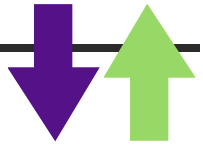
DEVELOPER GUIDE

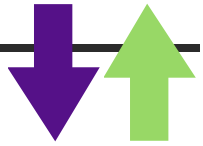
8/23/2013

Graduate Capstone

RECEIPT REWARDS

Table of Contents





1 Introduction

The purpose of this document is to provide a guide for developers as how to get started with the project. In addition, it aims to give a detailed explanation of the structure and organization of the project structure and files.

1.1 Intended Audience

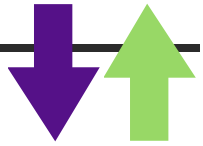
This document is intended mainly for developers who will be writing code for the project. The goal is for those individuals to learn about the organizations, conventions and purposes of each file and object and what they do in the project.

1.2 References

- http://developer-support-handbook.appspot.com/documentation.html

1.3 Revision History

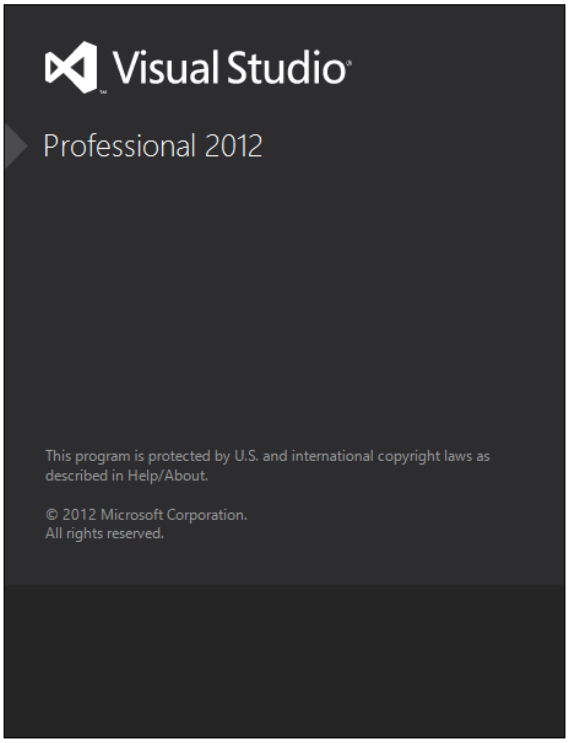
Name	Date	Reason For Change	Version
Andy Bottom	06/24/2013	Created the formatting of the document and added references	0.1
Andy Bottom	08/20/2013	Completed the developer guide and all the content.	1.0



2 Before You Start

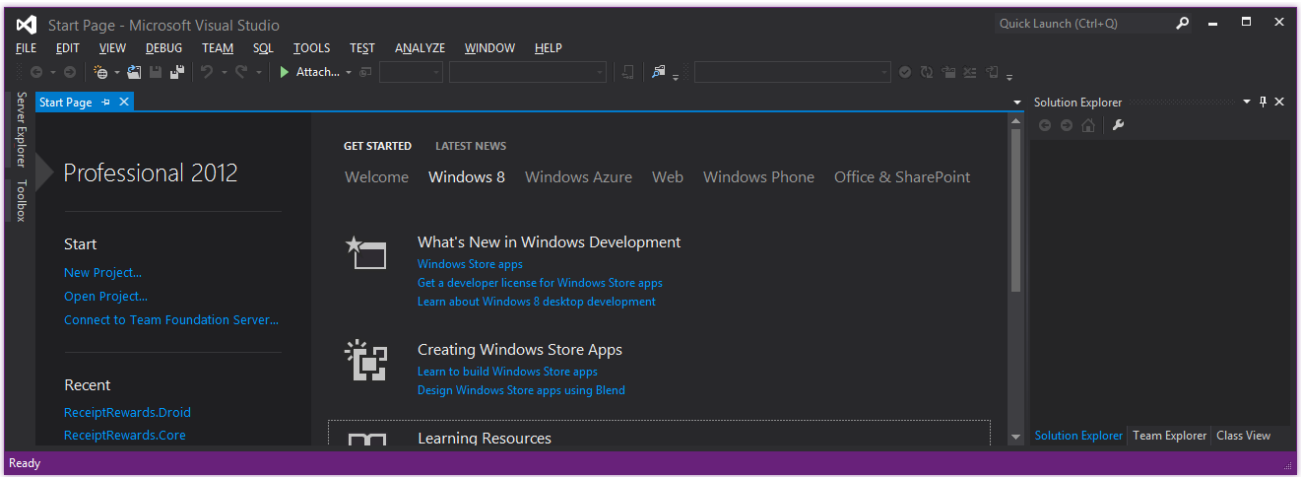
Before you can start, there are several steps that need to be done, such as configuring your environment

All software dependencies can be found in the **Software Requirement Specification**, but the following section explains the tricky setup steps that were needed in order to prepare the environment and have things running smoothly.



2.1 Visual Studios 2012

You will need to be running Visual Studios 2012 as it is designed to function properly with phone development and Xamarin plugins.



2.1.1 PCLs for Android and iOS

By default, PCL libraries only work for the Windows Platforms (Windows Phone, Windows Store, X-Box.) However, the project relies on the ability to put all the back-end logic into the PCL to be used by all Phone OS Platforms. Thus additional configuration must be done so that VS can recognize the PCL for all platforms.

To configure Visual Studios 2012 to create and use PCLs for Android and iOS, first navigate to the following directory:

C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework\.NETPortable\ v4.0\Profile\Profile104\Supporte

