



School of Information Technology

---

**Practical 07: Database Part 1 (Retrieve)**

**OBJECTIVES:**

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

- Create a database file (MDF)
- Create a table and display all records from the table in a web form

## Before you start:

Take a moment to understand requirement for each page:

Product page, **ProductView.aspx**

- Display all records from *Product* database table using a GridView control.
- When user selects a particular row (i.e. product), system will re-direct user to the ProductDetails.aspx where details of selected product is showed.

Product's detail page, **ProductDetails.aspx** (*we have created this in our shopping cart workshop*)

- At the Page Load event, display details of the selected product.

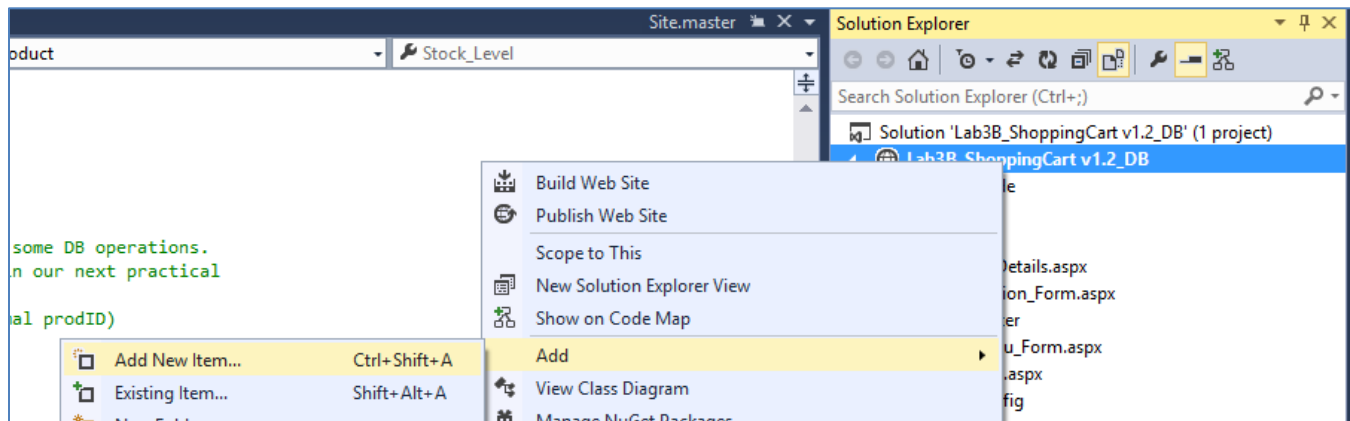
Product.cs

- Create 2 additional methods
  - getProduct() – returns a single Product Object by providing the ProductID.
  - getProductAll() – returns a list of All Products.

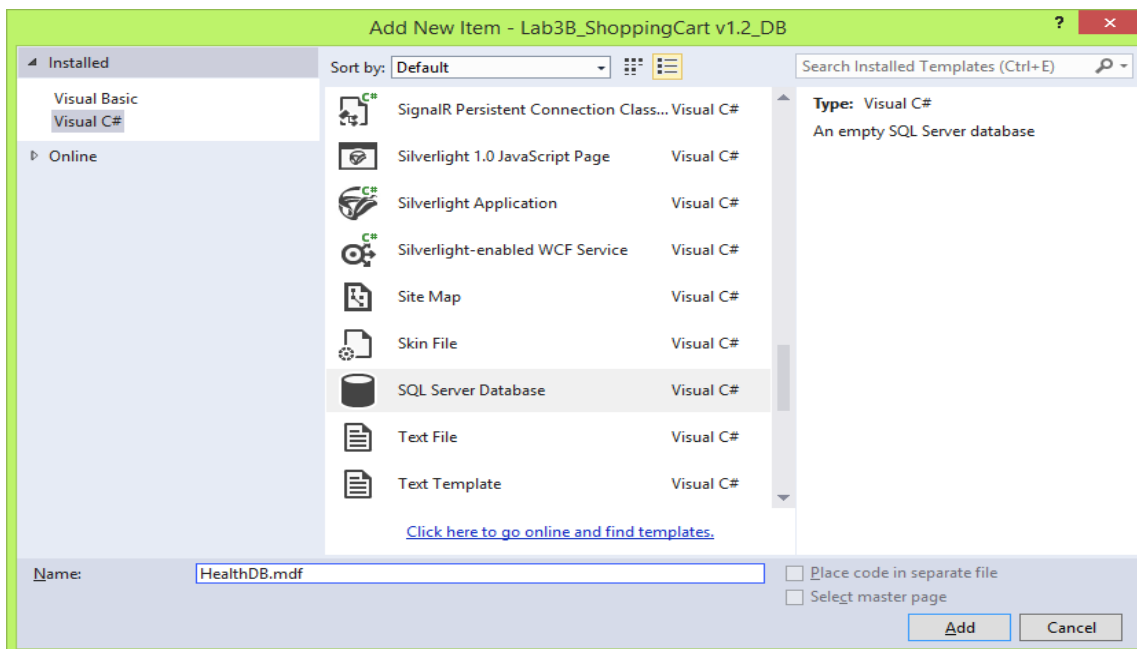
Let's Start!!

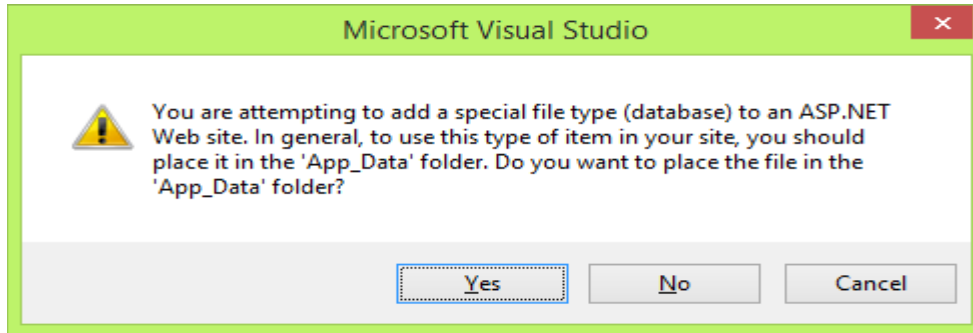
## Exercise 1 – Load Practical 06 Solution (shopping cart practical)

1. Start the Visual Studio.NET 2015. Select File → Open → Web Site → locate the root directory of your website.
2. Adding a Database file into your application. Right-Click “Web site name” → Add → Add new Item.

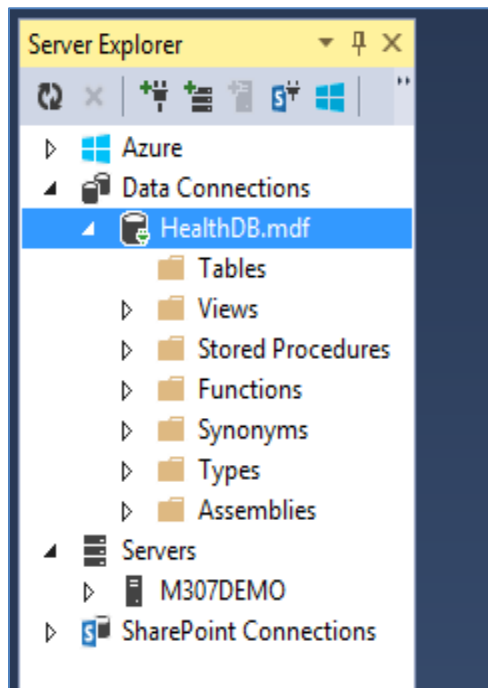


3. Under “Visual C#” section, select “SQL Server Database”. Name the file **HealthDB.mdf**.
4. Select “Yes” when asked if you would like to place your MDF file in App\_Data folder.
  - Also note that the App\_Data folder is the location where your MDF will be automatically saved, if you create a new or drag-drop a MDF file in VS.net.

