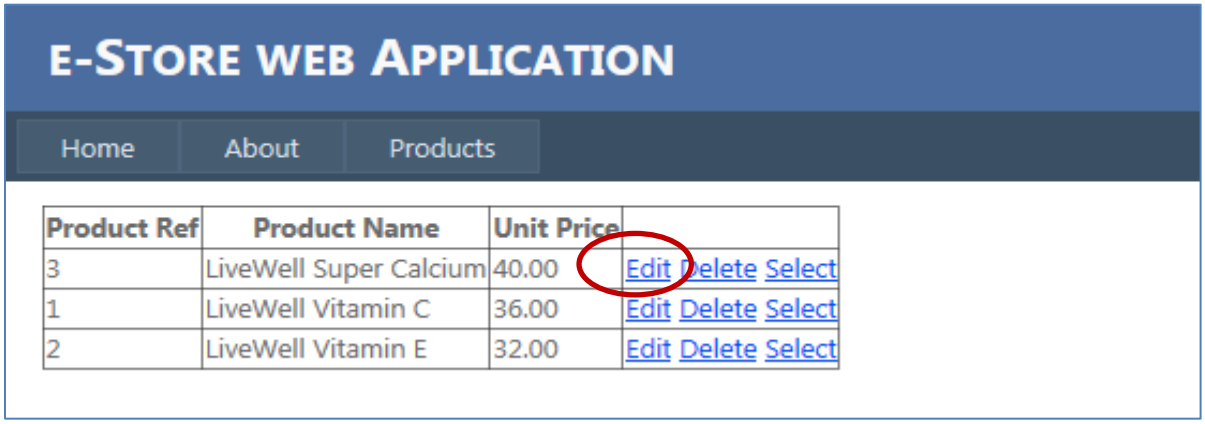


20. Include the codes for the RowEditing Event Handler. These codes will allow the selected gridvew rows to be editable.

```
protected void gvProduct_RowEditing(object sender, GridViewEditEventArgs e)
{
    gvProduct.EditIndex = e.NewEditIndex;
    bind();
}
```

