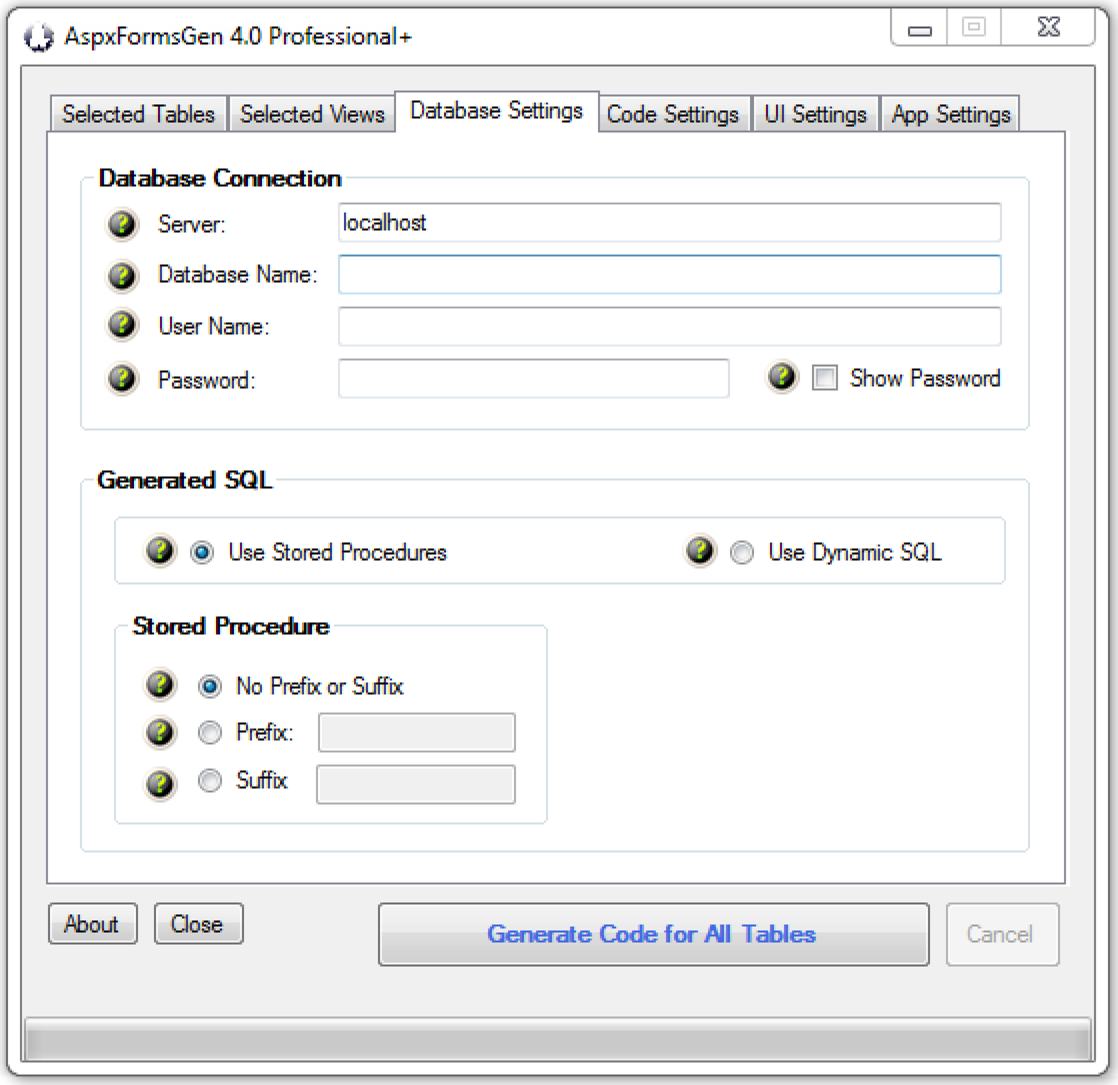
# Table of Contents

## Table of Contents

# Overview

AspxFormsGen 4.0 our Flagship product is finally here! AspxFormsGen 4.0 generates ASP.NET 4.0 Web Forms, Middle-Tier, Data-Tier, and Stored Procedures (or Dynamic SQL) in One Click. AspxFormsGen 4.0 generates databound ASP.NET 4.0 web forms (Professional Plus is databound, Express is not databound). AspxFormsGen 4.0 is a combination of our AspxFormsGen engine (generates ASP.NET web forms) and AspxCodeGen engine which generates Middle-Tier, Data-Tier, and Stored procedures or Dynamic SQL codes.

To keep AspxFormsGen 4.0 simple, there's only one main interface as shown in Figure 1. The main window consists of six (6) tabs.



Server: localhost
Database Name:
User Name:
Password:
[ ] Show Password

[✓ Use Stored Procedures] Use Dynamic SQL

[✓ No Prefix or Suffix]
Prefix:
Suffix

**Figure 1 Main Window**

# Tabs

## Selected Tables

AspxFormsGen generates code from all the tables in your database by default. You can choose to generate from selected tables only from the Code Settings tab, and then select just the tables to generate from on this tab. This is a **new feature** for AspxFormsGen 4.0.

- **Load Tables Button:** This button loads all the tables from the respective database into a check box list. Simply select the tables you want to generate code from by checking them. You need to select the *Selected Tables Only* option under the *Code Settings* tab, in the *Database to Generate From* group to enable this button. This button is disabled by default.

- **Clear Selection Button:** This button deselects all the selected/checked tables. When you select at least one table, this button will be enabled. This button is disabled by default.

## Selected Views [1]

You can choose to generate from selected views only from the Code Settings tab, and then select just the views to generate from on this tab. This is a **new feature** for AspxFormsGen 4.0.

- **Load Views Button:** This button loads all the views from the respective database into a check box list. Simply select the views you want to generate code from by checking them. You need to select the *Selected Views Only* option under the *Code Settings* tab, in the *Database to Generate From* group to enable this button. This button is disabled by default.

- **Clear Selection Button:** This button deselects all the selected/checked views. When you select at least one view, this button will be enabled. This button is disabled by default.

## Database Settings

This is where you enter the database you want to generate code from and whether you want to generate stored procedures or dynamic SQL. This is probably going to be your most used tab.

- **Database Connection**

    - **Server:** The name of the MS SQL Server where your database is located. E.g. localhost.
    - **Database Name:** The database you want to generate from. E.g. Northwind, AdventureWorks.
    - **User Name:** The user name you use to get access to your database. User user names that have administrator rights to your database. E.g. sa.
    - **Password:** The database password paired with the user name above. E.g. myPassword.
    - **Show Password:** Masks the password with an asterisk (*) when not checked. Shows the password in clear text when checked. **Note:** The password field is the only information that is not saved when you close the application if and when this Show Password is not checked, which is the default. Therefore you need to check this field if you want the application to remember the last password you entered every time you close the application.

- **Generated SQL** [1]
    - **Stored Procedures:** Generates stored procedures directly in the respective database. **Note:** If a stored procedure with the same name exists, it will be overwritten without warning.
        - **No Prefix or Suffix:** No prefix of suffix is added to the generated stored procedures. This option is selected by default.
        - **Prefix:** The prefix you want to add to the generated stored procedures. E.g. *myprefix_*StoredProcName.
        - **Suffix:** The suffix you want to add to the generated stored procedures. E.g. StoredProcName*_mySuffix*.
    - **Dynamic SQL:** Generates SQL script in class files. **Note:** All dynamic SQL classes are overwritten without warning. This is a **new feature** for AspxFormsGen 4.0.

## Code Settings

You'll find a selection here on where to generate your objects from: all tables, all views (**new**), selected tables (**new**), or selected views (**new**). This is also where you set the web site name, the root directory where you want the website to be generated, the namespace for your code, and most of all the language (either C# or VB.NET) you want the generated code to be in.

- **Database Objects to Generate From:** Here you can choose the database source from where to generate objects from. Each one of the options below will generate web forms, middle-tier classes, data-tier classes, and stored procedures or dynamic SQL.
    - **All Tables:** Generates objects for all tables in the respective database.
    - **All Views:** [1] Generates objects for all views in the respective database.
    - **Selected Tables Only:** Generates objects for selected tables only, in the respective database.
    - **Selected Views Only:** [1] Generates objects for selected views only, in the respective database.

- **Website:** The website that will be generated.
    - **Name:** Name of the website. This will be a folder. If this folder does not exist, it will be created in the directory below.
    - **Directory and Browse Button:** Root directory where you want the website to be generated in. You can use the browse button to choose the folder where you want the website to be generated in.

- **Business Layer and Data Layer:** The settings for the generated code.
    - **Namespace:** The root namespace that will be used in all generated code.
    - **Language:** The language all generated code will be in. You can choose either in C# or VB.NET.

## UI Settings [1]

You can customize your own settings for the generated ASP.NET web forms here. You can choose themes for the GridView control and JQuery UI controls. You can also choose the kind of page validation. You can also select which web forms to generate and the folders or file prefix to use for each web page. This is a **new feature** for AspxFormsGen 4.0.

- **Themes:** The themes used by certain controls in the generated user interface (ASP.NET web forms).
    - **GridView:** Theme used in the GridView web controls. To see a list of snapshots of the following themes, please visit our web site.
        - Slate
        - Colorful
        - Classic
        - Simple
        - Professional
        - Autumn
        - Oceanica
        - Brown Sugar
        - Sand & Sky
        - Rainy Day
        - Snow Pine
        - Lilacs In Mist
        - Black & Blue
        - Clover Field
        - Apple Orchard
        - Mocha
    - **JQuery UI:** Theme used in the JQuery UI controls used in the generated ASP.NET web forms. . To see a list of snapshots of the following themes, please visit our web site or JQuery UIs web site. JQuery UI is a free plug-in.
        - BlackTie
        - Blitzer
        - Cupertino
        - Dark-Hive
        - Dot-Luv
        - Eggplant
        - Excite-Bike
        - Hot-Sneaks
        - Humanity
        - Le-Frog
        - Mint-Choc
        - Overcast
        - Pepper-Grinder
        - Redmond
        - Smoothness
        - South-Street
        - Start
        - Sunny
        - Swanky-Purse
        - Trontastic
        - UI-Darkness
        - UI-Lightness
        - Vader

- **Validation:** The type of validation used in the generated ASP.NET web forms.
  - **JQuery Validation:** JQuery validation is a free client-side validation plug-in.
  - **ASP.NET Validation:** Uses ASP.NET validation controls such as the RequiredFieldValidator, and CompareValidator.
- **Organize Web Forms:** Organizes the generated web forms into their respective folders when checked. Puts all the generated web forms in the root directory of the generated web site when unchecked. A prefix is added to the respective web form type when the *Organize Web Form* is unchecked. You will notice that the group name toggles from *Folder Organization* to *Web Form Prefix* when you check and uncheck this field.
- **Web Forms to Generate and Folder Organization/Web Form Prefix**: These two groups are related. You will notice that as you toggle checking and unchecking the *Web Forms to Generate*, the related *Folder Organization* or *Web Form Prefix* is enabled and disabled respectively.

  You can choose the type of web forms you want to generate for *All Tables* or for *Selected Tables Only* (under the *Code Settings* tab) by checking the respective *Web Forms to Generate* item. Only one type of web form is generated when you choose *All Views* or *Selected Views Only* (under the *Code Settings* tab), therefore you don't have the flexibility of choosing the web forms to generate.

  The respective *Folder Organization* item or *Web Form Prefix* item is **required** and must be **unique**.

  - **GridView with Add, Edit Redirect & Delete:**
    - Contains a GridView Server Control that has CRUD (Create, Retrieve, Update, Delete) funtionality.
    - Adding a new record redirects to another page
    - Updating and existing record redirects to another page
    - Delete funtionality uses a JQuery UI Pop-up for delete confirmation
    - A link to a read-only Web Form is also provided for all Foreign Key columns (for details on the foreign key)
    - GridView uses a Sort Direction Image in the header
    - GridView uses Numeric Paging in the footer
    - One ASP.NET 4.0 Web Form is generated per table
  - **Add New & Edit Record:**
    - Contains JQuery Validation or ASP.Net Validation
    - Contains JQuery UI Date Controls for date fields
    - Contains bound DropDownList Web Control for foreign fields
    - You can get here from the "GridView with Add, Edit Redirect, & Delete" Web Form when clicking the Add New Record link or the Edit button
    - One ASP.NET 4.0 Web Form is generated per table
  - **Record Details (Read Only) :**
    - Shows details of a record (Read-Only)

    - You can get here from the "GridView with Add, Edit Redirect, & Delete" Web Form when clicking the foreign key links
    - One ASP.NET 4.0 Web Form is generated per table

- **GridView, Read-Only:**
    - Contains a GridView Server Control. No CRUD funtionality (read-only).
    - A JQuery Tooltip pop-up link is provided for all Foreign Key columns (for details on the foreign key)
    - GridView uses a Sort Direction Image in the header
    - GridView uses Numeric Paging in the footer
    - One ASP.NET 4.0 Web Form is generated per table

- **GridView with Add, Edit, & Delete (Same Page) :**
    - Contains a GridView Server Control that has CRUD (Create, Retrieve, Update, Delete) funtionality.
    - Add a new record on the same page with JQuery animation
    - Update an existing record on the same page with JQuery animation
    - Delete funtionality uses a JQuery UI Pop-up for delete confirmation
    - A JQuery Tooltip pop-up link is provided for all Foreign Key columns (for details on the foreign key)
    - GridView uses a Sort Direction Image in the header
    - GridView uses Numeric Paging in the footer
    - One ASP.NET 4.0 Web Form is generated per table

- **GridView within an Accordion (Grouping) :**
    - Contains a JQuery UI Accordion control with GridView within. No CRUD funtionality (read-only).
    - Shows grouping by the respective group
    - Shows count per respective group
    - E.g. Orders by Shipper, Territories by Region
    - One ASP.NET 4.0 Web Form is generated for each table referencing the current table

- **GridView Filtered By a Drop Down List:**
    - Contains a GridView Server Control. No CRUD funtionality (read-only).
    - Contains a DropDownList Control that filters the GridView's data on index change
    - GridView uses a Sort Direction Image in the header
    - GridView uses Numeric Paging in the footer
    - A JQuery Tooltip pop-up link is provided for all Foreign Key columns (for details on the foreign key)
    - One ASP.NET 4.0 Web Form is generated for each foreign key in each table

- **GridView with Totals:**
    - Contains a GridView Server Control. No CRUD funtionality (read-only).
    - Shows total number of records
    - Shows totals on the footer for money fields
    - GridView uses a Sort Direction Image in the header
    - GridView uses Numeric Paging in the footer

        - A JQuery Tooltip pop-up link is provided for all Foreign Key columns (for details on the foreign key)
        - One ASP.NET 4.0 Web Form is generated for tables that have money data fields

    - **GridView, More Information:**
        - Contains a GridView Server Control. No CRUD funtionality (read-only).
        - Each row can be viewed for more information on click of the respective button (animated)
        - GridView uses a Sort Direction Image in the header
        - GridView uses Numeric Paging in the footer
        - A JQuery Tooltip pop-up link is provided for all Foreign Key columns (for details on the foreign key)
        - One ASP.NET 4.0 Web Form is generated per table

    - **Unbound Web Form:**
        - Web Forms that are not bound to the database
        - Contains JQuery Validation or ASP.Net Validation
        - Contains JQuery UI Date Controls for date fields
        - Contains unbound DropDownList Web Control for foreign fields
        - One ASP.NET 4.0 Web Form is generated per table

## App Settings

These are application settings. Almost all generated code/web forms are overwritten every time you use AspxFormsGen 4.0. However, you can choose not to overwrite some key files from here. You can also reset all settings to its original default from here. This is a **new feature** for AspxFormsGen 4.0.

- **Overwrite Files:** [1] These files are overwritten by default (checked by default). However, if you want to add your own code to the following files, you can choose not to overwrite them by unchecking the respective file.

    - **Overwrite Master Page:** Overwrites the *MasterPage.master* file when checked. The generated master page is empty and is used by all the generated ASP.NET web forms. You can add you web site design here, but make sure to uncheck this setting when you do.

    - **Overwrite Dbase File:** Overwrites the *Dbase.cs* (or *Dbase.vb*) helper class when checked. The *Dbase* helper class contains the *Database Connection* settings you provided under the *Database Settings* tab. It contains static/shared helper methods that connect to the database. The helper methods are called by the Data Layer Base code. This file can be found in the *App_Code* folder under the *Helper* directory.

      **Recommendation:** Rather than keep the connection string in text form under the *GetConnection()* method, move it to the Web.config file to an app setting tag, then reference that configuration from the *GetConnection()* method. Make sure to uncheck this setting if you're planning to add your own code to it.

    - **Overwrite Web.config File:** Overwrites the *Web.config* file when checked. Make sure to uncheck this setting if you're planning to add configuration code to it.

    - **Overwrite Functions File:** Overwrites the *Functions.cs* (or *Functions.vb*) helper class when checked. The *Functions* helper file contains static/shared methods used by the GridView web control. Make sure to uncheck this setting if you're planning to add your own code to it. This file can be found in the *App_Code* folder under the *Helper* directory.

    - **Overwrite Global.css:** Overwrites the *Global.css* stylesheet file when checked. This stylesheet is used by all generated web forms and is referenced by the Master Page file. You can add additional styles here, just sure to uncheck this setting if you plan to add your own code to it. This file can be found in the *Styles* folder.

    - **Overwrite SkinFile.skin:** Overwrites the *SkinFile.skin* skin file when checked. It is under the *App_Themes* folder, in the *Theme1* theme. *Theme1* is the theme used by all generated web forms and is referenced in the *Web.config* file. Make sure to uncheck this setting if you're planning to add your own skins to it.

- **Automatically Open Selected Tables or Selected Views tab:** When you choose *Selected Tables Only* or *Selected Views Only* under the *Code Settings* tab, the *Selected Tables* or *Selected Views* tab is automatically opened respectively when this setting is checked. If you don't want to be automatically redirected to the respective tab, uncheck this setting.

- **App Files Directory:** The directory where the *AppFiles* folder was installed. This *AppFiles* folder contains miscellaneous files AspxFormsGen 4.0 copies to the generated web site during generation. Although for most parts you don't have to ever change this setting, in the event that you have a very unique folder/file structure in the computer where you installed AspxFormsGen 4.0, you can always correct the location of the *AppFiles* folder using this setting.

- **Restore All Settings to Default:** AspxFormsGen remembers the last settings you used when you close the application, this is the reason why after your first use, and you can keep generating code for the same database with **One Click!** If you want to reset all the settings to the original (default) values, simply click this button. A message asking for confirmation of the restore will pop up, click *Yes* to restore settings to default, otherwise *No*.

# Buttons (Outside Tabs)

These are buttons you can readily access anywhere in the application. You don't have to be in a specific tab to access these buttons.

- **Generate... Button:** This is the most important button in the application. You click this to generate code. You'll notice that the text on this button changes to the respective operation and it toggles from enabled to disabled when you change your selection under the *Code Settings* tab, in the *Database Object to Generate From*. Listed below are the events that trigger when the selection in the *Database Object to Generate From* changes.

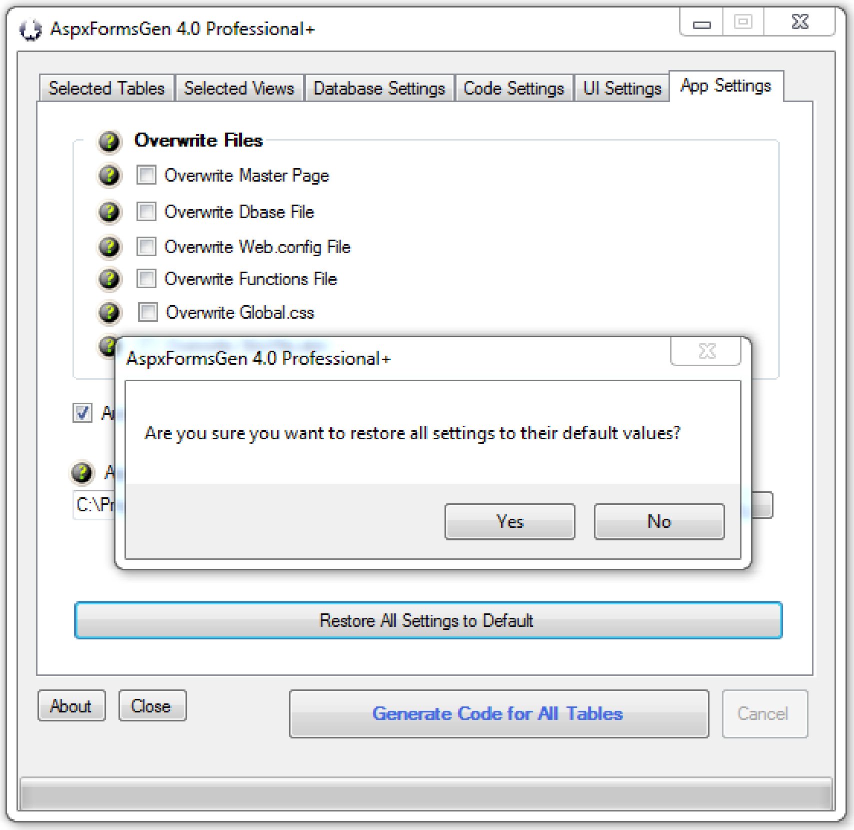
    - **All Tables:** Text changes to "*Generate Code for All Tables*". Button state is always enabled.

- **All Views:** [1] Text changes to "*Generate Code for All Views*". Button state is always enabled.
- **Selected Tables Only:** Text changes to "*Generate Code for Selected Tables Only*". Button state is disabled when there is no selected table under the *Selected Tables* tab, otherwise it's enabled.
- **Selected Views Only:** [1] Text changes to "*Generate Code for Selected Views Only*". Button state is disabled when there is no selected view under the *Selected Views* tab, otherwise it's enabled.

- **Cancel Button:** This button cancels code generation. It is disabled by default. It is only enabled once you click the *Generate…* button or once code generation is started.
- **About Button:** Shows information about AspxFormsGen 4.0.
- **Close Button:** Closes the application.

# Generating ASP.NET 4.0 Web Forms and Code

You have four options under the *Code Settings* tab, in the Database Objects to Generated From. These options affect the behavior of other settings in this application. This part of the document shows you how to generate code using each of these options. To start any tutorial, make sure to *Restore All Settings to Default* button, under the App Settings Tab. The following tutorials use Microsoft's Northwind database, Visual Studio 2010, and MS SQL 2008. See Figure 2.



Are you sure you want to restore all settings to their default values?

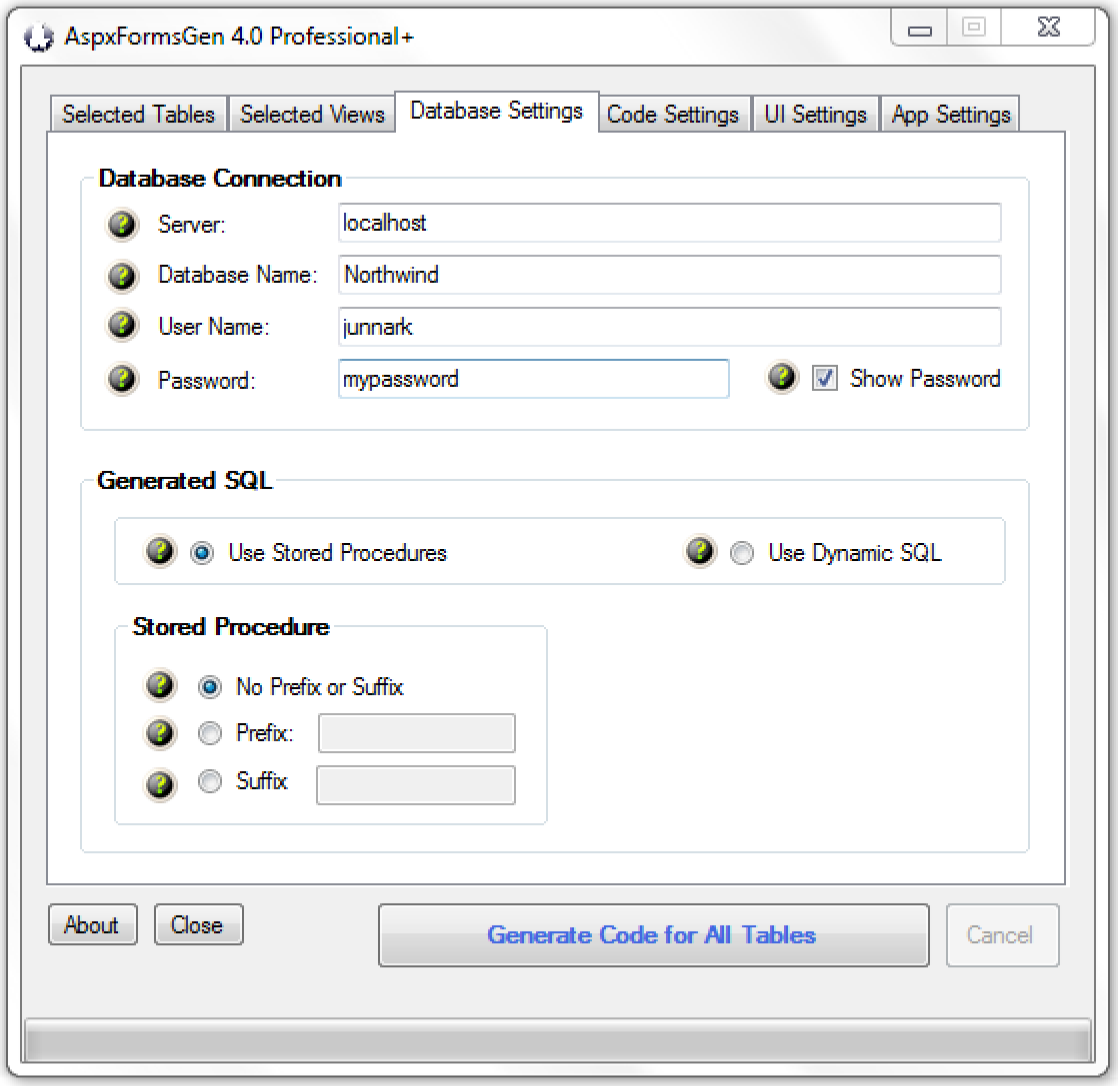
**Figure 2 Restore All Settings To Defaults**

## For All Tables

The *All Tables* option generates objects for all tables in the respective database.

1. Go to the *Database Settings* tab you will notice that *All Tables* under the *Database Objects to Generate From* is selected. This option is selected by default. Fill out all the required fields as shown below. Make sure to use your own *User Name* and *Password*. Also make sure to check *Show Password*, this will make the application remember this setting when we close the application.

   **One Click Feature:** Because AspxFormsGen 4.0 remembers your settings, the next time you open AspxFormsGen 4.0, all you have to do is click the *Generate…* button, this has always been our signature feature. See Figure 3.



AspxFormsGen 4.0 Professional+

Selected Tables | Selected Views | Database Settings | Code Settings | UI Settings | App Settings

Database Connection

Server: localhost

Database Name: Northwind

User Name: junnark

Password: mypassword [✓ Show Password]

Generated SQL

[✓ Use Stored Procedures] Use Dynamic SQL

Stored Procedure

[✓ No Prefix or Suffix]

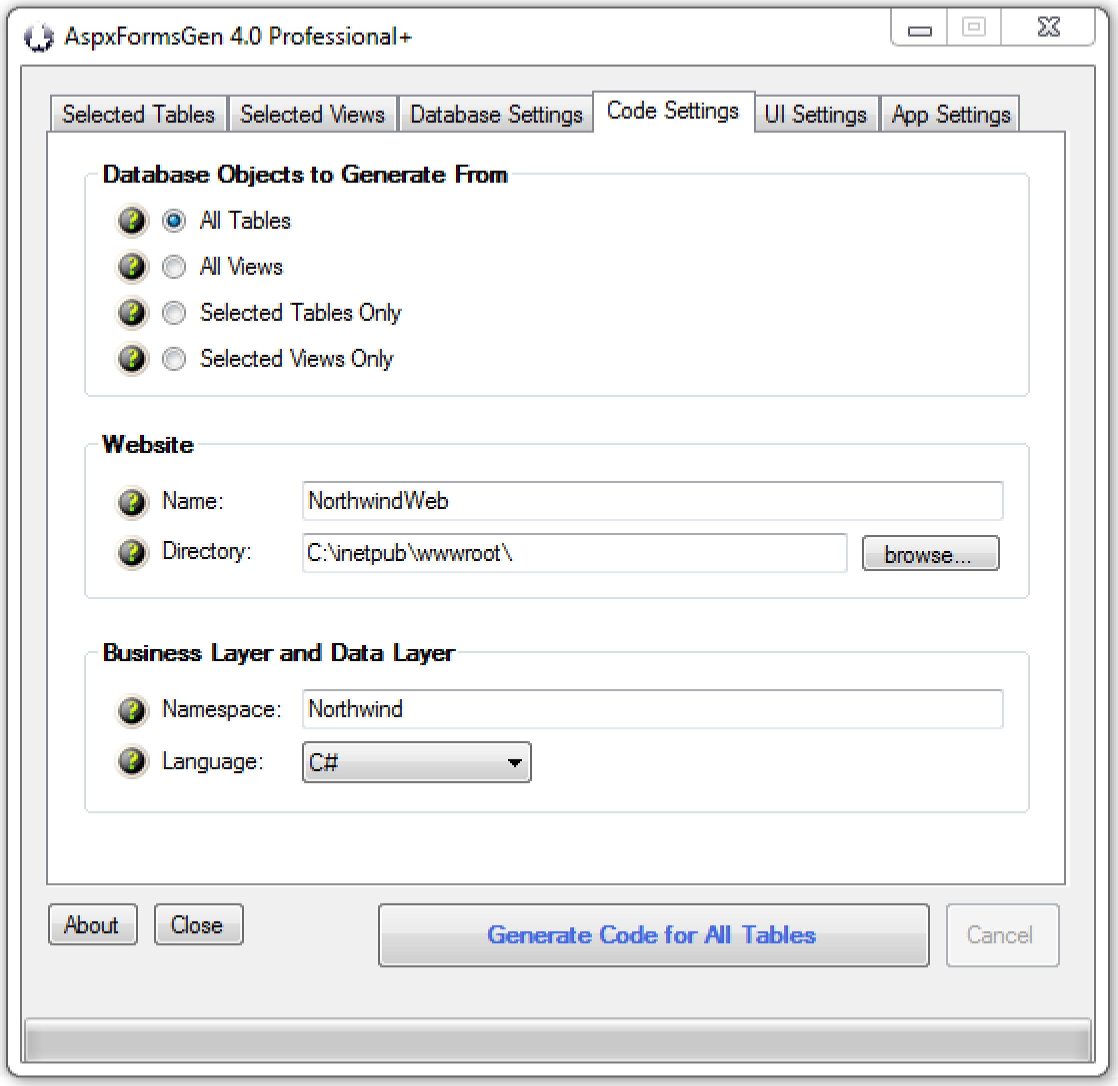
Prefix:

Suffix

**Figure 3 Database Settings**

2. Now go to the *Selected Tables* and *Selected Views* [1] tabs. You will notice that everything in these tabs is disabled. As you may already know, these tabs are dedicated for use by the *Selected Tables* and *Selected Views* options respectively; this is why they're disabled.

3. Go to the Code Settings tab and fill-out the rest of the required fields as shown in Figure 4.



Database Objects to Generate From
- (●) All Tables
- ( ) All Views
- ( ) Selected Tables Only
- ( ) Selected Views Only

Website

| | |
|---|---|
| Name: | NorthwindWeb |
| Directory: | C:\inetpub\wwwroot\ |

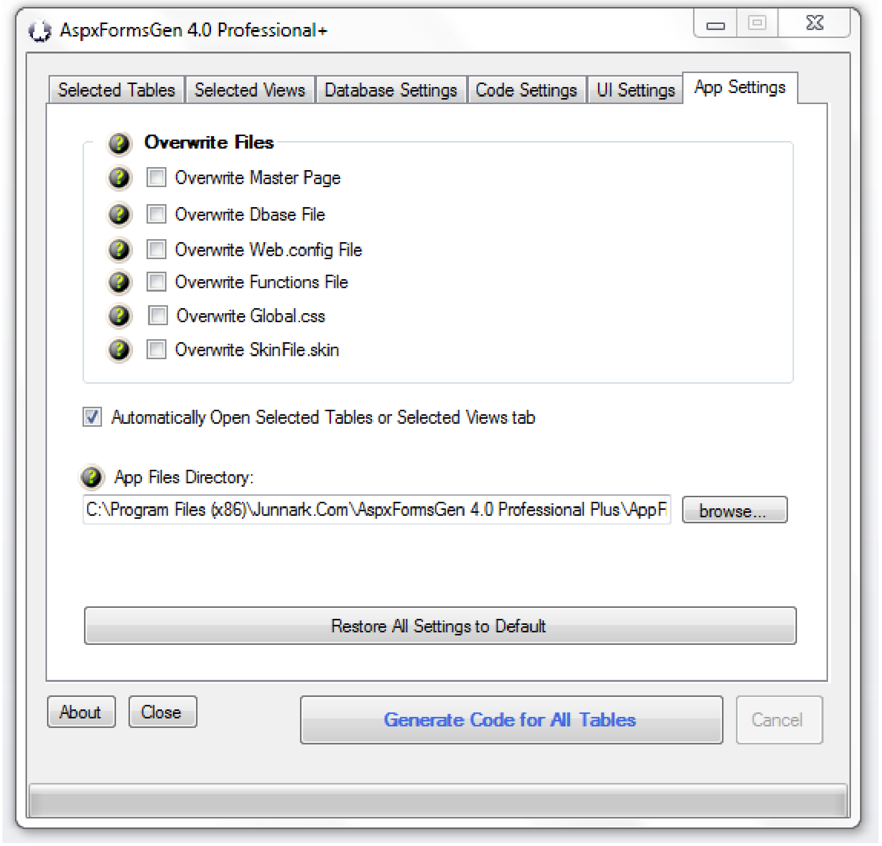
Business Layer and Data Layer

| | |
|---|---|
| Namespace: | Northwind |
| Language: | C# |

**Figure 4 Code Settings - All Tables**

4. Go to the *UI Settings* [1] tab and view the settings. We'll accept all these settings and do nothing on this tab.

5. Now, go to the *App Settings* tab and uncheck everything under the *Overwrite Files* [1] group. AspxFormsGen 4.0 will write these files once if they don't already exist. See Figure 4.



