CT5805701
Software Engineering in Construction Information System
Dept. of Construction Engineering, Taiwan Tech

Introduction to using Microsoft Expression Web to build data-aware web applications

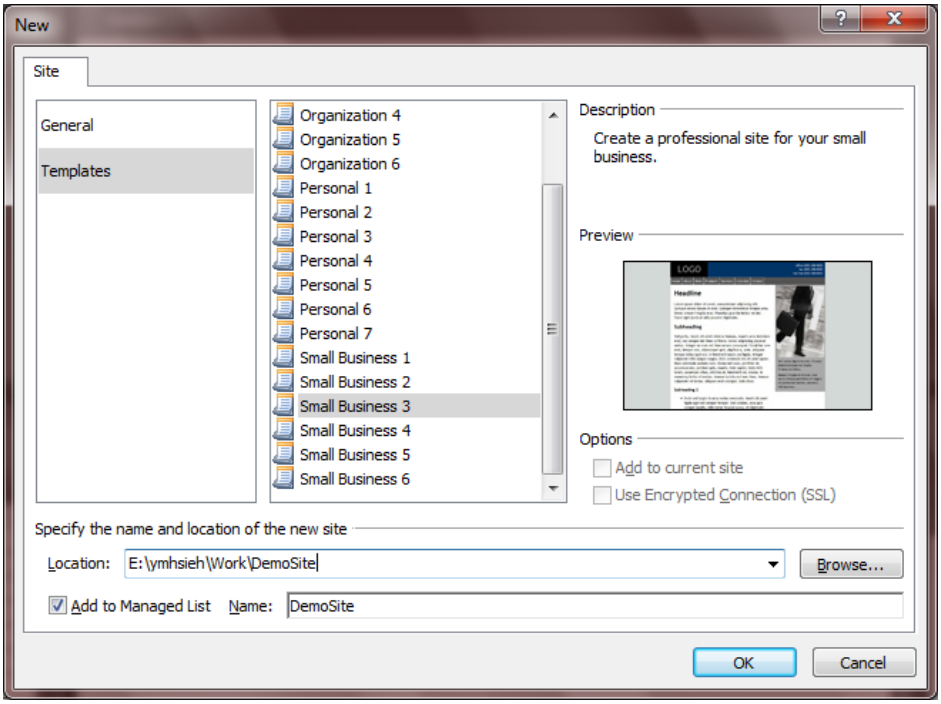
Yo-Ming Hsieh

2011 Winter Edition

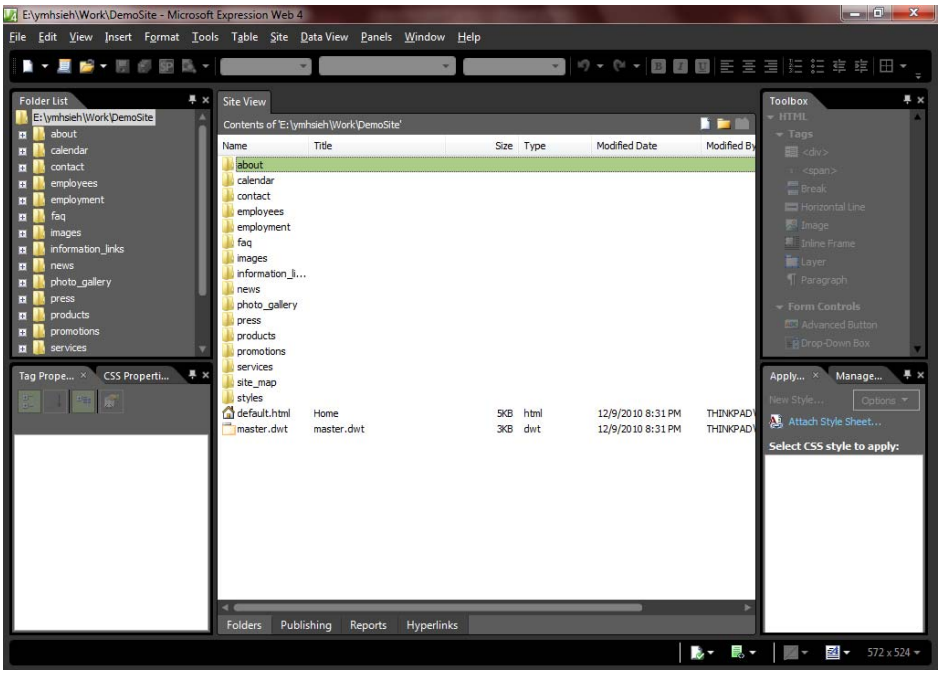
Contents

Preparation

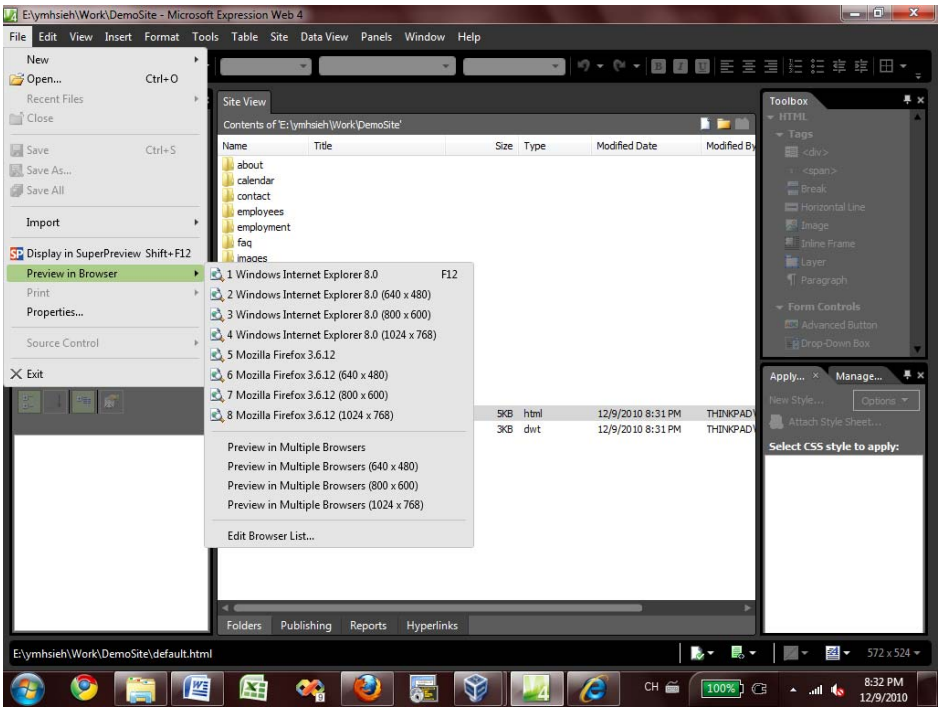
1. Open Expression Web → Site → New Site → Templates → Choose one template from Small Business 1 through 6, or organization 1 through 6. Also, remember or specify where you store your web site. → OK



2. Once the web site is created, you will be presented by a screen look like the following, with a tab "Web Site" showing you many folders, master.dwt, and default.htm.

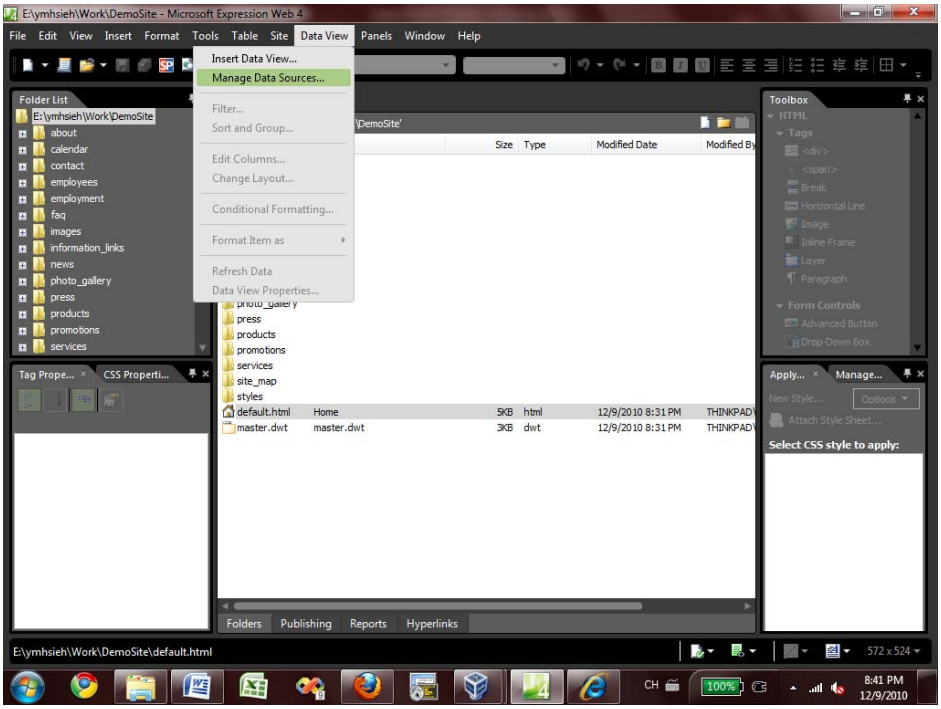


3. Click on “default.htm”, and press “F12” or File → Preview in Browser → Choose one web browser (that installed on your computer) to see how the web site looks like. You should discover that the web site has a consistent look and feel throughout the entire website.

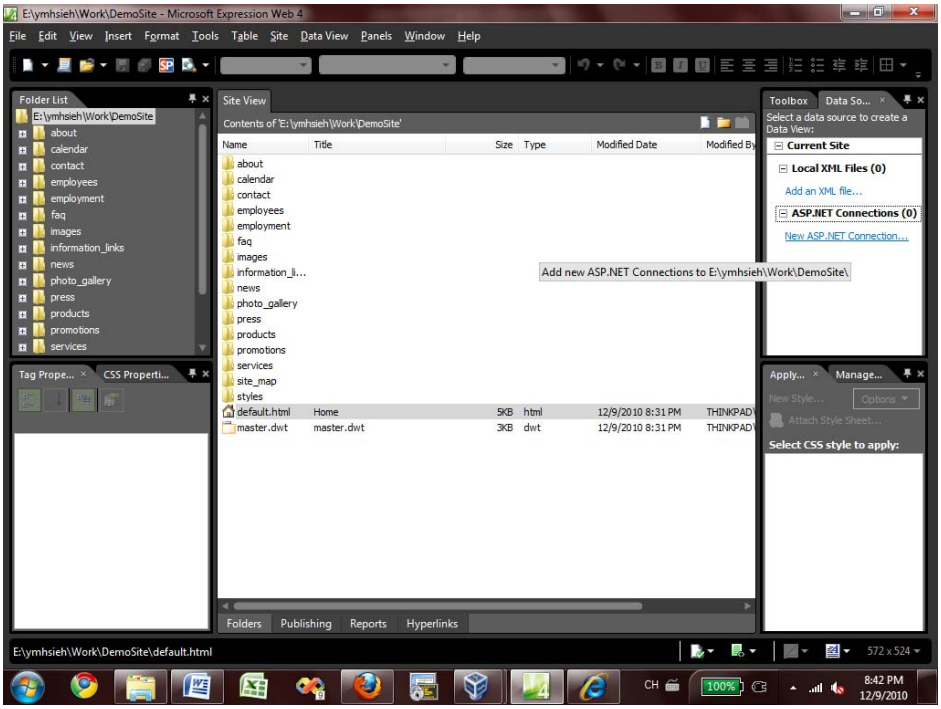


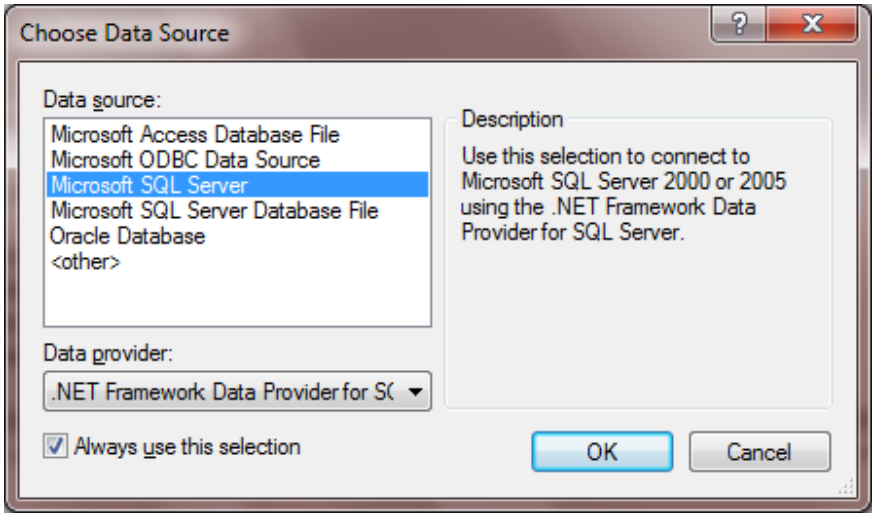
4. The consistent look and feel is done through “Dynamic Web Template” (*.dwt), and the template is located at the root of the created web site called “master.dwt”. Now go back to Expression Web and double click on master.dwt to open it up. You may change text and pictures as you like in the template. You may also design web pages one by one and links them all up by using hand coding and hyperlinks. It is up to you. For the rest of this handout, we will focus on database connectivity.
5. We will implement the following functions in this tutorial:
    - List all categories
    - Show products of a user-specified category
    - **Edit a product**
    - Edit/Delete a product through GridView (DANGEROUS)
    - Search for a product
    - *Add a new product* (I)
    - *Delete a product*

6. Before we start implementing these functions, we must establish a database connection. We will use the same SQL server as we did in the desktop applications:
    - Data View → Manage Data Sources

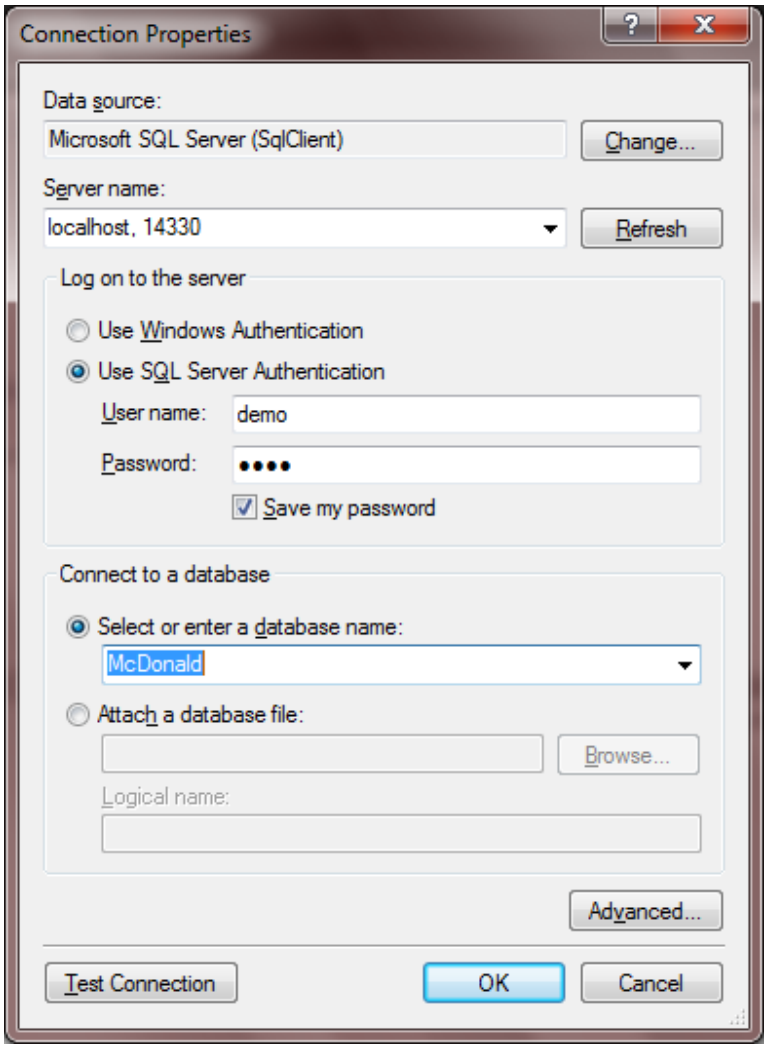


- In the "Data Source Library" Task Pane, choose "New ASP.NET Connection", Choose "Microsoft SQL Server" → "OK"

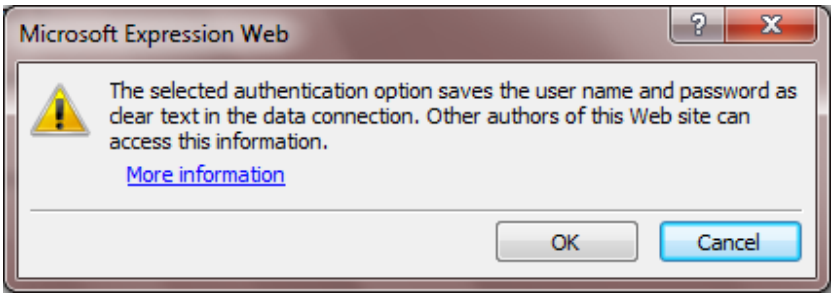




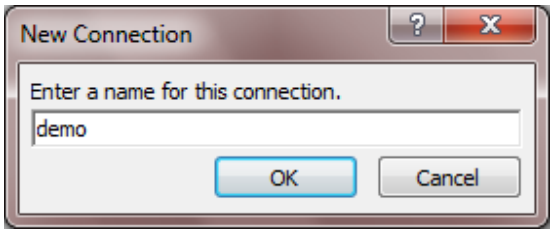
- In the “Connection Properties” dialog box, fill in “Server name”, “Use SQL Server Authentication”, “Username”, ”Password”, “Save my password” “Select or enter a database name”.



- Click on “Test Connection” to make sure it is setup properly. Click “OK” if the test is okay. Otherwise, recheck your inputs. Once you clicked on “OK”, you will be given the following warning dialog box to warn you that saving my password will save the entered account information in clear text. We are going to do this insecure practice. Click on “OK” to close it.



