In

School of Information Technology

Practical 06: Shopping Cart

OBJECTIVES:

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

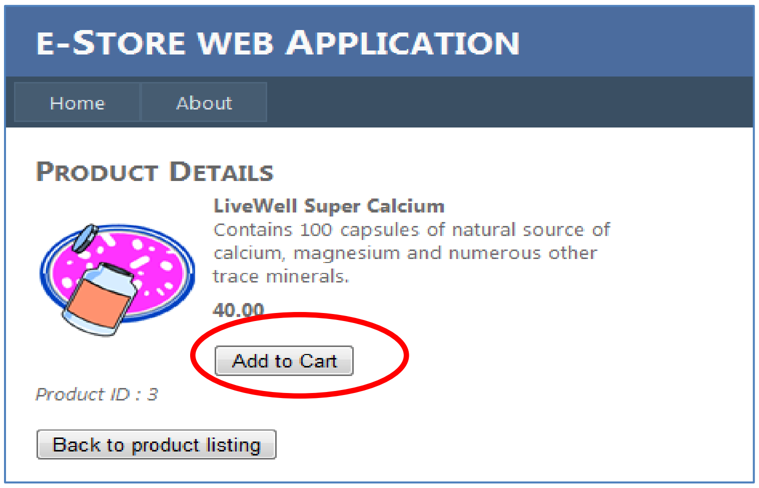
- Integrate Shopping Cart feature into a Web Application

Introduction

In this practical, you will learn to integration Shopping Cart feature into a Web Application that you have completed last week. The Shopping Cart consist of an AddCart, RemoveCart and UpdateCart feature. In addition, the Shopping Cart will compute the final total payment for the customer before Checking out.

Please download a template form Blackboard to start this practical.

The FINAL GUI for this practical are shown below :



PRODUCT DETAILS

LiveWell Super Calcium

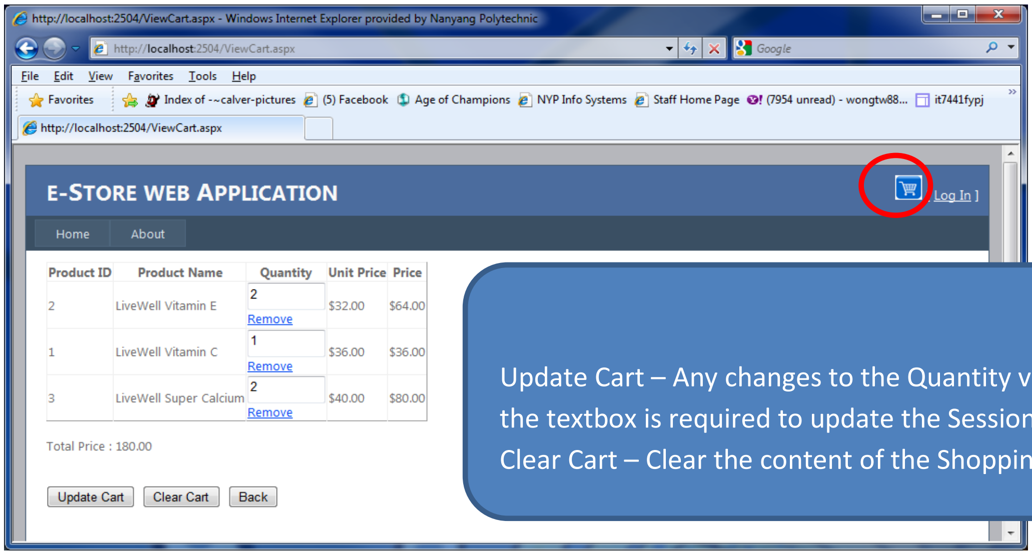
Contains 100 capsules of natural source of calcium, magnesium and numerous other trace minerals.

40.00

Product ID : 3

[ProductsDetails.aspx]

When user clicks on the “Add to Cart” button, the products details will be added to the session.


http://localhost:2504/ViewCart.aspx - Windows Internet Explorer provided by Nanyang Polytechnic
File Edit View Favorites Tools Help
E-STORE WEB APPLICATION
[Log In]
Home
About
Update Cart
Clear Cart
Back

Product ID	Product Name	Quantity	Unit Price	Price
2	LiveWell Vitamin E	2 Remove	$32.00	$64.00
1	LiveWell Vitamin C	1 Remove	$36.00	$36.00
3	LiveWell Super Calcium	2 Remove	$40.00	$80.00

Total Price : 180.00

Update Cart – Any changes to the Quantity value inside the textbox is required to update the Session.
Clear Cart – Clear the content of the Shopping Cart.

Major Steps involved in this practical

STEP 1: LOAD TEMPLATE AND SHOPPING CART CODES

- A template is provided for you to start the lab
- Product.cs class is provided
- ShoppingCart.cs class is provided
- ShoppingCartItem.cs class is provided
- Practical 06 Resources.zip

STEP 2: CREATE ProductDetails.aspx WITH DUMMY DATA

- With the absence of database, we will need to create dummy data in ProductDetails.aspx
- Create a Web Form.
- Add HTML tables to create the UI.
- Add Menu Item (Products) to link to ProductDetails.aspx

STEP 3: ACTIVATE “ADD TO CART” FEATURE

- Develop codes to add dummy data into the ShoppingCart Instance.
- Product can be added multiple times and quantity will be increased.

STEP 4: DISPLAYING YOUR SHOPPING CART IN ViewCart.aspx

- Create a Web Form to display the Shopping Cart and its items.

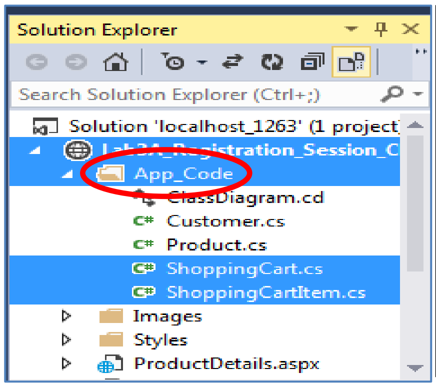
STEP 5: EXTENDING YOUR SHOPPING CART FEATURES

- Add more features such as Clear Shopping Cart and Update Shopping cart item quantity.

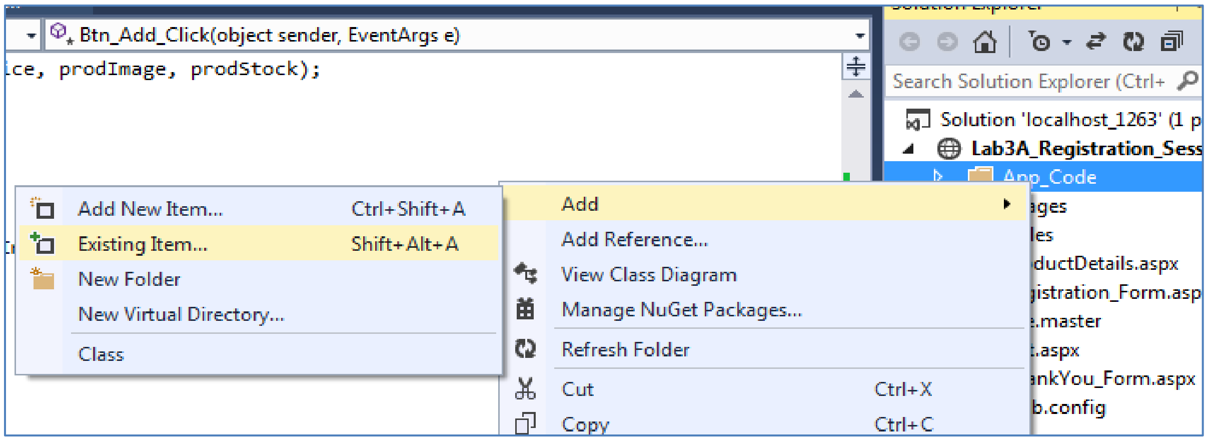
Let's begin!

STEP 1: LOAD TEMPLATE AND SHOPPING CART CODES

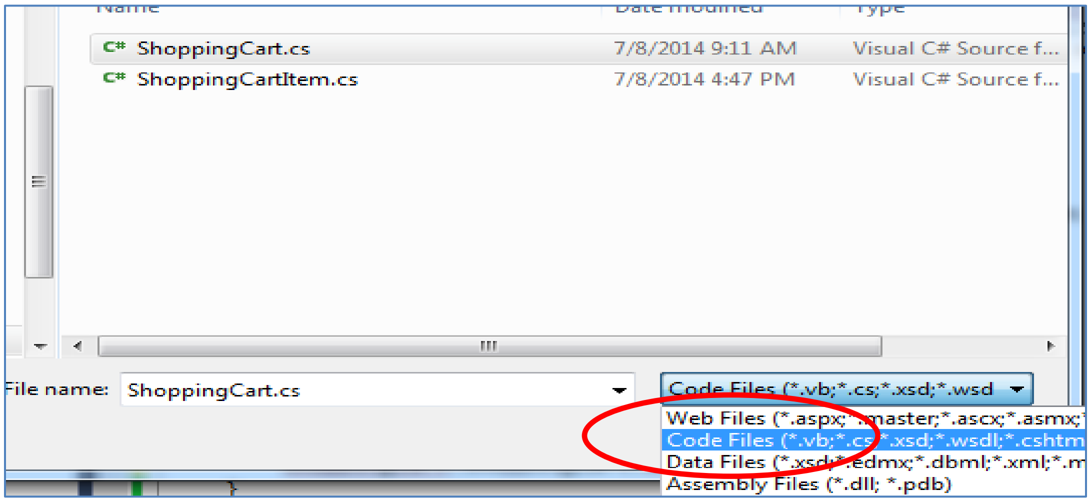
1. Download a Practical 06 Resources from blackboard to start the practical. Uncompressed the zip file. You will see **Shopping Cart App** and **Resources** Folder
2. Start the Visual Studio.NET 2015. Open the Web application template, Shopping Cart App.
3. Add 2 ShoppingCart Classes – ShoppingCart.cs , ShoppingCartItem.cs from existing files. Download **Resources** from Blackboard.



- Right- Click "App_Code"→ Add "Existing Item..."
- Browse and load the 2 files : **ShoppingCart.cs** and **ShoppingCartItem.cs**
- Please read through the codes.



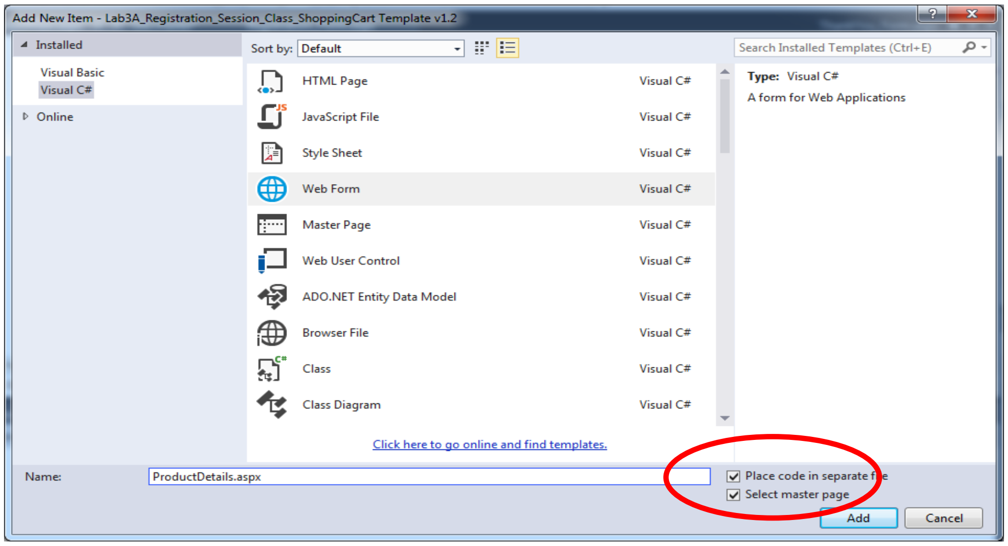
```
ice, prodImage, prodStock);
```


Name
Date modified
Type
ShoppingCart.cs 7/8/2014 9:11 AM Visual C# Source f...
ShoppingCartItem.cs 7/8/2014 4:47 PM Visual C# Source f...
File name: ShoppingCart.cs
Code Files (*.vb;*.cs;*.xsd;*.wsd
Web Files (*.aspx;*.master;*.ascx;*.asmx;
Code Files (*.vb;*.cs;*.xsd;*.wsdl;*.cshtm
Data Files (*.xsd;*.edmx;*.dbml;*.xml;*.m
Assembly Files (*.dll; *.pdb)

