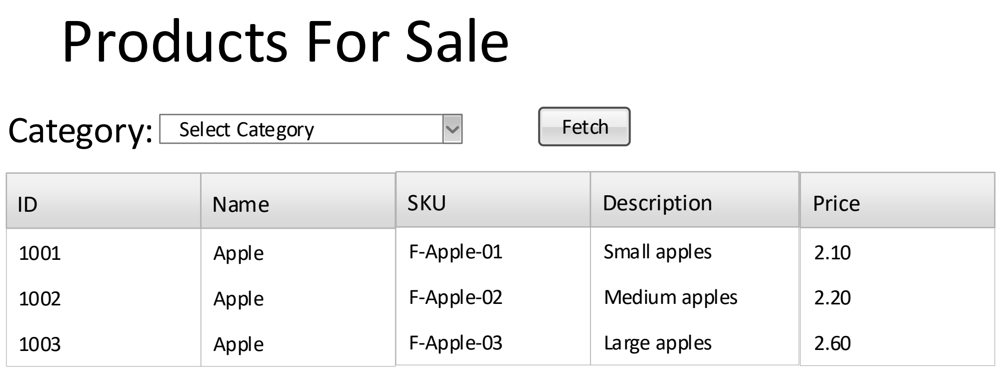
4.2.0 – ShoppingCart.aspx

Introduction

So far we have coded the classes that will return some data from the database. In this lesson, we want to display that data on a web form that looks like the prototype below:



Products For Sale

Category: Select Category ⌄ Fetch

ID	Name	SKU	Description	Price
1001	Apple	F-Apple-01	Small apples	2.10
1002	Apple	F-Apple-02	Medium apples	2.20
1003	Apple	F-Apple-03	Large apples	2.60

Figure 1: Prototype Web Form

The normal practice for adding web forms to a web site project is to put all related code in separate folders; we do not move the automatically generated web forms. Additionally, we want all the web forms to have the same look, thus we will be needing any new web form to use the `Site.Master` template system.

Setup

First, we need to create 2 new folders in the **eStoreWeb** project: **Sales** and **Orders**.

Adding Folders

1. Right-click the **eStoreWeb** project and select **Add**, then **New Folder**:

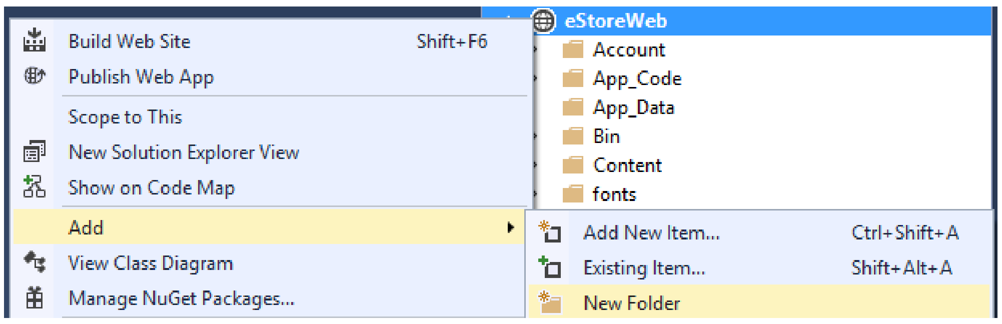
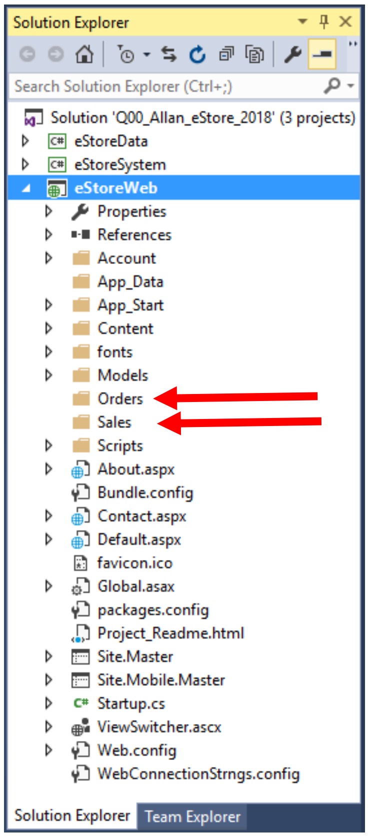


Figure 2: Adding a Folder to a Web Site Project

2. Rename it from **New Folder** to **`Sales`**.
3. Repeat step 1 and rename the new folder **`Orders`**.
4. When completed, your web project should look like:



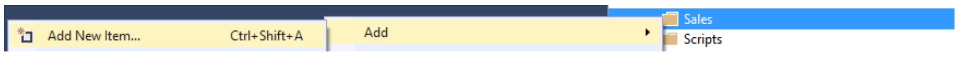
```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Q00_Allan_eStore_2018' (3 projects)
  eStoreData
  eStoreSystem
  eStoreWeb
    Properties
    References
    Account
    App_Data
    App_Start
    Content
    fonts
    Models
    Orders
    Sales
    Scripts
    About.aspx
    Bundle.config
    Contact.aspx
    Default.aspx
    favicon.ico
    Global.asax
    packages.config
    Project_Readme.html
    Site.Master
    Site.Mobile.Master
    Startup.cs
    ViewSwitcher.ascx
    Web.config
    WebConnectionStrngs.config
Solution Explorer   Team Explorer
```

Figure 3: New Folders Added to eStoreWeb

Next we need to add web forms to each of these folders:

Adding Web Forms

1. Right-click the **`Sales`** folder you just created and select **Add** then **Add New Item…**:



2. On the Add New Item wizard, select **Web Form**

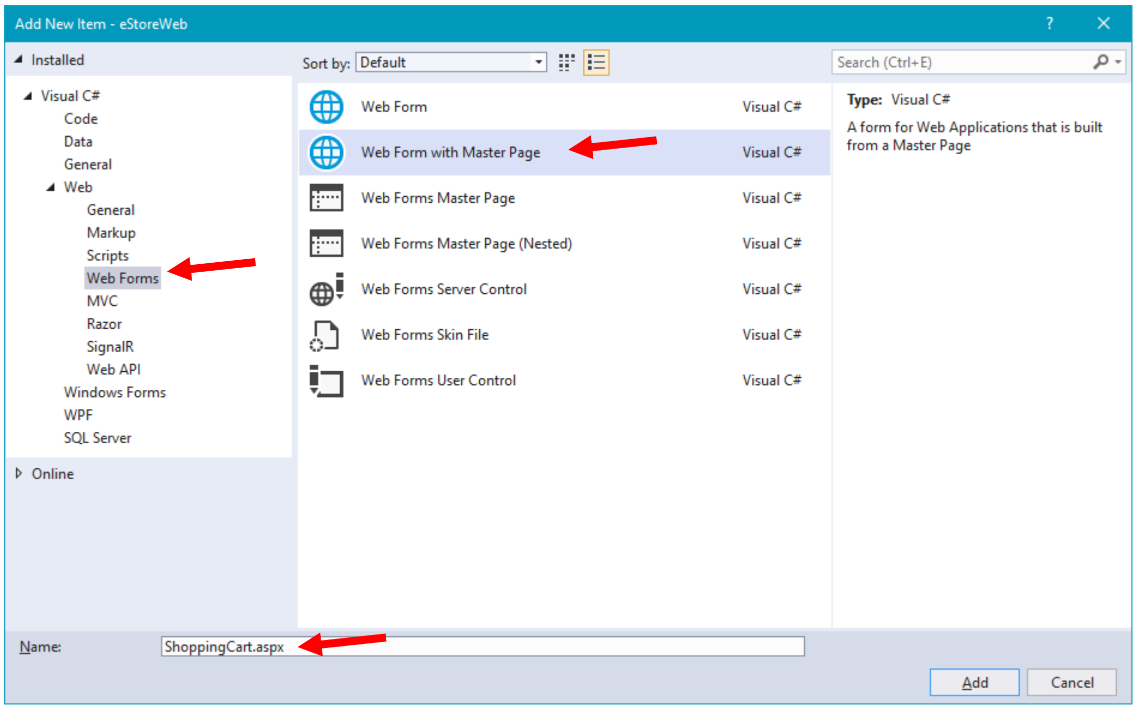
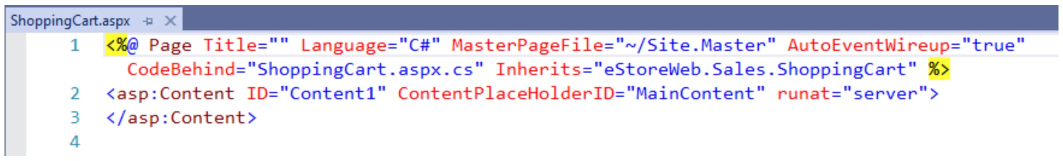

Add New Item - eStoreWeb
Installed
Visual C#
Code
Data
General
Web
General
Markup
Scripts
Web Forms
MVC
Razor
SignalR
Web API
Windows Forms
WPF
SQL Server
Online
Sort by: Default
Search (Ctrl+E)
Web Form Visual C#
Web Form with Master Page Visual C#
Web Forms Master Page Visual C#
Web Forms Master Page (Nested) Visual C#
Web Forms Server Control Visual C#
Web Forms Skin File Visual C#
Web Forms User Control Visual C#
Type: Visual C#
A form for Web Applications that is built from a Master Page
Name: ShoppingCart.aspx
Add
Cancel

Figure 4: Add Web Form

3. For the name, use **ShoppingCart.aspx**
4. Press **Add** and select **Site.Master** from the list, then press **OK**.
5. Your web form code should look like:



ShoppingCart.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true"
  CodeBehind="ShoppingCart.aspx.cs" Inherits="eStoreWeb.Sales.ShoppingCart" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">
</asp:Content>
```

6. Repeat steps 2 to 5 for the **Orders** folder. The new web form you will be creating will be named **PurchaseOrders.aspx**.
7. You should now have the following:

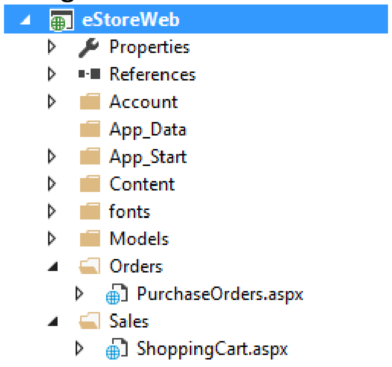


Figure 5: Web Forms Added

This lesson will focus on the web form **`ShoppingCart.aspx`**; in a later lesson we will code **`PurchaseOrders.aspx`**.

Update Navigation Menu

Now that we have added two new web forms to **eStoreWeb**, we need to update the menu in the **`Site.master`** file.

1. Open the **`Site.master`** file and scroll to find the menu which should look like:

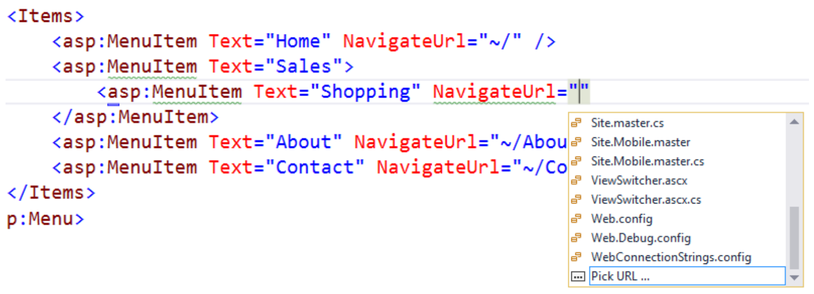
```
<asp:Menu ID="NavMenu" runat="server"
    BackColor="#222222"
    BorderColor="#ff0000"
    StaticMenuItemStyle-ForeColor="#9d9d9d"
    StaticMenuItemStyle-VerticalPadding="15px"
    StaticMenuItemStyle-HorizontalPadding="10px"
    RenderingMode="List"
    Orientation="Horizontal"
    DynamicMenuItemStyle-ForeColor="#9d9d9d"
    DynamicMenuStyle-BackColor="#222222">
    <Items>
        <asp:MenuItem Text="Home" NavigateUrl="~/" />
        <asp:MenuItem Text="About" NavigateUrl="~/About.aspx" />
        <asp:MenuItem Text="Contact" NavigateUrl="~/Contact.aspx" />
    </Items>
</asp:Menu>
```

   Each of the `asp:MenuItem` is a parent menu; there are no sub-menus. We want to create 2 new parent menus, with each having a sub menu.

2. Create a blank line below the `Text="Home"` menu item.
3. Add a new parent menu item as shown below:

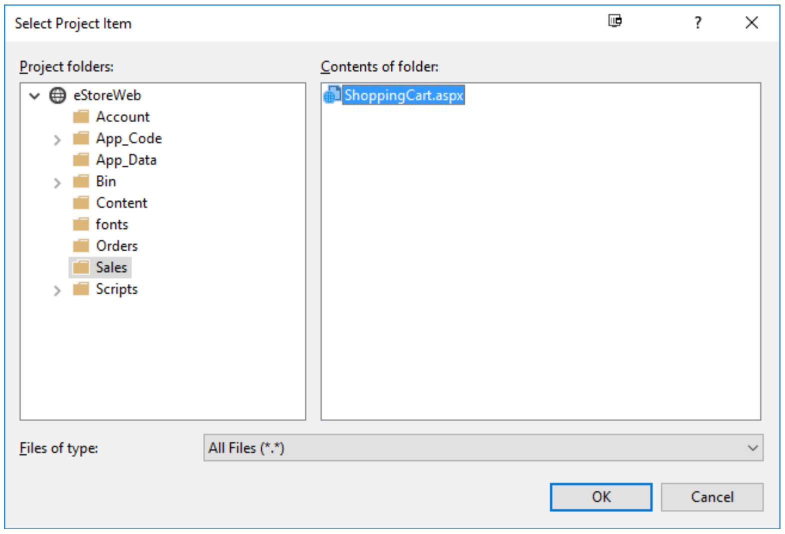
```
<Items>
    <asp:MenuItem Text="Home" NavigateUrl="~/" />
    <asp:MenuItem Text="Sales">