Overwrite Files

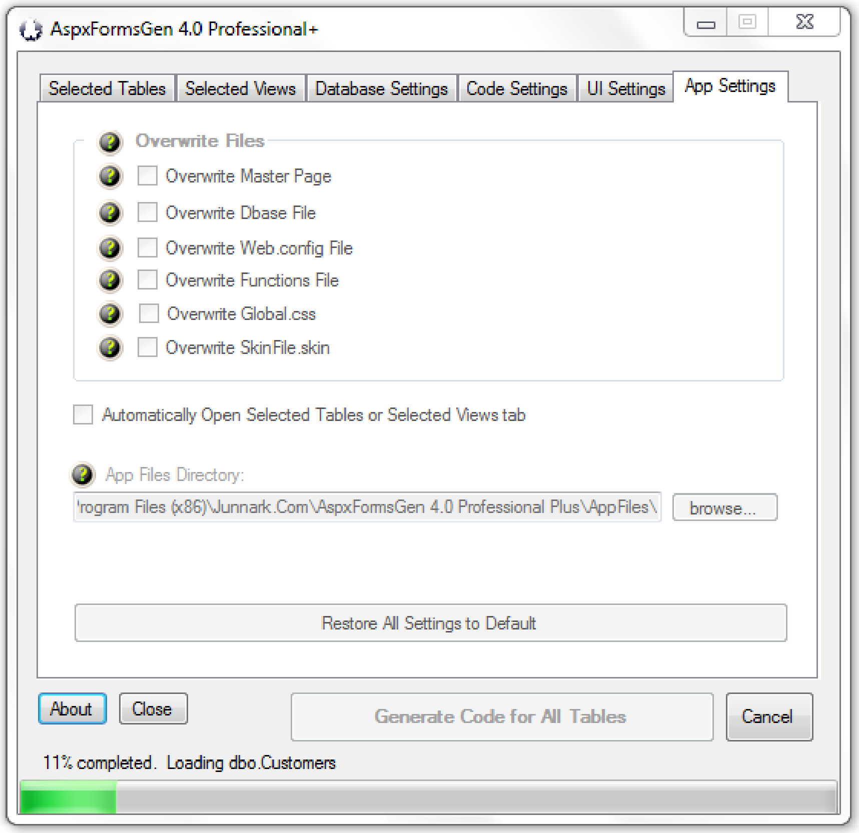
- [ ] Overwrite Master Page
- [ ] Overwrite Dbase File
- [ ] Overwrite Web.config File
- [ ] Overwrite Functions File
- [ ] Overwrite Global.css
- [ ] Overwrite SkinFile.skin

[✓] Automatically Open Selected Tables or Selected Views tab

App Files Directory: C:\Program Files (x86)\Junnark.Com\AspxFormsGen 4.0 Professional Plus\AppF

**Figure 4 Uncheck All Overwrite Files**

6. Click the *Generate Code for All Tables* button. See Figure 5.



Overwrite Files

- [ ] Overwrite Master Page
- [ ] Overwrite Dbase File
- [ ] Overwrite Web.config File
- [ ] Overwrite Functions File
- [ ] Overwrite Global.css
- [ ] Overwrite SkinFile.skin

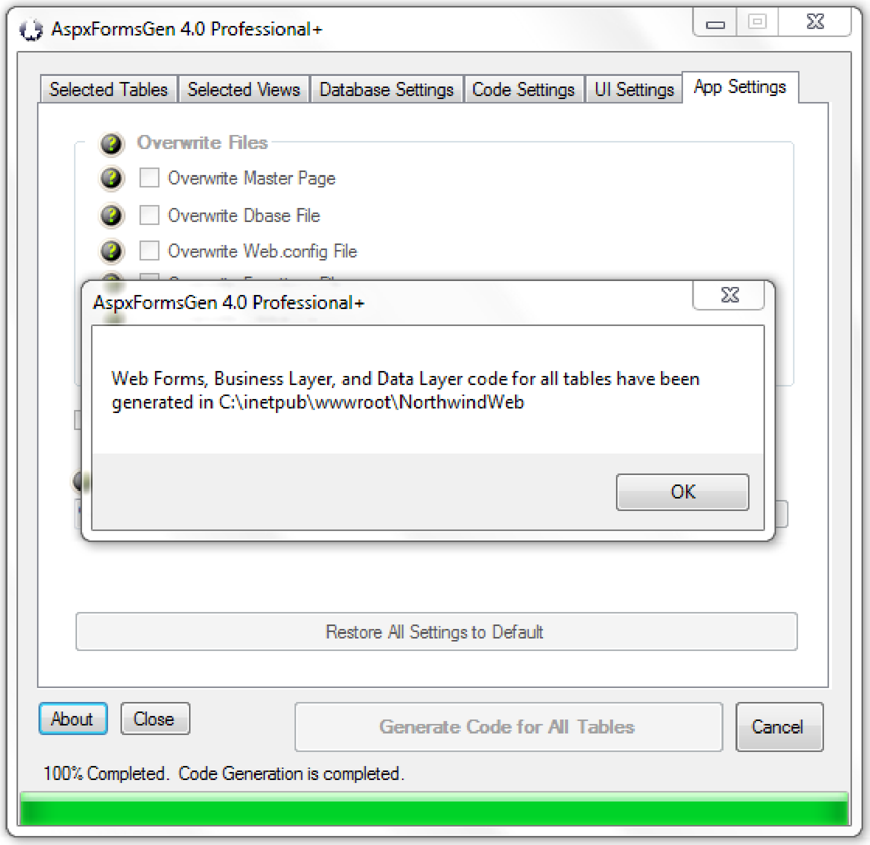
[ ] Automatically Open Selected Tables or Selected Views tab

App Files Directory: 'rogram Files (x86)\Junnark.Com\AspxFormsGen 4.0 Professional Plus\AppFiles\

11% completed. Loading dbo.Customers

**Figure 5 Generate Code for All Tables**

7. Once code generation is done, a message is shown, click *OK*. See Figure 6.



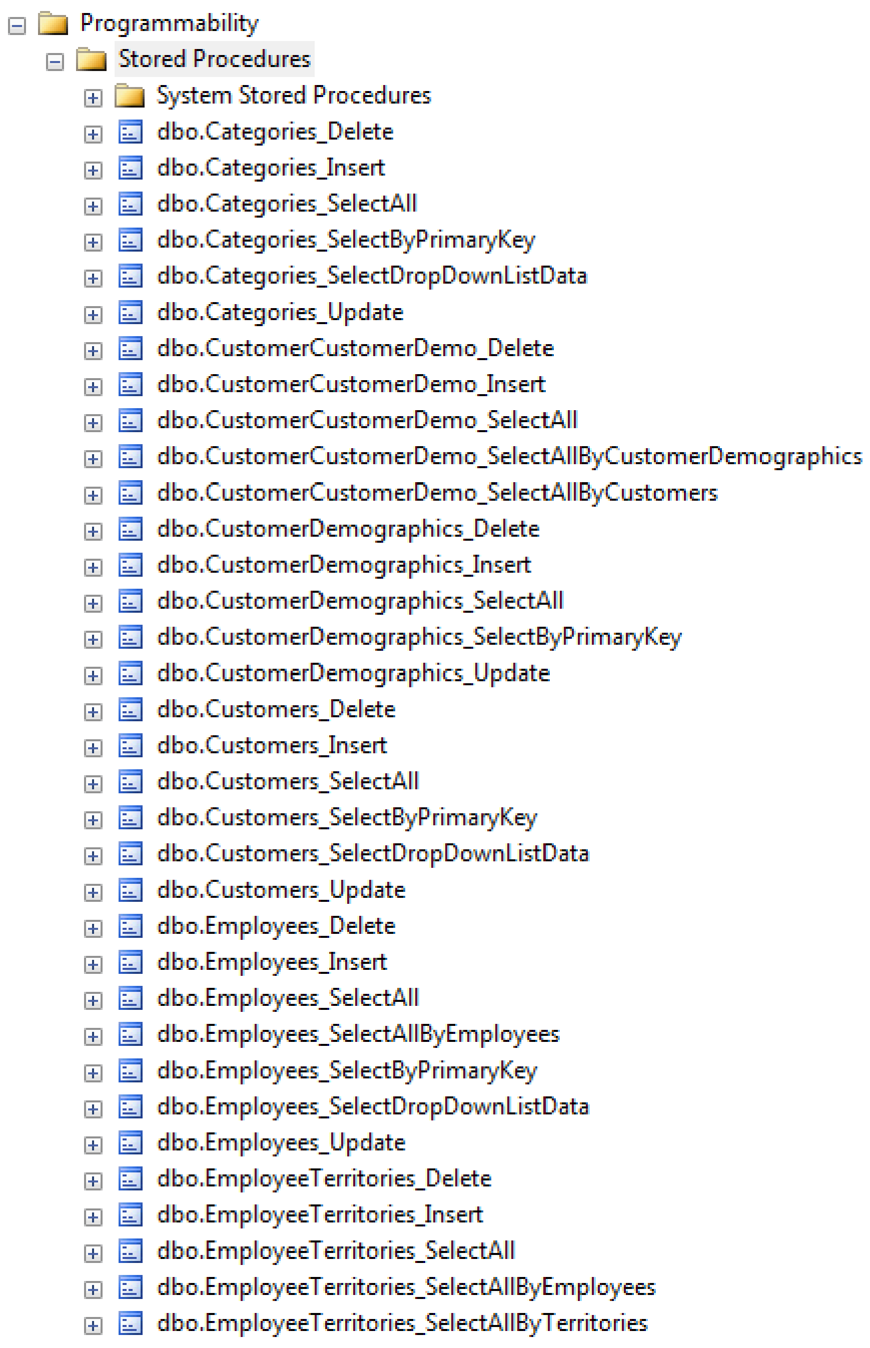
AspxFormsGen 4.0 Professional+

Web Forms, Business Layer, and Data Layer code for all tables have been generated in C:\inetpub\wwwroot\NorthwindWeb

100% Completed. Code Generation is completed.

**Figure 6 Code Generation is Done**

8. Open *MS SQL Server Management Studio* and drill down to the *Stored Procedures* [1] node to see the generated stored procedures. For now, we'll just view these stored procedures and we'll come back to it later and examine the generated code. See Figure 7.



Programmability
Stored Procedures
System Stored Procedures
dbo.Categories_Delete
dbo.Categories_Insert
dbo.Categories_SelectAll
dbo.Categories_SelectByPrimaryKey
dbo.Categories_SelectDropDownListData
dbo.Categories_Update
dbo.CustomerCustomerDemo_Delete
dbo.CustomerCustomerDemo_Insert
dbo.CustomerCustomerDemo_SelectAll
dbo.CustomerCustomerDemo_SelectAllByCustomerDemographics
dbo.CustomerCustomerDemo_SelectAllByCustomers
dbo.CustomerDemographics_Delete
dbo.CustomerDemographics_Insert
dbo.CustomerDemographics_SelectAll
dbo.CustomerDemographics_SelectByPrimaryKey
dbo.CustomerDemographics_Update
dbo.Customers_Delete
dbo.Customers_Insert
dbo.Customers_SelectAll
dbo.Customers_SelectByPrimaryKey
dbo.Customers_SelectDropDownListData
dbo.Customers_Update
dbo.Employees_Delete
dbo.Employees_Insert
dbo.Employees_SelectAll
dbo.Employees_SelectAllByEmployees
dbo.Employees_SelectByPrimaryKey
dbo.Employees_SelectDropDownListData
dbo.Employees_Update
dbo.EmployeeTerritories_Delete
dbo.EmployeeTerritories_Insert
dbo.EmployeeTerritories_SelectAll
dbo.EmployeeTerritories_SelectAllByEmployees
dbo.EmployeeTerritories_SelectAllByTerritories

**Figure 7 List of Generated Stored Procedures**

9. Open *Visual Studio 2010*. On the File menu click *Open Web Site*. See Figure 8.

10. Point to the web site directory, and then click *Open*. See Figure 9.

11. From the *Solution Explorer*, right-click on the *Default.htm* and then click *Set As Start Page*. **Note:** The purpose of this page is to show you the list of web pages that were generated; you don't have to make this as your start page. See Figure 10.

12. Run *Visual Studio* by pressing *F5*. You will see a list of all the generated ASP.NET 4.0 web forms. See Figure 11. You can click any link to preview the functionality of each of the generated web forms.

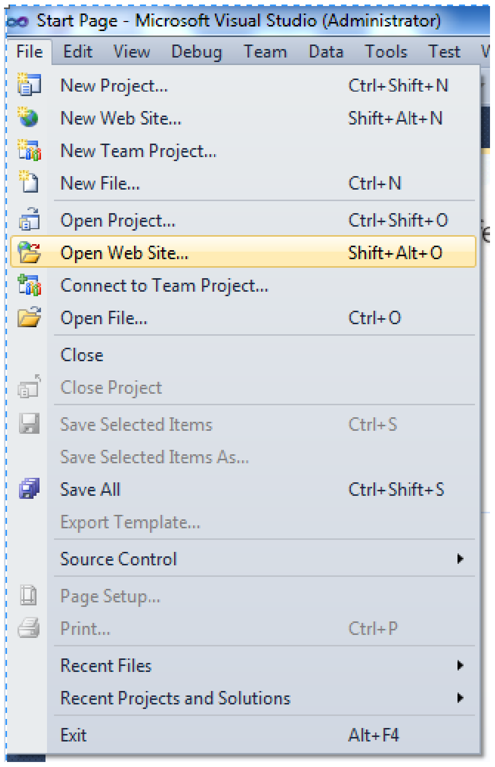
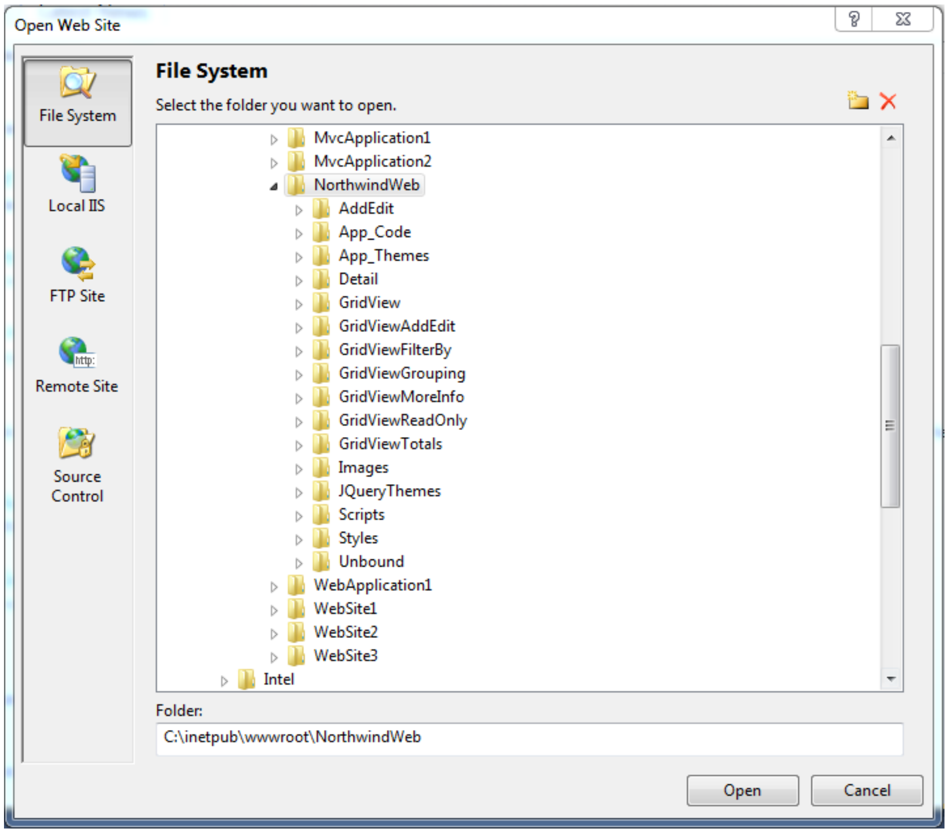

Start Page - Microsoft Visual Studio (Administrator)
File Edit View Debug Team Data Tools Test
New Project... Ctrl+Shift+N
New Web Site... Shift+Alt+N
New Team Project...
New File... Ctrl+N
Open Project... Ctrl+Shift+O
Open Web Site... Shift+Alt+O
Connect to Team Project...
Open File... Ctrl+O
Close
Close Project
Save Selected Items Ctrl+S
Save Selected Items As...
Save All Ctrl+Shift+S
Export Template...
Source Control
Page Setup...
Print... Ctrl+P
Recent Files
Recent Projects and Solutions
Exit Alt+F4

**Figure 8 Open Web Site**



Folder:
C:\inetpub\wwwroot\NorthwindWeb

**Figure 9 Generated Web Site Directory**

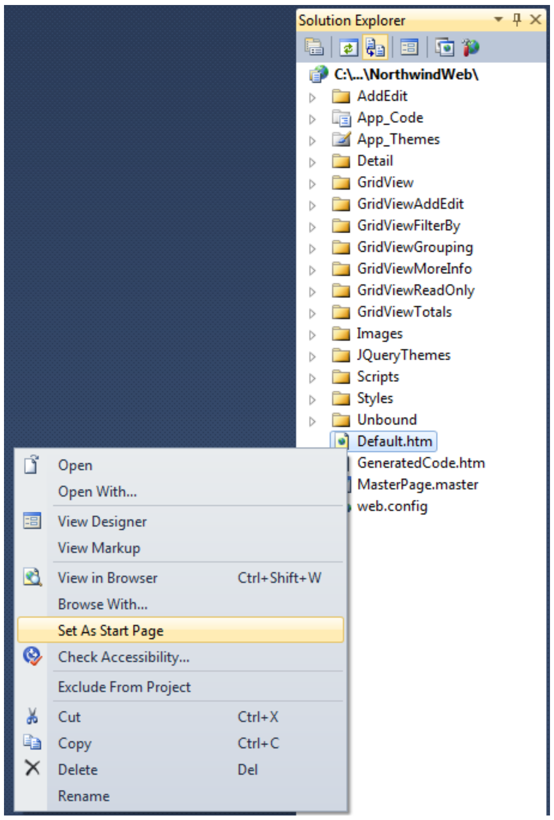
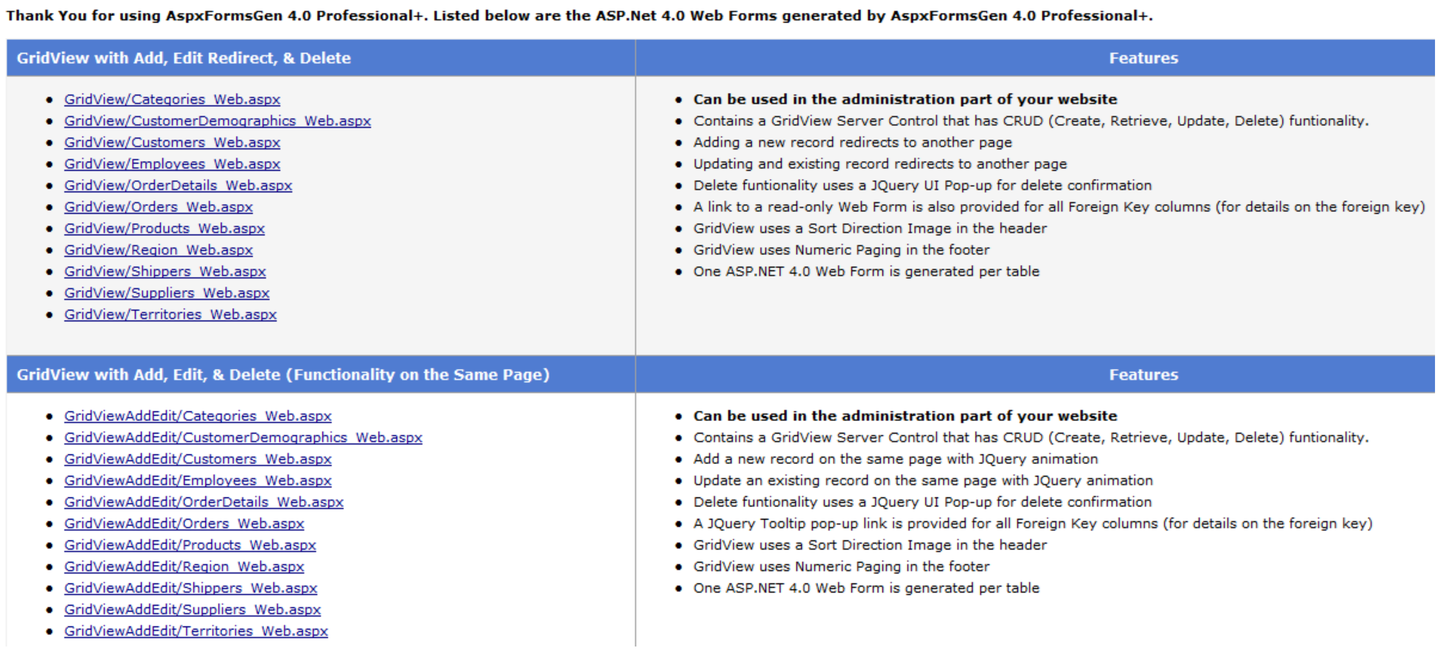


**Figure 10 Set As Start page**

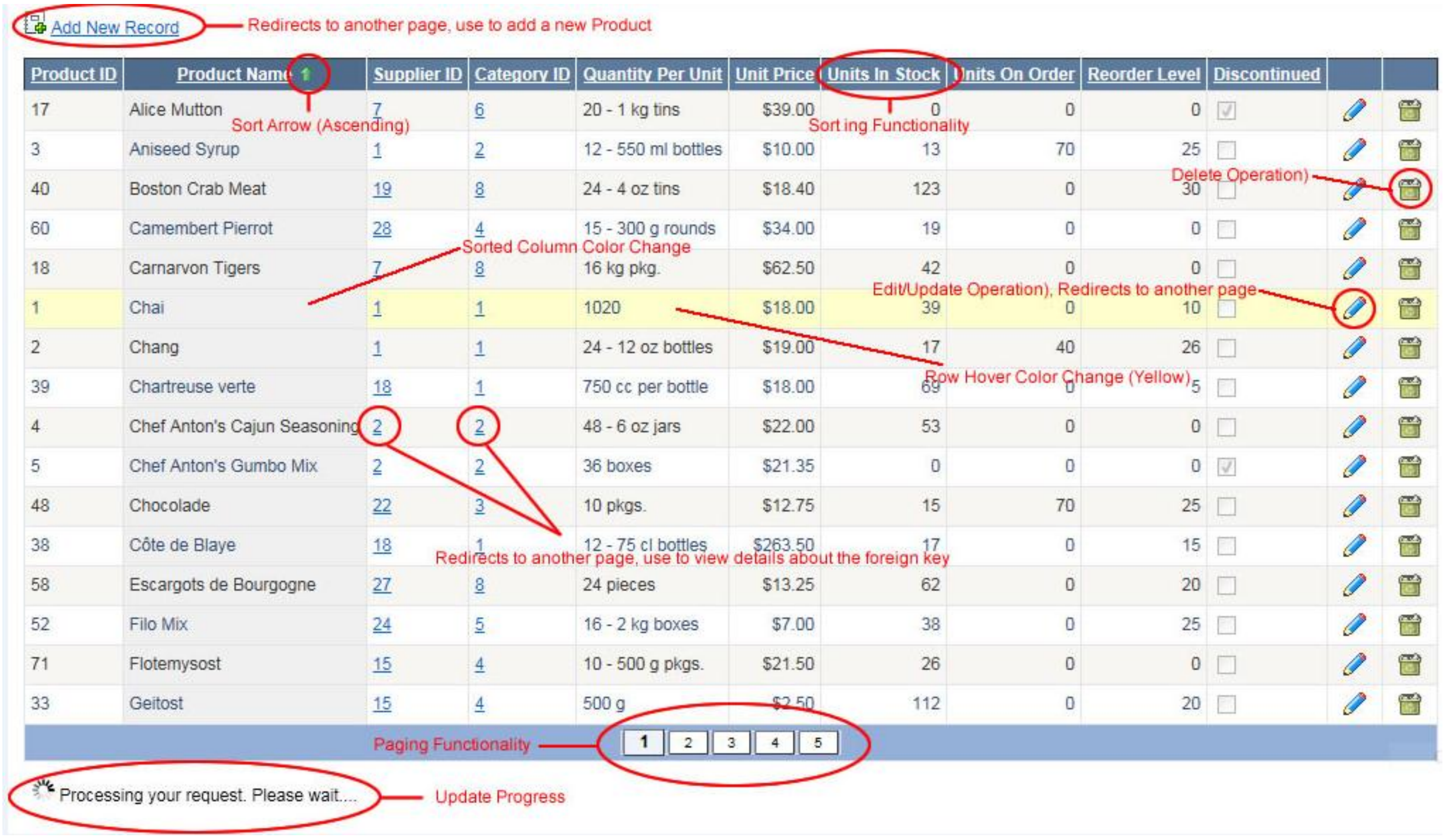


**Thank You for using AspxFormsGen 4.0 Professional+. Listed below are the ASP.Net 4.0 Web Forms generated by AspxFormsGen 4.0 Professional+.**

| GridView with Add, Edit Redirect, & Delete | Features |
|---|---|
| • GridView/Categories_Web.aspx<br>• GridView/CustomerDemographics_Web.aspx<br>• GridView/Customers_Web.aspx<br>• GridView/Employees_Web.aspx<br>• GridView/OrderDetails_Web.aspx<br>• GridView/Orders_Web.aspx<br>• GridView/Products_Web.aspx<br>• GridView/Region_Web.aspx<br>• GridView/Shippers_Web.aspx<br>• GridView/Suppliers_Web.aspx<br>• GridView/Territories_Web.aspx | • **Can be used in the administration part of your website**<br>• Contains a GridView Server Control that has CRUD (Create, Retrieve, Update, Delete) funtionality.<br>• Adding a new record redirects to another page<br>• Updating and existing record redirects to another page<br>• Delete funtionality uses a JQuery UI Pop-up for delete confirmation<br>• A link to a read-only Web Form is also provided for all Foreign Key columns (for details on the foreign key)<br>• GridView uses a Sort Direction Image in the header<br>• GridView uses Numeric Paging in the footer<br>• One ASP.NET 4.0 Web Form is generated per table |
| **GridView with Add, Edit, & Delete (Functionality on the Same Page)** | **Features** |
| • GridViewAddEdit/Categories_Web.aspx<br>• GridViewAddEdit/CustomerDemographics_Web.aspx<br>• GridViewAddEdit/Customers_Web.aspx<br>• GridViewAddEdit/Employees_Web.aspx<br>• GridViewAddEdit/OrderDetails_Web.aspx<br>• GridViewAddEdit/Orders_Web.aspx<br>• GridViewAddEdit/Products_Web.aspx<br>• GridViewAddEdit/Region_Web.aspx<br>• GridViewAddEdit/Shippers_Web.aspx<br>• GridViewAddEdit/Suppliers_Web.aspx<br>• GridViewAddEdit/Territories_Web.aspx | • **Can be used in the administration part of your website**<br>• Contains a GridView Server Control that has CRUD (Create, Retrieve, Update, Delete) funtionality.<br>• Add a new record on the same page with JQuery animation<br>• Update an existing record on the same page with JQuery animation<br>• Delete funtionality uses a JQuery UI Pop-up for delete confirmation<br>• A JQuery Tooltip pop-up link is provided for all Foreign Key columns (for details on the foreign key)<br>• GridView uses a Sort Direction Image in the header<br>• GridView uses Numeric Paging in the footer<br>• One ASP.NET 4.0 Web Form is generated per table |

**Figure 11 List of Generated Web Forms**

13. Let's preview one of the generated web pages. Under the *GridView with Add, Edit Redirect, & Delete*[1] category click *GridView/Products_Web.aspx* link. This will redirect you to the product page. Products here are shown in a GridView web control. All the features of this web page are listed in the *Default.htm*. For example, you can click the header column to sort by column. A list of the functionality is graphically shown in Figure 12.


Add New Record
Redirects to another page, use to add a new Product
Sort Arrow (Ascending)
Sort ing Functionality
Delete Operation)
Sorted Column Color Change
Edit/Update Operation), Redirects to another page
Row Hover Color Change (Yellow)
Redirects to another page, use to view details about the foreign key
Paging Functionality
1 2 3 4 5
Processing your request. Please wait....
Update Progress

| Product ID | Product Name | Supplier ID | Category ID | Quantity Per Unit | Unit Price | Units In Stock | Units On Order | Reorder Level | Discontinued | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 17 | Alice Mutton | 7 | 6 | 20 - 1 kg tins | $39.00 | 0 | 0 | 0 | ✓ | | |
| 3 | Aniseed Syrup | 1 | 2 | 12 - 550 ml bottles | $10.00 | 13 | 70 | 25 | ☐ | | |
| 40 | Boston Crab Meat | 19 | 8 | 24 - 4 oz tins | $18.40 | 123 | 0 | 30 | ☐ | | |
| 60 | Camembert Pierrot | 28 | 4 | 15 - 300 g rounds | $34.00 | 19 | 0 | 0 | ☐ | | |
| 18 | Carnarvon Tigers | 7 | 8 | 16 kg pkg. | $62.50 | 42 | 0 | 0 | ☐ | | |
| 1 | Chai | 1 | 1 | 1020 | $18.00 | 39 | 0 | 10 | ☐ | | |
| 2 | Chang | 1 | 1 | 24 - 12 oz bottles | $19.00 | 17 | 40 | 26 | ☐ | | |
| 39 | Chartreuse verte | 18 | 1 | 750 cc per bottle | $18.00 | 69 | 0 | 5 | ☐ | | |
| 4 | Chef Anton's Cajun Seasoning | 2 | 2 | 48 - 6 oz jars | $22.00 | 53 | 0 | 0 | ☐ | | |
| 5 | Chef Anton's Gumbo Mix | 2 | 2 | 36 boxes | $21.35 | 0 | 0 | 0 | ✓ | | |
| 48 | Chocolade | 22 | 3 | 10 pkgs. | $12.75 | 15 | 70 | 25 | ☐ | | |
| 38 | Côte de Blaye | 18 | 1 | 12 - 75 cl bottles | $263.50 | 17 | 0 | 15 | ☐ | | |
| 58 | Escargots de Bourgogne | 27 | 8 | 24 pieces | $13.25 | 62 | 0 | 20 | ☐ | | |
| 52 | Filo Mix | 24 | 5 | 16 - 2 kg boxes | $7.00 | 38 | 0 | 25 | ☐ | | |
| 71 | Flotemysost | 15 | 4 | 10 - 500 g pkgs. | $21.50 | 26 | 0 | 0 | ☐ | | |
| 33 | Geitost | 15 | 4 | 500 g | $2.50 | 112 | 0 | 20 | ☐ | | |

Figure 12 GridView with Add, Edit Redirect, & Delete Functionality

14. Go ahead and play around, checking the functionality of this web page and a few other web pages. Most of the functionalities are self-explanatory, so we will not dwell on these. The functionalities are also explained under *Code Settings* above. You can also view a live web demos from our web site in the specific product's page.

15. The generated web forms can be found in their respective folders. The web forms are in folders because we specified AspxFormsGen to *Organize Web Forms* under *UI Settings*. Please see explanation under *UI Settings* above. Also see Figure 13, it shows the folders where the respective web forms were generated into.

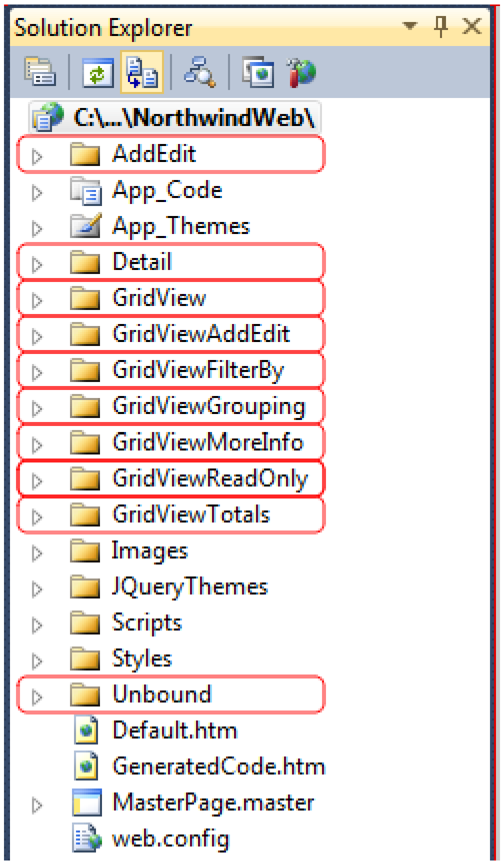


**Figure 13 Organize Web Forms into Folders**

16. Close the web page and go back to *Visual Studio 2010*. From the *Solution Explorer*, right-click on the *GeneratedCode.htm* and then click *Set As Start Page*. See Figure 14.

17. Run Visual Studio by pressing F5. You will see a list of all the generated middle-tier classes, data-tier classes, and stored procedures (or dynamic SQL classes). See Figure 15. You can hover over each of the link to see where each file is located.