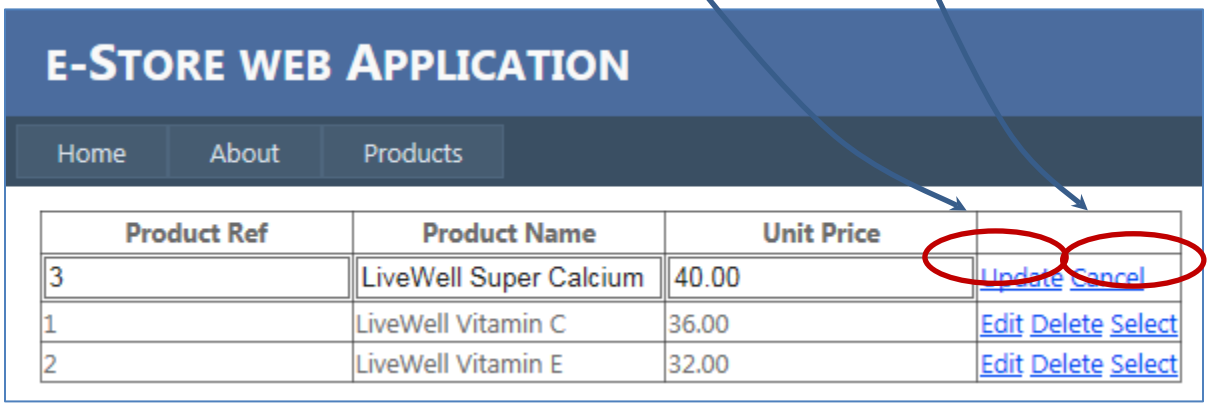
21. Test run ProductView.aspx

22. Try selecting the “Edit” link button in any rows.

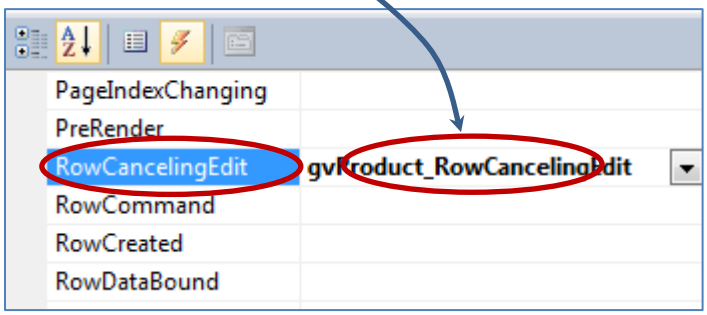


23. The selected rows will change to editing mode with 2 additional feature : Update and Cancel.

24. We need to handle both events as well – **RowUpdating** and **RowCancelingEdit**

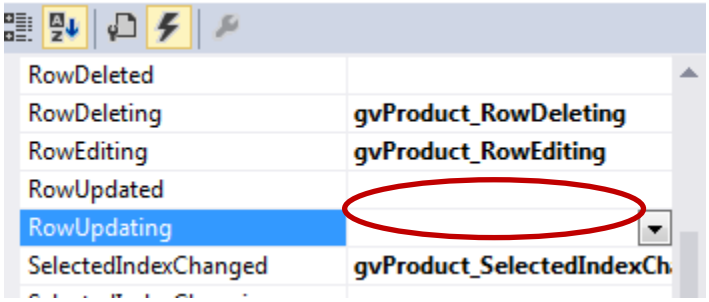


25. Implement the RowCancelingEdit event handler. Select GridView and search for the event handler name. Double-click on the empty value.




```
protected void gvProduct_RowCancelingEdit(object sender, GridViewCancelEditEventArgs e)
{
  gvProduct.EditIndex = -1;
  bind();
}
```

Implement the RowUpdating event handler.



26. Include codes to implement the RowUpdating EventHandler.

    Note : Product ID which is a primary key should be allowed to be updated.

```
protected void gvProduct_RowUpdating(object sender, GridViewUpdateEventArgs e)
{
    int result = 0;
    Product prod = new Product();
    GridViewRow row = (GridViewRow)gvProduct.Rows[e.RowIndex];
    string id = gvProduct.DataKeys[e.RowIndex].Value.ToString();
    string tid = ((TextBox)row.Cells[0].Controls[0]).Text;
    string tname = ((TextBox)row.Cells[1].Controls[0]).Text;
    string tprice = ((TextBox)row.Cells[2].Controls[0]).Text;

    result = prod.ProductUpdate(tid, tname, decimal.Parse(tprice));
    if (result > 0)
    {
        Response.Write("<script>alert('Product updated successfully');</script>");
    }
    else
    {
        Response.Write("<script>alert('Product NOT updated');</script>");
    }
    gvProduct.EditIndex = -1;
    bind();
}
```

27. Modify Product.cs to include a ProductUpdate () method

Product.cs (ProductUpdate() method)

```
public int ProductUpdate(string pId, string pName, decimal pUnitPrice)
{
    string queryStr = "UPDATE Products SET" +
          //" Product_ID = @productID, " +
       " Product_Name = @productName, " +
       " Unit_Price = @unitPrice " +
       " WHERE Product_ID = @productID";

    SqlConnection conn = new SqlConnection(_connStr);
    SqlCommand cmd = new SqlCommand(queryStr, conn);
    cmd.Parameters.AddWithValue("@productID", pId);
    cmd.Parameters.AddWithValue("@productName", pName);
    cmd.Parameters.AddWithValue("@unitPrice", pUnitPrice);

    conn.Open();
    int nofRow = 0;
    nofRow = cmd.ExecuteNonQuery();

    conn.Close();

    return nofRow;
     }//end Update
```

28. Test run your web app again.

--- The End --