

Table of Contents

Table of Contents

Overview	5
Tabs	6
Selected Tables	6
Load Tables Button	6
Clear Selection Button	6
Selected Views	6
Load Views Button.....	6
Clear Selection Button	6
Database Settings	6
Database Connection	6
Server	6
Database Name.....	6
User Name	6
Password.....	6
Show Password	6
Generated SQL.....	7
Stored Procedures	7
No Prefix or Suffix	7
Prefix.....	7
Suffix	7
Dynamic SQL	7
Code Settings	7
Database Object to Generate From	7
All Tables	7
All Views.....	7
Selected Tables Only	7
Selected Views Only.....	7
Website.....	7
Name.....	7

Directory and Browse Button	7
Business Layer and Data Layer	7
Namespace.....	7
Language	7
App Settings.....	7
Overwrite Files	8
Overwrite Dbase File.....	8
Automatically Open Selected Tables or Selected Views tab.....	8
Restore All Settings to Default	8
Buttons (Outside Tabs)	8
Generate...Button	8
All Tables.....	8
All Views	8
Selected Tables Only	8
Selected Views Only	8
Cancel Buton.....	9
About Button	9
Close Button.....	9
Generating Code.....	9
Create a Client Application.....	10
For All Tables.....	13
For All Views	28
For Selected Tables Only	31
For Selected Views Only	37
Generated Code.....	41
Middle-Tier Classes	41
BusinessObject Class.....	41
BusinessObjectBase Class	41
Methods.....	42
Properties.....	42
BusinessObjectCollection Class.....	42

Data-Tier Classes	42
DataLayer Class	43
DataLayerBase Class	43
Methods	43
Stored Procedures or Dynamic SQL Classes	43
Stored Procedures	43
Dynamic Class	43
Stored Procedures or Methods Generated by AspxCodeGen 4.0	44
SelectAll	49
SelectByPrimaryKey	49
SelectDropDownListData	49
SelectCollectioyBy Foreign Key	44
Insert	44
Update	44
Delete	44
Example Classes	44
Helper Classes	44
Dbase Class	44
Miscellaneous Files	44
GeneratedCode.htm	44
Adding Your Own Code	45
Helper Files	45
Dbase File	45
In C#	45
In VB.NET	45
Middle Tier Class	46
In C#	46
In VB.NET	47

Data Tier Class	48
In C#	48
In VB.NET.....	49
Stored Procedures	51
Dynamic SQL	52
In C#	52
In VB.NET.....	53
Using The Generated Middle Tier in Your Code	54
Tutorial on Creating a Class Library	54
Example Classes	58
In C#	58
In VB.NET.....	58
Requirements	58
Limitations	58
Recommendations	59
Notes	59

Overview

AspxCodeGen 4.0 generates Middle-Tier, Data-Tier, and Stored Procedures (or Dynamic SQL) in One Click.

To keep AspxCodeGen 4.0 simple, there's only one main interface as shown in Figure 1. The main window consists of six (5) tabs.

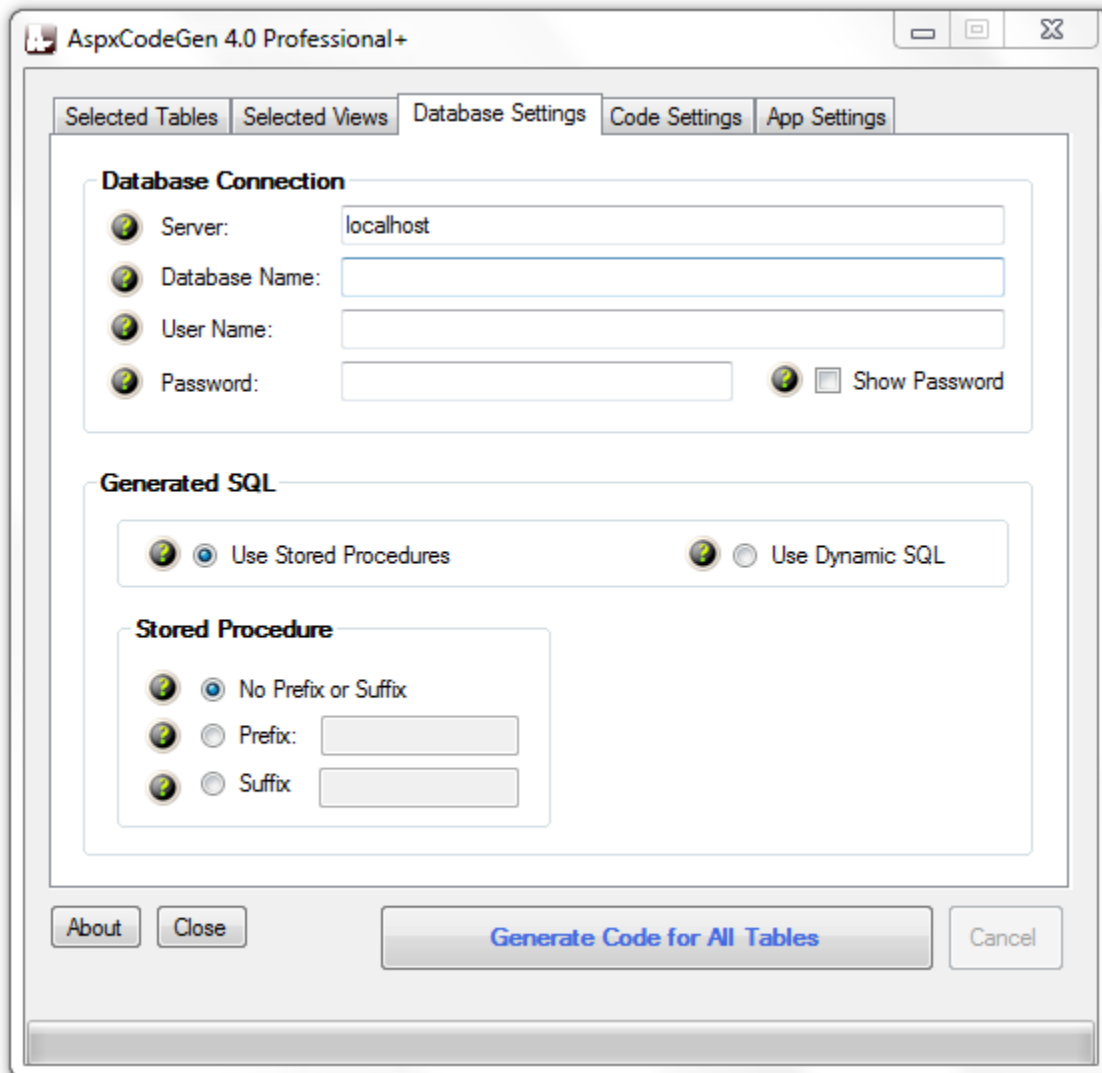


Figure 1 Main Window

Tabs

Selected Tables

AspxCodeGen 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 AspxCodeGen 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 AspxCodeGen 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 or 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 AspxCodeGen 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.

App Settings

These are application settings. Almost all generated code/web forms are overwritten every time you use AspxCodeGen 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 AspxCodeGen 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 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.
- **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.
- **Restore All Settings to Default:** AspxCodeGen 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 AspxCodeGen 4.0.
- **Close Button:** Closes the application.

Generating 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.

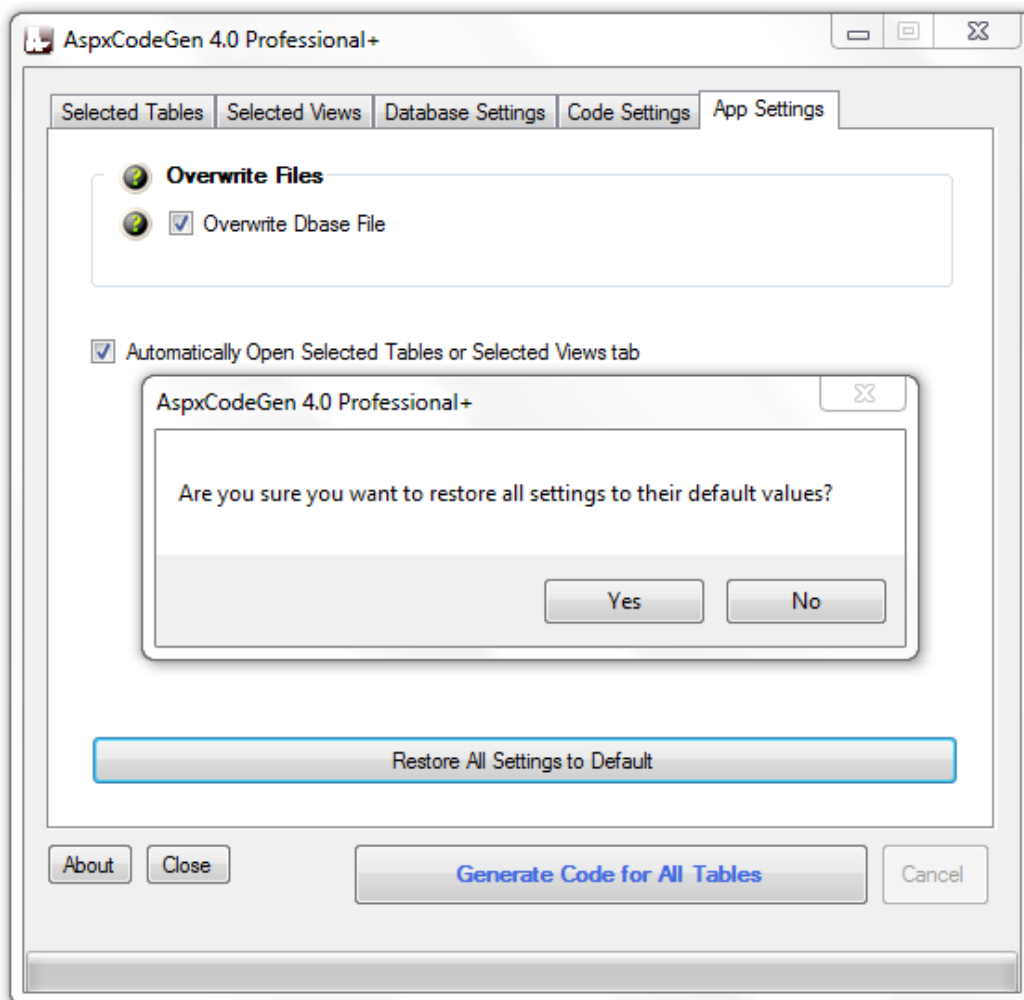


Figure 2 Restore All Settings To Defaults

Create a Client Application

AspCodeGen 4.0 generates middle-tier, data-tier, and stored procedures or dynamic SQL. But it needs a client application so you can output the responses to your user. You can use any client application, e.g. web forms, win forms, web services, wcf, Silverlight, etc. If you're a middle-tier or data-tier developer, you don't really need a client application to access AspCodeGen's generated code; you can simply put the generated code in a *Class Library* project so the code can be reused by other applications.

For the purposes of this tutorial, we need to create a client application. We will create a web site as our client application.

1. Create a website using *Visual Studio 2010* targeting the 4.0 .NET framework. Open *Visual Studio 2010*.
2. Select *File, New Web Site*.

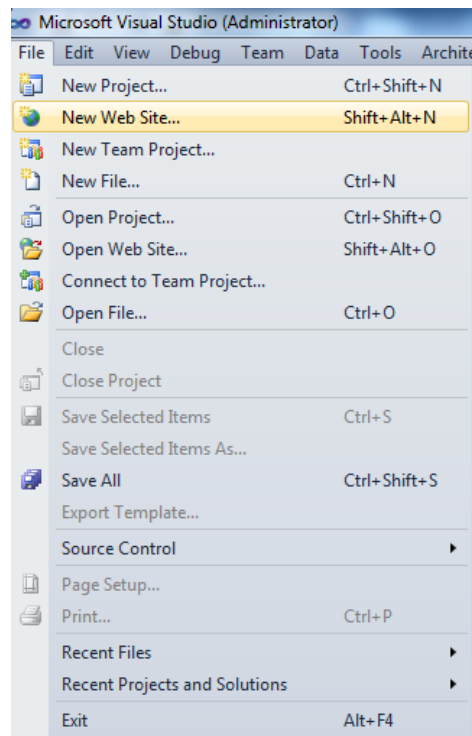


Figure 3 Creating a New Web Site

3. In the *New Web Site* dialog, Select *Visual C# (or Visual Basic)* under *Installed Templates*, select *.NET Framework 4, ASP.NET Empty Web Site*, name the website "*NorthwindWeb*", then click the *OK* button. See Figure 4.
4. A new empty website has now been created, with just the *web.config* file. See Figure 5.
5. Right-click on the *NorthwindWeb* node, select *Add ASP.NET Folder*, then choose *App_Code*. See Figure 6. This is where we're going to place the generated code later on.

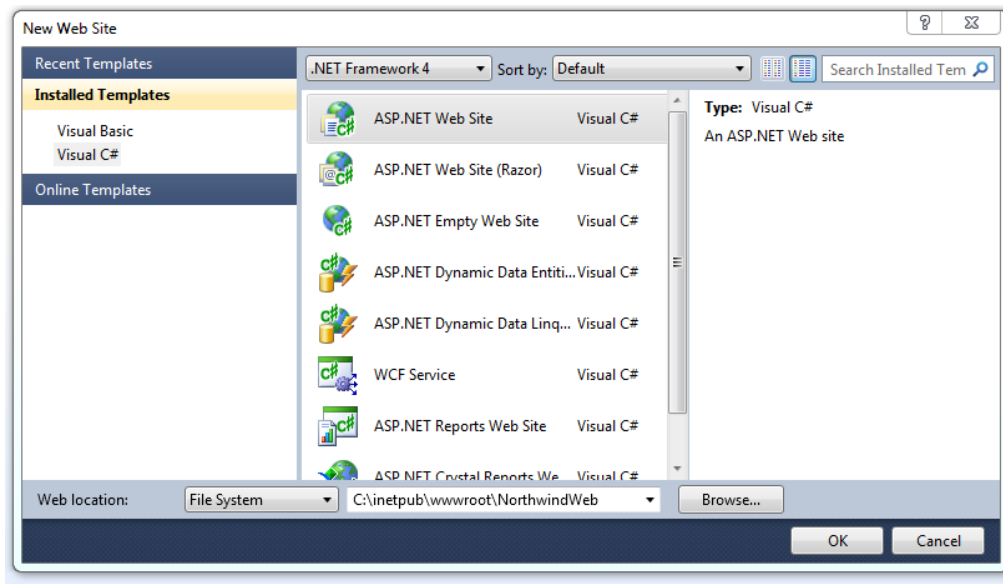


Figure 4 New Web Site

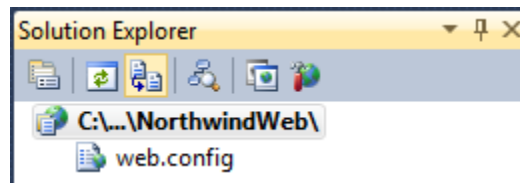


Figure 5 Empty Web Site

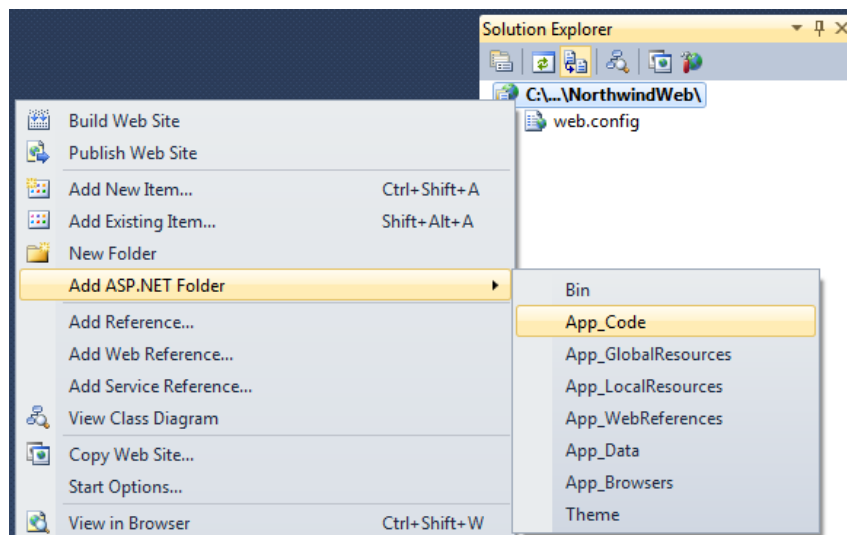


Figure 6 Add ASP.Net Folder App_Code

- Now add a web form. Right-click on the *NorthwindWeb* node and then choose *Add New Item*. See Figure 7.