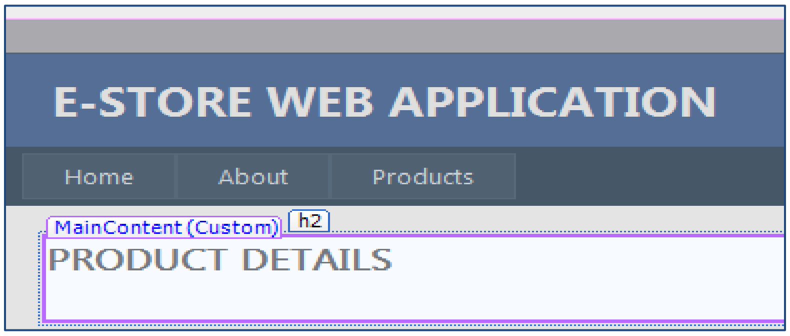
STEP 2: CREATE ProductDetails.aspx WITH DUMMY DATA

4. Create a Product Details page.

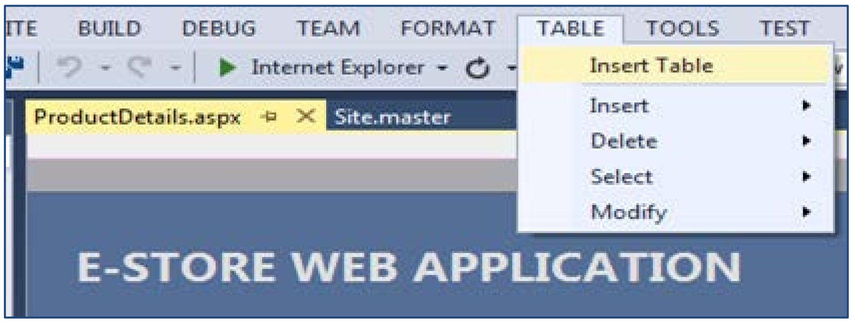
- Add a new Web Form named : ProductDetails.aspx
- Include Site.Master as the master file.


Add New Item - Lab3A_Registration_Session_Class_ShoppingCart Template v1.2
Installed
Visual Basic
Visual C#
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
HTML Page Visual C#
JavaScript File Visual C#
Style Sheet Visual C#
Web Form Visual C#
Master Page Visual C#
Web User Control Visual C#
ADO.NET Entity Data Model Visual C#
Browser File Visual C#
Class Visual C#
Class Diagram Visual C#
Type: Visual C#
A form for Web Applications
Click here to go online and find templates.
Name: ProductDetails.aspx
Place code in separate file
Select master page
Add
Cancel

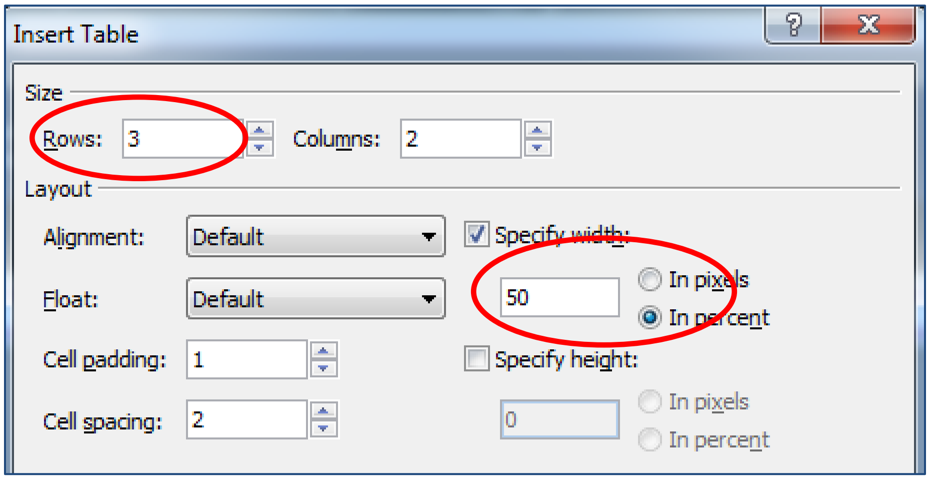
- Set your cursor onto the space just below the word “MainContent”.
- Enter the sentence “PRODUCT DETAILS” with Paragraph-heading 2.



- Add a table directly below the “Product Details”. 3 Rows and 2 Columns.

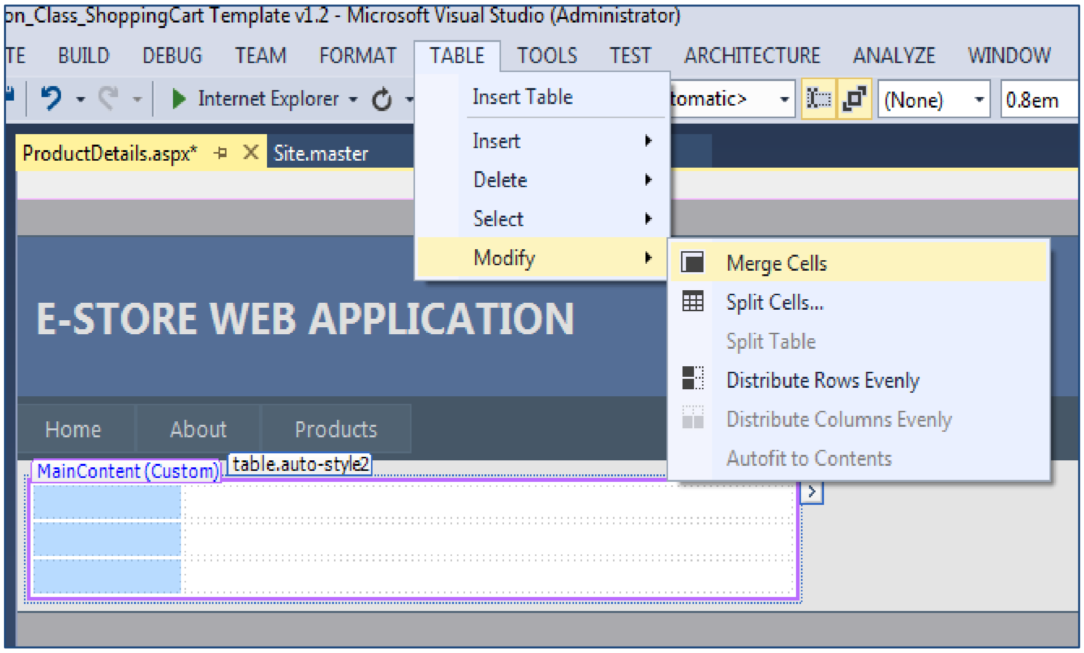


- Add 3 Rows and 2 Columns. Set Width to 50%.

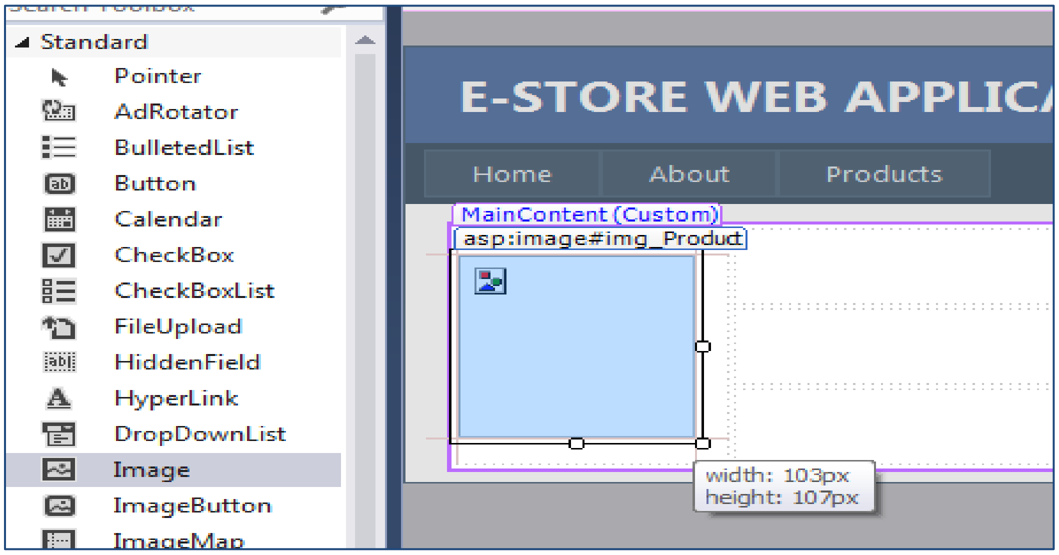


Rows: 3 Columns: 2
Cell padding: 1
Cell spacing: 2
[✓] Specify width: 50 [In pixels ✓ In percent]
[ ] Specify height: 0 In pixels In percent

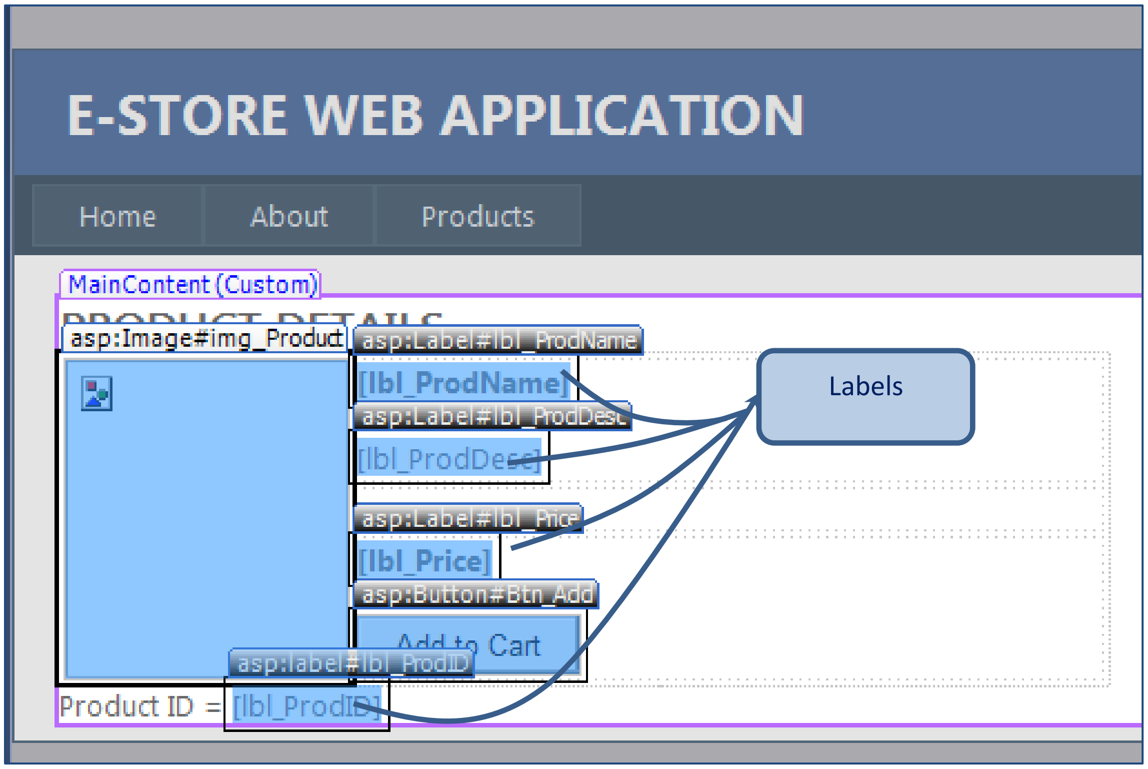
- Merge the 3 rows from the 1st Columns for displaying the Product photo.



- Drag and drop an "Image" control form toolbox. Set the Image ID to "img_Product". Pull and extend the Image box.



