Chapter 5

Creating a WPF Application

If you are going to build a WPF application you can use our Application Builder utility to create a new WPF project that already has the PDSA Providers and Libraries hooked up. You can then use Haystack to generate more add, edit, delete and search user controls to add to this project. Using our Application Builder utility and our template project is much quicker than you starting from scratch. In this chapter you will be guided step-by-step through this process.

Windows Presentation Foundation Application

To start this process you will run the PDSA Framework Utilities. From the Windows Start menu, select **PDSA Framework 5** and then select **PDSA Framework Utilities** from the menu. You can then click on the “Application Builder” button (Figure 1).

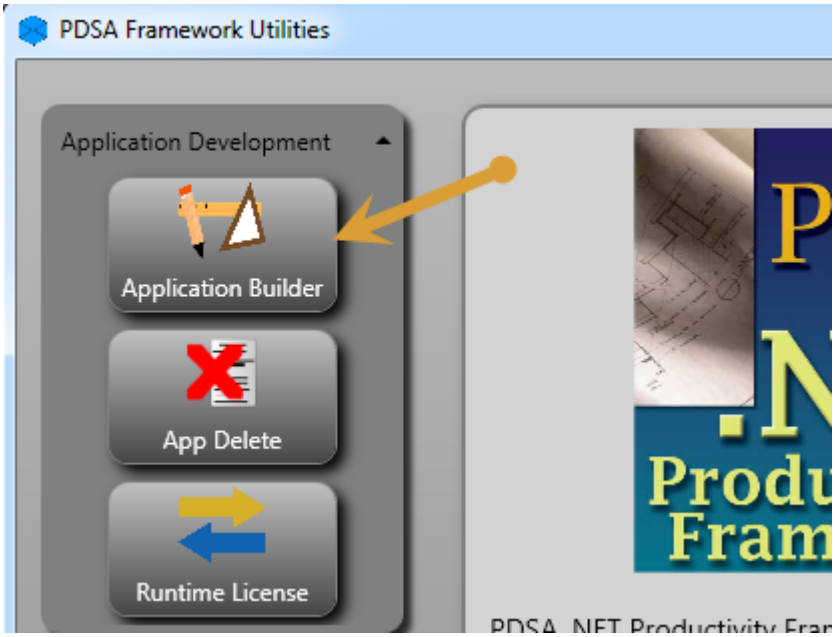

Figure 1: Application Builder Button

Step 1: Project Type

On the first tab (Figure 2) you will select the type of application that you will be building and the language you wish to use. Based on the type of application a specific project template will be selected. For this chapter, click on any of the **WPF…** templates

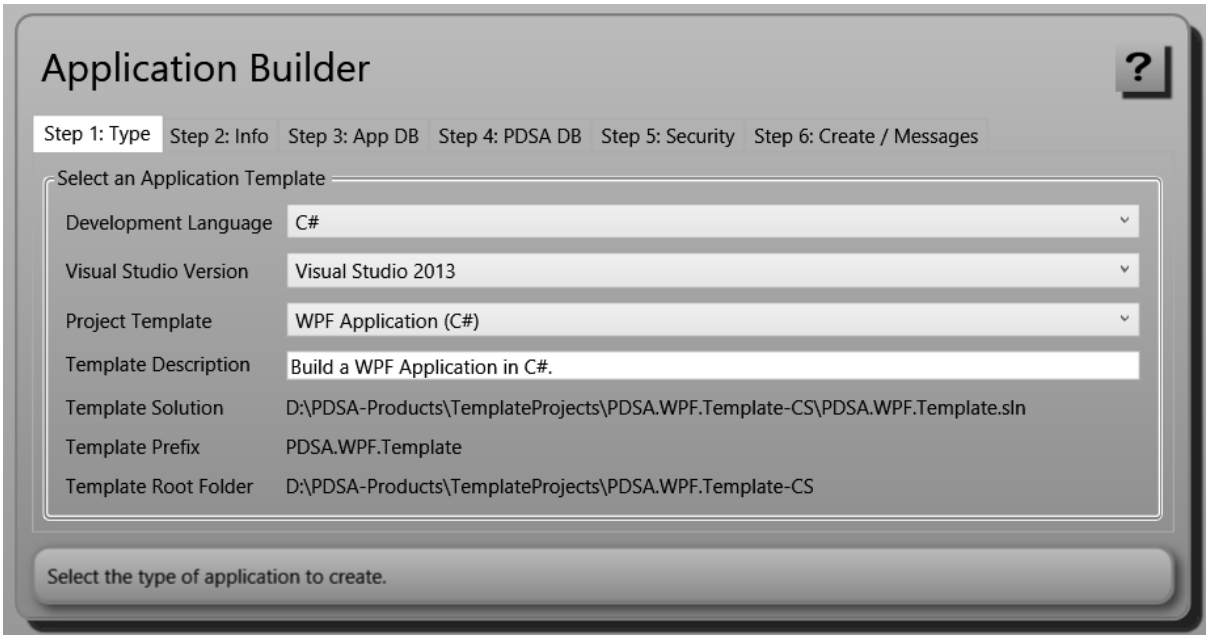

Figure 2: Select a Project Type

Step 2: Project Info

In Step 2 (Figure 3) you will type in the name and location of where you want to put your new project. You can also fill in some additional information in this step as well.