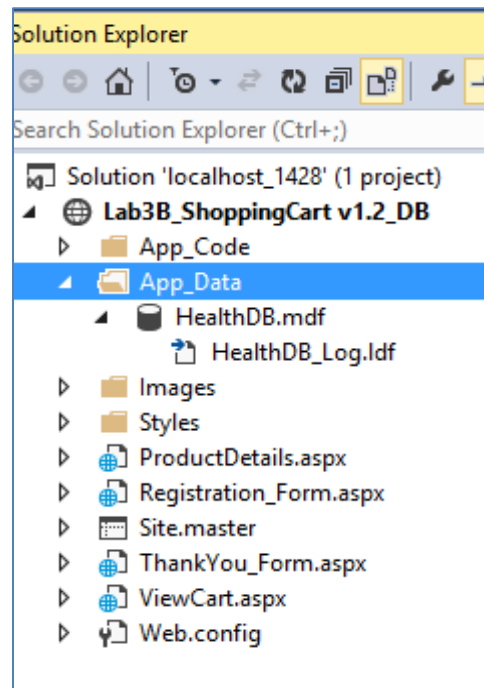




Notice the changes in Server Explorer and Solution Explorer after creating an EMPTY database file. A database connection is automatically added into the Server Explorer.



(Server Explorer)



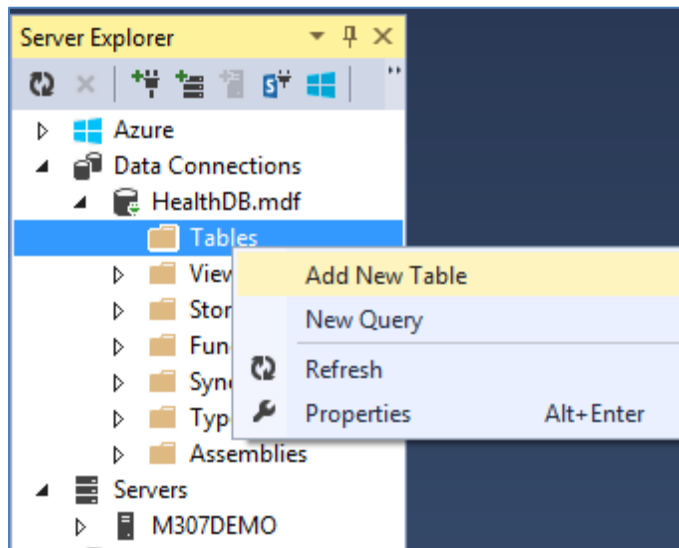
(Solution Explorer)

Note :

How to add a database file into your project..

- Drag drop the SQL Database File, xxyy.mdf, into App\_Data folder of VS.net.
- The log file, xxyy\_log.ldf, will be automatically added

5. Creating a Database Table in **Products**.

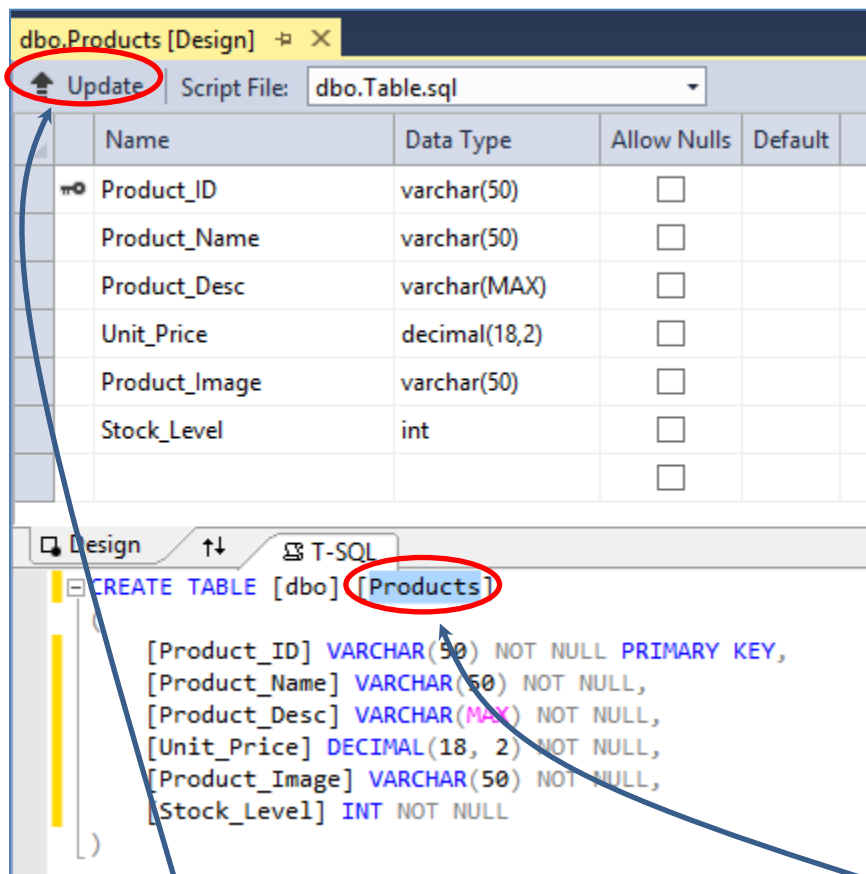


6. Set the Table Definition with the following fields and data type.

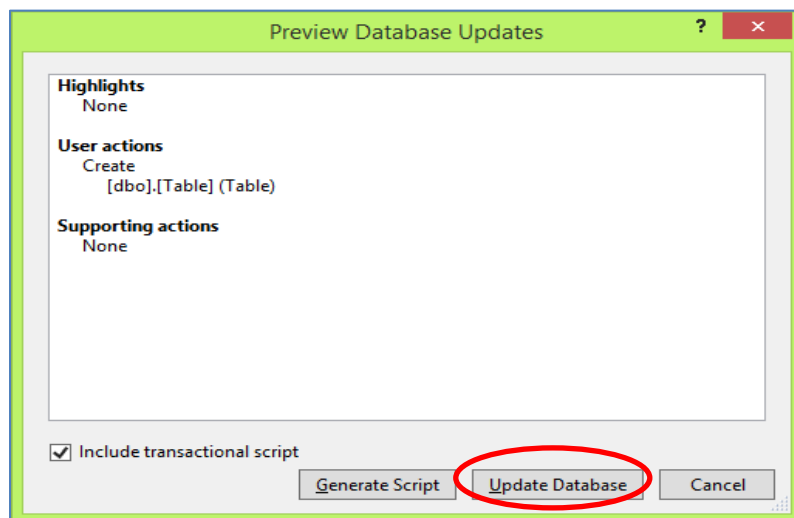
Product_ID	as varchar(50)
Product_Name	as varchar(50)
Product_Desc	as varchar(MAX)
Unit_Price	as decimal (18,2)
Product_Image	as varchar(50)
Stock_Level	as int

Compare these with  
your Product.cs class  
properties.

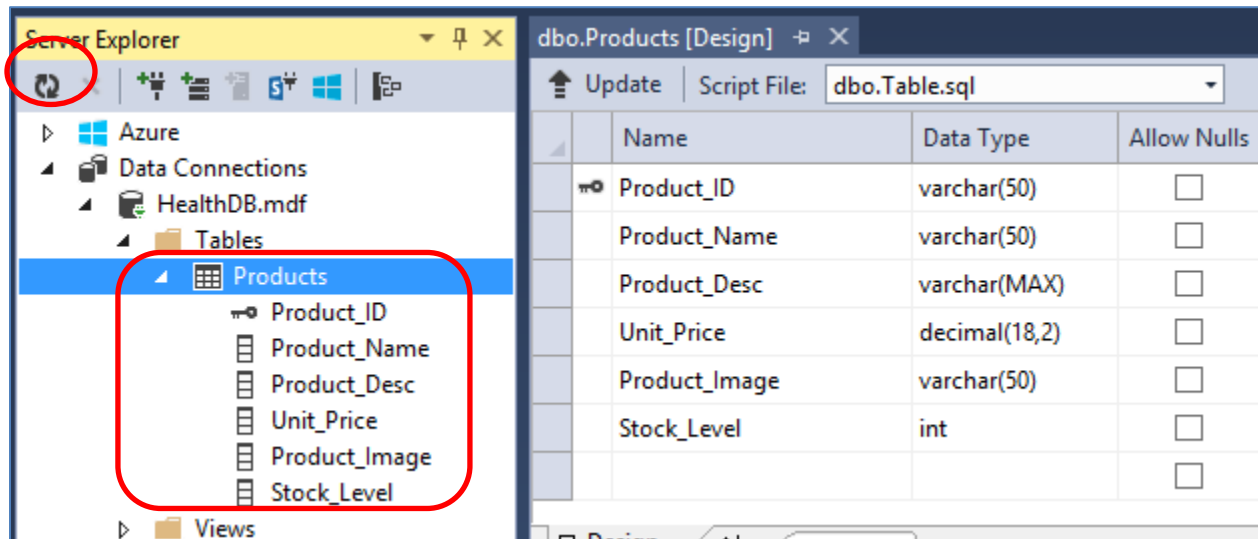
Note that *Product\_ID* is the primary key.