    </asp:MenuItem>
    <asp:MenuItem Text="About" NavigateUrl="~/About.aspx" />
    <asp:MenuItem Text="Contact" NavigateUrl="~/Contact.aspx" />
</Items>
```

4. In the newly created parent menu item, add the following:



```
<Items>
    <asp:MenuItem Text="Home" NavigateUrl="~/" />
    <asp:MenuItem Text="Sales">
        <asp:MenuItem Text="Shopping" NavigateUrl=""
    </asp:MenuItem>
    <asp:MenuItem Text="About" NavigateUrl="~/Abou
    <asp:MenuItem Text="Contact" NavigateUrl="~/Co
</Items>
p:Menu>
```

5. We could type in the location of the **ShoppingCart.aspx** web form, but to avoid mistakes, we will double click on the **Pick URL…** to correctly add the navigation URL (the location should be like that shown below):



6. Press **OK** and you should get:

```
<asp:MenuItem Text="Sales">
    <asp:MenuItem Text="Shopping" NavigateUrl="~/Sales/ShoppingCart.aspx"
</asp:MenuItem>
```

7. Now just close the child menu item by adding `/>` at the end of that line.
8. Repeat the above steps to create an **Orders** parent menu, with a child menu for **`PurchaseOrders.aspx`**. When completed, your menu should look like:

```
<Items>
    <asp:MenuItem Text="Home" NavigateUrl="~/" />
    <asp:MenuItem Text="Sales">
        <asp:MenuItem Text="Shopping" NavigateUrl="~/Sales/ShoppingCart.aspx" />
    </asp:MenuItem>
    <asp:MenuItem Text="Orders">
        <asp:MenuItem Text="Purchasing" NavigateUrl="~/Orders/PurchaseOrders.aspx" />
    </asp:MenuItem>
    <asp:MenuItem Text="About" NavigateUrl="~/About.aspx" />
    <asp:MenuItem Text="Contact" NavigateUrl="~/Contact.aspx" />
</Items>
```

9. The menu should now be tested before going any further. You should be able to navigate to these new web forms; as they do not have any code, a blank page will be displayed.

Code the Web Form

For the **`ShoppingCart.aspx`** web form we will need to add the following web controls so that it matches the prototype (see Figure 1):

- **`Label`** controls and HTML tags: to display text on the web form
- **`DropDownList`**: this will display the categories
- **`Button`**: to get the products for the selected category
- **`GridView`**: this will display the products for the selected category

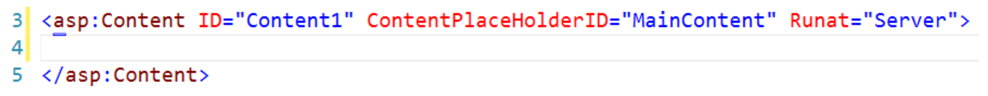
Additionally, we will be using 2 hidden web controls, **`ObjectDataSource`** controls, to interact between the web form and the BLL controller classes.

Finally, to add style to the web form, we will be using some **bootstrap** styling.

REMEMBER: All the code we write must be between lines 3 and 4; NO code allowed after the `</asp:Content>` line!

Steps

1. Create a blank line after line 3:



```
3  <asp:Content ID="Content1" ContentPlaceHolderID="MainContent" Runat="Server">
4  
5  </asp:Content>
```

2. Add the following code to the new blank line:

```
<div class="jumbotron">
    <h1>Products For Sale</h1>
    <p>This web form will load a DropDownList of Categories which the Customer can select from.
        Once a Category is selected, a list of Products for the Category will be displayed.</p>
</div>
```

You are adding a big banner, `class="jumbotron"`, then a heading, `<h1>`, and finally some text.

3. Next we need a row just for the `DropDownList` and `Button`. Add the following below the `</div>`line:

```
<div class="row">
    <asp:Label ID="CategoryLabel" runat="server" Text="Category: " />
    &nbsp;&nbsp;
    <asp:DropDownList ID="CategoryListDDL" runat="server"></asp:DropDownList>
    &nbsp;&nbsp;
    <asp:Button ID="FetchButton" runat="server" Text="Fetch" />
</div>
```

   Here we have the starting code so that we can get a list of the categories to display in a dropdown list.
4. Below the code added in step 3, add the following code:

```
<div class="row">
    <asp:GridView ID="ProductListGV" runat="server"></asp:GridView>
</div>
```

   This will hold all the products from a category selected from the `asp:DropDownList ID="CategoryListDDL"` control.
5. Next, we are going to add the 2 hidden data controls that will interact with the database (added below the div that contains the GridView):

```
<!-- ObjectDataSource Control(s) -->
<asp:ObjectDataSource ID="CategoryListODS" runat="server"></asp:ObjectDataSource>
<asp:ObjectDataSource ID="ProductListODS" runat="server"></asp:ObjectDataSource>
```

   These controls do not need to be in a row div as they are not displayed when the web form is displayed in a web browser.

The code entered above does not do anything yet. We need to add the code necessary to make this web form work.

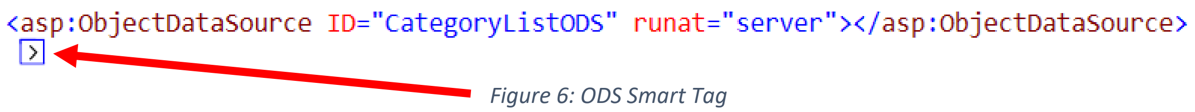
ObjectDataSource Controls

The ObjectDataSource (ODS) controls will be the web controls that connect to the classes in the **eStoreSystem.BLL** namespace (these controller classes were coded in the previous lesson).

CategoryListODS

This ODS control will need the `GetAllCategories()` method of the **`CategoryController.cs`** to retrieve all the Categories in the database. Later, we will bind this data to the **`CategoryListDDL`** for displaying all the Categories.

When the web form is opened in Source view, we can click on the code line and see the smart tag for the control:



```
<asp:ObjectDataSource ID="CategoryListODS" runat="server"></asp:ObjectDataSource>
```

Figure 6: ODS Smart Tag

Clicking the smart tag brings up the options of what is available for this control:

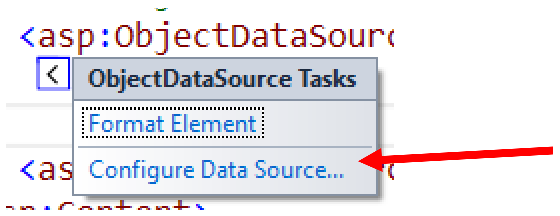
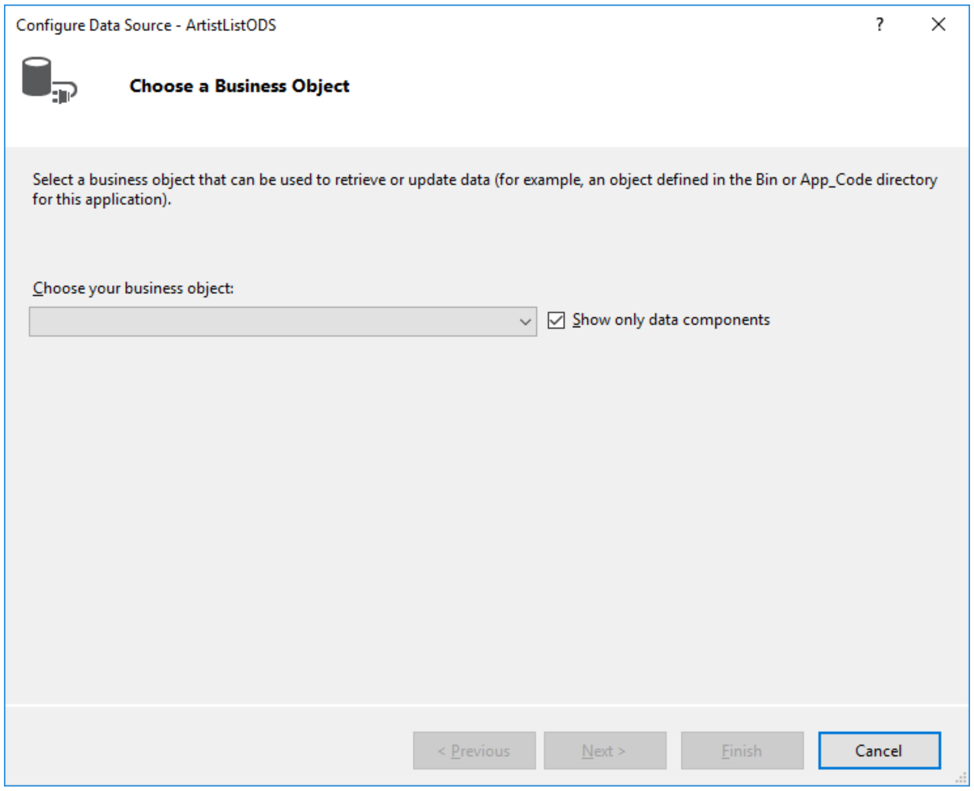


Figure 7: ODS Smart Tag Options

We want to configure the data source:



Configure Data Source - ArtistListODS

Choose a Business Object

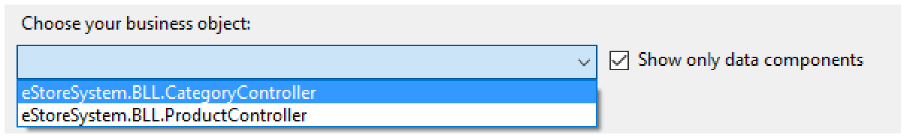
Select a business object that can be used to retrieve or update data (for example, an object defined in the Bin or App_Code directory for this application).

Choose your business object:

☑ Show only data components

Figure 8: Configure Data Source (1)

Use the drop down to select the correct business object. For this control, it will be the `CategoryController`. The **`CategoryController.cs`** class has the annotation of `[DataObject]` which makes is available in the drop down:



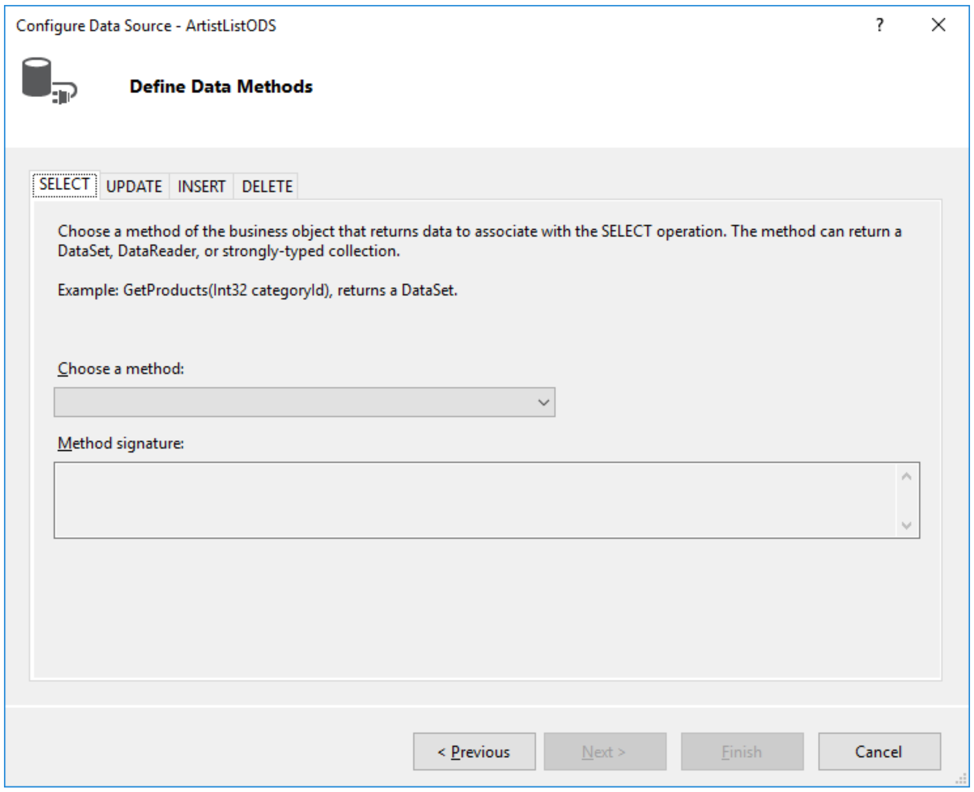
Choose your business object:

☑ Show only data components

eStoreSystem.BLL.CategoryController