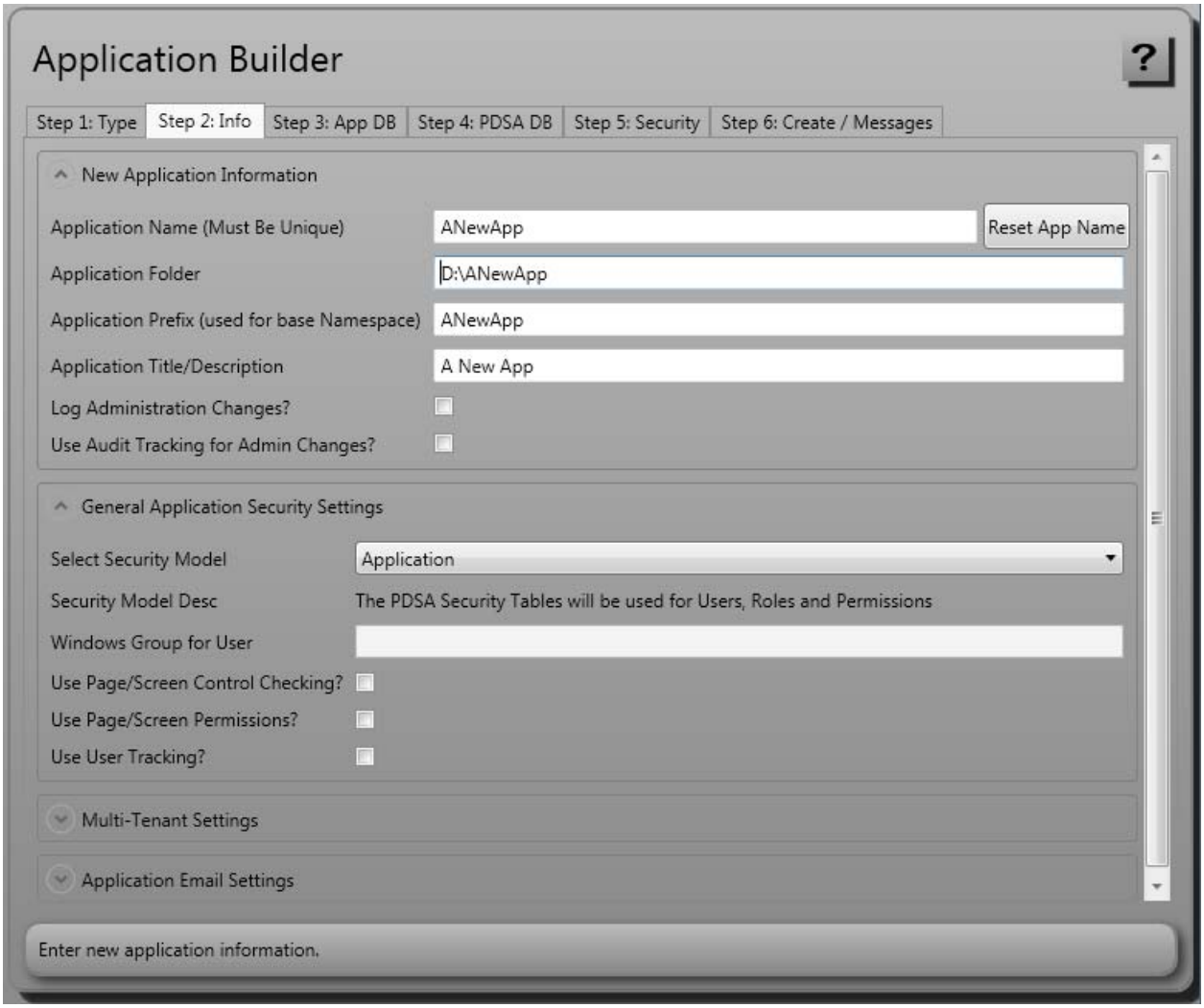

Figure 3: New Application Information

Step 3: Application Database

In Step 3 (Figure 4) you will select which database provider and connection string you wish to use for your application. This connection string will point to where you application tables are located.