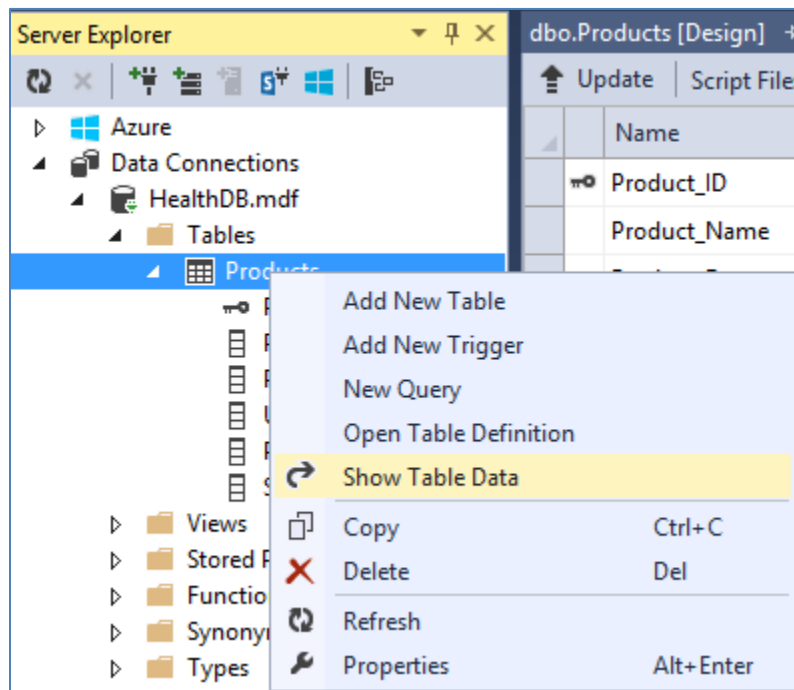
7. Remove the check for Allow Nulls
8. In the SQL Script windows (T-SQL), set the name of the DB table to “Products”
9. Click on “Update” when you are done.
10. In the Preview Database Updates windows, select “Update Database”. Select “Update Database” when prompted.



11. To reflect the latest changes, click on the “Refresh button”.



12. Right-Click on the “Product” table → Show Table Data.



13. Let's enter some sample data :

dbo.Products [Data] - X						
Max Rows: 1000						
Product_ID	Product_Name	Product_Desc	Unit_Price	Product_Image	Stock_Level	
1	LiveWell Vitamin C	A water and fat-soluble blend of Vitami...	36.00	Vitamin.png	100	
2	LiveWell Vitamin E	Livewell Vitamin E supply full spectrum ...	32.00	Vitamin.png	200	

Product\_ID – 1

Product\_Name - LiveWell Vitamin C

Product\_Desc - A water and fat-soluble blend of Vitamin C blended with a citrus bioflavonoids complex

Unit\_Price – 36.00

Product\_Image – Vitamin.png

Stock\_Level – 100

Product\_ID – 2

Product\_Name - LiveWell Vitamin E

Product\_Desc - Livewell Vitamin E supply full spectrum of vitamin E isomers including alpha, beta, delta and gamma tocopherols and tocotrienols.

Unit\_Price – 32.00

Product\_Image – Vitamin.png

Stock\_Level – 200

14. Specifying the Database Connection String in your Web.config file

Instead of hardcoding connection string (to Database File) in your codes, you can store the connection string in **Web.Config** file. This will make changing of connection string easier if there is a change in the location or database file name.

```
<connectionStrings>
  <add name="HealthDBContext"
    connectionString="Data Source=(LocalDB)\v11.0;
AttachDbFilename=|DataDirectory|\HealthDB.mdf; Integrated Security=True"
    providerName="System.Data.SqlClient"
  />
</connectionStrings>
```



```
<configuration>

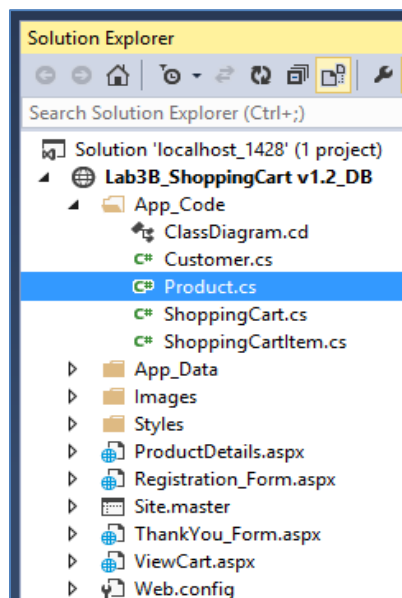
  <system.web>
    <compilation debug="true" targetFramework="4.5" />
    <httpRuntime targetFramework="4.5" />
  </system.web>

  <connectionStrings>
    <add name="HealthDBContext"
      connectionString="Data Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\HealthDB.mdf;Integrated Security=True"
      providerName="System.Data.SqlClient"
    />
  </connectionStrings>

</configuration>
```

#### 15. Adding database access codes to Product.cs.

- A Product class was created in previous practical. We will need to update the Product.cs file to include database codes along with some business logic.
- Update Product.cs to include the following:
  - Namespace for database operation:
    - `using System.Data;`
    - `using System.Data.SqlClient;`
  - Namespace for accessing ConfigurationManager
    - `using System.Configuration;`
  - Add a Class variable to point to the Database Connection String
    - `string _connStr = ConfigurationManager.ConnectionStrings["HealthDBContext"].ConnectionString;`
  - 2 class methods: `getProduct(productId)` and `getProductAll()`.
- Refer to Appendix A for the code listing for Product.cs.



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

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

/// <summary>
/// Summary description for Product
/// </summary>
23 references
public class Product
{
    //Private string _connStr = Properties.Settings.Default.DBConnStr;

    //System Configuration Connection String Settings _connStr:
    string connStr = ConfigurationManager.ConnectionStrings["HealthDBContext"].ConnectionString;
    private string _prodID = null;
    private string _prodName = string.Empty;
    private string _prodDesc = ""; // this is another way to specify empty string
    private decimal _unitPrice = 0;
    private string _prodImage = "";
    private int _stockLevel = 0;
```

16. Adding 2 methods to your Product.cs.

*Method : getProduct( )*

```
public Product getProduct(string prodID)
{
    Product prodDetail = null;

    string prod_Name, prod_Desc, Prod_Image;
    decimal unit_Price;
    int stock_Level;

    string queryStr = "SELECT * FROM Products WHERE Product_ID = @ProdID";
    SqlConnection conn = new SqlConnection(_connStr);
    SqlCommand cmd = new SqlCommand(queryStr, conn);
    cmd.Parameters.AddWithValue("@ProdID", prodID);
    conn.Open();
    SqlDataReader dr = cmd.ExecuteReader();

    //Check if there are any resultsets
    if (dr.Read())
    {
        prod_Name = dr["Product_Name"].ToString();
        prod_Desc = dr["Product_Desc"].ToString();
        Prod_Image = dr["Product_Image"].ToString();
        unit_Price = decimal.Parse(dr["Unit_Price"].ToString());
        stock_Level = int.Parse(dr["Stock_Level"].ToString());

        prodDetail = new Product(prodID, prod_Name, prod_Desc, unit_Price, Prod_Image, stock_Level);
    }
    else
    {
        prodDetail = null;
    }

    conn.Close();
    dr.Close();
    dr.Dispose();
    return prodDetail;
}
```

*Method : getProductAll( )*

*Returns a List of products – all products from the database*