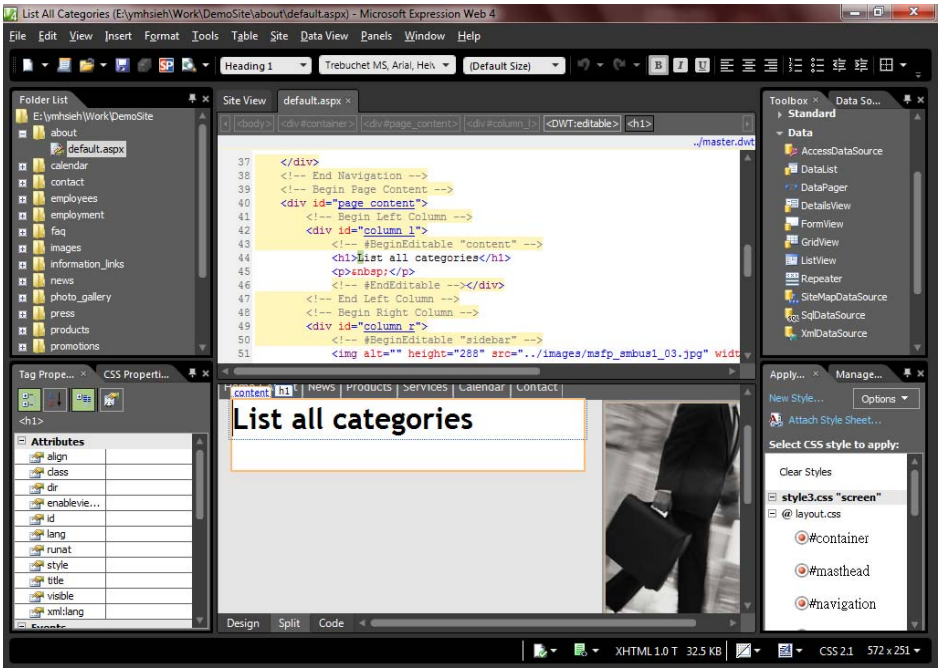
- You will then be prompted to enter a connection string, enter whatever you like as long as you can remember it. Click on “OK” to close it.



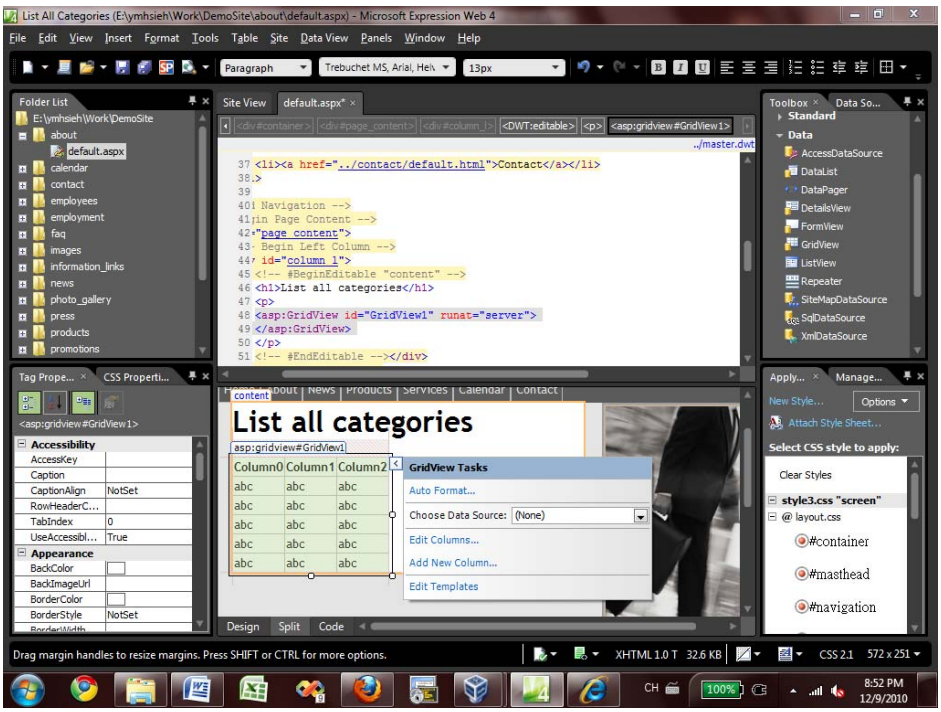
- You might notice that a file “web.config” appears in your web folder. That is the file containing how to connect to SQL server, along with the important username and password.
- We now have established a data connection, and we are ready to develop data-driven web site.
- In the implementation of the following functions a through e, you need to switch to the appropriate web page before you start working on it.

A. List all categories

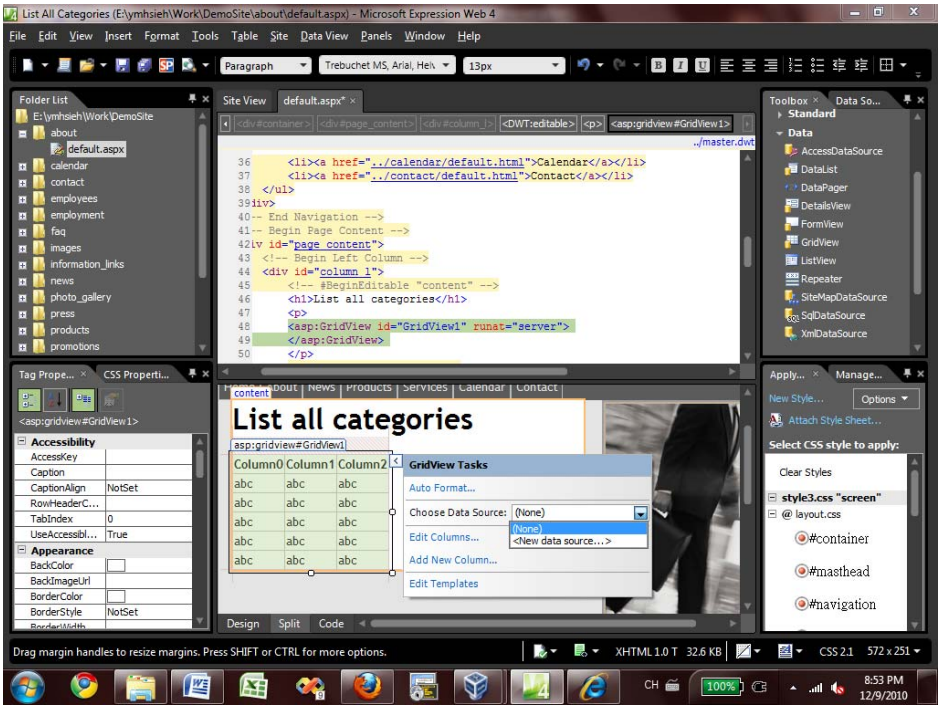
1. In the “List all categories” webpage (choose any page you like if you use template described previously) click on the appropriate place on the webpage to start working on the function. Note if the page should have database connectivity, rename it to have.aspx extension! Otherwise it will NOT work. Use the “Split” view to help you understand what happens to the underlying HTML webpage when you edit it through the editor.



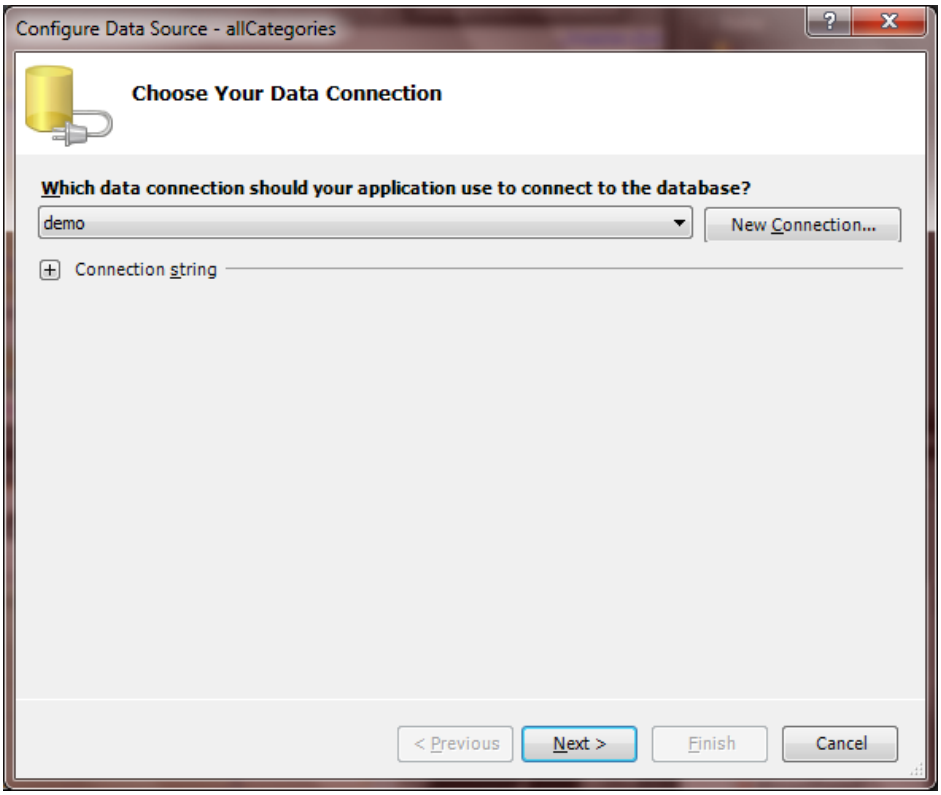
2. In the “Tool box” task pane, find “Data” under ASP.NET Controls, and choose “GridView”, drag and drop it onto the appropriate place on the webpage.



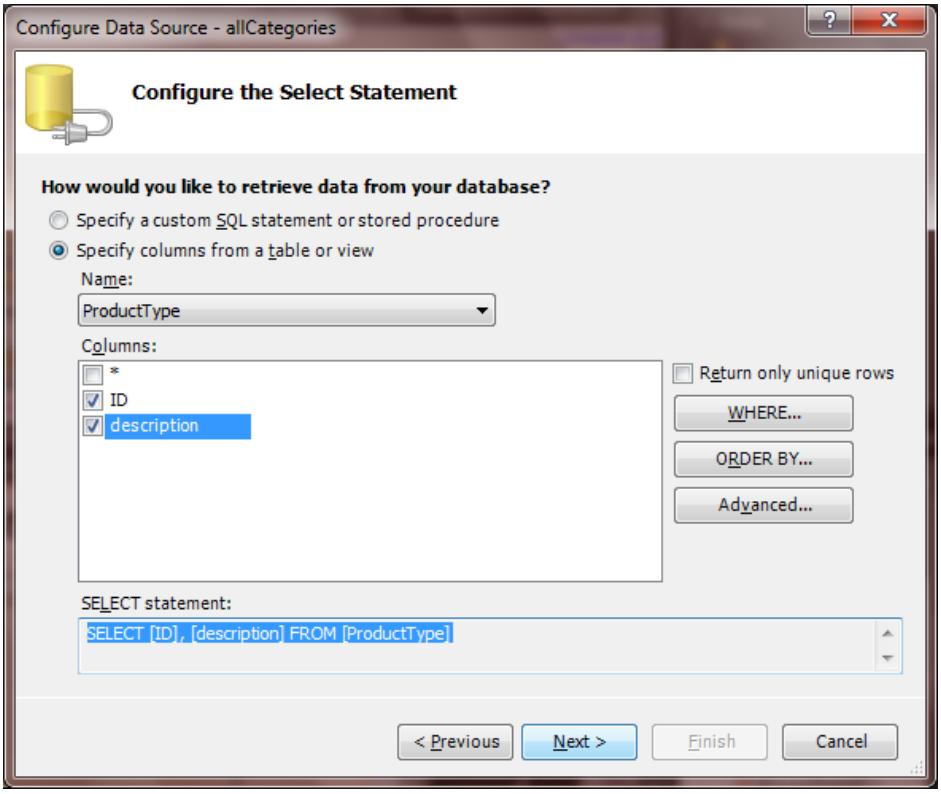
3. In “GridView Tasks”, choose “<New Data Source>“ in “Choose Data Source”, choose “Database” as “Where will the application get data from?”, and give the data source an ID “allCategories”. Click on “OK”.



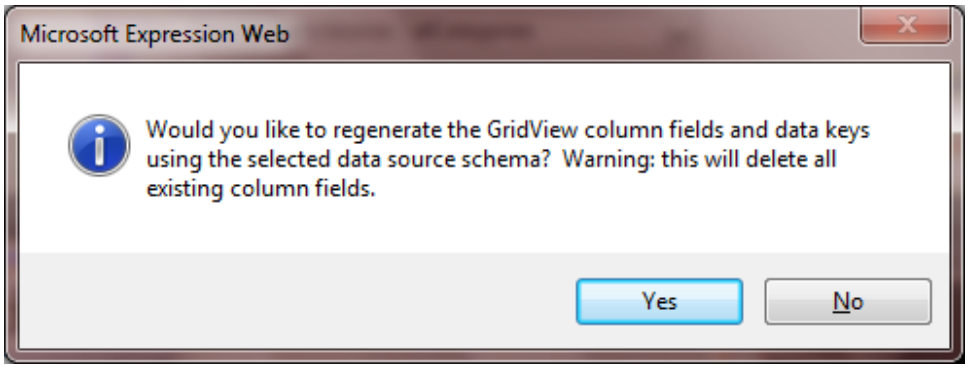
4. Choose “demo” or the name of your choice when we prepare the data connection. Click on “Next”.



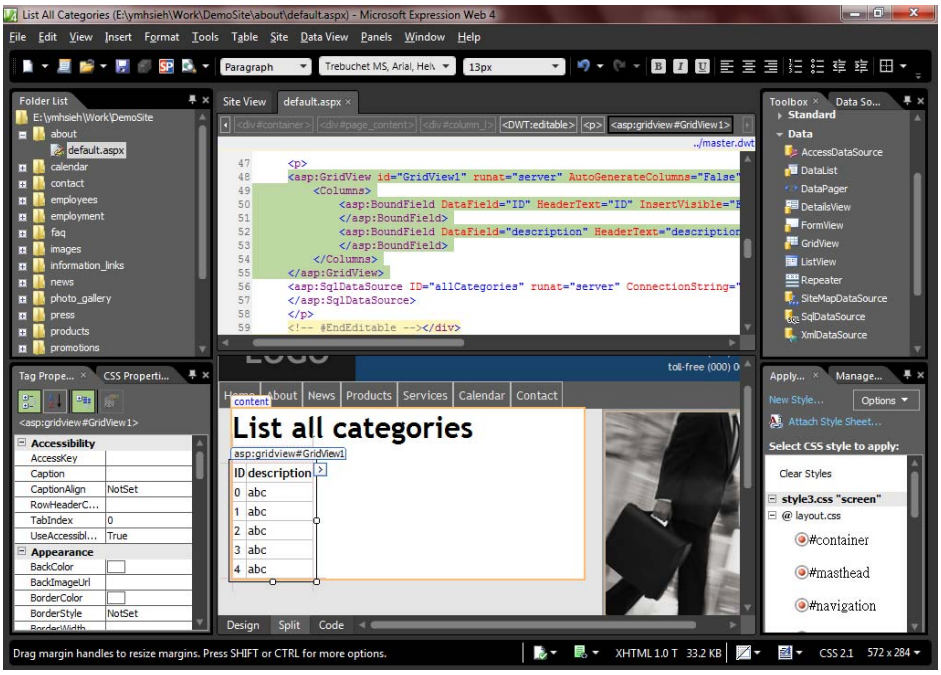
5. Next we need to configure the data source. In this example, we choose “Specify columns from a table or view”, choose “ProductType” as Name of the table or view, and check “ID”, “Nutritions” to create a SQL statement:
   SELECT [ID], [description] FROM [ProductType]
   Click on “Next>” and “Finish”.



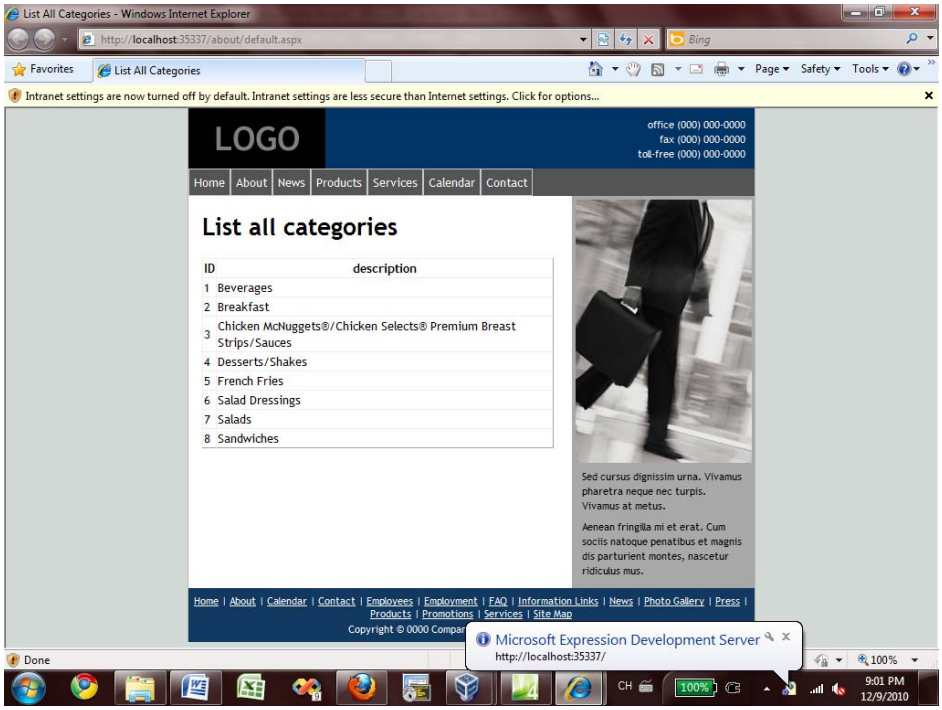
6. If you are warned to regenerate the GridView, choose “Yes” to reflect the columns that you selected in the previous dialog box into the GridView on the webpage.



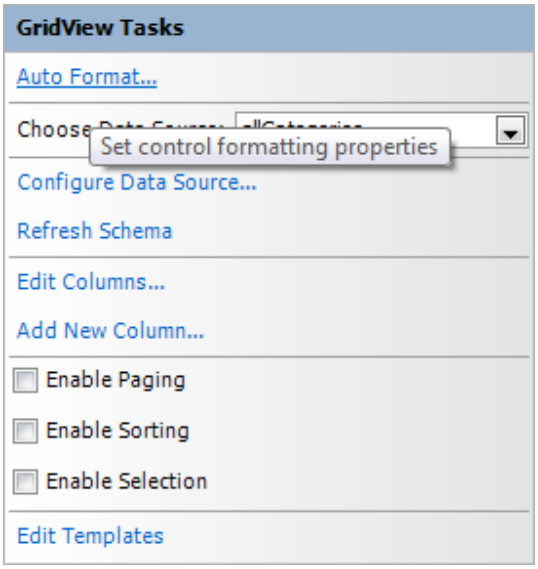
7. The webpage should now look like the following.

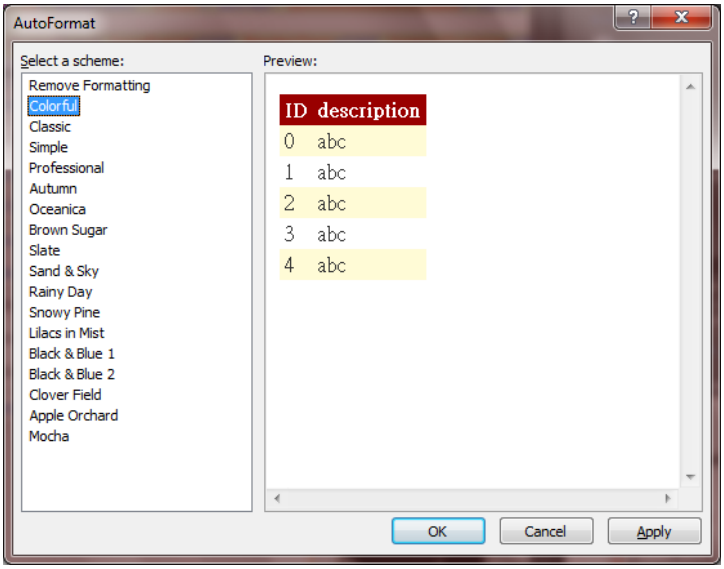


8. You may now press [F12] to preview this page. Note that when you preview this page, a local web server software is actually launched by ExpressionWeb, and the URL for your page is shown in the address bar of IE that looks like: http://localhost:35337... The port is picked randomly.