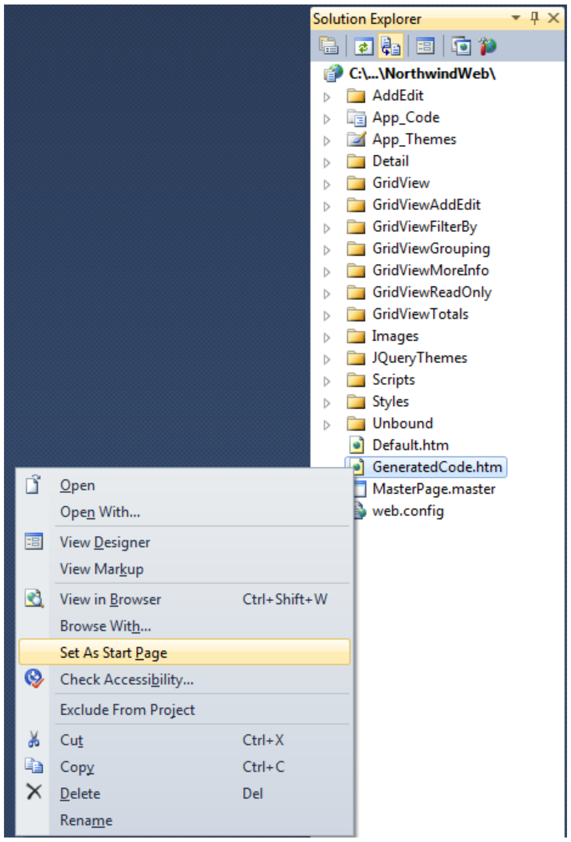
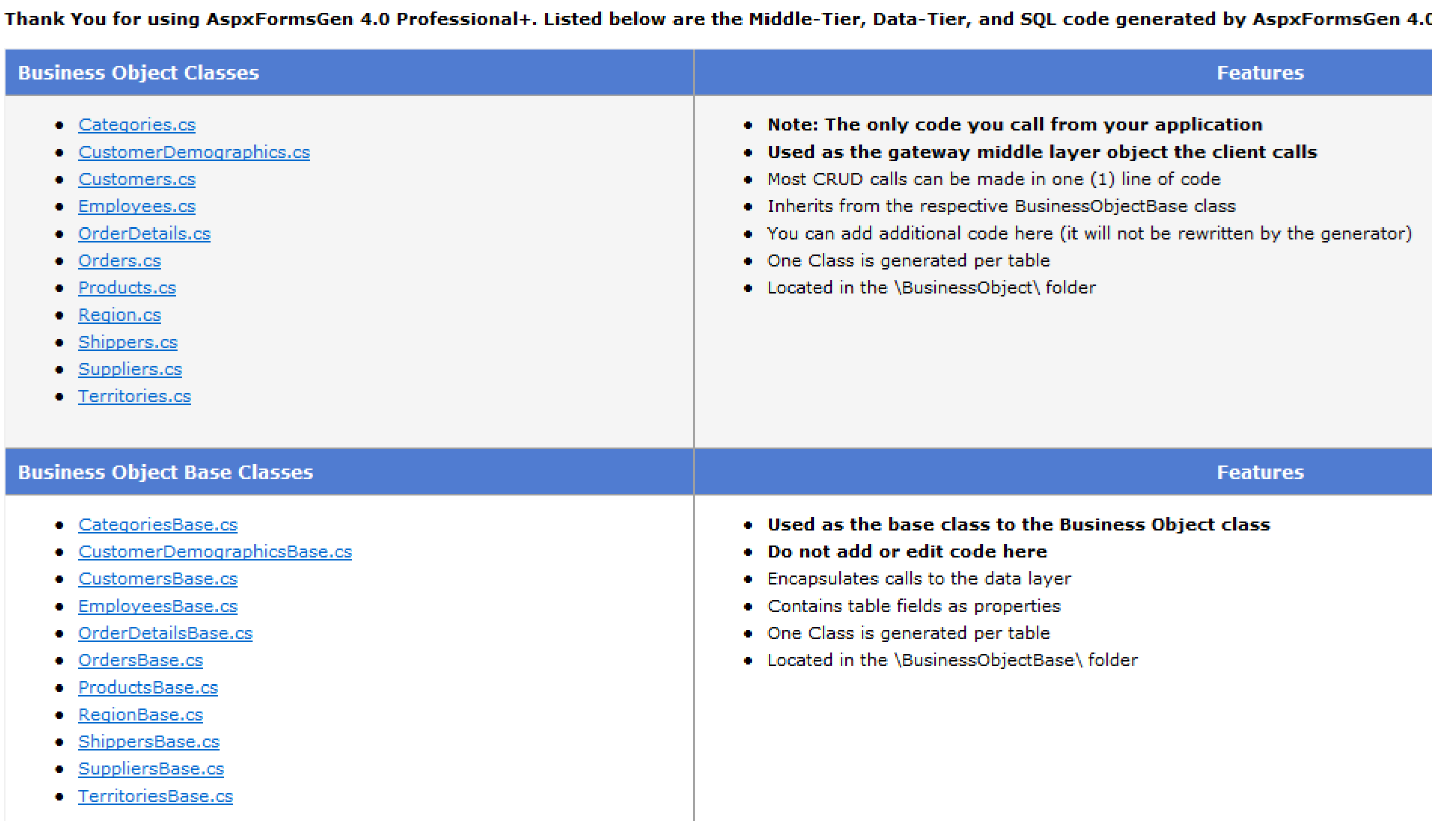


Figure 14 Set As Start Page



**Thank You for using AspxFormsGen 4.0 Professional+. Listed below are the Middle-Tier, Data-Tier, and SQL code generated by AspxFormsGen 4.0**

| Business Object Classes | Features |
|---|---|
| • Categories.cs<br>• CustomerDemographics.cs<br>• Customers.cs<br>• Employees.cs<br>• OrderDetails.cs<br>• Orders.cs<br>• Products.cs<br>• Region.cs<br>• Shippers.cs<br>• Suppliers.cs<br>• Territories.cs | • **Note: The only code you call from your application**<br>• **Used as the gateway middle layer object the client calls**<br>• Most CRUD calls can be made in one (1) line of code<br>• Inherits from the respective BusinessObjectBase class<br>• You can add additional code here (it will not be rewritten by the generator)<br>• One Class is generated per table<br>• Located in the \BusinessObject\ folder |
| **Business Object Base Classes** | **Features** |
| • CategoriesBase.cs<br>• CustomerDemographicsBase.cs<br>• CustomersBase.cs<br>• EmployeesBase.cs<br>• OrderDetailsBase.cs<br>• OrdersBase.cs<br>• ProductsBase.cs<br>• RegionBase.cs<br>• ShippersBase.cs<br>• SuppliersBase.cs<br>• TerritoriesBase.cs | • **Used as the base class to the Business Object class**<br>• **Do not add or edit code here**<br>• Encapsulates calls to the data layer<br>• Contains table fields as properties<br>• One Class is generated per table<br>• Located in the \BusinessObjectBase\ folder |

Figure 15 List of Middle-Tier, Data-Tier, and Stored Procedures (or Dynamic SQL)

18. Close the web page and go back to *Visual Studio 2010*. The generated middle-tier and data-tier classes can be found under the *App_Code* folder. Please see Figure 16.

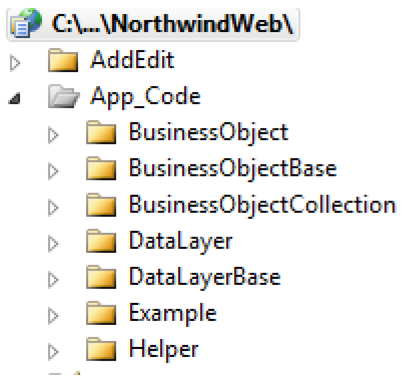


**Figure 16 Middle-Tier and Data-Tier Classes Under App_Code Folder**

19. The middle-tier classes can be found in folders:
    a. BusinessObject
    b. BusinessObjectBase
    c. BusinessObjectCollection

20. The data-layer classes can be found in folders:
    a. DataLayer
    b. DataLayerBase

21. You can find a deeper discussion on the generated web forms, middle-tier, data-tier, stored procedures or dynamic SQL under the Generated Code below. For now, this will be the end of this tutorial.

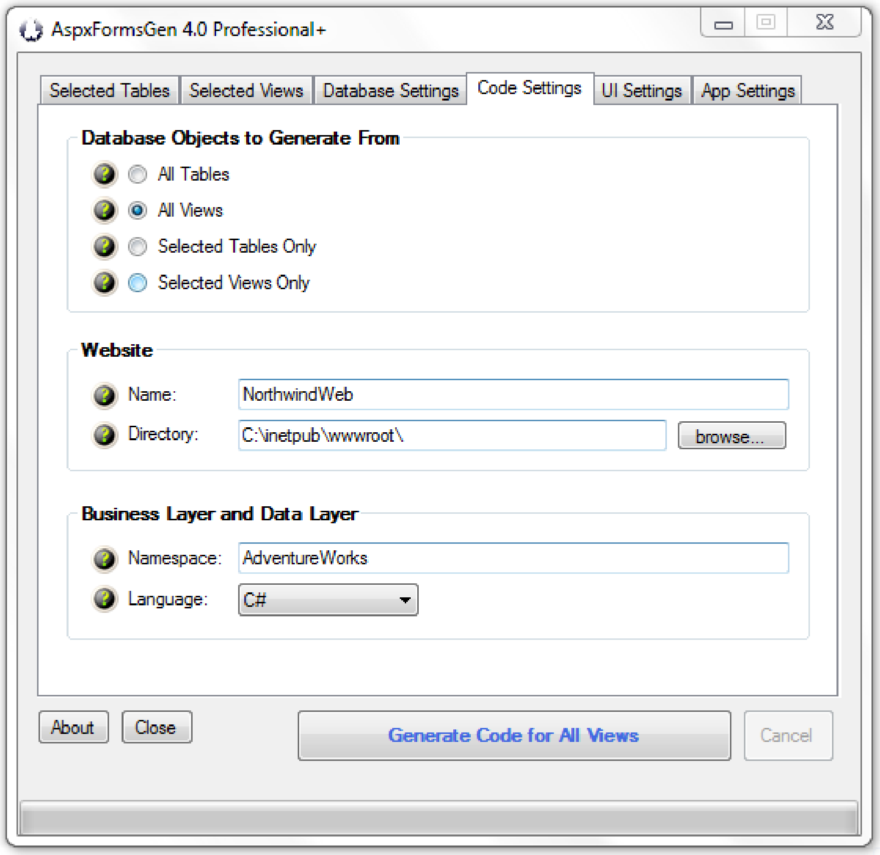
## For All Views [1]

The *All Views* option generates objects for all views in the respective database.

1. To follow this tutorial make sure to delete the *NorthwindWeb* web site we generated under the *For All Tables* tutorial to get a fresh start. Also delete all the Stored Procedures that was generated by the earlier tutorial.

2. Open AspxFormsGen 4.0. By now you will notice that the last settings were saved. We could easily use the **One Click** feature by clicking the *Generate…* button right away, but don't for now.

3. Open the *Code Settings* tab then select *All Views* under the *Database Objects to Generate From*. Keep the rest of the settings on this tab. See Figure 17.



Database Objects to Generate From

- All Tables
- [✓ All Views]
- Selected Tables Only
- Selected Views Only

Website

Name: NorthwindWeb

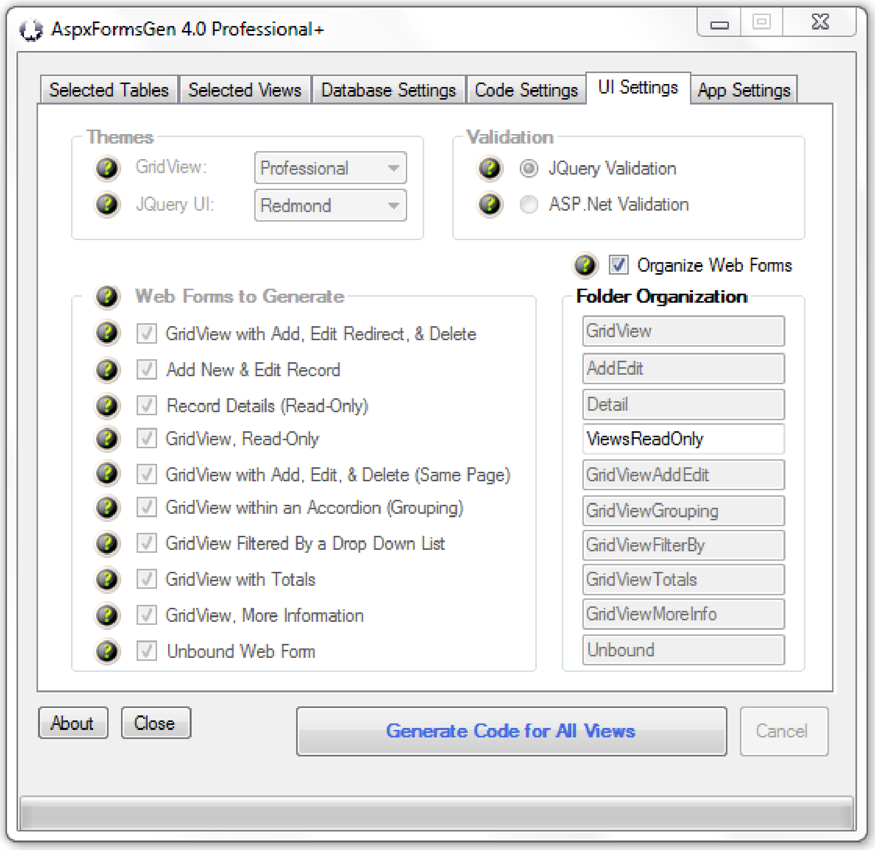
Directory: C:\inetpub\wwwroot\

Business Layer and Data Layer

Namespace: AdventureWorks

Language: C#

Figure 17 Code Settings Tab – All Views



Themes

GridView: Professional

JQuery UI: Redmond

Validation

- [✓ JQuery Validation]
- ASP.Net Validation

[✓] Organize Web Forms

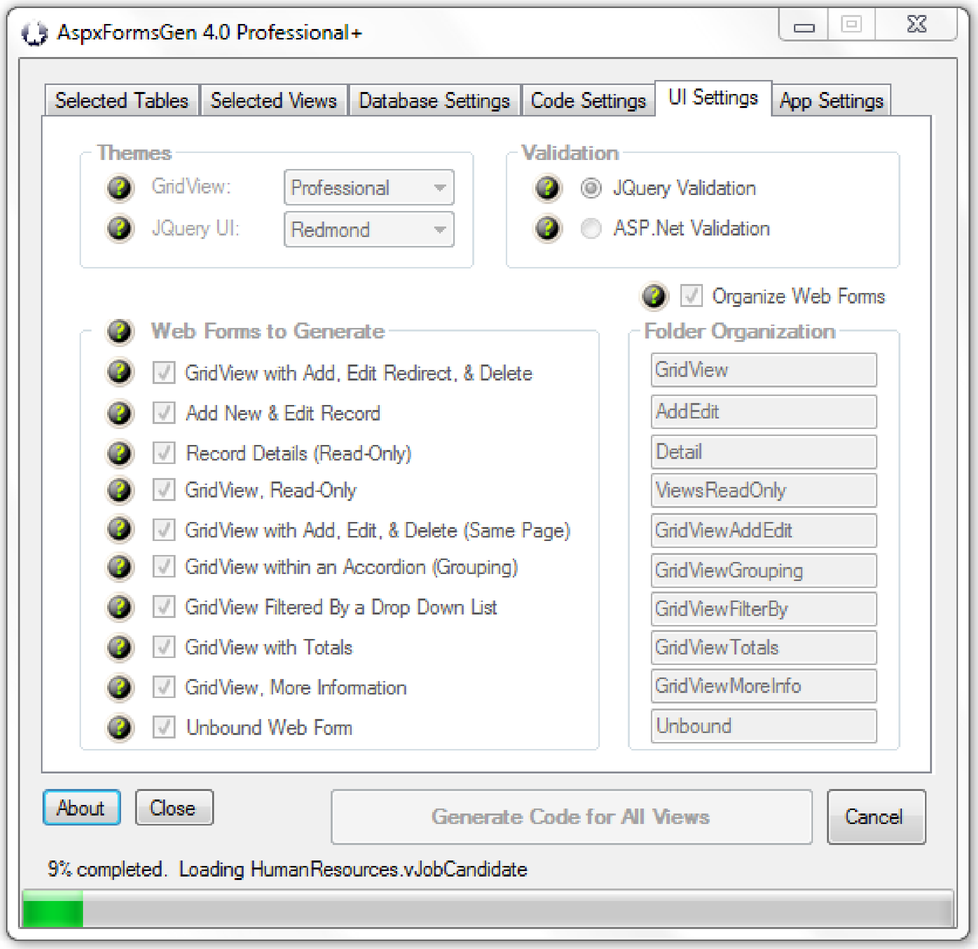
| Web Forms to Generate | Folder Organization |
|---|---|
| [✓] GridView with Add, Edit Redirect, & Delete | GridView |
| [✓] Add New & Edit Record | AddEdit |
| [✓] Record Details (Read-Only) | Detail |
| [✓] GridView, Read-Only | ViewsReadOnly |
| [✓] GridView with Add, Edit, & Delete (Same Page) | GridViewAddEdit |
| [✓] GridView within an Accordion (Grouping) | GridViewGrouping |
| [✓] GridView Filtered By a Drop Down List | GridViewFilterBy |
| [✓] GridView with Totals | GridViewTotals |
| [✓] GridView, More Information | GridViewMoreInfo |
| [✓] Unbound Web Form | Unbound |

Figure 18 UI Settings Tab

4. Open the *UI Settings* tab. You will notice that everything is disabled except for the *Organize Web Form* check box and the *GridView, Read-Only's* respective *Folder Organization/Web Form Prefix*. This is because views are **read-only**, that's why the only web forms that will be generated are read-only web forms. In short, there will be no CRUD operation for the generated web forms as well as the generated middle-tier, data-tier, and stored procedures or dynamic SQL.

   Let's put the generated web forms to a different folder, change the text "*GridViewReadOnly*" to "*ViewsReadOnly*", of course you can put any text here. See Figure 18.

5. We will keep all the settings under the *Database Settings* tab. Click the Generate Code for All Views button, AspxFormsGen will start generating code. See Figure 19.


AspxFormsGen 4.0 Professional+

Selected Tables | Selected Views | Database Settings | Code Settings | UI Settings | App Settings

Themes
GridView: Professional
JQuery UI: Redmond

Validation
JQuery Validation [✓]
ASP.Net Validation

[✓] Organize Web Forms

| Web Forms to Generate | Folder Organization |
|---|---|
| [✓] GridView with Add, Edit Redirect, & Delete | GridView |
| [✓] Add New & Edit Record | AddEdit |
| [✓] Record Details (Read-Only) | Detail |
| [✓] GridView, Read-Only | ViewsReadOnly |
| [✓] GridView with Add, Edit, & Delete (Same Page) | GridViewAddEdit |
| [✓] GridView within an Accordion (Grouping) | GridViewGrouping |
| [✓] GridView Filtered By a Drop Down List | GridViewFilterBy |
| [✓] GridView with Totals | GridViewTotals |
| [✓] GridView, More Information | GridViewMoreInfo |
| [✓] Unbound Web Form | Unbound |

About | Close | Generate Code for All Views | Cancel

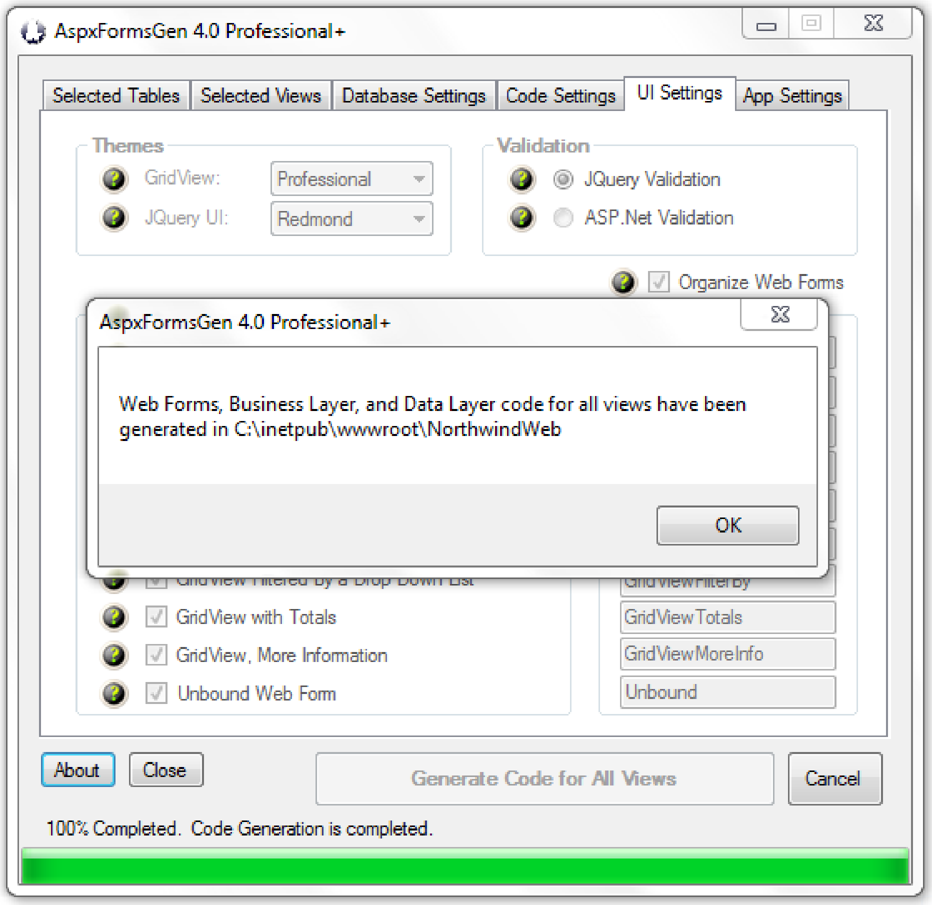
9% completed. Loading HumanResources.vJobCandidate

**Figure 19 Generate Code For All Views**

6. When done generating code, a message box is shown. Click OK, and then close AspxFormsGen. See Figure 20.

7. Open *Visual Studio 2010*. On the File menu click *Open Web Site*. See Figure 8 above.

8. Point to the web site directory, and then click *Open*. See Figure 9 above.

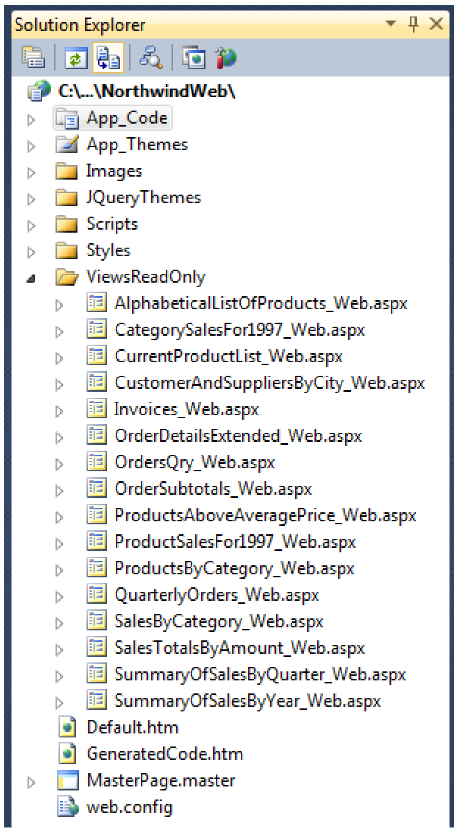


AspxFormsGen 4.0 Professional+

Web Forms, Business Layer, and Data Layer code for all views have been generated in C:\inetpub\wwwroot\NorthwindWeb

100% Completed. Code Generation is completed.

**Figure 20 Done Generating Code For All Views**



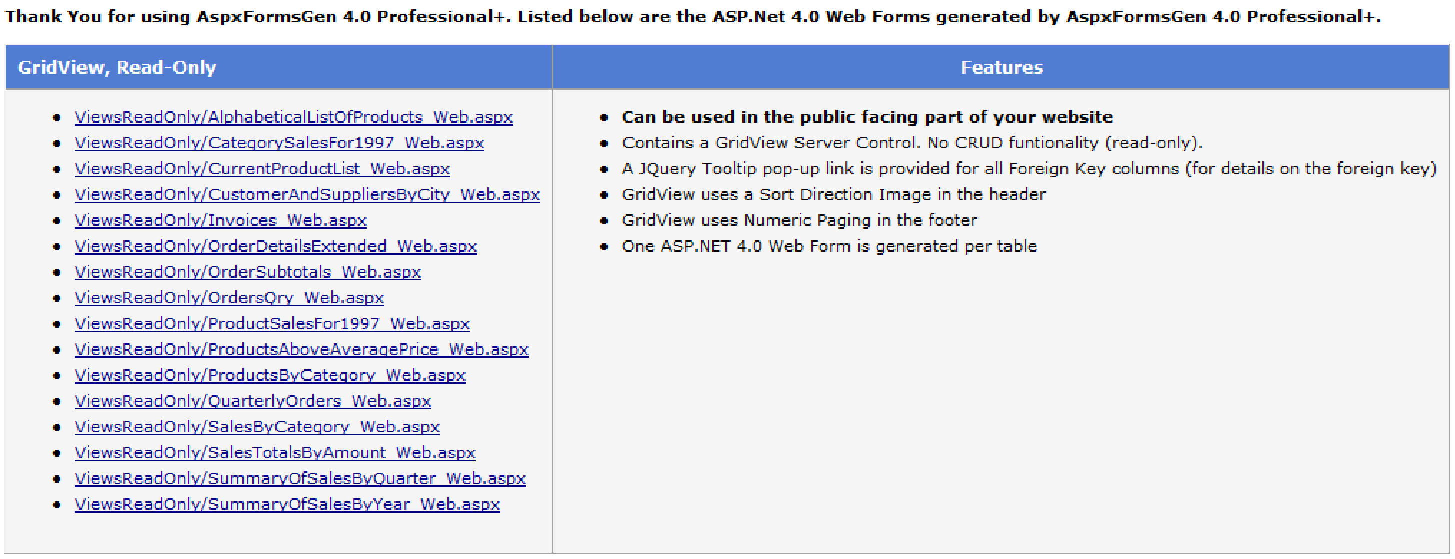
```
C:\...\NorthwindWeb\
  App_Code
  App_Themes
  Images
  JQueryThemes
  Scripts
  Styles
  ViewsReadOnly
    AlphabeticalListOfProducts_Web.aspx
    CategorySalesFor1997_Web.aspx
    CurrentProductList_Web.aspx
    CustomerAndSuppliersByCity_Web.aspx
    Invoices_Web.aspx
    OrderDetailsExtended_Web.aspx
    OrdersQry_Web.aspx
    OrderSubtotals_Web.aspx
    ProductsAboveAveragePrice_Web.aspx
    ProductSalesFor1997_Web.aspx
    ProductsByCategory_Web.aspx
    QuarterlyOrders_Web.aspx
    SalesByCategory_Web.aspx
    SalesTotalsByAmount_Web.aspx
    SummaryOfSalesByQuarter_Web.aspx
    SummaryOfSalesByYear_Web.aspx
  Default.htm
  GeneratedCode.htm
  MasterPage.master
  web.config
```

**Figure 21 Generated Web Site**

9. Let's pause for a moment and look at the generated objects under the *Solution Explorer*. You will notice that only one folder was generated for the web forms, the *ViewsReadOnly* folder, as we specified during code generation. See Figure 21.

10. Set the *default.htm* as Start page. See Figure 10 above.

11. Run the web site by pressing F5. You will now see a different list. See Figure 22.


**Thank You for using AspxFormsGen 4.0 Professional+. Listed below are the ASP.Net 4.0 Web Forms generated by AspxFormsGen 4.0 Professional+.**

| GridView, Read-Only | Features |
|---|---|
| • ViewsReadOnly/AlphabeticalListOfProducts_Web.aspx<br>• ViewsReadOnly/CategorySalesFor1997_Web.aspx<br>• ViewsReadOnly/CurrentProductList_Web.aspx<br>• ViewsReadOnly/CustomerAndSuppliersByCity_Web.aspx<br>• ViewsReadOnly/Invoices_Web.aspx<br>• ViewsReadOnly/OrderDetailsExtended_Web.aspx<br>• ViewsReadOnly/OrderSubtotals_Web.aspx<br>• ViewsReadOnly/OrdersQry_Web.aspx<br>• ViewsReadOnly/ProductSalesFor1997_Web.aspx<br>• ViewsReadOnly/ProductsAboveAveragePrice_Web.aspx<br>• ViewsReadOnly/ProductsByCategory_Web.aspx<br>• ViewsReadOnly/QuarterlyOrders_Web.aspx<br>• ViewsReadOnly/SalesByCategory_Web.aspx<br>• ViewsReadOnly/SalesTotalsByAmount_Web.aspx<br>• ViewsReadOnly/SummaryOfSalesByQuarter_Web.aspx<br>• ViewsReadOnly/SummaryOfSalesByYear_Web.aspx | • **Can be used in the public facing part of your website**<br>• Contains a GridView Server Control. No CRUD funtionality (read-only).<br>• A JQuery Tooltip pop-up link is provided for all Foreign Key columns (for details on the foreign key)<br>• GridView uses a Sort Direction Image in the header<br>• GridView uses Numeric Paging in the footer<br>• One ASP.NET 4.0 Web Form is generated per table |

**Figure 22 List of Generated Web Forms**

12. Notice that we only generated *GridView, Read-Only* type web forms. **Note:** When you choose *All Views* or *Selected Views Only* under the *Code Settings* tab in AspxFormsGen 4.0, this is the only type of web form that is generated.

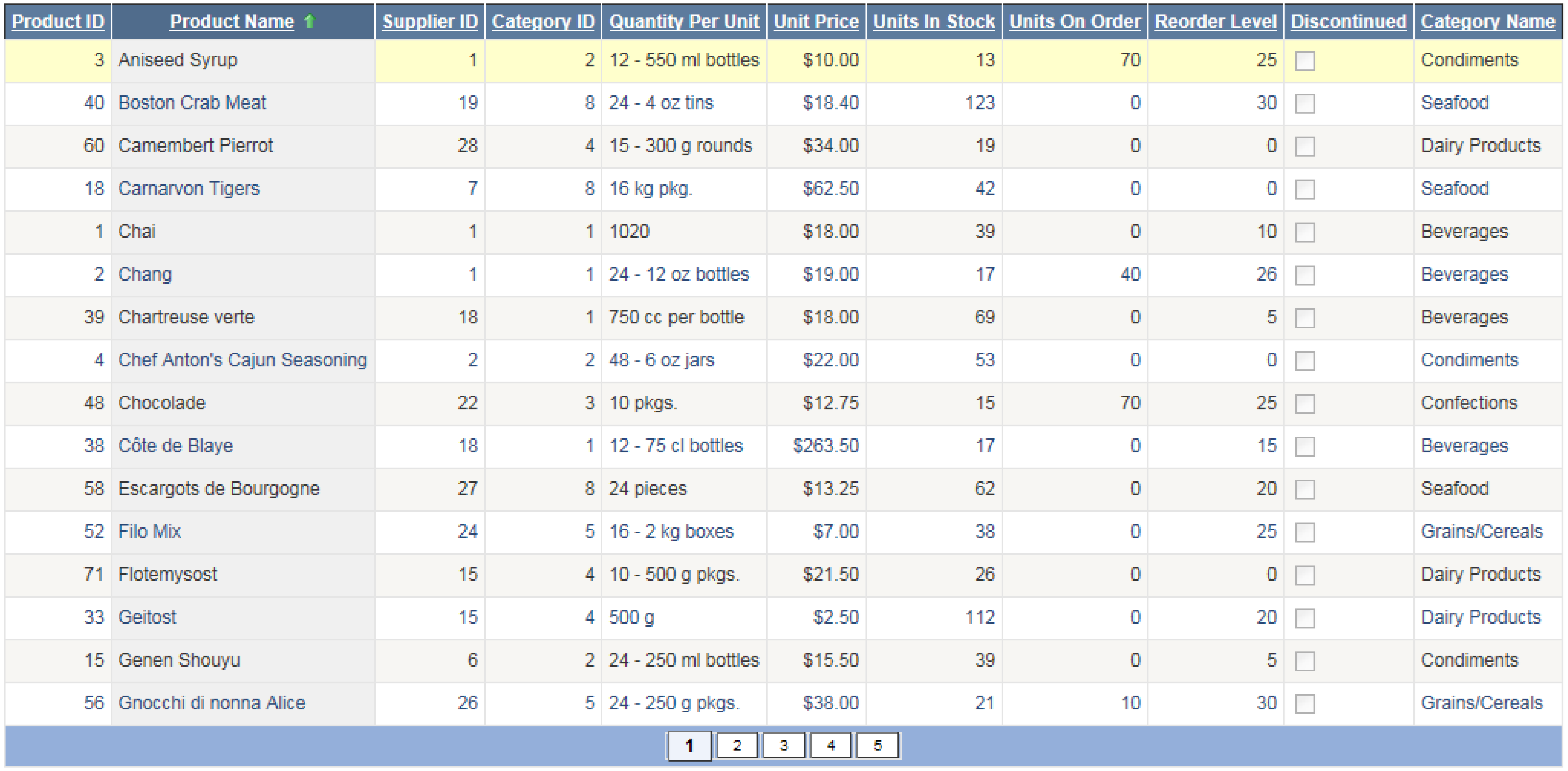
13. Let's preview one of the generated web forms. Click on the very first link. See Figure 23.

14. This web form is Read-Only. There's no Add, Edit/Update, and Delete operation on this web form unlike Figure 12.

15. Close the web page and go back to Visual Studio 2010. From the *Solution Explorer*, right-click on the *GeneratedCode.htm* and then click *Set As Start Page*. See Figure 14.