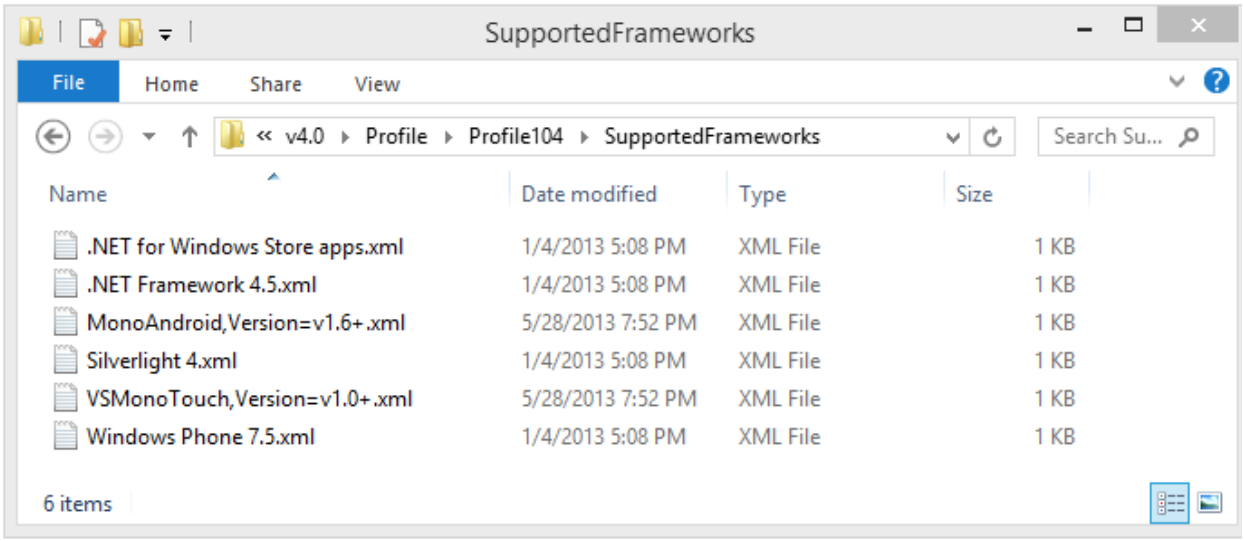
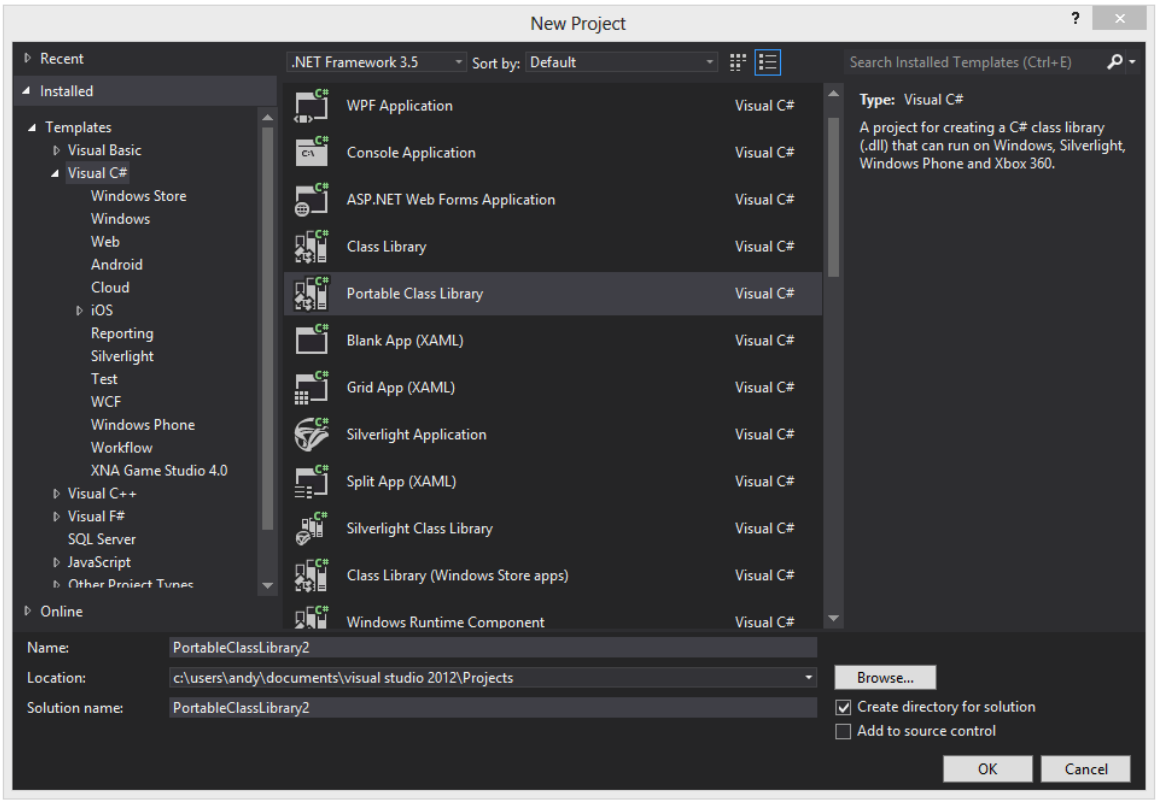


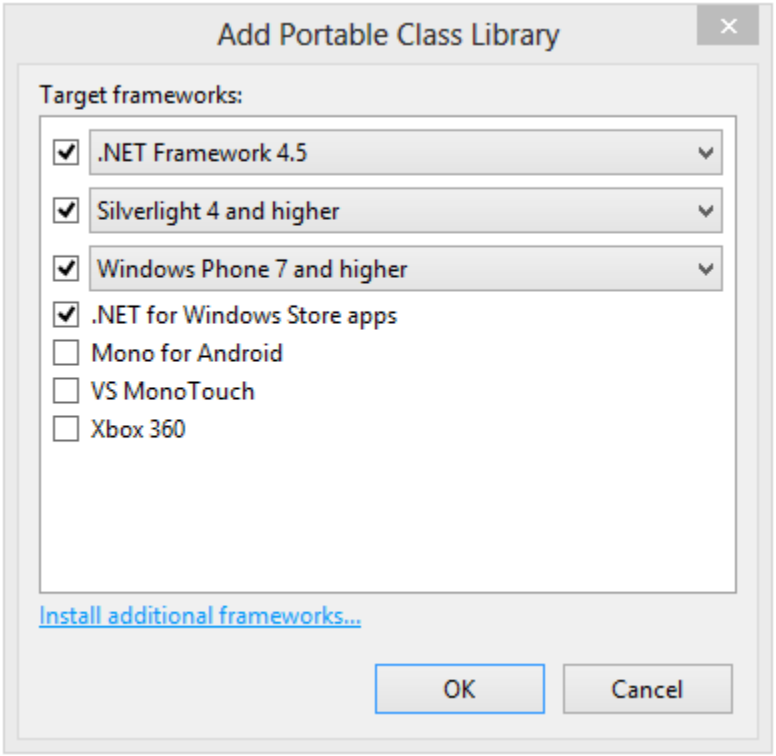
figure 2.1.1.1-1

In the developer guide resources, (that is provide in the code base,) you find files that will need to be added to this directory. Copy the files into this folder so that your directory should appear similar to **figure 2.1.1.1-1**:

- .NET for Windows Store apps.xml
- .NET Framework 4.5.xml
- MonoAndroid,Version=1.6_.xml
- Silverlight 4.xml
- VSMonoTouch,Version=v1.0+.xml
- Windows Phone 7.5.xml