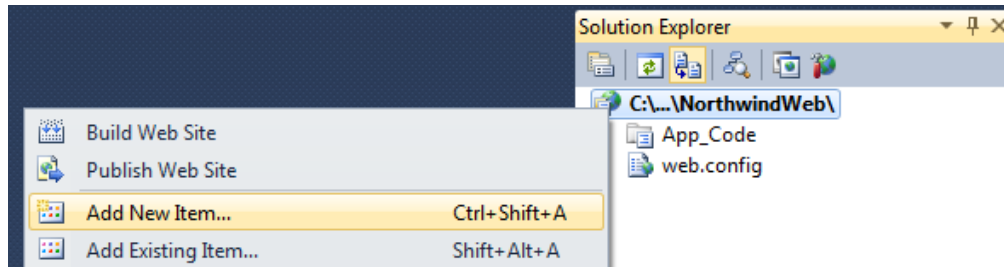


Figure7 Add New Item

7. Choose *Web Form*, then name the web form *Default.aspx*, then click the *Add* button, see Figure 8.

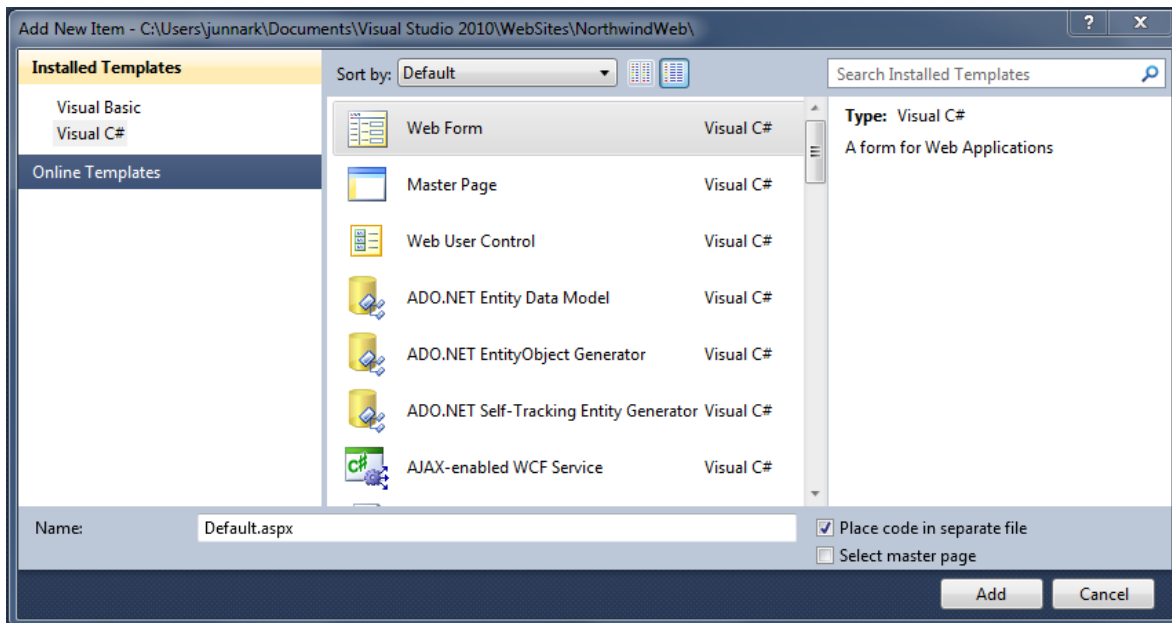


Figure 8 Add New Item Dialog

8. The structure of the newly created website should resemble Figure 9. We will go back to this website later, for now we will be generating code using AspxCodeGen 4.0.

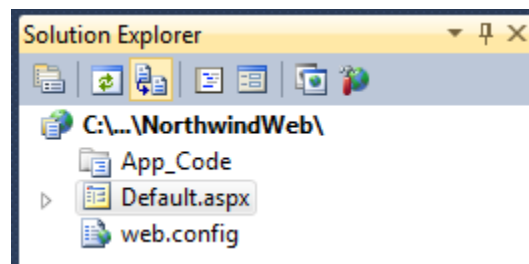


Figure 9 Structure of the New Web Site

9. We're done. We'll come back and reference this web site later.

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 AspxCodeGen 4.0 remembers your settings, the next time you open AspxCodeGen 4.0, all you have to do is click the *Generate...* button, this has always been our signature feature. See Figure 10.

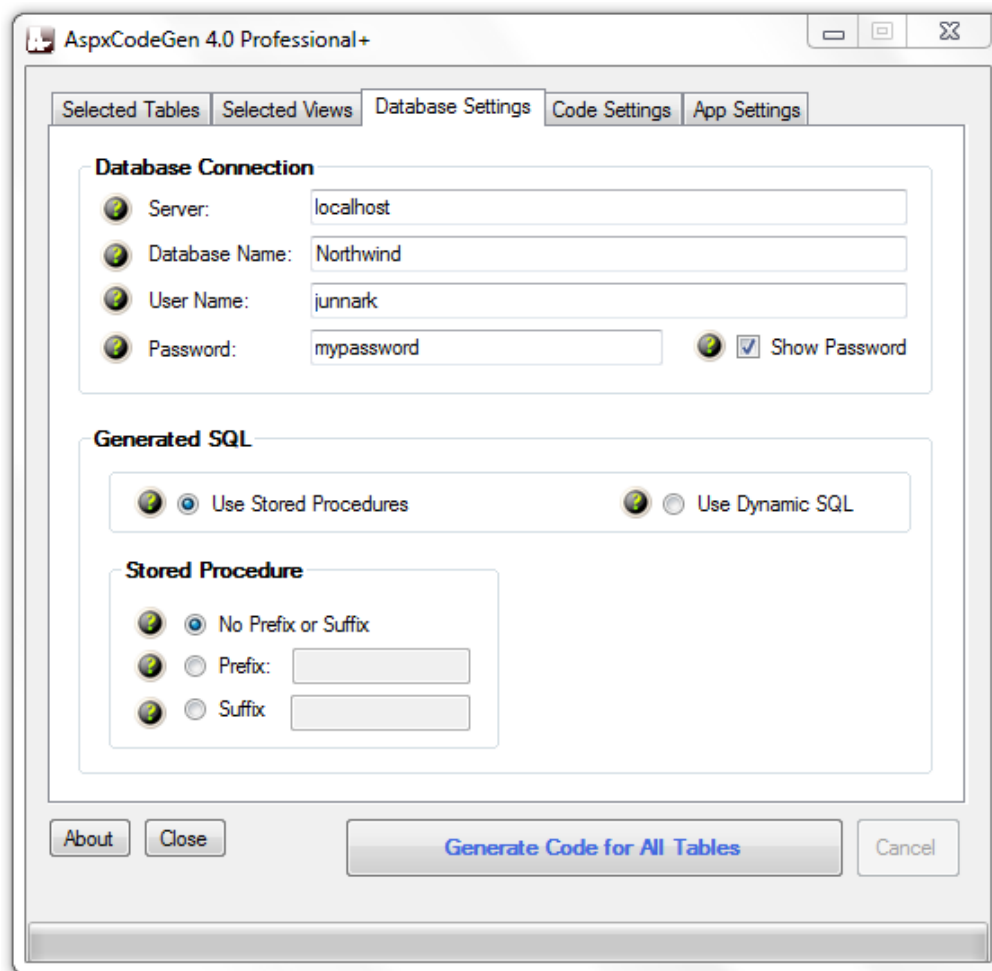


Figure 10 Database Settings

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. Notice that we're going to put the generated code in the *App_Code* folder of our client application (the web site we created earlier). See Figure 11.