- Create the rest of the UI - 4 labels and 1 button.
- Labels - lbl_ProdName, lbl_ProdDesc, lbl_Price, lbl_ProdID
- Button – Btn_Add

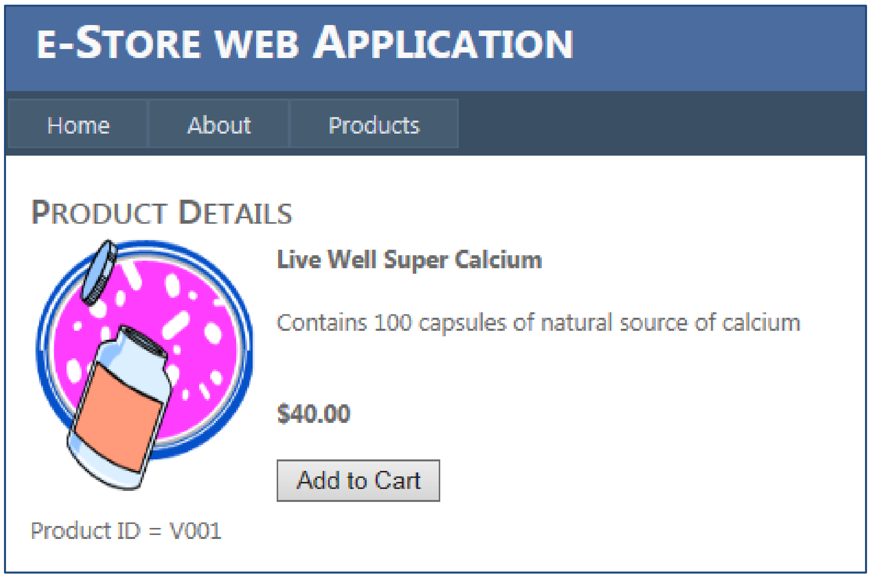


- Since we have not learn how to connect to a database yet, therefore we will need to create some dummy data to add into the Shopping Cart.
- Go to Code behind of ProductDetails.aspx.

Enter the codes into the Page_Load of ProductDetails.aspx to simulate loading of data from Database.

```
    Product prod = null;
    protected void Page_Load(object sender, EventArgs e)
    {
        //*** Set some dummy data until the next practical when we connected to a DB.***
        string prodID = "V001";
        string prodName = "Live Well Super Calcium";
        string prodDesc = "Contains 100 capsules of natural source of calcium";
        decimal prodPrice = 40.0M;
        string prodImage = "Vitamin.png";
        int prodStock = 10;
        prod = new Product(prodID,prodName, prodDesc, prodPrice, prodImage, prodStock);
        // *** End Dummy Data

        // **** Display the Dummy Data on the form ****
        lbl_ProdID.Text = prod.Product_ID.ToString();
        lbl_ProdName.Text = prod.Product_Name.ToString();
        lbl_ProdDesc.Text = prod.Product_Desc.ToString();
        img_Product.ImageUrl = "~\\Images\\" + prod.Product_Image;
        lbl_Price.Text = prod.Unit_Price.ToString("C");
        // **** End Display the Dummy Data on the form ****
     }
```

- Set ProductDetails.aspx as the “Start Page”.
- Test run your web application. It should be able to display all the dummy info.



PRODUCT DETAILS

Live Well Super Calcium

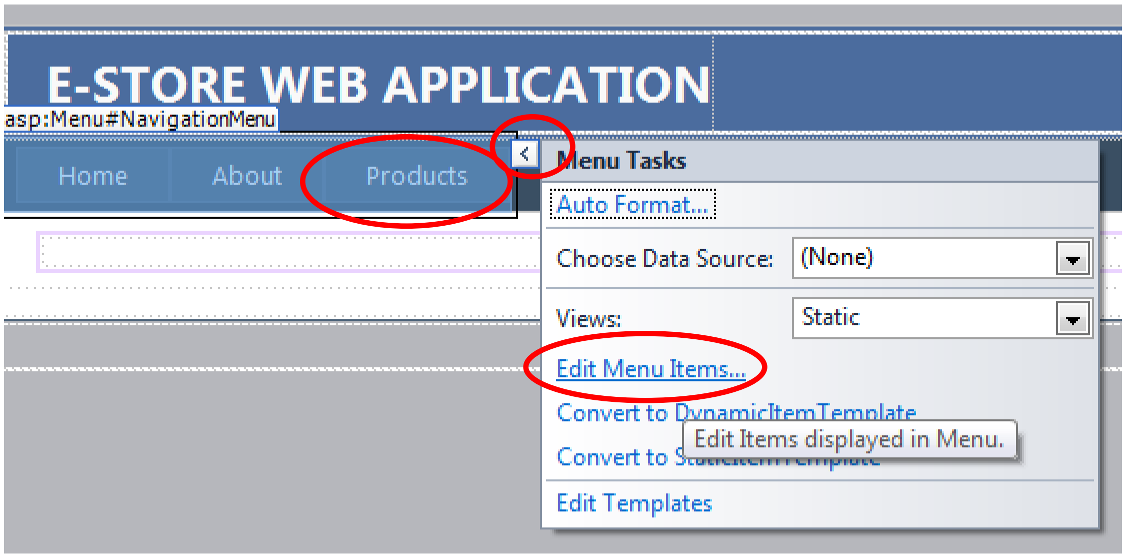
Contains 100 capsules of natural source of calcium

$40.00

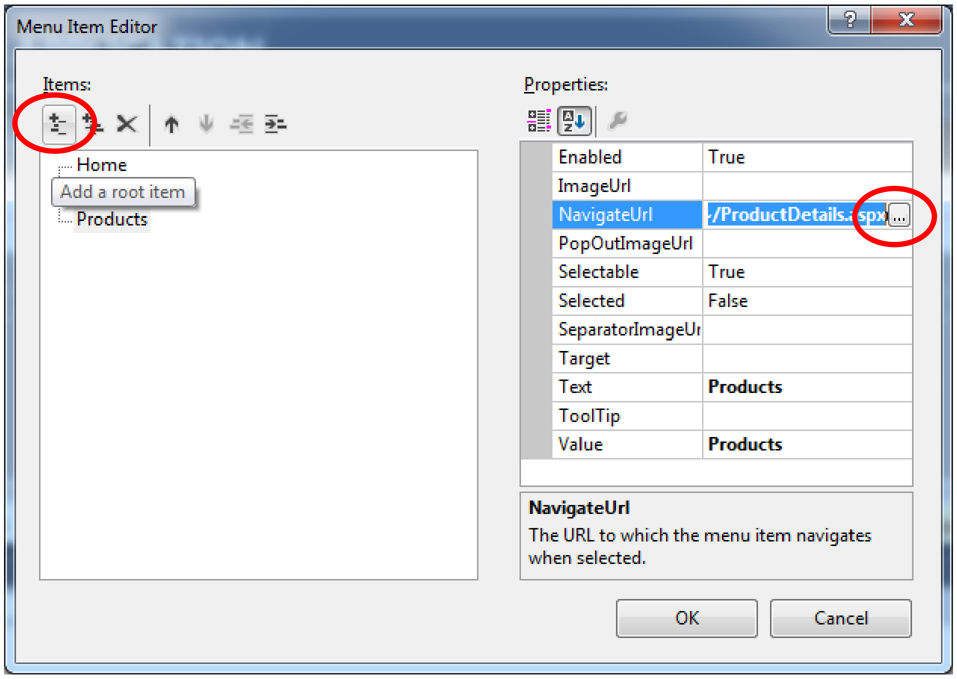
Product ID = V001

5. Add another navigation item (Products) into Menu to access ProductDetails.aspx.

   - Under Site.master design mode, select "Edit Menu Items"



   - Add another root item – **Products**. Point the NavigateUrl to **ProductDetails.aspx**.



Enabled	True
ImageUrl	
NavigateUrl	~/ProductDetails.aspx
PopOutImageUrl	
Selectable	True
Selected	False
SeparatorImageUr	
Target	
Text	**Products**
ToolTip	
Value	**Products**

NavigateUrl
The URL to which the menu item navigates when selected.

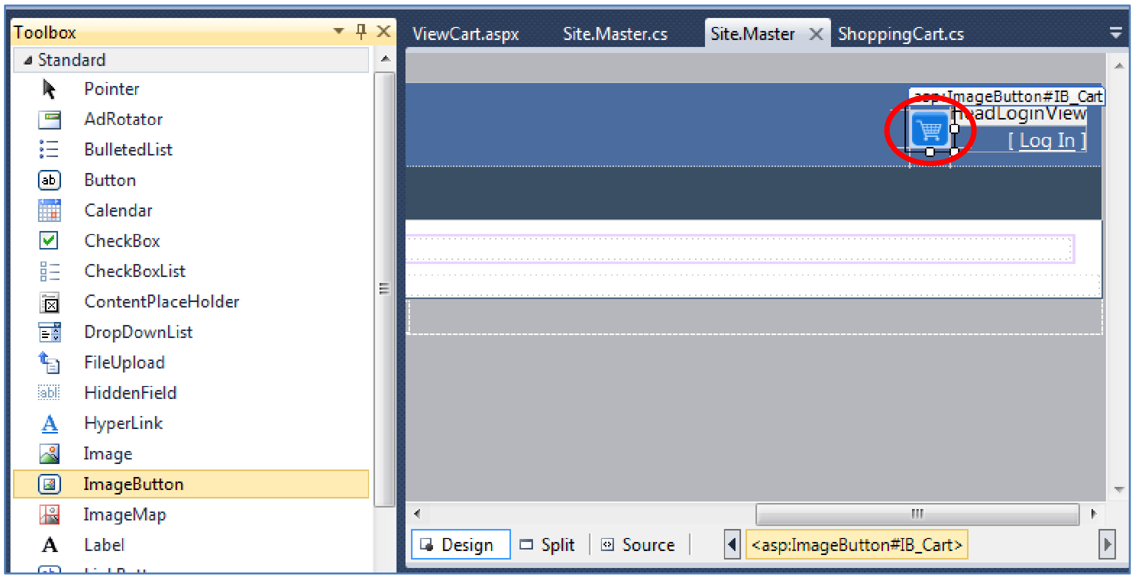
   - Test run your web application with the new Menu item.

6. Create a Shopping Cart Form.

   - Add a new Web Form named : **ViewCart.aspx**
   - Include Site.Master as the master file.
   - Leave it empty for the moment. We will revisit this page after we are done creating the Shopping cart icon.

7. Add event codes to the Shopping Cart icon in the Master Page (Site.Master). By double-clicking on this icon, user will be able to access their shopping cart immediately.

   - Go to Site.Master in Design View.



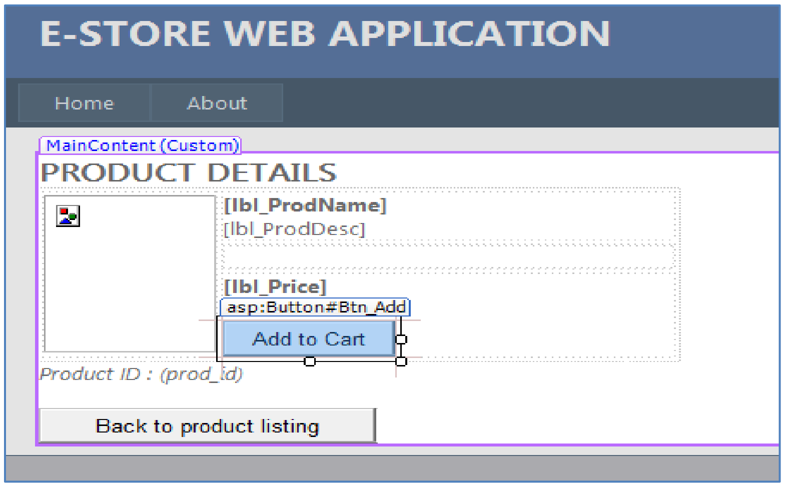
   - Double-click on the ShoppingCart icon. Add event codes :

```
        protected void IB_Cart_Click(object sender, ImageClickEventArgs e)

        {

             Response.Redirect("ViewCart.aspx");

        }
```

   - Test run your web application. ViewCart.aspx should be empty for now.

STEP 3: ACTIVATE "ADD TO CART" FEATURE

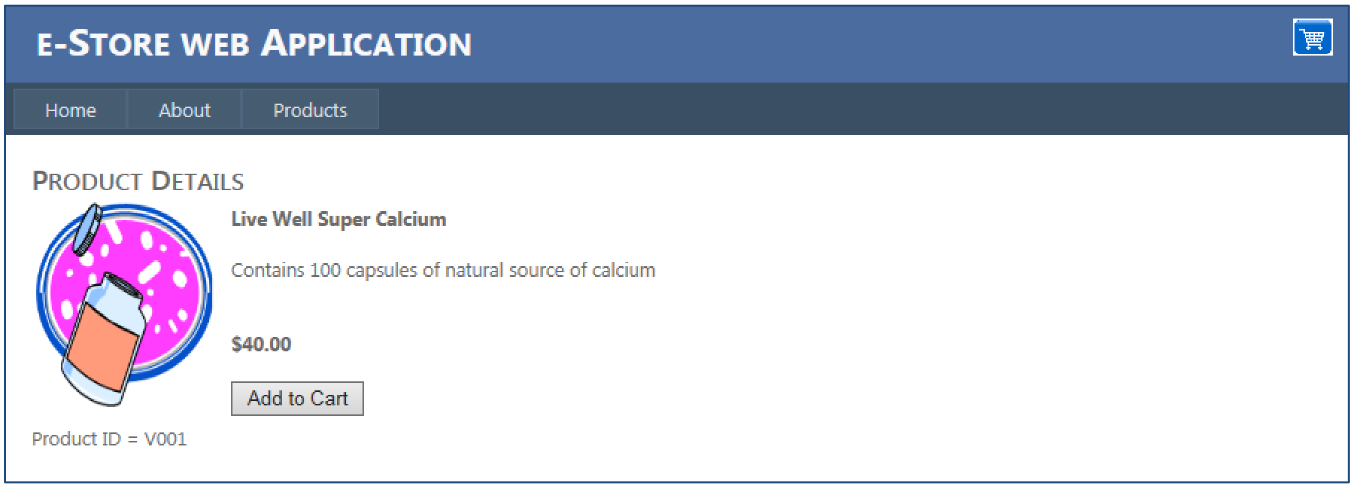
8. Activate "Add to Cart" function. See Appendix B.