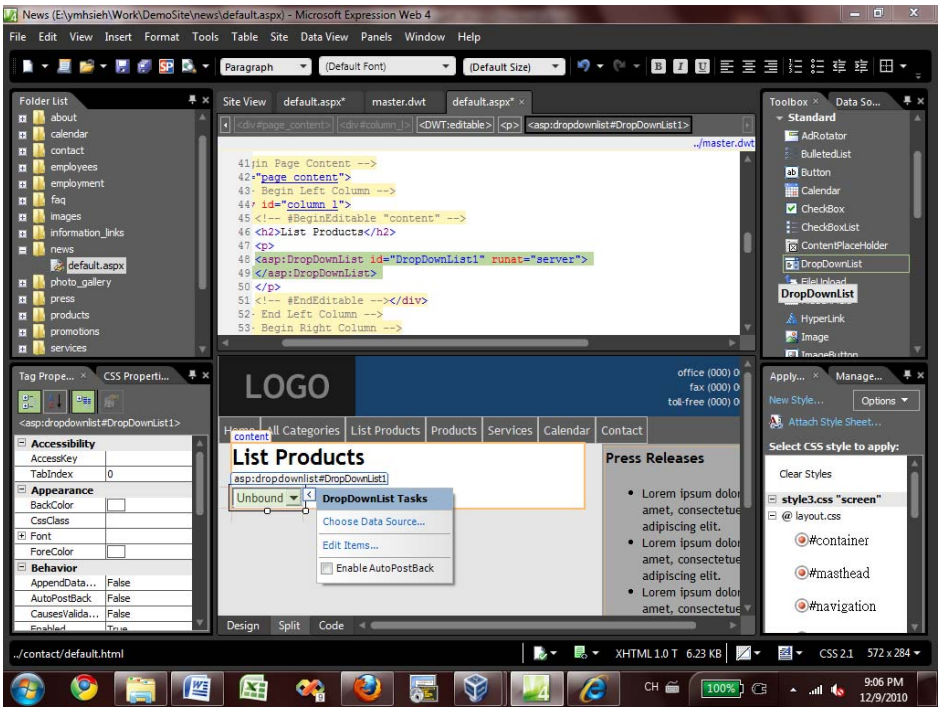
9. You may want to play around the GridView control to customize how it looks by opening up the “GridView Tasks”. Now we have a webpage capable of displaying all categories.



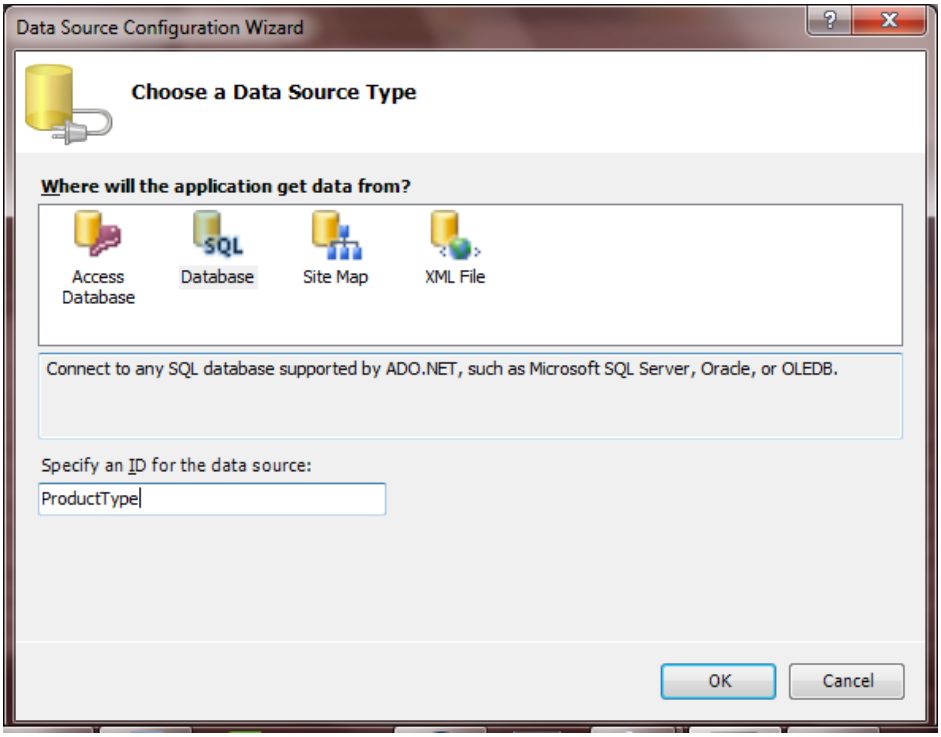


B. Show products of a user-specified category

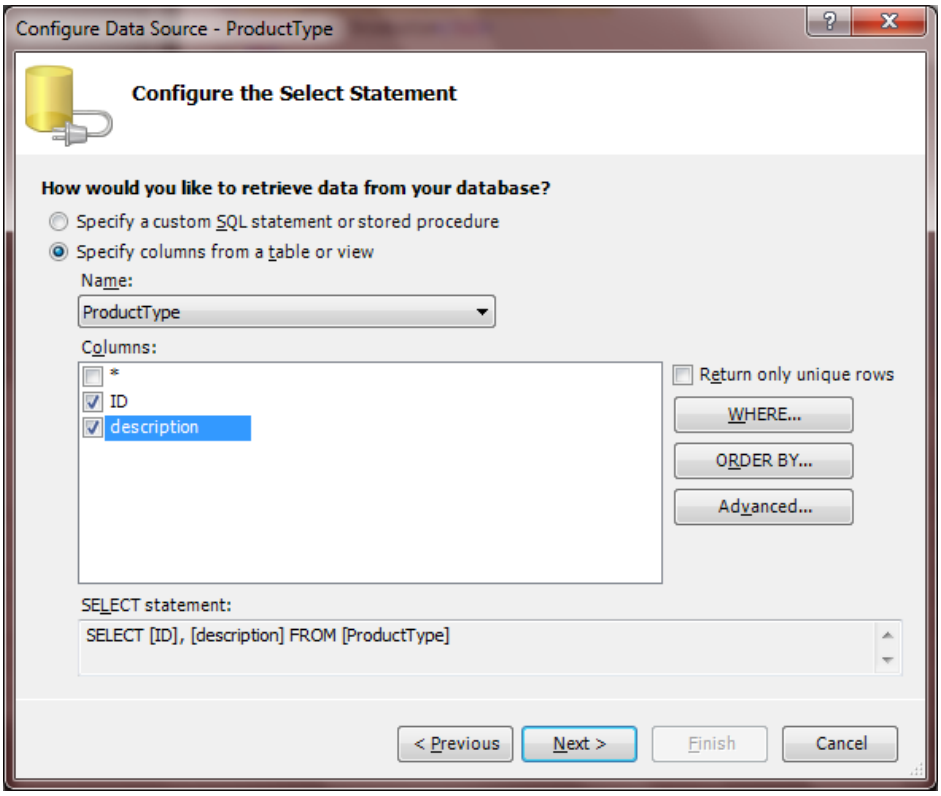
1. On the new page, find “DropDownList” under “Standard” of “ASP.NET Controls” in the “Toolbox” task pane. Drag and drop “DropDownList” onto the new webpage.

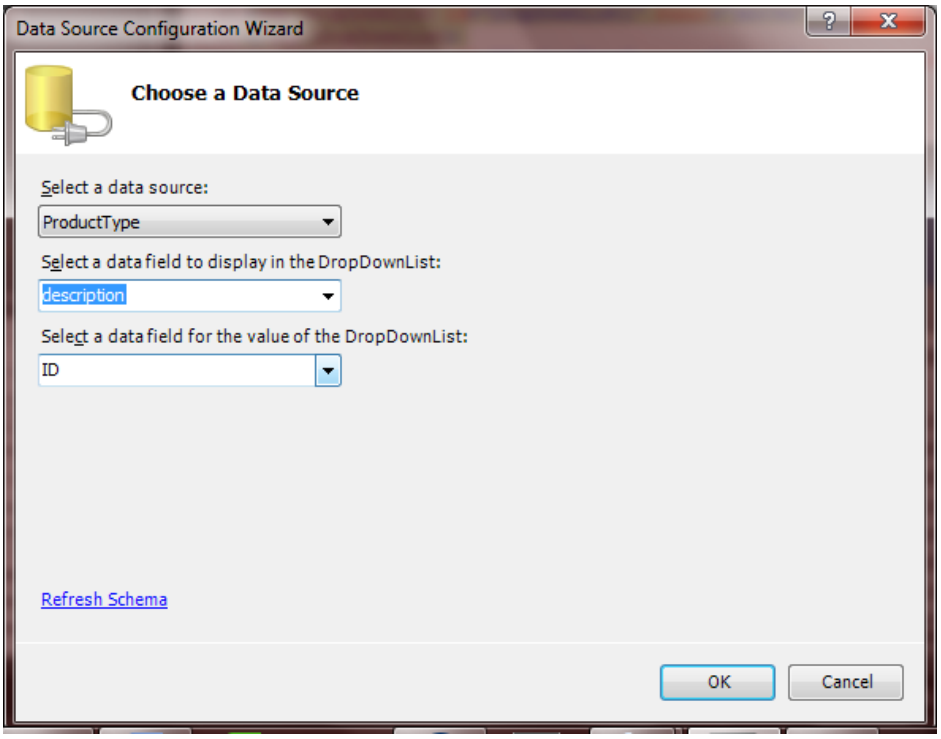


2. Configure the newly added “DropDownList” by “Choose Data Source” → <New Data Source> → Database → specify “ProductType” as the ID for the data source.

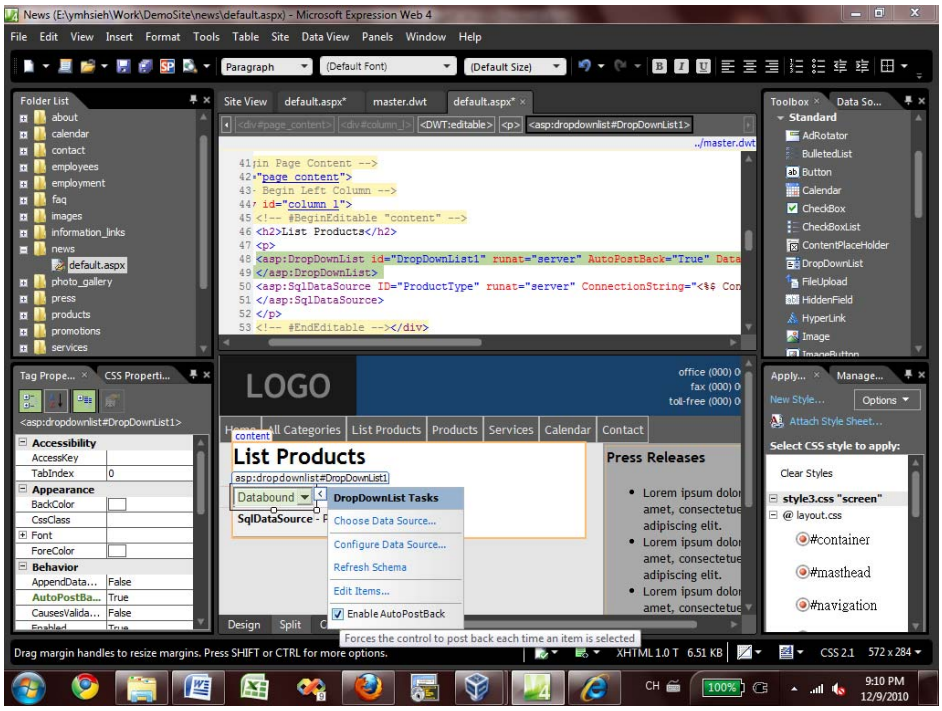


3. Click on “OK” → choose “demo” as the data connection → “Next>” → Use “ProductType” table to specify columns from a table or view → Check ID and description for Columns → “Next>” → Display “description” in the dropdown list and use “ID” for its value → “OK”.

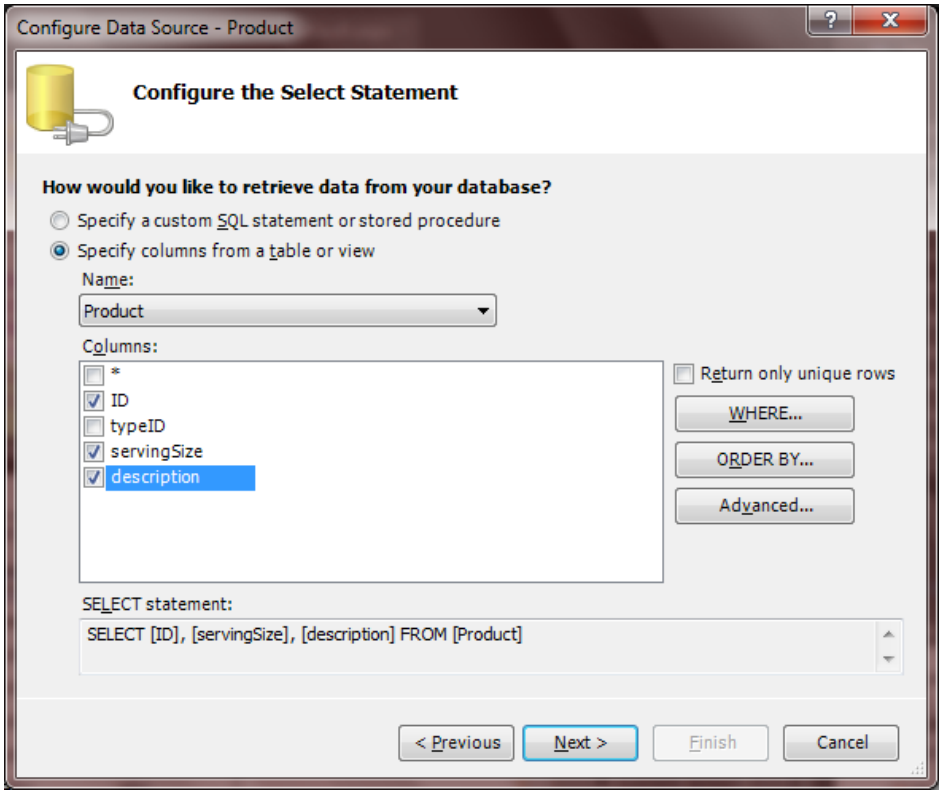




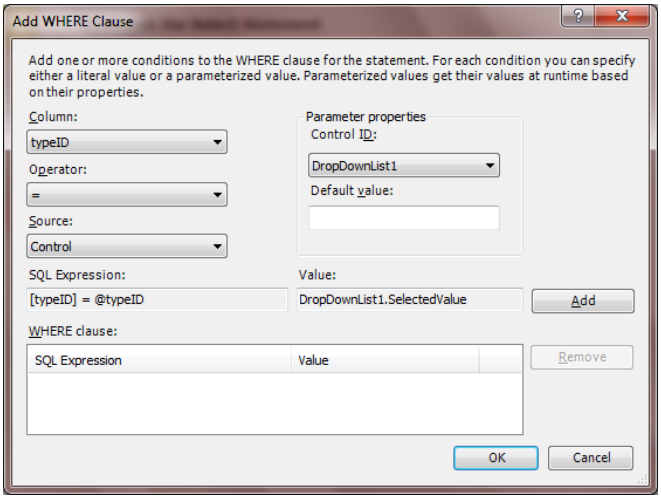
4. Also check the “Enable AutoPostBack” for the DropDownList. Note we now have a “SqlDataSource – ProductType” right after the DropDownList.

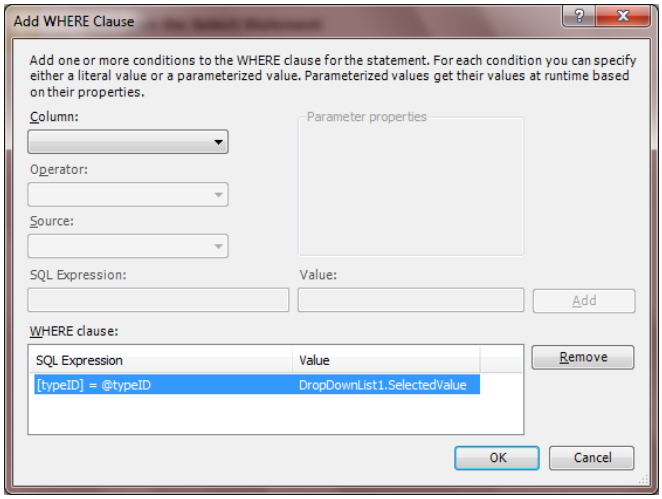


5. Now drag and drop a “GridView” right after the DropDownList we just created. For the new “GridView”, we create a “<New Data Source>” named “Product”, and use the same connection “demo” we created. For the new data source, specify “Product” to get columns, check ID, servingSize, and description columns, and then click on “WHERE” button.



6. Next, you will be presented a dialog box for adding the WHERE Clause. Choose “typeID” as Column, “=” for Operator, “Control” for Source, “DropDownList1” as Control ID, and finally click on “Add”. We are building the query for showing products for a specified product type, which is selected using the DropDownList.

