You have reached to the final step of developing your first Windows 8 XAML application following the MVVM design pattern—implementing the ViewModel of the FinanceHub application!

As part of implementing the ViewModel, this chapter will

- first build the MVVM framework that requires creating a foundation of building separation between the presentation (Views) and the data source (Model).
- then later implement ViewModels of the FinanceHub application that would create a collection of properties and bind them with the built Views.

This implementation will eventually lead us to connect to already implemented data source (the Model) and will enable population of the stocks and stock details information (the View) in our application following the MVVM design pattern. For this we will also make minor changes in the current View definition, where we hardcoded some of the values for the demonstration purpose in Chapter 2.

Building MVVM Framework for FinanceHub Application

With the introduction of XAML and the capabilities of binding data within XAML for Windows 8, WPF, and Silverlight applications, a new design pattern, Model-View-ViewModel (MVVM), emerged. With that, many custom MVVM frameworks, such as MVVM Light (http://mvvmlight.codeplex.com/), Caliburn.Micro (http://caliburnmicro.codeplex.com/), and Cinch (http://cinch.codeplex.com/), are developed, which provide pre-built helper classes, allowing the building of scalable MVVM pattern-bases application rapidly. If you visit these sites, you will notice that these frameworks are getting updated to support Windows 8 WinRT. You should keep an eye on these frameworks, on the progress they make to support the Windows 8 WinRT development. I would recommend exploring the use of one of these frameworks in your LoB WinRT-based Windows 8 application development as these frameworks mature.

However, for the purpose of learning the MVVM implementation for Windows 8 application in this book, we are going to implement a minimum footprint of the core services that would require building any MVVM-based Windows 8 application using XAML.
We will implement (or add as a dependency) the following five core services in this section to build the MVVM framework for our application:

- **IoC Container as Dependency** - Inversion of Control (IoC) container is used to implement dependency injection.

- **EventAggregator Class** – is a central container of registered events/messages, which supports pub/sub model, providing decoupling between the publisher and subscriber objects.

- **Navigation Service** – provides abstracted navigation by separating it from the View and enabling it within the ViewModel.

- **ViewModel Locator Class** – is based on the service locator pattern, which enables a clean way of assigning and initializing ViewModel and setting up the DataContext of the control in context, following IoC and dependency injection implementation approach.

- **Delegate Command Class** – allows creation of a typesafe and bindable reusable command (using ICommand interface) in ViewModels.

If you revisit the Setting MVVM Architecture in FinanceHub Project section of Chapter 1, you will notice that we created Dependencies and Infrastructure folders, which we will populate as part of this section to build the required MVVM framework. Let’s implement each service one by one in the following sections.

Open the latest FinanceHub project (which you updated last in Chapter 3) to implement ViewModel in this chapter.

**IoC Container Dependency**

One of the key concepts of building service-oriented and loosely coupled services is to decouple the application components from their dependencies. The Inversion of Control (IoC) is object-oriented design practice, which allows objects coupling at runtime. You would use IoC while implementing dependency injection in order for Views to find the required ViewModels at runtime.

For Microsoft .NET many IoC containers are available for WPF and Silverlight applications and eventually will be available for the WinRT also. In this book we will use a native WinRT IoC container, available on the codeplex site named metroioc ([http://metroioc.codeplex.com/](http://metroioc.codeplex.com/)), which can be used for Windows 8 WinRT-based applications.

---

**Note** To get details on available IoC resources and dependency injection containers for .NET, you can visit the blog of Scot Hanselman at [http://www.hanselman.com/blog/ListofNETDependencyInjectionContainersIOC.aspx](http://www.hanselman.com/blog/ListofNETDependencyInjectionContainersIOC.aspx).