9. Add codes to the "Add to Cart" function.

```
protected void btn_Add_Click(object sender, EventArgs e)
{
        string iProductID = prod.Product_ID.ToString();
        ShoppingCart.Instance.AddItem(iProductID, prod);
}
```

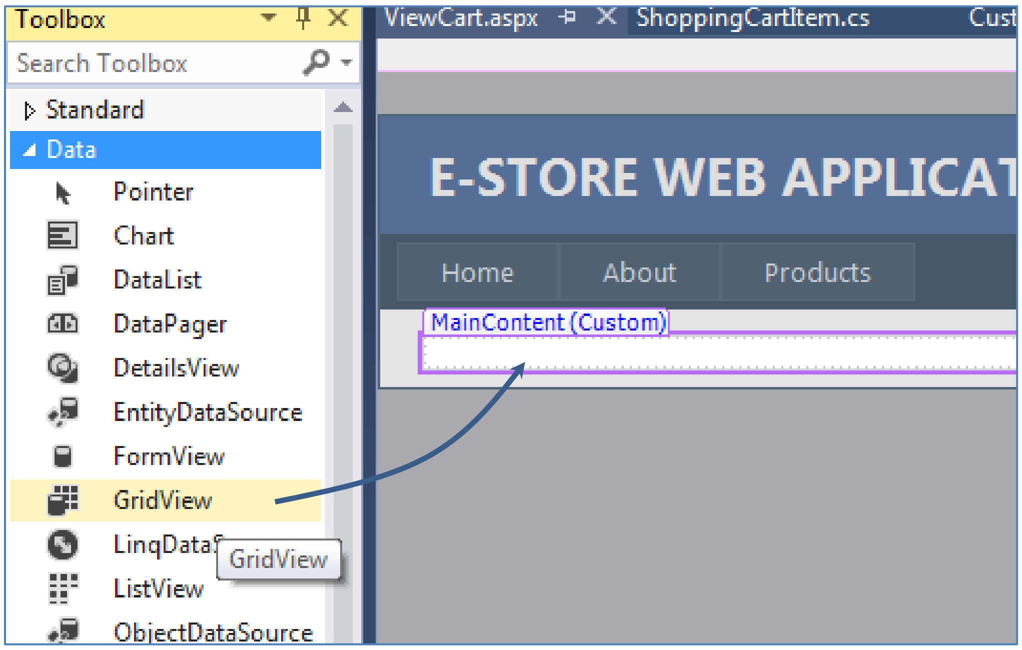
10. Test run your application! At this point, your shopping cart should be empty since you have not develop the codes to display the content in ShoppingCart.

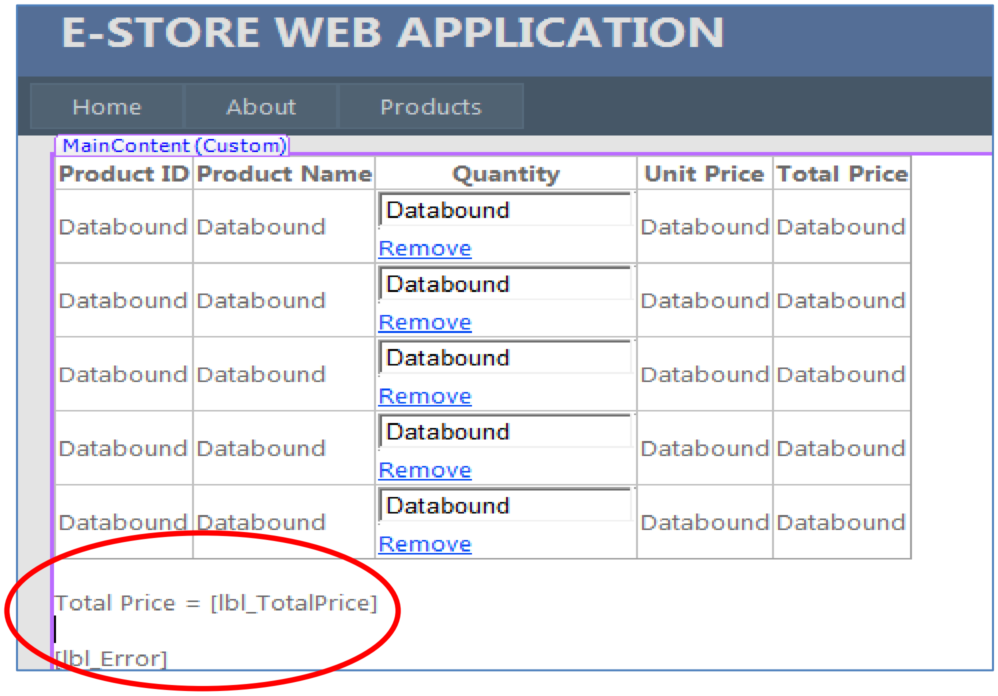


STEP 4: DISPLAYING YOUR SHOPPING CART

11. Add codes and logic to "ViewCart.aspx"

- Drag a GridView. Set Properties : ID – **gv_CartView**





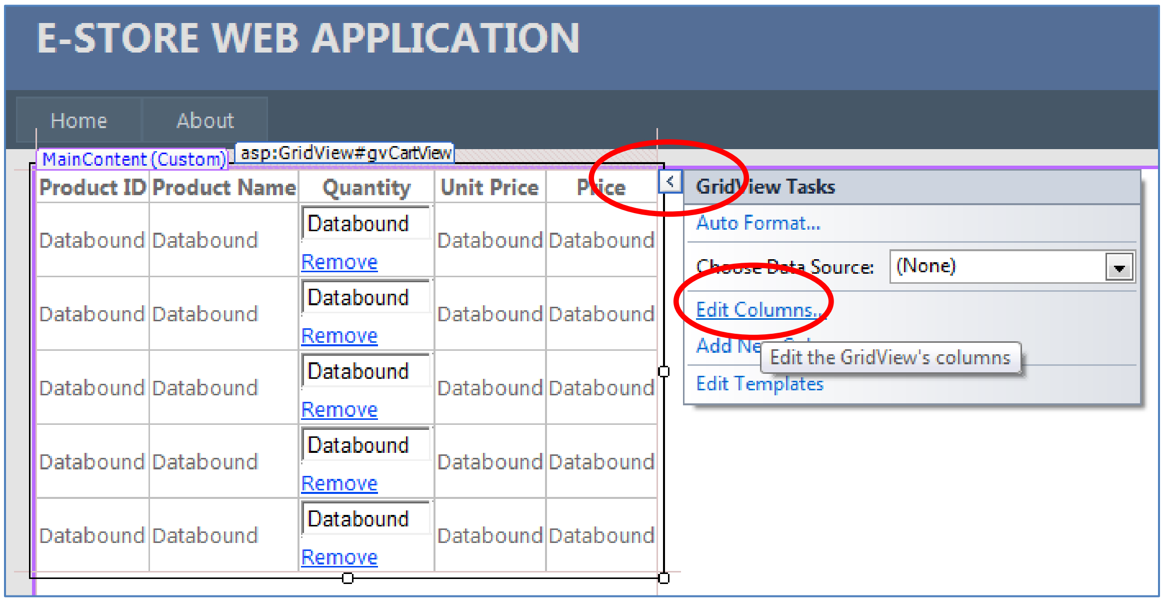
Product ID	Product Name	Quantity	Unit Price	Total Price
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound

Total Price = [lbl_TotalPrice]

[lbl_Error]

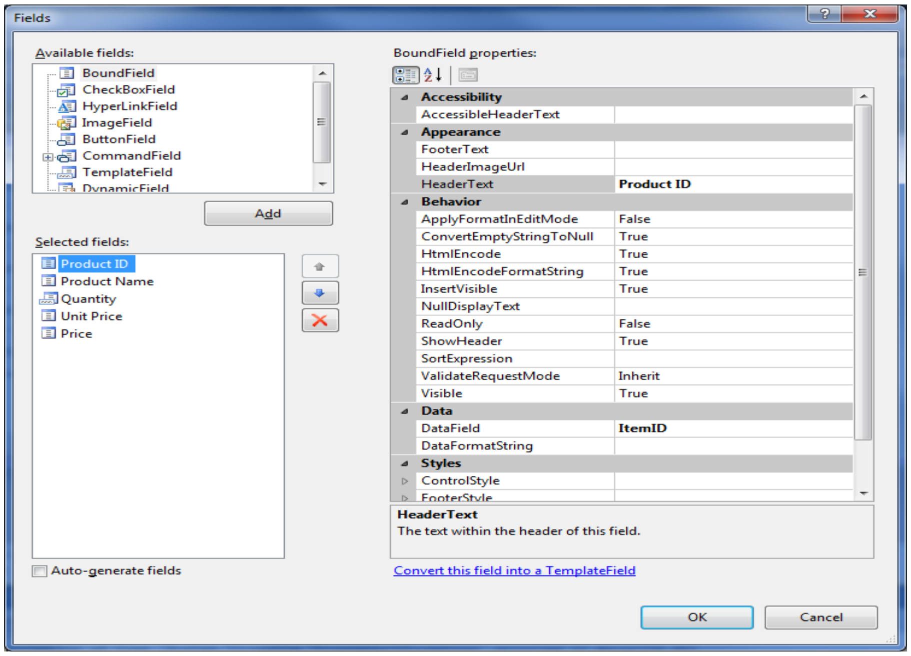
- Add 2 labels just below the GridView. **lbl_TotalPrice** and **lbl_Error**.

- Configure the GridView. Set the ID to **gv_CartView**
- Set the Grid View Column as shown below. The values from the session will populate the GridView nicely if it is configured correctly.
- Edit Columns.