```
public List<Product> getProductAll()
{
    List<Product> prodList = new List<Product>();

    string prod_Name, prod_Desc, Prod_Image, prod_ID;
    decimal unit_Price;
    int stock_Level;

    string queryStr = "SELECT * FROM Products Order By Product_Name";

    SqlConnection conn = new SqlConnection(_connStr);
    SqlCommand cmd = new SqlCommand(queryStr, conn);
    conn.Open();
    SqlDataReader dr = cmd.ExecuteReader();

    //Continue to read the resultsets row by row if not the end
    while (dr.Read())
    {
        prod_ID = dr["Product_ID"].ToString();
        prod_Name = dr["Product_Name"].ToString();
        prod_Desc = dr["Product_Desc"].ToString();
        Prod_Image = dr["Product_Image"].ToString();
        unit_Price = decimal.Parse(dr["Unit_Price"].ToString());
        stock_Level = int.Parse(dr["Stock_Level"].ToString());
        Product a = new Product(prod_ID, prod_Name, prod_Desc, unit_Price, Prod_Image, stock_Level);
        prodList.Add(a);
    }

    conn.Close();
    dr.Close();
    dr.Dispose();

    return prodList;
}
```

17. Next step will be to create the UI to display (view) all the products in the database.

## Exercise 2 – Create the ProductView.aspx

### ProductView.aspx

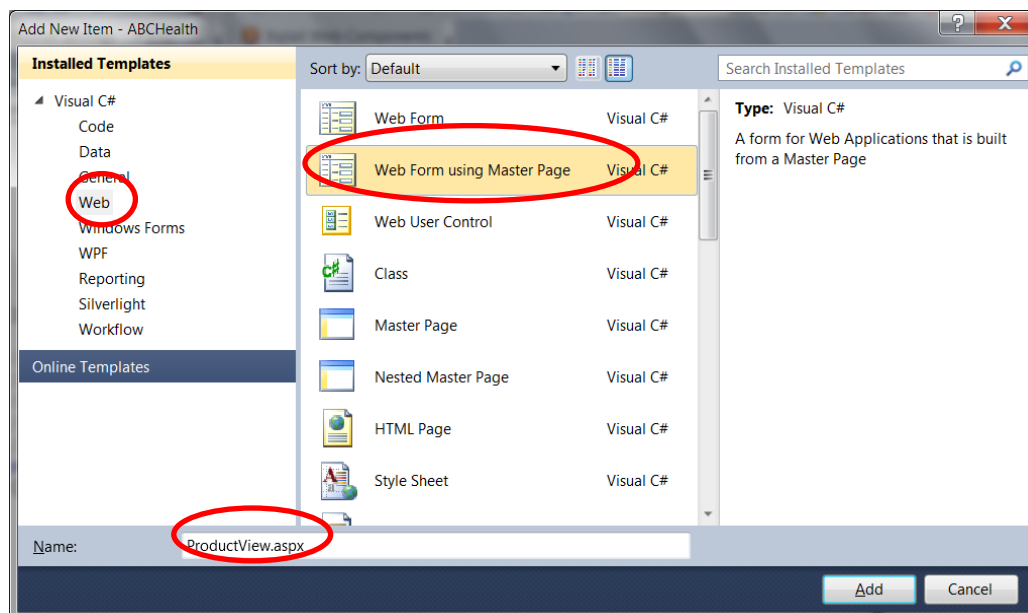
- Display all products using GridView control.