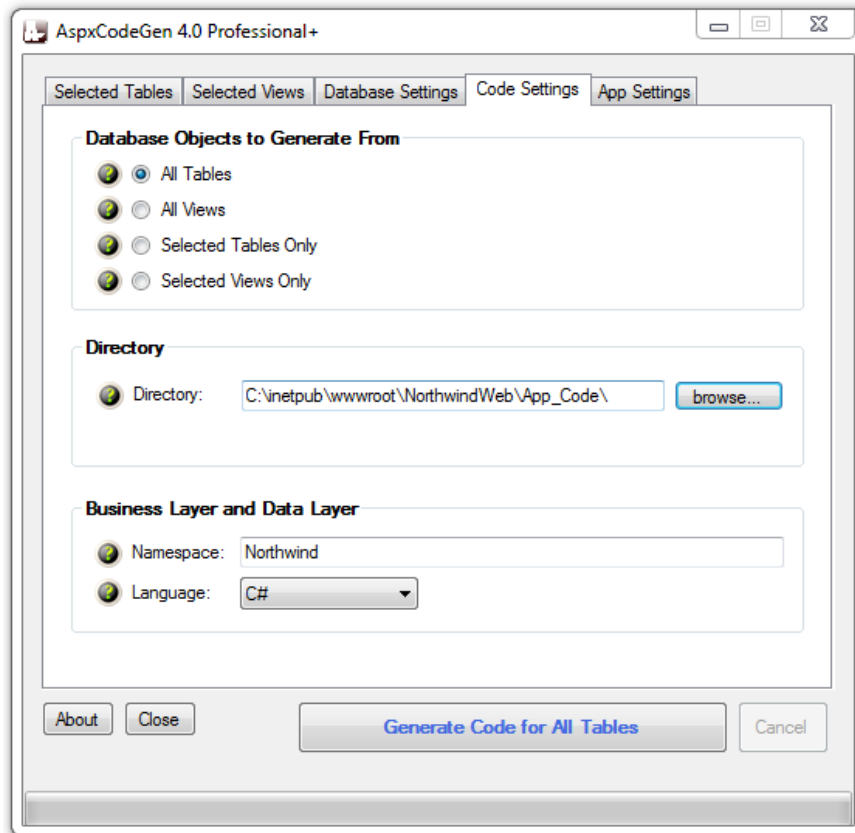


Figure 11 Code Settings - All Tables

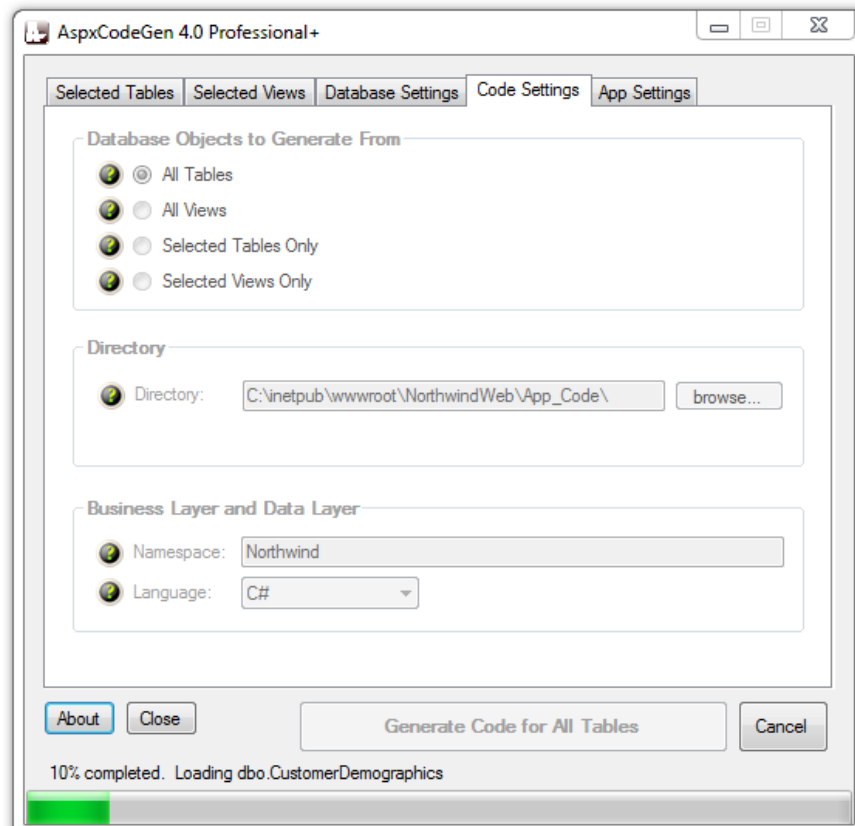


Figure 12 Generate Code for All Tables

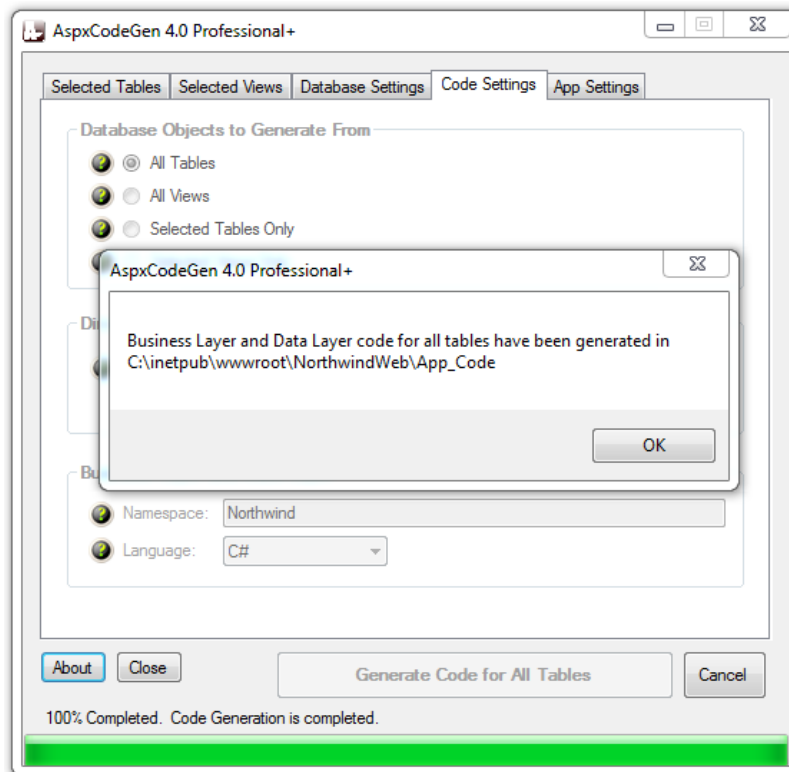


Figure 13 Code Generation is Done

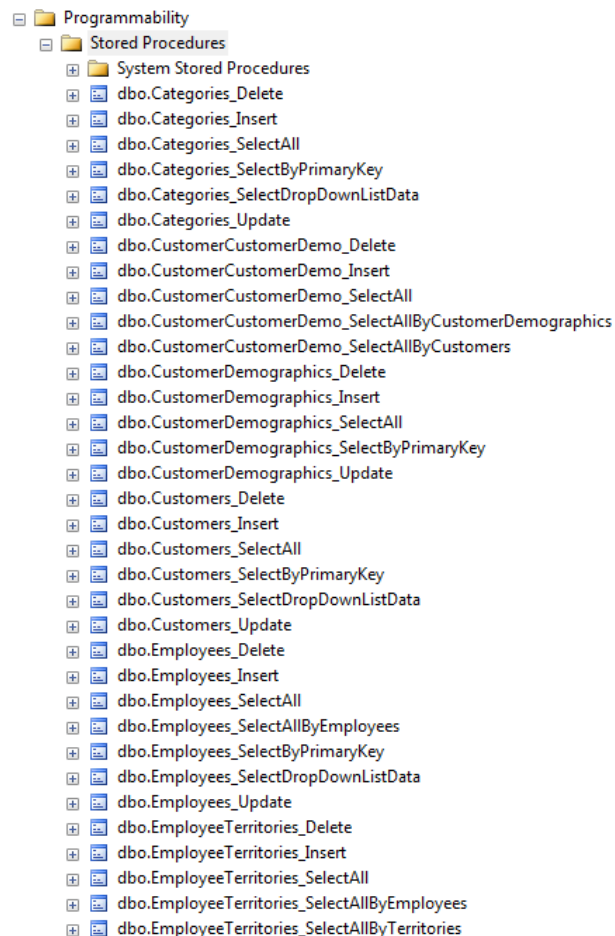


Figure 14 List of Generated Stored Procedures

4. Click the *Generate Code for All Tables* button. See Figure 12.
5. Once code generation is done, a message is shown, click *OK*. See Figure 13.
6. 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 14.
7. Let's go back to the web site we created earlier. Open *Visual Studio 2010*. On the *File* menu click *Open Web Site*. See Figure 15.
8. Point to the directory of the *NorthwindWeb* website, and then click *Open*. See Figure 16.
9. Expand the *App_Code* folder and you will see the generated code arranged in folders. See Figure 17.
10. The middle-tier classes can be found in folders:
 - a. *BusinessObject*
 - b. *BusinessObjectBase*
 - c. *BusinessObjectCollection*
11. The data-layer classes can be found in folders:
 - a. *DataLayer*
 - b. *DataLayerBase*

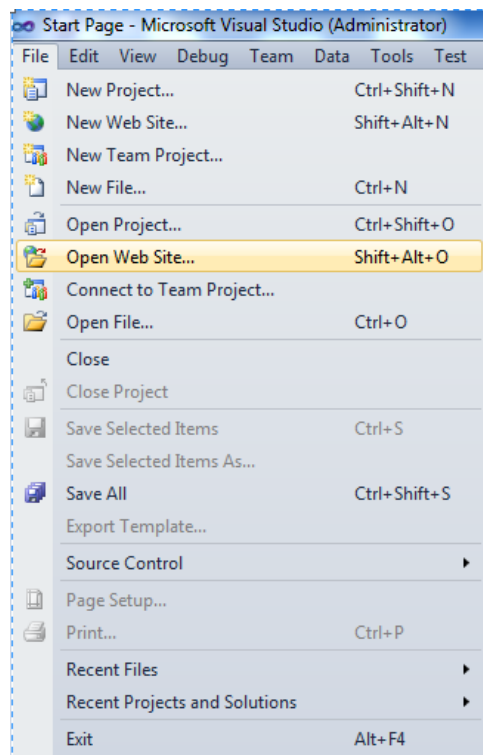


Figure 15 Open Web Site

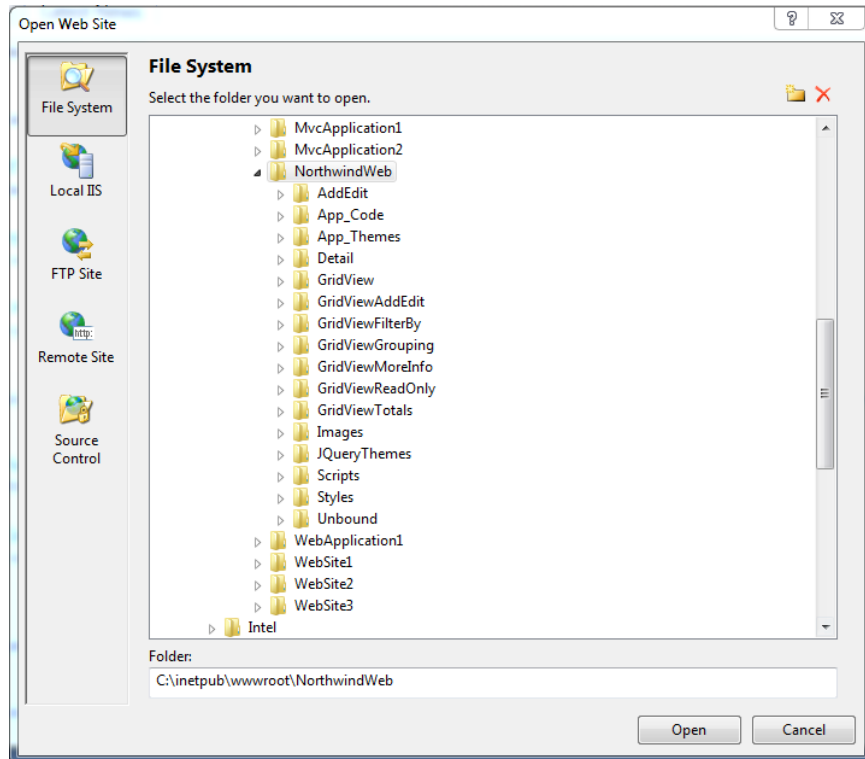


Figure 16 Generated Web Site Directory

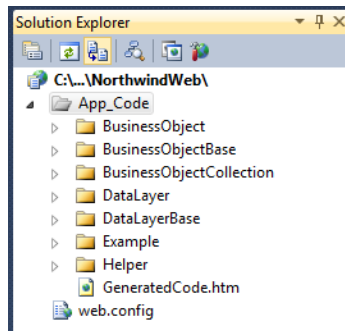


Figure 17 Generated Code

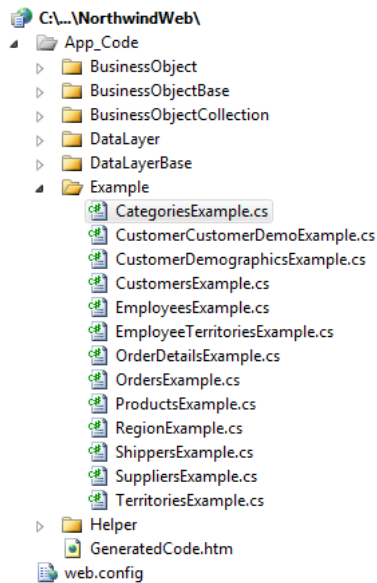


Figure 18 Example Folder

```

using ...
// using System.Windows.Forms; // Note: remove comment

/// <summary> ...
public sealed class CategoriesExample
{
    private CategoriesExample() ...

    private void SelectAll() ...

    private void SelectByPrimaryKey() ...

    /// <summary> ...
    private void SelectCategoriesDropDownListData() ...

    private void Insert() ...

    private void Update() ...

    private void Delete() ...
}

```

Figure 19 CategoriesExample.cs in Colapsed mode

```

Imports System
Imports Northwind.BusinessObject
Imports System.Web.UI.WebControls
' Imports System.Windows.Forms ' Note: remove comment w

These are data-centric code examples for the Categories_t
Public NotInheritable Class CategoriesExample

    Private Sub New() ...

    Private Sub SelectAll() ...

    Private Sub SelectByPrimaryKey() ...

    Selects CategoryID and CategoryName columns for use wi
    Private Sub SelectCategoriesDropDownListData() ...

    Private Sub Insert() ...

    Private Sub Update() ...

    Private Sub Delete() ...

End Class

```

Figure 20 CategoriesExample.vb in Colapsed mode

12. For now, double-click on the *Example* folder under the *App_Code*. As you can see there are 13 code examples classes, one for each of the tables in the *Northwind* database. See Figure 18.
13. Double-click on the *CategoriesExample.cs* class. The class will be opened up in the *Visual Studio IDE*. See Figures 19 and 20 in collapsed mode.
14. Now let's add some code to the *Default.aspx* web form. Copy and paste the completed code for the *Default.aspx* page. Note make sure to change the *Language* to "VB" and the extension to ".vb" for VB.NET (highlighted below).

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!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 id="Head1" runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
      <asp:Button ID="BtnSelectCategoryByPrimaryKey"
        Text="Select Category By Primary Key" runat="server"
        onclick="BtnSelectCategoryByPrimaryKey_Click" />&nbsp;&nbsp;&nbsp;&nbsp; 

      <asp:Button ID="BtnInsert" runat="server"
        Text="Add a Category"
        onclick="BtnInsert_Click" />&nbsp;&nbsp;&nbsp;&nbsp; 

      <asp:Button ID="BtnUpdate" runat="server"
        Text="Update a Category"
        onclick="BtnUpdate_Click" />&nbsp;&nbsp;&nbsp;&nbsp; 

      <asp:Button ID="BtnSelectAll" runat="server"
        Text="SelectAllCategories"
        onclick="BtnSelectAll_Click" />&nbsp;&nbsp;&nbsp;&nbsp; 

      <asp:Button ID="BtnDelete" runat="server"
        Text="Delete a Category"
        onclick="BtnDelete_Click" />

      <br /><br />
      <asp:Literal ID="LitContent" runat="server" />

      <br /><br />
      <asp:GridView ID="GridView1" AutoGenerateColumns="true" runat="server" />
    </div>
  </form>
</body>
</html>
```

Code Behind in C#

```
using System;

public partial class _Default : System.Web.UI.Page
{
    protected void BtnSelectCategoryByPrimaryKey_Click(object sender, EventArgs e)
    {
    }

    protected void BtnInsert_Click(object sender, EventArgs e)
    {
    }

    protected void BtnUpdate_Click(object sender, EventArgs e)
    {
    }

    protected void BtnSelectAll_Click(object sender, EventArgs e)
    {
    }

    protected void BtnDelete_Click(object sender, EventArgs e)
    {
    }
}
```

Code Behind in VB.NET

```

Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub BtnSelectCategoryByPrimaryKey_Click(sender As Object, e As System.EventArgs)
        Handles BtnSelectCategoryByPrimaryKey.Click
    End Sub

    Protected Sub BtnInsert_Click(sender As Object, e As System.EventArgs) Handles BtnInsert.Click
    End Sub

    Protected Sub BtnUpdate_Click(sender As Object, e As System.EventArgs) Handles BtnUpdate.Click
    End Sub

    Protected Sub BtnSelectAll_Click(sender As Object, e As System.EventArgs) Handles BtnSelectAll.Click
    End Sub

    Protected Sub BtnDelete_Click(sender As Object, e As System.EventArgs) Handles BtnDelete.Click
    End Sub
End Class

```

15. Go back to the *CategoriesExample* class, and then copy the code inside the *SelectByPrimaryKey* method to the *BtnSelectCategoryByPrimaryKey* button click event in the *Default.aspx.cs* (or *.vb*). It should look like the code below. This code will retrieve a category from the database that has a primary key which is 1. It will also retrieve all the related *Products* that has a categoryID = 1. The related *Products* is null until you use the *.Value* extension method, this uses the lazy loader pattern which only fetches data when we need them. This gives us a light-weight *objCategories* object. Note: Make sure to resolve the reference for the *Northwind.BusinessObject*.

In C#

```

using System;
using Northwind.BusinessObject;

public partial class _Default : System.Web.UI.Page
{
    protected void BtnSelectCategoryByPrimaryKey_Click(object sender, EventArgs e)
    {
        // select a record by primary key(s)
        Categories objCategories = Categories.SelectByPrimaryKey(1);

        if (objCategories != null)
        {
            // if record is found, a record is returned
            int categoryID = objCategories.CategoryID;
            string categoryName = objCategories.CategoryName;
            string description = objCategories.Description;
        }
    }
}
Code removed for cleanness...

```

In VB.NET

```

Imports Northwind.BusinessObject

Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub BtnSelectCategoryByPrimaryKey_Click(sender As Object, e As System.EventArgs)
        Handles BtnSelectCategoryByPrimaryKey.Click

        ' select a record by primary key(s)
        Dim objCategories As Categories = Categories.SelectByPrimaryKey(1)

        If objCategories IsNot Nothing Then

```

```

        ' if record is found, a record is returned
        Dim categoryID As Integer = objCategories.CategoryID
        Dim categoryName As String = objCategories.CategoryName
        Dim description As String = objCategories.Description
    End If
End Sub
Code removed for cleanness...

```

16. Modify the code in the *BtnSelectCategoryByPrimaryKey* button event so we can output the retrieved Category in the *Default.aspx* web form. Make sure to reference the *System.Text* in the header of the class, e.g. *using System.Text;* (or *Imports System.Text*).

In C#

```

StringBuilder sb = new StringBuilder();