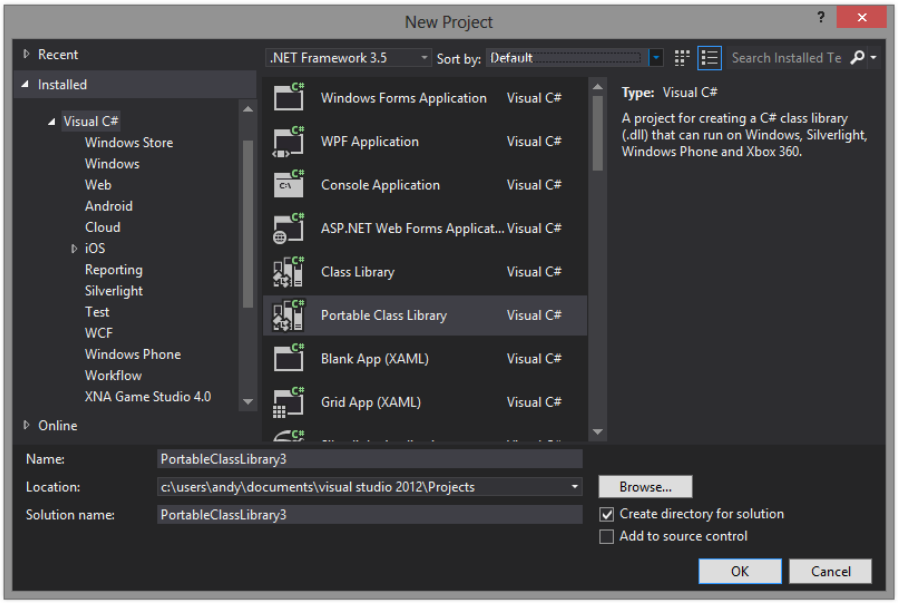


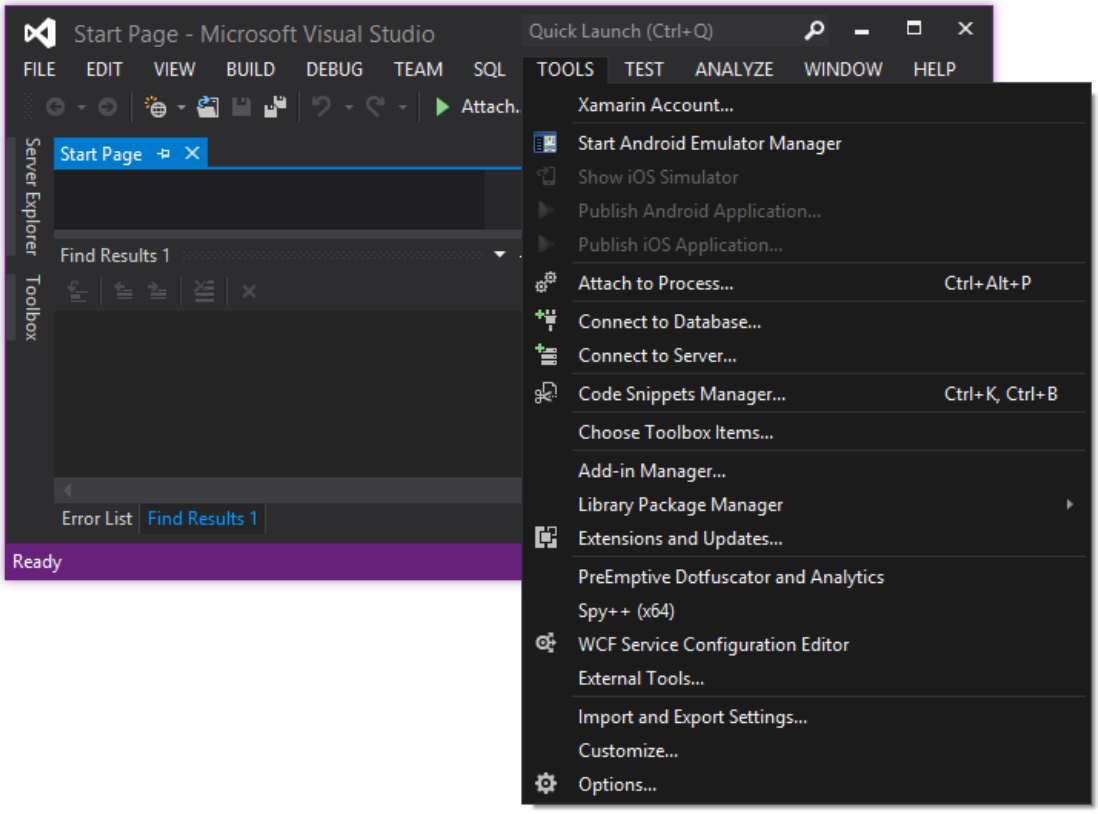
Once these files are in the folder system, open up Visual Studios 2012. When you create a new PCL library, a dialog will appear asking for what versions you want the PCL to target. If you are successful, all the platforms including Windows Phone, Android and iOS should appear.



2.1.2 Xamarin Plug-In

The project utilizes the Xamarin Platform, so the plugins must be installed. Below is a screenshot as to how the Xamarin Plugin appears in Visual Studios 2012.





2.2 NetBeans IDE

The NetBeans IDE is used to create the Web Services. Other Java IDEs could be used, but I use this one, thus instructions about how to set it up will be given.

2.2.1 Glassfish with SQL Server Support

Since the Database is on a Microsoft SQL Server, the glassfish server must contain the jar resource so that it can successfully make this connection. To configure this, follow the following steps.

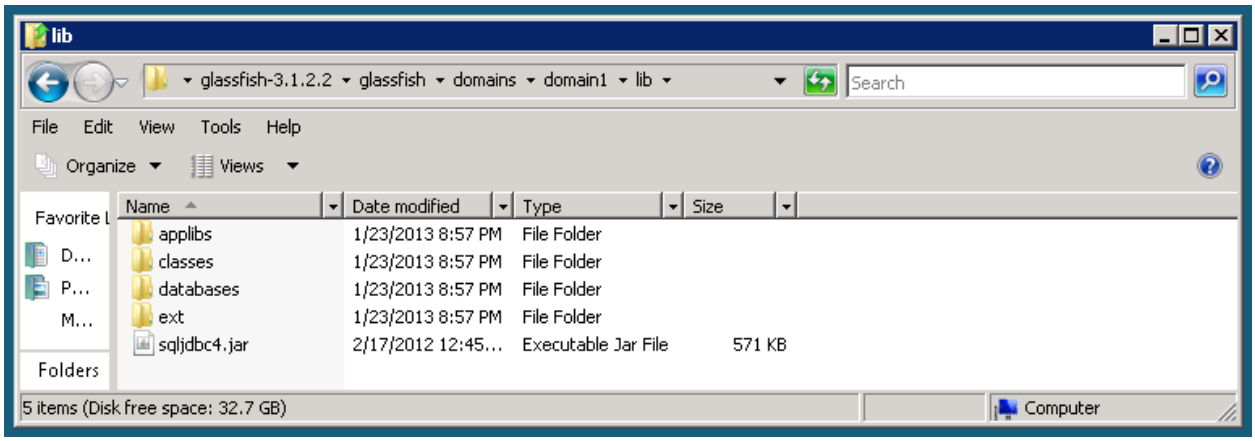


figure 2.2.1-1

First, find the location of the glassfish folder. Then, navigate to the relative URI, glassfish/domains/domainX/lib/ (where x is the domain number, default is 1). Once located in the libs folder, put the file sqljdbc4.jar into this folder, (the file can be found in the Developer Guide Resources Help Package.) The strucutre should be similar as seen in **figure 2.2.1-1**.

This will then fix allow for the server to have the jdbc connection loaded so that the web services can utilize the jar for the connection. To verify that the jar was properly loaded, navigate to the admin console windows of the glassfish server. Next, navigate to the JDBC Resources sections. In here will be the list of connection resource. If you create a new one, you will see several dropdown boxes. In the Driver field, find the option that says Microsoft SQL Server. If this field is present, then the JDBC Driver was successfully loaded.