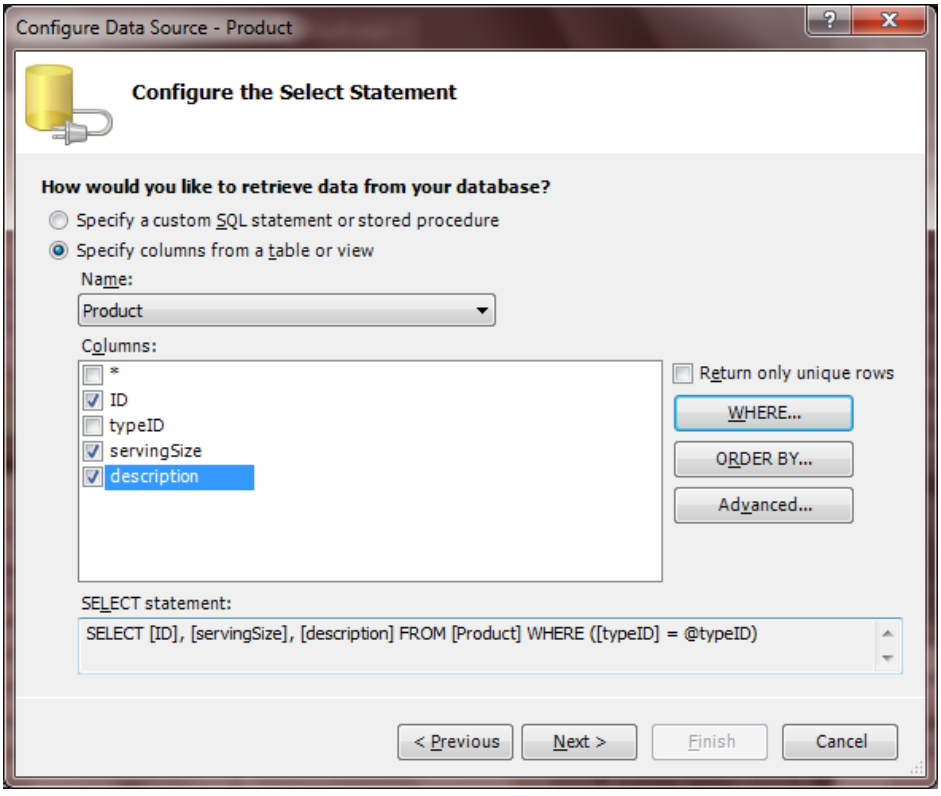




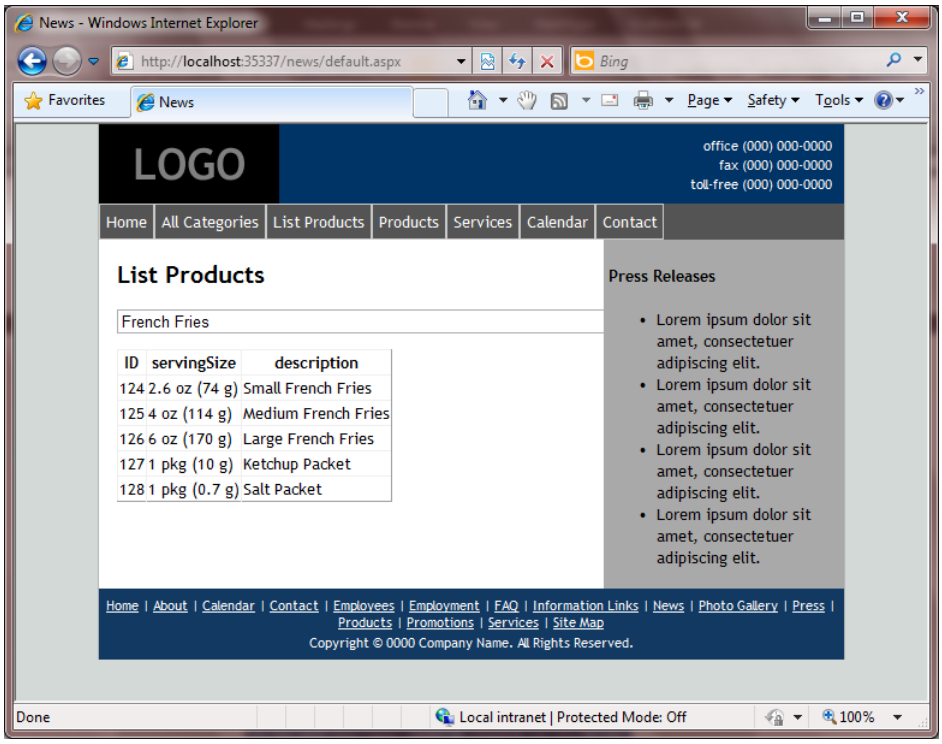
7. Click “OK” to finish adding WHERE Clause. Note that it generate a SELECT statement:

```
SELECT [ID], [servingSize], [description]
FROM [Product] WHERE ([typeID] = @typeID)
```

Click “Next” and then “Finish”.

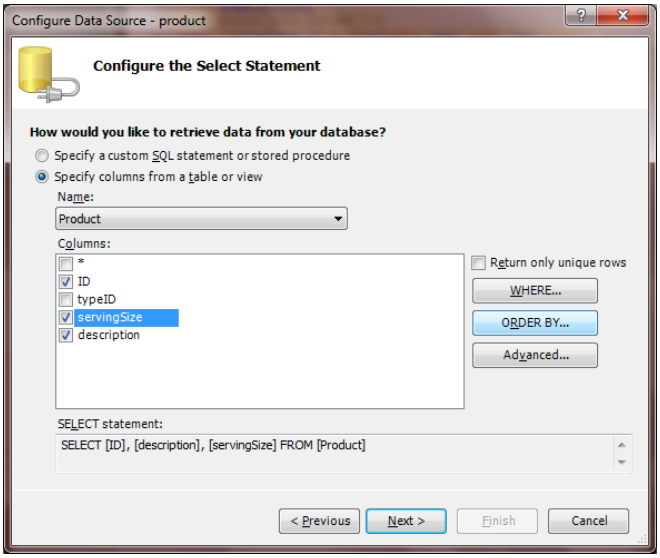


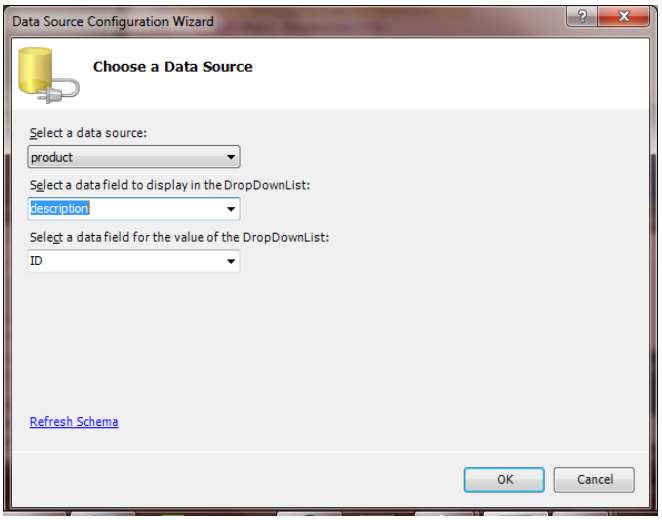
8. Now you can test flight this webpage by previewing it. On this webpage, we use a DropDownList (ComboBox in VC#) to select a particular product type (French Fries), and the GridView will show all products under the specified product type.



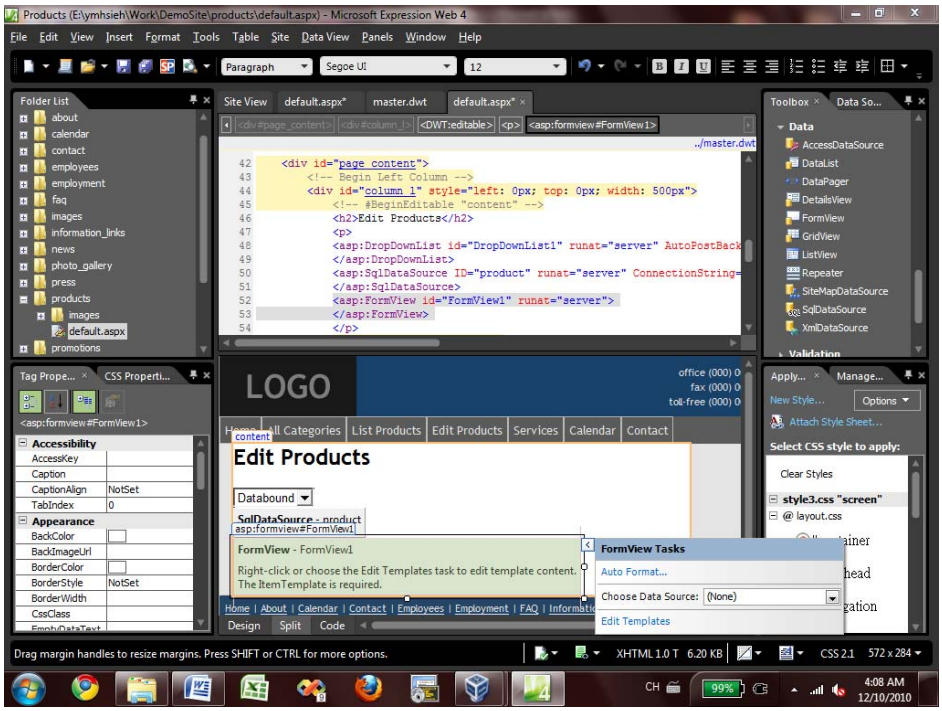
C. Edit a product

1. In this example, we are going to construct a webpage to update a product's information. The design consists of two controls, one control using DropDownList to enable users select a product to update its information, and the other control FormView to update the selected product's data.
2. First, drag-and-drop a DrawDownList control onto the webpage you want to edit. The dropdown list uses the same procedure as the previous example, and we choose ID, servingSize, and description columns for this control. We want to display the description, and use ID as the control's value. Again, don't forget to do "Enable AutoPostBack" .

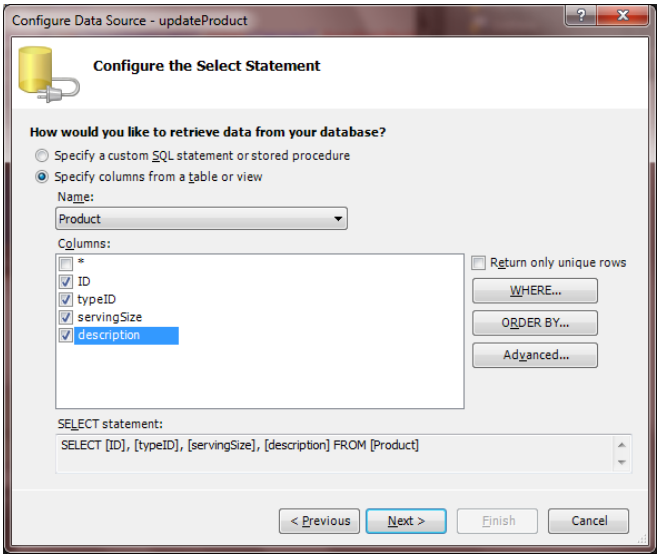


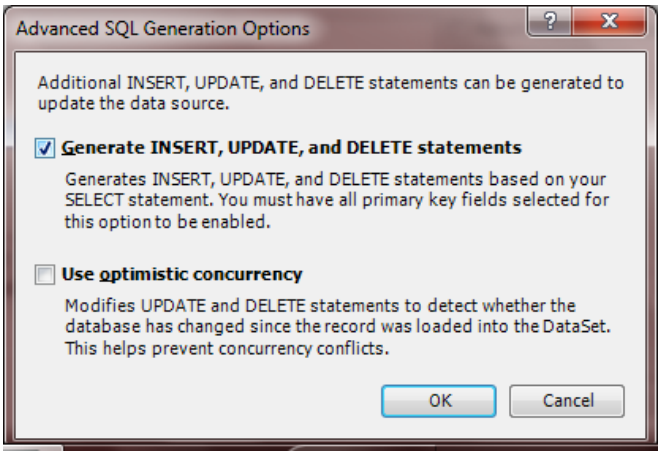


3. Now, if you preview the webpage, you should be able to see a DropDownList showing all product's descriptions. You should observe how the webpage's source code is varying during your operations. Now, we want to add a "FormView" to update the selected product's information. Thus, find "FormView" control under "Data" of ASP.NET Controls and drag-and-drop it onto the webpage.



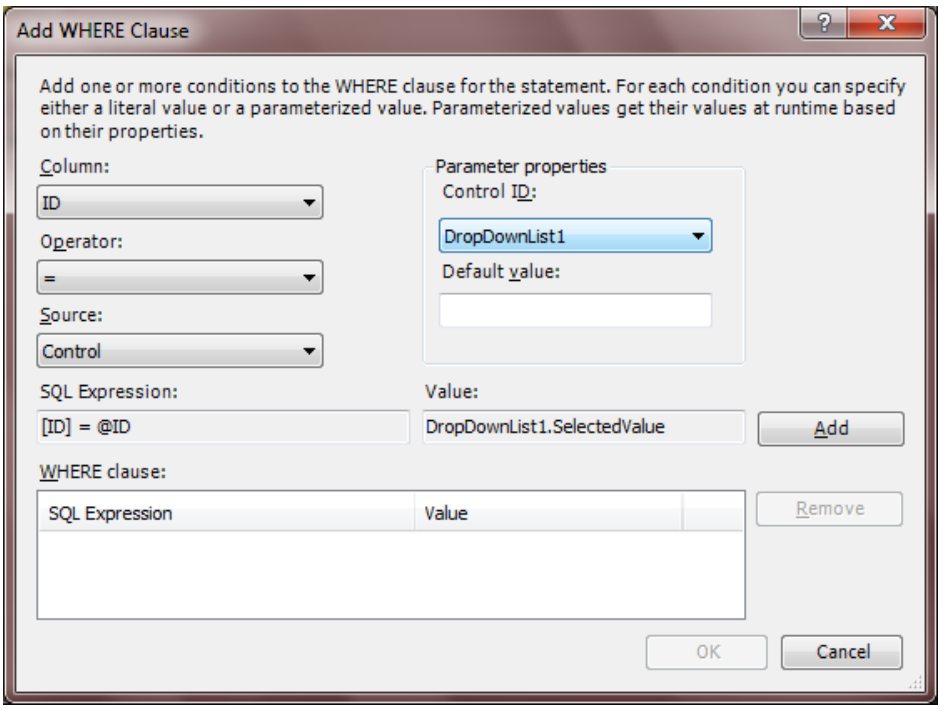
4. Similarly to the previous steps, we choose “New Data Source”, create a new data source using Database and name the data source “updateProduct”. Every step is almost the same as the previous data sources. But this time at “Configure the Select Statement” step, we need to choose “Advanced” and check “Generate INSERT, UPDATE, and DELETE statements” and then press “OK”.



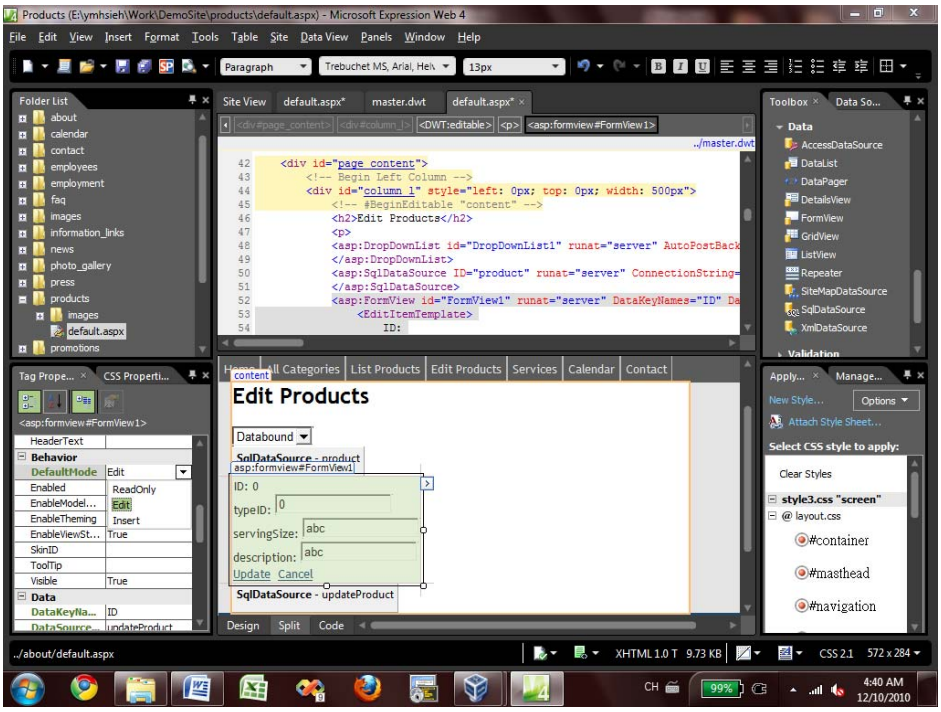


5. Also, at the step “Configure the Select Statement”, use “WHERE” to limit the record to be updated to be the selected Product from the control from DropDownList1. Set the Column to be “ID”, Operator to be “=”, Source to be “Control”, Control ID to be “DrawDownList1”, and finally press “Add” to add the “WHERE clause” . So that we construct a SQL statement:

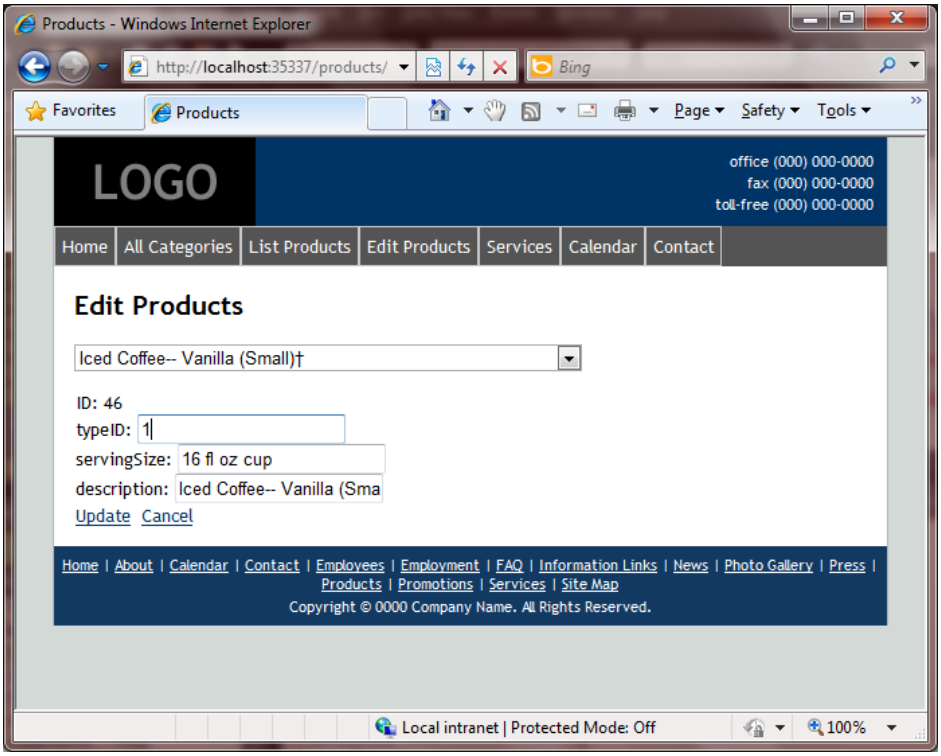
```
SELECT [ID], [typeID], [servingSize], [description]
FROM [Product] WHERE ([ID] = @ID)
```



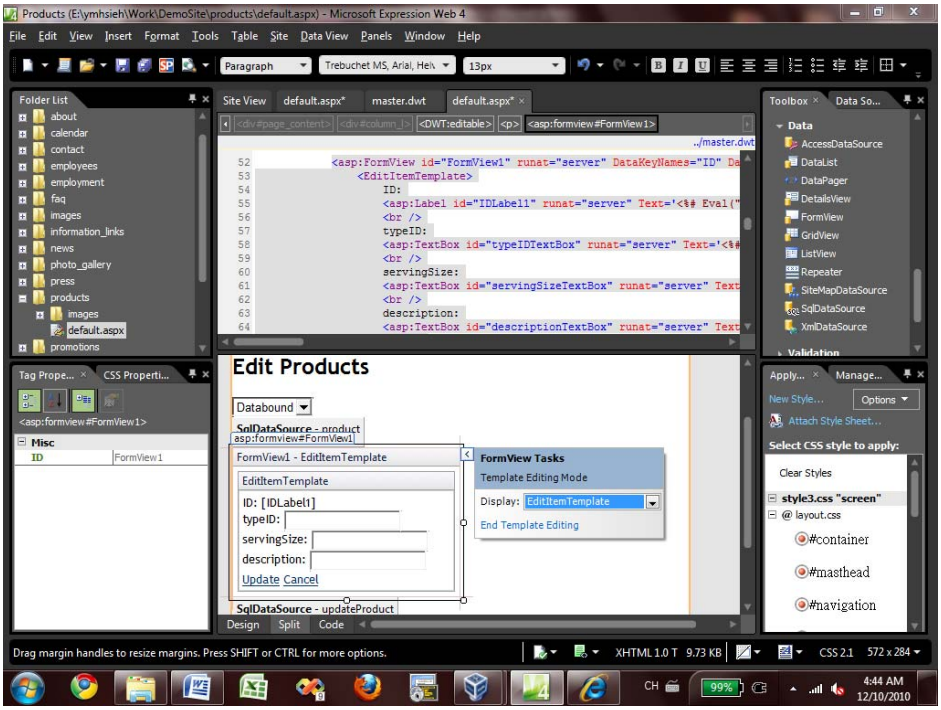
6. The “FormView” can be used to Insert, Update, and View data. By default, the FormView shows the data item, but we only want to use it for updating data. Therefore, we need to change the default behavior. Click on the FormView control, look for “DefaultMode” in the “Tag Properties” task pane, and change it from ”Read Only” to “Edit”.