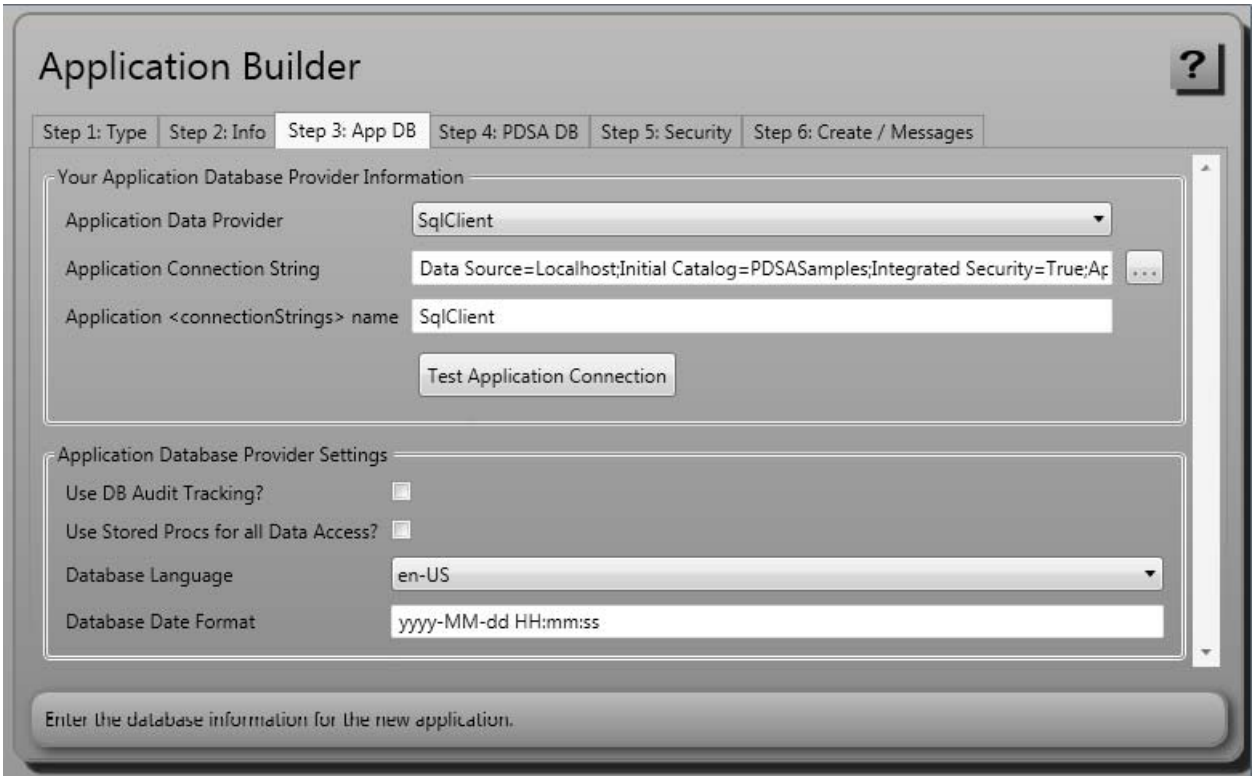

Figure 4: Application Database Information

Click on the **Test Application Connection** button to test this connection.

Step 4: PDSA Database

In Step 4 (Figure 5) you will select which database provider and connection string you wish to use to put the PDSA tables. This connection string will point to where the PDSA tables are located or where you wish to install them.

NOTE: You will need to create these PDSA Tables prior to running the Create process. You can either click on the Submit Script button if you have DBA rights, or you can open the script file and send it to a DBA to install the tables into the database you specify in this step.

Click on the **Test PDSA Database Connection** button to test this connection.