- At Page Load event, call bind() method to bind the database and datagrid.
- When user select a particular product from the GridView control, get Product ID of the selected product.
- Re-direct to the Product Detail page, ProductDetails.aspx. Product ID of the selected product will be passed over to Product Detail page using URL tagging mechanism.

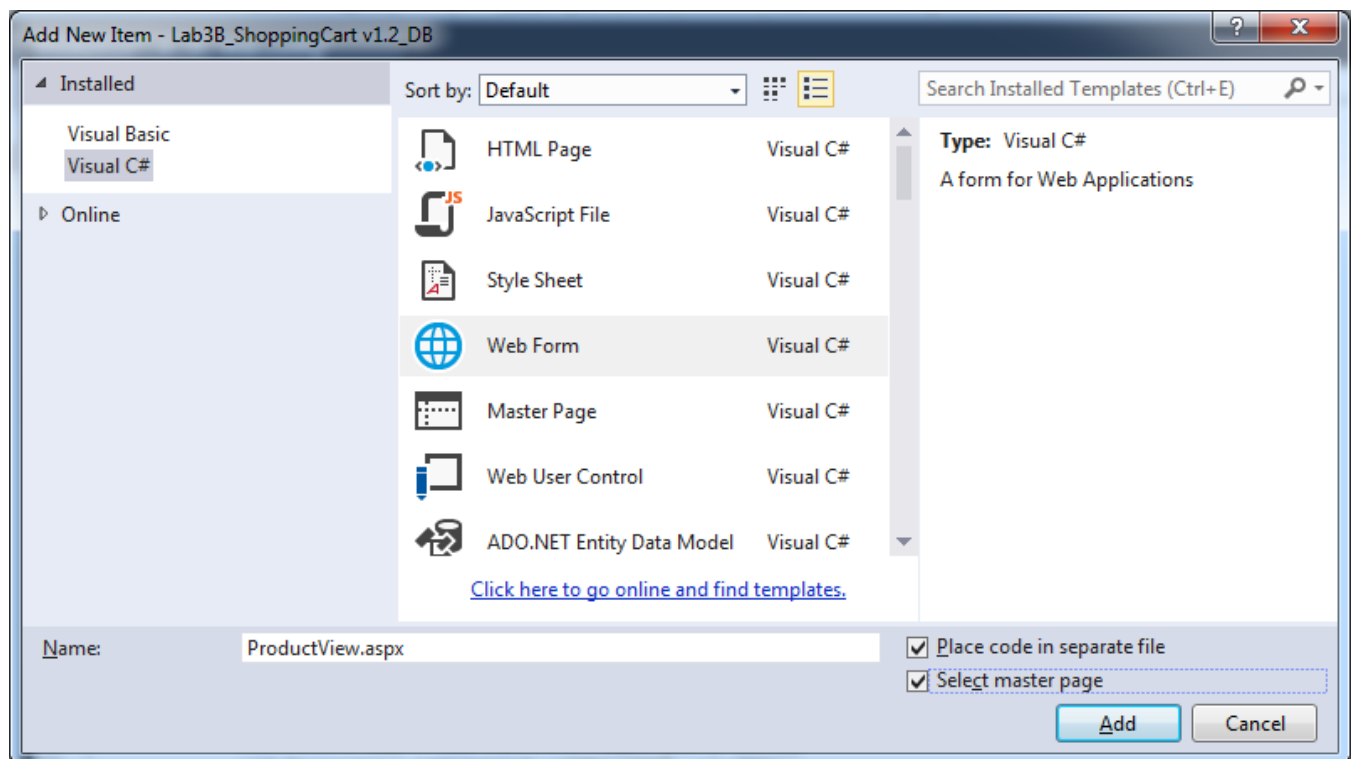
### ProductDetails.aspx (Already created in previous practical.)

- Display details of a selected product.
- At Page Load event, retrieve Product ID from URL using Request.QueryString() method.
- Call the getProdDetail() method of the Business Logic Layer, passing in Product ID as the input parameter. This method returns a Product object.
- Display Product name, description, image and price.

#### 1. Add web form **ProductView.aspx** .

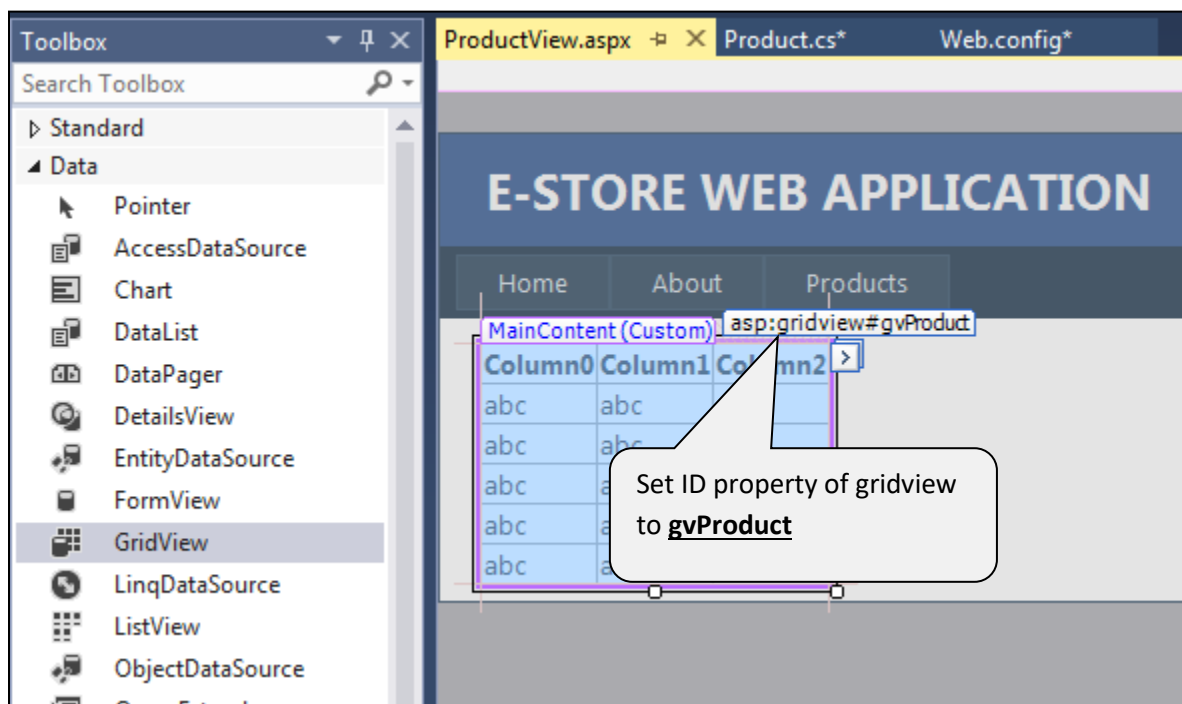
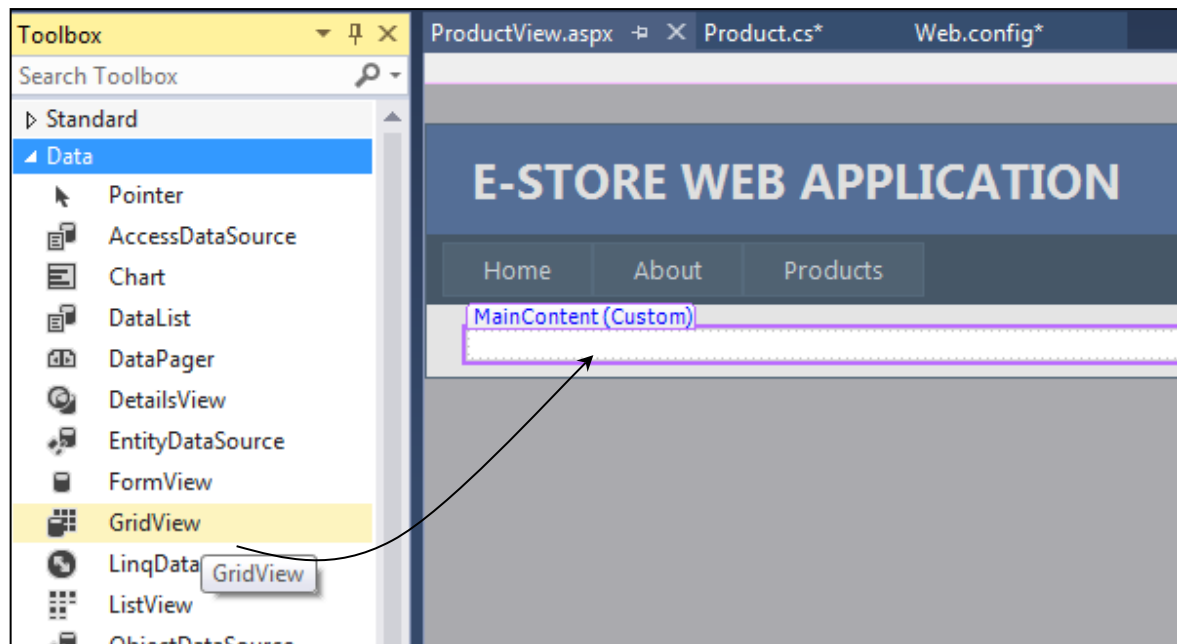
Right-click presentation layer (in this case, it's ABCHealth), choose Add ► New Item..

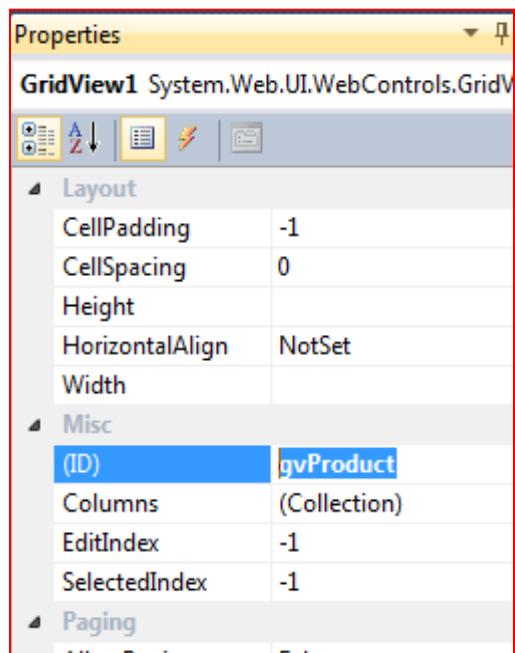




**ProductView.aspx**

2. In ProductView.aspx, add a gridview control from the toolbox (use Ctrl-Alt-x to access the toolbox).





3. At Page Load method, call the bind method to retrieve all products from database. Bind the result to gridview.