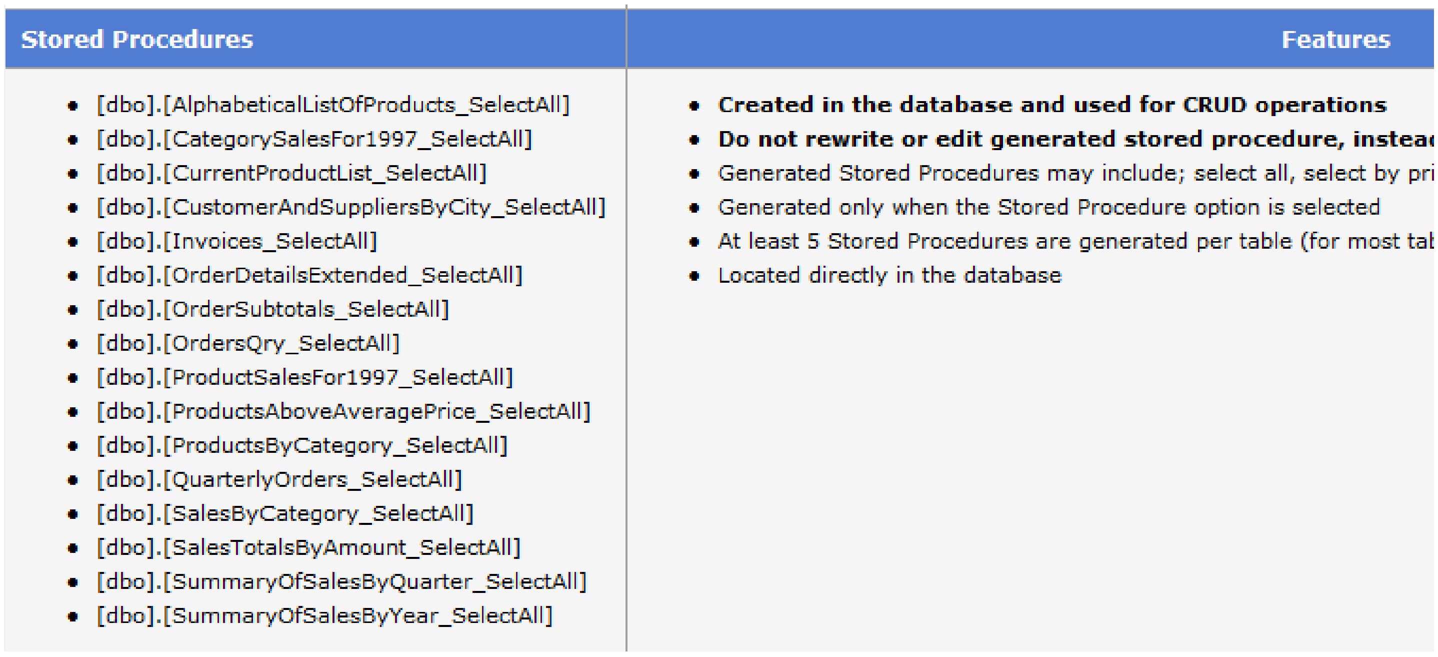
16. Run Visual Studio by pressing F5. You will see a list of all the generated middle-tier classes, data-tier classes, and stored procedures (or dynamic SQL classes). See Figure 24. You can hover over each of the link to see where each file is located.

    Notice that under the *Stored Procedures* everything is "*Select All*", there's no SelectByPrimaryKey, Insert, Update, etc. Again for when you choose *All Views* or *Selected Views Only* under the *Code Settings* tab in AspxFormsGen 4.0, this is the only type of stored procedures (or dynamic SQL methods) that is generated.



| Product ID | Product Name ↑ | Supplier ID | Category ID | Quantity Per Unit | Unit Price | Units In Stock | Units On Order | Reorder Level | Discontinued | Category Name |
|---|---|---|---|---|---|---|---|---|---|---|
| 3 | Aniseed Syrup | 1 | 2 | 12 - 550 ml bottles | $10.00 | 13 | 70 | 25 | ☐ | Condiments |
| 40 | Boston Crab Meat | 19 | 8 | 24 - 4 oz tins | $18.40 | 123 | 0 | 30 | ☐ | Seafood |
| 60 | Camembert Pierrot | 28 | 4 | 15 - 300 g rounds | $34.00 | 19 | 0 | 0 | ☐ | Dairy Products |
| 18 | Carnarvon Tigers | 7 | 8 | 16 kg pkg. | $62.50 | 42 | 0 | 0 | ☐ | Seafood |
| 1 | Chai | 1 | 1 | 1020 | $18.00 | 39 | 0 | 10 | ☐ | Beverages |
| 2 | Chang | 1 | 1 | 24 - 12 oz bottles | $19.00 | 17 | 40 | 26 | ☐ | Beverages |
| 39 | Chartreuse verte | 18 | 1 | 750 cc per bottle | $18.00 | 69 | 0 | 5 | ☐ | Beverages |
| 4 | Chef Anton's Cajun Seasoning | 2 | 2 | 48 - 6 oz jars | $22.00 | 53 | 0 | 0 | ☐ | Condiments |
| 48 | Chocolade | 22 | 3 | 10 pkgs. | $12.75 | 15 | 70 | 25 | ☐ | Confections |
| 38 | Côte de Blaye | 18 | 1 | 12 - 75 cl bottles | $263.50 | 17 | 0 | 15 | ☐ | Beverages |
| 58 | Escargots de Bourgogne | 27 | 8 | 24 pieces | $13.25 | 62 | 0 | 20 | ☐ | Seafood |
| 52 | Filo Mix | 24 | 5 | 16 - 2 kg boxes | $7.00 | 38 | 0 | 25 | ☐ | Grains/Cereals |
| 71 | Flotemysost | 15 | 4 | 10 - 500 g pkgs. | $21.50 | 26 | 0 | 0 | ☐ | Dairy Products |
| 33 | Geitost | 15 | 4 | 500 g | $2.50 | 112 | 0 | 20 | ☐ | Dairy Products |
| 15 | Genen Shouyu | 6 | 2 | 24 - 250 ml bottles | $15.50 | 39 | 0 | 5 | ☐ | Condiments |
| 56 | Gnocchi di nonna Alice | 26 | 5 | 24 - 250 g pkgs. | $38.00 | 21 | 10 | 30 | ☐ | Grains/Cereals |

Figure 23 GridView, Read-Only Web Form



| Stored Procedures | Features |
|---|---|
| • [dbo].[AlphabeticalListOfProducts_SelectAll]<br>• [dbo].[CategorySalesFor1997_SelectAll]<br>• [dbo].[CurrentProductList_SelectAll]<br>• [dbo].[CustomerAndSuppliersByCity_SelectAll]<br>• [dbo].[Invoices_SelectAll]<br>• [dbo].[OrderDetailsExtended_SelectAll]<br>• [dbo].[OrderSubtotals_SelectAll]<br>• [dbo].[OrdersQry_SelectAll]<br>• [dbo].[ProductSalesFor1997_SelectAll]<br>• [dbo].[ProductsAboveAveragePrice_SelectAll]<br>• [dbo].[ProductsByCategory_SelectAll]<br>• [dbo].[QuarterlyOrders_SelectAll]<br>• [dbo].[SalesByCategory_SelectAll]<br>• [dbo].[SalesTotalsByAmount_SelectAll]<br>• [dbo].[SummaryOfSalesByQuarter_SelectAll]<br>• [dbo].[SummaryOfSalesByYear_SelectAll] | • **Created in the database and used for CRUD operations**<br>• **Do not rewrite or edit generated stored procedure, instea**<br>• Generated Stored Procedures may include; select all, select by pri<br>• Generated only when the Stored Procedure option is selected<br>• At least 5 Stored Procedures are generated per table (for most tal<br>• Located directly in the database |

**Figure 24 List of Generated Code, Stored Procedures List**

17. Close the web page and go back to *Visual Studio 2010*. The generated middle-tier and data-tier classes are the same as seen in Figure 16 above and can also be found under the *App_Code* folder.

18. You can find a deeper discussion on the generated web forms, middle-tier, data-tier, stored procedures or dynamic SQL under the Generated Code below. For now, this will be the end of this tutorial.

## For Selected Tables Only

The *Selected Tables Only* option generates objects for selected tables only, in the respective database.

1. To follow this tutorial make sure to delete the *NorthwindWeb* web site we generated earlier to get a fresh start. Also delete all the Stored Procedures that was generated by the earlier tutorial.

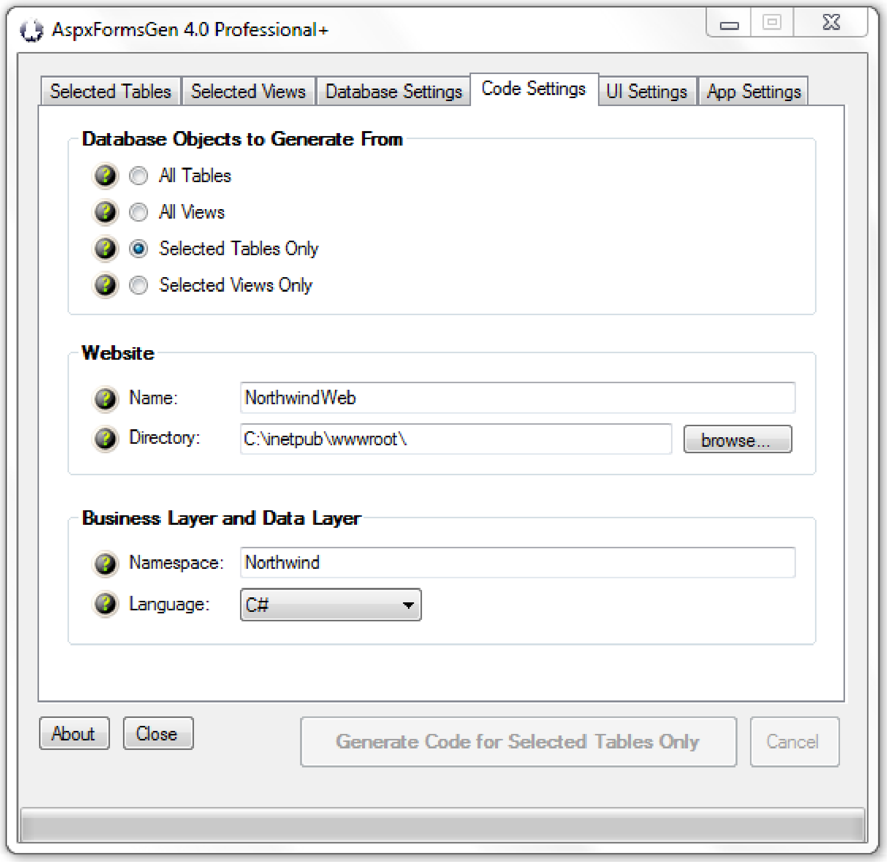
2. Open AspxFormsGen 4.0. By now you will notice that the last settings were saved. We could easily use the **One Click** feature by clicking the *Generate…* button right away, but don't for now.

3. Open the *Code Settings* tab then select *Selected Tables Only* under the *Database Objects to Generate From*. Keep the rest of the settings on this tab. See Figure 25. Selecting the *Selected Tables Only* option will open the *Selected Tables* tab by default; you can change this behavior under *the App Settings* tab if you want, simply uncheck the *Automatically Open Selected Tables or Selected View* tab.

4. The *Load Table* button is now enabled. See Figure 26.

5. Click the Load Table button, and then select the following tables as shown in Figure 27.

6. Open the *Database Settings*[1] tab and select *Use Dynamic SQL*[1] under the *Generated SQL* group. We're doing this so we can see another option in generating SQL code. See Figure 28.


AspxFormsGen 4.0 Professional+

Selected Tables | Selected Views | Database Settings | Code Settings | UI Settings | App Settings

**Database Objects to Generate From**

- All Tables
- All Views
- [✓ Selected Tables Only]
- Selected Views Only

**Website**

Name: NorthwindWeb

Directory: C:\inetpub\wwwroot\ browse...

**Business Layer and Data Layer**

Namespace: Northwind

Language: C#

Figure 25 Code Settings Tab – Selected Tables Only

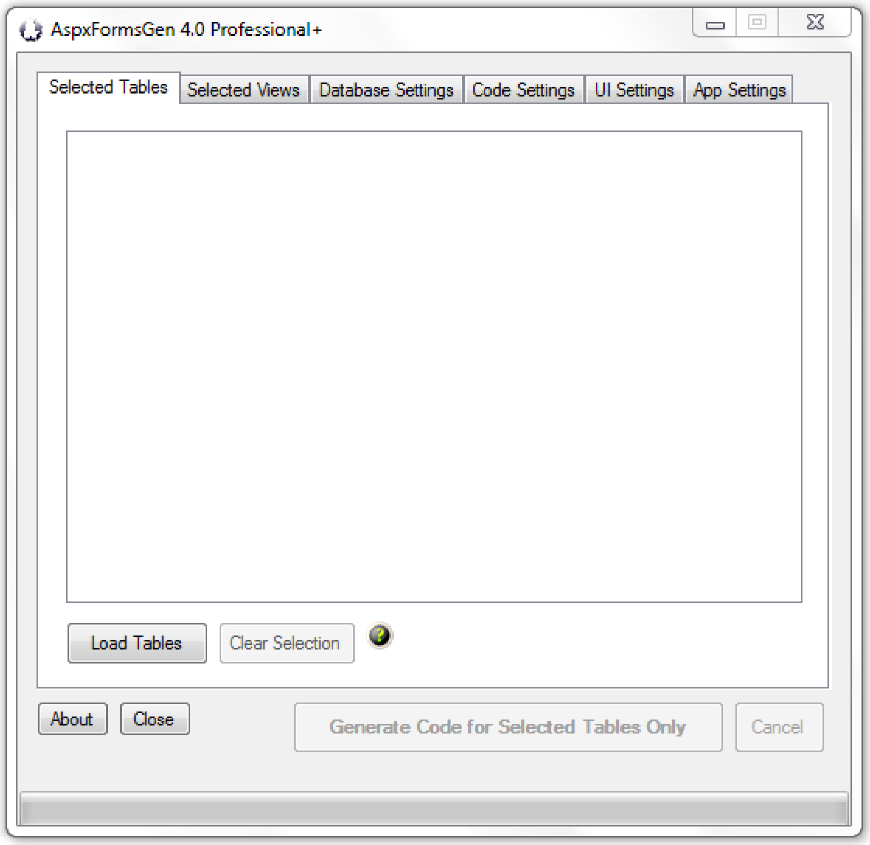
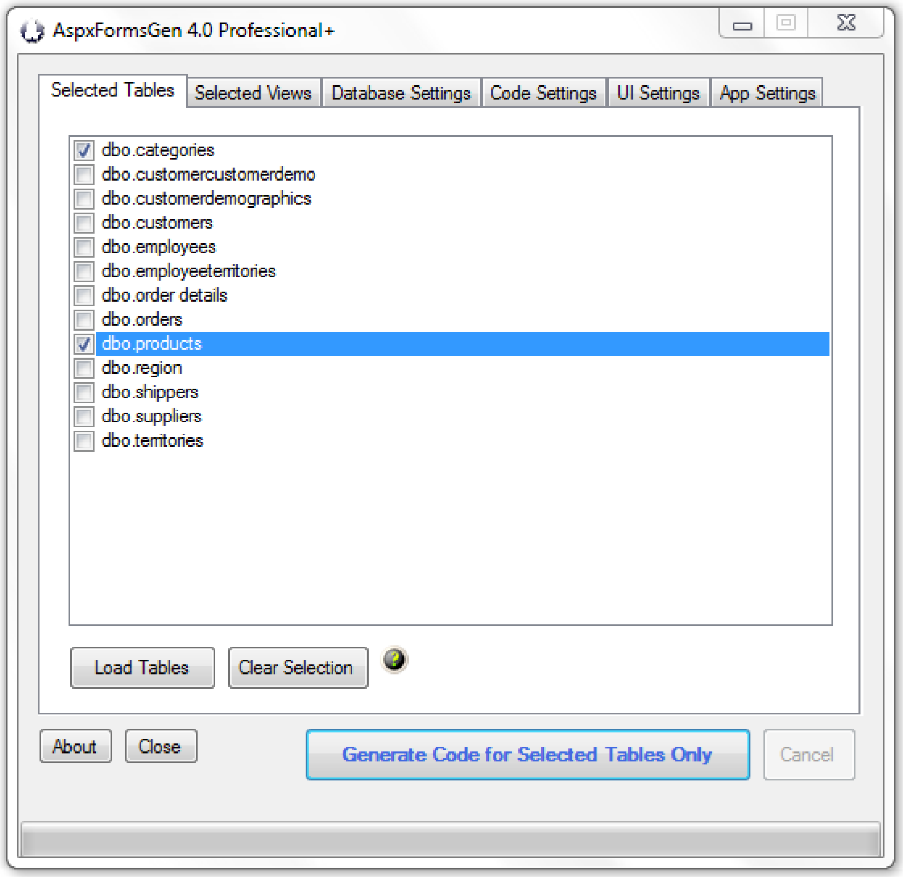
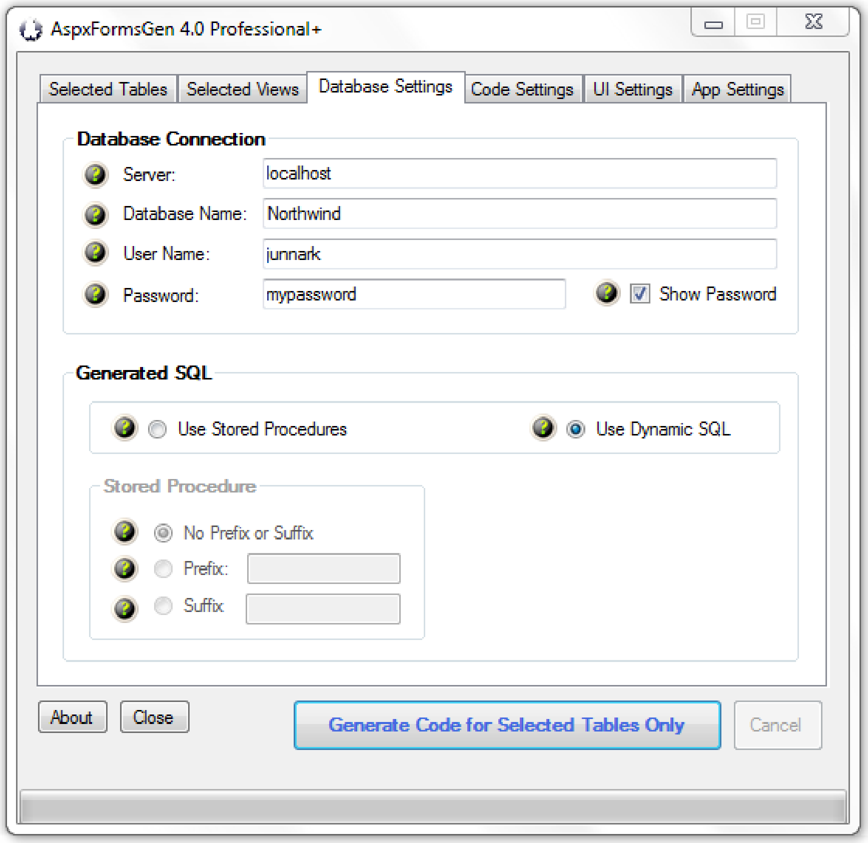

AspxFormsGen 4.0 Professional+

Figure 26 Selected Tables Tab



- [✓] dbo.categories
- [ ] dbo.customercustomerdemo
- [ ] dbo.customerdemographics
- [ ] dbo.customers
- [ ] dbo.employees
- [ ] dbo.employeeterritories
- [ ] dbo.order details
- [ ] dbo.orders
- [✓] dbo.products
- [ ] dbo.region
- [ ] dbo.shippers
- [ ] dbo.suppliers
- [ ] dbo.territories

**Figure 27 Load Tables, Select Tables**



Database Connection

Server: localhost

Database Name: Northwind

User Name: junnark

Password: mypassword [✓] Show Password

Generated SQL

( ) Use Stored Procedures [✓ Use Dynamic SQL]

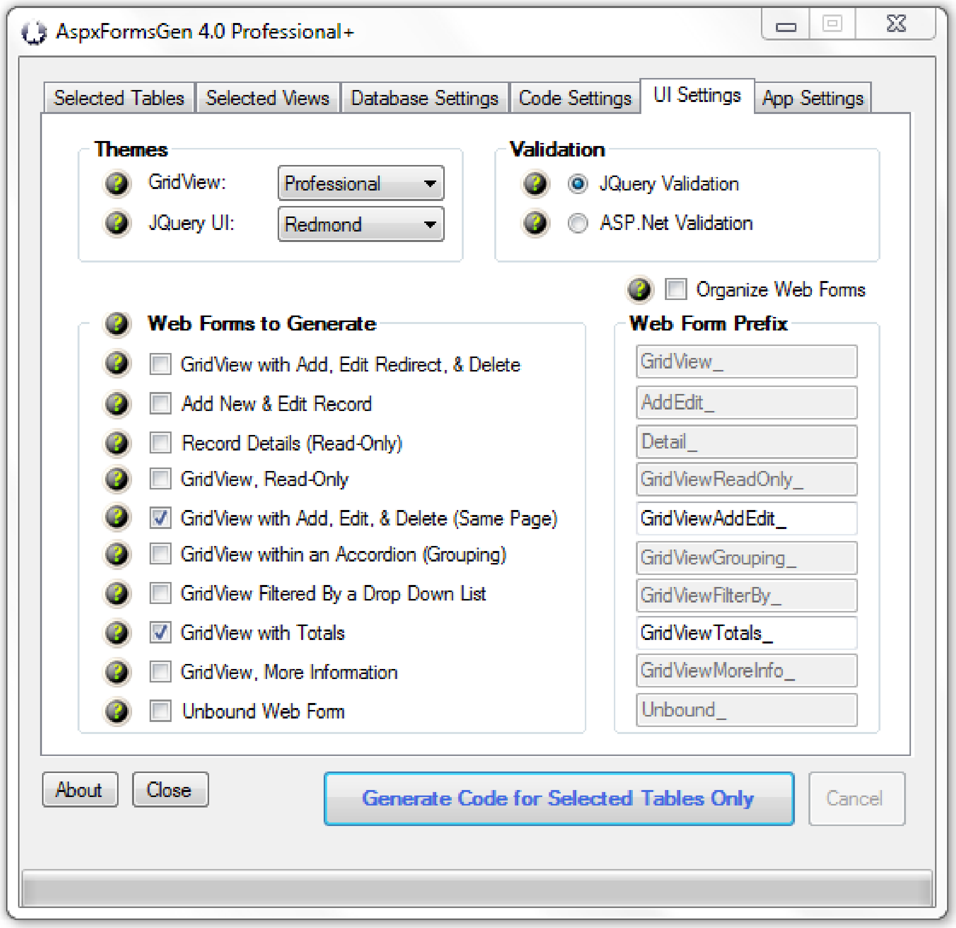
Stored Procedure

[✓ No Prefix or Suffix] ( ) Prefix: ( ) Suffix

**Figure 28 Generated SQL - Use Dynamic SQL**

7. Open the *UI Settings*[1] tab, and then uncheck *Organize Web Forms*[1]. In this tutorial, we would like to generate just a few types of web forms, so we will uncheck a few items under the *Web Forms to Generate* group.

   Notice that as you uncheck *GridView with Add, Edit Redirect, & Delete* or *Add New & Edit Record*, or *Record Details (Read-Only)* under the *Web Forms to Generate* group, other two would also toggle. This is because these 3 options are related. And then Keep the rest of the settings. See Figure 29.



Themes
GridView: Professional
JQuery UI: Redmond

Validation
[✓] JQuery Validation
[ ] ASP.Net Validation

[ ] Organize Web Forms

| Web Forms to Generate | Web Form Prefix |
|---|---|
| [ ] GridView with Add, Edit Redirect, & Delete | GridView_ |
| [ ] Add New & Edit Record | AddEdit_ |
| [ ] Record Details (Read-Only) | Detail_ |
| [ ] GridView, Read-Only | GridViewReadOnly_ |
| [✓] GridView with Add, Edit, & Delete (Same Page) | GridViewAddEdit_ |
| [ ] GridView within an Accordion (Grouping) | GridViewGrouping_ |
| [ ] GridView Filtered By a Drop Down List | GridViewFilterBy_ |
| [✓] GridView with Totals | GridViewTotals_ |
| [ ] GridView, More Information | GridViewMoreInfo_ |
| [ ] Unbound Web Form | Unbound_ |

**Figure 28 UI Settings - Options**

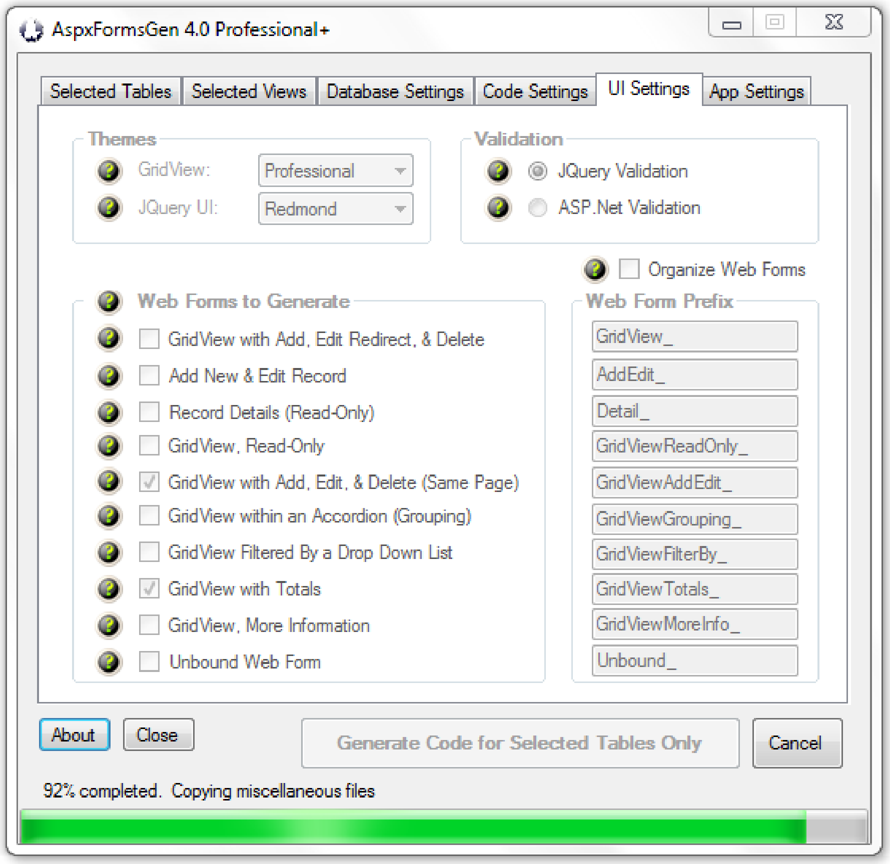
8. Click the *Generate Code for Selected Tables Only* button, AspxFormsGen will start generating code. See Figure 29.

9. When done generating code, a message box is shown. Click OK, and then close AspxFormsGen. See Figure 30.

10. Open *Visual Studio 2010*. On the File menu click *Open Web Site*. See Figure 8 above.

11. Point to the web site directory, and then click *Open*. See Figure 9 above.

12. Let's pause for a moment and look at the generated objects under the *Solution Explorer*. You will notice that no folder was generated for the web forms, instead, a prefix is added to each web form as we specified during code generation. Only two types of web forms where generated. See Figure 31.

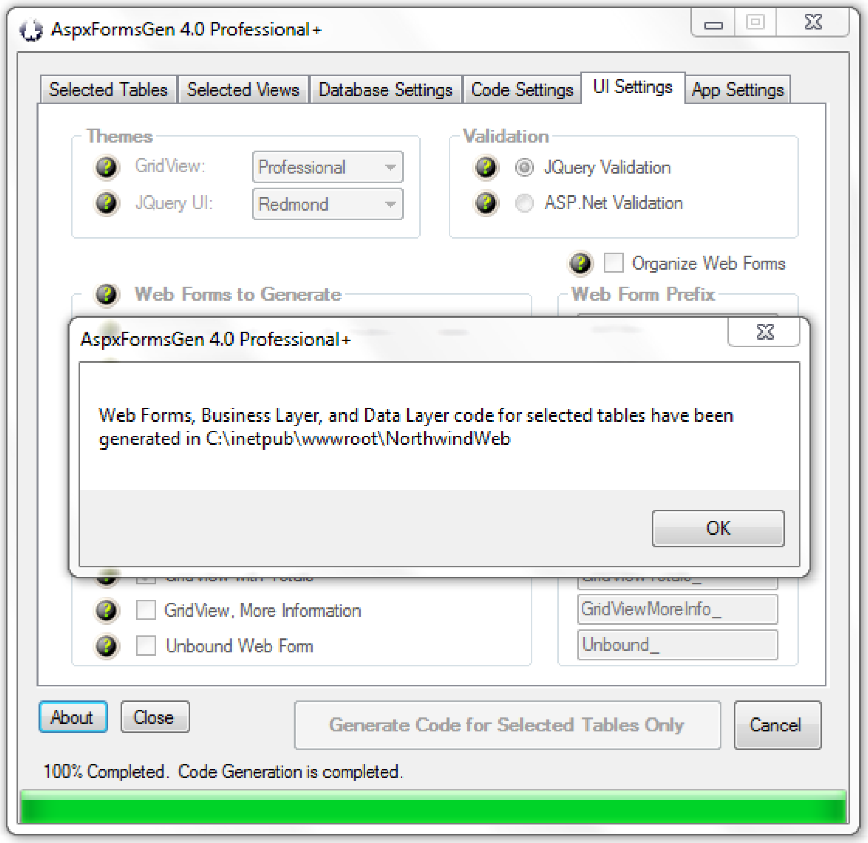


GridView: Professional
JQuery UI: Redmond
(●) JQuery Validation
( ) ASP.Net Validation
[ ] Organize Web Forms

| Web Forms to Generate | Web Form Prefix |
|---|---|
| [ ] GridView with Add, Edit Redirect, & Delete | GridView_ |
| [ ] Add New & Edit Record | AddEdit_ |
| [ ] Record Details (Read-Only) | Detail_ |
| [ ] GridView, Read-Only | GridViewReadOnly_ |
| [✓] GridView with Add, Edit, & Delete (Same Page) | GridViewAddEdit_ |
| [ ] GridView within an Accordion (Grouping) | GridViewGrouping_ |
| [ ] GridView Filtered By a Drop Down List | GridViewFilterBy_ |
| [✓] GridView with Totals | GridViewTotals_ |
| [ ] GridView, More Information | GridViewMoreInfo_ |
| [ ] Unbound Web Form | Unbound_ |

92% completed. Copying miscellaneous files

**Figure 29 Generate Code for Selected Tables Only**

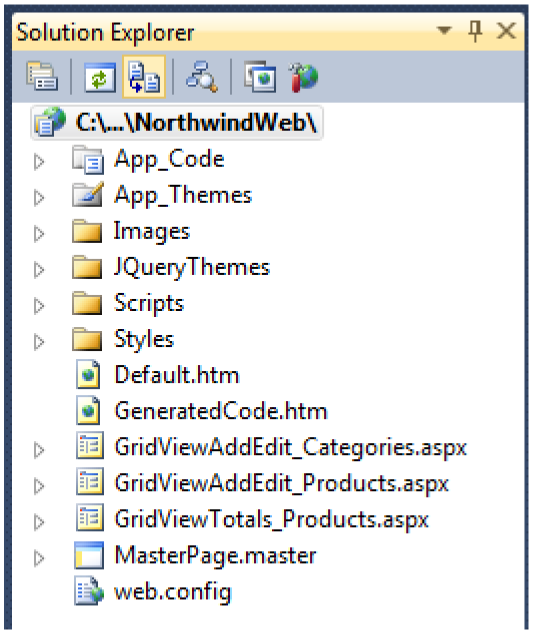


AspxFormsGen 4.0 Professional+

Web Forms, Business Layer, and Data Layer code for selected tables have been generated in C:\inetpub\wwwroot\NorthwindWeb

100% Completed. Code Generation is completed.