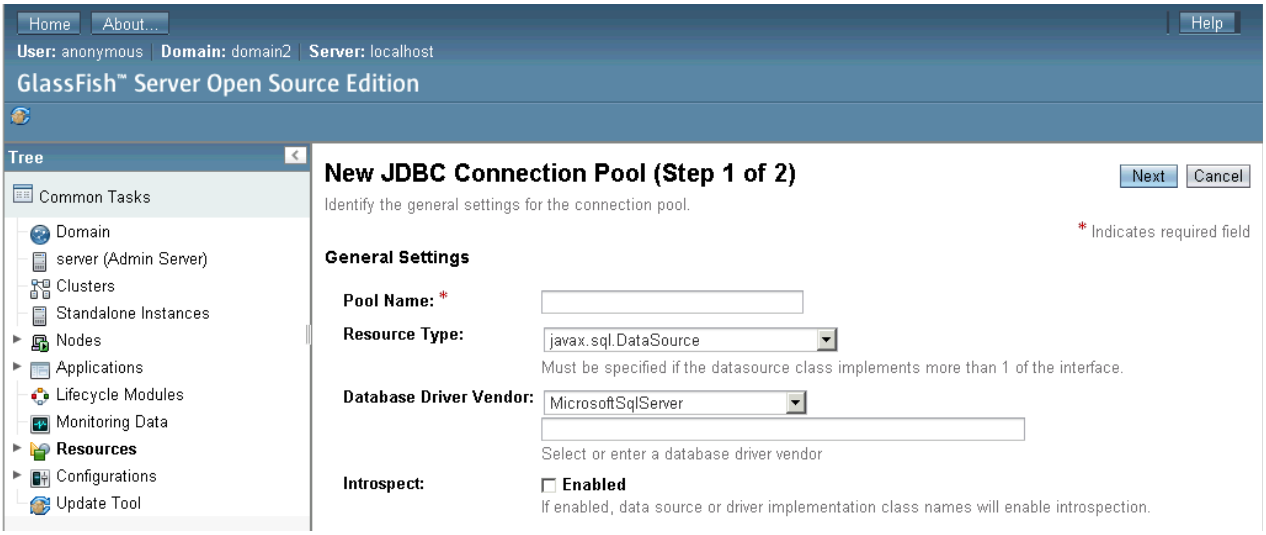


figure 2.2.1-2

2.3 Virtual Machines and Emulators

During the testing of the phone app, you will be using the Phone Emulators. However, you may run into a problem where you receive an error message, as seen in **figure 2.3–1**, and the Emulators fail to start.

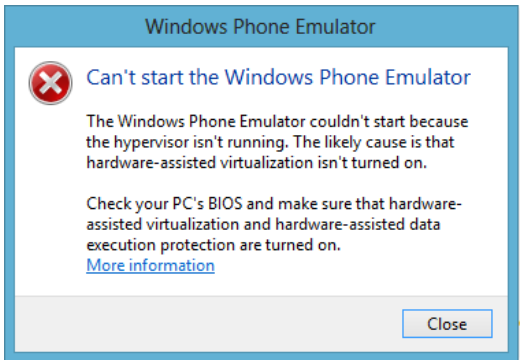


figure 2.3–1

The cause of this is that the PC is not set up to have multiple operating systems co-running on the system. This is a configuration found in the BIOS of the computer.

To fix this, restart computer. Prior to the OS starting, initialize the BIOS configuration screen of the PC. From here navigate to advanced PC settings. In this area, there are several properties. The following properties must be enabled to allow for virtual machines and emulators to be ran.

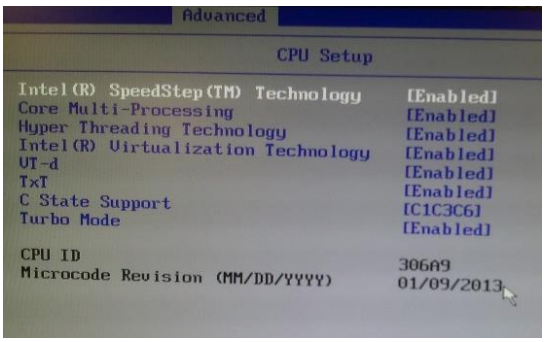


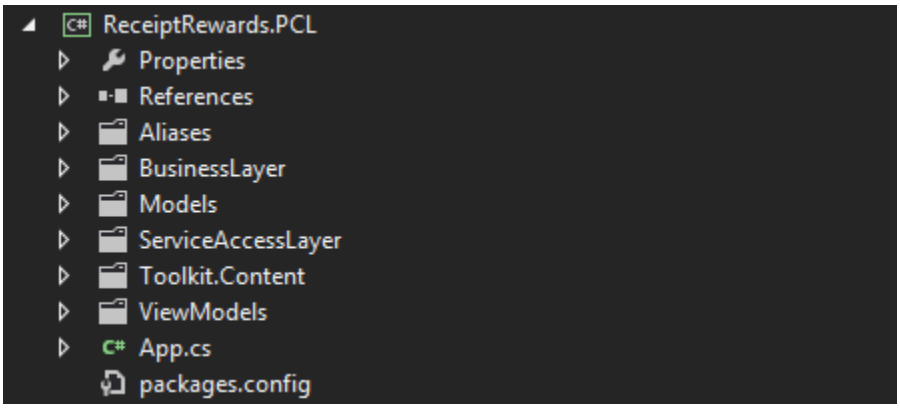
figure 2.3–2

- Hyper Threading Technology – Enabled
- Intel ® Virtualization Technology – Enabled
- VT-d – Enabled
- TxT - Enabled

Once these configurations are done, save the changes and restart the computer. Next time that you attempt to use a virtual machine or emulators, you will find that the error is gone and that they should be working normally.

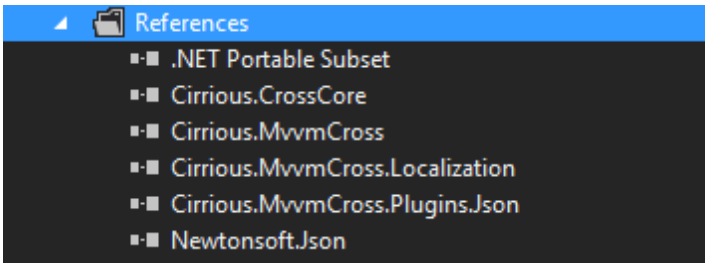
3 PCL

The portable class library is the common functionality needed for all version of the phone application. Think of the PCL as containing everything needed for the phone app, just without the user interface and handlers. The phone applications will utilize this special library to make the app work and function.



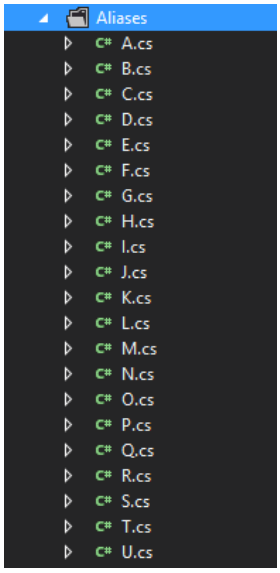
The PCL contains certain packages which will be described in greater detail below.

3.1 References Package



The references folder contains all the libraries that the PCL code is dependent on. As you can see in the structure below, the MCCM Cross Platform is used mainly for the ability to perform the Web Services JSON Requests, and the Newtonsoft is used to easily parse the JSON into objects.

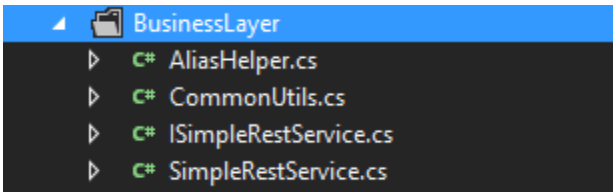
3.2 Aliases Package



The alias classes are the objects used to receive and transfer the standard objects to and from the Restful Services via an anonymous structure. This allows the request to be substantially smaller, thus improving the performance.