```
public partial class ProductView : System.Web.UI.Page
{
    Product aProd = new Product();
    0 references
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            bind();
        }
    }

    1 reference
    protected void bind() {
        List<Product> prodList = new List<Product>();
        prodList = aProd.getProductAll();
        gvProduct.DataSource = prodList;
        gvProduct.DataBind();
    }
}
```

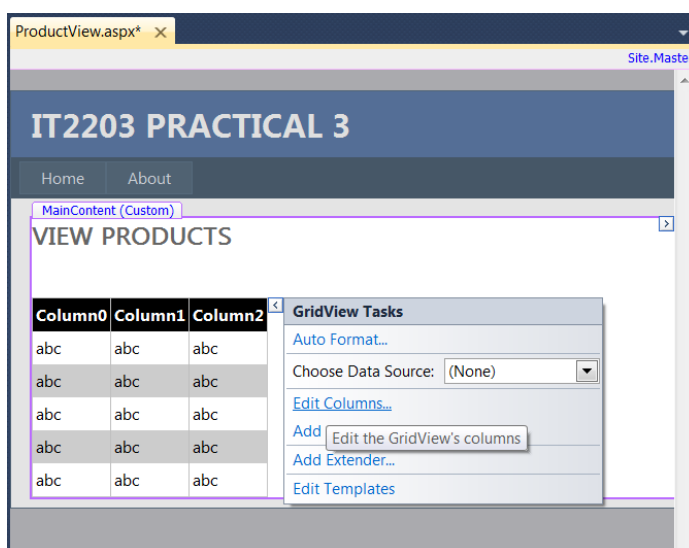


4. Set this ProductView.aspx as Start Page (Right-Click ProductView.aspx → Set as Start page). Test run. You should see the following:

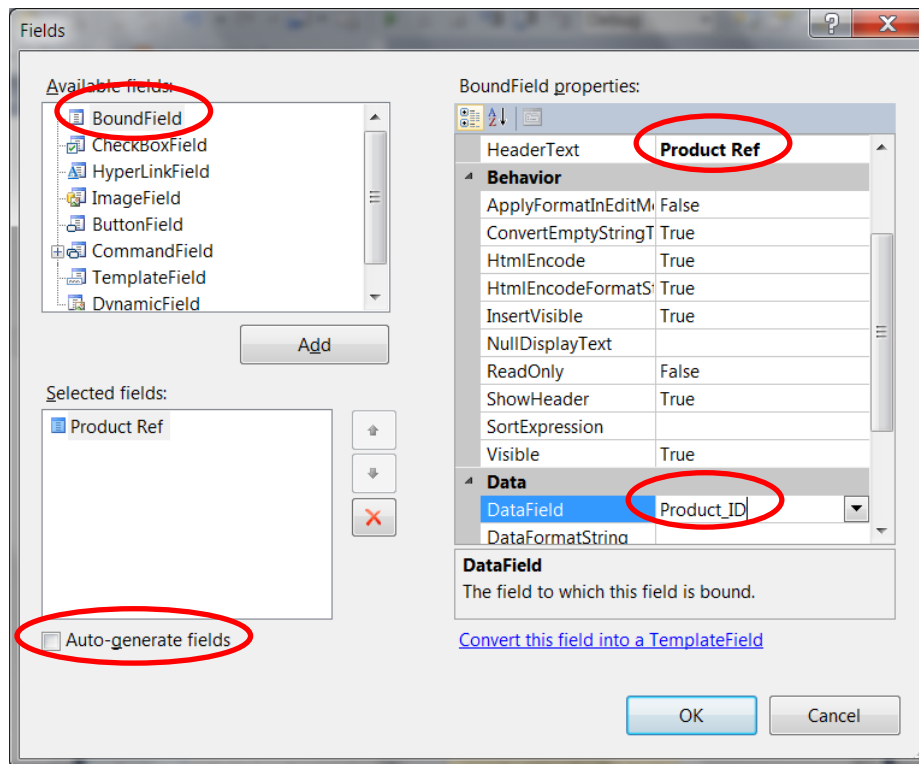
E-STORE WEB APPLICATION					
[ Log In ]					
Home About					
Product_ID	Product_Name	Product_Desc	Unit_Price	Product_Image	Stock_Level
3	LiveWell Super Calcium	Contains 100 capsules of natural source of calcium, magnesium and numerous other trace minerals.	40.00	Vitamin.png	50
1	LiveWell Vitamin C	A water and fat-soluble blend of Vitamin C blended with a citrus bioflavonoids complex	36.00	Vitamin.png	100
2	LiveWell Vitamin E	Livewell Vitamin E supply full spectrum of vitamin E isomers including alpha, beta, delta and gamma tocopherols and tocotrienols.	32.00	Vitamin.png	200

5. Next we are going to customize the gridview to display only 3 data fields (Product\_ID, Product\_Name, Unit\_Price) of data and it will have a link button for user to select the row.
6. Right-click on the gridview and choose Edit Columns.

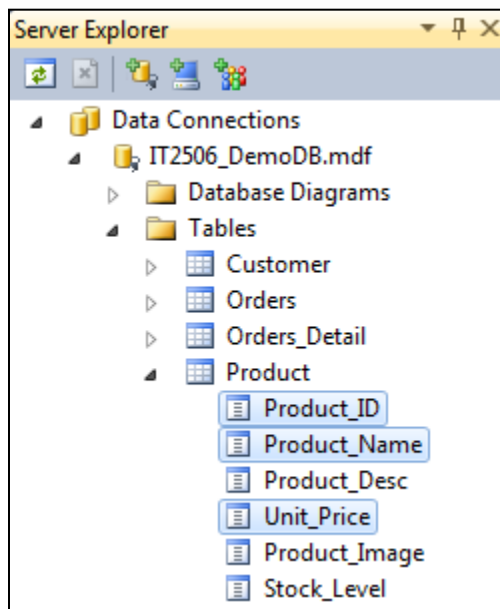
Note that when you drag the gridview to the web form, you should have given it a meaningful name. In this example, the gridview is named as gvProduct.



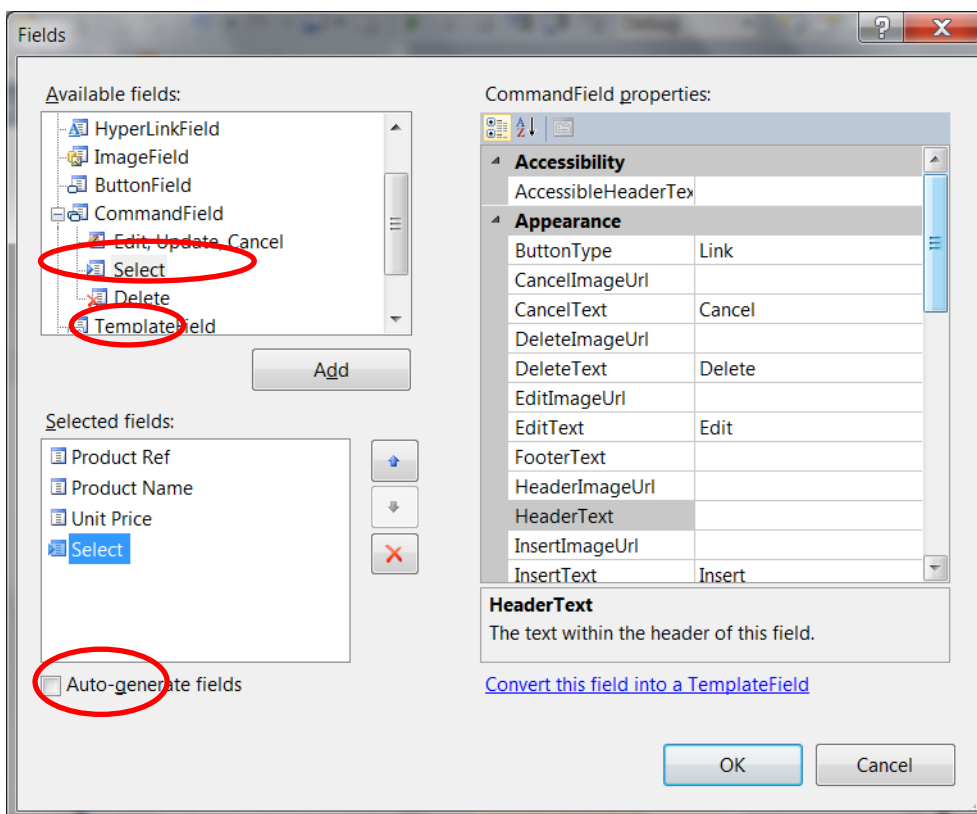
Add a Bound column for data field Product\_ID. Un-check the “Auto-generated field”.




Add two more bound fields – one for **Product\_Name** and another for **Unit\_Price**. Give both an appropriate HeaderText. Product\_Name and Unit\_Price matches the column name in the DB.

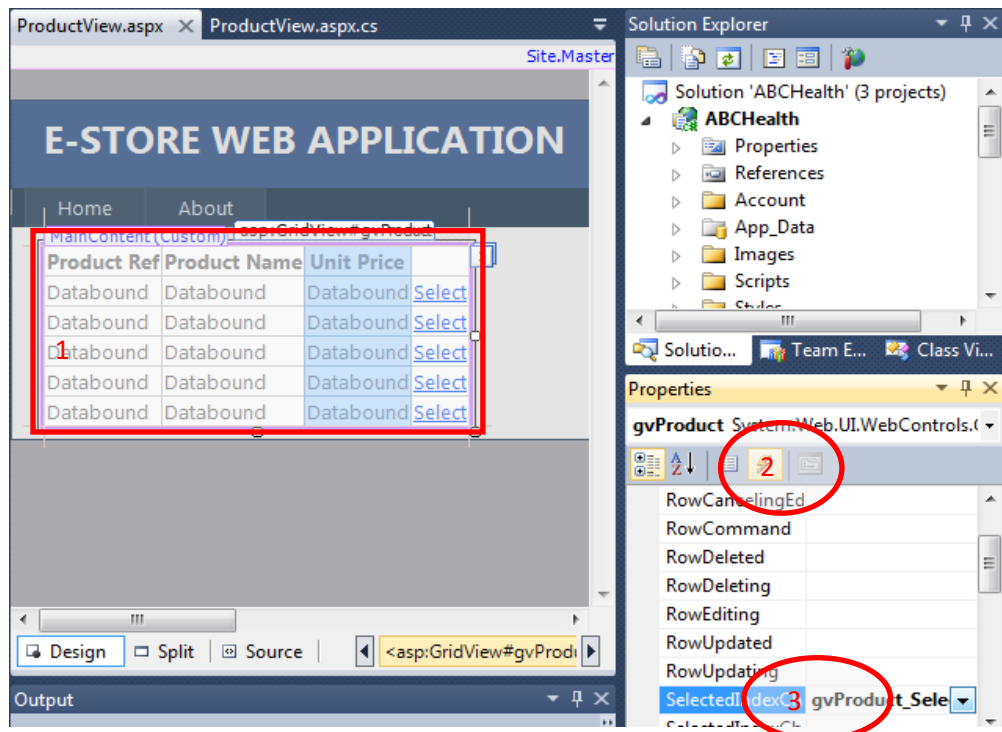


7. Add a CommandField. It will be displayed as a Link Button on screen.



Activate the event “OnSelectedIndexChanged” when users click on the “Select” link.

1. Select the gridview.
2. Select the  icon
3. Double click on the space on the dropdown listbox.



At the code-behind file, ProductView.aspx.cs, add program code inside gvProduct\_SelectedIndexChanged method.

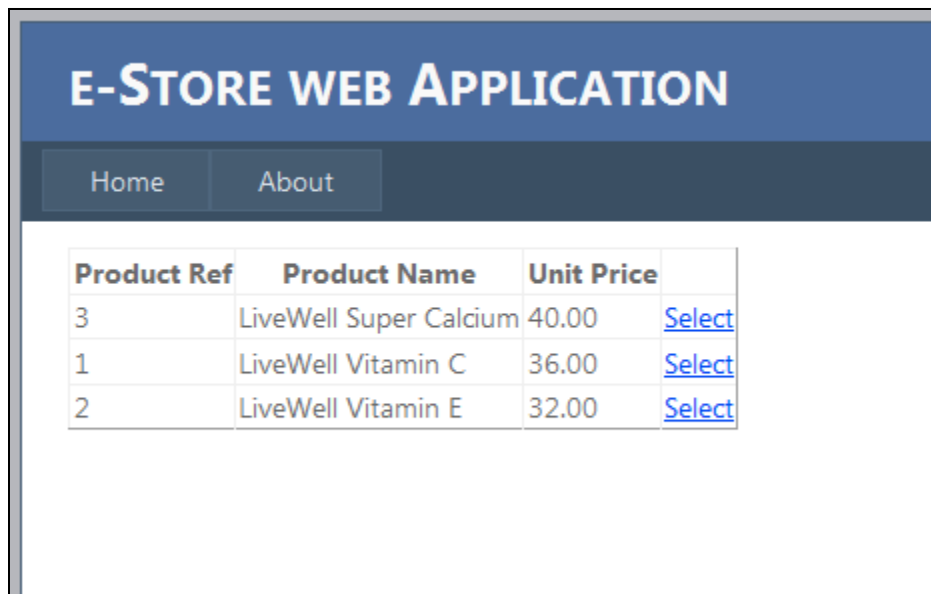
(Note: Alternatively, you may double-click on the gridview control to get to the gvProduct\_SelectedIndexChanged method)