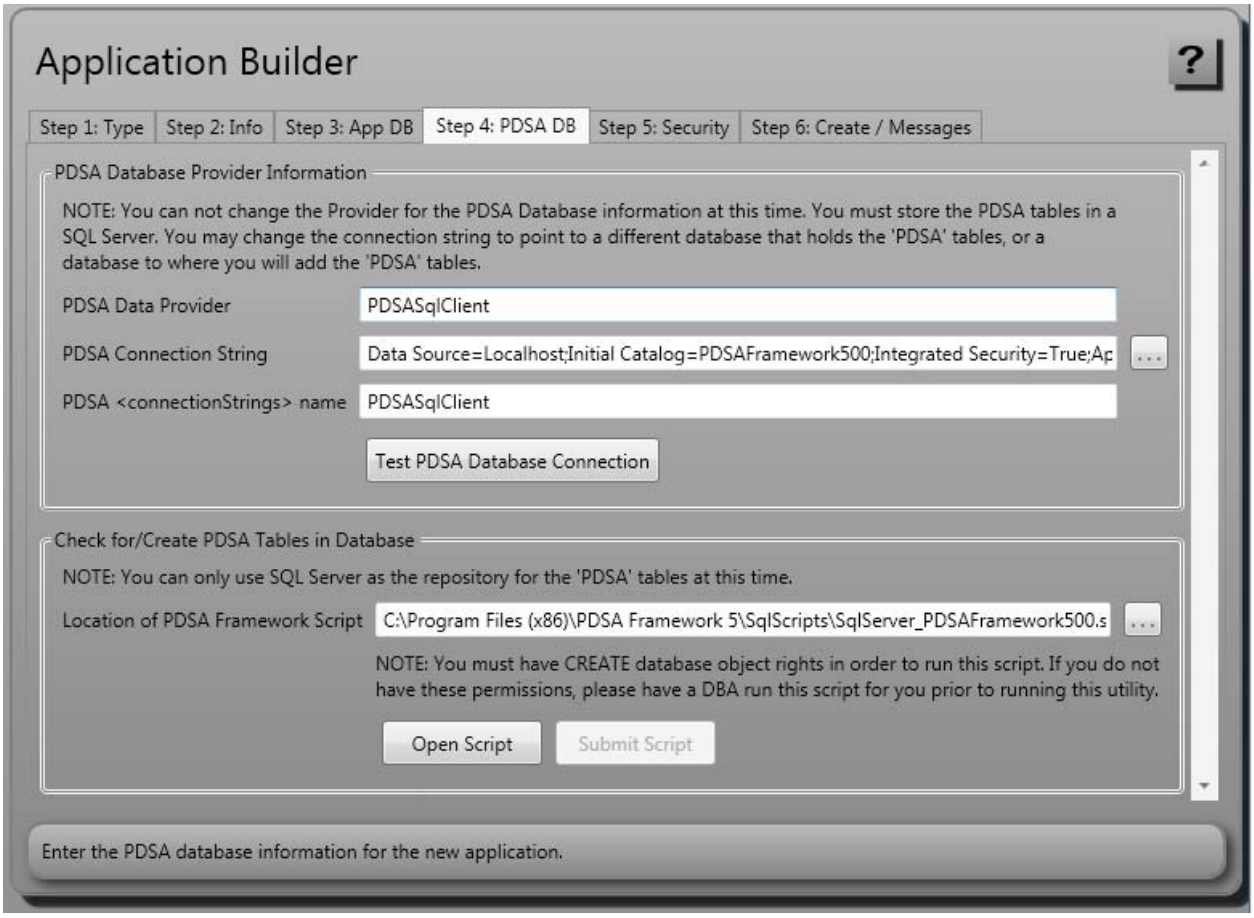

Figure 5: PDSA Database Information

Step 5: Security

If you will be using PDSA Security (not Active Directory) then you will need to provide an initial login id and password for an administrator to login to the application after it is built.

Click on the **Does User Exist? Button**.