3.3 Business Layer Package

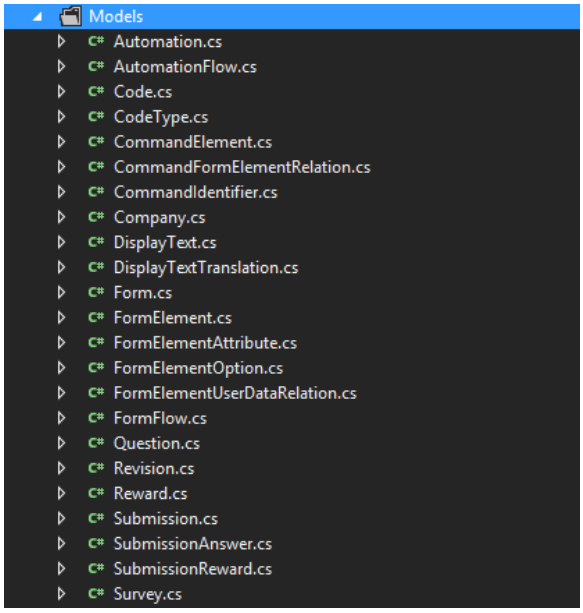
The business layer of the PCL contains static classes of methods that are commonly used throughout the rest of the PCL classes, and even for other code bases to use. By having this package, instills an emphasis on code reuse.



- *Alias Helper:* Contains the logic to convert all the alias objects to and from the standard objects.
- *Simple Rest Service:* Contains the logic to perform an asynchronous call to the Web Service Layer. The requests are based on JSON.
- *Common Utils:* Contains very miscellaneous functions that were used. Simply a code reuse of most common functions.

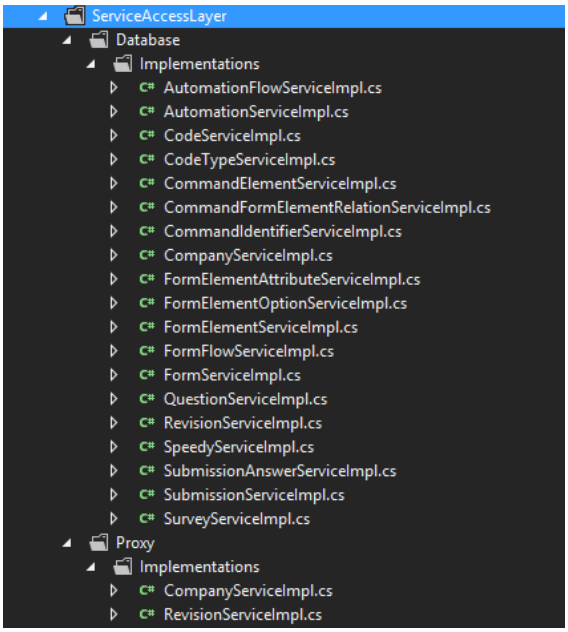
3.4 Models Package

The models contain all the objects used in the system for the C# code base.



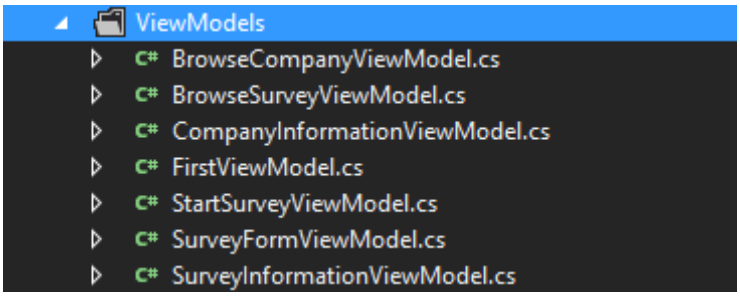
3.5 Service Access Layer Package

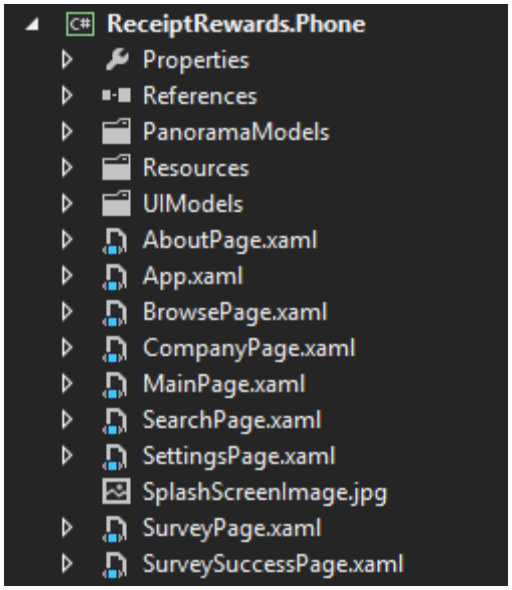
The Service Access Layer contains all the definitions to both the Web Services and the Reverse Proxy URL Locations.



3.6 View Models

The View Models is the most important aspect of the PCL in regards to the phone applications. These classes are logical representations of all the screens that will be in all the phone applications. In other words, these are only the business logic and functionality of a page. These are everything but the UI and event handlers. The actual applications will utilize a view model and hook up the UI and handlers to the logic located in the view models.

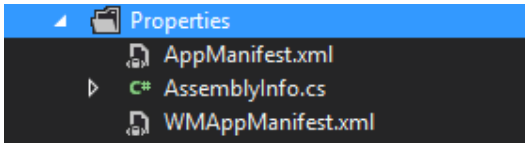




4 Windows Phone

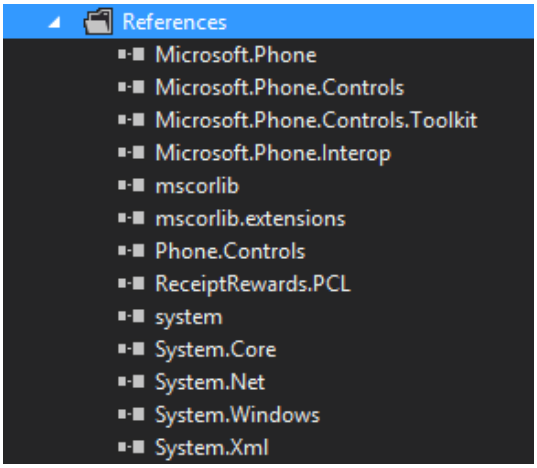
The windows phone project is the source code of creating the phone application. The structure to the project is follows the standard structure of the windows phone applications.

4.1 Properties Folder



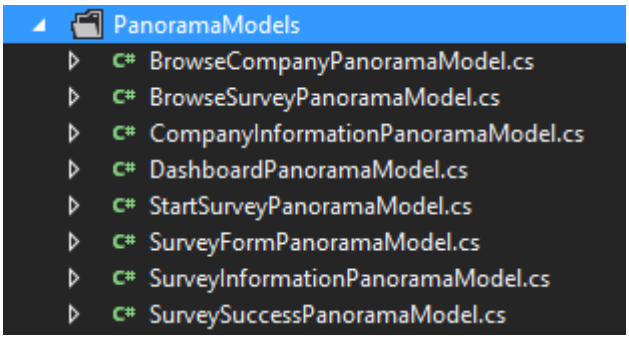
The properties folder contains the configuration files involved with how the app functions on the operating system.