```
protected void gvProduct_SelectedIndexChanged(object sender, EventArgs e)
{
    // Get the currently selected row.
    GridViewRow row = gvProduct.SelectedRow;

    // Get Product ID from the selected row, which is the
    // first row, i.e. index 0.
    string prodID = row.Cells[0].Text;

    // Redirect to next page, with the Product Id added to the URL,
    // e.g. ProductDetails.aspx?ProdID=1
    Response.Redirect("ProductDetails.aspx?ProdID="+prodID);
}
```

8. Test run your application. It should look like this:



When the “Select” link is clicked, it will be re-directed to ProductDetails.aspx page, which at this moment is an hardcoded page.

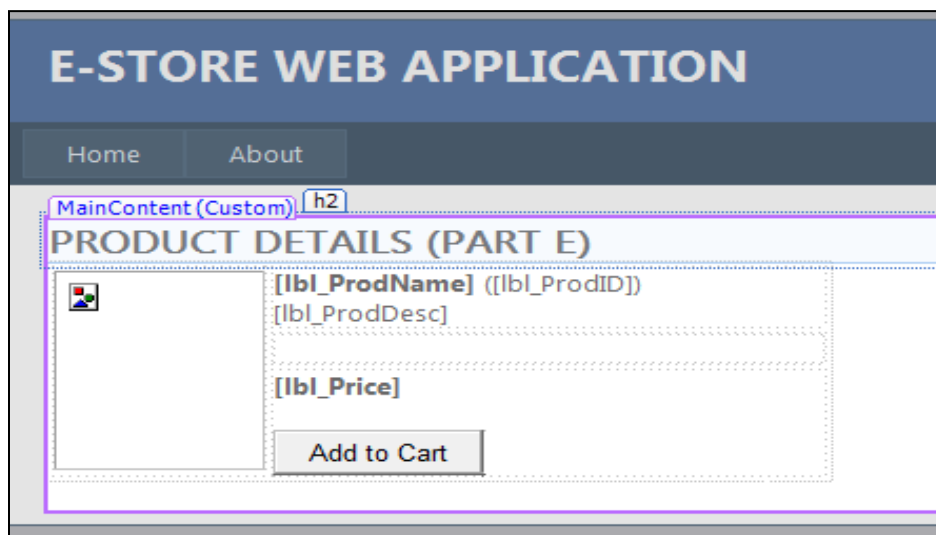
The “Select” look like a hyperlink but it’s a link button.

**ProductDetails.aspx**

This page was created in your previous practical. In this practical, we will modify ProductDetails.aspx by adding additional database codes to extract the product details from the database.

9. Re-visit ProductDetails.aspx page,

An image control is used for displaying product graphic, 3 labels are used for displaying product name, description and price.



10. Modify Page Load event with these database codes.

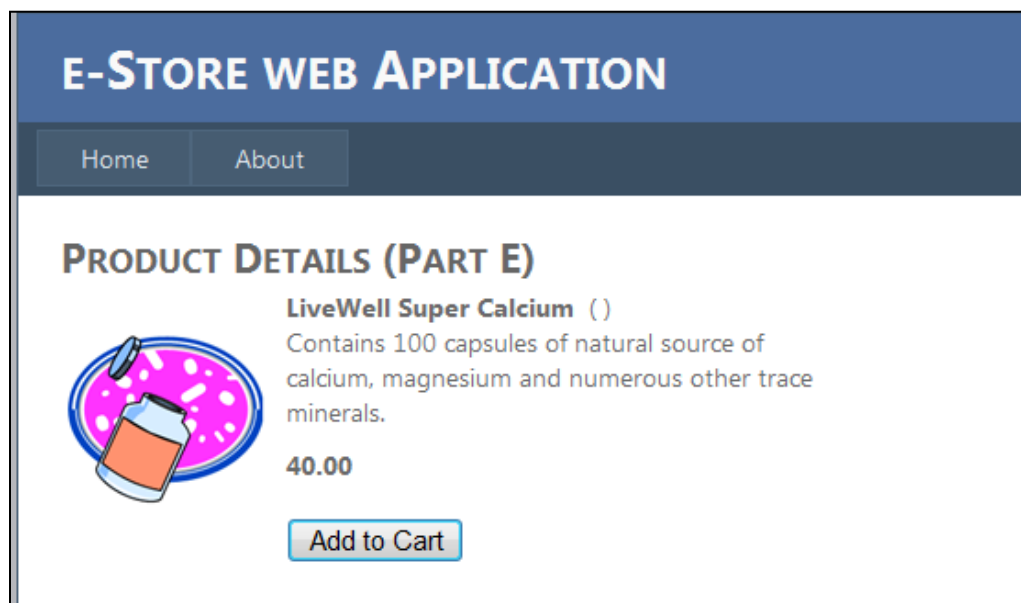
```
Product prod = null;
protected void Page_Load(object sender, EventArgs e)
{
    Product aProd = new Product();