eStoreSystem.BLL.ProductController

Figure 9: Select Business Object

After the business object is selected, press next to get the next screen of the wizard:


Configure Data Source - ArtistListODS

Define Data Methods

SELECT | UPDATE | INSERT | DELETE

Choose a method of the business object that returns data to associate with the SELECT operation. The method can return a DataSet, DataReader, or strongly-typed collection.

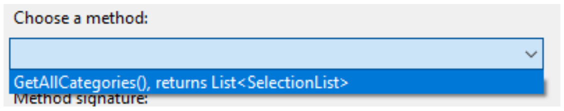
Example: GetProducts(Int32 categoryId), returns a DataSet.

Choose a method:

Method signature:

Figure 10: Configure Data Source (2)

Use the drop down to select the method. In our code, so far, we only have one method:


Choose a method:

GetAllCategories(), returns List<SelectionList>

*Figure 11: Select the **Select** Method*

Press Finish to close the wizard. Visual Studio will write out the code shown below (your instructor has formatted the code so that is easier to read and understand):

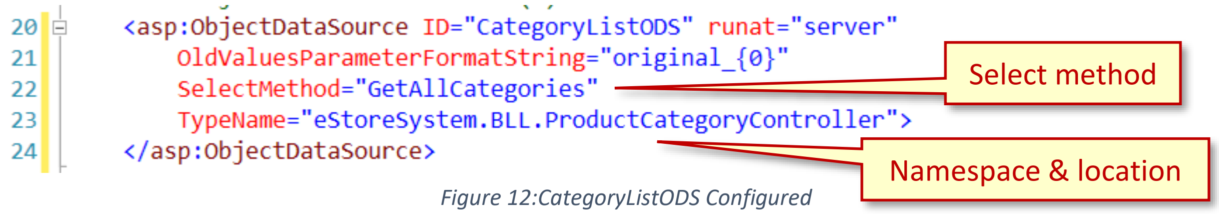
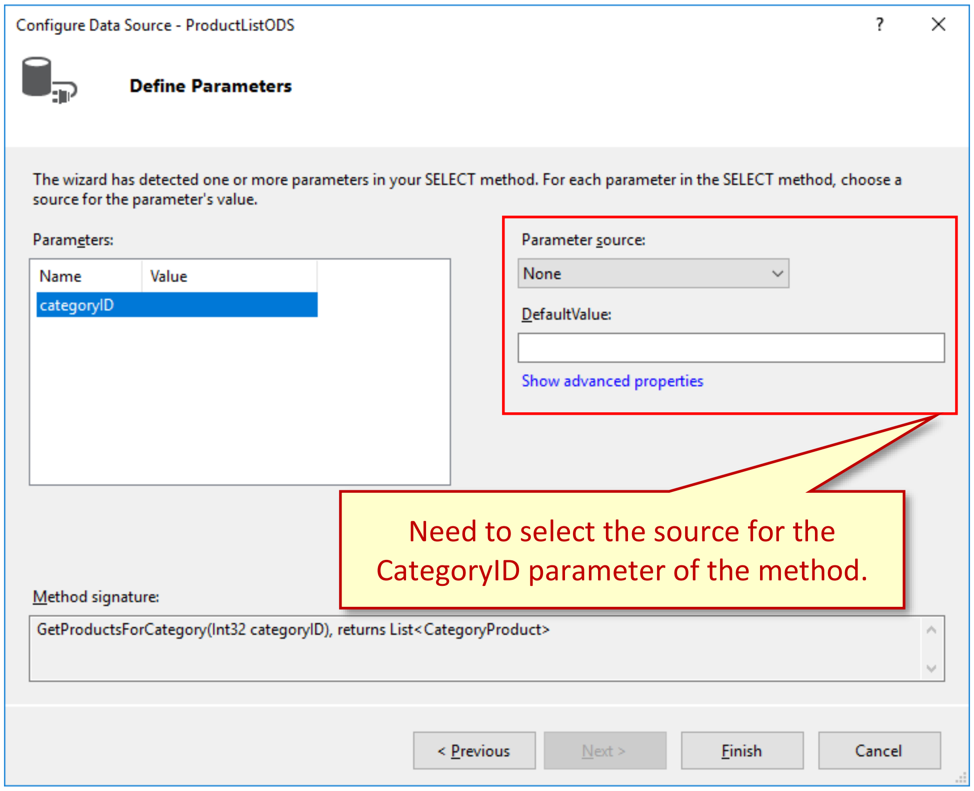