**Figure 30 Done Generating Code for Selected Tables Only**

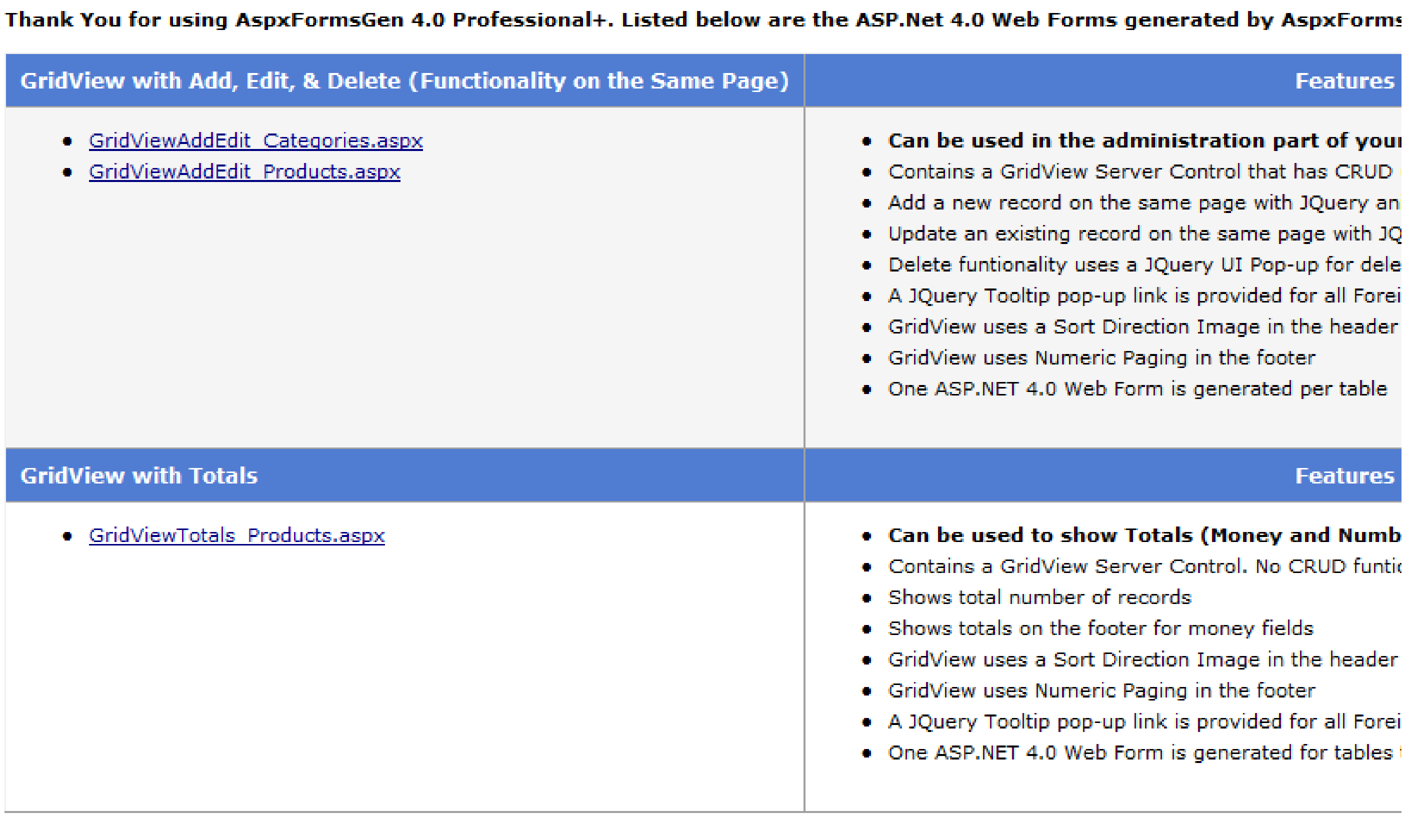


C:\...\NorthwindWeb\
- App_Code
- App_Themes
- Images
- JQueryThemes
- Scripts
- Styles
- Default.htm
- GeneratedCode.htm
- GridViewAddEdit_Categories.aspx
- GridViewAddEdit_Products.aspx
- GridViewTotals_Products.aspx
- MasterPage.master
- web.config

**Figure 31 Generated Web Site**

13. Set the *default.htm* as Start page. See Figure 10 above.

14. Run the web site by pressing *F5*. You will now see a different list. See Figure 32.



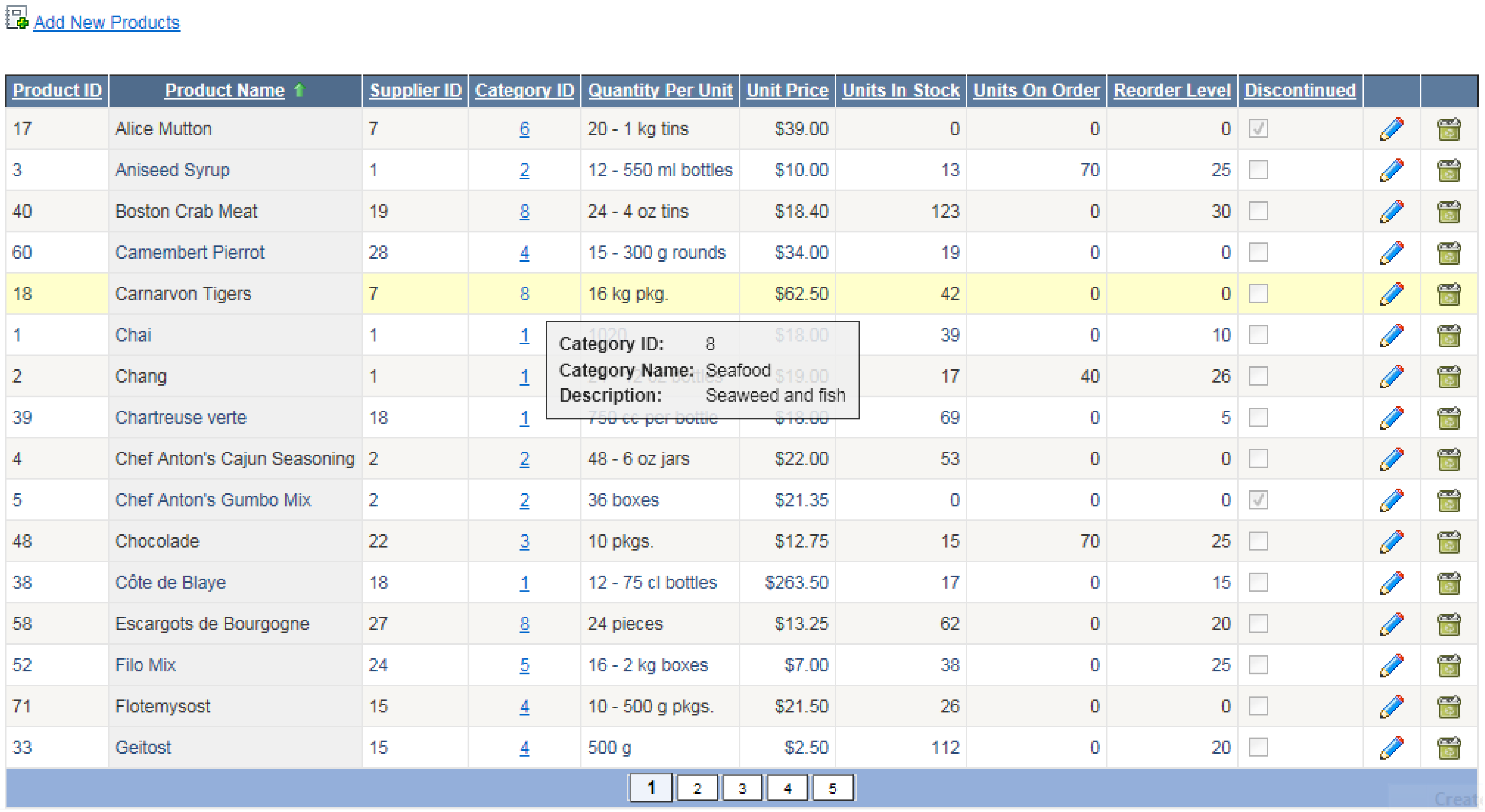
**Thank You for using AspxFormsGen 4.0 Professional+. Listed below are the ASP.Net 4.0 Web Forms generated by AspxForms**

| GridView with Add, Edit, & Delete (Functionality on the Same Page) | Features |
|---|---|
| • GridViewAddEdit_Categories.aspx<br>• GridViewAddEdit_Products.aspx | • **Can be used in the administration part of you**<br>• Contains a GridView Server Control that has CRUD<br>• Add a new record on the same page with JQuery an<br>• Update an existing record on the same page with JQ<br>• Delete funtionality uses a JQuery UI Pop-up for dele<br>• A JQuery Tooltip pop-up link is provided for all Forei<br>• GridView uses a Sort Direction Image in the header<br>• GridView uses Numeric Paging in the footer<br>• One ASP.NET 4.0 Web Form is generated per table |
| **GridView with Totals** | **Features** |
| • GridViewTotals_Products.aspx | • **Can be used to show Totals (Money and Numb**<br>• Contains a GridView Server Control. No CRUD funti<br>• Shows total number of records<br>• Shows totals on the footer for money fields<br>• GridView uses a Sort Direction Image in the header<br>• GridView uses Numeric Paging in the footer<br>• A JQuery Tooltip pop-up link is provided for all Forei<br>• One ASP.NET 4.0 Web Form is generated for tables |

**Figure 32 List of Generated Web Forms**

15. Notice that we only generated *GridView with Add, Edit, & Delete (Functionality on the Same Page)* [1] and the *GridView with Totals*[1] type web forms.

16. Let's preview one of the generated web forms. Click on the *GridViewAddEdit_Products.aspx* link. See Figure 33.


Add New Products

| Product ID | Product Name | Supplier ID | Category ID | Quantity Per Unit | Unit Price | Units In Stock | Units On Order | Reorder Level | Discontinued | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 17 | Alice Mutton | 7 | 6 | 20 - 1 kg tins | $39.00 | 0 | 0 | 0 | ✓ | ✎ | 🗑 |
| 3 | Aniseed Syrup | 1 | 2 | 12 - 550 ml bottles | $10.00 | 13 | 70 | 25 | ☐ | ✎ | 🗑 |
| 40 | Boston Crab Meat | 19 | 8 | 24 - 4 oz tins | $18.40 | 123 | 0 | 30 | ☐ | ✎ | 🗑 |
| 60 | Camembert Pierrot | 28 | 4 | 15 - 300 g rounds | $34.00 | 19 | 0 | 0 | ☐ | ✎ | 🗑 |
| 18 | Carnarvon Tigers | 7 | 8 | 16 kg pkg. | $62.50 | 42 | 0 | 0 | ☐ | ✎ | 🗑 |
| 1 | Chai | 1 | 1 | [illegible] | $18.00 | 39 | 0 | 10 | ☐ | ✎ | 🗑 |
| 2 | Chang | 1 | 1 | [illegible] | $19.00 | 17 | 40 | 26 | ☐ | ✎ | 🗑 |
| 39 | Chartreuse verte | 18 | 1 | 750 cc per bottle | $18.00 | 69 | 0 | 5 | ☐ | ✎ | 🗑 |
| 4 | Chef Anton's Cajun Seasoning | 2 | 2 | 48 - 6 oz jars | $22.00 | 53 | 0 | 0 | ☐ | ✎ | 🗑 |
| 5 | Chef Anton's Gumbo Mix | 2 | 2 | 36 boxes | $21.35 | 0 | 0 | 0 | ✓ | ✎ | 🗑 |
| 48 | Chocolade | 22 | 3 | 10 pkgs. | $12.75 | 15 | 70 | 25 | ☐ | ✎ | 🗑 |
| 38 | Côte de Blaye | 18 | 1 | 12 - 75 cl bottles | $263.50 | 17 | 0 | 15 | ☐ | ✎ | 🗑 |
| 58 | Escargots de Bourgogne | 27 | 8 | 24 pieces | $13.25 | 62 | 0 | 20 | ☐ | ✎ | 🗑 |
| 52 | Filo Mix | 24 | 5 | 16 - 2 kg boxes | $7.00 | 38 | 0 | 25 | ☐ | ✎ | 🗑 |
| 71 | Flotemysost | 15 | 4 | 10 - 500 g pkgs. | $21.50 | 26 | 0 | 0 | ☐ | ✎ | 🗑 |
| 33 | Geitost | 15 | 4 | 500 g | $2.50 | 112 | 0 | 20 | ☐ | ✎ | 🗑 |

1 2 3 4 5

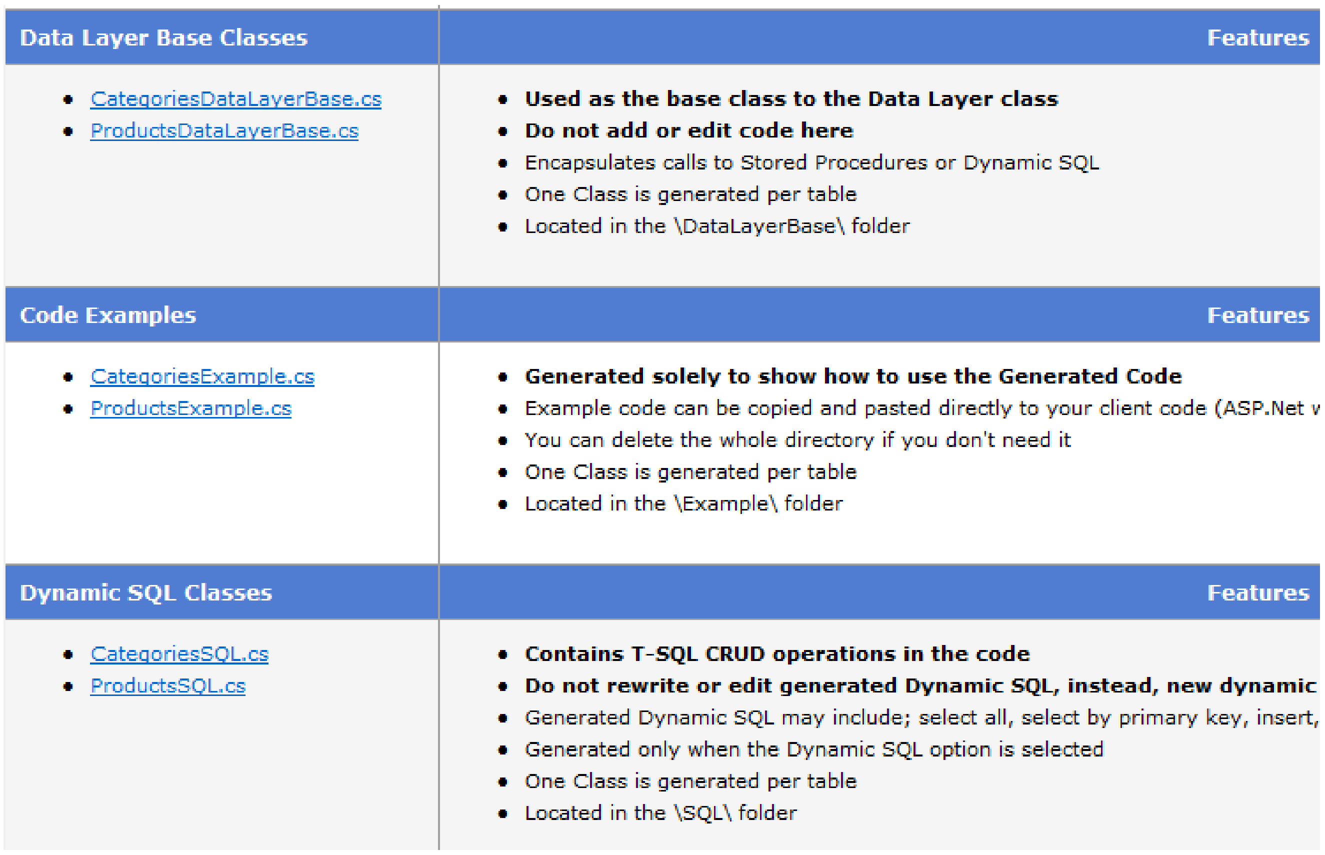
Category ID: 8
Category Name: Seafood
Description: Seaweed and fish

Figure 33 GridViewAddEdit Web Form

17. Unlike the *GridView with Add, Edit Redirect, & Delete*[1] web form seen in Figure 12, you can Add a New Record and Update an existing record on this same web page. Go ahead and play around using the web page's functionalities.

18. Close the web page and go back to Visual Studio 2010. From the *Solution Explorer*, right-click on the *GeneratedCode.htm* and then click *Set As Start Page*. See Figure 14 above.

19. Run Visual Studio by pressing *F5*. You will see a list of all the generated middle-tier classes, data-tier classes, and dynamic SQL classes instead of stored procedures as we specified in AspxFormsGen 4.0. See Figure 34. You can hover over each of the link to see where each file is located.



| Data Layer Base Classes | Features |
|---|---|
| • CategoriesDataLayerBase.cs<br>• ProductsDataLayerBase.cs | • **Used as the base class to the Data Layer class**<br>• **Do not add or edit code here**<br>• Encapsulates calls to Stored Procedures or Dynamic SQL<br>• One Class is generated per table<br>• Located in the \DataLayerBase\ folder |
| **Code Examples** | **Features** |
| • CategoriesExample.cs<br>• ProductsExample.cs | • **Generated solely to show how to use the Generated Code**<br>• Example code can be copied and pasted directly to your client code (ASP.Net v<br>• You can delete the whole directory if you don't need it<br>• One Class is generated per table<br>• Located in the \Example\ folder |
| **Dynamic SQL Classes** | **Features** |
| • CategoriesSQL.cs<br>• ProductsSQL.cs | • **Contains T-SQL CRUD operations in the code**<br>• **Do not rewrite or edit generated Dynamic SQL, instead, new dynamic**<br>• Generated Dynamic SQL may include; select all, select by primary key, insert,<br>• Generated only when the Dynamic SQL option is selected<br>• One Class is generated per table<br>• Located in the \SQL\ folder |

**Figure 34 List of Generated Code, Dynamic SQL List**

20. Close the web page and go back to *Visual Studio 2010*. The generated middle-tier and data-tier classes are the same as seen in Figure 16 above and can also be found under the *App_Code* folder. However, a new folder called "*SQL*" has been added. This is where all the Dynamic SQL code was generated into. See Figure 35.

21. If you open *MS SQL Server Management Studio*, under the *Stored Procedures*[1] node, no stored procedures were generated, this is because AspxFormsGen 4.0 generated *Dynamic SQL*[1] instead, just like we specified.

22. You can find a deeper discussion on the generated web forms, middle-tier, data-tier, stored procedures or dynamic SQL under the Generated Code below. For now, this will be the end of this tutorial.

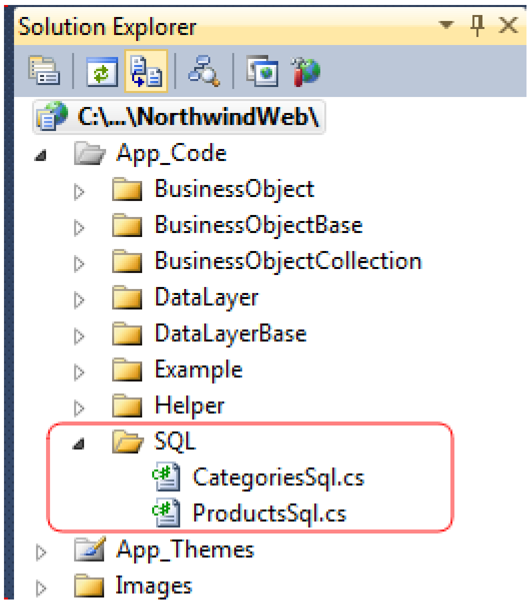


**Figure 35 Dynamic SQL Folder**

## For Selected Views Only [1]

The *Selected Views Only* option generates objects for selected views only, in the respective database.

1. To follow this tutorial make sure to delete the *NorthwindWeb* web site we generated earlier to get a fresh start. Also delete all the Stored Procedures (if any) that was generated by the earlier tutorial.

2. Open AspxFormsGen 4.0. By now you will notice that the last settings were saved. We could easily use the **One Click** feature by clicking the *Generate…* button right away, but don't for now.

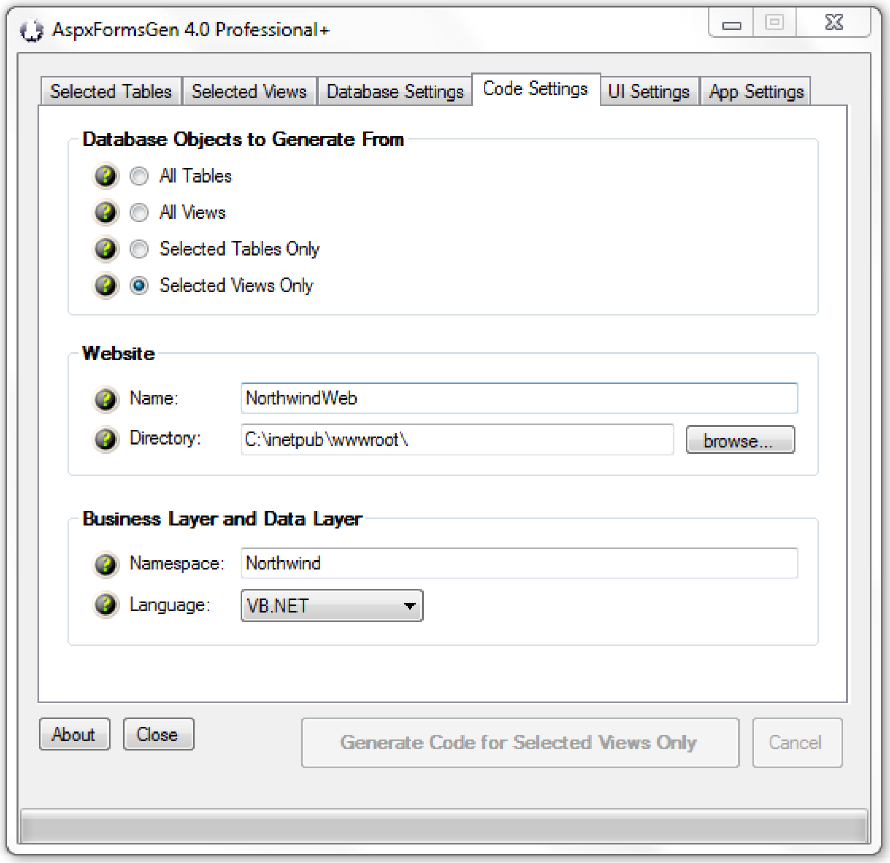
3. Open the *Code Settings* tab then select *Selected Views Only* under the *Database Objects to Generate From* and change the *Language* under *Business Layer and Data Layer group* to VB.NET. Keep the rest of the settings on this tab. See Figure 36. Selecting the *Selected Views Only* option will open the *Selected Views* tab by default; you can change this behavior under *the App Settings* tab if you want, simply uncheck the *Automatically Open Selected Tables or Selected View* tab.

   **Note:** If you get redirected to the *Selected Views* tab, simply go back to the *Code Settings* tab and then under the *Database Objects to Generate From* and change the *Language* under *Business Layer and Data Layer group* to VB.NET. See Figure 36. Now go back to the *Selected Views* tab.

4. The *Load Views* button is now enabled. See Figure 37.

5. Click the *Load Views* button, and then select the following views as shown in Figure 38.

6. Open the *Database Settings* tab and select *Use Stored Procedures* under the *Generated SQL* group, and then choose the *Prefix* option under *Stored procedures*, enter a *Prefix* for the *Stored Procedures*.. See Figure 39.



**Database Objects to Generate From**

( ) All Tables
( ) All Views
( ) Selected Tables Only
(●) Selected Views Only

**Website**

Name: NorthwindWeb
Directory: C:\inetpub\wwwroot\

**Business Layer and Data Layer**

Namespace: Northwind
Language: VB.NET

Figure 36 Code Settings Tab – Selected Views Only

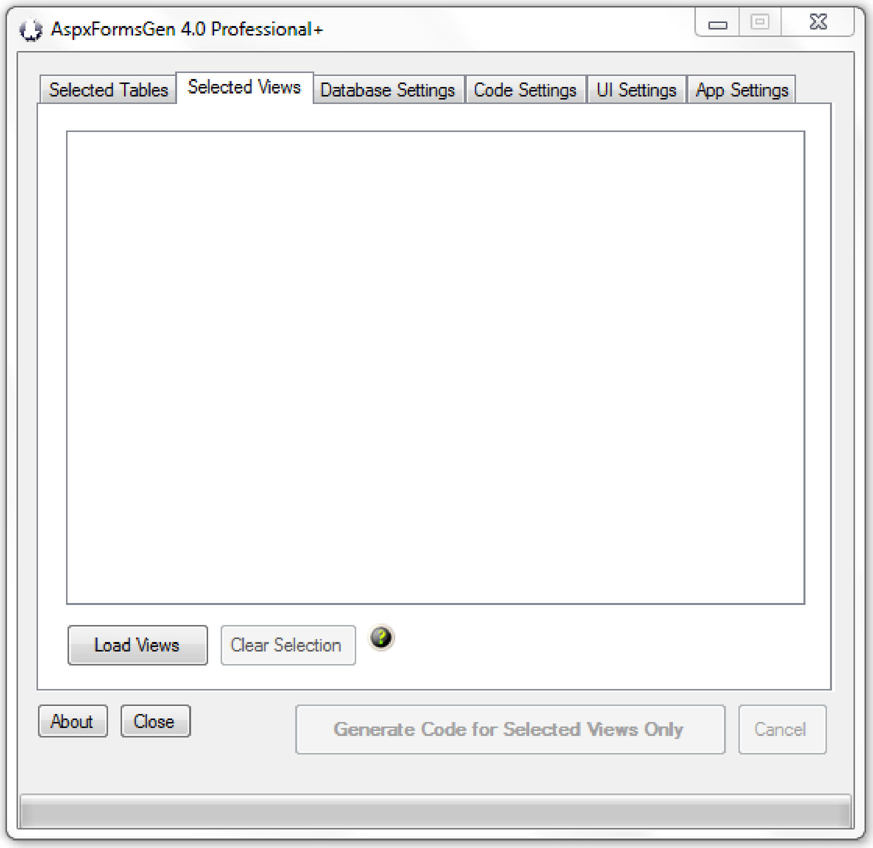
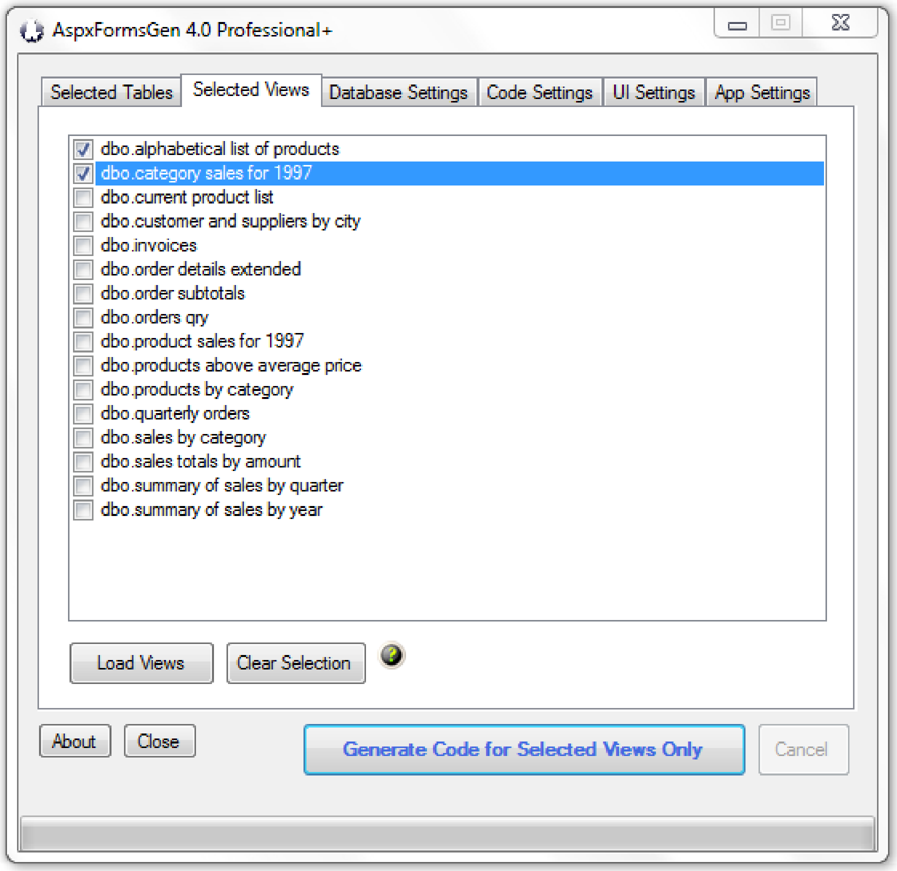
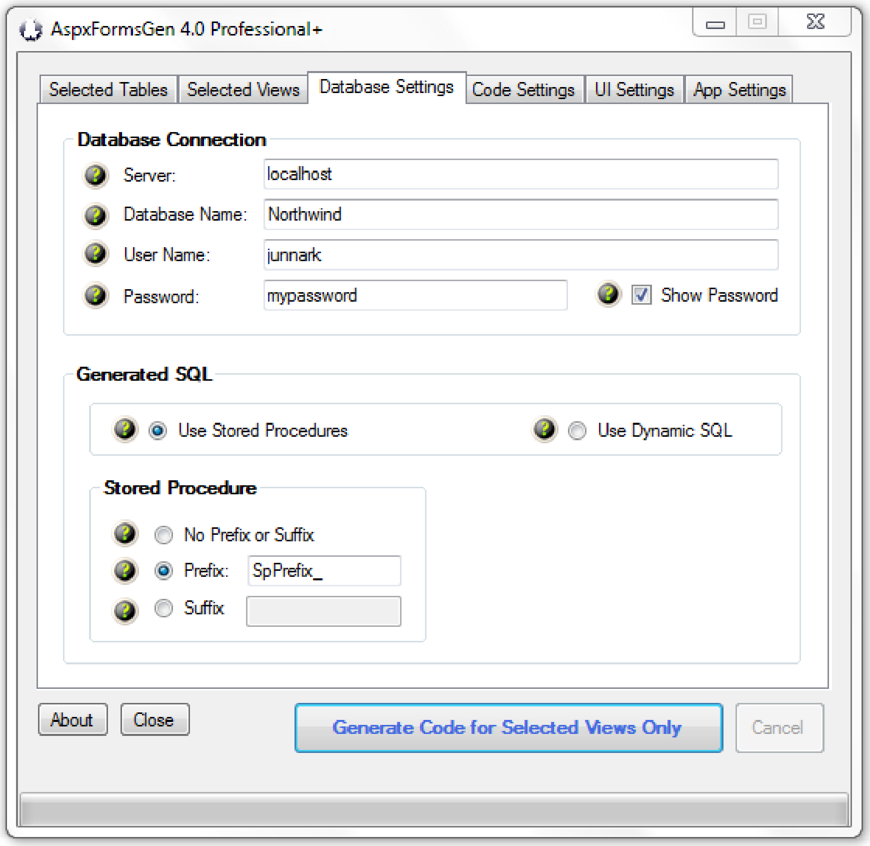


Figure 37 Selected Views Tab



- [✓] dbo.alphabetical list of products
- [✓] dbo.category sales for 1997
- [ ] dbo.current product list
- [ ] dbo.customer and suppliers by city
- [ ] dbo.invoices
- [ ] dbo.order details extended
- [ ] dbo.order subtotals
- [ ] dbo.orders qry
- [ ] dbo.product sales for 1997
- [ ] dbo.products above average price
- [ ] dbo.products by category
- [ ] dbo.quarterly orders
- [ ] dbo.sales by category
- [ ] dbo.sales totals by amount
- [ ] dbo.summary of sales by quarter
- [ ] dbo.summary of sales by year

**Figure 38 Load Views, Select Views**



**Database Connection**

Server: localhost
Database Name: Northwind
User Name: junnark
Password: mypassword
[✓] Show Password

**Generated SQL**

[✓ Use Stored Procedures] Use Dynamic SQL

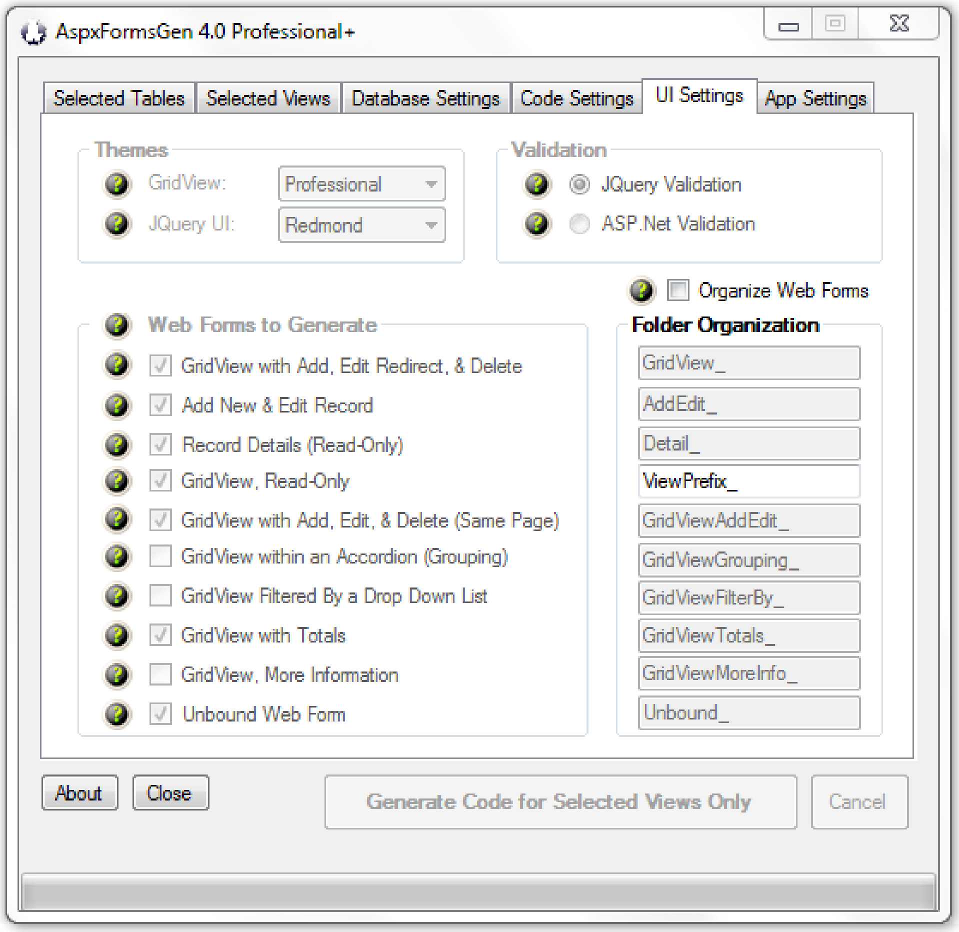
**Stored Procedure**

No Prefix or Suffix
[✓ Prefix:] SpPrefix_
Suffix

**Figure 39 Database Settings – Stored Procedures with Prefix**

7. Open the *UI Settings* tab. You will notice that everything is disabled except for the *Organize Web Form* check box and the *GridView, Read-Only's* respective *Folder Organization/Web Form Prefix*. This is because views are **read-only**, that's why the only web forms that will be generated are read-only web forms. In short, there will be no CRUD operation for the generated web forms as well as the generated middle-tier, data-tier, and stored procedures or dynamic SQL.