// select a record by primary key(s)
Categories objCategories = Categories.SelectByPrimaryKey(1);

if (objCategories != null)
{
    sb.Append("Category ID: " + objCategories.CategoryID + "<br/>");
    sb.Append("Name: " + objCategories.CategoryName + "<br/>");
    sb.Append("Description: " + objCategories.Description + "<br/>");

    // output the category in the Literal control
    LitContent.Text = sb.ToString();

    if (objCategories.ProductsCollection.Value != null)
    {
        if (objCategories.ProductsCollection.IsValueCreated)
        {
            // output all the related products in a GridView control
            GridView1.DataSource = objCategories.ProductsCollection.Value;
            GridView1.DataBind();
        }
    }
}

```

In VB.NET

```

Dim sb As New StringBuilder

' select a record by primary key(s)
Dim objCategories As Categories = Categories.SelectByPrimaryKey(1)

If objCategories IsNot Nothing Then
    sb.Append("Category ID: " & objCategories.CategoryID & "<br/>")
    sb.Append("Name: " & objCategories.CategoryName & "<br/>")
    sb.Append("Description: " & objCategories.Description & "<br/>")

    ' output the category in the Literal control
    LitContent.Text = sb.ToString()
End If

```

17. Run the website by clicking *F5*. A pop-up may show asking if you want to modify the *web.config*, click the *OK* button to modify the *web.config* file. And then you'll see the web page. See Figure 21.

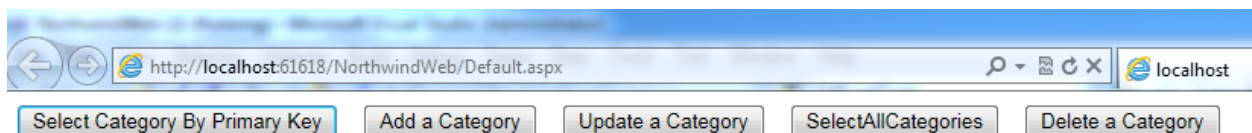


Figure 21 Default.aspx Web Page

18. Click the *Select Category By Primary Key* button. The specific Category is retrieved from the database and written in the web page. See Figure 22.

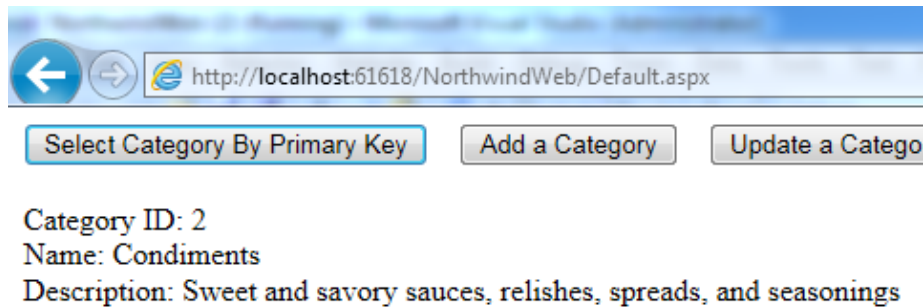


Figure 22 Select By Primary Key

19. Close the browser and let's add more code to the *Default.aspx.cs (.vb)* code file. Go back to the *CategoriesExample* class, then copy the code inside the *Insert()* method to the *BtnInsert_Click* button event in the *Default.aspx.cs (.vb)*. Since *CategoryID* is an auto-generating primary key (e.g. like an identity filled), we don't need to supply the *CategoryID*, make sure to supply a valid value for all the required properties. Finally, inserting a record in a table will return the inserted primary key. Add the highlighted code (last line) to output the *newlyCreatedPrimaryKey*.

In C#

```
protected void BtnInsert_Click(object sender, EventArgs e)
{
    // first instantiate a new Categories
    Categories objCategories = new Categories();

    // assign values you want inserted
    objCategories.CategoryName = "123 My New Category";
    objCategories.Description = "A new category for demo";

    // finally, insert a new record
    // the insert method returns the newly created primary key
    int newlyCreatedPrimaryKey = objCategories.Insert();

    LitContent.Text = "New Category ID: " + newlyCreatedPrimaryKey;
}
```

In VB.NET

```
Protected Sub BtnInsert_Click(sender As Object, e As System.EventArgs) Handles BtnInsert.Click
    ' first instantiate a new Categories
    Dim objCategories As New Categories()

    ' assign values you want inserted
    objCategories.CategoryName = "aaaa abcde"
    objCategories.Description = "here you go"

    ' finally, insert a new record
    ' the insert method returns the newly created primary key
    Dim newlyCreatedPrimaryKey As Integer = objCategories.Insert()

    LitContent.Text = "New Category ID: " & newlyCreatedPrimaryKey
End Sub
```

20. Run the website by clicking *F5*. Click the *Add a Category* button and you should see Figure 23. The best way to check if the new category was added is to query the *Categories* table in the *Northwind* database, see the query editor snapshot in Figure 24.

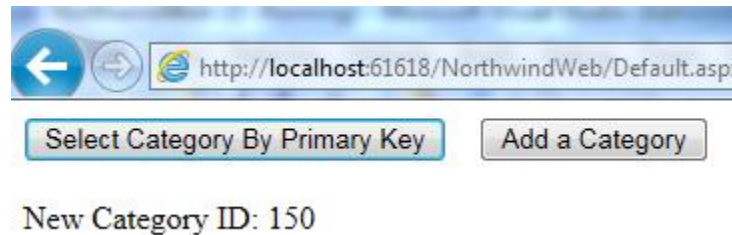


Figure 23 Insert a Record

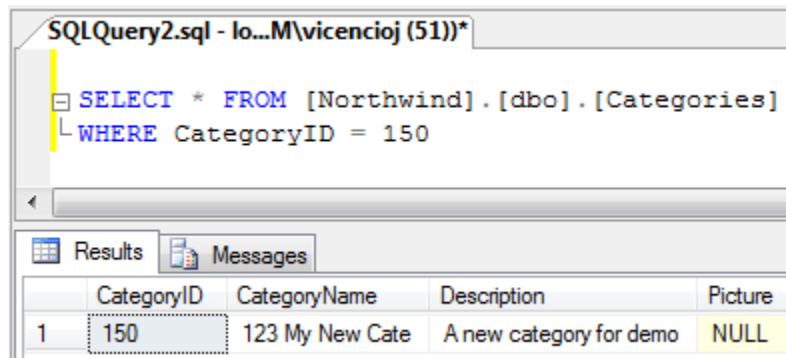


Figure 24 Inserted Record

21. Close the browser and let's add more code to the *Default.aspx* code behind. Go back to the *CategoriesExample* class, and then copy the code inside the *Update()* method to the *BtnUpdate_Click* button event in the *Default.aspx* code behind. Change the text in the *CategoryName* and *Description*, and then add the highlighted Literal assignment as shown below. Also make sure to supply a primary key that exist in your database.

In C#

```
protected void BtnUpdate_Click(object sender, EventArgs e)
{
    // first instantiate a new Categories
    Categories objCategories = new Categories();

    // assign the existing primary key(s)
    // of the record you want updated
    objCategories.CategoryID = 150;

    // assign values you want updated
    objCategories.CategoryName = "123 Updated";
    objCategories.Description = "A new category for demo Updated";

    // finally, update an existing record
    objCategories.Update();

    LitContent.Text = "Updated Category ID: " + objCategories.CategoryID;
}
```

In VB.NET

```

Protected Sub BtnUpdate_Click(sender As Object, e As System.EventArgs) Handles BtnUpdate.Click
    ' first instantiate a new Categories
    Dim objCategories As New Categories()

    ' assign the existing primary key(s)
    ' of the record you want updated
    objCategories.CategoryID = 150

    ' assign values you want updated
    objCategories.CategoryName = "123 Updated"
    objCategories.Description = "A new category for demo Updated"

    ' finally, update an existing record
    objCategories.Update()

    LitContent.Text = "Updated Category ID: " & objCategories.CategoryID
End Sub

```

22. Run the website by clicking *F5*. Click the *Update a Category* button and you should see Figure 23. The best way to check if the new category was added is to query the *Categories* table in the *Northwind* database, see the query editor snapshot in Figure 25.

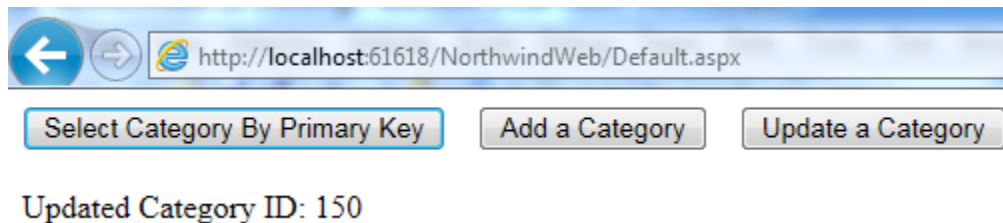


Figure 25 Update a Record

23. Figure 26 shows that the specified record has been updated in the database.

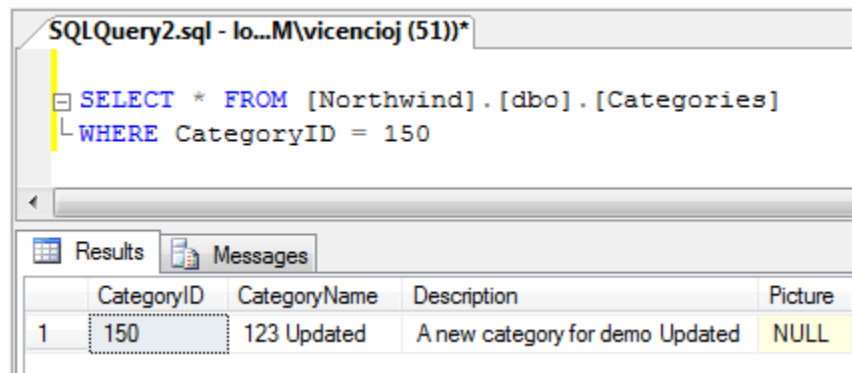


Figure 26 Updated Record

24. Close the browser and let's add more code. Go back to the *CategoriesExample* class, then copy the code inside the *SelectAll()* method to the *BtnSelectAll_Click* button event in the *Default.aspx* code behind. See code below.

In C#

```

// select all records
CategoriesCollection objCategoriesCol = Categories.SelectAll();

```



```
// Example 1: you can optionally sort the collection in ascending order by your chosen field
objCategoriesCol.Sort(Categories.ByCategoryName);

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

// Example 3: directly bind to a GridView
GridView grid = new GridView();
grid.DataSource = objCategoriesCol;
grid.DataBind();

// Example 4: loop through all the Categories(s)
foreach (Categories objCategories in objCategoriesCol)
{
    int categoryID = objCategories.CategoryID;
    string categoryName = objCategories.CategoryName;
    string description = objCategories.Description;
}
```

In VB.NET

```
' select all records
Dim objCategoriesCol As CategoriesCollection = Categories.SelectAll()

' Example 1: you can optionally sort the collection in ascending order by your chosen field
objCategoriesCol.Sort(Categories.ByCategoryName)

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

' Example 3: directly bind to a GridView
Dim grid As GridView = New GridView()
grid.DataSource = objCategoriesCol
grid.DataBind()

' Example 4: loop through all the Categories
For Each objCategories As Categories In objCategoriesCol
    Dim categoryID As Integer = objCategories.CategoryID
    Dim categoryName As String = objCategories.CategoryName
    Dim description As String = objCategories.Description
Next
```

25. Let's modify the code in the *BtnSelectAll_Click* button event so we can output all the retrieved categories in the *Default.aspx* web form. Each of the category has related products (products by *CategoryID*), but we're not going to worry about this for this example. Let's remove and modify some code. This code will output all the categories in the GridView.

In C#

```
protected void BtnSelectAll_Click(object sender, EventArgs e)
{
    // select all records
    CategoriesCollection objCategoriesCol = Categories.SelectAll();

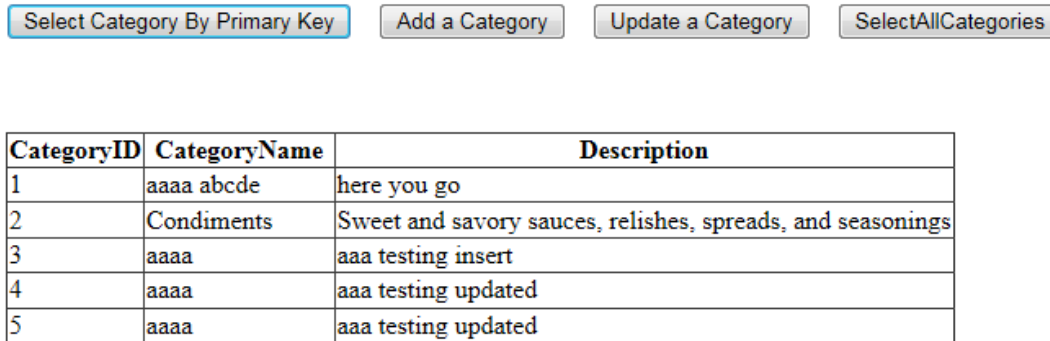
    // Example 3: directly bind to a GridView
    GridView1.DataSource = objCategoriesCol;
    GridView1.DataBind();
}
```

In VB.NET

```
Protected Sub BtnSelectAll_Click(sender As Object, e As System.EventArgs) Handles BtnSelectAll.Click
    ' select all records
    Dim objCategoriesCol As CategoriesCollection = Categories.SelectAll()
```

```
' Example 3: directly bind to a GridView
GridView1.DataSource = objCategoriesCol
GridView1.DataBind()
End Sub
```

26. Run the website by clicking *F5*. And then click the *Select All Categories*. See Figure 27.



CategoryID	CategoryName	Description
1	aaaa abcde	here you go
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
3	aaaa	aaa testing insert
4	aaaa	aaa testing updated
5	aaaa	aaa testing updated

Figure 27 Select All

27. Close the browser. Let's examine the code above just a bit more. When retrieving collections using the AspxCodeGen 4.0 generated code, you are able to sort ascending and sort descending by using the *Sort* and *Reverse* extension methods respectively as seen above. To sort ascending just call the *Sort* extension method, to sort descending, you need to call the *Sort* first, then the *Reverse* extension methods.

28. Each of the *BaseBusinessObject* class properties can be sorted on (with some restrictions); in this case we can sort *ByCategoryID*. See Figure 24.