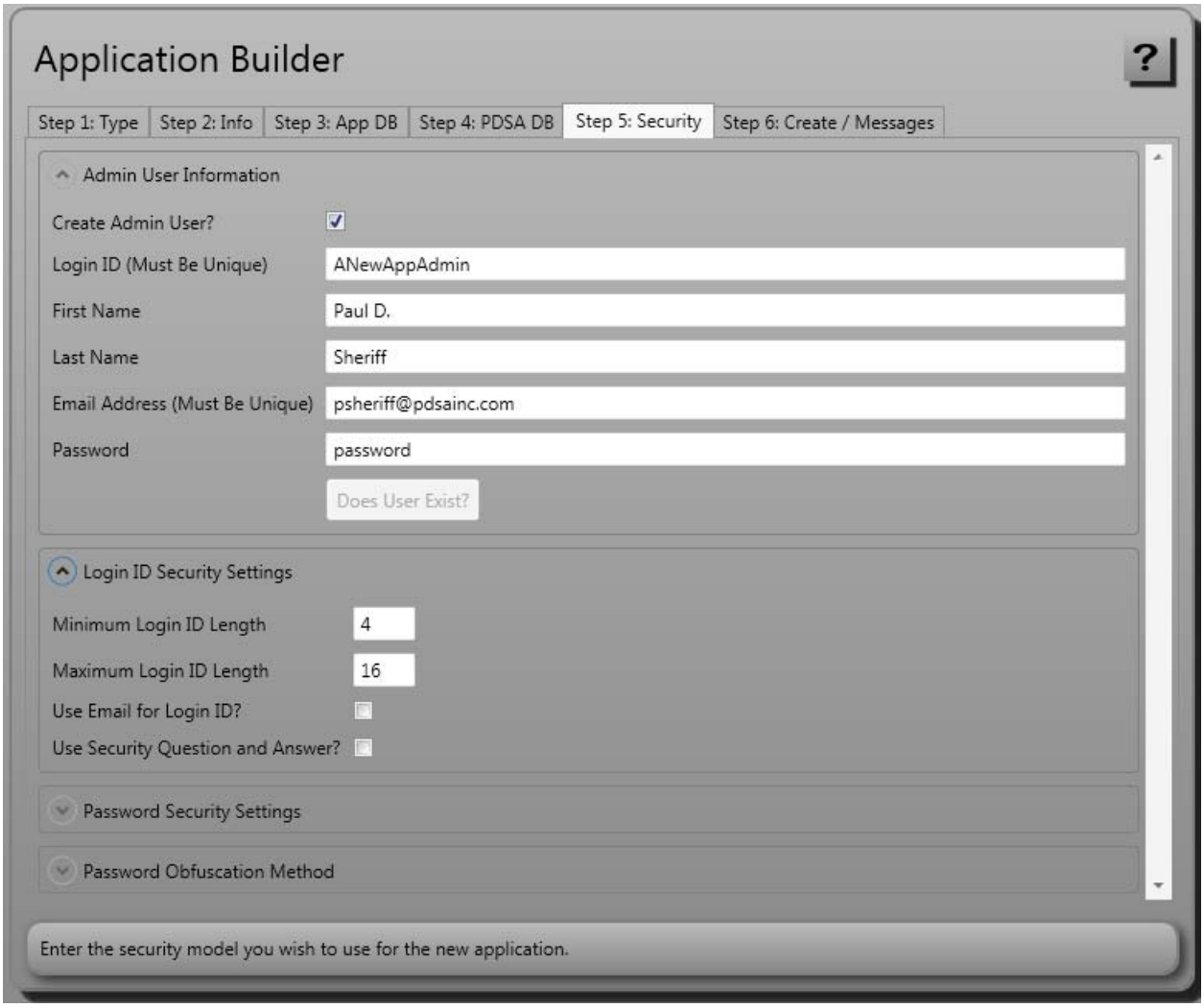

Figure 6: Security Setup

Step 6: Create the Project

Click the **Create** button (Figure 7) now and our template project will be copied to your new folder location. All of the files will be renamed to your project prefix. And you will then have a new Visual Studio 2010 project with the same name as your project prefix.

Figure 7: Create new Application

Your New Project

After the new Visual Studio solution is built it will be opened. You solution will have several projects that should looks similar to Figure 8.

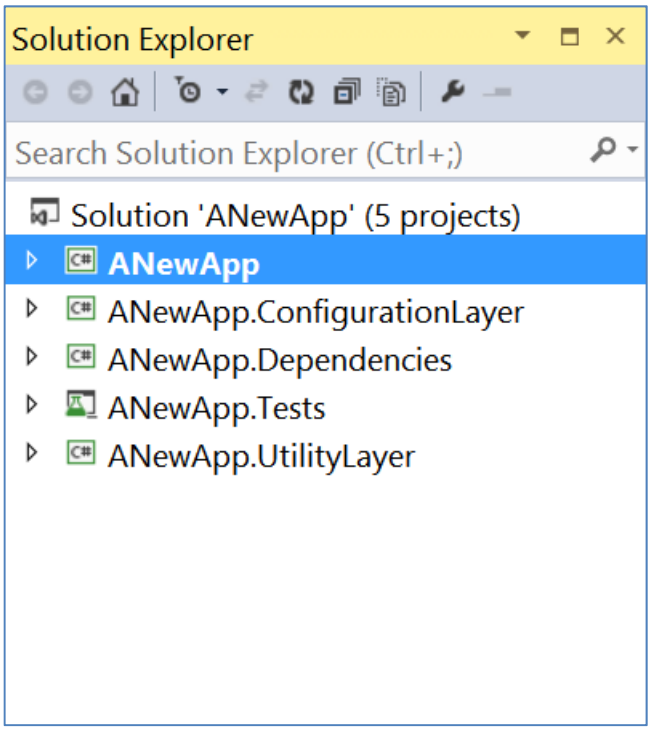

Figure 8: Your Project Solution

Here is a rundown of the different projects and what belongs in each one.

Project	Description
ANewApp	This is your main WPF application.
.ConfigurationLayer	This project has a class called AppSettings that is used to retrieve global settings from your App.Config file, an XML file, the registry, or wherever you wish to store your configuration settings.
.Dependencies	This project is not used in the project, it simply contains all of the .DLLs that are referenced by the other projects in this solution.
.Tests	This is a Unit Test project.
.UtilityLayer	This project contains classes to help you work with logging, caching, view models, etc.

You can build this application and run it immediately to test it out. This sample has a connection string in the .Config file that points to the database you used for your application tables and for the PDSA tables.

Generate Code Using Haystack

Go into Haystack and create a new project with the exact same namespace that you filled into your **Application Prefix** field in the application builder. This is very important to keep the namespace exactly the same.

Be sure to check “Is PDSA Framework 5.x Project?” as shown in Figure 9.