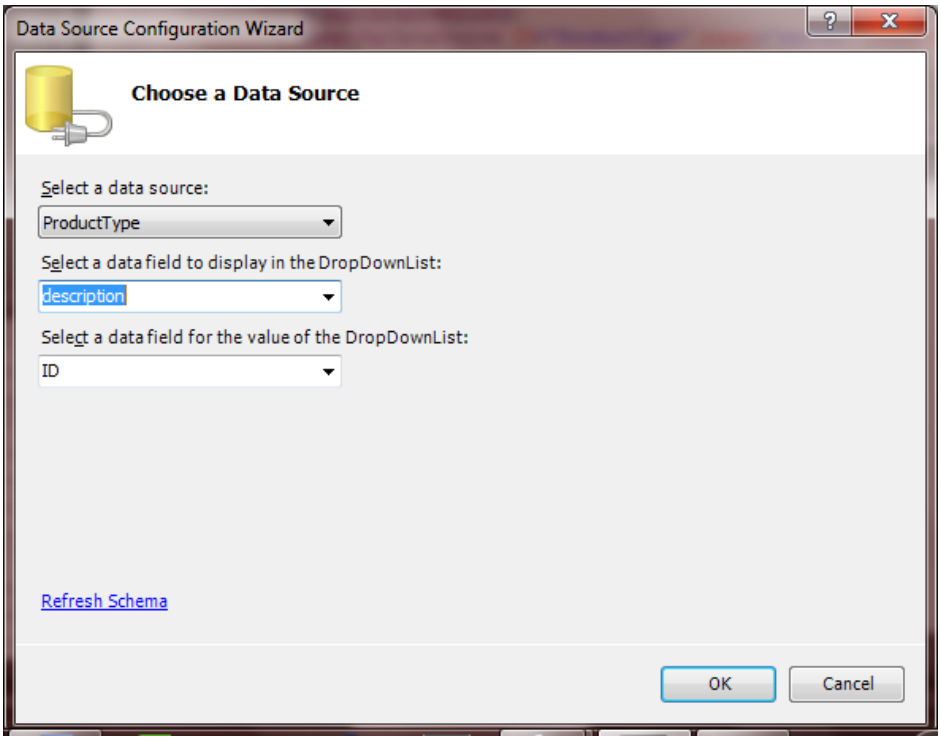
7. Now the page can be used to update product information. By previewing:



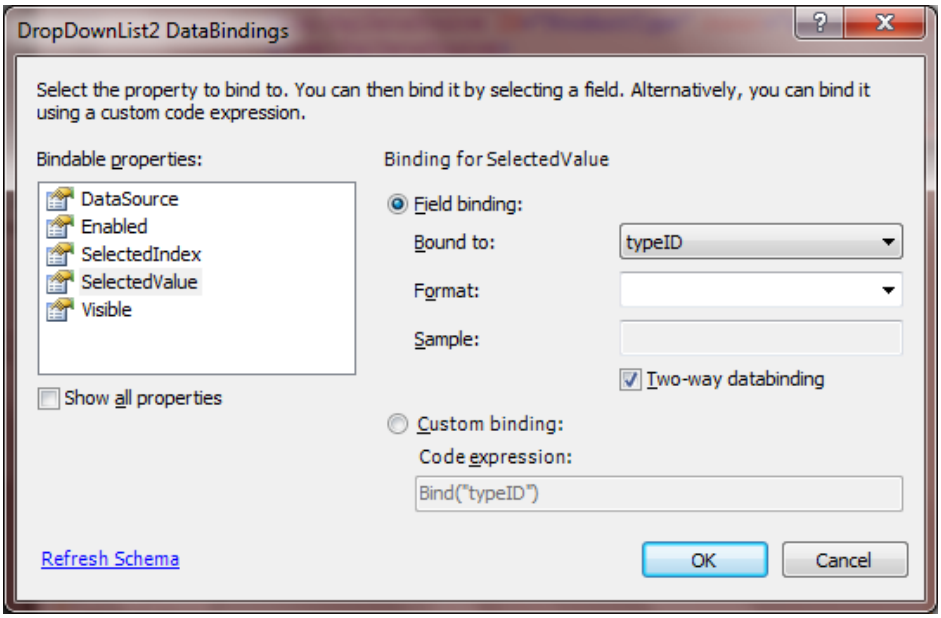
8. However, for the field “ProductType”, it is better to use “DropDownList” to update its value to avoid any potential error in the value of that field. Using DropdownList is also more user-friendly. To do so, we edit the template for the FormView, and change the “Display:” to “EditItemTemplate”



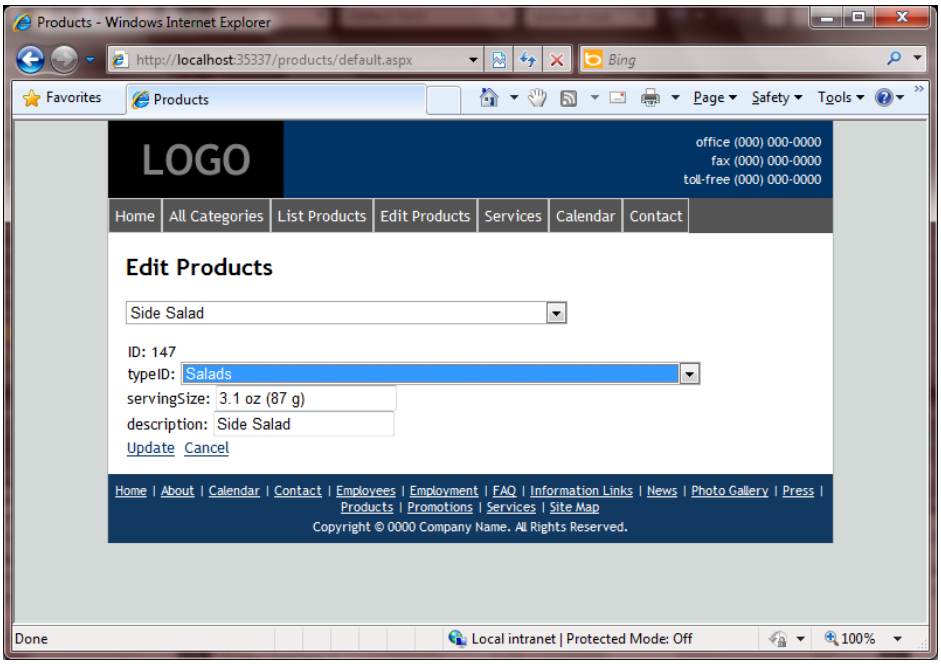
9. Click on the TextBox for field ProductType, and delete it. Then, drag-and-drop a DropDownList to replace the original TextBox. Then we edit its DataBindings. Use “ProductType” as its binding source (you need to create a new one), display “description” and use “ID” for value. This determines what shows in the DropDownList and what the value is when the DropDownList is selected.



10. Then we edit the DropDownList's data bindings, so that when user hit the Update link, the data sent to database will be the selected value of the DropDownList. For this DropDownList, we do not need to enable "AutoPostBack".

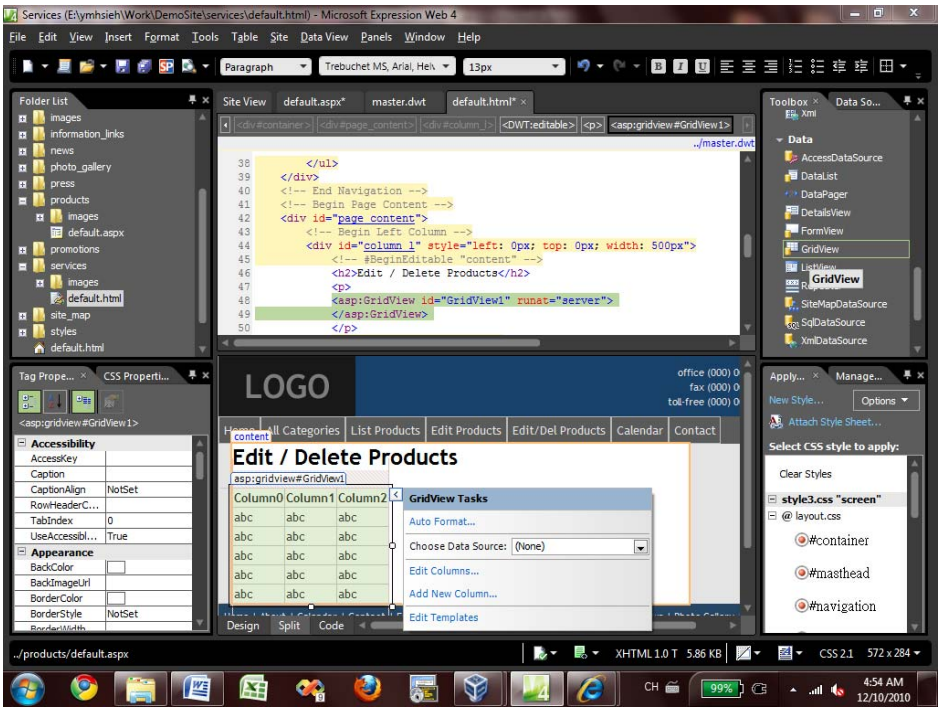


11. Now, you can preview this webpage. Now it uses the DropDownList to show and to change a product's type.

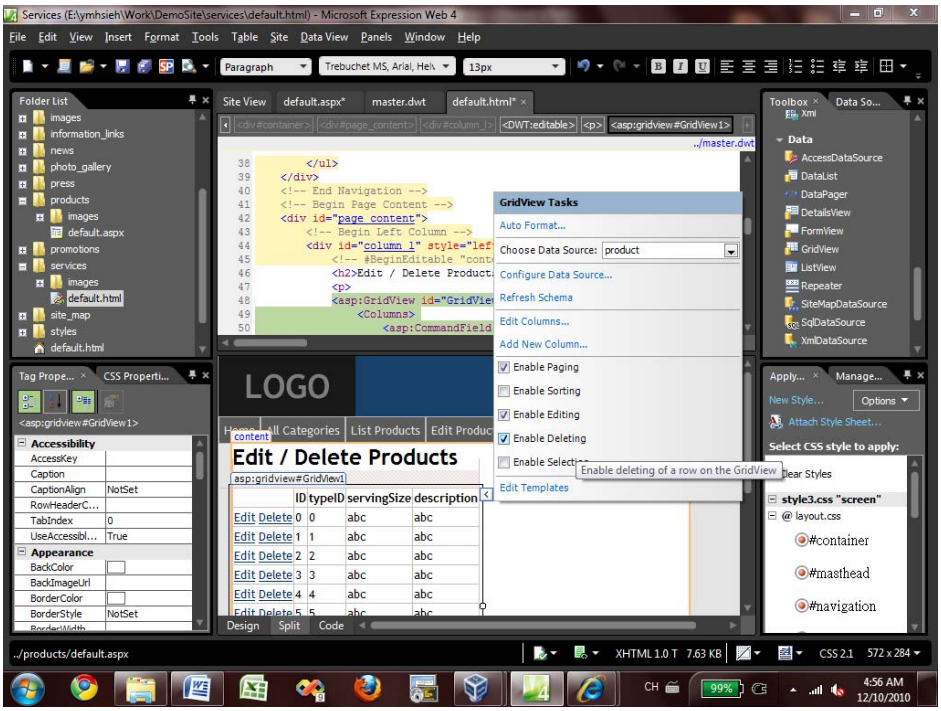


D. Edit/Delete a product through GridView

1. Although we have learned on how to update a product. Let's try a different approach through GridView. On a new webpage, drag and drop a "GridView" onto the webpage.



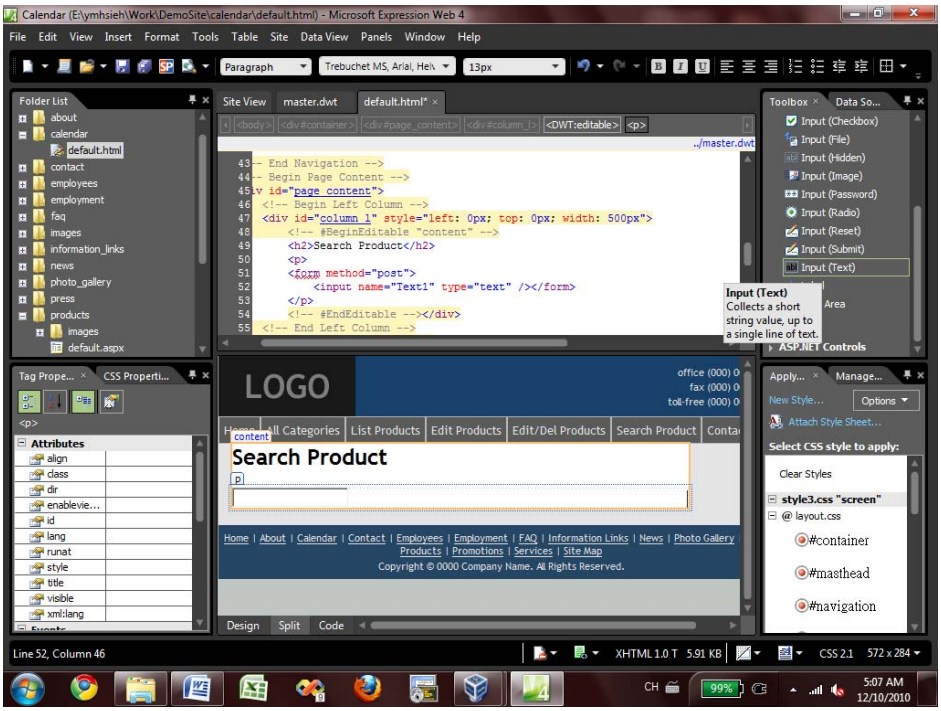
2. Add new data source to the GridView as we always do. Remember click on "Advanced" to generate "INSERT, UPDATE, and DELETE statements" for the new data source called "Product". Once finished, check "Enable Paging", "Enable Editing", "Enable Deleting" for the GridView.



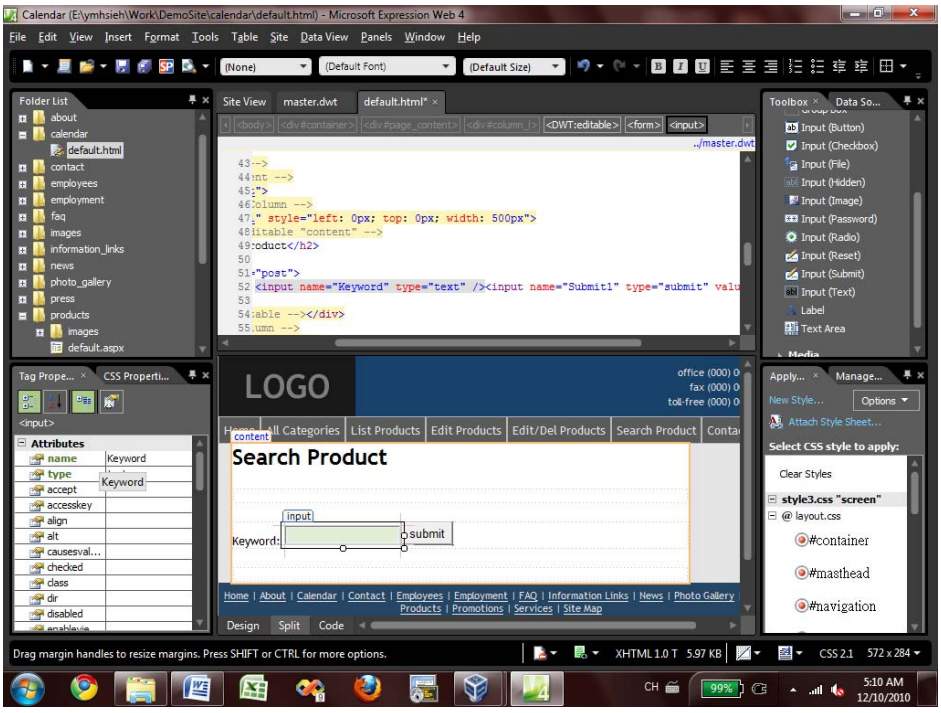
3. Now we have a webpage that allows us to update and delete products using GridView! But DO remember to setup cascading DELETION in order not to break the referential integrity between tables! (e.g. Nutrition facts referring to non-exist products).

E. Search for a product

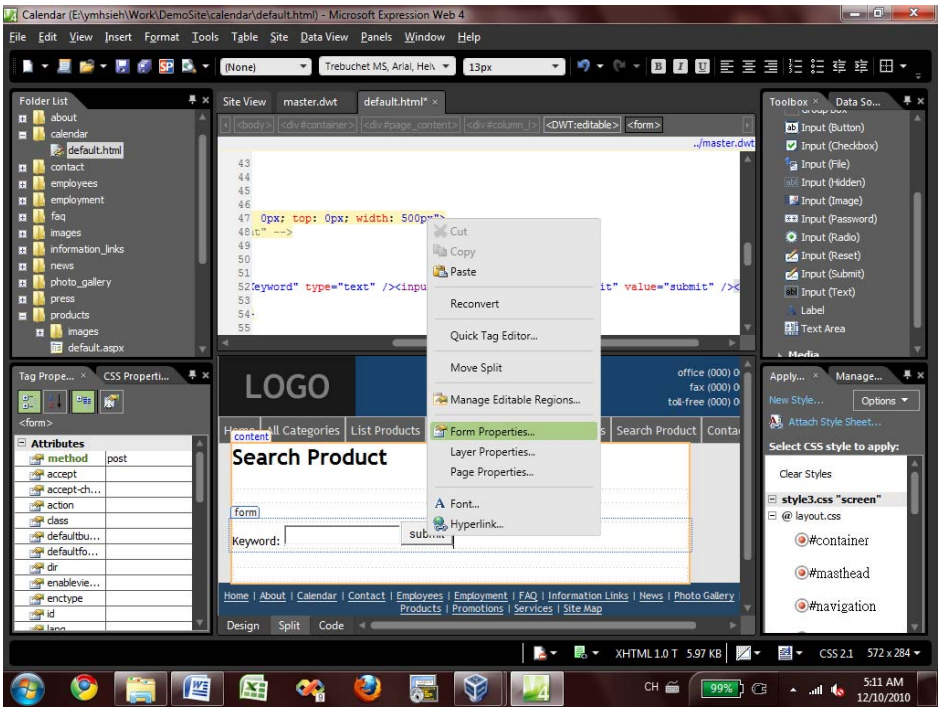
1. In this example, we are going to create a standard HTML form, and use ASP.NET to process the input of the form. On a new webpage (*.html), drag-and-drop an “Input (Text)” control on your webpage. It should be noted in the source code view, a HTML element <form> … </form> is automatically inserted for you to wrap up the <input name… /> element.