```
protected void BtnSelectAll_Click(object sender, EventArgs e)
{
    // select all records
    CategoriesCollection objCategoriesCol = Categories.SelectAll();

    // you can optionally sort ascending the collection by your chosen field
    objCategoriesCol.Sort(Categories.ByCategoryName);

    // to sort descending, add this line
    objCategoriesCol.Reverse();
}
```

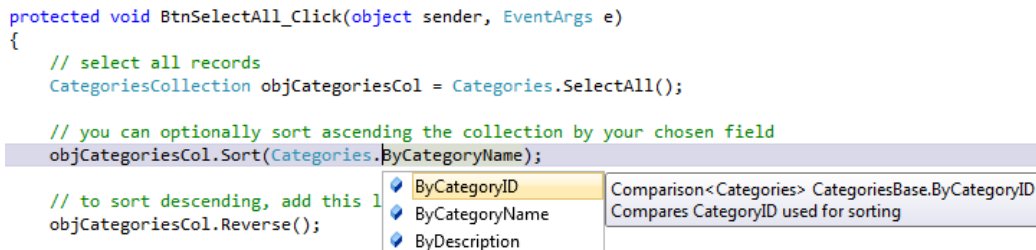


Figure 24 Sort By Property Name

29. Lastly, let's add more code to the *Default.aspx* code behind. Go back to the *CategoriesExample* class, and then copy the code inside the *Delete()* method to the *BtnDelete_Click* button event of the *Default.aspx* code behind file. See code below.

In C#

```
protected void BtnDelete_Click(object sender, EventArgs e)
{
    // delete a record by primary key
    Categories.Delete(150);
}
```

In VB.NET

```
Protected Sub BtnDelete_Click(sender As Object, e As System.EventArgs) Handles BtnDelete.Click
    ' delete a record by primary key
    Categories.Delete(149)
End Sub
```

30. Modify the code in the *BtnDelete_Click* button event. First let's show that the Category ID = 150 (or whatever record id you're trying to delete) record is still in the database by getting the category by primary key. Then we will delete the record and show that the record has been deleted. See the modified code below.

In C#

```
protected void BtnDelete_Click(object sender, EventArgs e)
{
    StringBuilder sb = new StringBuilder();
    int categoryID = 150;
    Categories objCategories = Categories.SelectByPrimaryKey(categoryID);

    if (objCategories != null)
        sb.Append("Category ID = " + categoryID + " " + " was retrieved.<br/>");
    else
        sb.Append("Category ID = " + categoryID + " not found.<br/>");

    // delete a record by primary key
    Categories.Delete(categoryID);

    // now check if category id still exist
    objCategories = Categories.SelectByPrimaryKey(categoryID);

    if (objCategories != null)
        sb.Append("Category ID = " + categoryID + " " + " was retrieved.<br/>");
    else
        sb.Append("Category ID = " + categoryID + " not found.<br/>");

    LitContent.Text = sb.ToString();
}
```

In VB.NET

```
Protected Sub BtnDelete_Click(sender As Object, e As System.EventArgs) Handles BtnDelete.Click
    Dim sb As New StringBuilder()
    Dim categoryID As Integer = 150
    Dim objCategories As Categories = Categories.SelectByPrimaryKey(categoryID)

    If objCategories IsNot Nothing Then
        sb.Append("Category ID = " & categoryID + " " & " was retrieved.<br/>")
    Else
        sb.Append("Category ID = " & categoryID & " not found.<br/>")
    End If

    ' delete a record by primary key
    Categories.Delete(categoryID)

    ' now check if category id still exist
    objCategories = Categories.SelectByPrimaryKey(categoryID)

    If objCategories IsNot Nothing Then
        sb.Append("Category ID = " & categoryID & " " + " was retrieved.<br/>")
    Else
        sb.Append("Category ID = " & categoryID & " not found.<br/>")
    End If

    LitContent.Text = sb.ToString()
End Sub
```

31. Run the website by clicking *F5*. Click the *Delete a Category* button. You should see Figure 25.

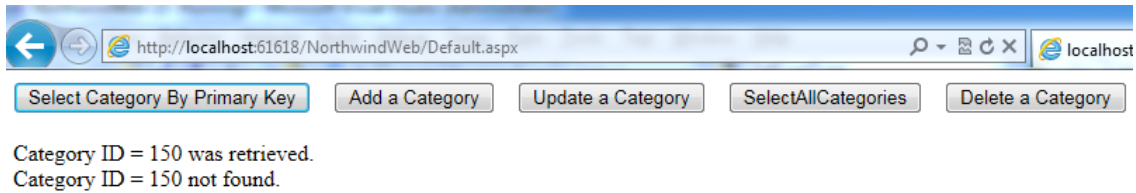


Figure 25 Delete a Category

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

33. **Note:** Don't close the *NorthwindWeb* web site, we will use it on the next tutorials.

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 all the code/folders in the *App_Code* folder of 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. Also delete the *Default.aspx* file.
2. Open AspxCodeGen 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 26.

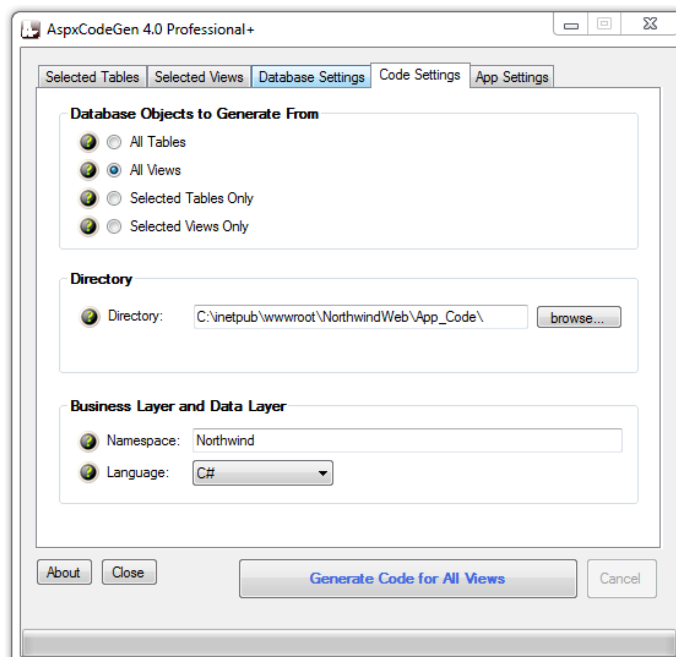


Figure 26 Code Settings Tab – All Views

4. We will keep all the settings under the *Database Settings* tab, and then Click the *Generate Code for All Views* button, AspxCodeGen will start generating code. See Figure 27.

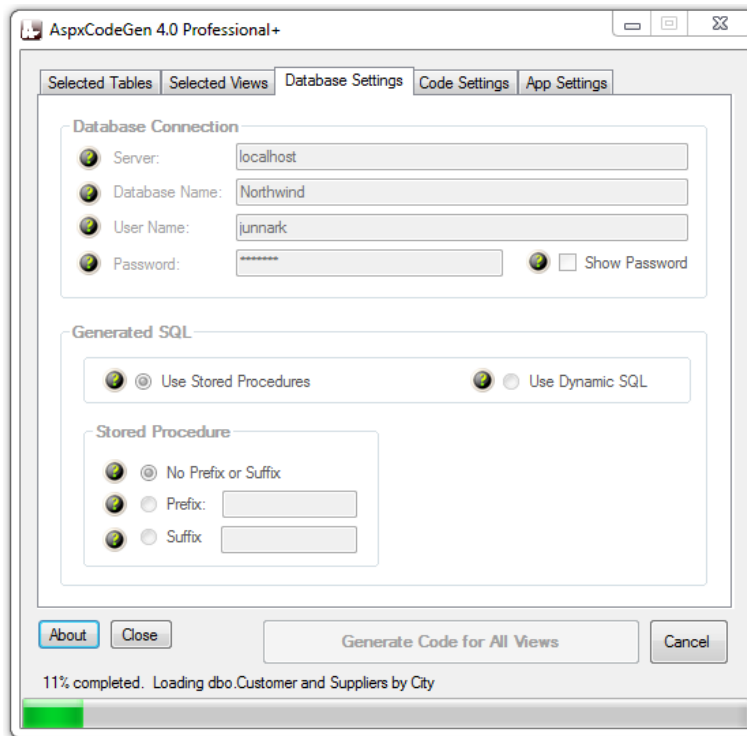


Figure 27 Generate Code For All Views

5. When done generating code, a message box is shown. Click *OK*, and then close AspxCodeGen. See Figure 28.

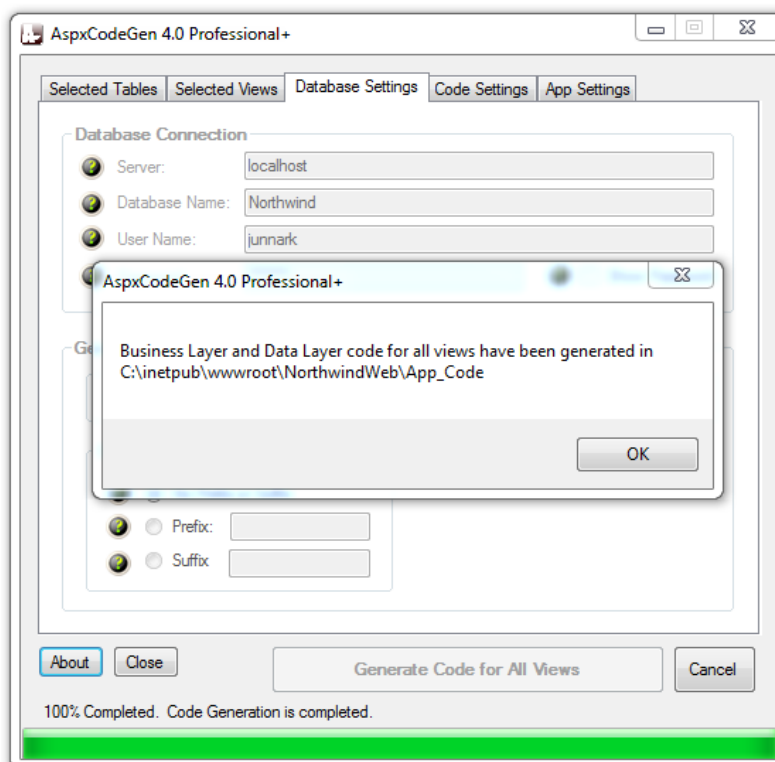


Figure 28 Done Generating Code For All Views

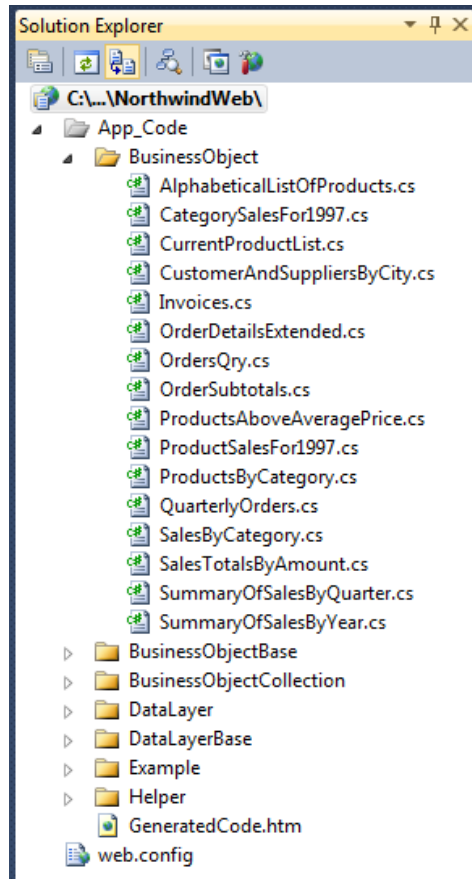


Figure 29 Generated Code For All Views

6. Let's pause for a moment and look at the generated objects under the *Solution Explorer*. You will notice that the names of the class resemble the names of all the views in the *Northwind* database. Naturally, classes for tables were not generated. See Figure 29.
7. When you double click on one of the *BusinessObjectBase* class you will notice that the only method showing here is the *SelectAll* method/function. This is because views are read-only. There will be no *SelectByPrimaryKey* or *Insert*, etc., just *SelectAll*. See Figure 30.

```

/// <summary> ...
public AlphabeticalListOfProductsBase() ...

/// <summary> ...
public static AlphabeticalListOfProductsCollection SelectAll() ...