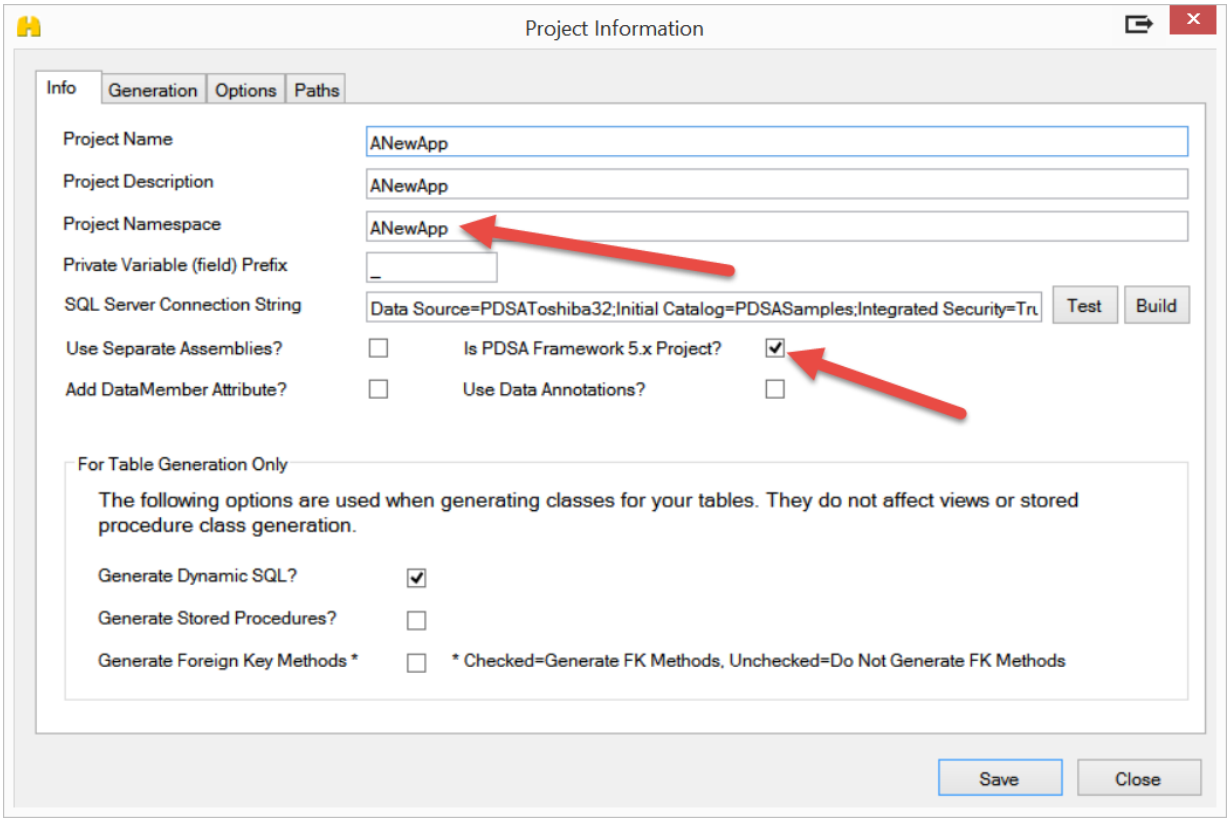

Figure 9: Project Information Screen in Haystack

On the **Generation** tab select the two **PDSA FW 5.x - MVC*** templates as shown in Figure 10.