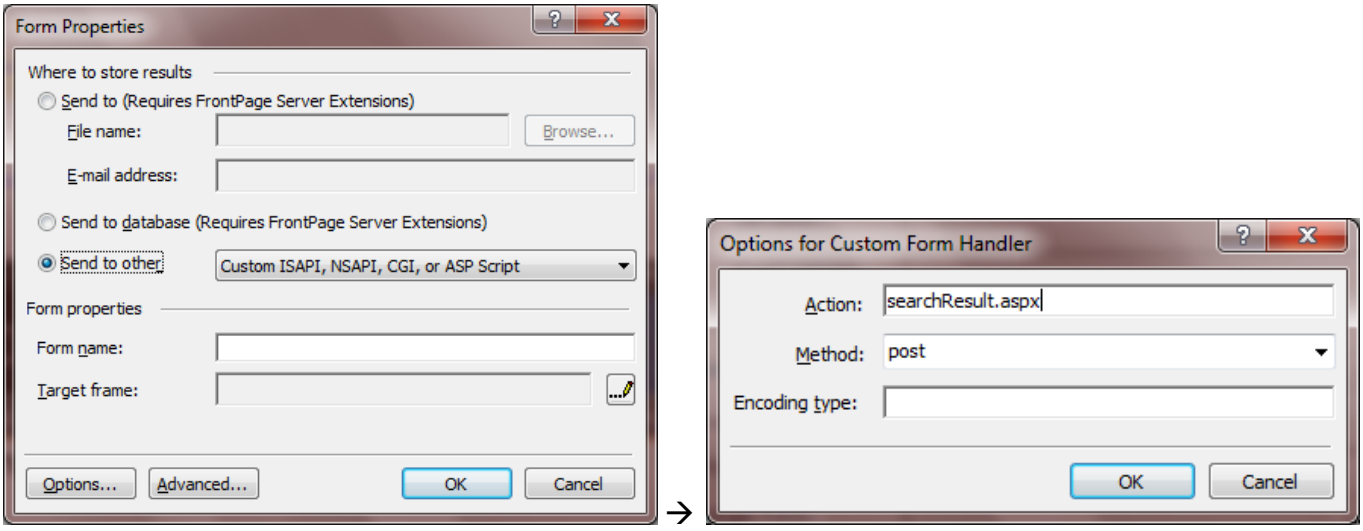
2. In front of the Input control, enter plain text “Keyword: “. Then, drag-and-drop an “Input (Submit)” right after the input (Text) control. Also, click on the Input(Text) control, and change its name from Text1 to “Keyword” in the “Tag Properties” pane.



3. In the area of the form (surrounded by a light-dotted area), right click and choose “Form Properties”

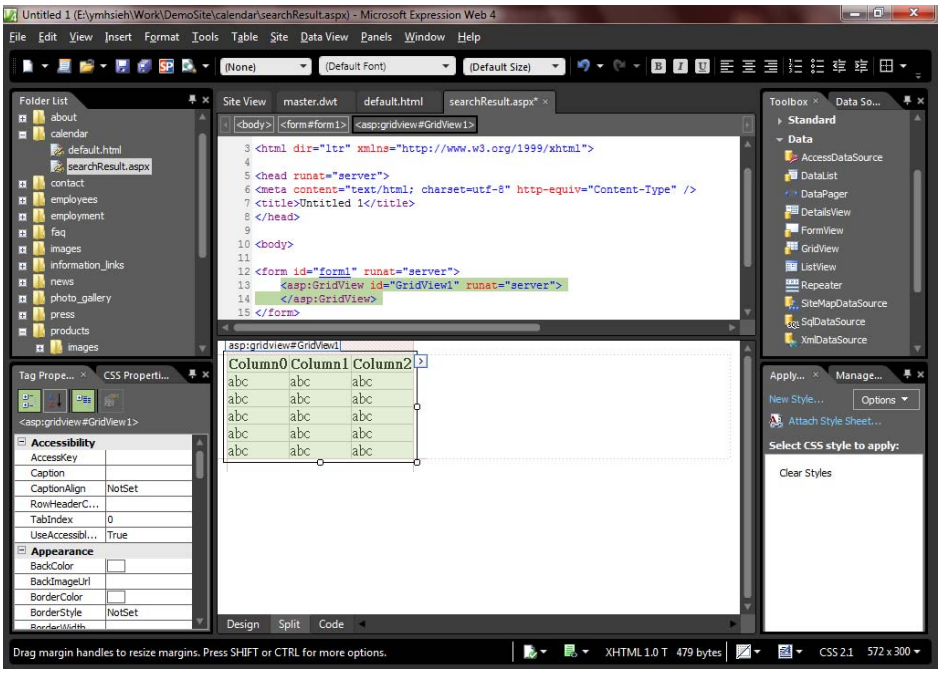


4. Choose “Send to other: …” (the default) → Click on Options → enter “searchResult.aspx” as the “Action”. Click “OK” → “OK” to close the dialog box.

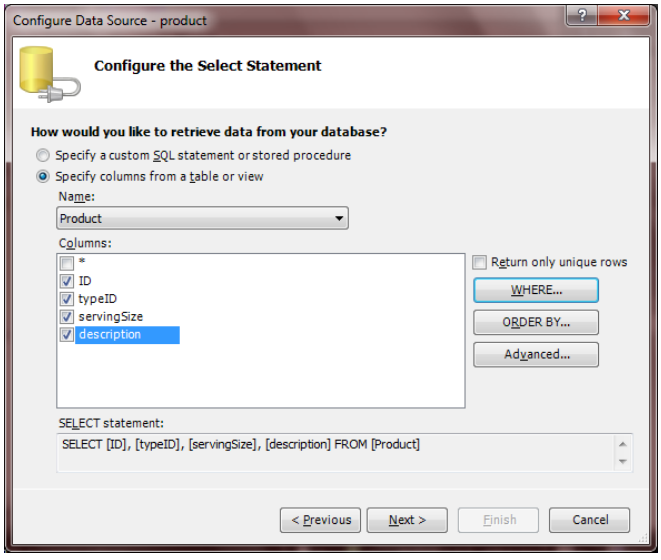


5. Now we need to add a new webpage “searchResult.aspx” under the same directory to process the search form. Create a new webpage called “searchResult.aspx” under the same folder and edit it.

6. On the new webpage “searchResult.aspx”, drag-and-drop a “GridView”, one of ASP.NET Controls under Data category, inside form#form1 on your webpage.

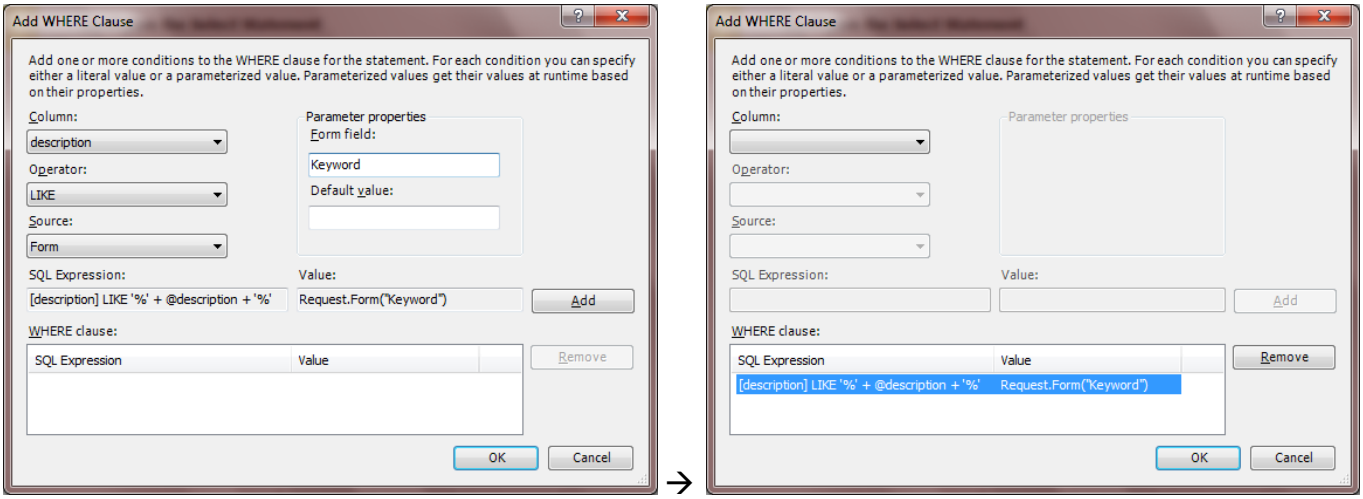


7. For the new GridView control, create a new data sources “Product”, use “demo” as the “data connection”, choose “Product” as the “Name: “, check “ID”, “typeID”, “servingSize”, and “description”, and click on “WHERE” button to add WHERE clause.

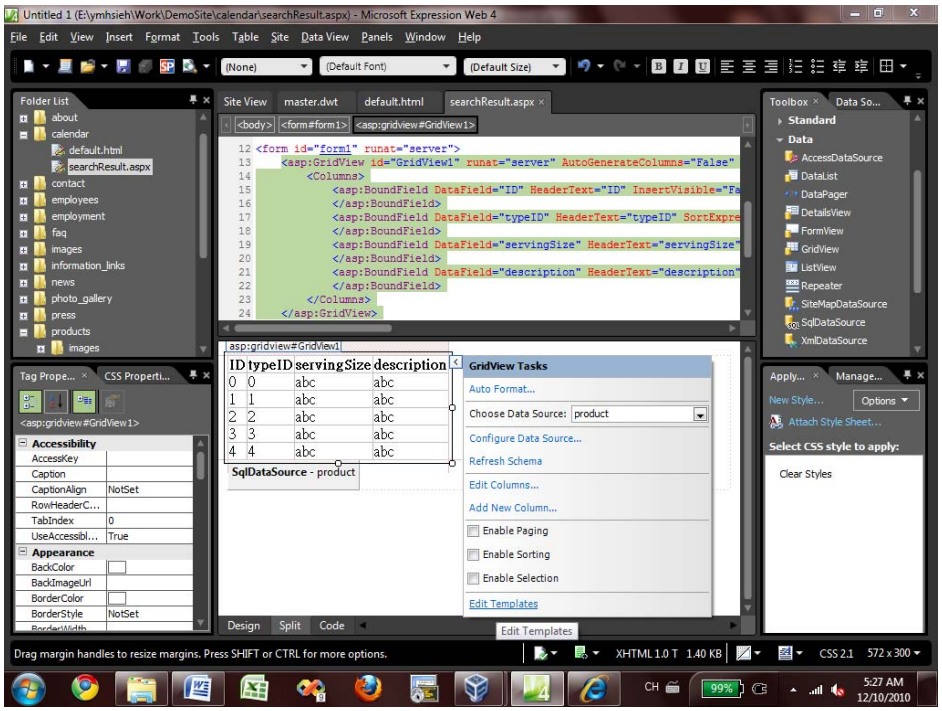


8. In the “Add WHERE Clause” dialog box, we input “description”, “LIKE”, “Form”, “Keyword” for corresponding inputs of “Columns”, “Operator”, “Source”, and “Form Field:”, respectively. Click on “Add” and “OK” to finish. The “SELECT statement at the bottom of the “Configure Data Source” dialog box should read like:

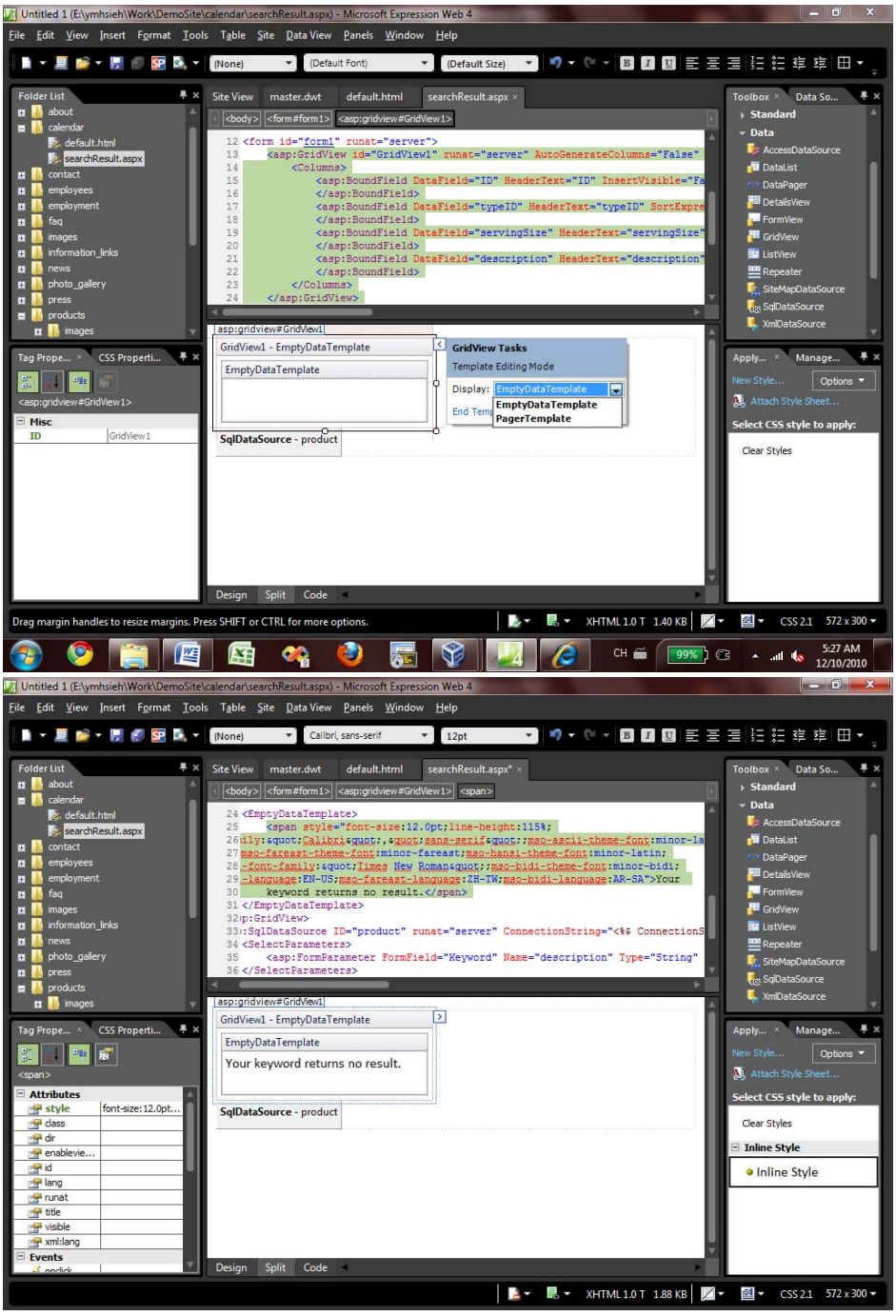
```
SELECT [ID], [typeID], [servingSize], [description]
FROM [Product]
WHERE ([description] LIKE '%' + @description + '%')
```



9. Now we have completed a webpage for searching products based on user-specified keyword. However, if user enters a keyword that really cannot find a product that contains the keyword, it will show an empty page. This is not good. So we would like to give it some message saying no results based on its given keyword. We can achieve that by editing the template for the GridView control. In GridView Tasks, choose “Edit Templates”.



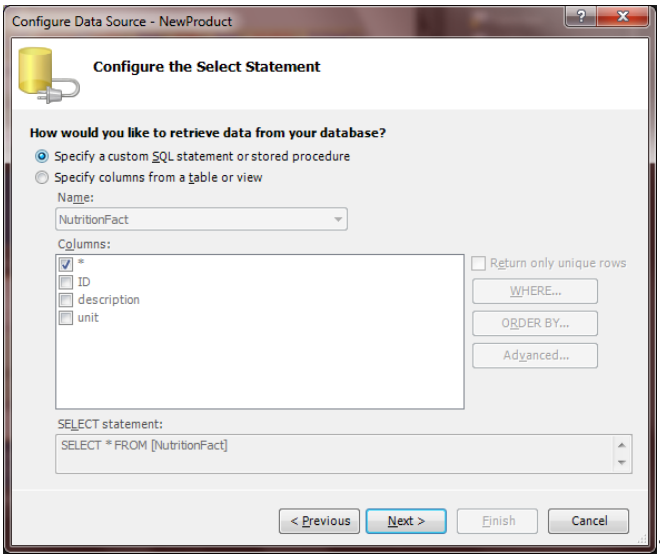
10. Choose the EmptyData template to edit, and enter text “Your keyword returns no result.” Save the webpage and try it out. ☺

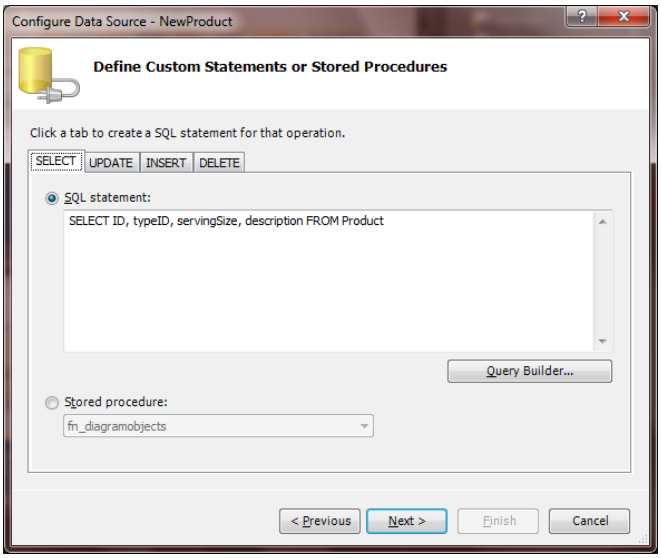


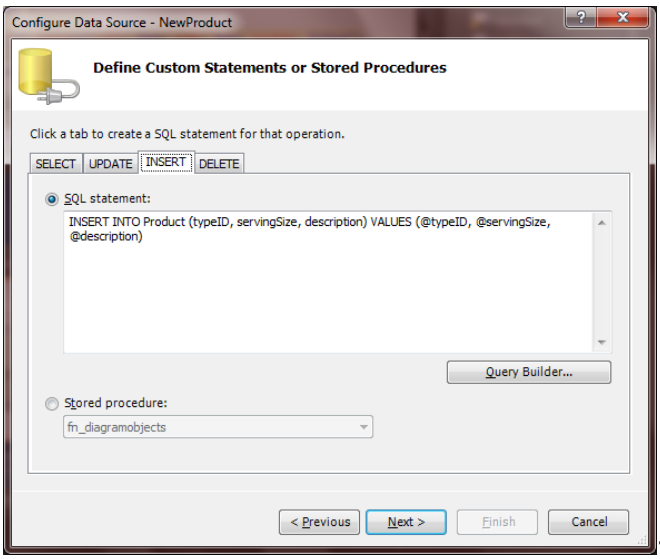
11. Now we have completed a webpage for searching products. Also, the searchResult.aspx looks unique comparing to other webpages, because it has not template attached to it. To attach the master template, use Menu “Format” → Dynamic Web Template → Attach Dynamic Web Template. After few more clicks, you are done.