/// <summary> ...
public static AlphabeticalListOfProductsCollection SelectAll(string sortExpression) ...

```

Figure 30 Generated Methods/Functions For All Views

8. 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 AspxCodeGen 4.0, this is the only type of stored procedures (or dynamic SQL methods) that is generated. See Figure 31.

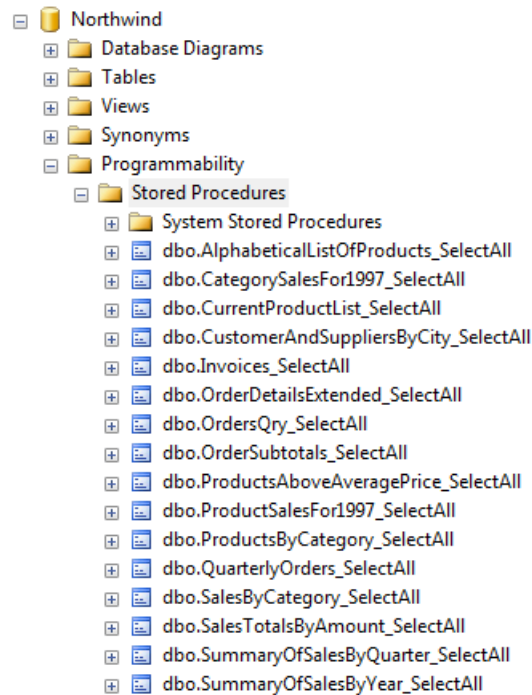


Figure 31 Stored Procedures for All Views

9. End of tutorial.

For Selected Tables Only

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

- To follow this tutorial make sure to delete all the code/folders in the *App_Code* folder of 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. Also delete the *Default.aspx* file.
- Open AspxCodeGen 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.
- 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 32. 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.
- The *Load Table* button is now enabled. See Figure 33.
- Click the *Load Table* button, and then select the following tables as shown in Figure 34.
- 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 35.

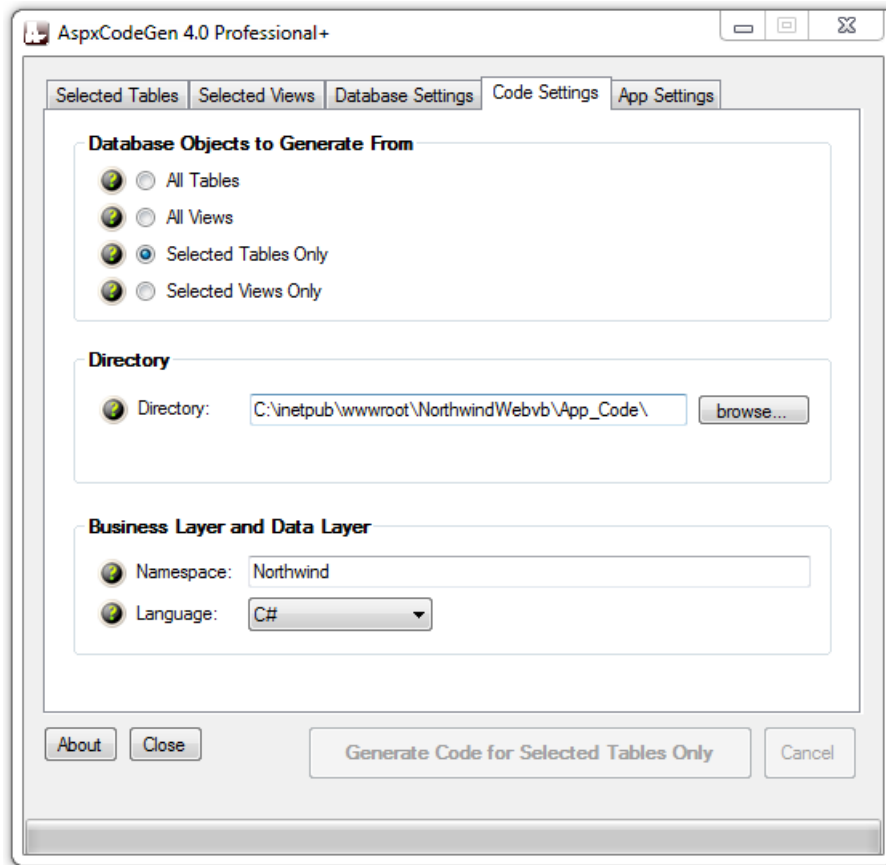


Figure 32 Code Settings Tab – Selected Tables Only

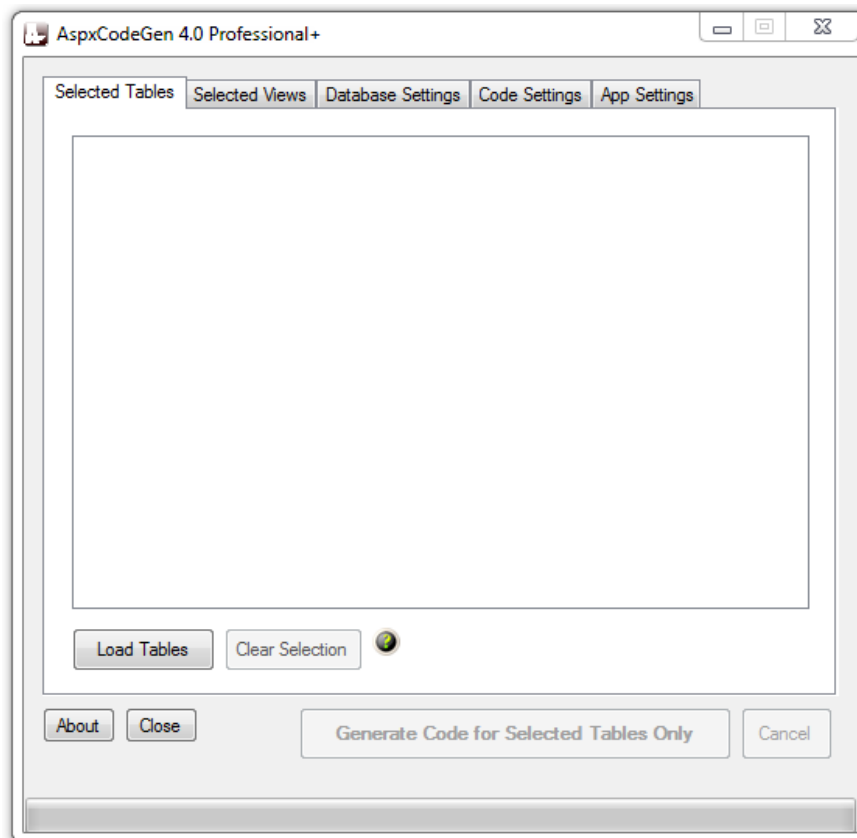


Figure 33 Selected Tables Tab

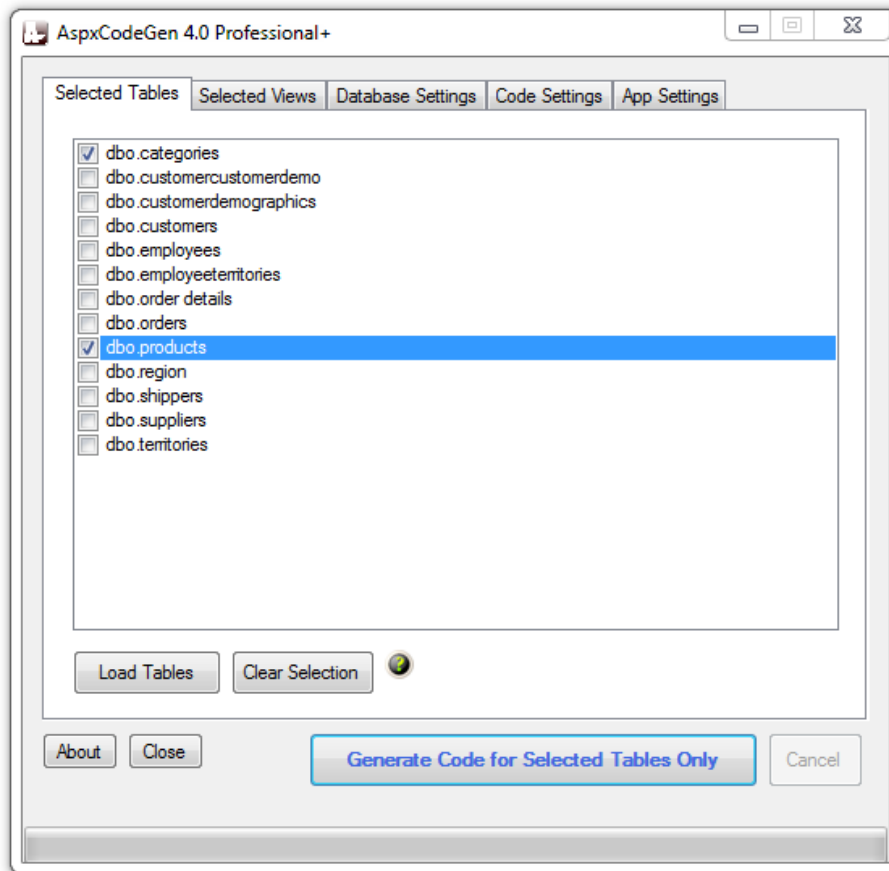


Figure 34 Load Tables, Select Tables

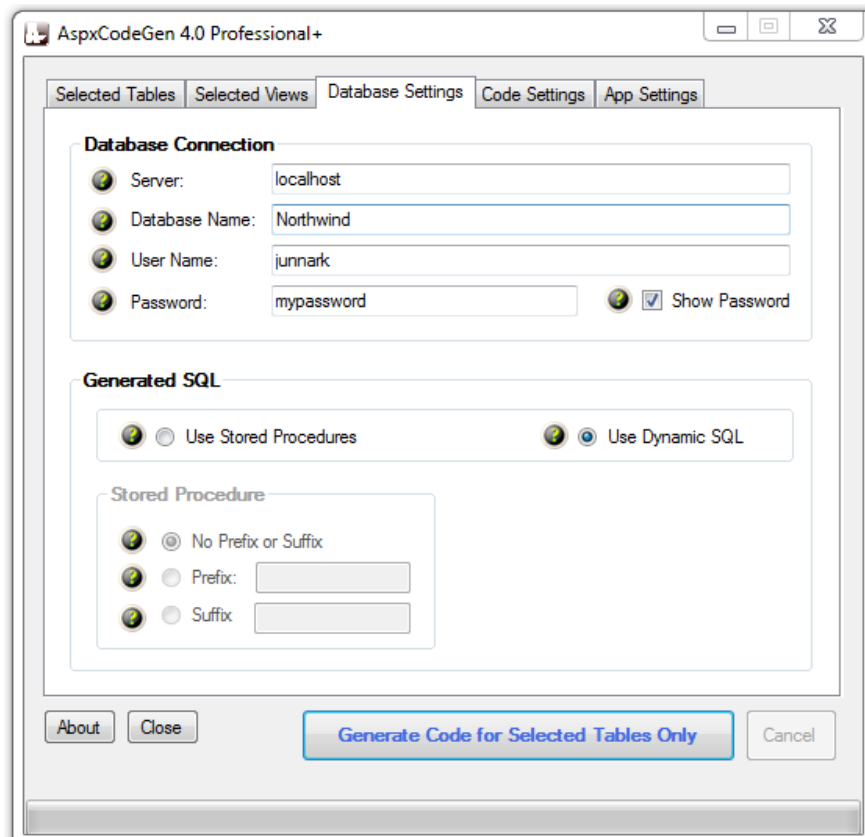


Figure 35 Generated SQL - Use Dynamic SQL

6. Click the *Generate Code for Selected Tables Only* button, AspxCodeGen will start generating code. See Figure 36.
7. When done generating code, a message box is shown. Click OK, and then close AspxCodeGen. See Figure 37.
8. Go back to the *NorthwindWeb* web site. Under the *App_Code* folder a SQL folder was added. Code located here are the dynamic SQL classes. And when you look in the *MS SQL Server Management Studio* under the *Stored Procedures* node, no stored procedure was generated. See Figure 38.
9. When you double click any of the class, you will notice that the methods returns SQL code as String.

In C#

```
using System;
using System.Text;

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

        public static string SelectByPrimaryKey()
        {
            string selectStatement = GetSelectStatement();
            StringBuilder sb = new StringBuilder();

            sb.Append(selectStatement);
            sb.Append(" WHERE ");
            sb.Append("[CategoryID] = @categoryID ");

            return sb.ToString();
        }

        public static string SelectAll()
        {
            string selectStatement = GetSelectStatement();
            StringBuilder sb = new StringBuilder();

            sb.Append(selectStatement);

            return sb.ToString();
        }
    }
}
```

Code removed for cleanness...

In VB.NET

```
Imports System
Imports System.Text

Namespace Northwind.DataLayer.Base
    Public NotInheritable Class CategoriesSql
        Private Sub New()
            End Sub

        Public Shared Function SelectByPrimaryKey() As String
            Dim selectStatement As String = GetSelectStatement()
            Dim sb As New StringBuilder()

            sb.Append(selectStatement)
            sb.Append(" WHERE ")
            sb.Append("[CategoryID] = @categoryID ")
        End Function
    End Class
End Namespace
```

```

Return sb.ToString()
End Function

Public Shared Function SelectAll() As String
Dim selectStatement As String = GetSelectStatement()
Dim sb As New StringBuilder()

sb.Append(selectStatement)