```
<asp:ObjectDataSource ID="CategoryListODS" runat="server"
    OldValuesParameterFormatString="original_{0}"
    SelectMethod="GetAllCategories"
    TypeName="eStoreSystem.BLL.ProductCategoryController">
</asp:ObjectDataSource>
```

Figure 12:CategoryListODS Configured

ProductListODS

The ProductListODS control will need the `GetProductsForCategory(categoryID)` method of the **`ProductController.cs`** to retrieve all the Products for the Category that has the Primary Key value of the `cataegoryID` parameter.

Click the control to get the smart tag, and select Configure Data Source. Remember that this ODS control needs the `ProductController` and the method that we coded. Once the method has been selected you must press Next:


Configure Data Source - ProductListODS

Define Parameters

The wizard has detected one or more parameters in your SELECT method. For each parameter in the SELECT method, choose a source for the parameter's value.

Parameters:

Name	Value
categoryID	

Parameter source: None

DefaultValue:

Need to select the source for the CategoryID parameter of the method.

Method signature:

```
GetProductsForCategory(Int32 categoryID), returns List<CategoryProduct>
```

Figure 13: Configure Data Source Needing a Paramter

In our example, the Customer will select the Artist from the `CategoryListDDL` `DropDownList` control. Therefore, we need to select the Control from the options:

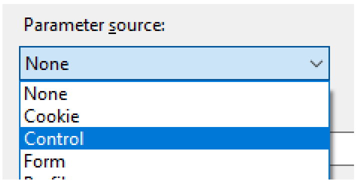
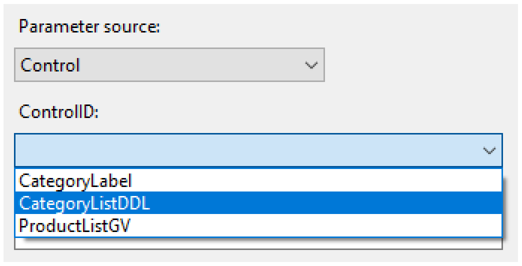


Figure 14: Select Parameter Source - Control

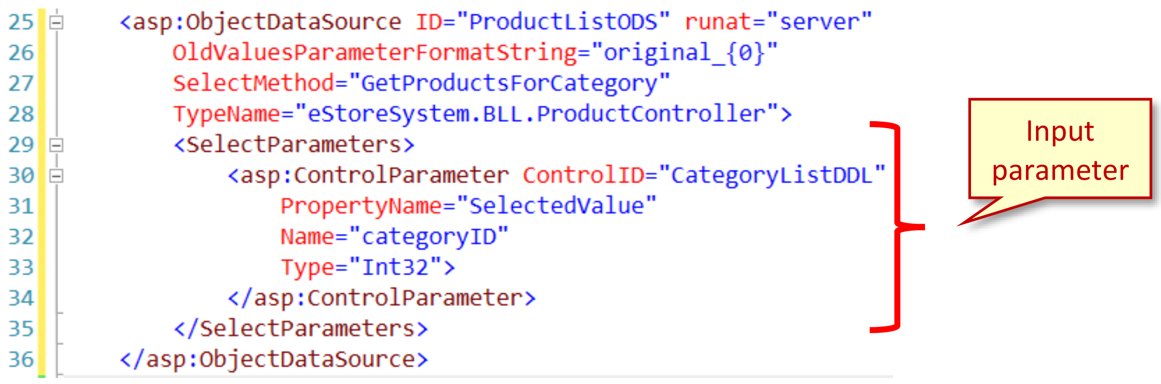
Next, we need to identify which control on the web form that we will be using, which is the `CategoryListDDL` control:



Parameter source:
Control
ControlID:
CategoryLabel
CategoryListDDL
ProductListGV

Figure 15: Select the Control ID

Press Finish to exit the wizard. The code that Visual Studio writes is shown below (formatted for reading):



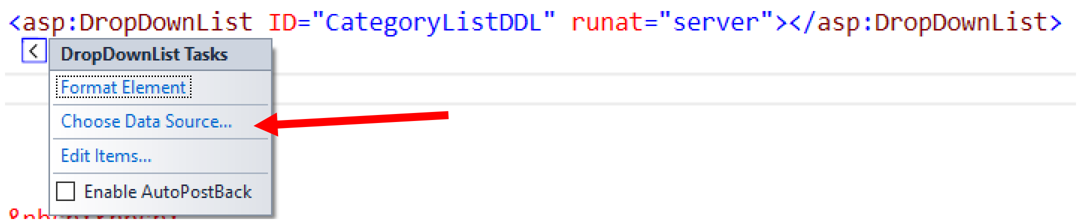
```
    <asp:ObjectDataSource ID="ProductListODS" runat="server"
        OldValuesParameterFormatString="original_{0}"
        SelectMethod="GetProductsForCategory"
        TypeName="eStoreSystem.BLL.ProductController">
        <SelectParameters>
            <asp:ControlParameter ControlID="CategoryListDDL"
                PropertyName="SelectedValue"
                Name="categoryID"
                Type="Int32">
            </asp:ControlParameter>
        </SelectParameters>
    </asp:ObjectDataSource>
```

Figure 16: ProductListODS Configured

DropDownList

Now that we should be able to get data from the database, using the ODS controls, we can now code the `CategoryListDDL` control to display the Categories.

A DropDownList control also has a smart tag. When clicked you should see the following options:



```
<asp:DropDownList ID="CategoryListDDL" runat="server"></asp:DropDownList>
```

Figure 17: CategoryDDL Smart Tag Options

Here we want to select the **Choose Data Source…** option. When we click on that option, we want to select the **`CategoryListODS`** control; the **`CategoryListODS`** control gets all the Categories from the database:

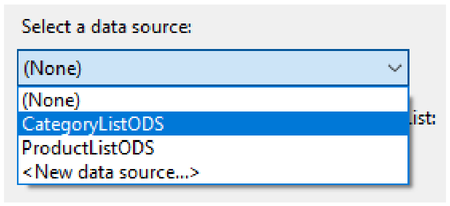


Figure 18: Select Data Source - ArtistListODS

Once selected, you should see the following:

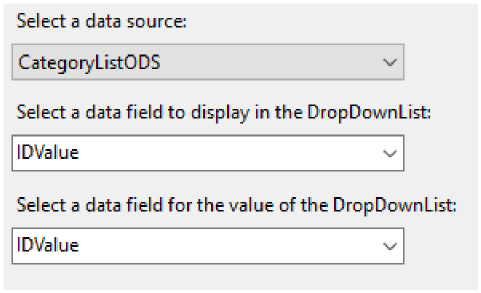

Select a data source:
CategoryListODS
Select a data field to display in the DropDownList:
IDValue
Select a data field for the value of the DropDownList:
IDValue

Figure 19: Data Source Option Fields

We can leave the bottom (data field for the value) as ID as this field represents the Primary Key. We do need to change the top (data field to display) to Name:

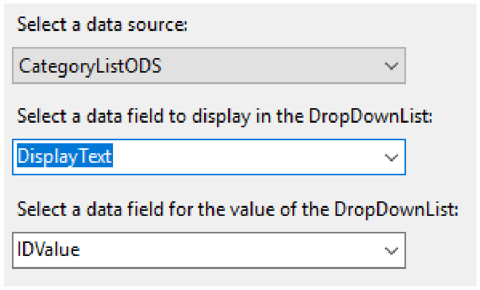
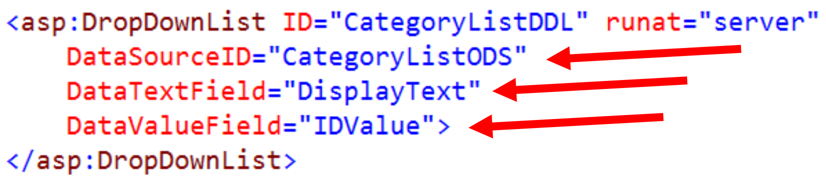

Select a data source:
CategoryListODS
Select a data field to display in the DropDownList:
DisplayText
Select a data field for the value of the DropDownList:
IDValue

Figure 20: Data Source Options Selected

Press OK to close the wizard to see the following code (formatted for ease of reading):