4.2 References Folder



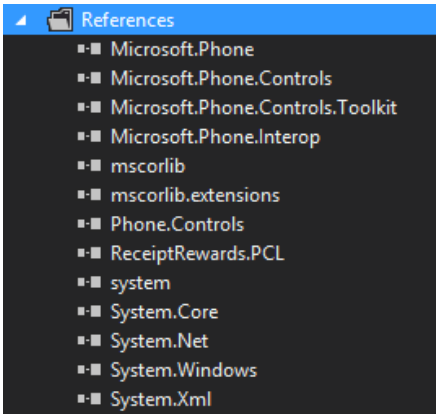
The references folder contains all the libraries and jar files that gets used in the phone application.

Specifically the libraries are the default jar files that are be default loaded into phone applications. Also, there are some open source controls that are installed too.



4.3 Panorama Model Package

The panorama package contains all the view objects that are the actual pages. These are the objects that get bind to the view models located inside the PCL library.

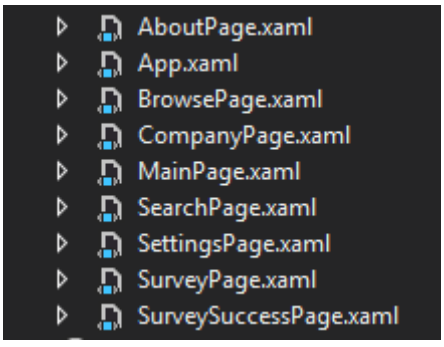


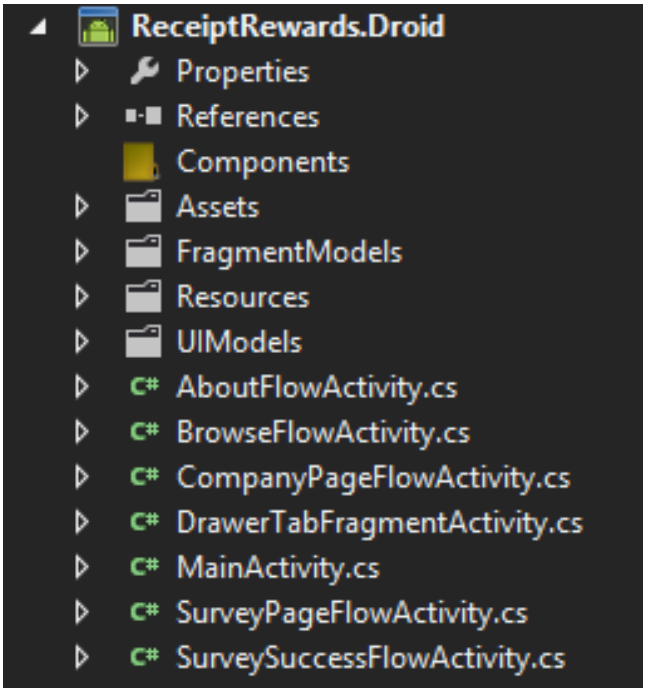
4.4 Resources Folder

The resources folder contains all of the extra files that the phone will need. Specifically this contains images that are displayed to the user.

4.5 Pages

These are the XAML pages are essentially an empty wrapper that is referenced to a backend .net instance. That instance is what creates programmatically a panorama view or single page view. It then calls the views from the Panorama Model package to fill with dynamic content. The content is the actual implementation of the view models.



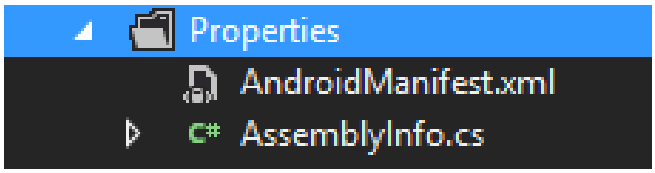


5 Android

The android files system also follows a very standard structure. This file structure also follows the MVVM Cross suggested structure.

5.1 Properties

The properties folder contains the configuration of how the app will work on the Android OS. Some of these properties may include:

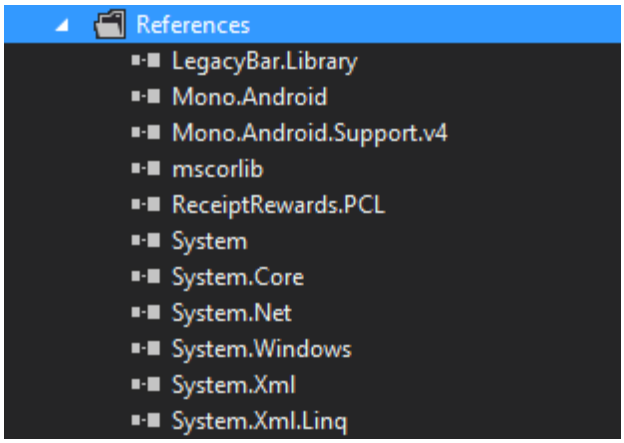


- Internet Connectivity
- App Nam
- Logo
- Author Information

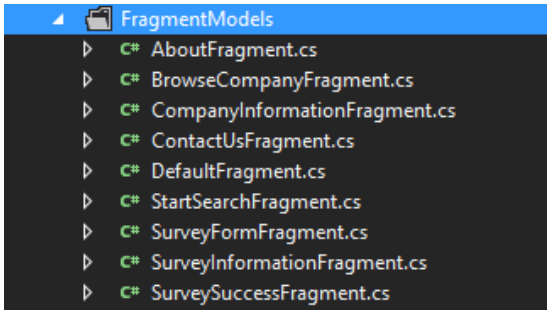
5.2 References

The references files structure contains all the dependencies of the Android Application. Most of these dependencies will include:

- Android OS Libraries (Mono.Android)
- .Net Libraries
- Android OS Component Jars (for display elements)

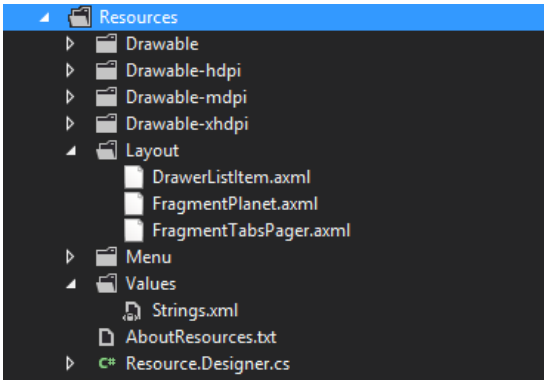


5.3 Fragment Models Package



The Fragment Models Package is the realization of the View Models (of the PCL.) These create programmatically a User Interface and event handlers, which it then hooks up to the View Model logic so that event handlers can automatically call the backend logic and update the UI when appropriate.

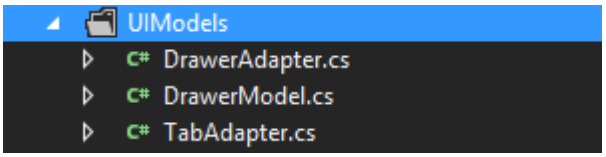
5.4 Resources Package



The resources folder contains a simple file system. These files include Pictures, logos and graphics that are displayed out to the user. It may also contain text files and lists which are used as static text to be display to the user, and create menu systems.