   Uncheck *Organize Web Forms.* And then change the text "*GridViewReadOnly_"* to "*ViewPrefix_*", of course you can put any text here. See Figure 40.



Themes

| | |
|---|---|
| GridView: | Professional |
| JQuery UI: | Redmond |

Validation

[✓ JQuery Validation] ASP.Net Validation

☐ Organize Web Forms

| Web Forms to Generate | Folder Organization |
|---|---|
| ✓ GridView with Add, Edit Redirect, & Delete | GridView_ |
| ✓ Add New & Edit Record | AddEdit_ |
| ✓ Record Details (Read-Only) | Detail_ |
| ✓ GridView, Read-Only | ViewPrefix_ |
| ✓ GridView with Add, Edit, & Delete (Same Page) | GridViewAddEdit_ |
| ☐ GridView within an Accordion (Grouping) | GridViewGrouping_ |
| ☐ GridView Filtered By a Drop Down List | GridViewFilterBy_ |
| ✓ GridView with Totals | GridViewTotals_ |
| ☐ GridView, More Information | GridViewMoreInfo_ |
| ✓ Unbound Web Form | Unbound_ |

**Figure 40 Code Settings - Options**

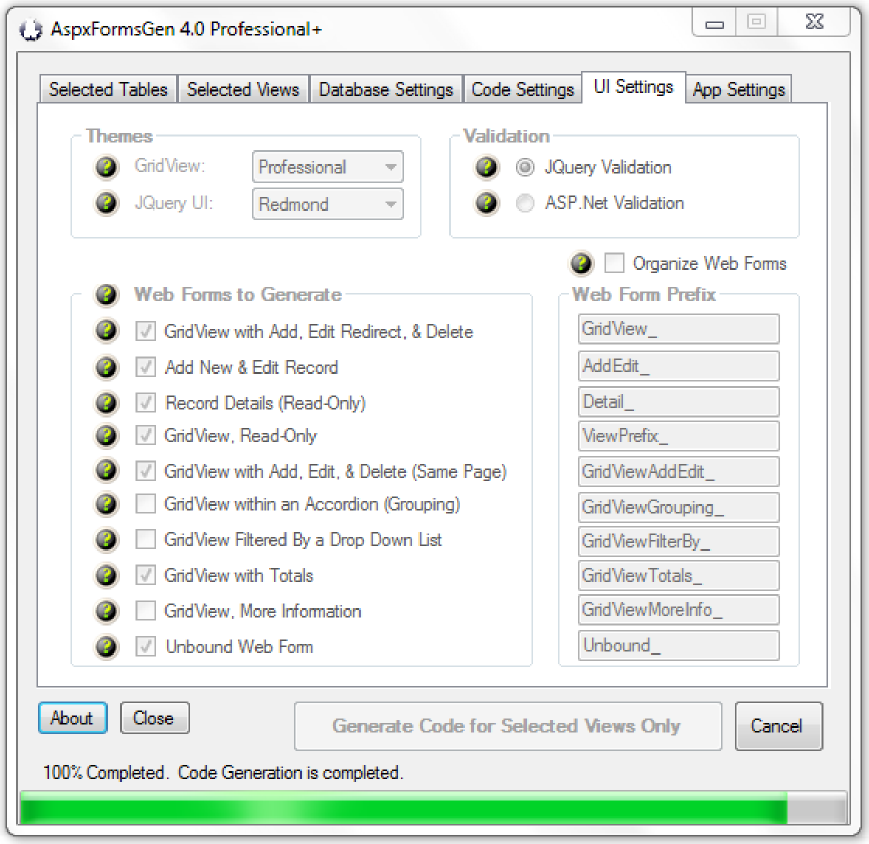
8. Click the *Generate Code for Selected Views Only* button, AspxFormsGen will start generating code. See Figure 41.

9. When done generating code, a message box is shown. Click OK, and then close AspxFormsGen. See Figure 42.

10. Open *Visual Studio 2010*. On the File menu click *Open Web Site*. See Figure 8 above.

11. Point to the web site directory, and then click *Open*. See Figure 9 above.

12. Let's pause for a moment and look at the generated objects under the *Solution Explorer*. You will notice that no folder was generated for the web forms, instead, a prefix is added to each web form as we specified during code generation. Only two types of web forms where generated. See Figure 43.



AspxFormsGen 4.0 Professional+

Selected Tables | Selected Views | Database Settings | Code Settings | UI Settings | App Settings

Themes
GridView: Professional
JQuery UI: Redmond

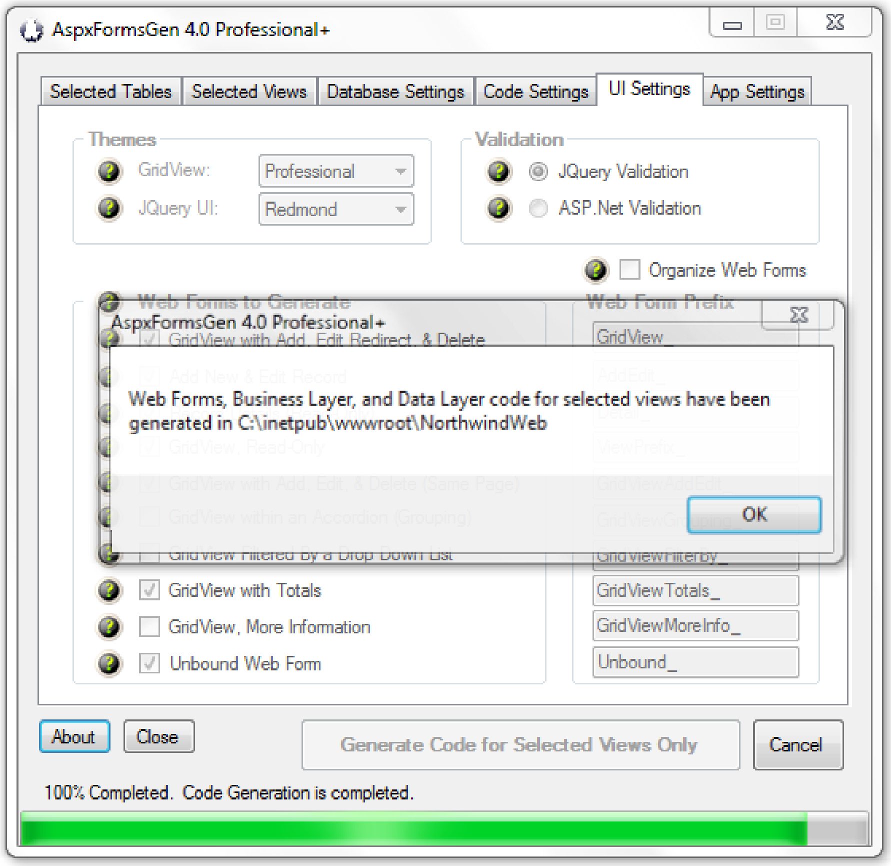
Validation
[✓ JQuery Validation]
ASP.Net Validation

☐ Organize Web Forms

| Web Forms to Generate | Web Form Prefix |
|---|---|
| ✓ GridView with Add, Edit Redirect, & Delete | GridView_ |
| ✓ Add New & Edit Record | AddEdit_ |
| ✓ Record Details (Read-Only) | Detail_ |
| ✓ GridView, Read-Only | ViewPrefix_ |
| ✓ GridView with Add, Edit, & Delete (Same Page) | GridViewAddEdit_ |
| ☐ GridView within an Accordion (Grouping) | GridViewGrouping_ |
| ☐ GridView Filtered By a Drop Down List | GridViewFilterBy_ |
| ✓ GridView with Totals | GridViewTotals_ |
| ☐ GridView, More Information | GridViewMoreInfo_ |
| ✓ Unbound Web Form | Unbound_ |

100% Completed. Code Generation is completed.

**Figure 41 Generate Code For Selected Views Only**



AspxFormsGen 4.0 Professional+

Selected Tables | Selected Views | Database Settings | Code Settings | UI Settings | App Settings

Themes
GridView: Professional
JQuery UI: Redmond

Validation
[✓ JQuery Validation]
ASP.Net Validation

☐ Organize Web Forms

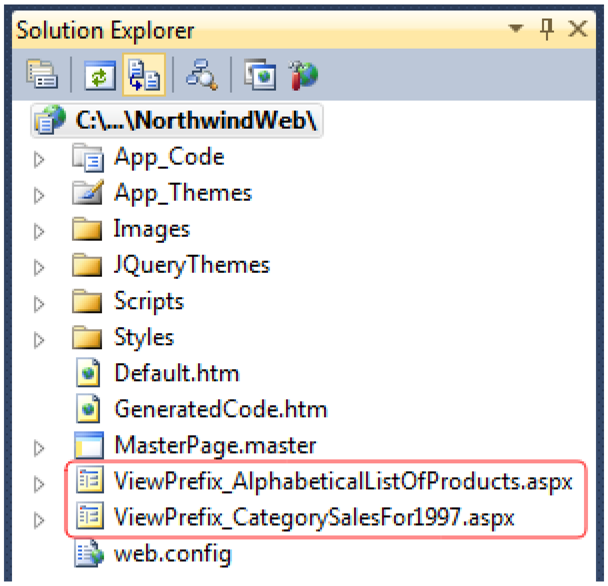
AspxFormsGen 4.0 Professional+

Web Forms, Business Layer, and Data Layer code for selected views have been generated in C:\inetpub\wwwroot\NorthwindWeb

| Web Forms to Generate | Web Form Prefix |
|---|---|
| ✓ GridView with Totals | GridViewTotals_ |
| ☐ GridView, More Information | GridViewMoreInfo_ |
| ✓ Unbound Web Form | Unbound_ |

100% Completed. Code Generation is completed.

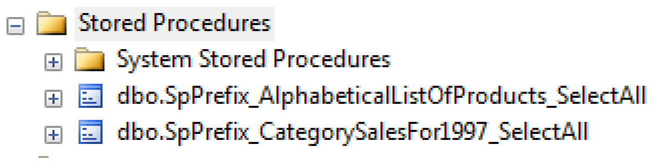
**Figure 42 Done Generating Code For Selected Views Only**



C:\...\NorthwindWeb\
- App_Code
- App_Themes
- Images
- JQueryThemes
- Scripts
- Styles
- Default.htm
- GeneratedCode.htm
- MasterPage.master
- ViewPrefix_AlphabeticalListOfProducts.aspx
- ViewPrefix_CategorySalesFor1997.aspx
- web.config

**Figure 43 Generated Web Site**

13. Open *MS SQL Server Management Studio* and then navigate to the *Stored Procedures* node of the respective database. Notice that the stored procedures that were generated have the prefix we specified in AspxFormsGen 4.0. See Figure 44.

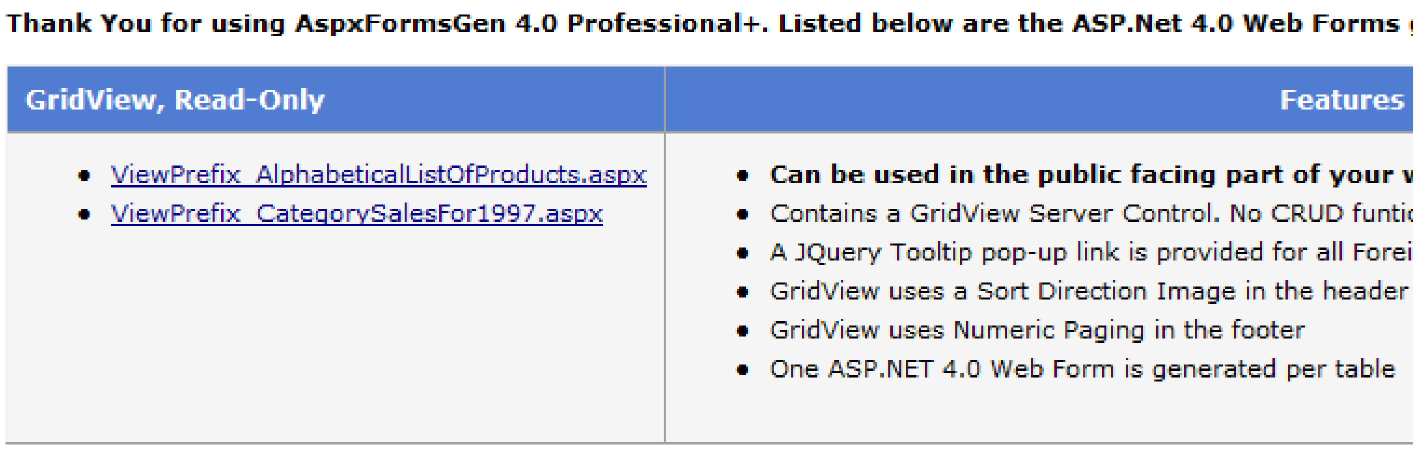


Stored Procedures
- System Stored Procedures
- dbo.SpPrefix_AlphabeticalListOfProducts_SelectAll
- dbo.SpPrefix_CategorySalesFor1997_SelectAll

**Figure 44 Generated Stored Procedures with Prefix**

14. Set the *default.htm* as Start page. See Figure 10 above.

15. Run the web site by pressing F5. You will now see a different list. See Figure 45.



**Thank You for using AspxFormsGen 4.0 Professional+. Listed below are the ASP.Net 4.0 Web Forms**

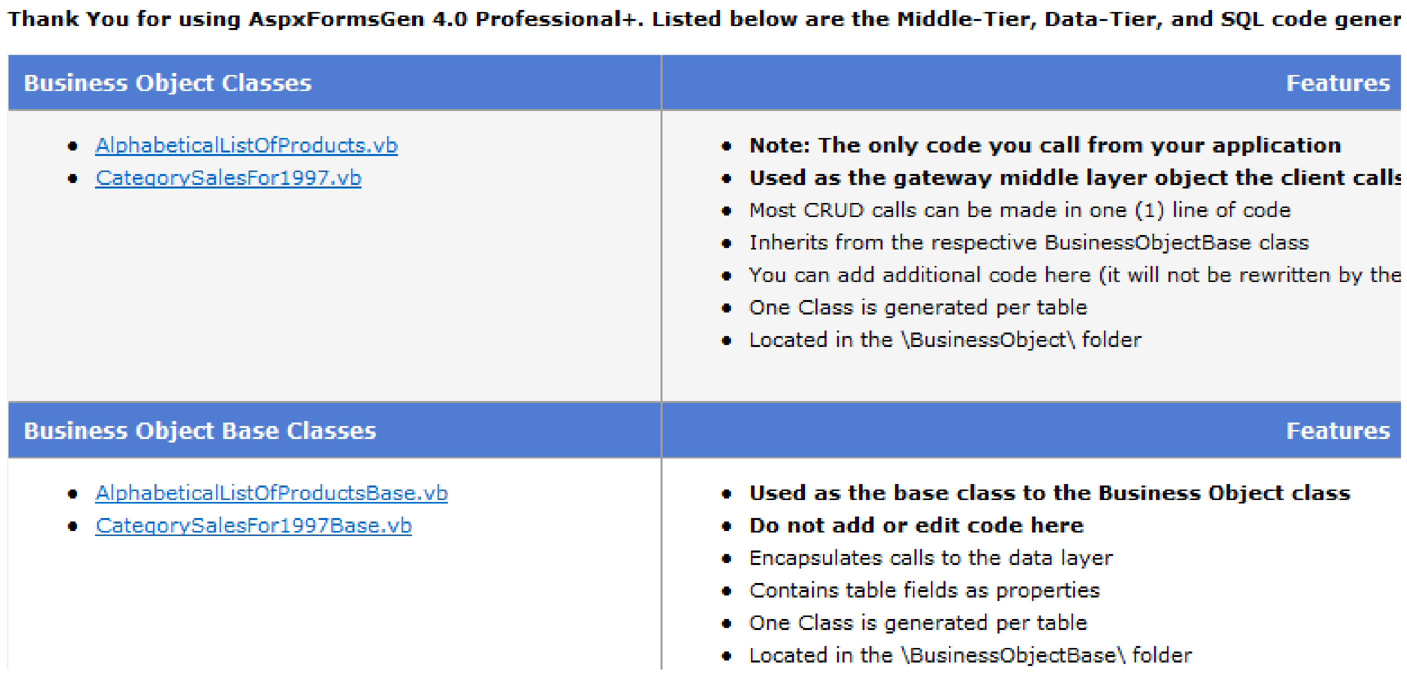
| GridView, Read-Only | Features |
|---|---|
| • ViewPrefix_AlphabeticalListOfProducts.aspx<br>• ViewPrefix_CategorySalesFor1997.aspx | • **Can be used in the public facing part of your**<br>• Contains a GridView Server Control. No CRUD funti<br>• A JQuery Tooltip pop-up link is provided for all Forei<br>• GridView uses a Sort Direction Image in the header<br>• GridView uses Numeric Paging in the footer<br>• One ASP.NET 4.0 Web Form is generated per table |

**Figure 45 List of Generated Web Forms**

16. Notice that the generated web forms have a prefix as we specified in AspxFormsGen 4.0 under the *UI Settings tab*.

17. Open *Visual Studio 2010*. From the *Solution Explorer*, right-click on the *GeneratedCode.htm* and then click *Set As Start Page*. See Figure 14 above.

18. Run Visual Studio by pressing F5. You will see a list of all the generated middle-tier classes, data-tier classes, and stored procedures with prefixes as we specified in AspxFormsGen 4.0. See Figure 46. You can hover over each of the link to see where each file is located.



**Thank You for using AspxFormsGen 4.0 Professional+. Listed below are the Middle-Tier, Data-Tier, and SQL code gener**

| Business Object Classes | Features |
|---|---|
| • AlphabeticalListOfProducts.vb<br>• CategorySalesFor1997.vb | • **Note: The only code you call from your application**<br>• **Used as the gateway middle layer object the client calls**<br>• Most CRUD calls can be made in one (1) line of code<br>• Inherits from the respective BusinessObjectBase class<br>• You can add additional code here (it will not be rewritten by the<br>• One Class is generated per table<br>• Located in the \BusinessObject\ folder |
| **Business Object Base Classes** | **Features** |
| • AlphabeticalListOfProductsBase.vb<br>• CategorySalesFor1997Base.vb | • **Used as the base class to the Business Object class**<br>• **Do not add or edit code here**<br>• Encapsulates calls to the data layer<br>• Contains table fields as properties<br>• One Class is generated per table<br>• Located in the \BusinessObjectBase\ folder |

**Figure 46 List of Generated Code, In VB.NET**

19. Close the web page, for now this will be the end of this tutorial. You can find a deeper discussion on the generated web forms, middle-tier, data-tier, stored procedures or dynamic SQL under the Generated Code below.

# Generated Code

AspxFormsGen 4.0 generates ASP.NET 4.0 web forms, middle-tier and data tier classes, example classes, dynamic SQL classes, and stored procedures. All code other than the stored procedures is generated in either C# or VB.NET. AspxFormsGen 4.0 can generate from Tables or Views as source. AspxFormsGen 4.0 is made up of 2 main engines, the AspxFormsGen engine which generates the web forms and the AspxCodeGen engine, which generates the middle-tier, data-tier, dynamic SQL and stored procedures. This portion will discuss the parts of the generated code.

AspxFormsGen 4.0 generates a 3-tier structure web site.

1. User Interface (Front-end) – ASP.NET 4.0 Web Forms (client).
2. Business Objects (Middle Layer) – Middle Tier Classes.
3. Data Layer – Data Tier Classes, Stored Procedures or Dynamic SQL Classes.

## ASP.NET 4. 0 Web Forms

The following web forms will be generated when they are selected (checked) under the *UI Settings* tab in the *Web Forms to Generate* group.

1. GridView with Add, Edit Redirect & Delete[1]
2. Add New & Edit Record [1]
3. Record Details (Read Only) [1]
4. GridView, Read-Only: [1] **Note:** The only web form type generated when *All Views* or *Selected View Only* is selected.
5. GridView with Add, Edit, & Delete (Same Page) [1]
6. GridView within an Accordion (Grouping) [1]
7. GridView Filtered By a Drop Down List[1]
8. GridView with Totals[1]
9. GridView, More Information[1]
10. Unbound Web Form (**Note:** the only web form generated for the Express Edition)

Each one of the 10 web form types above will either be generated organized in folders or named with prefix and placed in the root website directory. You can organize them into folders by checking the *Organize Web Forms* check box under the *UI Settings*. If this setting is unchecked, the generated web forms will be placed in the root web site directory, and each web form name will be prefixed by the respective prefixes which are found under the *Web Form Prefix* group. Please see the description of the respective web form type above under the *UI Settings* discussion.

## Middle-Tier Classes

The middle-tier class encapsulates the respective data layer (data-tier classes). These are the classes that you should access from your client code. The middle-tier class makes it simple for any client (e.g. ASP.NET web forms, win forms, Silverlight, WCF, web services, etc.) to access the database without having to know how the operation (or business process) was accomplished. These middle-tier objects are not only for use by the generated web site, you can also use them as an API to other projects that accesses the same database, simple put these generated objects into a Class Library project and then reference the project from you client program.

**Note:** It is best practice not to access the data layer code directly from your client code.

Below are the middle-tier class types that are generated by the AspxCodeGen engine that is integrated with AspxFormsGen 4.0.

1. **BusinessObject Class:** Your client code should always access this class directly.

    - **Note: The only code you call from your application**
    - **Used as the gateway middle layer object the client calls**
    - Most CRUD calls can be made in one (1) line of code
    - Inherits from the respective *BusinessObjectBase* class
    - You can add additional code here (it will not be rewritten by the generator)
    - One Class is generated per table
    - Located in the *BusinessObject* folder

2. **BusinessObjectBase Class:** Contains all the properties and methods that encapsulate the data layer can be found here.

    - Used as the base class to the Business Object class
    - **Do not add or edit code here**
    - This class is overwritten every time you generate code
    - The methods encapsulates calls to the data layer
    - Contains table fields as properties
    - One Class is generated per table
    - Located in the *BusinessObjectBase* folder

**Methods**

a. **SelectAll:** Selects all records from a specific table or view. **Note:** The only method generated when *All Views* or *Selected Views Only* is selected under *Code Settings.*
b. **SelectByPrimaryKey:** Selects a record by primary key.
c. **SelectDropDownListData:** Selects 2 fields from the specific table for use with a DropDownList control source (or combo box, etc).
d. **SelectCollectionBy Foreign Key:** Selects all records by foreign key.
e. **Insert:** Inserts a record in the table.
f. **Update:** Updates an existing record in the table by primary key.
g. **Delete:** Deletes a record from a table by primary key.
h. **Comparison Methods:** Methods used for sorting.

**Properties**

Each field from a table or view is generated as a property in each *BusinessObjectBase* class. Also, each related table will be a property, e.g. An order (*Order* table) has related customers (*Customer* table), so in the *OrderObjectBase* class a property called *Customers* is generated. The *Customers* property will return all the customers related to this order. The related properties uses lazy initialization.

**3. BusinessObjectCollection Class:** Rather than using the generic List object as a type, use this instead because it's strongly-typed.

- Used as the Collection of the Business Object Class
- Do not add or edit code here
- One Class is generated per table
- Located in the *BusinessObjectCollection* folder

## Data-Tier Classes

The data-tier classes encapsulate calls to the database. These classes are called or accessed by the middle layer code. It encapsulates calls to a stored procedure or dynamic SQL.

**1. DataLayer Class**

- Used as the gateway data layer object the middle tier objects call
- Inherits from the respective *DataLayerBase* class
- You can add additional code here (it will not be rewritten by the generator)
- One Class is generated per table
- Located in the *DataLayer* folder

**2. DataLayerBase Class**

- Used as the base class to the Data Layer class
- **Do not add or edit code here**
- The methods encapsulates calls to Stored Procedures or Dynamic SQL[1]
- One Class is generated per table
- Located in the *DataLayerBase* folder

**Methods**

This class contains identical method names as the *BusinessObjectBase* class. The only difference is that the methods here encapsulate calls to Stored Procedures or Dynamic SQL instead. [1]

## Stored Procedures or Dynamic SQL Classes[1]

The AspxCodeGen engine (integrated in AspxFormsGen 4.0) generates stored procedures directly to your database or dynamic SQL classes in the *SQL* folder, under the *App_Code* folder of the generated web site. The difference with Stored Procedures and Dynamic SQL is that, SQL script for stored procedures are in the database, while embedded as string in methods for dynamic SQL.

1. **Stored Procedures**

    - Created in the database and used for CRUD operations
    - Do not rewrite or edit generated stored procedure, instead, add a new one
    - Generated Stored Procedures may include; select all, select by primary key, insert, update, delete, and more operations
    - Generated only when the Stored Procedure option is selected
    - At least 5 Stored Procedures are generated per table (for most tables)
    - Located directly in the database

2. **Dynamic SQL Class**

    - Contains T-SQL CRUD operations in the code
    - Do not rewrite or edit generated Dynamic SQL, instead, new dynamic SQL should be added in the *DataLayer* class
    - Generated Dynamic SQL may include; select all, select by primary key, insert, update, delete, and more operations
    - Generated only when the Dynamic SQL option is selected
    - One Class is generated per table
    - Located in the *SQL* folder

**Stored Procedures Or Methods Generated by AspxFormsGen 4.0**

a. **SelectAll:** Selects all records from a specific table or view. **Note:** The only method generated when *All Views* or *Selected Views Only* is selected under *Code Settings.*
b. **SelectByPrimaryKey:** Selects a record by primary key.
c. **SelectDropDownListData:** Selects 2 fields from the specific table for use with a DropDownList control source (or combo box, etc).
d. **SelectCollectionBy Foreign Key:** Selects all records by foreign key.
e. **Insert:** Inserts a record in the table.
f. **Update:** Updates an existing record in the table by primary key.
g. **Delete:** Deletes a record from a table by primary key.

## Example Classes

- **Generated solely to show how to use the Generated Code**
- Example code can be copied and pasted directly to your client code (ASP.Net web forms, Win Forms, Web Services, etc.)
- You can delete the whole directory if you don't need it
- One Class is generated per table
- Located in the *Example* folder

## Helper Classes [1]

Two helper classes are generated: Dbase.cs (or Dbase.vb) and Functions.cs (or Functions.vb):

1. **Dbase class:** Contains static/shared methods/functions that connect to the database. Also contains the connection string to the database.

2. **Functions class:** Contains static/shared functions/methods used in GridViews.

## Miscellaneous Files

AspxFormsGen 4.0 also creates files that a standard ASP.NET 4.0 web site needs. And because we're using a few JQuery plug-ins, the script for these are also copied onto the generated website.

1. **SkinFile.skin:** Skin file in the *Theme1* theme, under the *App_Themes* folder.

2. **Images Folder:** Approximately 11 [1] images are copied onto an images folder.

3. **JQuery Themes:** [1] Several JQuery UI themes are copied onto this folder.

4. **Scripts Folder:** [1] Scripts used by JQuery, JQuery UI plugin, JQuery validation plugin and web forms are copied onto this folder.

5. **Styles Folder:** Styles used by web forms and some JQuery plugins[1] are located here.

6. **MasterPage.master:** An empty master page is generated and used by all the web forms.

7. **Web.config**: ASP.NET Configuration file. A setting to use Theme1 for all pages can be found here. References to the gridview's sort arrow images are also here.

8. **Default.htm:** List of all the generated web forms.

9. **GeneratedCode.htm:** List of all the generated middle-tier, data-tier, stored procedures[1] or dynamic SQl[1], example classes, and helper classes.

# Adding Your Own Code

Yes you can add your own code to the objects generated by AspxFormsGen 4.0. **But we warned: Almost all generated files are overwritten by AspxFormsGen 4.0 without warning**, of course with some exceptions. Please follow this tutorial carefully. **Codes shown below are just examples.** Added or modified code is highlighted.

## ASP.NET Files

AspxFormsGen 4.0 will not overwrite these files if they are **unchecked** [1] in the *App Settings* tab, under the *Overwrite Files* [1] group.  Please see the App Settings discussion above for more information on these files.

1. **Master Page:**
   Add a logo image to the Master Pager file.

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs"
     Inherits="Northwind.MasterPage" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <link rel="stylesheet" type="text/css" href="Styles/global.css" />
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="MasterPageForm1" class="cmxform" runat="server">
    <div>
        <img src="Images/Logo.png" alt="My Logo" width="100px" height="50px" />
        <br /><br />
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

        </asp:ContentPlaceHolder>
    </div>
    </form>
</body>
</html>
```

2. **Dbase File** [1]
   Modified the connection string.  Moved the connection string to the Web.config file.

   **In C#:**

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

namespace Northwind.DataLayer
{
     public sealed class Dbase
     {
         private Dbase()
         {
         }

         public static SqlConnection GetConnection()
         {
             string connectionString = ConfigurationManager.AppSettings["MyConnectionString"];
             SqlConnection connection = new SqlConnection(connectionString);
             connection.Open();
             return connection;
         }
```

   **In VB.NET**

```
Imports System
Imports System.Data
Imports System.Data.SqlClient
```

```
Namespace Northwind.DataLayer
     Public NotInheritable Class Dbase
         Private Sub New()
         End Sub

         Public Shared Function GetConnection() As SqlConnection
            Dim connectionString As String = ConfigurationManager.AppSettings("MyConnectionString")
            Dim connection As New SqlConnection(connectionString)
            connection.Open()
            Return connection
         End Function
```

## 3. Web.config File

Added an app setting.

```
<?xml version="1.0"?>
<configuration>
    <appSettings>
        <add key="ArrowUp" value="Images/ArrowUp.png" />
        <add key="ArrowDown" value="Images/ArrowDown.png" />
        <add key="MyConnectionString"
             value="Data Source=localhost;Initial Catalog=Northwind;User ID=junnark;Password=junnark" />
    </appSettings>
```

## 4. Functions File [1]

Add a function or method.

### In C#:

```
using System;
using System.Web.UI.WebControls;
using System.Configuration;
using System.Text.RegularExpressions;
using System.Web.UI;
using System.IO;

namespace Northwind
{
    public sealed class Functions
    {
        private Functions()
        {
        }

        public static int Add(int num1, int num2)
        {
            return num1 + num2;
        }
```

### In VB.NET

```
Imports System
Imports System.Web.UI.WebControls
Imports System.Configuration
Imports System.Text.RegularExpressions
Imports System.Web.UI
Imports System.IO

Namespace Northwind
    Public NotInheritable Class Functions
        Private Sub New()
        End Sub

        Public Shared Function Add(num1 As Integer, num2 As Integer) As Integer
            Return num1 + num2
        End Function
```

5. **Global.css**
   Add a style.

```
/*************************** jquery ui ******************************/
@import "../JQueryThemes/redmond/jquery-ui-1.8.11.custom.css";

/*************************** jquery validate ************************/
@import "core.css";
@import "cmxformTemplate.css";
@import "cmxform.css";

form.MasterPageForm1 { width: 50em; }

.myClass {
    font-weight: bold;
    font-size: larger;
}
```

6. **SkinFile.skin**
   Add a skin for a button.

```
<asp:Textbox runat="Server" Font-Size="12px" Width="250px" />
<asp:Textbox SkinID="TextBoxDate" runat="Server" Font-Size="12px" Width="234px" />
<asp:DropDownList runat="Server" Font-Size="12px" Width="256px" />
<asp:Label runat="Server" Font-Size="12px" />
<asp:Button runat="server" Width="150px" />
<asp:Button SkinID="MyButton" runat="server" Width="100px" />
```

## ASP.NET Web Forms

Yes you can add web forms to the generated web site **just make sure that it does not have the same name as the ones that are going to be generated by AspxFormsGen 4.0**.

For this tutorial, we will add a new ASP.NET 4.0 web form to the generated web site. This web form will be bound to the *Northwind* database just like the rest of the generated web forms. In this example, we will add middle-tier classes, data-tier classes, stored procedures and dynamic SQL. Each of the respective object will be discussed in their respective parts.

1. Add a new web form to the generated web site by right-clicking on the web site and then choose *Add New Item*. See Figure 47. Then on the Add New Item dialog copy the following as seen in Figure 48. Make sure to check *Select master page*. You can also choose either Visual C# or Visual Basic. Then click *Add*.

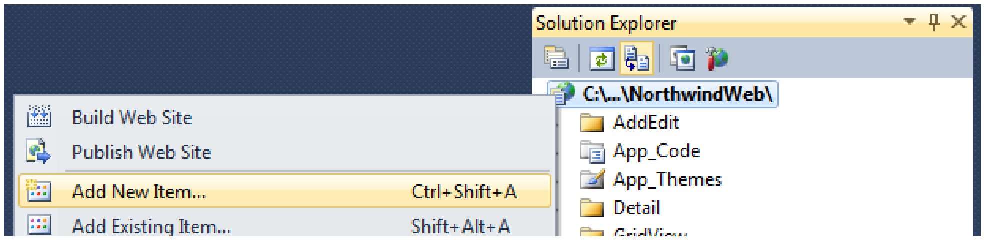


**Figure 47 Add New Item**

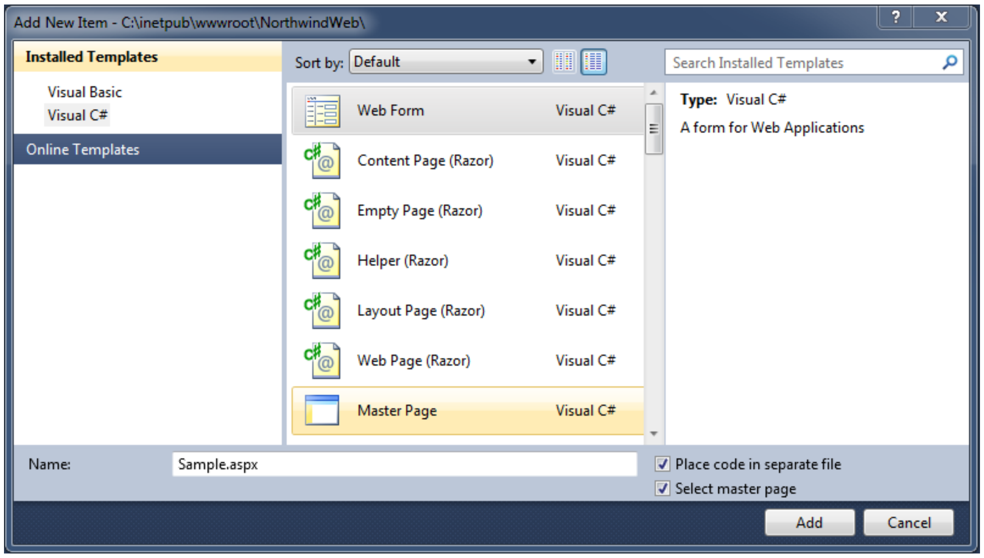

Add New Item - C:\inetpub\wwwroot\NorthwindWeb\
Installed Templates
Visual Basic
Visual C#
Online Templates
Sort by: Default
Search Installed Templates
Web Form Visual C#
Content Page (Razor) Visual C#
Empty Page (Razor) Visual C#
Helper (Razor) Visual C#
Layout Page (Razor) Visual C#
Web Page (Razor) Visual C#
Master Page Visual C#
Type: Visual C#
A form for Web Applications
Name: Sample.aspx
Place code in separate file
Select master page
Add
Cancel

**Figure 48 Add New Item Dialog**

2. A new web form called “Sample.aspx” is now added.

3. Let’s stop for a moment and create a new stored procedure. Please see the Stored Procedures tutorial below.

4. Now that we’re done creating our stored procedure, we will now call this stored procedure through our data layer. Please see the Data Tier Class tutorial below.