```
<asp:DropDownList ID="CategoryListDDL" runat="server"
    DataSourceID="CategoryListODS"
    DataTextField="DisplayText"
    DataValueField="IDValue">
</asp:DropDownList>
```

Figure 21: CategoryListDDL with Data Source

We want to be able to have an option in our drop down list as “Select Category”. To do this change the code of the `CategoryListDDL` to be:

Listing 1: CategoryListDDL Completed Code

```
<asp:DropDownList ID="CategoryListDDL" runat="server"
```

```
    DataSourceID="CategoryListODS"
    DataTextField="DisplayText"
    DataValueField="IDValue"
    AppendDataBoundItems="true">
    <asp:ListItem Value="0">Select Category</asp:ListItem>
</asp:DropDownList>
```

There is never going to be a Primary Key with the value of 0, therefore we have `Value="0"`. The `Select Category` is the text that will display in the drop down list, and will be the default when the web form runs.

GridView

A GridView control will display the data that it gets as a table in the browser. This is a very useful control and can be customized in many ways. It too, has a smart tag. When clicked it has the following options:

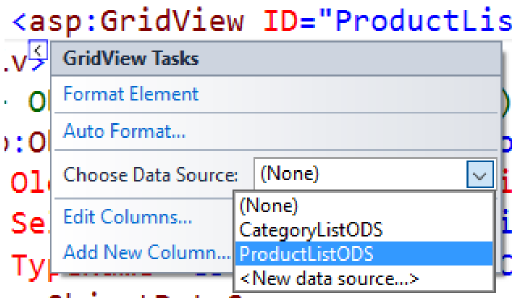
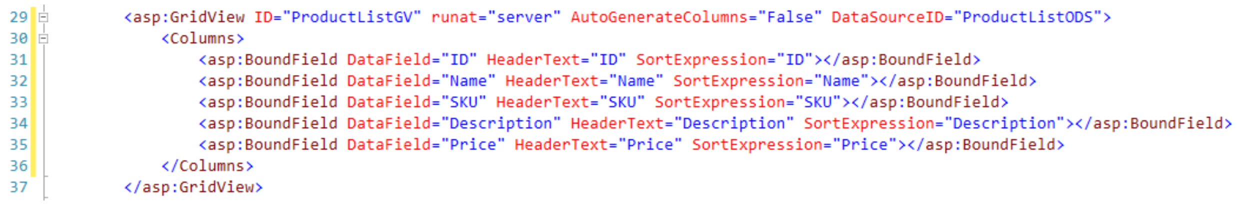


Figure 22: GridView Smart Tag Options

The simplest way to get the data on to the GridView is to **Choose Data Source:**. In doing so, select the correct data source control, **`ProductListODS`** in this example, and Visual Studio will write out the code for you:



```
<asp:GridView ID="ProductListGV" runat="server" AutoGenerateColumns="False" DataSourceID="ProductListODS">
    <Columns>
        <asp:BoundField DataField="ID" HeaderText="ID" SortExpression="ID"></asp:BoundField>
        <asp:BoundField DataField="Name" HeaderText="Name" SortExpression="Name"></asp:BoundField>
        <asp:BoundField DataField="SKU" HeaderText="SKU" SortExpression="SKU"></asp:BoundField>
        <asp:BoundField DataField="Description" HeaderText="Description" SortExpression="Description"></asp:BoundField>
        <asp:BoundField DataField="Price" HeaderText="Price" SortExpression="Price"></asp:BoundField>
    </Columns>
</asp:GridView>
```

Figure 23: ProductListGV with Data Source

There are a few things to know about the code that Visual Studio wrote for you:

- `AutoGenerateColumns="False"`: This attribute-value pair overrides the default display of all the data coming in to the control. In our example, we are needing all the data columns coming in as data, but we could change it as we are using this attribute-value pair.
- `<Columns>:` Because we have the auto generate columns turned off, we **must** tell the GridView control which columns to use
- `<asp:BoundField:` This means that the data for the column of data is data bound.

- `HeaderText:` The text that will be displayed at the top of the column; we can change this to be anything.
- `DataField:` This is the column in the record coming in as data; it **must** match the definition of the data set being used, in this case **CategoryProduct**.
- `SortExpression:` Sets up the default sorting order for each column.

The `GridView` is almost complete. For all our `GridView` controls we will be adding a template that will be displayed only if there is no data in the data set being sent to the `GridView`. Modify the `ProductListGV` to be:

Listing 2: ProductListGV Code Complete

```
<asp:GridView ID="ProductListGV" runat="server" AutoGenerateColumns="False"
DataSourceID="ProductListODS">
    <Columns>
        <asp:BoundField DataField="ID" HeaderText="ID"
            SortExpression="ID"></asp:BoundField>
        <asp:BoundField DataField="Name" HeaderText="Name"
            SortExpression="Name"></asp:BoundField>
        <asp:BoundField DataField="SKU" HeaderText="SKU"
            SortExpression="SKU"></asp:BoundField>
        <asp:BoundField DataField="Description" HeaderText="Description"
            SortExpression="Description"></asp:BoundField>
        <asp:BoundField DataField="Price" HeaderText="Price"
            SortExpression="Price"></asp:BoundField>
    </Columns>
    <EmptyDataTemplate>
        No Products for the selected Category!
    </EmptyDataTemplate>
</asp:GridView>
```

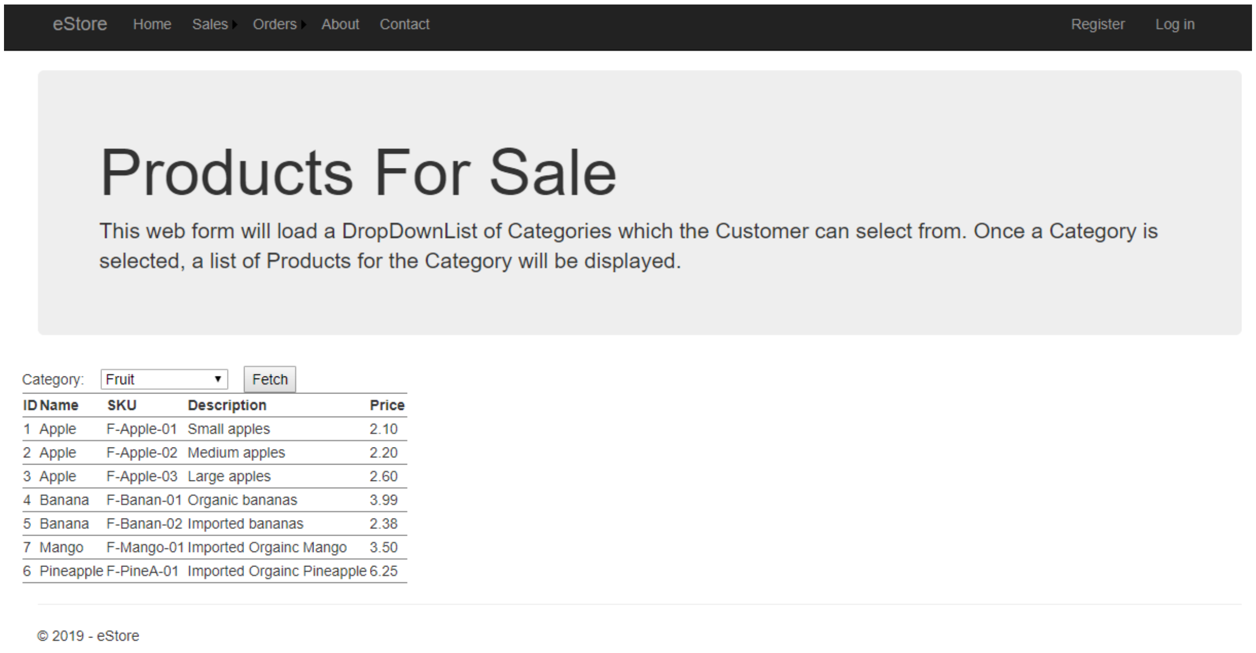
Customize GridView (Optional)

One thing you can do to customize the look of a GridView is to turn off the border, and only use horizontal grid lines. Modify the GridView code to be:

```
<asp:GridView ID="ProductListGV" runat="server" AutoGenerateColumns="False"
DataSourceID="ProductListODS" BorderStyle="None" GridLines="Horizontal">
```

TEST!

Run the web form in the browser. Select a Category, press the Fetch button, and you should see results like:



Products For Sale

This web form will load a DropDownList of Categories which the Customer can select from. Once a Category is selected, a list of Products for the Category will be displayed.

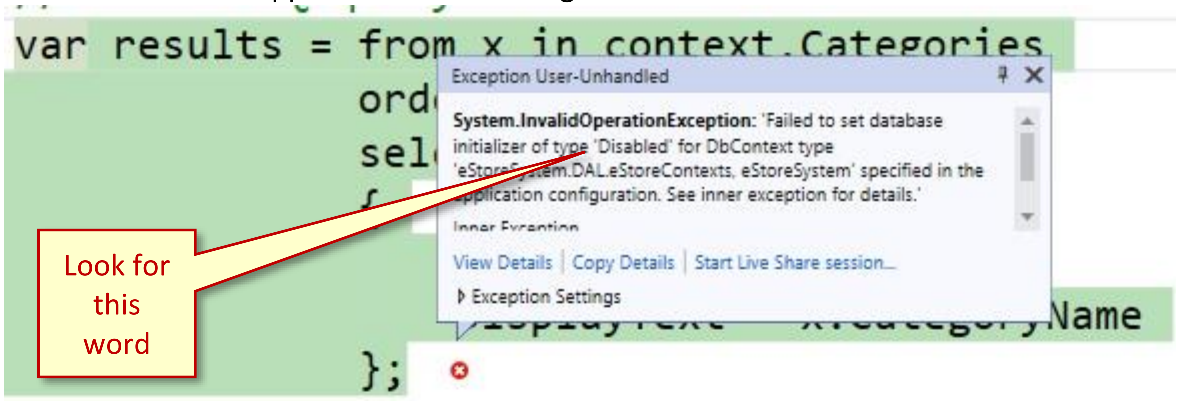
ID	Name	SKU	Description	Price
1	Apple	F-Apple-01	Small apples	2.10
2	Apple	F-Apple-02	Medium apples	2.20
3	Apple	F-Apple-03	Large apples	2.60
4	Banana	F-Banan-01	Organic bananas	3.99
5	Banana	F-Banan-02	Imported bananas	2.38
7	Mango	F-Mango-01	Imported Orgainc Mango	3.50
6	Pineapple	F-PineA-01	Imported Orgainc Pineapple	6.25

© 2019 - eStore

Figure 24: ShoppingCart.aspx in Web Browser

Possible Error Messages

One error that often happens is show in the figures below:



```
var results = from x in context.Categories
              ord
              sel
              {
                  
              };
```

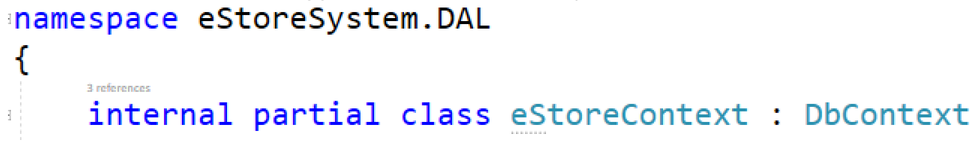
System.InvalidOperationException: 'Failed to set database initializer of type 'Disabled' for DbContext type 'eStoreSystem.DAL.eStoreContexts, eStoreSystem' specified in the application configuration. See inner exception for details.'

Figure 25: Disable Database Initialization Exception

To fix this error you need to check the following code in your Web.config file:

```
<entityFramework>
    <!-- To prevent the creation of a database if the database does not exist -->
    <contexts>
        <!-- type: projectName.DAL.ContextName, ProjectName -->
        <context type="eStoreSystem.DAL.eStoreContext, eStoreSystem"
          disableDatabaseInitialization="true"/>
    </contexts>
```

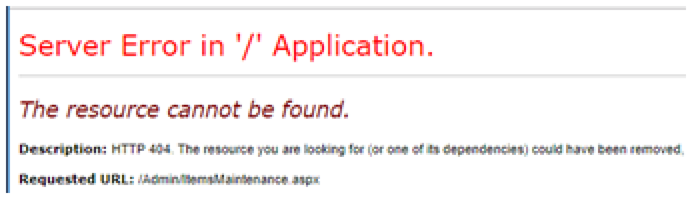
And you need to make sure the **type** matches that of your DbContext class:



```
namespace eStoreSystem.DAL
{
    3 references
    internal partial class eStoreContext : DbContext
```

Figure 26: DbContext

Sometimes you will get an error like the figure below:



Server Error in '/' Application.

The resource cannot be found.

Description: HTTP 404. The resource you are looking for (or one of its dependencies) could have been removed,

Requested URL: /Admin/ItemsMaintenance.aspx

Figure 27: Server Error

This is caused by a timing issue between Visual Studio and the web server. To fix this error, shutdown the browser, close Visual Studio, wait a bit, then restart Visual Studio and open your solution.

Exercise

Complete Exercise 4.2.1.