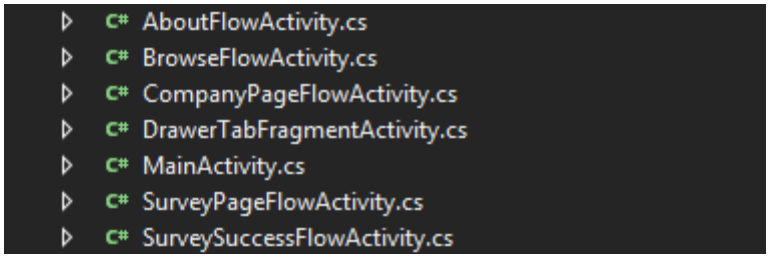
5.5 UI Models Package

UI Models is a package which contains static methods which are used to generate a User Interface object. The reason to put these components in here is to get reusability for elements that are recreated throughout the application. The structures in here are the Drawer and the Tab system. UI components in this package typically larger amount of complicated code and logic which also makes it advantageous to separate out.



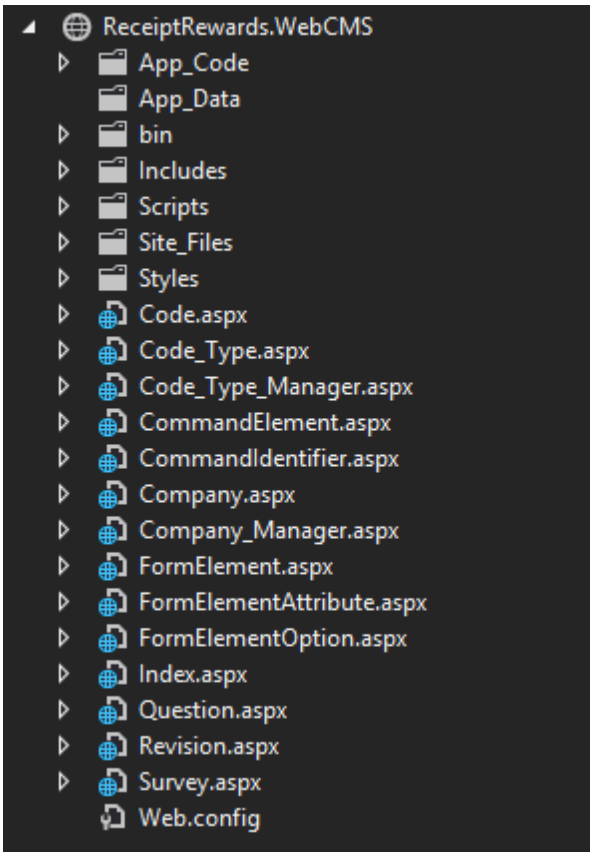
5.6 Pages

The main Android pages as you can see are located in the root view of the application. These pages are actually containers to hold the actual content, (the Fragments.) The pages are used when linking between pages. When a page is loaded, the onLoad function programmatically adds all the fragments into the main view.



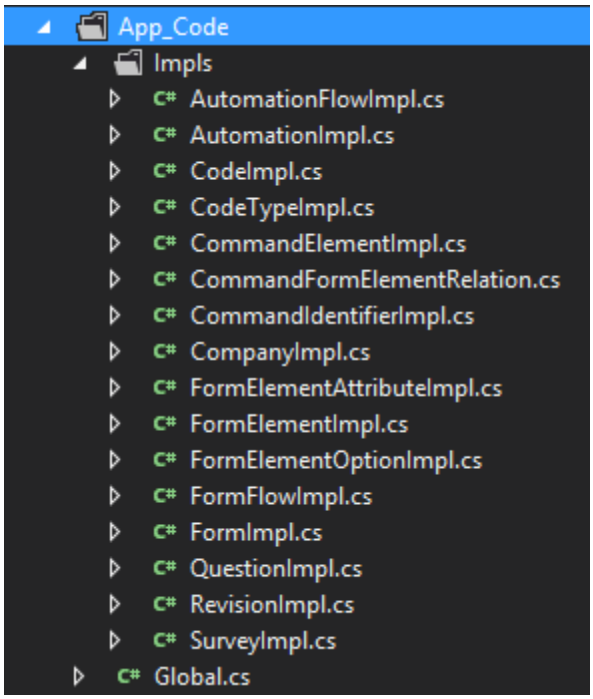
6 Web Admin CMS

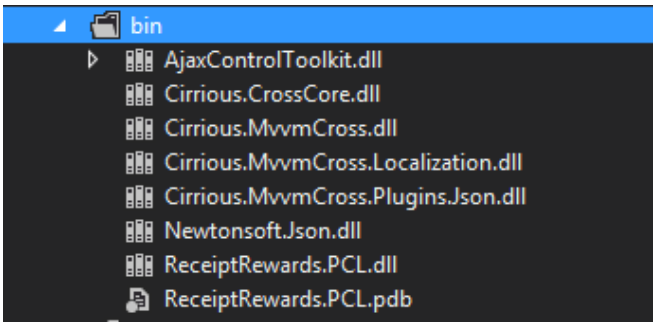
This System contains the code base which creates the Web Admin CMS application. This application is used to manage all the data in the database. Since this is a typical .Net Web Application, it follows a very similar file structure utilizing folders such as App_Code. The structure also contains standard website organization such as a Styles and Scripts folder.



6.1 App Code

The App_Code package contains all the C# static classes which contain logic that is utilized by the code_behind files. These Implantations are broken down by the models and contain the logic for restful services. They also include logic to generate the breadcrumbs for their given page.

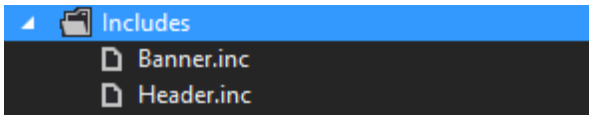




6.2 Bin

The bin folder contains all the jars and dll files which are the libraries that are used in the application. Some of these dependencies include .Net Controls.

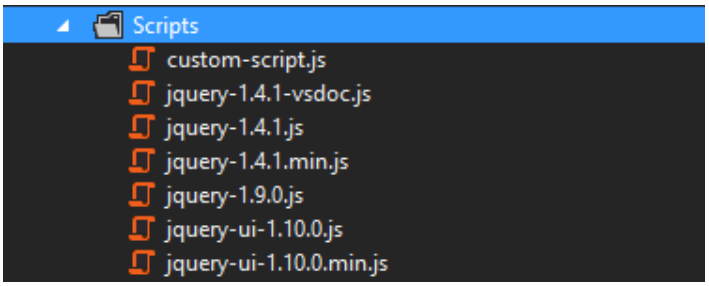
6.3 Includes



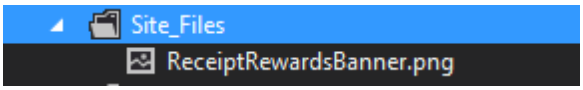
The Includes folder contains HTML / JavaScript that it used on every page. The Banner file is the logic for the actual generation of the banner and breadcrumbs. The Header contains the logic and includes for all the pages. This package demonstrates an emphasis of code reusability.