    // Get Product ID from querystring
    string prodID = Request.QueryString["ProdID"].ToString();
    prod = aProd.getProduct(prodID);

    lbl_ProdName.Text = prod.Product_Name;
    lbl_ProdDesc.Text = prod.Product_Desc;
    lbl_Price.Text = prod.Unit_Price.ToString("c");
    img_Product.ImageUrl = "~/\\Images\\" + prod.Product_Image;
```

```
lbl_Price.Text = prod.Unit_Price.ToString();  
lbl_ProdID.Text = prodID.ToString();  
}  
  
public partial class ProductDetails : System.Web.UI.Page  
{  
    Product prod = null;  
    0 references  
    protected void Page_Load(object sender, EventArgs e)  
    {  
        Product aProd = new Product();  
  
        // Get Product ID from querystring  
        string prodID = Request.QueryString["ProdID"].ToString();  
        prod = aProd.getProduct(prodID);  
        // Display product details on web form  
        lbl_ProdName.Text = prod.Product_Name;  
        lbl_ProdDesc.Text = prod.Product_Desc;  
        lbl_Price.Text = prod.Unit_Price.ToString("c");  
        img_Product.ImageUrl = "~/\\Images\\" + prod.Product_Image;  
  
        // Store the value in invisible fields  
        lbl_Price.Text = prod.Unit_Price.ToString();  
        lbl_ProdID.Text = prodID.ToString();  
    }  
}
```

11. Test run your program:



## C# Sample Code for ProductView.aspx

```

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

0 references
public partial class ProductView : System.Web.UI.Page
{
    Product aProd = new Product();
    0 references
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            bind();
        }
    }

    1 reference
    protected void bind() {

        List<Product> prodList = new List<Product>();
        prodList = aProd.getProductAll();
        gvProduct.DataSource = prodList;
        gvProduct.DataBind();
    }

    protected void gvProduct_SelectedIndexChanged(object sender, EventArgs e)
    {
        // Get the currently selected row.
        GridViewRow row = gvProduct.SelectedRow;

        // Get Product ID from the selected row, which is the
        // first row, i.e. index 0.
        string prodID = row.Cells[0].Text;

        // Redirect to next page, with the Product Id added to the URL,
        // e.g. ProductDetails.aspx?ProdID=1
        Response.Redirect("ProductDetails.aspx?ProdID=" + prodID);
    }
}

```



**C# Sample Code for ProductDetails.aspx**

```
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)
    {
        Product aProd = new Product();

        // Get Product ID from querystring
        string prodID = Request.QueryString["ProdID"].ToString();

        // Get Product Details from database
        prod = aProd.getProduct(prodID);

        // Display product details on web form
        lbl_ProdName.Text = prod.Product_Name;
        lbl_ProdDesc.Text = prod.Product_Desc;
        lbl_Price.Text = prod.Unit_Price.ToString("c");
        img_Product.ImageUrl = "~/Images\\" + prod.Product_Image;
        lbl_Price.Text = prod.Unit_Price.ToString();
        lbl_ProdID.Text = prodID.ToString();
    }
    protected void Btn_Add_Click(object sender, EventArgs e)
    {
        string iProductID = prod.Product_ID.ToString();
        ShoppingCart.Instance.AddItem(iProductID, prod);
    }
}
```

Appendix A  
(Product.cs)

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

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

/// <summary>
/// Summary description for Product
/// </summary>
public class Product
{

    //Private string _connStr = Properties.Settings.Default.DBConnStr;

    //System.Configuration.ConnectionStringSettings _connStr;
    string _connStr = ConfigurationManager.ConnectionStrings["HealthDBContext"].ConnectionString;
    private string _prodID = null;
    private string _prodName = string.Empty;
    private string _prodDesc = ""; // this is another way to specify empty string
    private decimal _unitPrice = 0;
    private string _prodImage = "";
    private int _stockLevel = 0;

    // Default constructor
    public Product()
    {
    }

    // Constructor that take in all data required to build a Product object
    public Product(string prodID, string prodName, string prodDesc,
        decimal unitPrice, string prodImage, int stockLevel)
    {
    }
```

```
_prodID = prodID;
_prodName = prodName;
_prodDesc = prodDesc;
_unitPrice = unitPrice;
_prodImage = prodImage;
_stockLevel = stockLevel;
}

// Constructor that take in all except product ID
public Product(string prodName, string prodDesc,
    decimal unitPrice, string prodImage, int stockLevel)
    : this(null, prodName, prodDesc, unitPrice, prodImage, stockLevel)
{
}

// Constructor that take in only Product ID. The other attributes will be set to 0 or empty.
public Product(string prodID)
    : this(prodID, "", "", 0, "", 0)
{
}

// Get/Set the attributes of the Product object.
// Note the attribute name (e.g. Product_ID) is same as the actual database field name.
// This is for ease of referencing.
public string Product_ID
{
    get { return _prodID; }
    set { _prodID = value; }
}
public string Product_Name
{
    get { return _prodName; }
    set { _prodName = value; }
}
public string Product_Desc
{
    get { return _prodDesc; }
    set { _prodDesc = value; }
}
public decimal Unit_Price
```

```
{
    get { return _unitPrice; }
    set { _unitPrice = value; }
}
public string Product_Image
{
    get { return _prodImage; }
    set { _prodImage = value; }
}
public int Stock_Level
{
    get { return _stockLevel; }
    set { _stockLevel = value; }
}
```

//Below as the Class methods for some DB operations.

```
public Product getProduct(string prodID)
{
    Product prodDetail = null;

    string prod_Name, prod_Desc, Prod_Image;
    decimal unit_Price;
    int stock_Level;

    string queryStr = "SELECT * FROM Products WHERE Product_ID = @ProdID";

    SqlConnection conn = new SqlConnection(_connStr);
    SqlCommand cmd = new SqlCommand(queryStr, conn);
    cmd.Parameters.AddWithValue("@ProdID", prodID);

    conn.Open();
    SqlDataReader dr = cmd.ExecuteReader();

    if (dr.Read())
    {
        prod_Name = dr["Product_Name"].ToString();
        prod_Desc = dr["Product_Desc"].ToString();
    }
}
```

```
Prod_Image = dr["Product_Image"].ToString();
unit_Price = decimal.Parse(dr["Unit_Price"].ToString());
stock_Level = int.Parse(dr["Stock_Level"].ToString());

prodDetail = new Product(prodID, prod_Name, prod_Desc, unit_Price, Prod_Image, stock_Level);
}
else
{
    prodDetail = null;
}

conn.Close();
dr.Close();
dr.Dispose();

return prodDetail;
}

public List<Product> getProductAll()
{
    List<Product> prodList = new List<Product>();

    string prod_Name, prod_Desc, Prod_Image, prod_ID;
    decimal unit_Price;
    int stock_Level;

    string queryStr = "SELECT * FROM Products Order By Product_Name";

    SqlConnection conn = new SqlConnection(_connStr);
    SqlCommand cmd = new SqlCommand(queryStr, conn);

    conn.Open();
    SqlDataReader dr = cmd.ExecuteReader();

    while (dr.Read())
    {
        prod_ID = dr["Product_ID"].ToString();
        prod_Name = dr["Product_Name"].ToString();
        prod_Desc = dr["Product_Desc"].ToString();
        Prod_Image = dr["Product_Image"].ToString();
    }
}
```

```
        unit_Price = decimal.Parse(dr["Unit_Price"].ToString());
        stock_Level = int.Parse(dr["Stock_Level"].ToString());
        Product a = new Product(prod_ID, prod_Name, prod_Desc, unit_Price, Prod_Image, stock_Level);
        prodList.Add(a);
    }

    conn.Close();
    dr.Close();
    dr.Dispose();

    return prodList;
}

//public int ProductInsert()
//{

//} //end Insert

//public int ProductDelete(decimal ID)
//{

//} //end Delete

//public int ProductUpdate(string pId, string pName, decimal pUnitPrice)
//{

//} //end Update

}
```