Return sb.ToString()
End Function

```

Code removed for cleanness...

10. Also notice that only 2 classes were generated, just like we specified in AspxCodeGen 4.0. See Figure 38
11. 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 were generated. See Figure 38.

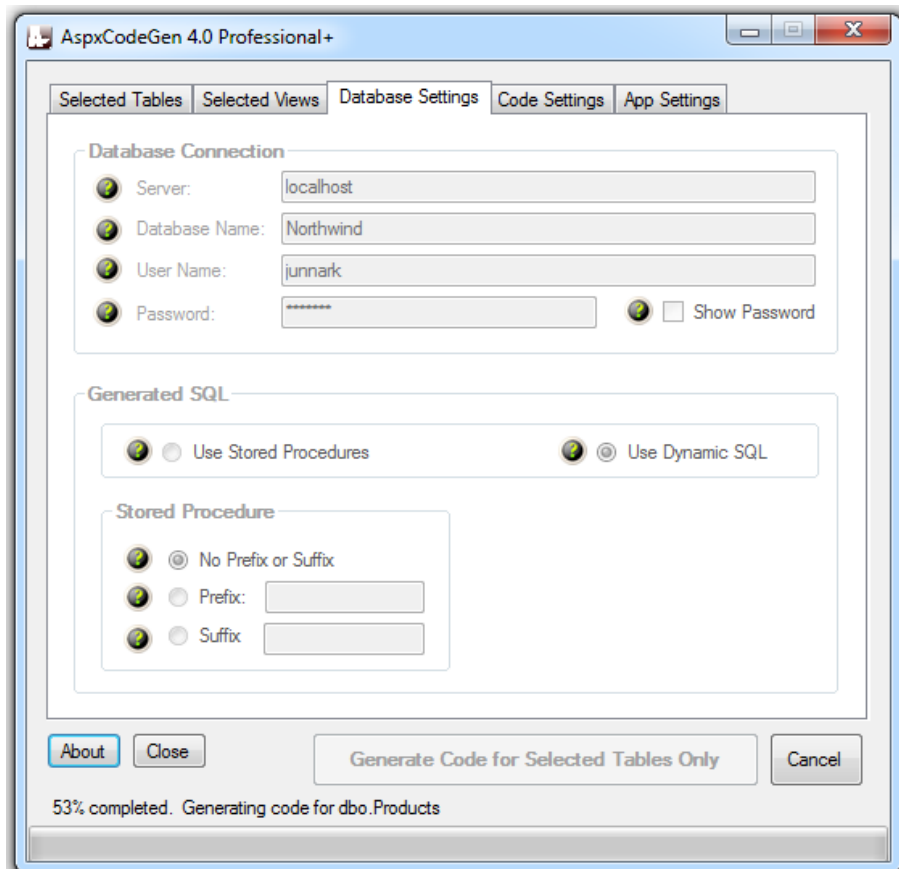


Figure 36 Generate Code for Selected Tables Only

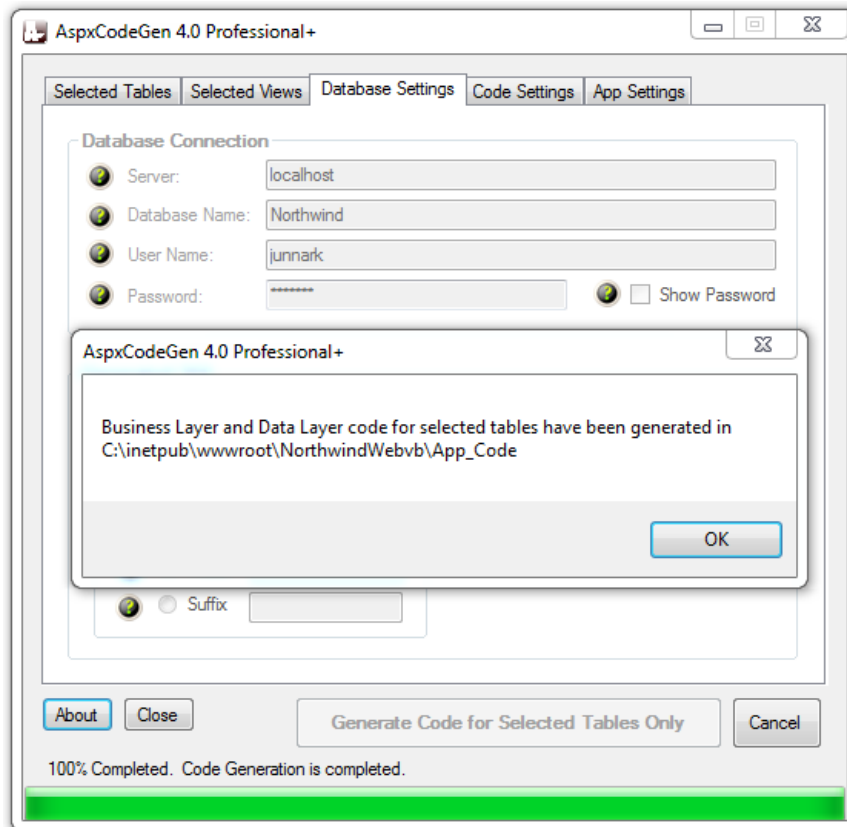


Figure 37 Done Generating Code for Selected Tables Only

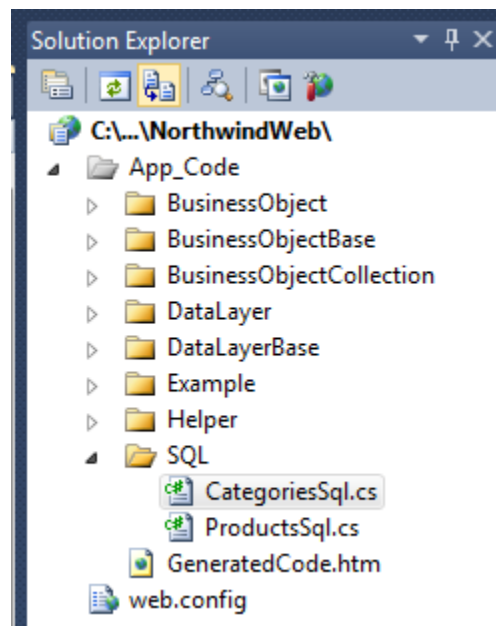


Figure 38 Generated Code for Selected Tables Only

12. End of tutorial.

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 all the code/folders in the *App_Code* folder of 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. Also delete the *Default.aspx* file.
2. Open AspCodeGen 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 39. 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 39. Now go back to the *Selected Views* tab.

4. The *Load Views* button is now enabled. See Figure 40.
5. Click the *Load Views* button, and then select the following views as shown in Figure 41.
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 42.

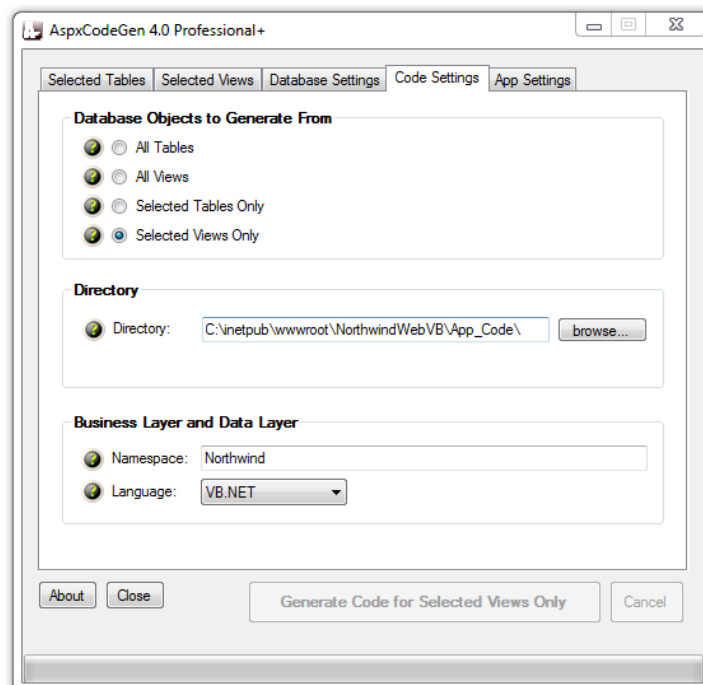


Figure 39 Code Settings Tab – Selected Views Only

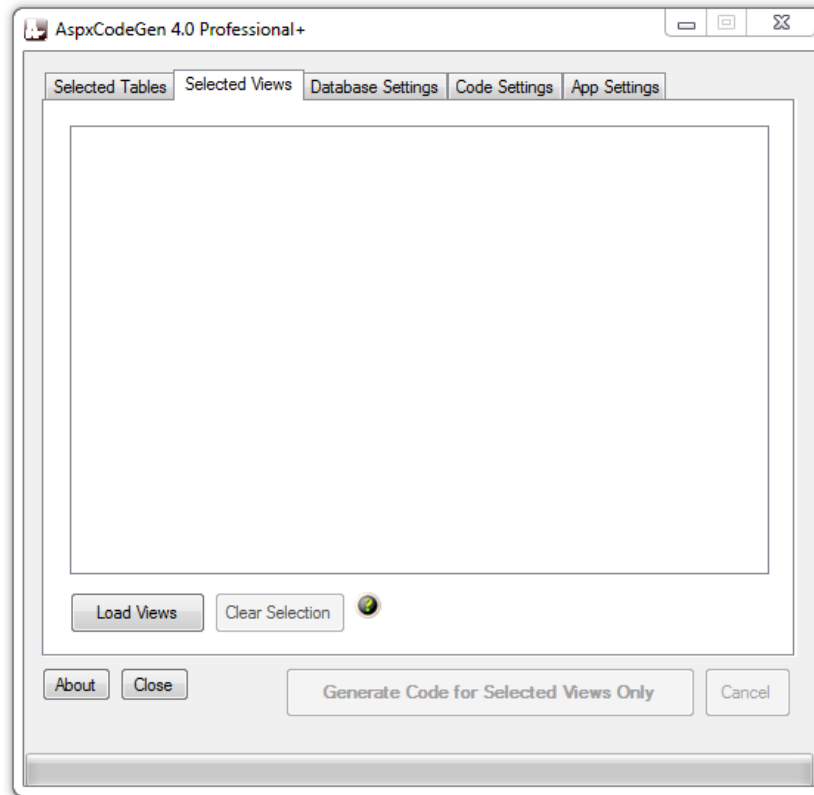


Figure 40 Selected Views Tab

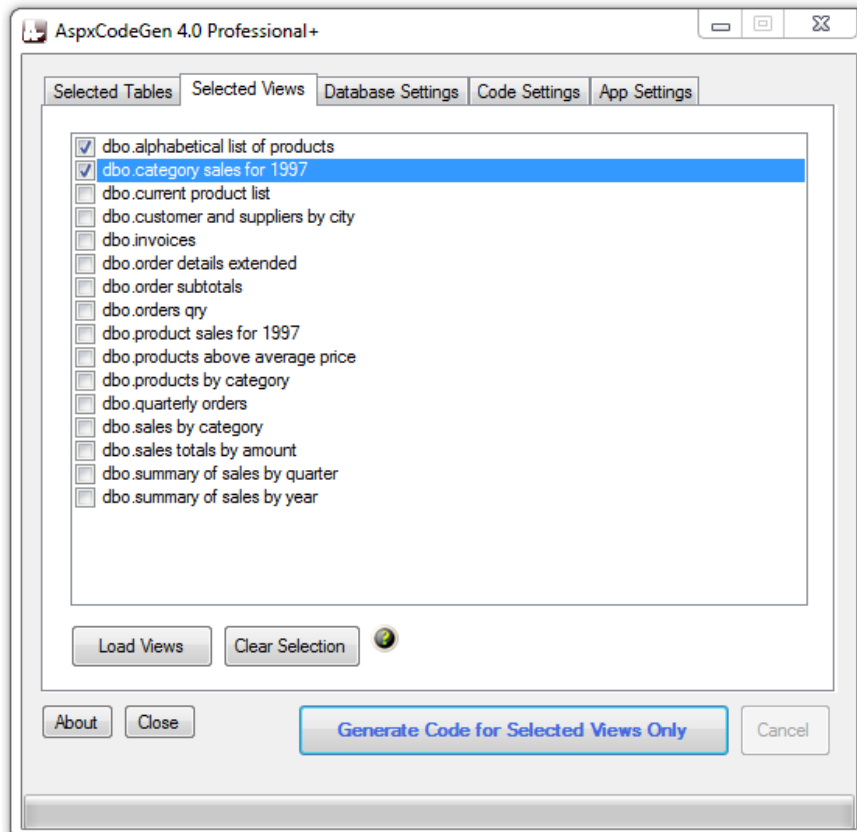


Figure 41 Load Views, Select Views

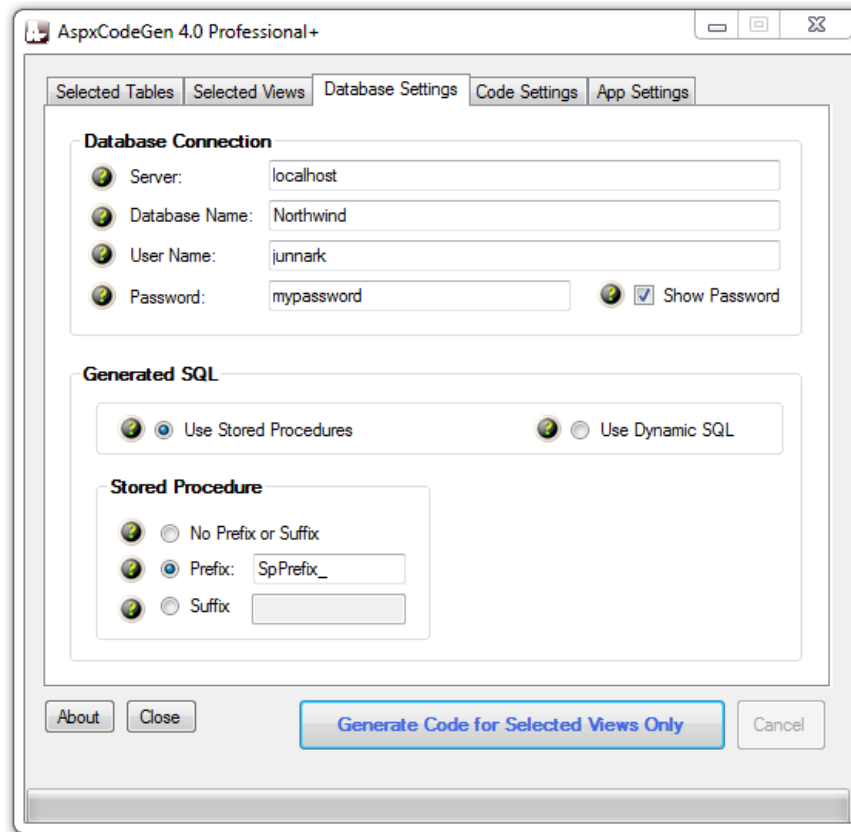


Figure 42 Database Settings – Stored Procedures with Prefix

7. Click the *Generate Code for Selected Views Only* button, AspxCodeGen will start generating code.
8. When done generating code, a message box is shown. Click OK, and then close AspxCodeGen. See Figure 43.

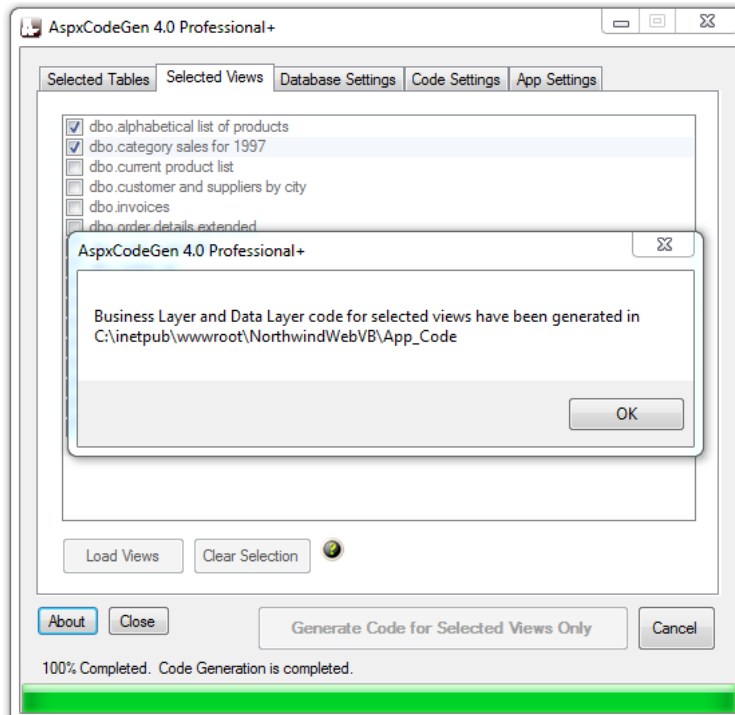


Figure 43 Done Generating Code For Selected Views Only

- 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 AspxCodeGen 4.0. See Figure 44.

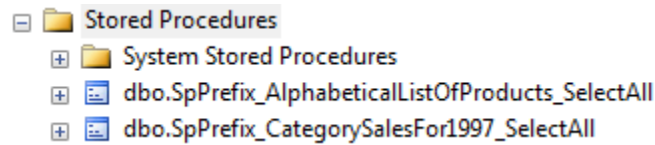


Figure 44 Generated Stored Procedures with Prefix

- Go back to the *NorthwindWeb* web site, under the *App_Code* folder, you will notice that only 2 classes were generated. These are the views we selected during code generation. See Figure 45

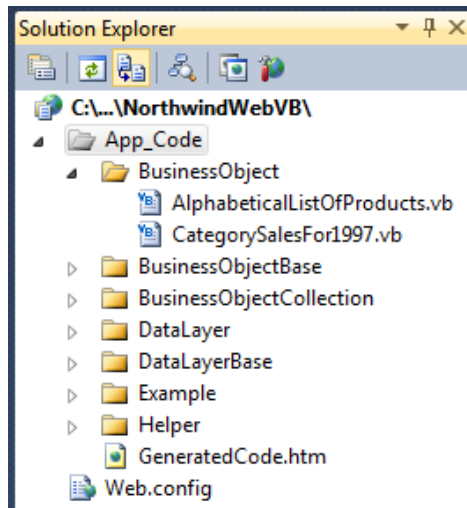


Figure 45 Selected Views

- End of tutorial.

Generated Code

AspxCodeGen 4.0 generates 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.

AspxCodeGen 4.0 can generate from Tables or Views as source. This portion will discuss the parts of the generated code.

AspxCodeGen 4.0 generates a 2-tier structure code.

1. Business Objects (Middle Layer) – Middle Tier Classes.
2. Data Layer – Data Tier Classes, Stored Procedures or Dynamic SQL Classes.

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, simply put these generated objects into a Class Library project and then reference the project from your 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 AspxCodeGen 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 SQL¹
- 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 Classes¹

The AspxCodeGen engine (integrated in AspxCodeGen 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 AspxCodeGen 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

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

Miscellaneous Files

1. **GeneratedCode.htm:** List of all the generated middle-tier, data-tier, stored procedures¹ or dynamic SQL¹, example classes, and helper classes.

Adding Your Own Code

Yes you can add your own code to the objects generated by AspxCodeGen 4.0. **But we warned: Almost all generated files are overwritten by AspxCodeGen 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.

Helper Files

AspxCodeGen 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. Dbase File¹

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
    End Class
End Namespace
```

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 AspxCodeGen 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 AspxCodeGen 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.