Product ID	Product Name	Quantity	Unit Price	Price
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound

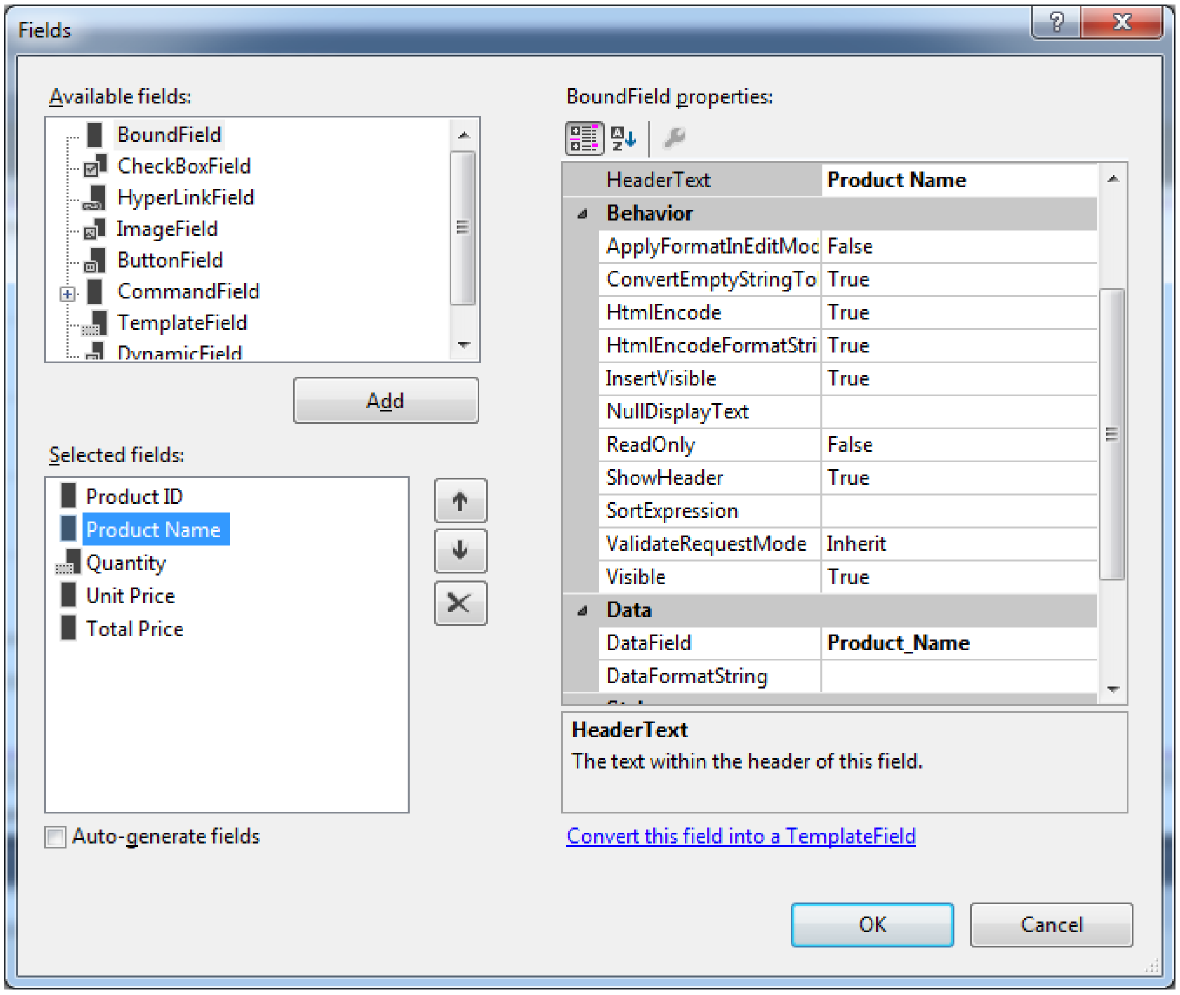
- Uncheck **Auto-generate fields**
- Select **BoundField** → Add
- Set **Header Text** – Product ID
- Set **DataField** – ItemID (similar to ShoppingCartItem class)



Accessibility	
AccessibleHeaderText	
Appearance	
FooterText	
HeaderImageUrl	
HeaderText	Product ID
Behavior	
ApplyFormatInEditMode	False
ConvertEmptyStringToNull	True
HtmlEncode	True
HtmlEncodeFormatString	True
InsertVisible	True
NullDisplayText	
ReadOnly	False
ShowHeader	True
SortExpression	
ValidateRequestMode	Inherit
Visible	True
Data	
DataField	ItemID
DataFormatString	
Styles	
ControlStyle	
FooterStyle	

HeaderText
The text within the header of this field.

- Select **BoundField** → Add
- Set **HeaderText** – Product Name
- Set **DataField** - Product_Name

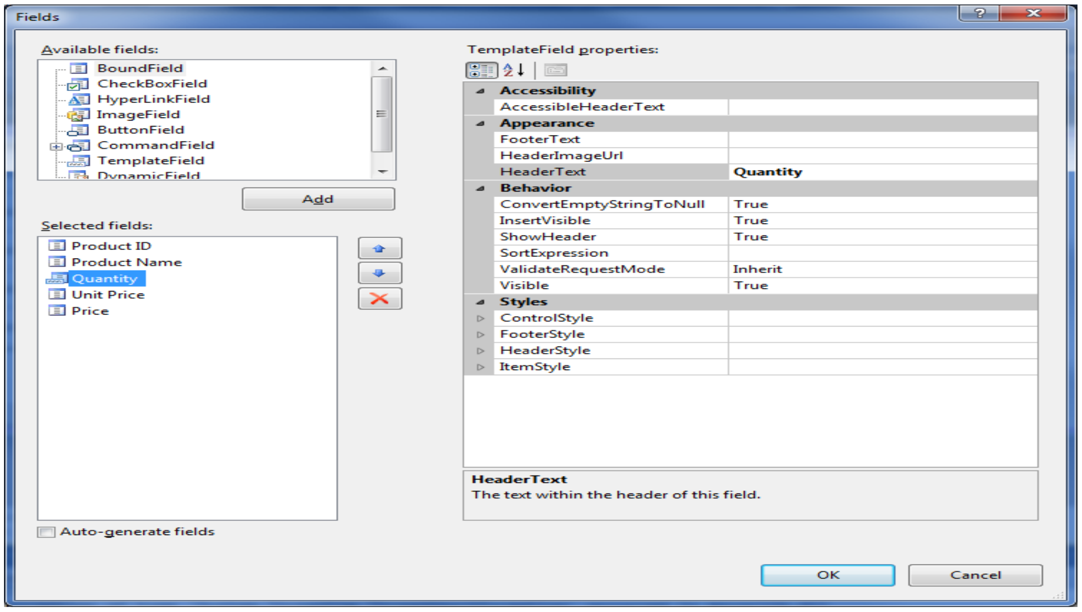


HeaderText	**Product Name**
Behavior	
ApplyFormatInEditMoc	False
ConvertEmptyStringTo	True
HtmlEncode	True
HtmlEncodeFormatStri	True
InsertVisible	True
NullDisplayText	
ReadOnly	False
ShowHeader	True
SortExpression	
ValidateRequestMode	Inherit
Visible	True
Data	
DataField	**Product_Name**
DataFormatString	

HeaderText
The text within the header of this field.

Not BoundField

- Select **Template Field** – Add
- Set **HeaderText** - Quantity

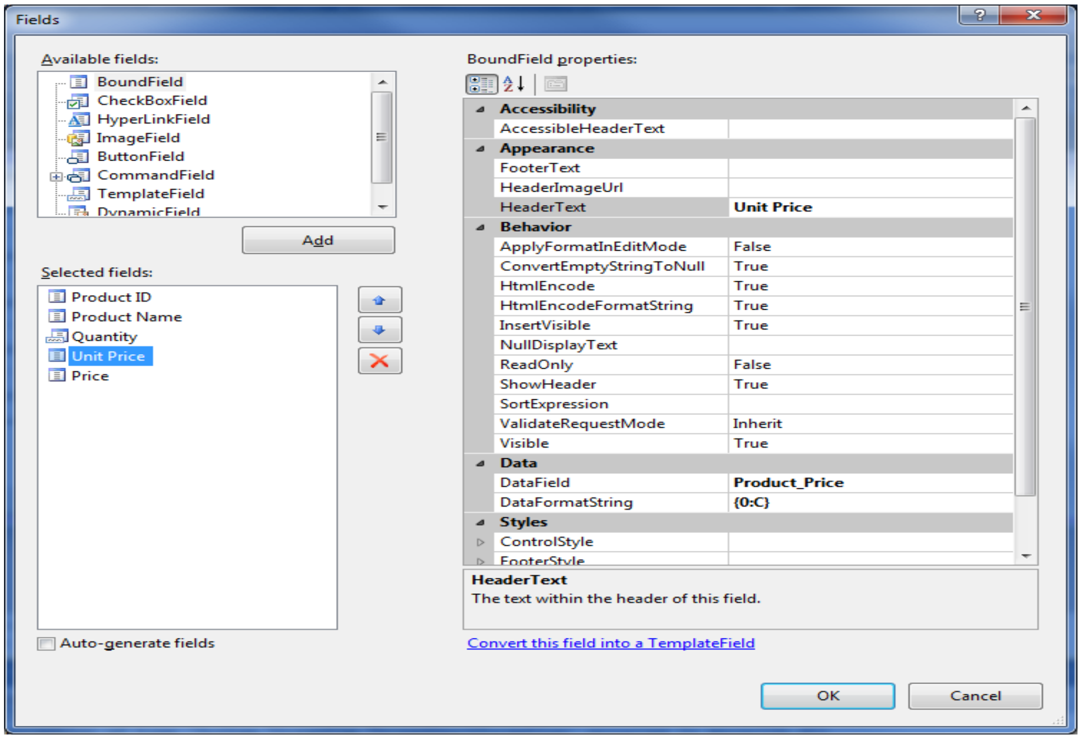


TemplateField properties:

Accessibility	
AccessibleHeaderText	
Appearance	
FooterText	
HeaderImageUrl	
HeaderText	**Quantity**
Behavior	
ConvertEmptyStringToNull	True
InsertVisible	True
ShowHeader	True
SortExpression	
ValidateRequestMode	Inherit
Visible	True
Styles	
ControlStyle	
FooterStyle	
HeaderStyle	
ItemStyle	

HeaderText
The text within the header of this field.

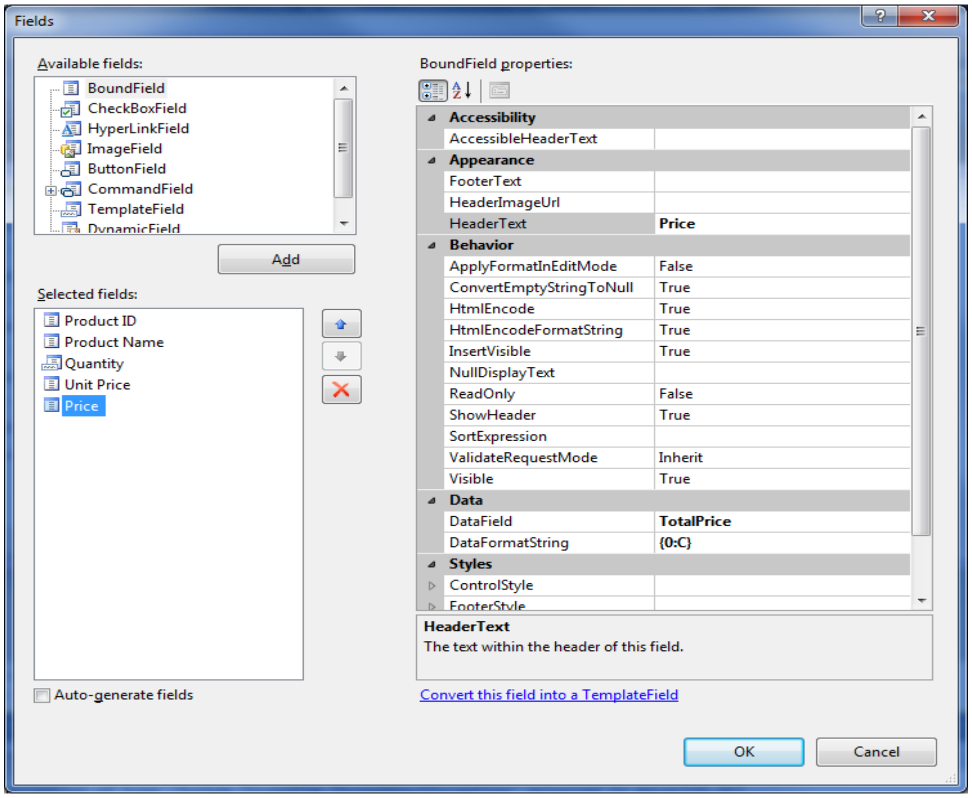
- Select **BoundField** – Add
- Set **HeaderText** – Unit Price
- Set **DataField** – Product_Price
- Set **DataFormatString** – **{0:C}** to display the $ sign.



Accessibility	
AccessibleHeaderText	
Appearance	
FooterText	
HeaderImageUrl	
HeaderText	**Unit Price**
Behavior	
ApplyFormatInEditMode	False
ConvertEmptyStringToNull	True
HtmlEncode	True
HtmlEncodeFormatString	True
InsertVisible	True
NullDisplayText	
ReadOnly	False
ShowHeader	True
SortExpression	
ValidateRequestMode	Inherit
Visible	True
Data	
DataField	**Product_Price**
DataFormatString	**{0:C}**
Styles	
ControlStyle	
FooterStyle	

HeaderText
The text within the header of this field.

- Set **BoundField**
- Set **HeaderText** – Price
- Set **DataField** – TotalPrice
- Set **DataFormatString** – **{0:C}** to display the $ sign.