6.4 Scripts

The script folder contains all the JavaScript libraries that are used in the application. As you can see the only libraries that are used are jQuery libs. The custom-script.js contains custom written JavaScript for the Admin Site that is reused throughout the pages. An example of the code in here is the logic to collapse and expand the buttons on pages.

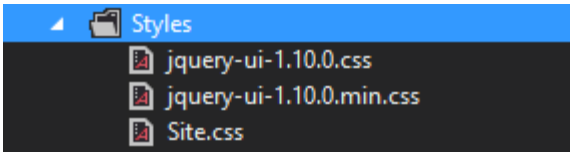


6.5 Site Files



The site files folder contains files that are used in the website. Types of files would include images, pdfs, txts, etc. As you can see, the application is pretty streamlined with only the banner graphic being the only image used in the website.

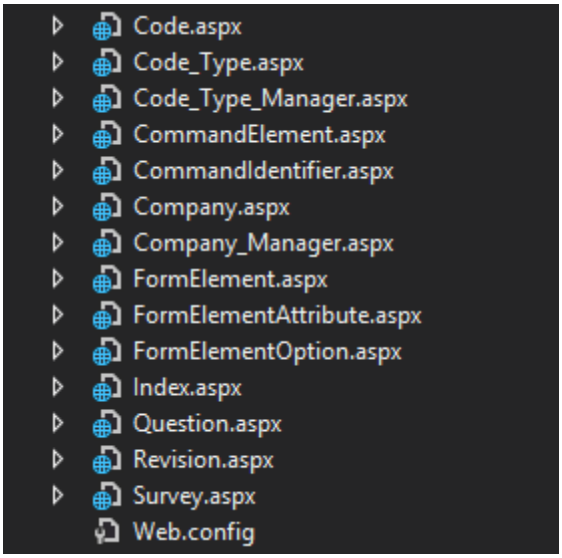
6.6 Styles



The style directory is the folder in which all the CSS Styling files will be placed. This provides easy organization.

6.7 Pages

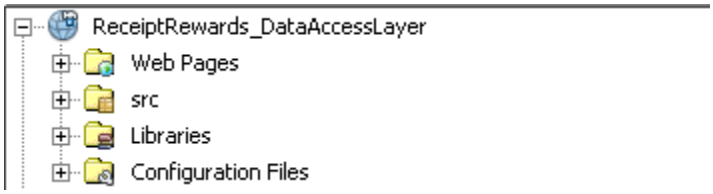
All the pages are located in the Root directory for the admin site. As you can see, there is no sub-navigation or nesting of the pages into subdirectories because the pages were self-evident and work in a very linear fashion.



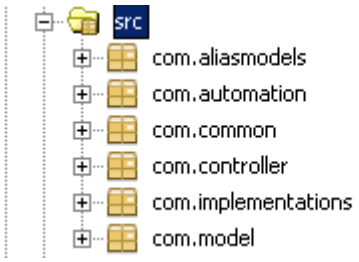
All the web pages are ASPX files, which is a special extension for the .Net Framework. Every page has an associated “code-behind” file, which contains C# code that contains the logic of event handlers. The .Net framework automatically binds the ASP controls with these functions.

7 Web Services

The web services are a Java Project that you will need NetBeans to manage.



The structure for the web services are all primarily in the src folder. The source is broken down into packages of which will be discussed more in-depth next.

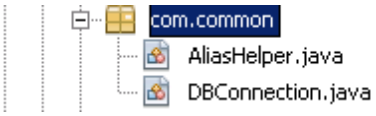


7.1 Alias Models Package

The Alias Models package contains the list of objects that are alias versions of the models.

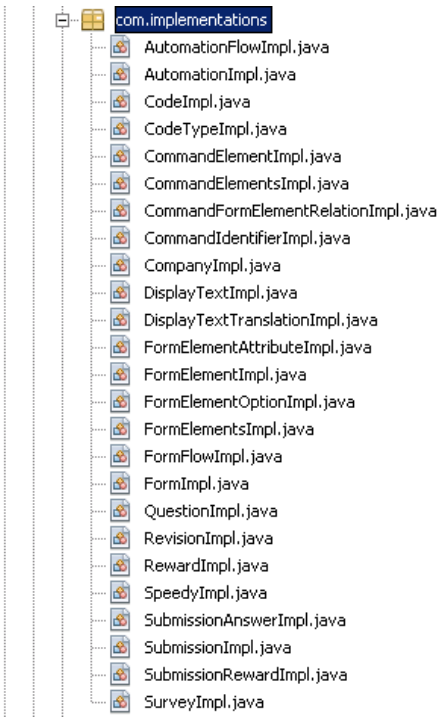
7.2 Automation Package

The automation package contains the classes that are involved with performing the Web Driver Automation logic. The methods are kicked off by a web service.



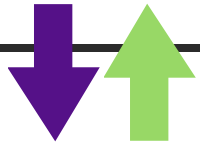
7.3 Common Package

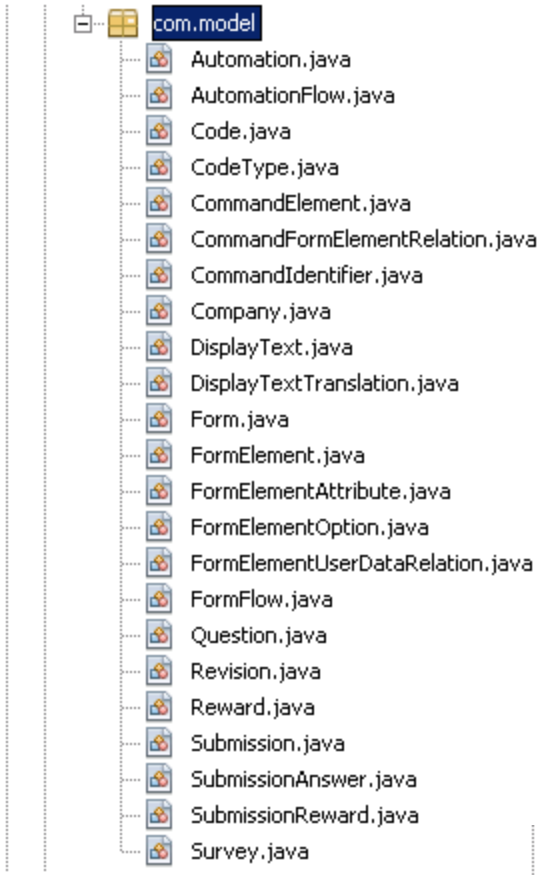
The common package contains static methods that are used throughout the application. You can think of them as helper files. The AliasHelper contains all the logic to convert models to and from their alias model counterparts. The DBConnection is used with all the data access methods. It creates the connection to the database.



7.4 Controller Package

The Controller Packages contains classes that map the web services to a specific URL. These methods are divided to contain all the CRUD actions for a particular model object. The methods contain the definitions for what the requests and responses of the Web Services.





7.5 Models Package

The Models Package contains the Models for this layer. Each object is a representation of the Object Orientated design that the database and the system follows.

7.6 Libraries

The libraries section contains a list of all the external libraries and dependencies that are used in the web services layer. As seen below, there is a strong dependency to the spring framework.