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

- 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.

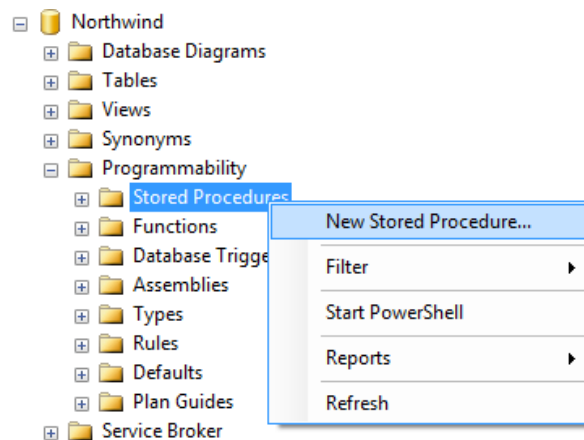


Figure 50 Add New Stored Procedure

- 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
```

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

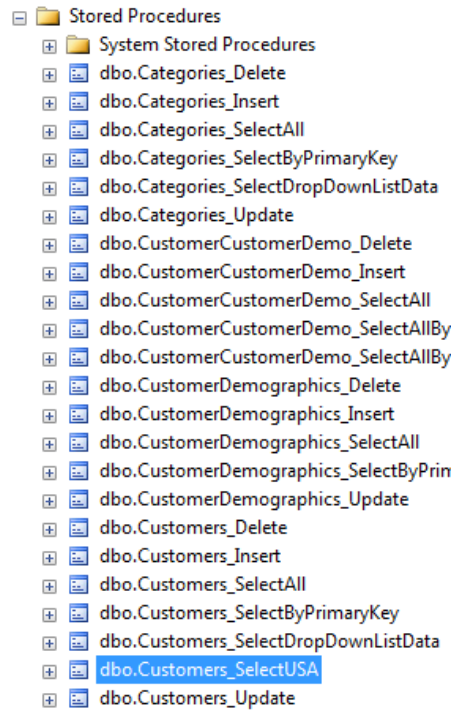


Figure 51 Newly Added Stored Procedure

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 *SelectUSAOnly* 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
    End Function

```

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 AspxCodeGen 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 *NorthwindAPI*. See Figures 52 and 53.

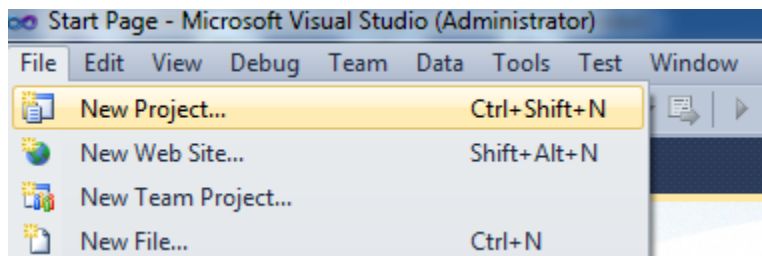


Figure 52 Create a New Project

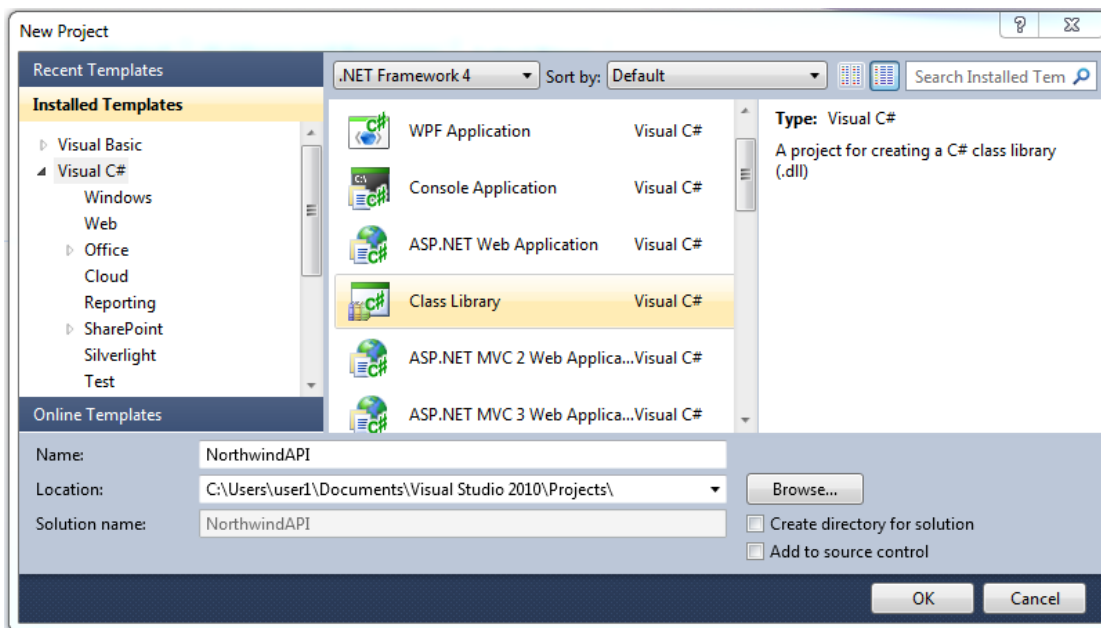


Figure 53 Class Library

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

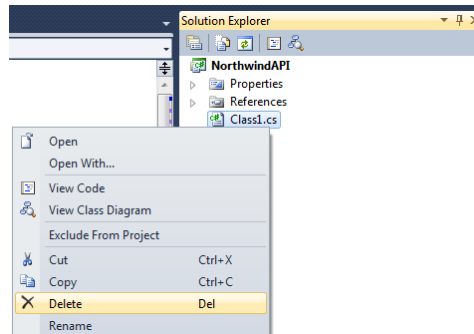


Figure 54 Delete Class1

- 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.

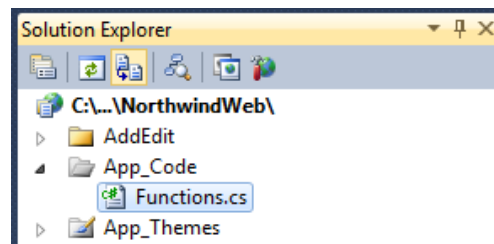


Figure 55 NortwindWeb Web Site – App_Code Directory

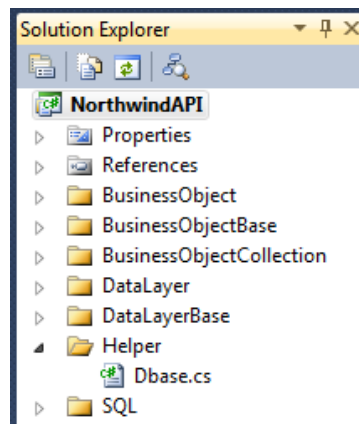


Figure 56 NorthwindAPI Class Library Project

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

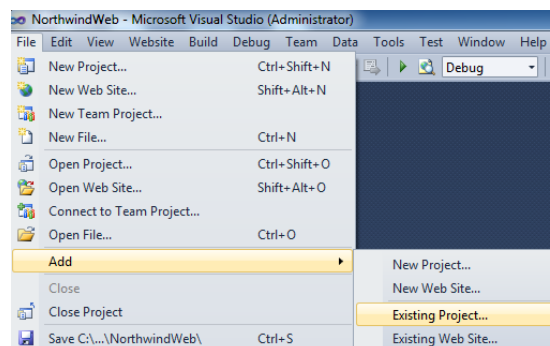


Figure 57 Add Existing Project

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

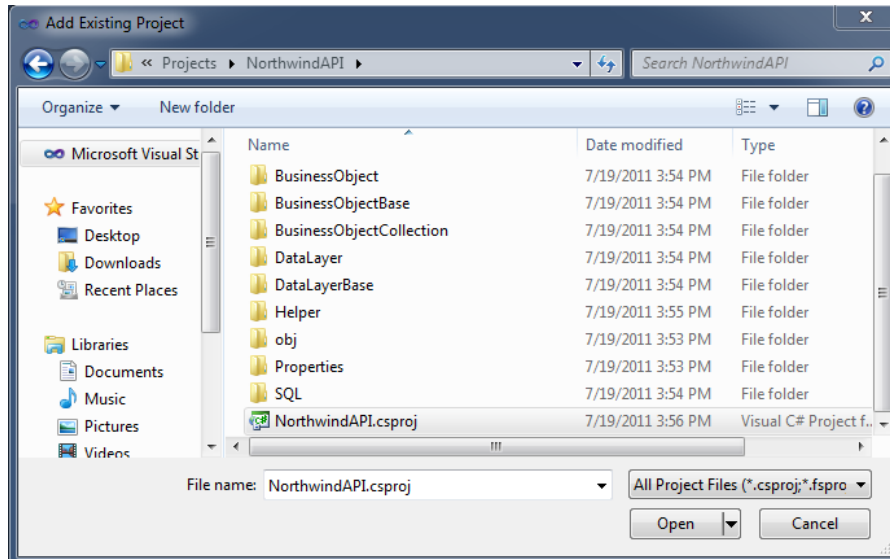


Figure 58 Add Existing Project Dialog

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

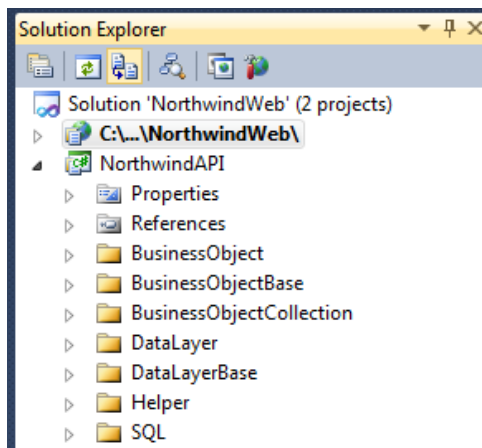


Figure 58 2 Projects – NorthwindWeb and NorthwindAPI

7. If you used AspxCodeGen 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.
- `using System.Web.Script.Serialization;` (C#)
 - `[ScriptIgnore]` (C#)
 - `Imports System.Web.Script.Serialization` (VB.NET)
 - `<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.

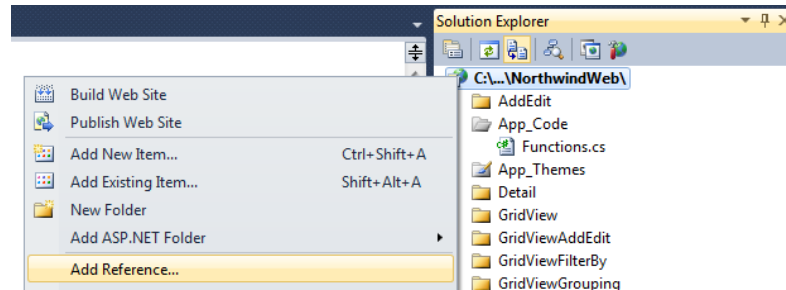


Figure 59 Add a Reference

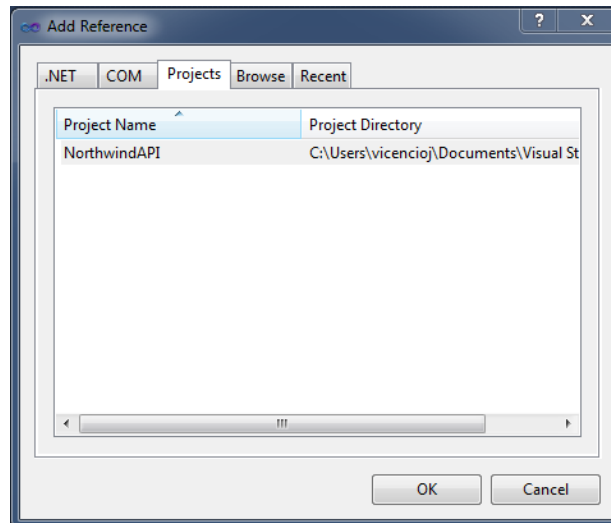


Figure 60 Add a Reference Dialog

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

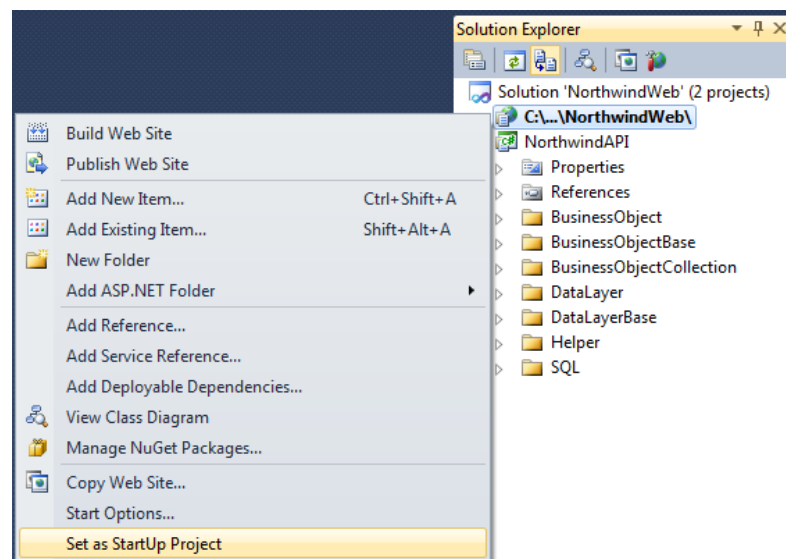


Figure 60 Add a Reference Dialog

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

11. End of tutorial

Example Classes

AspxCodeGen 4.0 is so easy to use, we even generated example code for you, and 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()
```

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, HierarchyId, 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 AspxCodeGen must have admin or enough privileges 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. *MS SQL Server 2008* is a product of Microsoft. Please see more info here:
<http://www.microsoft.com/sqlserver/2008/en/us/default.aspx>
4. *Visual Studio 2010* is a product of Microsoft. Please see more info here:
<http://www.microsoft.com/visualstudio/en-us/products/2010-editions>
5. *Northwind* Database is a product of Microsoft. Please see more info here:
<http://www.microsoft.com/download/en/details.aspx?id=23654>