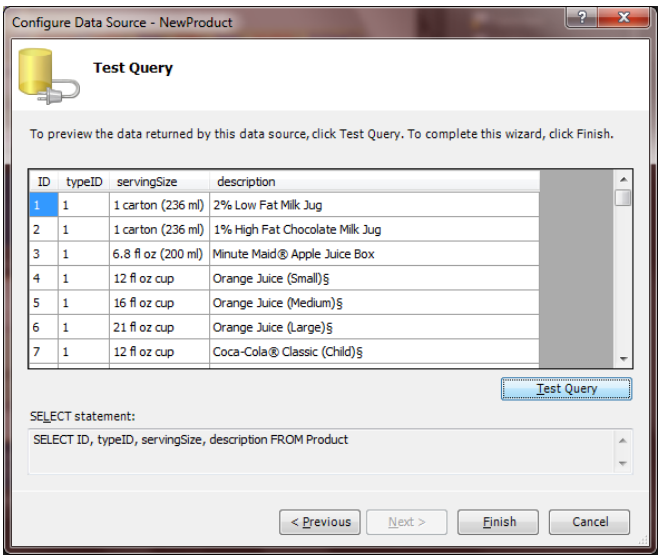
F. Add a new product

1. Adding products also need to add its nutrition facts, just like what we did in the VC#.
   The overall procedure looks like: 1) add a new product using form view, 2) add nutrition facts regarding the new product. We simply do these two functions separately.
2. On the new page, drag-and-drop a "FormView" control under Data of ASP.NET Controls. Create a new data source for it with ID NewProduct. For this data source, we use custom SQL statements instead of using specifying tables or views. and use the following two custom SQL statements for this data source:
   INSERT: `INSERT INTO Product (typeID, servingSize, description) VALUES (@typeID, @servingSize, @description)`
   SELECT: `SELECT ID, typeID, servingSize, description FROM Product`
   You can click "Test Query" before your finish configuring the data source to verify your select statement.

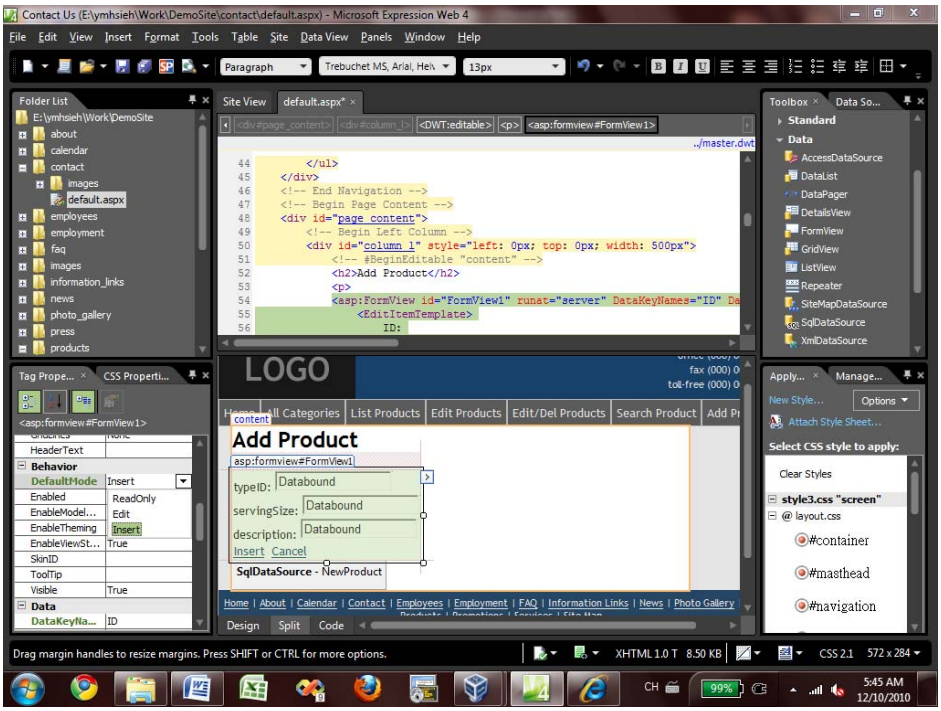




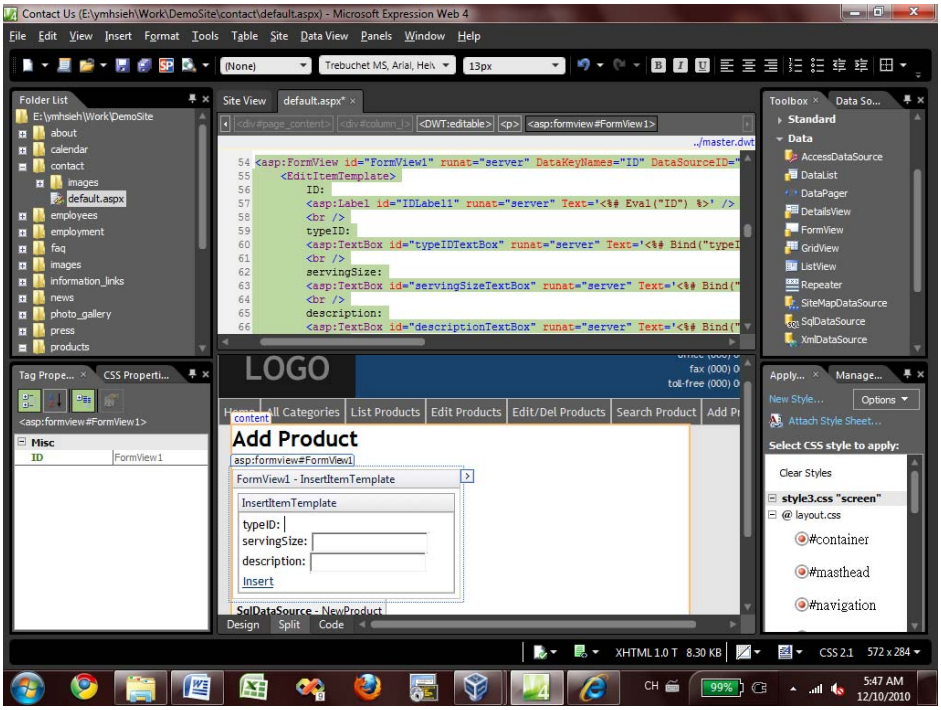




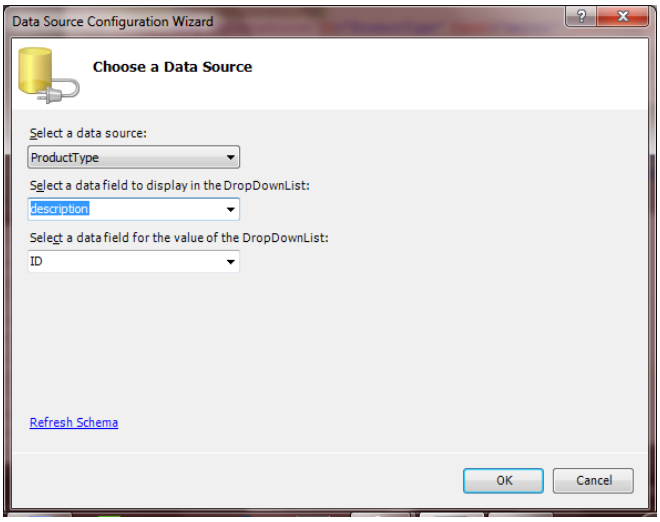
3. For the newly created FormView, change its property “DefaultMode” from Readonly to “Insert”

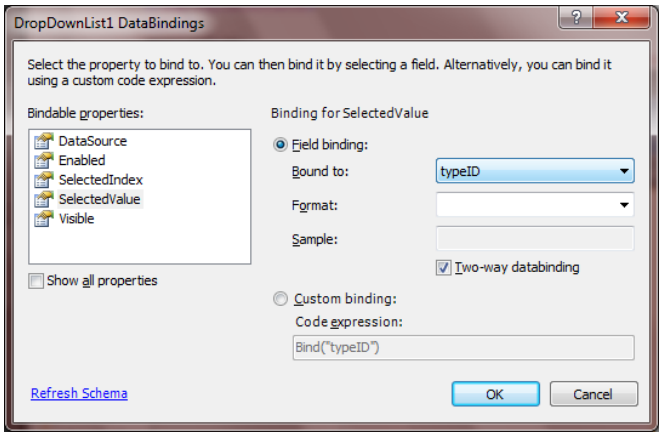


4. Now, edit the “InsertItemTemplate” template, then 1) Remove the TextBox for typeID, and 2) Remove the Cancel button.

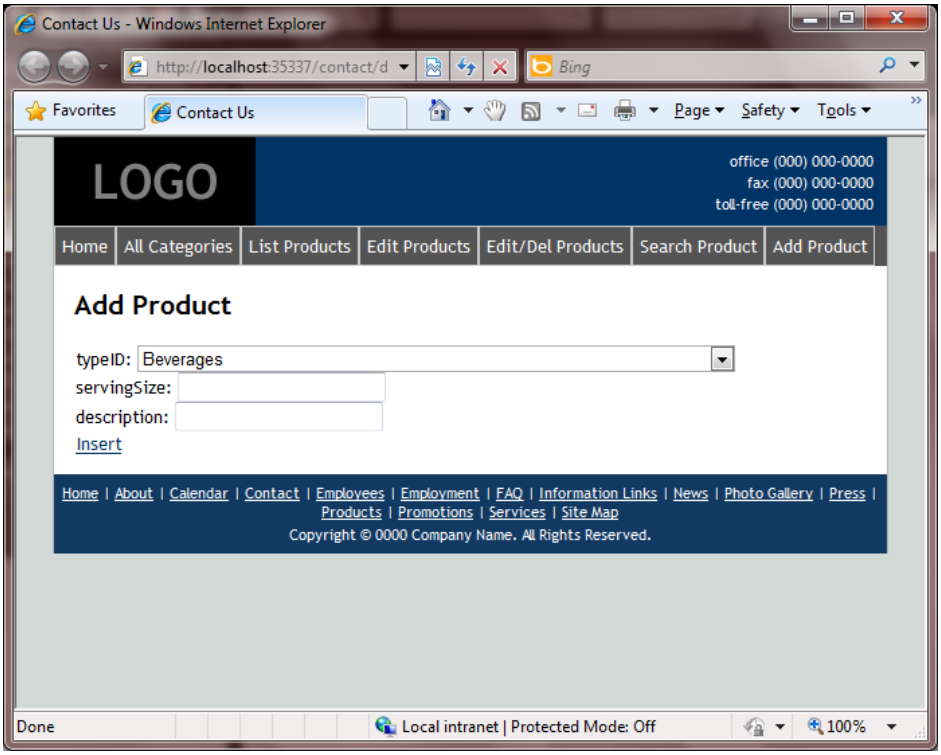


5. Drag-and-drop a DropDownList into the position after typeID and configure its DataSource properly as we did in the function Edit Product. Then, similarly, edit its DataBindings to bind it to type ID field. These procedures to identical to the one we did in the Edit Product.



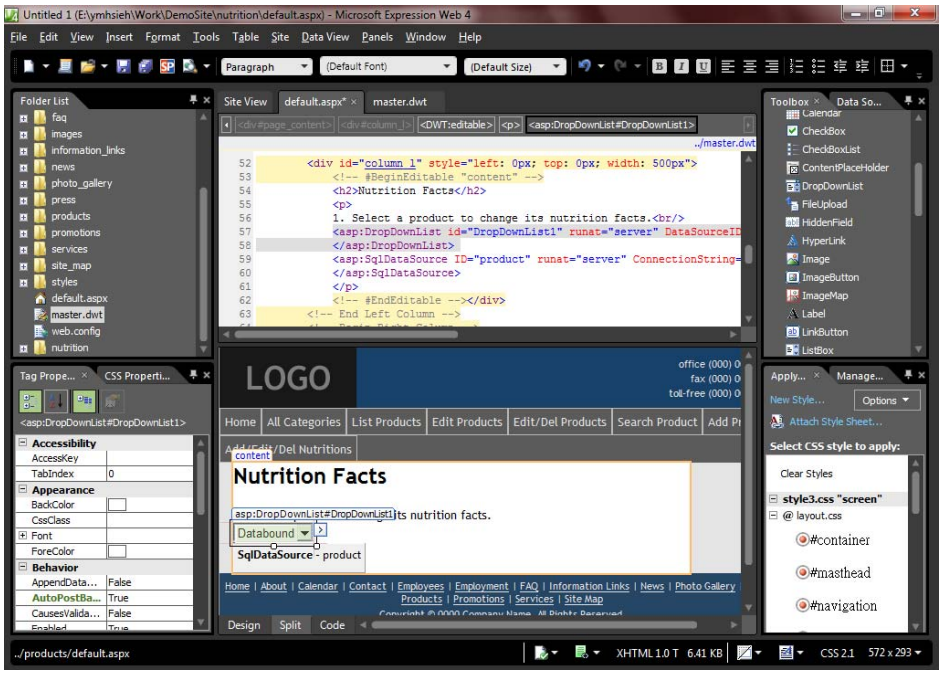


6. Now this page can be used to add products. However, we have not added nutrition facts. Let's create another page for adding nutrition facts.

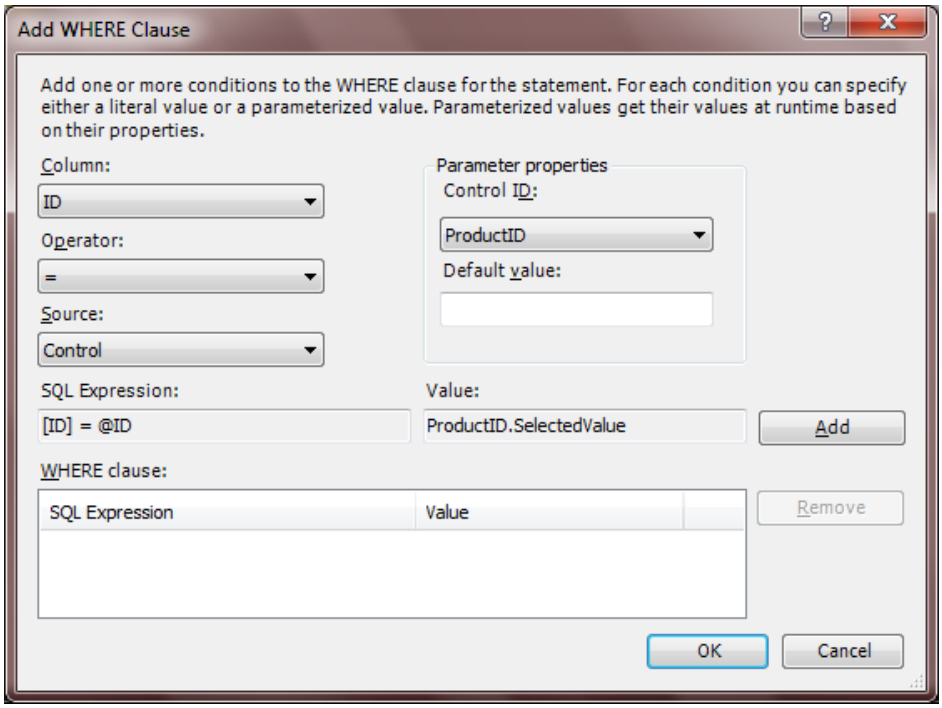


G. Add/Update/Delete nutrition facts about a product.

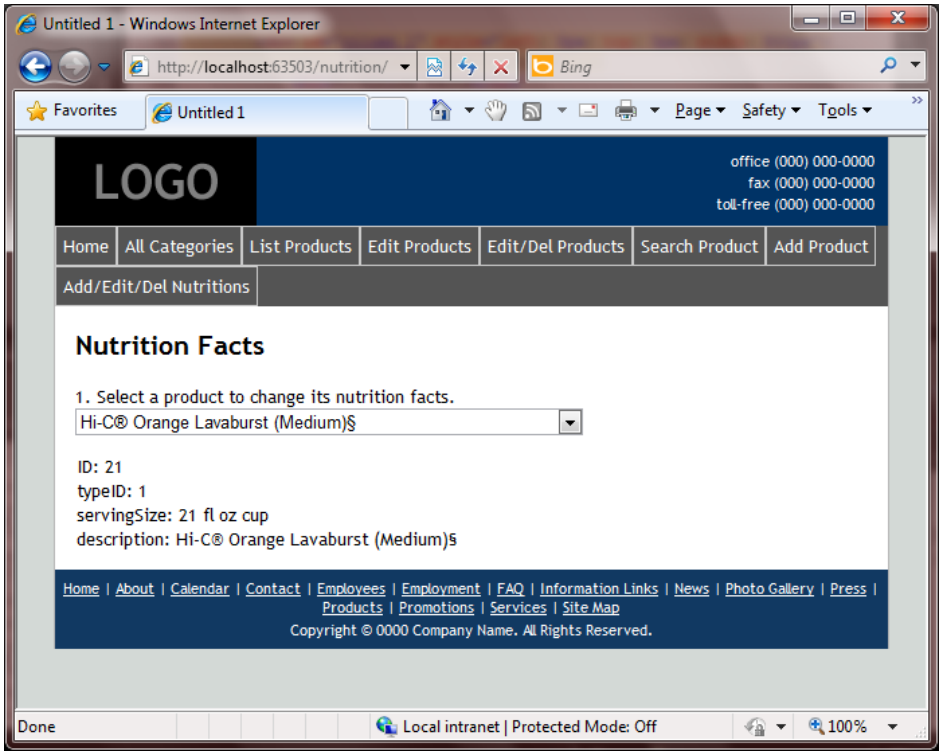
1. Create a new page for add/update/delete nutrition facts about a product. On this first page, we want to use a) DropDownList to select a product, use b) FormView to view details of this product, and a c) button to proceed to the next page for add/update/delete nutrition facts for the chosen product.
2. Drag-and-drop a DropDownList control onto the webpage, name it "**ProductID**". Choose "New Data Source" for the new DropDownList, choose "Product" to get its data, select columns "Product" and "ID", display "Product" field and select "ID" for value field. Make sure it has "AutoPostBack" enabled.