5. Moving on, we will now encapsulate the data layer method that we created in the data tier. Please see the Middle Tier Class tutorial below.

6. Now that the stored procedure, data tier, and middle tier codes are done, we can now go back to the *Sample.aspx* web form.

7. Copy and paste the following code in the *Sample.aspx* file. Note: If you’re using VB.NET change the language to “”.VB

```
<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master"
AutoEventWireup="true" CodeFile="Sample.aspx.cs" Inherits="Northwind.Sample" %>

<asp:Content ID="Content3" ContentPlaceHolderID="head" Runat="Server">
    <script type="text/javascript">
        function SetGridViewHoverOn(gridView) {
            if (gridView != null) {
                gridView.originalBgColor = gridView.style.backgroundColor;
                gridView.style.backgroundColor = '#FFFFCC';
            }
        }

        function SetGridViewHoverOff(gridView) {
            if (gridView != null) {
                gridView.style.backgroundColor = gridView.originalBgColor;
            }
        }
    </script>
</asp:Content>

<asp:Content ID="Content4" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <asp:ScriptManager ID="ScriptManager1" runat="server" />
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
          <asp:GridView ID="GridView1" runat="server" DataKeyNames="CustomerID" DataSourceID="ObjectDataSource1"
              Onrowdatabound="GridView1_RowDataBound" onrowcreated="GridView1_RowCreated"
             SkinID="GridViewProfessional">
            <Columns>
                <asp:BoundField DataField="CustomerID" HeaderText="Customer ID"
                      ReadOnly="true" SortExpression="CustomerID" />
                <asp:BoundField DataField="CompanyName" HeaderText="Company Name"
                      ReadOnly="true" SortExpression="CompanyName" />
                 <asp:BoundField DataField="ContactName" HeaderText="Contact Name"
                      ReadOnly="true" SortExpression="ContactName" />
                 <asp:BoundField DataField="Phone" HeaderText="Phone"
                      ReadOnly="true" SortExpression="Phone" />
                 <asp:BoundField DataField="CityAndCountry" HeaderText="City And Country"
                      ReadOnly="true" SortExpression="CityAndCountry" />
             </Columns>
          </asp:GridView>
        </ContentTemplate>
    </asp:UpdatePanel>

    <asp:UpdateProgress ID="UpdateProgress1" AssociatedUpdatePanelID="UpdatePanel1"
        runat="server" DisplayAfter="0">
        <ProgressTemplate>
          <br />
          <img src="../Images/ActivityIndicator.gif" alt="" /> Processing your request.  Please wait....
        </ProgressTemplate>
    </asp:UpdateProgress>

    <asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
        TypeName="Northwind.BusinessObject.Customers"
        SelectMethod="SelectUSAOnly" SortParameterName="sortExpression">
    </asp:ObjectDataSource>
</asp:Content>
```

8. Things to note here are highlighted. This code looks just like the one in the *GridViewReadOnly* folder for the Customers web form, with minor changes.

9. Now copy and paste the code behind for your respective language.

**In C#, Sample.aspx.cs**

```
using System;

namespace Northwind
{
    public partial class Sample : System.Web.UI.Page
    {
        protected void GridView1_RowDataBound(object sender, System.Web.UI.WebControls.GridViewRowEventArgs e)
        {
            Functions.GridViewRowDataBound(sender, e, 1);
        }

        protected void GridView1_RowCreated(object sender, System.Web.UI.WebControls.GridViewRowEventArgs e)
        {
            Functions.GridViewRowCreated(sender, e, 0);
        }
    }
}
```

**In VB.NET, Sample.aspx.vb**

```
Imports System

Namespace Northwind
```

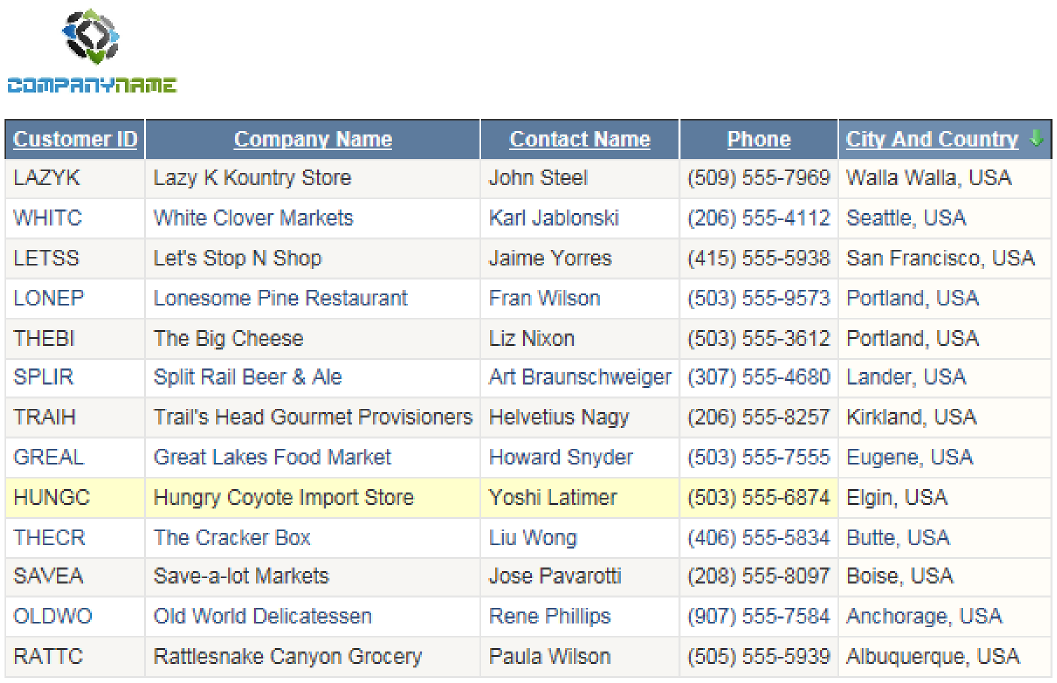
```
        Partial Class Sample
            Inherits System.Web.UI.Page

            Protected Sub GridView1_RowDataBound(sender As Object,
            e As System.Web.UI.WebControls.GridViewRowEventArgs)

                 Functions.GridViewRowDataBound(sender, e, 1)
            End Sub

            Protected Sub GridView1_RowCreated(sender As Object, e As System.Web.UI.WebControls.GridViewRowEventArgs)
                Functions.GridViewRowCreated(sender, e, 0)
            End Sub
        End Class
    End Namespace
```

10. We can now test the new web page by running the web site.  But first make sure to set *Sample.aspx* as Start Page.  Click *F5* in *Visual Studio 2010*.  See Figure 49.



| Customer ID | Company Name | Contact Name | Phone | City And Country |
|---|---|---|---|---|
| LAZYK | Lazy K Kountry Store | John Steel | (509) 555-7969 | Walla Walla, USA |
| WHITC | White Clover Markets | Karl Jablonski | (206) 555-4112 | Seattle, USA |
| LETSS | Let's Stop N Shop | Jaime Yorres | (415) 555-5938 | San Francisco, USA |
| LONEP | Lonesome Pine Restaurant | Fran Wilson | (503) 555-9573 | Portland, USA |
| THEBI | The Big Cheese | Liz Nixon | (503) 555-3612 | Portland, USA |
| SPLIR | Split Rail Beer & Ale | Art Braunschweiger | (307) 555-4680 | Lander, USA |
| TRAIH | Trail's Head Gourmet Provisioners | Helvetius Nagy | (206) 555-8257 | Kirkland, USA |
| GREAL | Great Lakes Food Market | Howard Snyder | (503) 555-7555 | Eugene, USA |
| HUNGC | Hungry Coyote Import Store | Yoshi Latimer | (503) 555-6874 | Elgin, USA |
| THECR | The Cracker Box | Liu Wong | (406) 555-5834 | Butte, USA |
| SAVEA | Save-a-lot Markets | Jose Pavarotti | (208) 555-8097 | Boise, USA |
| OLDWO | Old World Delicatessen | Rene Phillips | (907) 555-7584 | Anchorage, USA |
| RATTC | Rattlesnake Canyon Grocery | Paula Wilson | (505) 555-5939 | Albuquerque, USA |

**Figure 49 Sample.aspx**

11. Notice the new *City And Country* field.  Play around to see how the sample web page behaves.

12. Close the web page.  We showed you how to use a stored procedure, what if you'd like to code your SQL into your classes (dynamic SQL)?  We'll show you this as well.  Please see the Dynamic SQL tutorial.

13. You can run the Sample.aspx one more time and notice that it behaves exactly as if you're using a stored procedure.

14. End of tutorial.

## Middle Tier Class

The middle object encapsulates the data layer code, making sure that the calling client code does not need to bother with the specifics of how the information was obtained.

**Note:** Always add code in the **BusinessObject** class. **Do not add code in the BusinessObjectBase class and the BusinessObjectCollection class, these classes will be overwritten every time you generate code using AspxFormsGen 4.0.**

1. Add a method in the *Customers* class that encapsulates the *SelectUSAOnly* method/function we added in the *CustomersDataLayer* class (Data Tier Class tutorial). The *Customers* class is located under the *BusinessObject* folder. See method below.

   **In C#**

```
using System;
using Northwind.BusinessObject.Base;
using Northwind.DataLayer;

namespace Northwind.BusinessObject
{
     /// <summary>
     /// This file will not be overwritten.  You can put
     /// additional Customers Business Layer code in this class.
     /// </summary>
     public class Customers : CustomersBase
     {
         public string CityAndCountry { get; set; }

         // constructor
         public Customers()
         {
         }

         public static CustomersCollection SelectUSAOnly(string sortExpression)
         {
             CustomersCollection objCustomersCol = CustomersDataLayer.SelectUSAOnly();
             return SortByExpression2(objCustomersCol, sortExpression);
         }

         private static CustomersCollection SortByExpression2(CustomersCollection objCustomersCol,
         string sortExpression)
         {
             bool isSortDescending = sortExpression.Contains(" DESC");

             if (isSortDescending)
                 sortExpression = sortExpression.Replace(" DESC", "");

             switch (sortExpression)
             {
                 case "CustomerID":
                     objCustomersCol.Sort(Northwind.BusinessObject.Customers.ByCustomerID);
                     break;
                 case "CompanyName":
                     objCustomersCol.Sort(Northwind.BusinessObject.Customers.ByCompanyName);
                     break;
                 case "ContactName":
                     objCustomersCol.Sort(Northwind.BusinessObject.Customers.ByContactName);
                     break;
                 case "Phone":
                     objCustomersCol.Sort(Northwind.BusinessObject.Customers.ByPhone);
                     break;
                 case "CityAndCountry":
                     objCustomersCol.Sort(Northwind.BusinessObject.Customers.ByCityAndCountry);
                     break;
                 default:
                     break;
             }

             if (isSortDescending)
                 objCustomersCol.Reverse();

             return objCustomersCol;
         }

         public static Comparison<Customers> ByCityAndCountry = delegate(Customers x, Customers y)
         {
             string value1 = x.CityAndCountry ?? String.Empty;
             string value2 = y.CityAndCountry ?? String.Empty;
             return value1.CompareTo(value2);
         };
     }
}
```

### In VB.NET

```
Imports System
Imports Northwind.BusinessObject.Base
Imports Northwind.DataLayer

Namespace Northwind.BusinessObject

     ''' <summary>
     ''' This file will not be overwritten.  You can put
     ''' additional Customers Business Layer code in this class.
     ''' </summary>
     Public Class Customers
        Inherits CustomersBase

        Private _cityAndCountry As String

        Public Property CityAndCountry() As String
            Get
                Return _cityAndCountry
            End Get
            Set(ByVal value As String)
                _cityAndCountry = value
            End Set
        End Property

        ' constructor
        Public Sub New()
        End Sub

        Public Shared Function SelectUSAOnly(sortExpression As String) As CustomersCollection
            Dim objCustomersCol As CustomersCollection = CustomersDataLayer.SelectUSAOnly()
            Return SortByExpression2(objCustomersCol, sortExpression)
        End Function

        Public Shared Function SortByExpression2(objCustomersCol As CustomersCollection,
        sortExpression As String) As CustomersCollection
            Dim isSortDescending As Boolean = sortExpression.Contains(" DESC")

            If isSortDescending Then
                sortExpression = sortExpression.Replace(" DESC", "")
            End If

            Select Case sortExpression
                Case "CustomerID"
                    objCustomersCol.Sort(Northwind.BusinessObject.Customers.ByCustomerID)
                    Exit Select
                Case "CompanyName"
                    objCustomersCol.Sort(Northwind.BusinessObject.Customers.ByCompanyName)
                    Exit Select
                Case "ContactName"
                    objCustomersCol.Sort(Northwind.BusinessObject.Customers.ByContactName)
                    Exit Select
                Case "Phone"
```

```
                    objCustomersCol.Sort(Northwind.BusinessObject.Customers.ByPhone)
                    Exit Select
                Case "CityAndCountry"
                    objCustomersCol.Sort(Northwind.BusinessObject.Customers.ByCityAndCountry)
                    Exit Select
                Case Else
                    Exit Select
            End Select

            If isSortDescending Then
                objCustomersCol.Reverse()
            End If

            Return objCustomersCol
        End Function

        Public Shared ByCityAndCountry As Comparison(Of Customers) =
         Function(x As Customers, y As Customers)
             Dim value1 As String = If(x.CityAndCountry, String.Empty)
             Dim value2 As String = If(y.CityAndCountry, String.Empty)
             Return value1.CompareTo(value2)
         End Function
     End Class
End Namespace
```

2. Notice that the *SelectUSAOnly* method/function looks just like the SelectAll method/function in the *CustomersBase* class.

3. Notice also that we created a new method/function *SortByExpression2* which is very similar to the *SortByExpression* method/function in the *CustomersBase* class, but with a fewer *sortExpression* choices. We don't really need to add the *SortByExpression2* method/function if we didn't add a new field, *CityAndCountry*. That is the same reason we added the *Comparison* for *ByCityAndCountry*, so that we can sort by the new field *CityAndCountry*.

4. End of middle tier tutorial.

## Data Tier Class

The data layer encapsulates SQL code, making sure that the calling middle layer code does not need to bother with the specifics of connecting to the database, or any data specific connections.

**Note:** Always add code in the **DataLayer** class. **Do not add code in the DataLayerBase class, this class will be overwritten every time you generate code using AspxFormsGen 4.0.**

1. Add a method in the *CustomersDataLayer* class that encapsulates the stored procedure we created in the Stored Procedure tutorial. The *CustomersDataLayer* class is located under the *DataLayer* folder. See method below.

   **In C#**

```
using System;
using Northwind.DataLayer.Base;
using Northwind.BusinessObject;
using System.Data.SqlClient;
using System.Data;
```

```
namespace Northwind.DataLayer
{
     /// <summary>
     /// This file will not be overwritten.  You can put
     /// additional Customers DataLayer code in this class
     /// </summary>
     public class CustomersDataLayer : CustomersDataLayerBase
     {
         // constructor
         public CustomersDataLayer()
         {
         }

         public static CustomersCollection SelectUSAOnly()
         {
             string storedProcName = "[dbo].[Customers_SelectUSA]";
             SqlConnection connection = Dbase.GetConnection();
             SqlCommand command = Dbase.GetCommand(storedProcName, connection);

             DataSet ds = Dbase.GetDbaseDataSet(command);
             CustomersCollection objCustomersCol = new CustomersCollection();
             Customers objCustomers;

             if (ds.Tables[0].Rows.Count > 0)
             {
                 foreach (DataRow dr in ds.Tables[0].Rows)
                 {
                     objCustomers = new Customers();
                     objCustomers.CustomerID = dr["CustomerID"].ToString();
                     objCustomers.CompanyName = dr["CompanyName"].ToString();

                     if (dr["ContactName"] != System.DBNull.Value)
                         objCustomers.ContactName = dr["ContactName"].ToString();
                     else
                         objCustomers.ContactName = null;

                     if (dr["City"] != System.DBNull.Value)
                         objCustomers.City = dr["City"].ToString();
                     else
                         objCustomers.City = null;

                     if (dr["Country"] != System.DBNull.Value)
                         objCustomers.Country = dr["Country"].ToString();
                     else
                         objCustomers.Country = null;

                     if (dr["Phone"] != System.DBNull.Value)
                         objCustomers.Phone = dr["Phone"].ToString();
                     else
                         objCustomers.Phone = null;

                     objCustomers.CityAndCountry = objCustomers.City + ", " + objCustomers.Country;

                     objCustomersCol.Add(objCustomers);
                 }
             }

             command.Dispose();
             connection.Close();
             connection.Dispose();
             ds.Dispose();

             return objCustomersCol;
         }
     }
}
```

### In VB.NET

```
Imports System
Imports Northwind.DataLayer.Base
Imports Northwind.BusinessObject
Imports System.Data.SqlClient
Imports System.Data

Namespace Northwind.DataLayer

    ''' <summary>
    ''' This file will not be overwritten.  You can put
    ''' additional Customers DataLayer code in this class
    ''' </summary>
     Public Class CustomersDataLayer
         Inherits CustomersDataLayerBase

        ' constructor
        Public Sub New()
        End Sub

        Public Shared Function SelectUSAOnly() As CustomersCollection
            Dim storedProcName = "[dbo].[Customers_SelectUSA]"
            Dim connection As SqlConnection = Dbase.GetConnection()
            Dim command As SqlCommand = Dbase.GetCommand(storedProcName, connection)

            Dim ds As DataSet = Dbase.GetDbaseDataSet(command)
            Dim objCustomersCol As New CustomersCollection()
            Dim objCustomers As Customers

            If ds.Tables(0).Rows.Count > 0 Then
                For Each dr As DataRow In ds.Tables(0).Rows
                    objCustomers = New Customers()
                    objCustomers.CustomerID = dr("CustomerID").ToString()
                    objCustomers.CompanyName = dr("CompanyName").ToString()

                    If Not dr("ContactName").Equals(System.DBNull.Value) Then
                        objCustomers.ContactName = dr("ContactName").ToString()
                    Else
                        objCustomers.ContactName = Nothing
                    End If

                    If Not dr("City").Equals(System.DBNull.Value) Then
                        objCustomers.City = dr("City").ToString()
                    Else
                        objCustomers.City = Nothing
                    End If

                    If Not dr("Country").Equals(System.DBNull.Value) Then
                        objCustomers.Country = dr("Country").ToString()
                    Else
                        objCustomers.Country = Nothing
                    End If

                    If Not dr("Phone").Equals(System.DBNull.Value) Then
                        objCustomers.Phone = dr("Phone").ToString()
                    Else
                        objCustomers.Phone = Nothing
                    End If

                    objCustomers.CityAndCountry = objCustomers.City & ", " & objCustomers.Country

                    objCustomersCol.Add(objCustomers)
                Next
            End If

            command.Dispose()
            connection.Close()
            connection.Dispose()
            ds.Dispose()

            Return objCustomersCol
        End Function
     End Class
End Namespace
```

2. Notice that the *SelectUSAOnly* method/function above looks just like the *SelectShared* method/function in the *CustomersDataLayerBase* class, with a few modifications.

3. Also notice the stored procedure name we're using is the same stored procedure we created in the Stored Procedures tutorial below.

4. End of data tier tutorial.

## Stored Procedures

To add a new stored procedure in your database, make sure that you give it a **unique** name so that AspxFormsGen 4.0 will not overwrite it. In this tutorial, we will add a stored procedure that retrieves customers in the USA. It will also retrieve 6 fields instead of all the fields.

1. Add a new stored procedure from *MS SQL Management Studio*. Open the *Northwind* database node. Then right-click on the *Stored Procedure* node under the *Programmability* node and click *New Stored Procedure*. See Figure 50.

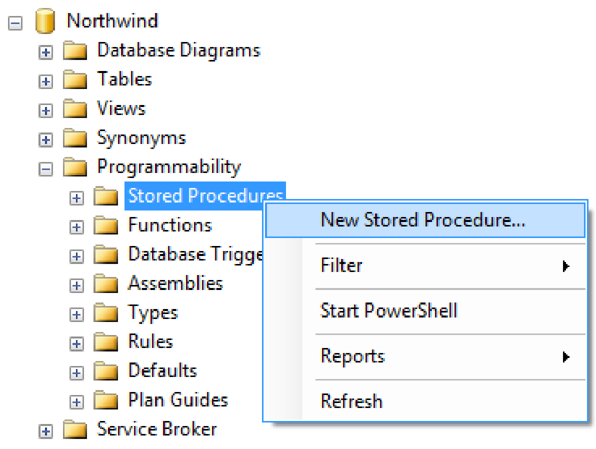


**Figure 50 Add New Stored Procedure**

2. Copy the script below and click the *Execute* button on *MS SQL Server Management Studio*.

```
 CREATE PROCEDURE [dbo].[Customers_SelectUSA]
 AS
 BEGIN
        SET NOCOUNT ON;

        SELECT [CustomerID], [CompanyName], [ContactName],[Phone], [City], [Country]
        FROM [Northwind].[dbo].[Customers]
        WHERE Country = 'USA'
        ORDER BY Country
END
GO
```

3. Notice that the newly added stored procedure is now under the *Stored Procedure* node. See Figure 51.

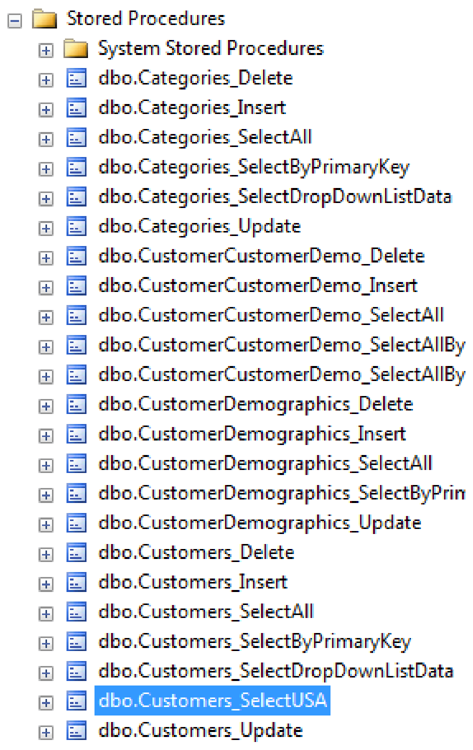

Stored Procedures
System Stored Procedures
dbo.Categories_Delete
dbo.Categories_Insert
dbo.Categories_SelectAll
dbo.Categories_SelectByPrimaryKey
dbo.Categories_SelectDropDownListData
dbo.Categories_Update
dbo.CustomerCustomerDemo_Delete
dbo.CustomerCustomerDemo_Insert
dbo.CustomerCustomerDemo_SelectAll
dbo.CustomerCustomerDemo_SelectAllBy
dbo.CustomerCustomerDemo_SelectAllBy
dbo.CustomerDemographics_Delete
dbo.CustomerDemographics_Insert
dbo.CustomerDemographics_SelectAll
dbo.CustomerDemographics_SelectByPrin
dbo.CustomerDemographics_Update
dbo.Customers_Delete
dbo.Customers_Insert
dbo.Customers_SelectAll
dbo.Customers_SelectByPrimaryKey
dbo.Customers_SelectDropDownListData
dbo.Customers_SelectUSA
dbo.Customers_Update

**Figure 51 Newly Added Stored Procedure**

4. End of stored procedure tutorial.

## Dynamic SQL

You can use dynamic SQL (SQL in your code) instead of stored procedures. **Recommendation:** Add your SQL code in the *DataLayer* class. This tutorial will be using the same code as the Data Tier tutorial with a minor modification.

Note: We're just showing the partial CustomerDataLayer class in these examples. To see the full class, go to the Data Tier tutorial above.

1. Open the *CustomerDataLayer* class as shown in the Data Tier tutorial. Modify the code as seen below. Note: we are modifying the first 3 lines of the *SelecteUSAOnly* method/function.

**In C#**

```
using System;
using Northwind.DataLayer.Base;
using Northwind.BusinessObject;
using System.Data.SqlClient;
using System.Data;

namespace Northwind.DataLayer
{
     /// <summary>
     /// This file will not be overwritten.  You can put
     /// additional Customers DataLayer code in this class
     /// </summary>
     public class CustomersDataLayer : CustomersDataLayerBase
     {
         // constructor
         public CustomersDataLayer()
         {
         }

         public static CustomersCollection SelectUSAOnly()
         {
             //string storedProcName = "[dbo].[Customers_SelectUSA]";
             //SqlConnection connection = Dbase.GetConnection();
             //SqlCommand command = Dbase.GetCommand(storedProcName, connection);

             string sql = "SELECT [CustomerID], [CompanyName], [ContactName],[Phone], [City], [Country]" +
                "FROM [Northwind].[dbo].[Customers]" +
                "WHERE Country = 'USA'" +
                "ORDER BY Country";

             SqlConnection connection = Dbase.GetConnection();
             SqlCommand command = Dbase.GetCommand(connection, sql);

             DataSet ds = Dbase.GetDbaseDataSet(command);
             CustomersCollection objCustomersCol = new CustomersCollection();
             Customers objCustomers;
```

**In VB.NET**

```
Imports System
Imports Northwind.DataLayer.Base
Imports Northwind.BusinessObject
Imports System.Data.SqlClient
Imports System.Data

Namespace Northwind.DataLayer

    ''' <summary>
    ''' This file will not be overwritten.  You can put
    ''' additional Customers DataLayer code in this class
    ''' </summary>
     Public Class CustomersDataLayer
         Inherits CustomersDataLayerBase

        ' constructor
        Public Sub New()
        End Sub

        Public Shared Function SelectUSAOnly() As CustomersCollection
            'Dim storedProcName = "[dbo].[Customers_SelectUSA]"
            'Dim connection As SqlConnection = Dbase.GetConnection()
            'Dim command As SqlCommand = Dbase.GetCommand(storedProcName, connection)

            Dim sql = "SELECT [CustomerID], [CompanyName], [ContactName],[Phone], [City], [Country]" & _
                "FROM [Northwind].[dbo].[Customers]" & _
                "WHERE Country = 'USA'" & _
                "ORDER BY Country"

            Dim connection As SqlConnection = Dbase.GetConnection()
            Dim command As SqlCommand = Dbase.GetCommand(connection, sql)

            Dim ds As DataSet = Dbase.GetDbaseDataSet(command)
            Dim objCustomersCol As New CustomersCollection()
            Dim objCustomers As Customers
```

2. End of dynamic SQL tutorial.

# Using the Generated Middle Tier in Your Code

The AspxCodeGen engine generated code can be used beyond the generated web forms for AspxFormsGen 4.0. As a matter of fact, you can use it in just about any .NET client. Your client could be web forms like the examples in this document, or it could also be win forms, a web service, Silverlight app, etc. To use/share the generated middle tier and data tier objects in other projects (see note below), put the code in a *Class Library* project and reference the *Class Library* in the client project.

**Note: You can use it in more than one project, any project that will use the same database or objects**)

## Tutorial on How to Create a Class Library

1. Create a *Class Library Project* in *Visual Studio 2010*. Name is *NortwindAPI*. See Figures 52 and 53.

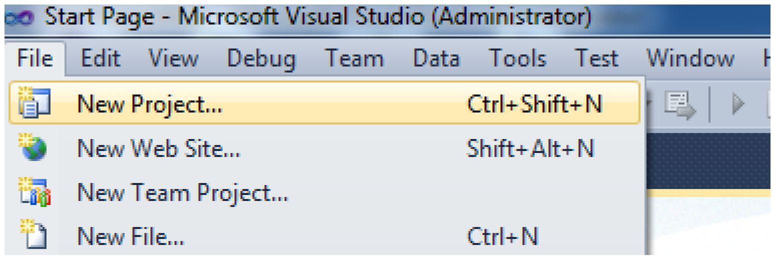
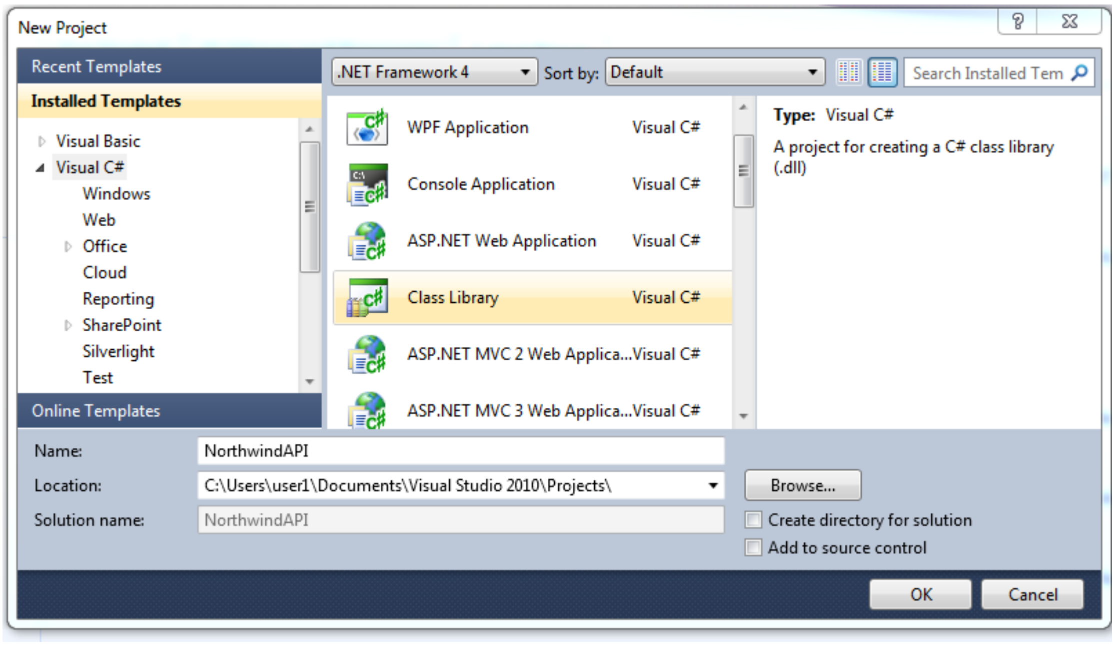

Start Page - Microsoft Visual Studio (Administrator)
File Edit View Debug Team Data Tools Test Window
New Project... Ctrl+Shift+N
New Web Site... Shift+Alt+N
New Team Project...
New File... Ctrl+N

**Figure 52 Create a New Project**



Name: NorthwindAPI
Location: C:\Users\user1\Documents\Visual Studio 2010\Projects\
Solution name: NorthwindAPI

**Figure 53 Class Library**

2. In Figure 53 click the *OK* button. A new Class Library project is now created. Delete *Class1*. See Figure 54.

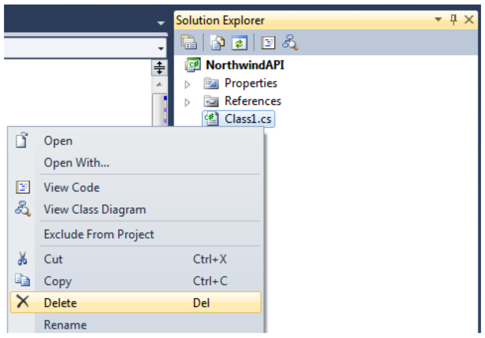


**Figure 54 Delete Class1**

3. Remove all the code from the *NorthwindWeb* web site, under the *App_Code* folder except the *Functions* class under the *Helper* folder and move it to the *NorthwindAPI* class library project. See Figures 55 and 56.