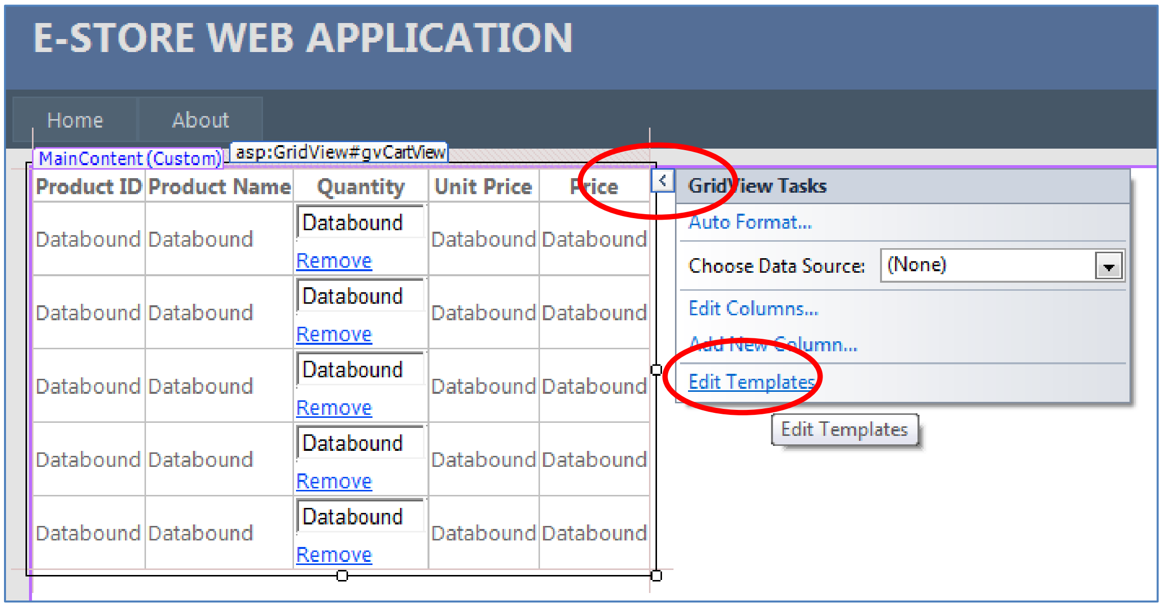


Selected fields:
- Product ID
- Product Name
- Quantity
- Unit Price
- Price

Accessibility	
AccessibleHeaderText	
Appearance	
FooterText	
HeaderImageUrl	
HeaderText	**Price**
Behavior	
ApplyFormatInEditMode	False
ConvertEmptyStringToNull	True
HtmlEncode	True
HtmlEncodeFormatString	True
InsertVisible	True
NullDisplayText	
ReadOnly	False
ShowHeader	True
SortExpression	
ValidateRequestMode	Inherit
Visible	True
Data	
DataField	**TotalPrice**
DataFormatString	**{0:C}**
Styles	
ControlStyle	
FooterStyle	

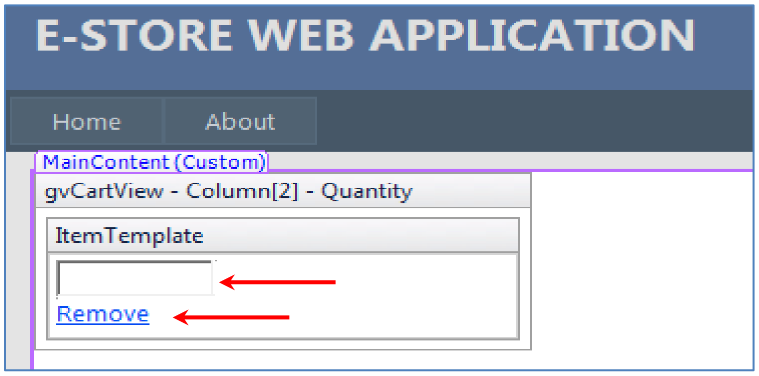
HeaderText
The text within the header of this field.

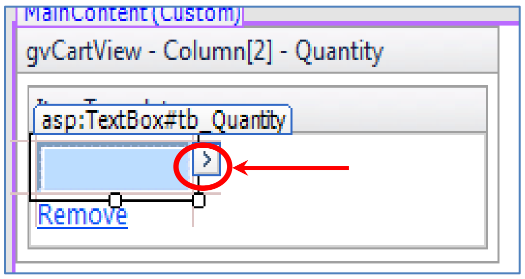
- Customize the Quantity Template Field. Select “Edit Template”.



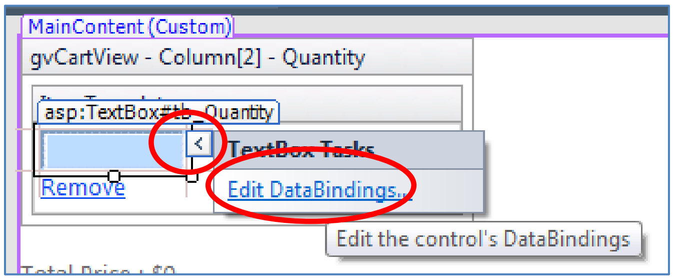
Product ID	Product Name	Quantity	Unit Price	Price
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound

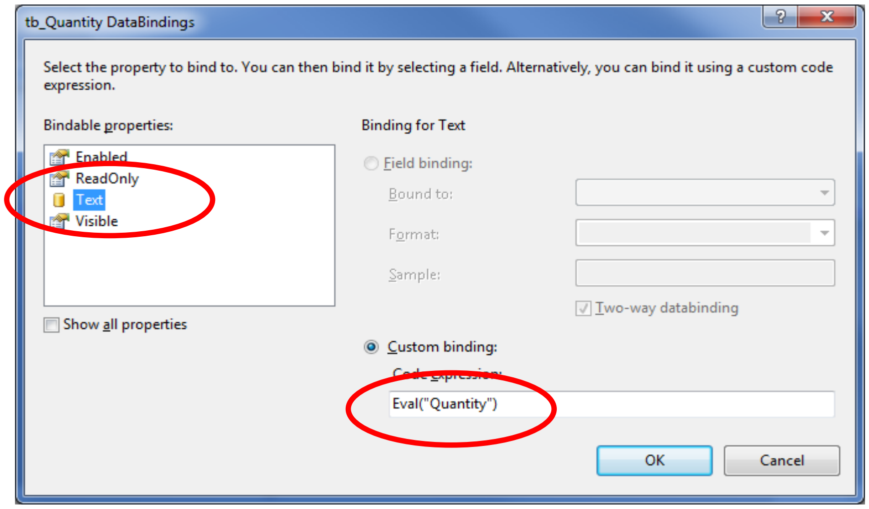
- Drag a TextBox into the ItemTemplate. Set Properties : ID – **tb_Quantity**
- Drag a **Link** Button into the ItemTemplate. Set Properties : ID – **btn_Remove**





- Edit the DataBindings for Textbox. Set Code Expression to Eval(“Quantity”).
- By doing so, data from the Quantity property of the ShoppingCartItem will be displayed in the textbox provided due to databinding operation.




tb_Quantity 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

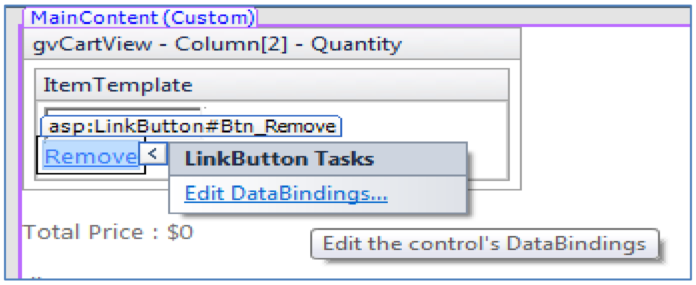
Show all properties

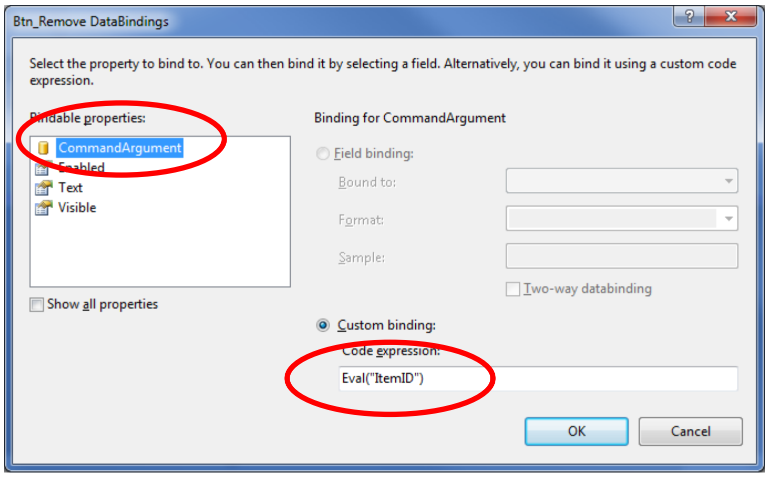
Binding for Text

Field binding:
Bound to:
Format:
Sample:
Two-way databinding

Custom binding:
Code expression:
Eval("Quantity")

- Edit the DataBindings for Link Button. Set Code Expression of the CommandArgument to Eval(“ItemID”). By doing so, we will be able to identify the correct Product ID (ItemID) that is to be removed from the Cart.




Btn_Remove 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:
- CommandArgument
- Enabled
- Text
- Visible

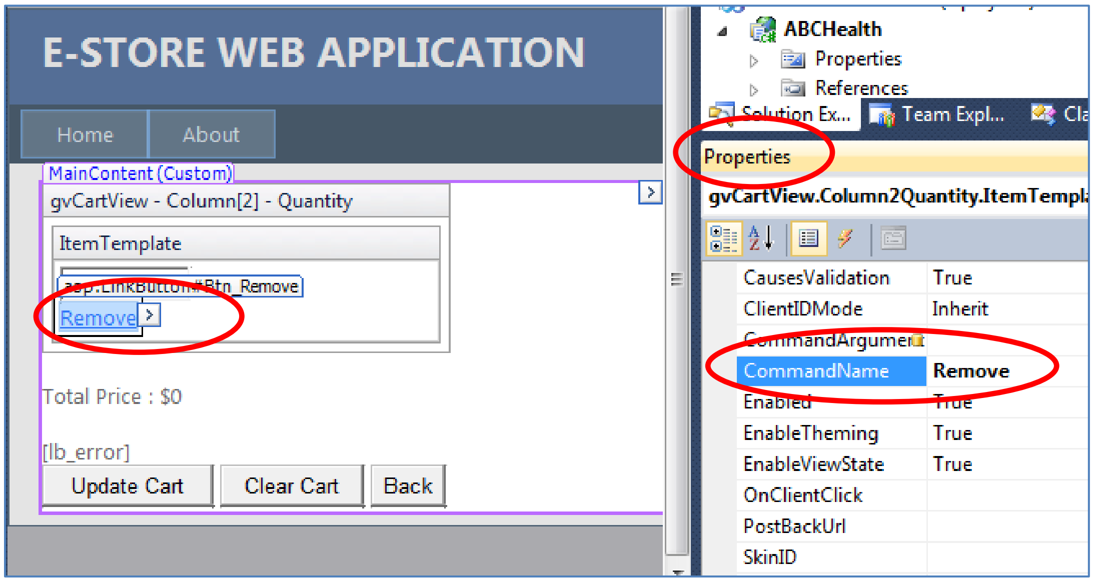
Show all properties

Binding for CommandArgument

Field binding:
Bound to:
Format:
Sample:
Two-way databinding

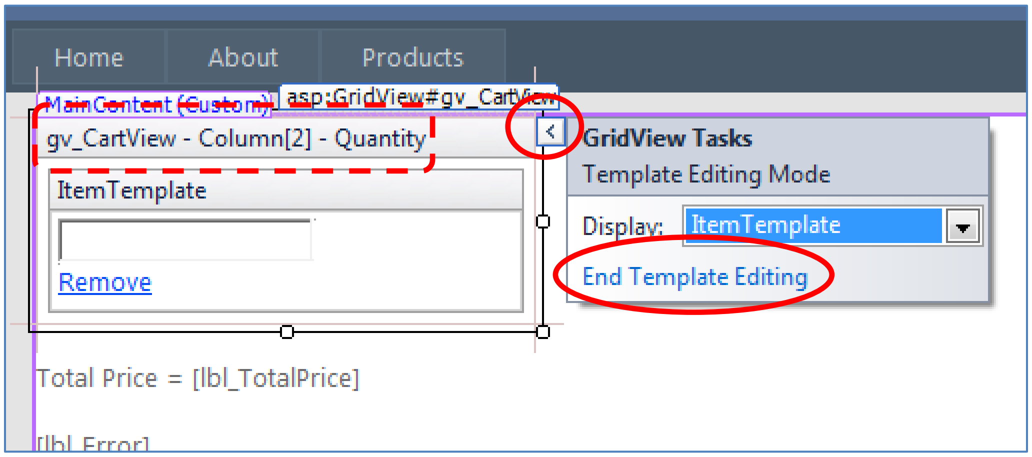
Custom binding:
Code expression:
Eval("ItemID")

- Set properties of "btn_Remove" : CommandName as 'Remove'
- Without this setting, the "Remove" link button will be rendered useless.