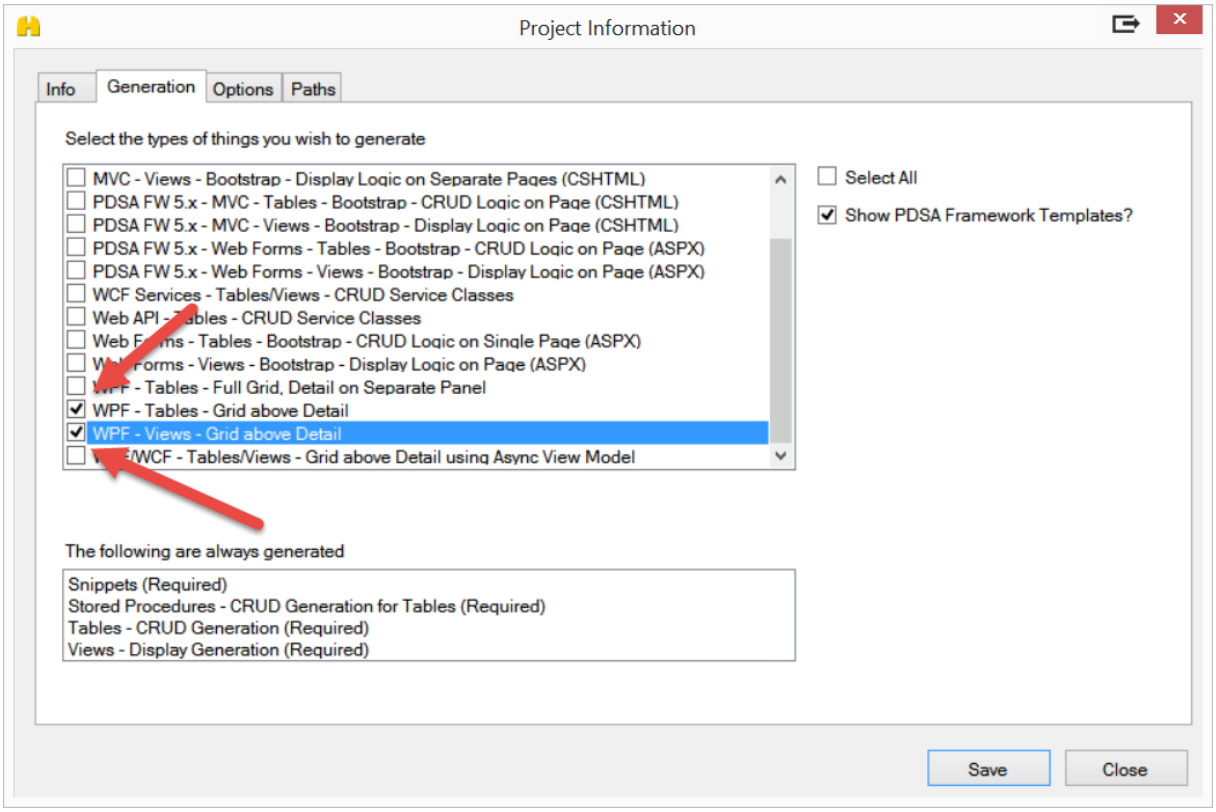

Figure 10: Select the WPF templates.

Now generate some classes for your table and then you can copy the generated code into the appropriate Project(s) in your new Solution. You will see a generation folder that looks like Figure 11.

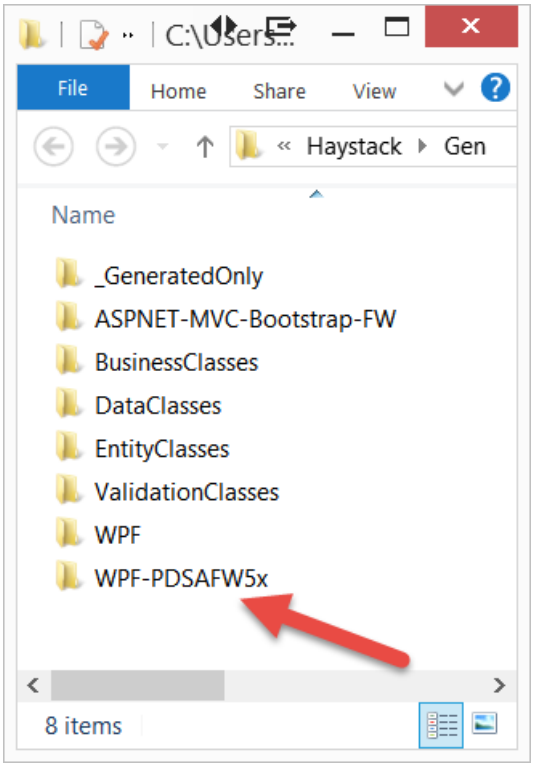

Figure 11: The files generated are in the WPF-PDSAFW5x folder.

Copy Generated Folders/Files into WPF Application

For a WPF Application you will copy folders/files from the **\Gen\WPF-PDSAFW5x** folder (Figure 12) into your solution.