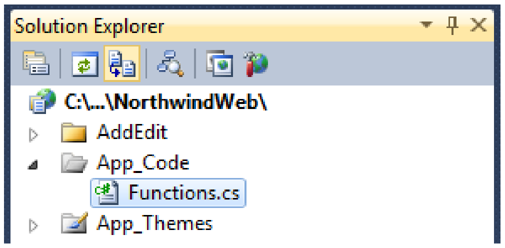


**Figure 55 NortwindWeb Web Site – App_Code Directory**

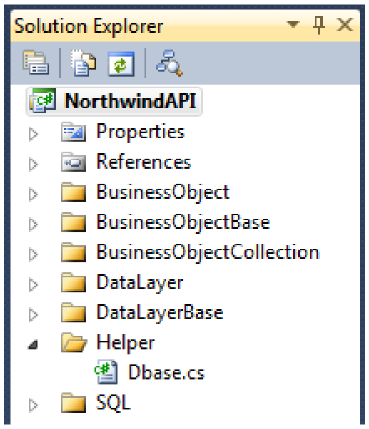


**Figure 56 NorthwindAPI Class Library Project**

4. From the *NorthwindWeb* web site, in the *File* menu, add an existing project as seen in Figure 57.

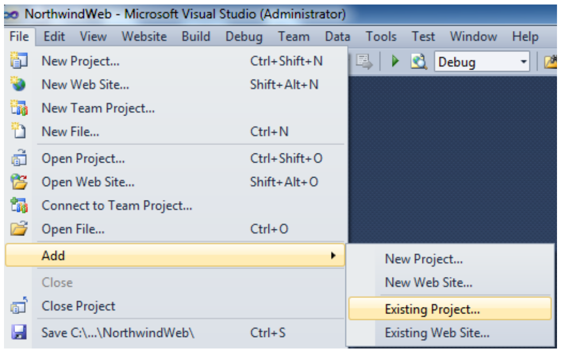
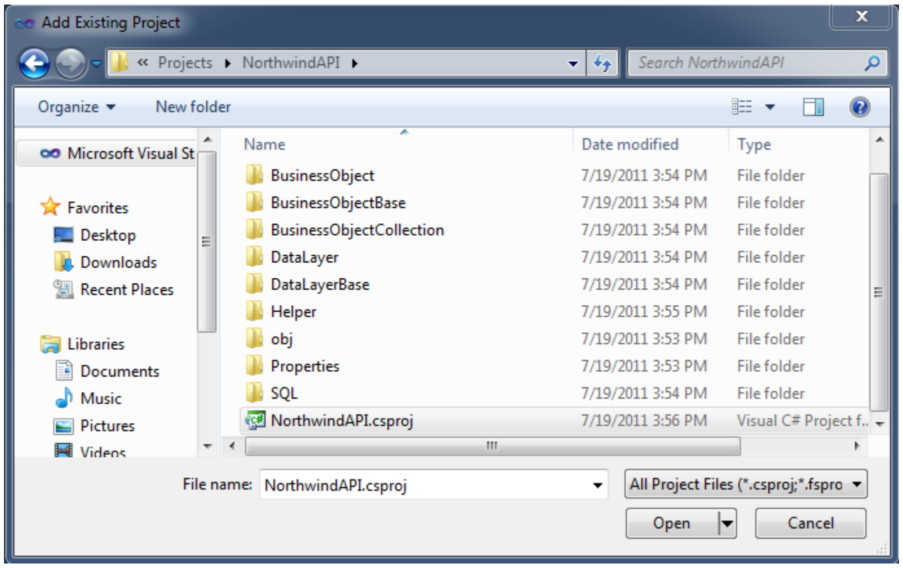

NorthwindWeb - Microsoft Visual Studio (Administrator)
File Edit View Website Build Debug Team Data Tools Test Window Help
New Project... Ctrl+Shift+N
New Web Site... Shift+Alt+N
New Team Project...
New File... Ctrl+N
Open Project... Ctrl+Shift+O
Open Web Site... Shift+Alt+O
Connect to Team Project...
Open File... Ctrl+O
Add
Close
Close Project
Save C:\...\NorthwindWeb\ Ctrl+S
Debug
New Project...
New Web Site...
Existing Project...
Existing Web Site...

**Figure 57 Add Existing Project**

5. Drill down to the *NorthwindAPI* project file and then click the *Open* button. See Figure 58.

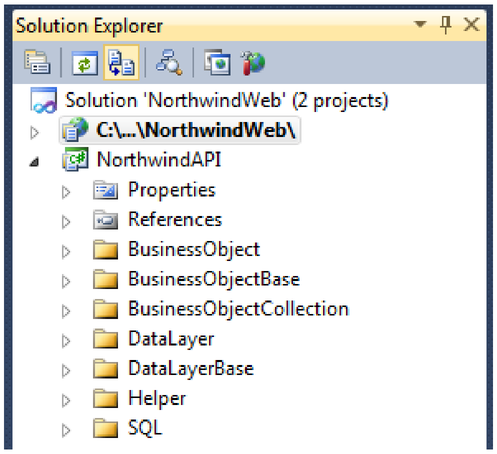


| Name | Date modified | Type |
|---|---|---|
| BusinessObject | 7/19/2011 3:54 PM | File folder |
| BusinessObjectBase | 7/19/2011 3:54 PM | File folder |
| BusinessObjectCollection | 7/19/2011 3:54 PM | File folder |
| DataLayer | 7/19/2011 3:54 PM | File folder |
| DataLayerBase | 7/19/2011 3:54 PM | File folder |
| Helper | 7/19/2011 3:55 PM | File folder |
| obj | 7/19/2011 3:53 PM | File folder |
| Properties | 7/19/2011 3:53 PM | File folder |
| SQL | 7/19/2011 3:54 PM | File folder |
| NorthwindAPI.csproj | 7/19/2011 3:56 PM | Visual C# Project f... |

File name: NorthwindAPI.csproj
All Project Files (*.csproj;*.fspro

**Figure 58 Add Existing Project Dialog**

6. You will now notice that there are 2 projects in the Solution Explorer; *NorthwindWeb* and *NorthwindAPI*. See Figure 59.



```
Solution 'NorthwindWeb' (2 projects)
  C:\...\NorthwindWeb\
  NorthwindAPI
    Properties
    References
    BusinessObject
    BusinessObjectBase
    BusinessObjectCollection
    DataLayer
    DataLayerBase
    Helper
    SQL
```

**Figure 58 2 Projects – NorthwindWeb and NorthwindAPI**

7. If you used AspxFormsGen 4.0 to generate the code, there are a few things that need to be corrected in the *NorthwindAPI*. First delete the *Example* folder, as you can see in Figure 58 there is no *Example* folder. And then, remove the following in all codes; you can do this by using Visual Studio's Find and Replace dialog. Replace the following with an empty string.

```
a. using System.Web.Script.Serialization; (C#)
b. [ScriptIgnore]  (C#)
c. Imports System.Web.Script.Serialization (VB.NET)
d. <ScriptIgnore()> _ (VB.NET)
```

8. Rebuild the *NorthwindAPI* project and then from the *NorthwindWeb* web site, add a reference to the *NorthwindAPI* project. See Figures 59 and 60.

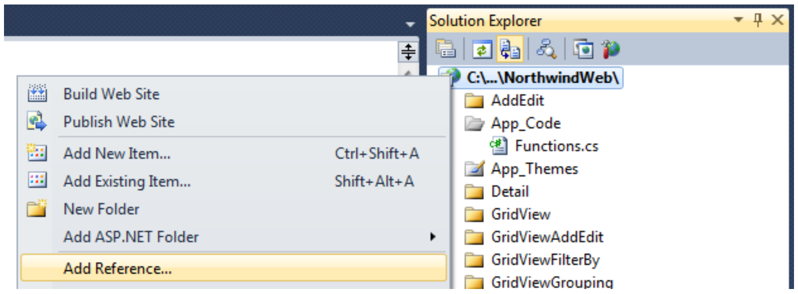
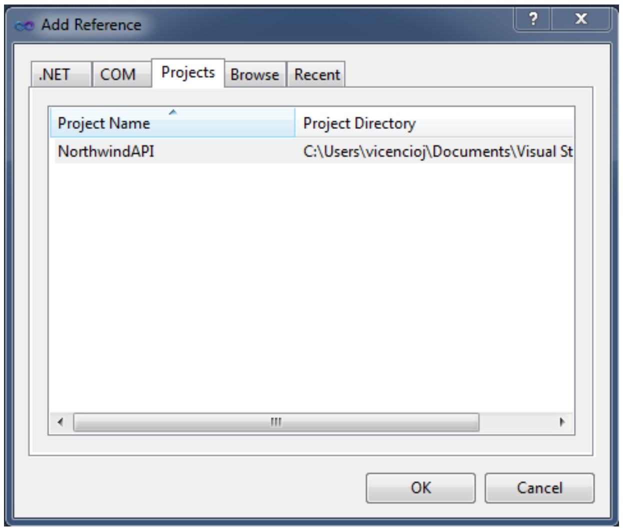


**Figure 59 Add a Reference**


Add Reference

.NET | COM | Projects | Browse | Recent

| Project Name | Project Directory |
|---|---|
| NorthwindAPI | C:\Users\vicencioj\Documents\Visual St |

**Figure 60 Add a Reference Dialog**

9. Click *OK* in Figure 60. Set *NorthwindWeb* as *StartUp Project*. See Figure 61.

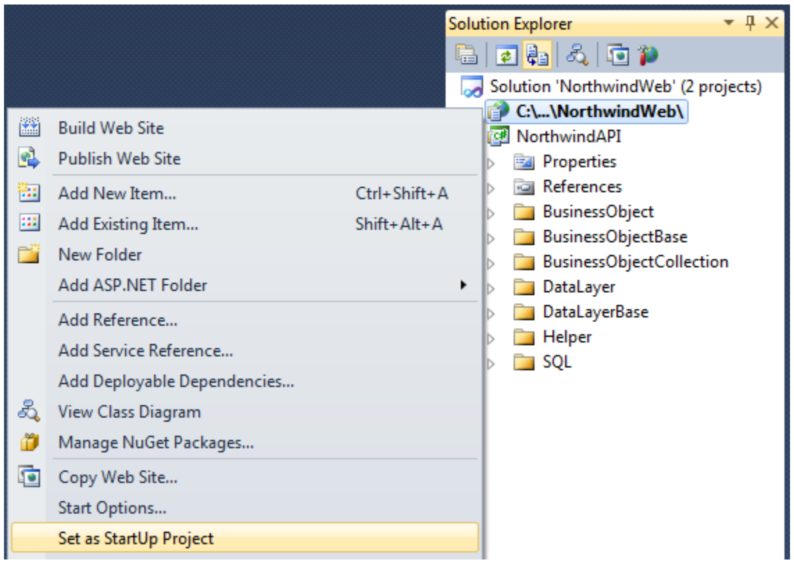


**Figure 60 Add a Reference Dialog**

10. You can now run the web site by pressing *F5*.

11. End of tutorial

## Example Classes

AspxFormsGen 4.0 is so easy to use, we even generated example code for you, all you have to do for most parts is copy and paste code.

The *Example* classes generated in the *Example* folder are example code that you can use in your client application. Each code example is placed in a method/function. You can copy the code inside each method/function into your client application. See examples below.

**Note: You don't need this in your application; it's just there to show you example code. In short, you can delete it, and it won't affect your application.**

For example you can copy a portion of the code from the *SelectAll* method/function and paste it in your client application. The example below shows how to sort the customers by company name in descending order.

**In C#**

```
// select all records
CustomersCollection objCustomersCol = Customers.SelectAll();

// Example 1:  you can optionally sort the collection in ascending order by your chosen field
objCustomersCol.Sort(Customers.ByCompanyName);

// Example 2:  to sort in descending order, add this line to the Sort code in Example 1
objCustomersCol.Reverse();
```

**In VB.NET**

```
' select all records
Dim objCustomersCol As CustomersCollection = Customers.SelectAll()

' Example 1:  you can optionally sort the collection in ascending order by your chosen field
objCustomersCol.Sort(Customers.ByCompanyName)

' Example 2:  to sort in descending order, add this line to the Sort code in Example 1
objCustomersCol.Reverse()
```

# Code Walk-Through

**Note: This will not walk you through all the generated code, instead, it will walk you through some code to get a general idea of how things flow.**

Let's see the events that take place when something is clicked, or loaded, etc. This discussion is mostly on generated web forms that use the JQuery validation, since ASP.NET validation is self-explanatory. There are common behaviors with the generated web forms that have a GridView web control.

## GridView's Data Source

Gridviews uses an *ObjectDataSource* web control to get its data. This is done by assigning the Gridview's *DataSourceID* to the *ObjectDataSources ID*. See code below.
**GridView:**

```
<asp:GridView ID="GridView1" runat="server" DataKeyNames="CustomerID" DataSourceID="ObjectDataSource1"...
```

**ObjectDataSource:**

```
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"...
```

## ObjectDataSource

The *ObjectDataSource* gets its data from the respective middle-tier object, specifically the *BusinessObject* classes. The specific *BusinessObject* class being referenced should contain the *SelectMethod* and *DeleteMethod* which are used by the *ObjectDataSource*.

The code below references the *Customers* middle tier class, *TypeName* shows the fully qualified name of the middle-tier object being referenced. It will look for a *SelectAll* method/function that has a *String* parameter called *sortExpression,* and a *Delete* method/function that expects a parameter called *CustomerID* in the *Customers* class.

**Note:** The *Customers* class (*BusinessObject*) does not have any methods at all (by default), but because it inherits from the respective base class *CustomersBase* (*BusinessObjectBase*) which contains all these methods and properties, the objects contained in the base class are now available to the derived class.

**Note: Always reference the *BusinessObject* class from any client code, nothing else.**

```
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
    TypeName="Northwind.BusinessObject.Customers"
    SelectMethod="SelectAll" SortParameterName="sortExpression"
    DeleteMethod="Delete" ondeleted="ObjectDataSource1_Deleted">
    <DeleteParameters>
          <asp:Parameter Name="CustomerID" Type="String" />
    </DeleteParameters>
</asp:ObjectDataSource>
```

**In C#**

```
public class Customers : CustomersBase
```

**In VB.NET**

```
Public Class Customers
    Inherits CustomersBase
```

## Common Behaviors for GridViews

1. **Sort Direction:** When you click the column heading of a gridview data is sorted and an arrow is added to the specific heading showing the sort direction for that column. Clicking the column once more will toggle the arrow direction and the sort direction to its opposite. The *SelectAll* method/function of the respective *BusinessObject* class is called every time you click the heading column.

2. **Paging:** The *SelectAll* method/function of the respective *BusinessObject* class is called every time you click the any of the paging numbers in the footer.

3. **Tooltip for Foreign Keys:** The tooltip functionality is not available to all the generated web forms that have a gridview web control. When a database table has foreign keys, these foreign keys are shown as a link in the gridview. When you hover your mouse over the link, the information about that foreign key pops-up. For web forms with tooltip functionality here's what happens:

   **Note:** Code examples for the tooltip discussion are taken from the (*GridView with Add, Edit, & Delete (Same Page)* type web form) *GridViewAddEdit* folder for *Products_Web.aspx*.

   a. When the web page loads, the *JQuery Tooltip* plugin is initialized by calling the *InitializeToolTip* function from the client.

   ```
   $(function () {
       InitializeDeleteConfirmation();
       InitializeAddEditRecord();
       InitializeToolTip();
       InitializeValidation();
   });
   ```

   b. This makes it possible to call the following code and show the tooltip. Code is removed for clearness.

   ```
   <div class="tag">
        <a href="#" class="gridViewToolTip"><%# Eval("SupplierID")%></a>
        <div id="tooltip" style="display: none;">
              <table>
                  <tr>
                     <td style="white-space: nowrap;"><b>Supplier ID:</b>&nbsp;</td>
                     <td><%# Eval("Suppliers.Value.SupplierID")%></td>
                  </tr>
                  <tr>
                      <td style="white-space: nowrap;"><b>Company Name:</b>&nbsp;</td>
                      <td><%# Eval("Suppliers.Value.CompanyName")%></td>
                  </tr>
   ```

   c. Notice the highlighted code above. This is a *Products* page, but it's referencing *Suppliers*, e.g. *Suppliers.Value.SupplierID*. This is because it's referencing a foreign key, which then references the related Supplier for this specific Product.

   d. Why does it use "*.Value*" for the Suppliers? E.g. *Suppliers.**Value**.SupplierID* instead of just *Suppliers.SupplierID*. This is because related tables use the Lazy loader pattern (lazy initialization), which retrieves related values **only when needed**. Notice that the *ProductsBase* (*BusinessObjectBase*) class contains the *Suppliers* property which is lazily loaded. Code is removed for clearness.

**In C#**

```
public Lazy<Suppliers> Suppliers
```

**In VB.NET**

```
 Public ReadOnly Property Suppliers() As Lazy(Of Suppliers)
```

4. **Deleting a Record**: The delete functionality is not available to all the generated web forms that have a gridview web control. But for web forms with delete functionality here's what happens:

   a. When the web page loads, the *JQuery UI's Dialog* widget is initialized by calling the *InitializeDeleteConfirmation* function from the client. This dialog confirms whether the user really wants to delete the specified item.

```
$(function () {
    InitializeDeleteConfirmation();
});
```

   b. When you click the trash can Image Button on the gridview, it calls the *deleteItem* javascript function. The *ID* of the item you want to delete is passed as the *AlternateText* value of the delete image button.

   c. The trash can button calls the *deleteItem* function passing the id (*CustomerID).*

```
<asp:ImageButton ID="IBtnDelete" runat="server" ToolTip="Click to delete"
    CommandArgument='<%# Eval("CustomerID") %>'
    OnClientClick="javascript:return deleteItem(this.name, this.alt);"
    ImageUrl="~/Images/Delete.png" AlternateText='<%# Eval("CustomerID") %>'
    CommandName="Delete" />
```

   d. The *deleteItem* function shows the JQuery UI dialog. Clicking the *Delete* button in the dialog does a post back (highlighted below), clicking *Cancel* just closes the dialog.

```
function deleteItem(uniqueID, itemID) {
    var dialogTitle = 'Permanently Delete Item ' + itemID + '?';

    $("#deleteConfirmationDialog").html('<p><span class="ui-icon ui-icon-
    alert" style="float:left; margin:0 7px 20px 0;">
    </span>Please click delete to confirm deletion.</p>');

    $("#deleteConfirmationDialog").dialog({
         title: dialogTitle,
         buttons: {
              "Delete": function () { __doPostBack(uniqueID, ''); $(this).dialog("close"); },
              "Cancel": function () { $(this).dialog("close"); }
         }
    });

    $('#deleteConfirmationDialog').dialog('open');
    return false;
}
```

   e. Clicking the *Delete* button in the dialog does a post back which calls the *Delete* method of the respective *BusinessObject* class, passing the ID (*CustomerID*) which is specified in *DeleteMethod* of the *ObjectDataSource*.

```
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
    TypeName="Northwind.BusinessObject.Customers"
    SelectMethod="SelectAll" SortParameterName="sortExpression"
    DeleteMethod="Delete" ondeleted="ObjectDataSource1_Deleted">
    <DeleteParameters>
        <asp:Parameter Name="CustomerID" Type="String" />
    </DeleteParameters>
</asp:ObjectDataSource>
```

f. If the deletion is successful, the item you deleted will be removed from the gridview.

g. If there's an error during deletion, an exception is raised and caught by the *ObjectDataSource1_Deleted* event of the *ObjectDataSource*.

```
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
    TypeName="Northwind.BusinessObject.Customers"
    SelectMethod="SelectAll" SortParameterName="sortExpression"
    DeleteMethod="Delete" ondeleted="ObjectDataSource1_Deleted">
    <DeleteParameters>
     <asp:Parameter Name="CustomerID" Type="String" />
    </DeleteParameters>
</asp:ObjectDataSource>
```

h. The *ObjectDataSource* calls the event shown below.

**in C#**

```
protected void ObjectDataSource1_Deleted(object sender,
System.Web.UI.WebControls.ObjectDataSourceStatusEventArgs e)
{
    Functions.ObjectDataSourceDeleted(sender, e, this.Page);
}
```

**In VB.NET**

```
Protected Sub ObjectDataSource1_Deleted(sender As Object,
e As System.Web.UI.WebControls.ObjectDataSourceStatusEventArgs)
    Functions.ObjectDataSourceDeleted(sender, e, Me.Page)
End Sub
```

i. Here's the code for the *ObjectDataSourceDeleted* from the *Functions* method.

**In C#**

```
public static void ObjectDataSourceDeleted(object sender,
System.Web.UI.WebControls.ObjectDataSourceStatusEventArgs e, Page page)
{
    if (e.Exception != null)
     {
        // show delete error
        string errorMessage = Functions.RemoveSpecialChars(e.Exception.InnerException.Message);
        ScriptManager.RegisterStartupScript(page, page.GetType(),
              "err_msg", "ShowError('" + errorMessage + "');", true);
        e.ExceptionHandled = true;
      }
}
```

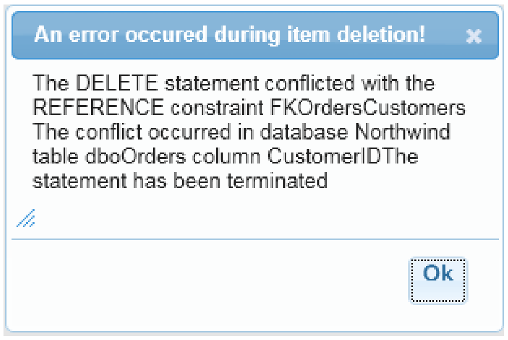
**In VB.NET**

```
Public Shared Sub ObjectDataSourceDeleted(sender As Object,
e As System.Web.UI.WebControls.ObjectDataSourceStatusEventArgs, page As Page)
    If e.Exception IsNot Nothing Then
        ' show delete error
        Dim errorMessage As String = Functions.RemoveSpecialChars(e.Exception.InnerException.Message)
        ScriptManager.RegisterStartupScript(page, page.[GetType](), "err_msg", "ShowError('" +
              errorMessage + "');", True)
        e.ExceptionHandled = True
    End If
End Sub
```

j. The code above calls a javascript function that shows the error to the user.

```
function ShowError(errorMessage) {
    $(document).ready(function () {
    $("#deleteErrorDialog").text(errorMessage);
    $("#deleteErrorDialog").dialog({
         modal: true,
         buttons: {
              Ok: function () {
                  $(this).dialog("close");
              }
         }
       });
    });
}
```

k. Error shown as a *JQuery UI Dialog* shown to user.


An error occured during item deletion!

The DELETE statement conflicted with the REFERENCE constraint FKOrdersCustomers The conflict occurred in database Northwind table dboOrders column CustomerIDThe statement has been terminated

5. **Adding a New Record**: The *"Add a New Record"* functionality is only available in the *GridView with Add, Edit, & Delete (Same Page)* type web form.

   **Note:** Code examples for the Adding a New Record discussion are taken from the (*GridView with Add, Edit, & Delete (Same Page)* type web form) *GridViewAddEdit* folder for *Products_Web.aspx*.

   a. When the web page loads, the client side click event handler for the *Add New Products* link and the *Cancel* button is initialized by calling the *InitializeAddEditRecord*.

```
$(function () {
    InitializeDeleteConfirmation();
    InitializeAddEditRecord();
    InitializeToolTip();
    InitializeValidation();
});
```

b. The *InitializeAddEditRecord* toggles the showing and hiding of the div (*divAddEditRecord*) tag which contains the fields to add or edit a record.

```
<div id="divAddEditRecord" class="ui-widget-content" style="display: ...code removed for clearness >
    ...code removed for clearness
</div>
```

c. The *InitializeAddEditRecord* also changes the box title to "*Add New Products*". This is because the same function is used to show or hide the div (*divAddEditRecord*) tag when you edit a record, discussed later.

d. When the *Add New Products* link is clicked the addItem javascript function is called, which also clears all the fields and resets all the validation errors if any.

```
function addItem() {
    clearFields();
    showHideItem(addEditTitle, null);
    resetValidationErrors();
    return false;
}
```

e. When the *Add Record* button is clicked, validation errors show if there are errors. The *JQuery Validation* plugin error is triggered when the requirement is not satisfied in the inline *CssClass*. E.g. the code below shows that the Product Name is required.

```
<asp:TextBox ID="TxtProductName" MaxLength="40"
    CssClass="{required:true, messages:{required:'Product Name is required!'}}" runat="server" />
```

f. The validation is possible because on the header of the web page we initialized the *JQuery Validation* plugin by loading the plugin and calling the *InitializeValidation* method.

**Loading the *JQuery Validation* plugin**

```
loadJavaScriptFile("../Scripts/jquery-1.4.1.min.js");
loadJavaScriptFile("../Scripts/jquery.validate.min.js");
```

**Calling the *InitializeValidation* method**

```
$(function () {
     InitializeDeleteConfirmation();
     InitializeAddEditRecord();
     InitializeToolTip();
     InitializeValidation();
});
```

g. In the InitializeValidation function we're telling the *JQuery Validation* plugin to validate everything under the *MasterPageForm1* (*id* of the *form* web control located in the master page). If there's an error, show the error in the next *td* tag from where the error occurred, and if it's valid we add a style named *success* (shows the check image) and a word "*ok!*"

**MasterPage's Form Web Control**

```
<form id="MasterPageForm1" class="cmxform" runat="server">
```

### InitializeValidation Function

```
function InitializeValidation() {
    validator = $("#MasterPageForm1").bind("invalid-form.validate", function () { }).validate({
        errorElement: "em",
        errorPlacement: function (error, element) {
            error.appendTo(element.parent("td").next("td"));
        },
        success: function (label) {
            label.text("ok!").addClass("success");
        }
    });
}
```

h. When you click the *Add Record* button and the page is valid, the button's event handler catches the click and calls the *AddOrUpdateRecord* method. Just like how the method sounds, it is used by both the adding of a new record as well as editing an existing record. This method uses the middle tier (*BusinessObject*) to add or edit a *Product*. When adding a new record, it instantiates a new *Product*, but when editing an existing record, it retrieves the *Product* by primary key. It then starts assigning values to the Product's properties and calls the appropriate operation towards the end.

### In C#

```
private void AddOrUpdateRecord(string operation)
{
    if (IsValid)
    {
        Products objProducts;

        if (operation == "update")
            objProducts =
              Northwind.BusinessObject.Products.SelectByPrimaryKey(Convert.ToInt32(HfldProductID.Value));
        else
        {
            objProducts = new Products();
        }

        objProducts.ProductName = TxtProductName.Text;
        objProducts.Discontinued = CbxDiscontinued.Checked;

        // Code removed for clearness...

        // the insert method returns the newly created primary key
        int newlyCreatedPrimaryKey;

        if (operation == "update")
            objProducts.Update();
        else
            newlyCreatedPrimaryKey = objProducts.Insert();

        GridView1.DataBind();
    }
}
```

### In VB.NET

```
Private Sub AddOrUpdateRecord(operation As String)
    If IsValid Then
        Dim objProducts As Products

        If operation = "update" Then
            objProducts = Northwind.BusinessObject.Products.SelectByPrimaryKey(
                                  Convert.ToInt32(HfldProductID.Value))
        Else
            objProducts = New Products()
```

```
                End If

                objProducts.ProductName = TxtProductName.Text
                objProducts.Discontinued = CbxDiscontinued.Checked

                ' Code removed for clearness...


                ' the insert method returns the newly created primary key
                Dim newlyCreatedPrimaryKey As Integer

                If operation = "update" Then
                    objProducts.Update()
                Else
                    newlyCreatedPrimaryKey = objProducts.Insert()
                End If

                GridView1.DataBind()
            End If
        End Sub
```

i. When the new record has been added, or the existing record has been update, the gridview is refresh to reflect the changes by calling the *GridView1.DataBind().*

6. **Editing an Existing Record**: The *"Clict to Edit Record*" functionality is only available in the *GridView with Add, Edit, & Delete (Same Page)* type web form.

   **Note:** Code examples for the Editing an Existing Record discussion are taken from the (*GridView with Add, Edit, & Delete (Same Page)* type web form) *GridViewAddEdit* folder for *Products_Web.aspx*.

   a. Please refer to *Adding a New Record's* letters *a, b,* and *c* about the loading procedure.

   b. When the *Pencil Image* in the gridview is clicked the *editItem* javascript function is called, which clears all the fields, resets all the validation errors if any.

```
function editItem(parameterName, itemID) {
    var paramNameArray = parameterName.split("|");
    var itemIDArray = itemID.split("|");
    var commaDelimParams = '';

    for (i = 0; i < paramNameArray.length; i++) {
        if (i == 0)
            commaDelimParams = commaDelimParams + "'" + paramNameArray[i] + "':'" + itemIDArray[i] + "'";
        else
            commaDelimParams = commaDelimParams + ",'" + paramNameArray[i] + "':'" + itemIDArray[i] + "'";
    }

    callWebMethod(urlAndMethod, commaDelimParams);
    showHideItem(addEditTitle, commaDelimParams);
    resetValidationErrors();
    return false;
}
```

   c. It also calls a *WebMethod* (web service, server-side) via JQuerys *$.ajax* (client-side) command.

```
function callWebMethod(urlAndMethod, parameter) {
    $.ajax({
        type: "POST",
        url: urlAndMethod,
        data: "{" + parameter + "}",
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: function (msg) {
            assignRetrievedItems(msg);
        }
    });
}
```

d. The *urlAndMethod* value is where the URL the method/function to access is located. This is assigned early on in the web page.

```
var urlAndMethod = "Products_Web.aspx/GetProducts";
```

e. This will call the *GetProducts* method/function (web service) from the *Products_Web.aspx* page. The *GetProducts* method/function retrieves the specific product by primary key using our middle tier object.

**In C#**

```
[WebMethod]
public static Products GetProducts(string productID)
{
    return Northwind.BusinessObject.Products.SelectByPrimaryKey(Convert.ToInt32(productID));
}
```

**In VB.NET**

```
<WebMethod()> _
Public Shared Function GetProducts(productID As String) As Products
    Return Northwind.BusinessObject.Products.SelectByPrimaryKey(Convert.ToInt32(productID))
End Function
```

f. Looking back at the *callWebMethod* code (in letter *c*), when the call to the web service is a success, it then calls the *assignRetrievedItems* function returning the *Product* business object from the server-side web service as JSON (*msg = Products* business object).

g. The *assignRetrievedItems* function assigns the retrieved Product client-side.

```
function assignRetrievedItems(msg) {
      $("#<%=HfldProductID.ClientID %>").val(msg.d.ProductID);
      $("#<%=TxtProductID.ClientID %>").attr('disabled', true);
       $("#<%=TxtProductID.ClientID %>").val(ConvertNullToString(msg.d.ProductID));
       $("#<%=TxtProductName.ClientID %>").val(ConvertNullToString(msg.d.ProductName));
       $("#<%=DdlSupplierID.ClientID %>").val(ConvertNullToString(msg.d.SupplierID));
       $("#<%=DdlCategoryID.ClientID %>").val(ConvertNullToString(msg.d.CategoryID));
       $("#<%=TxtQuantityPerUnit.ClientID %>").val(ConvertNullToString(msg.d.QuantityPerUnit));
       $("#<%=TxtUnitPrice.ClientID %>").val(ConvertNullToString(msg.d.UnitPrice));
       $("#<%=TxtUnitsInStock.ClientID %>").val(ConvertNullToString(msg.d.UnitsInStock));
       $("#<%=TxtUnitsOnOrder.ClientID %>").val(ConvertNullToString(msg.d.UnitsOnOrder));
       $("#<%=TxtReorderLevel.ClientID %>").val(ConvertNullToString(msg.d.ReorderLevel));
       $("#<%=CbxDiscontinued.ClientID %>").val(ConvertNullToString(msg.d.Discontinued));
  }
```

h. There are a few things to notice here: During an edit we disable the *TxtProductID* text box. Because it is disabled, we cannot access the value assigned to it server-side, so we use a hidden field (*HfldProductID*) so we can get the *ProductID* value server-side. See code behind below.

**In C#**

```
private void AddOrUpdateRecord(string operation)
{
    if (IsValid)
    {
        Products objProducts;

        if (operation == "update")
            objProducts =
               Northwind.BusinessObject.Products.SelectByPrimaryKey(Convert.ToInt32(HfldProductID.Value));
        else
        {
            objProducts = new Products();
        }
```

**In VB.NET**

```
Private Sub AddOrUpdateRecord(operation As String)
    If IsValid Then
        Dim objProducts As Products

        If operation = "update" Then
            objProducts =
               Northwind.BusinessObject.Products.SelectByPrimaryKey(Convert.ToInt32(HfldProductID.Value))
        Else
            objProducts = New Products()
        End If
```

i. The rest of the events will be similar to the *Add a New Record* shown above

# Requirements

- .Net Framework 4.0
- Microsoft SQL Server 2000, 2005, or 2008 or an Attached SQL Express.
- Windows Vista or Windows 7

# Limitations

- Generates English code only in either C# or VB.NET.
- Does not support retrieval of Large Value Data Type Columns (binary data types).
- Does not support new data types in MS SQL 2008 such as Geometry, Geography, HeirachyId, etc.
- Does not generate code for database tables that has no explicit primary key definition.
- Sorting an XML data type field is not supported.
- Sysname data types are not supported, although for most parts this will work with the generated code.
- User-Defined data types are not supported, although for most parts this will work with the generated code.
- Non alphanumeric characters in table names, view names, column names, etc are replaced by an underscore.

## Recommendations

- Username/password used in AspxFormsGen must have admin or enough priviledges in the database that you're going to work on.
- Use a local MS SQL Server if possible.
- Use No spaces when creating Table names or Field names in your database.
- Use alphanumeric characters only when creating Table names or Field names in your database.
- Use upper case letters for Table names and Field names if you have any plans of using Oracle in the future.
- Create explicit relationships for your tables using Diagrams.

## Notes

1. Not Available in the Express Edition.

2. The generated *DataLayerBase* Classes for the Express editions do not contain code that encapsulates stored procedures or dynamic SQL. You need to add the code yourself in the *DataLayer* class.

3. *JQuery UI-Plugin* is a free plugin that uses the JQuery Framework. Please see more info here: http://jqueryui.com

4. *JQuery Validation Plugin* is a free plugin that uses the JQuery Framework. Please see more info here: http://bassistance.de/jquery-plugins/jquery-plugin-validation/

5. *MS SQL Server 2008* is a product of Microsoft. Please see more info here: http://www.microsoft.com/sqlserver/2008/en/us/default.aspx

6. *Visual Studio 2010* is a product of Microsoft. Please see more info here: http://www.microsoft.com/visualstudio/en-us/products/2010-editions

7. *Northwind* Database is a product of Microsoft. Please see more info here: http://www.microsoft.com/download/en/details.aspx?id=23654

8. *JQuery Tooltip Plugin* is a free plugin that uses the JQuery Framework. Please see more info here: http://bassistance.de/jquery-plugins/jquery-plugin-tooltip/