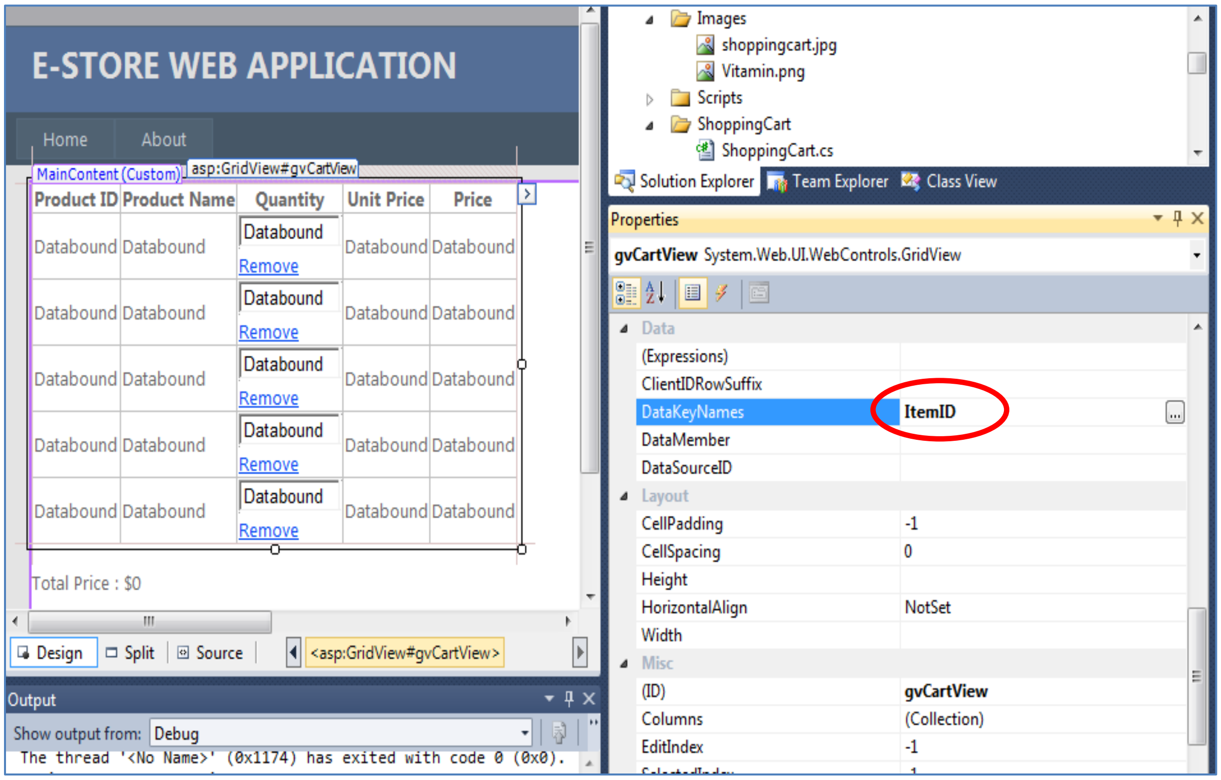


CausesValidation	True
ClientIDMode	Inherit
CommandArgument	
CommandName	Remove
Enabled	True
EnableTheming	True
EnableViewState	True
OnClientClick	
PostBackUrl	
SkinID	

- To end the "Edit Template" session, click "**gv_CartView – Column** ..." and select "End Template Editing".



- Set GridView Properties : DataKeyNames – **ItemID** for indexing purpose.



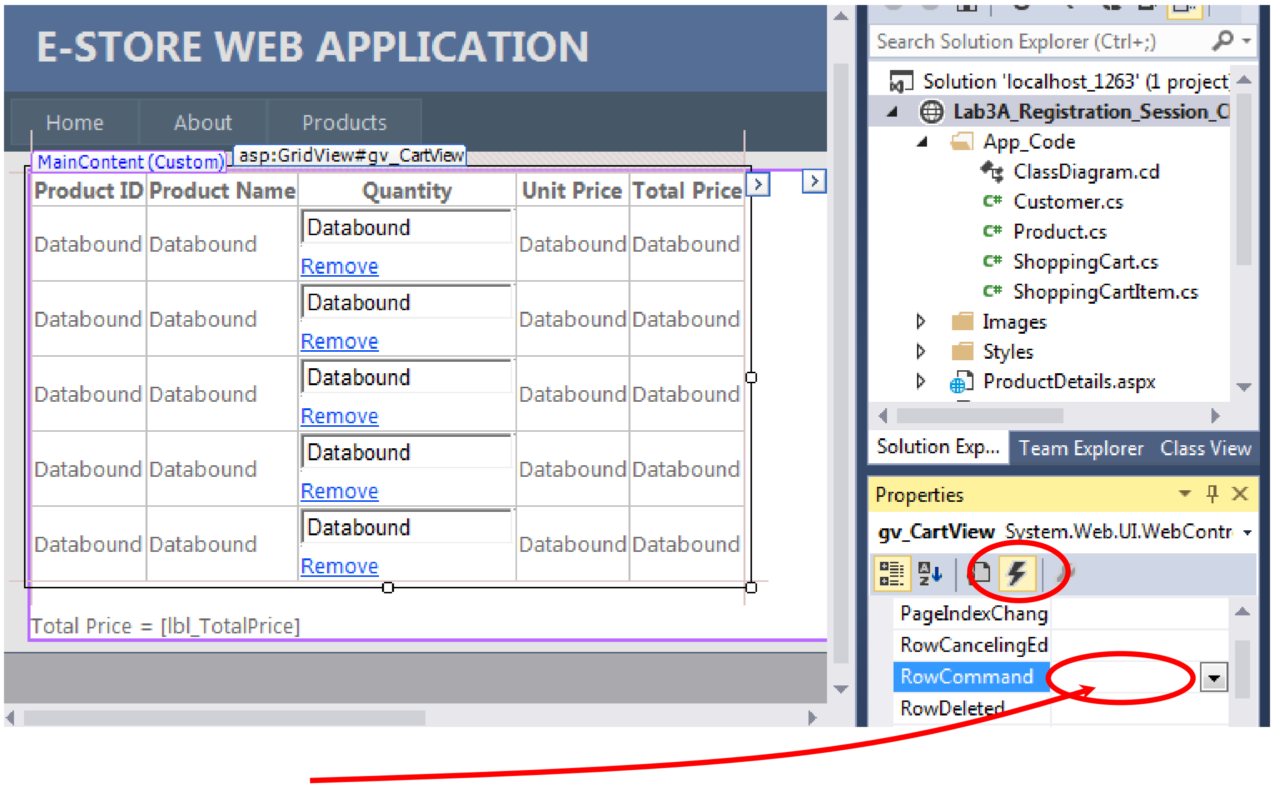
Product ID	Product Name	Quantity	Unit Price	Price
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound

gvCartView System.Web.UI.WebControls.GridView

Data	
(Expressions)	
ClientIDRowSuffix	
DataKeyNames	**ItemID**
DataMember	
DataSourceID	
Layout	
CellPadding	-1
CellSpacing	0
Height	
HorizontalAlign	NotSet
Width	
Misc	
(ID)	**gvCartView**
Columns	(Collection)
EditIndex	-1

```
The thread '<No Name>' (0x1174) has exited with code 0 (0x0).
```

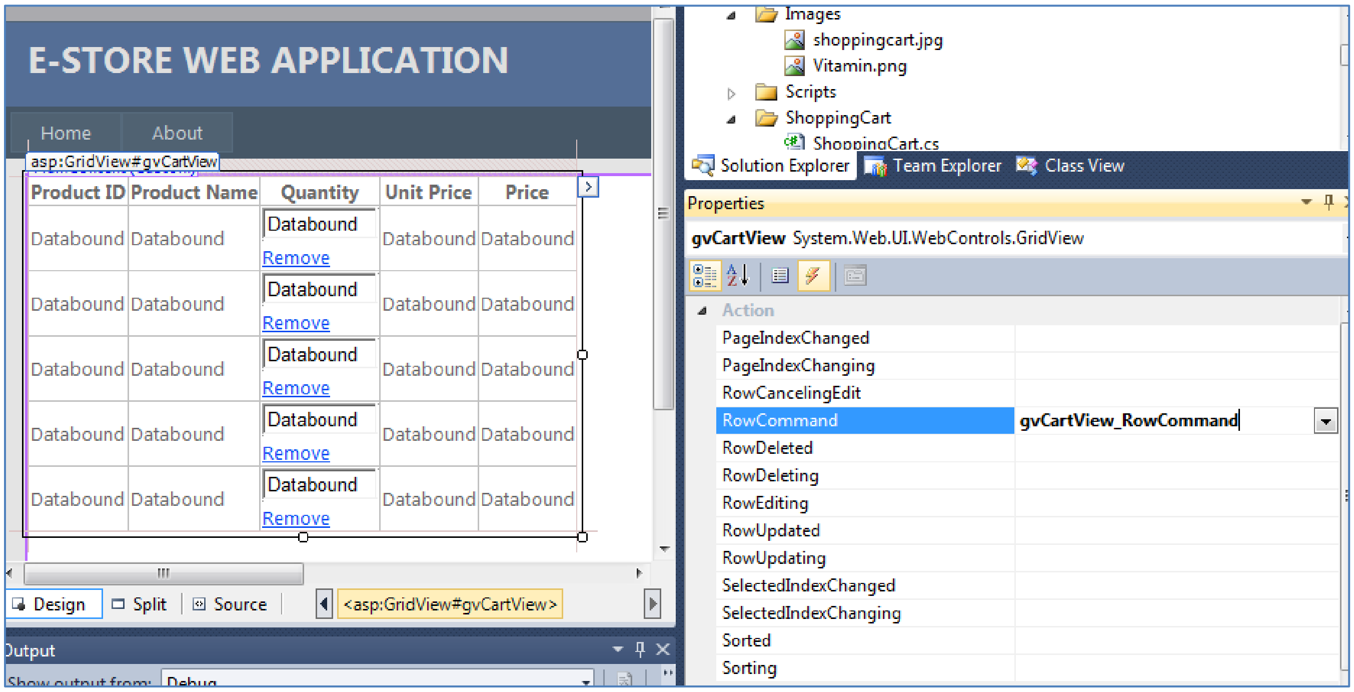
- Under GridView Properties, set RowCommand properties and add codes.



Product ID	Product Name	Quantity	Unit Price	Total Price
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound

Total Price = [lbl_TotalPrice]

- Double-click on the space shown to activate the event handler (Do not copy-paste the code, it wont work)



E-STORE WEB APPLICATION

Product ID	Product Name	Quantity	Unit Price	Price
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound

Properties

gvCartView System.Web.UI.WebControls.GridView

Action	
PageIndexChanged	
PageIndexChanging	
RowCancelingEdit	
RowCommand	gvCartView_RowCommand
RowDeleted	
RowDeleting	
RowEditing	
RowUpdated	
RowUpdating	
SelectedIndexChanged	
SelectedIndexChanging	
Sorted	
Sorting	

- Add codes to the code-behind.

```
protected void gv_CartView_RowCommand(object sender, GridViewCommandEventArgs e)
{
     if (e.CommandName == "Remove")
     {
       lbl_Error.Text = "Message : " + e.CommandArgument.ToString();
       string productId = e.CommandArgument.ToString();
       ShoppingCart.Instance.RemoveItem(productId);
       LoadCart();
     }
}
```

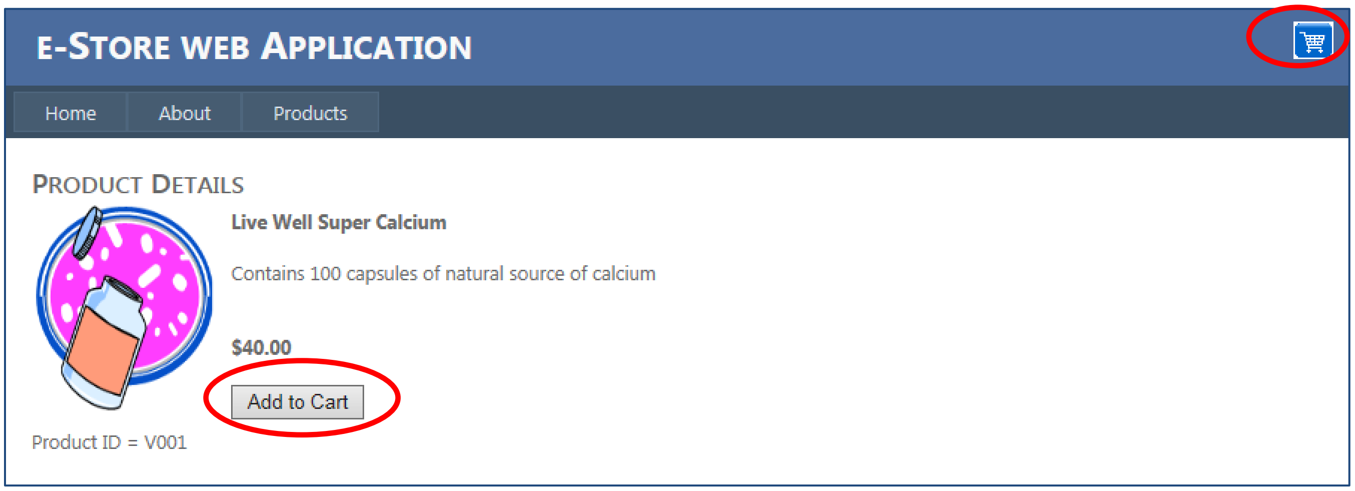
- Update ViewCart.aspx.cs with the following codes (Check Appendix A):

//read through before copy and paste.