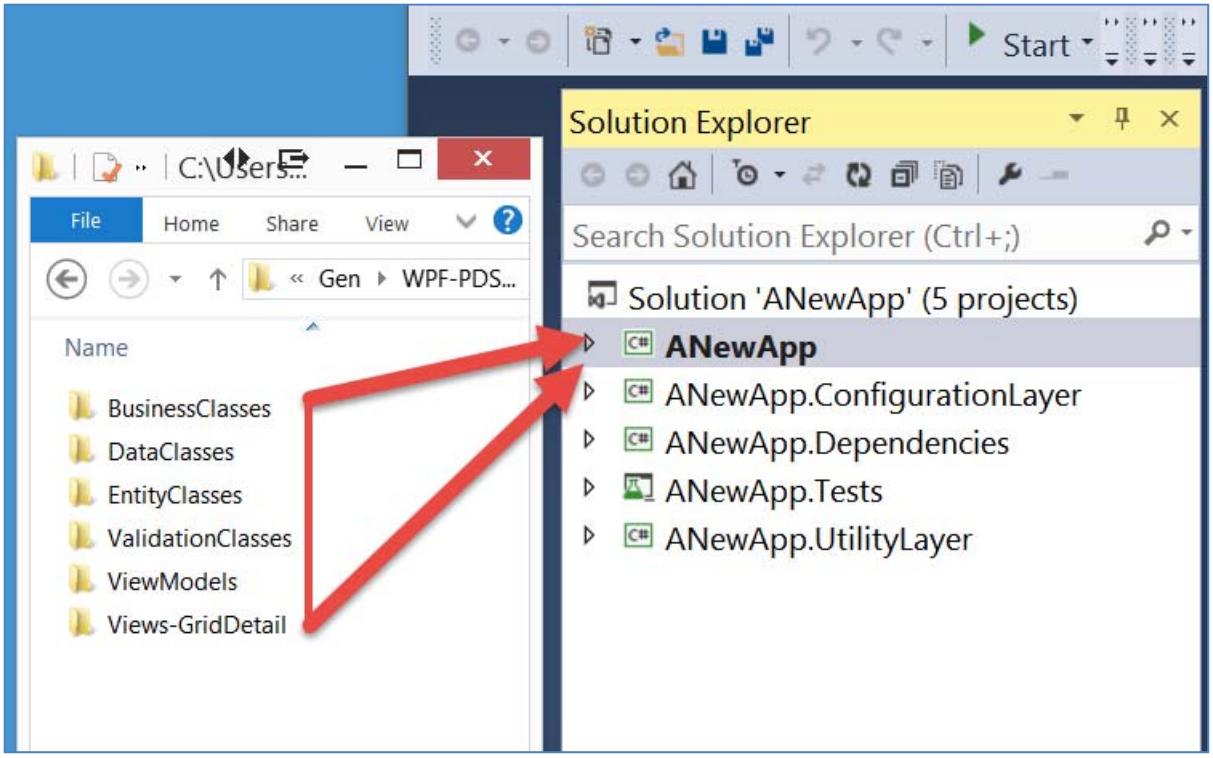

Figure 12: Copy all folders from the generated WPF folder into your WPF Project

Call Your Generated Control

After you add a Haystack-generated control to your project you will want to add a new menu button and call it from the MainWindow.xaml.

Open the MainWindow.xaml in design mode. It should look similar to Figure 13.

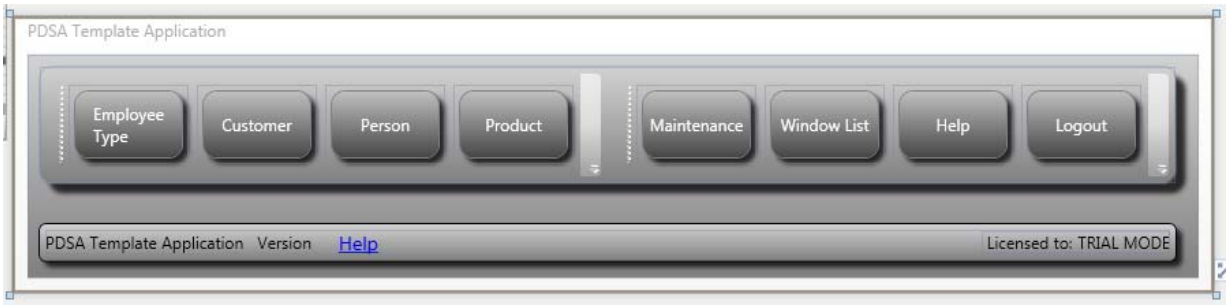

Figure 13: The MainWindow where you will add menus

Click on one of the menu items such as the Customer menu. In the XAML, copy and paste the complete button just below the Customer menu. Be sure to copy the complete <Button>…</Button> tag.

```
<Button Click="LoadUserControl_Click"
        Tag="ANewApp.ucCustomer">
  <uc:ucMenuItem MenuTitle="Customer"
                 AddToFormList="True"
                 DisplayInWindow="False"
                 HandleSpecial="True"
                 RemoveFromFormList="False"
                 MenuToolTip="Customer Information" />
</Button>
```

After pasting the button in, modify the areas shown in bold above. The following table describes what each of the above attributes are used for.

Attribute	Description
Tag	Put your complete Namespace and user control in here. Reflection is used to load this control and display it in your application.
MenuTitle	The text to display on this menu.
AddToFormList	Whether or not to add this user control to the global form tracker list.
HandleSpecial	If set to true, then you will need to write special code to do what you want when this menu item is clicked.
DisplayInWindow	Whether or not to display the user control in a Pop-Up Window.
RemoveFromFormList	Whether or not to remove this user control from the global form tracker list when a new control is loaded.
MenuToolTip	A tooltip to display when the user hovers over this menu item.

Run the Application

So, assuming you have set the connection string in the App.Config file to the same connection string you used in Haystack then you should be able to run your WPF application at this point and see your new user control appear when you click on your button.

Summary

In this chapter you learned how to use the PDSA .NET Productivity Framework Application Builder utility to generate a new WPF application. You then put in the generated classes and views from Haystack and created a complete add, edit and delete screen for a specified table.