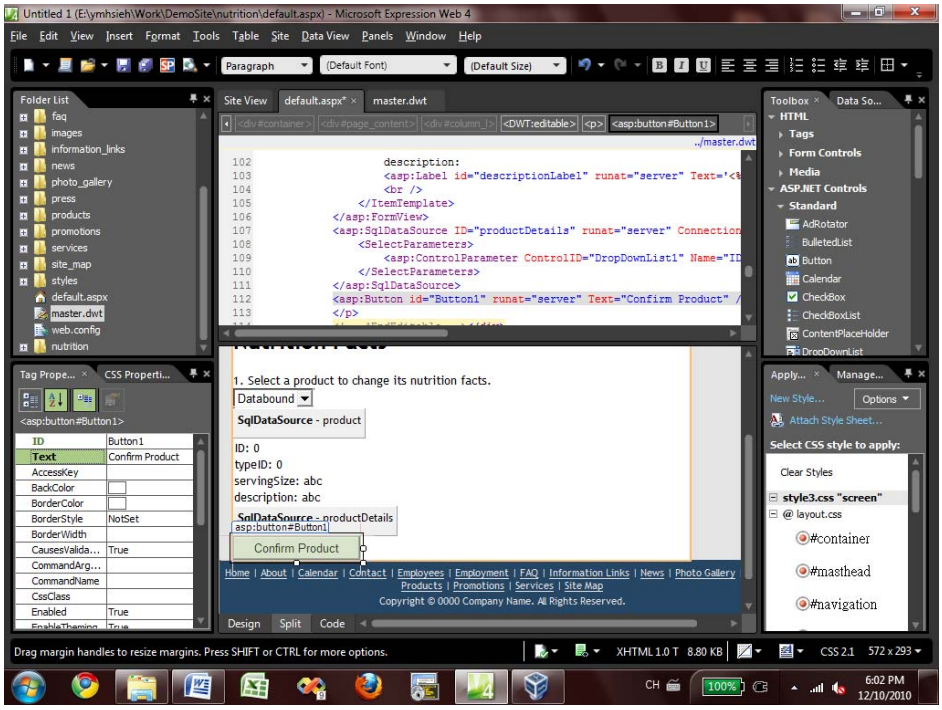
3. Add a FormView control (to show detailed information for the selected product) under DropDownList just added. Use another data source for it, use "Product" table to get data, check all four fields (ID, Product, ServingSize, and ProductType), add the WHERE part:



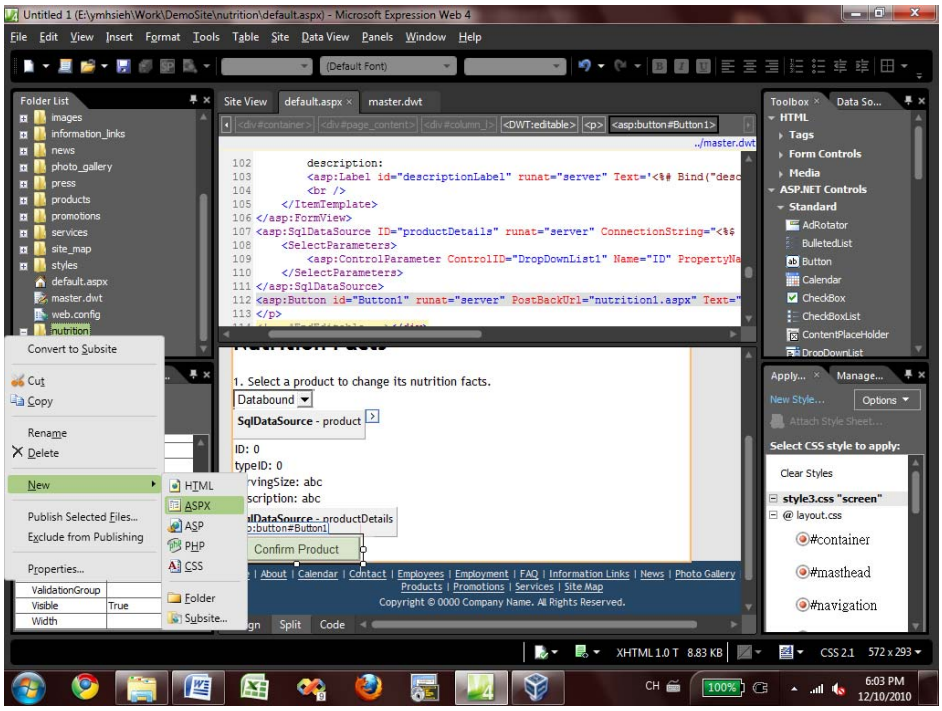
4. Now we preview this webpage, and it should look like:



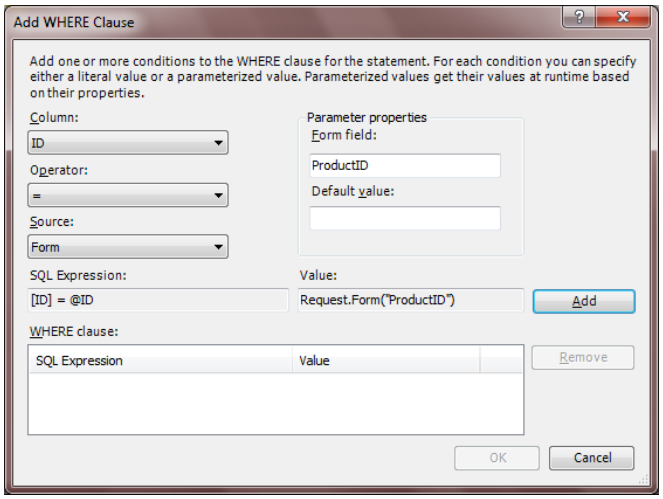
5. Then, we add a button to navigate to the next page for add/update/delete nutrition facts regarding the product. Find “Button” under ASP.NET→Standard, drag-and-drop it onto the webpage. Use its “Text” property to modify the text it displays on the webpage, and modify “PostBackURL” property to “nutrition1.aspx”, which is the webpage for handling add/update/delete nutrition facts regarding the selected product. We will add that page next.



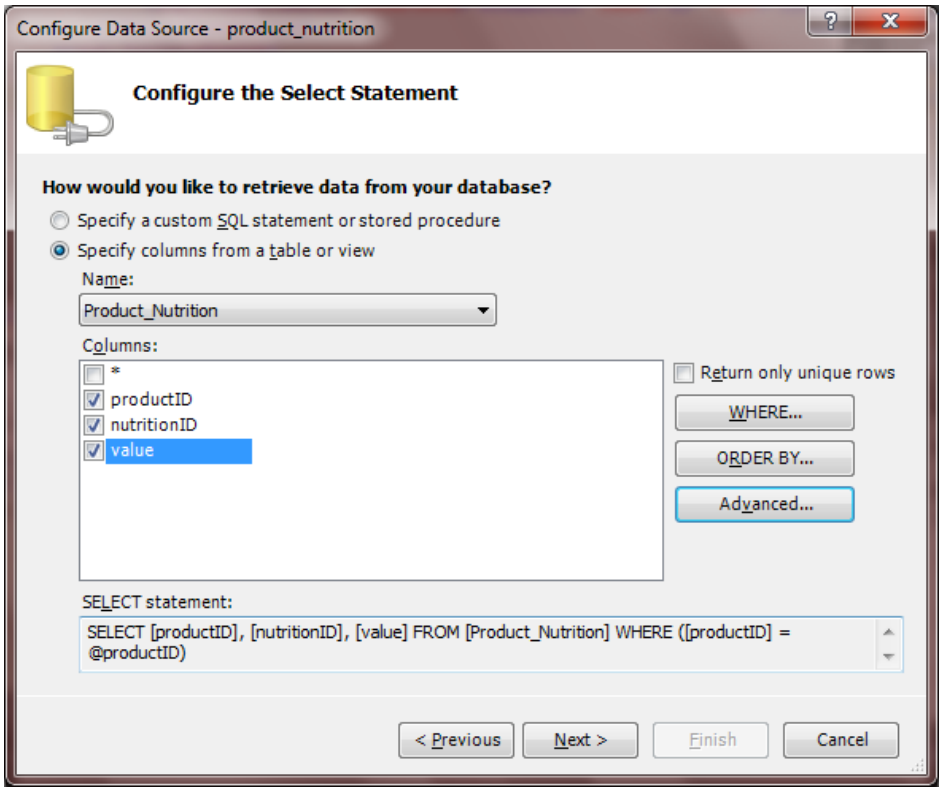
6. In Folder List, right click the folder that has the previous webpage and choose New → ASPX → nutrition1.aspx. Then double click on the new page to edit it.



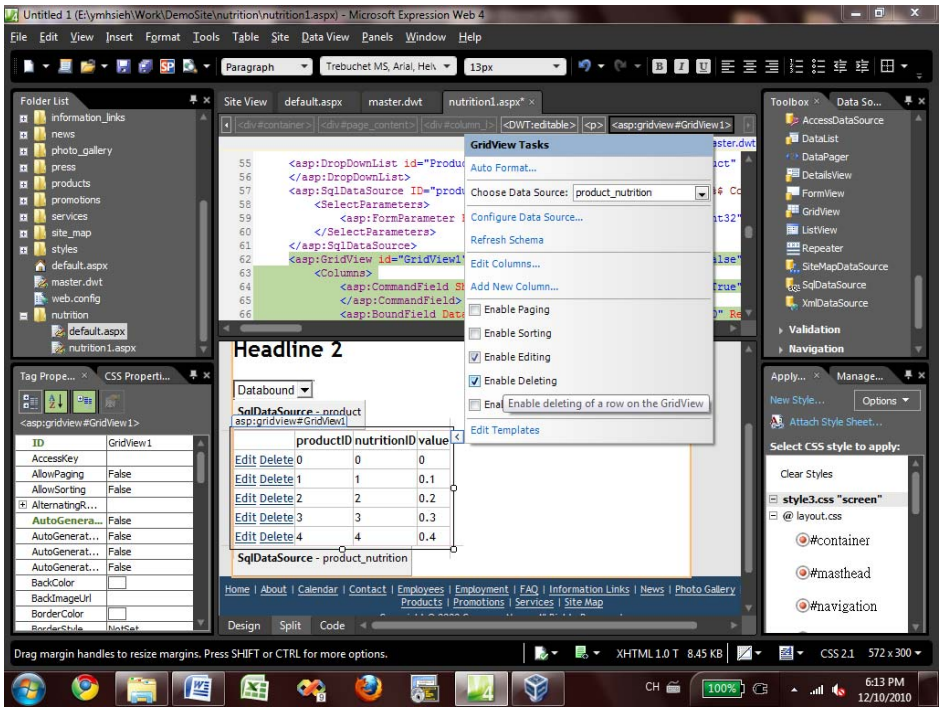
7. In Format → Dynamic Web Template → Attach Dynamic Web Template, choose “master.dwt” in the root folder of your web site in order to let the new webpage has consistent look-and-feel with the rest part of the website.
8. On the new webpage, our plan is to use a DropDownList to get the product selected in the previous webpage. Then we need a GridView to see/edit/delete the nutrition facts associated with the current product, and finally we need a FormView in order to add new nutrition facts.
9. On the new webpage, drag-and-drop a DropDownList onto the webpage. For this DropDownList, customize its data source similar to what we had before, select “ID” and “Product” field of the Product table, add the WHERE part. However, this time the source comes from “Form”, and the form field is “**ProductID**”, which is the name for the DropDownList on the previous webpage customized in Step 2. They need to match in order to get the selected product on the previous webpage. Also, name this DropDownList to be “ProductID” for easy identification.



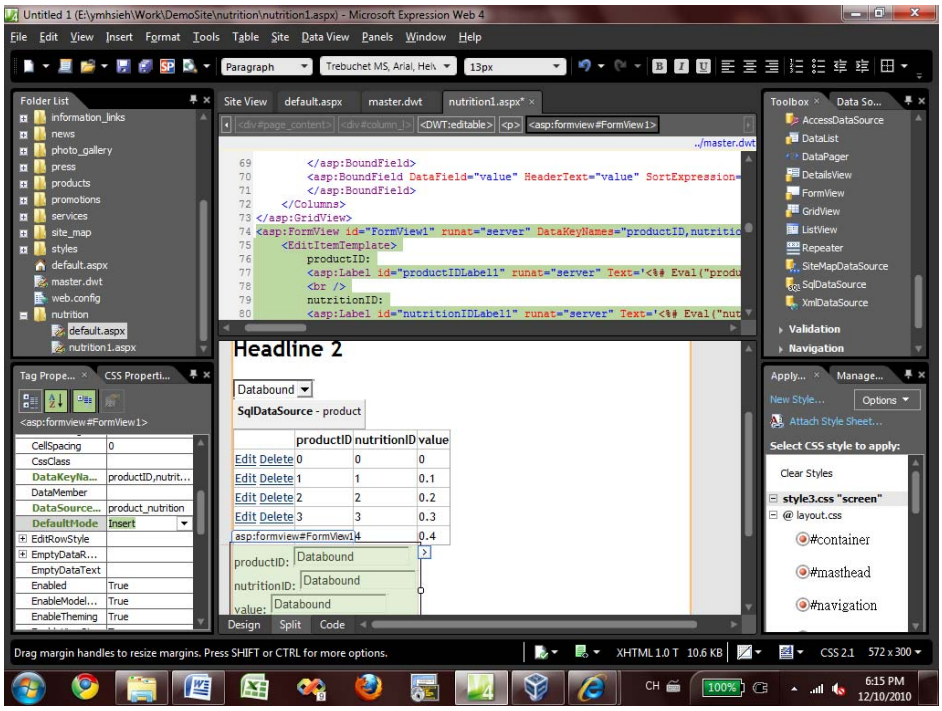
10. Now, drag-and-drop a GridView control on the webpage. The GridView has data source from Table "Product_Nutrition", and its WHERE part just like the previous step. Also, in "Advanced", generate INSERT, UPDATE, and DELETE statements.



11. For the GridView previously added on the webpage, Enable "Editing, Deleting". Then, using this webpage, we can update and delete nutrition fact for a given product.



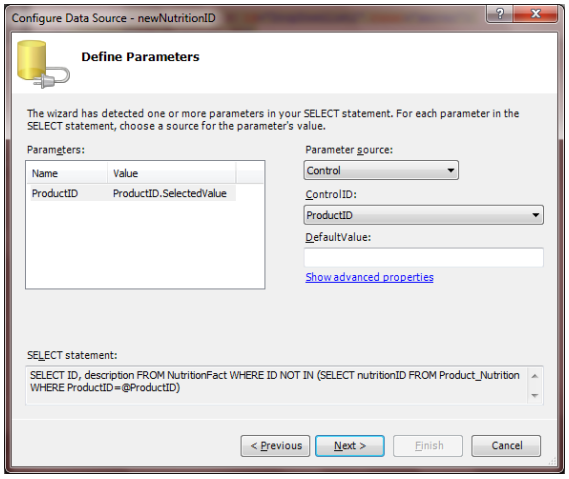
12. Now, add a FormView after the GridView. They can share the same data source (since they are updating the same table). Change its DefaultMode to “Insert”.

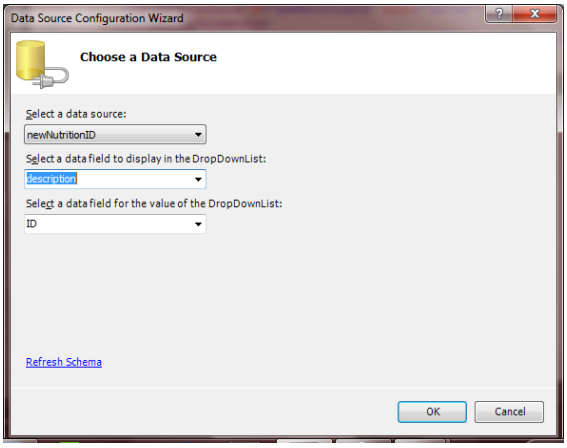


13. Edit its “InsertItemTemplate”, remove the “TextBox” after ProductID. Replace it with a DropDownList constructed using exactly the same way (and we add a new data source for this) as we did for the first DropDownList for this webpage. Remove the cancel button. Delete the TextBox after the “NutritionID”. Replace it with a DropDownList with a new DataSource with the following SQL:

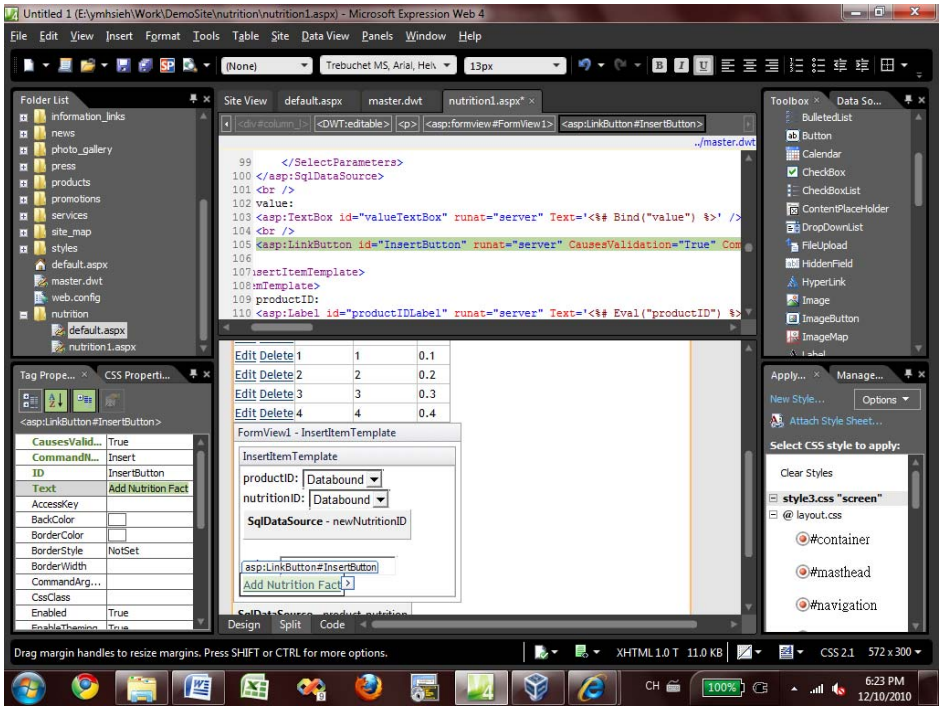
```
SELECT ID, description FROM NutritionFact
WHERE ID NOT IN
(SELECT nutritionID FROM Product_Nutrition WHERE ProductID=@ProductID).
```

You will be asked for the source of the parameter @ProductID





14. Rename “Insert” to be “Add Nutrition Fact”



15. Now we have completed the webpage for editing/deleting/adding new Nutrition Facts for a specified product.

Notes

Sometimes after you finished a webpage with data connectivity, and it fails to work with the following error:

Control 'GridView1' of type 'GridView' must be placed inside a form tag with runat=server.

This error seems to be caused by having links updated by expression web (e.g. you have changed some linked webpages in DWT). In this case, you can fix it by placing the following opening tag right after **<!-- #BeginEditable "content" -->**:

```
<form id="form1" runat="server">
```

And the following closing tag right before **<!-- #EndEditable "content" --> :**

```
</form>
```