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

```
    protected void LoadCart()
    {
        //bind the Items inside the Session/ShoppingCart Instance with the Datagrid
        gv_CartView.DataSource = ShoppingCart.Instance.Items;
        gv_CartView.DataBind();

        decimal total = 0.0m;
        foreach (ShoppingCartItem item in ShoppingCart.Instance.Items)
        {
            total = total + item.TotalPrice;
        }
        lbl_TotalPrice.Text = total.ToString();
    }
```

12. Test run your web application now. Make sure that ProductDetails.aspx is set to the "start up" page.



PRODUCT DETAILS

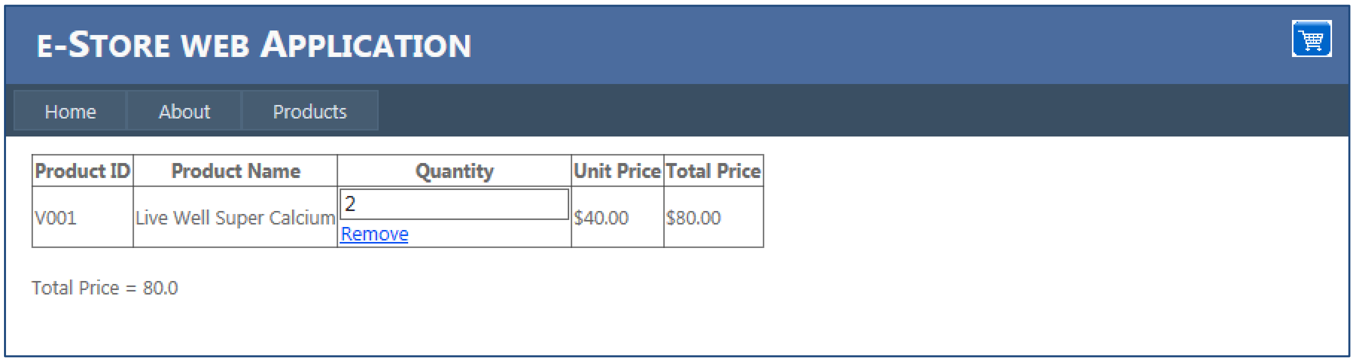
Live Well Super Calcium

Contains 100 capsules of natural source of calcium

$40.00

Product ID = V001

- Add 2 similar products into the Cart by clicking on the "Add to Cart" button twice.
- Next, click on the Shopping Cart at the upper right corner.

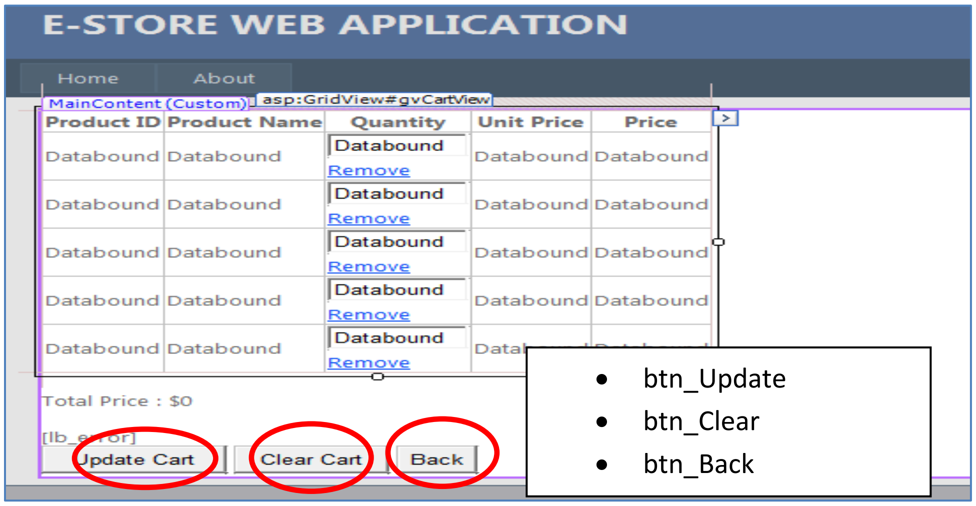


Product ID	Product Name	Quantity	Unit Price	Total Price
V001	Live Well Super Calcium	2 Remove	$40.00	$80.00

Total Price = 80.0

STEP 5: EXTENDING YOUR SHOPPING CART FEATURES

13. Extending your Shopping Cart functionalities. Although now that you are able to add items into the shopping cart, you are still not able to edit the items inside the shopping cart or go back to the Product details page to add more items.

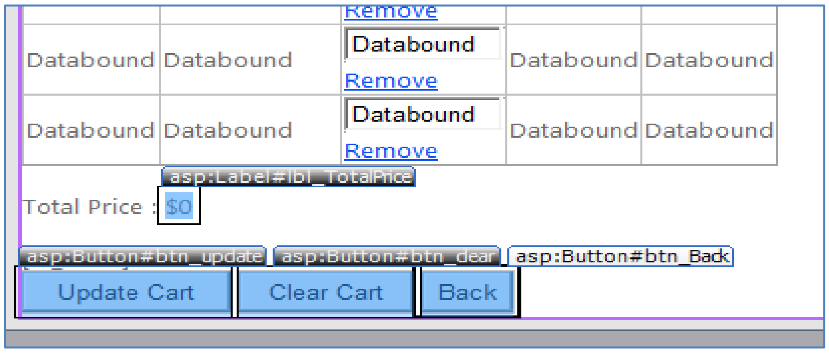


Product ID	Product Name	Quantity	Unit Price	Price
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Datal	

Total Price : $0

[lb_error]

- btn_Update
- btn_Clear
- btn_Back



		Remove		
Databound	Databound	Databound Remove	Databound	Databound
Databound	Databound	Databound Remove	Databound	Databound

Total Price : $0

14. Add codes to all the functions in the ViewCart.aspx.cs. See Appendix A & B for all codes.

15. Double click on “Back” button. Add codes :

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

- Test your web app. Add more items by using the “Back” button.
-

16. Double click on “Clear Cart” button. Add codes :

```
protected void btn_Clear_Click(object sender, EventArgs e)
{
   ShoppingCart.Instance.Items.Clear();
   LoadCart();
}
```

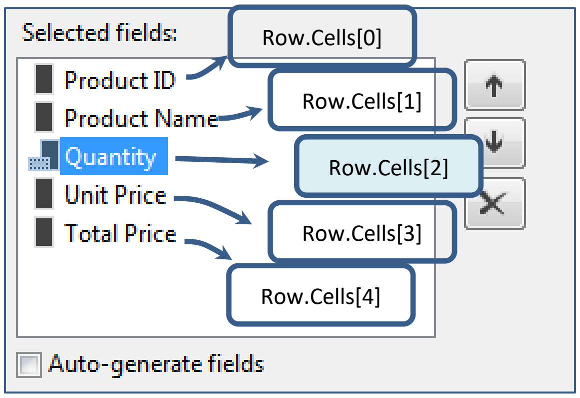
- Test your web app. Add more items by using the “Back” button and then clear the Shopping Cart.

17. “Update Cart” should allow users to make changes to the quantity of the items they are about to purchase.

18. Double click on “Update Cart” button. Add codes :

```
 //After clicking on the Update function, the Quantity in the textbox needs to be updated back to the
ShoppingCart
 protected void btn_Update_Click(object sender, EventArgs e)
 {
      foreach (GridViewRow row in gv_CartView.Rows)
      {
          if (row.RowType == DataControlRowType.DataRow)
          {
              try
              {
                  string productId = gv_CartView.DataKeys[row.RowIndex].Value.ToString();

                  //row.Cells[2] means that the quantity textbox must be in column 3.
                  int quantity = int.Parse(((TextBox)row.Cells[2].FindControl("tb_Quantity")).Text);
                   ShoppingCart.Instance.SetItemQuantity(productId, quantity);
              }
              catch (FormatException e1) {
                  lbl_Error.Text = e1.Message.ToString();
              }
         }
      }
      LoadCart();
 }
```

Note : Please take note of the position of the row cells. If your sequence is different from the one shown here, please make necessary adjustment to your codes.



Appendix A

(ViewCart.cs)

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

public partial class ViewCart : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            LoadCart();
        }
    }

    protected void LoadCart()
    {
        //bind the Items inside the Session/ShoppingCart Instance with the Datagrid
        gv_CartView.DataSource = ShoppingCart.Instance.Items;
        gv_CartView.DataBind();

        decimal total = 0.0m;
        foreach (ShoppingCartItem item in ShoppingCart.Instance.Items)
        {
            total = total + item.TotalPrice;
        }
        lbl_TotalPrice.Text = total.ToString();
    }


    protected void gv_CartView_RowCommand1(object sender, GridViewCommandEventArgs e)
    {
        if (e.CommandName == "Remove")
        {
            lbl_Error.Text = "Message : " + e.CommandArgument.ToString();
            string productId = e.CommandArgument.ToString();
            ShoppingCart.Instance.RemoveItem(productId);
            LoadCart();
        }

    }
    protected void btn_Back_Click(object sender, EventArgs e)
    {
        Response.Redirect("ProductDetails.aspx");
    }
    protected void btn_Clear_Click(object sender, EventArgs e)
    {
        ShoppingCart.Instance.Items.Clear();
        LoadCart();

    }

 protected void btn_Update_Click(object sender, EventArgs e)
 {
        foreach (GridViewRow row in gv_CartView.Rows)
        {
           if (row.RowType == DataControlRowType.DataRow)
           {
             try
             {
                string productId = gv_CartView.DataKeys[row.RowIndex].Value.ToString();
                //row.Cells[2] means that the quantity textbox must be in column 3.
                int quantity =
int.Parse(((TextBox)row.Cells[2].FindControl("tb_Quantity")).Text);
                ShoppingCart.Instance.SetItemQuantity(productId, quantity);
              }
              catch (FormatException e1)
              {
                lbl_Error.Text = e1.Message.ToString();
              }
          }
        }
        LoadCart();
  }
}
```

Appendix B
(ProductDetails.cs)

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


public partial class ProductDetails : System.Web.UI.Page
{
    Product prod = null;
    protected void Page_Load(object sender, EventArgs e)
    {
        //*** Set some dummy data until the next practical when we connected to a DB.***

        string prodID = "V001";
        string prodName = "Live Well Super Calcium";
        string prodDesc = "Contains 100 capsules of natural source of calcium";
        decimal prodPrice = 40.0M;
        string prodImage = "Vitamin.png";
        int prodStock = 10;
        prod = new Product(prodID,prodName, prodDesc, prodPrice, prodImage, prodStock);
        // *** End Dummy Data

        // **** Display the Dummy Data on the form ****
        lbl_ProdID.Text = prod.Product_ID.ToString();
        lbl_ProdName.Text = prod.Product_Name.ToString();
        lbl_ProdDesc.Text = prod.Product_Desc.ToString();
        img_Product.ImageUrl = "~\\Images\\" + prod.Product_Image;
        lbl_Price.Text = prod.Unit_Price.ToString("C");
        // **** End Display the Dummy Data on the form ****

    }
    protected void Btn_Add_Click(object sender, EventArgs e)
    {
        string iProductID = prod.Product_ID.ToString();
        ShoppingCart.Instance.AddItem(iProductID, prod);

    }
}
```

--- The End --