Windows Phone Mango编程实践

***Windows Phone Mango Programming Practice***

第二篇 Silverlight交互篇

# MVVM设计模式的应用

MVVM设计模式犹如孙子兵法中的地形：“地形有通者，有挂者，有支者，有隘者，有险者，有远者。”夫地形者，兵之助也；MVVM则有助于应用程序的测试，维护和升级。熟练掌握MVVM的类及其相互作用，乃“将之至任，不可不察也”。故曰：“知此而用战者必胜，不知此而用战者必败”。

## MVVM设计模式概述

本章参考和引用Windows Phone官方教程和开发培训包，以及MSDN Windows Phone开发文档。

Model-View-ViewModel (MVVM)设计模式清晰的分离用户界面（UI）的业务逻辑和视图。分离业务逻辑和表示逻辑有助于解决许多设计问题，使应用程序更容易测试，维护和升级，可以大大提高代码重用的机会。MVVM使得用户界面设计人员专注于界面设计，开发人员专注于代码的业务逻辑，并使得开发人员和用户界面设计人员在软件制作过程能够更容易地合作。

应用MVVM设计模式的开放数据协议Open Data Protocol (OData)使用标准的HTTP协议实现查询、创建、 更新和删除数据的服务。关于开放数据协议 Open Data Protocol (OData) 我们在后面的章节中介绍。

在MVVM设计模式中，应用程序的用户界面和底层表示，以及业务逻辑被分成三个不同的部分：视图（View）、视图模型（ViewMode）和模型（Model）。MVVM设计模式犹如孙子兵法中的地形：“地形有通者，有挂者，有支者，有隘者，有险者，有远者。”夫地形者，兵之助也，MVVM则有助于应用程序的测试，维护和升级。熟练掌握MVVM的类及其相互作用，乃“将之至任，不可不察也”。故曰：“知此而用战者必胜，不知此而用战者必败”。

* 视图View—封装了用户界面和UI逻辑视图模型；

视图通常是某种形式的用户界面元素，使用户与应用程序交互。每个视图由可扩展应用程序标记语言 (XAML) 文件和其相应的代码组成。

* 视图模型ViewModel—封装表示逻辑和状态；

视图模型是视图和模型之间的中间层。

* 模型Model—封装了应用程序的业务逻辑和数据；

此对象模型通常不包含任何用户界面有关的信息。

下图展示了MVVM类及其相互作用。



图14-1 MVVM类及其相互作用

## 动手实践——应用MVVM设计模式的数独游戏

本实例参考Adam Miller发表在MSDN Magazine的文章Sudoku for Windows Phone 7。数独游戏（日语：数独すうどく）是一种源自18世纪末的瑞士，后在美国发展、并在日本得以发扬光大的数学智力拼图游戏。拼图是九宫格（即3格宽×3格高）的正方形状，每一格又细分为一个九宫格。在每一个小九宫格中，分别填上1至9的数字，让整个大九宫格每一行、每一列的数字都不重复。数独游戏的玩法逻辑简单，数字排列方式千变万化，不少教育者认为数独游戏是智力锻炼的好方法。

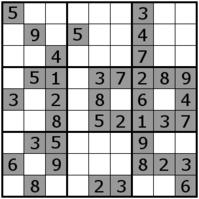


图14-5 数独游戏

### 创建应用程序

从开始 | 所有程序 | Microsoft Visual Studio 2010 Express | Microsoft Visual Studio 2010 Express for Windows Phone中打开 。或者从开始 | 所有程序| Microsoft Visual Studio 2010打开Visual Studio 2010。

首先按照常用 MVVM 模式创建两个新文件夹，即Views和ViewModels。

数独游戏在概念上可分为三个类型：各个方格（9x9 游戏板中通常共有 81 个方格）；容纳这些方格的整体游戏板；用于输入数字1到数字9的网格。若要创建这些项的视图，请右键单击 Views 文件夹，再选择“Add | New Item”。从对话框中选择“Windows Phone User Control”，命名为 GameBoardView.xaml。对SquareView.xaml InputView.xaml重复上述操作。

此时，在 ViewModel 文件夹中添加以下类：GameBoardViewModel 和 SquareViewModel。为 ViewModels创建一个基类以避免代码重复。向 ViewModels 文件夹添加 ViewModelBase类。

数独游戏的实现MVVM设计模式的Views和ViewModels文件结构如下图14-6 Sudoku Windows Phone解决方案所示。

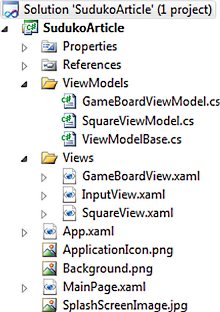


图14-6 Sudoku Windows Phone解决方案

### ViewModelBase类

ViewModelBase类实现System.ComponentModel 中的INotifyPropertyChanged 接口。此接口允许将 ViewModels 中的公共属性绑定到视图中的控件。在本例中INotifyPropertyChanged 接口的实现只需实现 PropertyChanged事件即可。

Silverlight Project: SudukoArticle File: ViewModels\ViewModelBase.cs

namespace SudukoArticle.ViewModels

{

public class ViewModelBase : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

protected void NotifyPropertyChanged(String info)

{

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs(info));

}

}

}

}

大多数第三方 MVVM 框架将包括一个 ViewModel 基类，其中包含此样本代码。所有的ViewModel都将从ViewModelBase继承。将UI绑定到的 ViewModel 中的属性必须调用setter中的 NotifyPropertyChanged。这是允许UI在属性值发生更改时自动更新的设置。

### 实现各个方格

首先实现 SquareViewModel 类。将 Value、Row、Column 的公共属性添加为整数；添加 IsSelected、IsValid 和 IsEditable，设定类型为布尔值。虽然可将UI直接绑定到Value属性，但这将导致出现问题，因为将为未分配的方格显示“0”。若要解决此问题，可以实现绑定转换器或创建只读“StringValue”属性，该属性将在Value属性为零时返回空字符串。

Silverlight Project: SudukoArticle File: ViewModels\SquareViewModel.cs

public int Value

{

get

{

return \_value;

}

set

{

if (IsEditable)

{

\_value = value;

NotifyPropertyChanged("Value");

NotifyPropertyChanged("StringValue");

UpdateState();

}

}

}

public string StringValue

{

get

{

string result = "";

if (\_value > 0)

result = \_value.ToString();

return result;

}

}

SquareViewModel 还负责向 UI 通知其当前状态。此应用程序中的单个方格具有四种状态，即“Default”（默认）、“Invalid”（无效）、“Selected”（已选定）和“UnEditable”（不可编辑）。通常，这将作为枚举实现；但 Silverlight 框架中的枚举不包含完整 Microsoft .NET Framework 的枚举所具有的几种方法。这会导致在序列化期间引发异常，因此已将状态实现为常数：

Silverlight Project: SudukoArticle File: Utility\BoxStates.cs

namespace SudukoArticle.Utility

{

public class BoxStates

{

public const int Default = 1;

public const int Invalid = 2;

public const int Selected = 3;

public const int UnEditable = 4;

}

}

此应用程序将使用与用户选定的主题匹配的颜色和字体样式。MSDN Library 上的“Theme Resources for Windows Phone”中对这些资源进行了描述。可通过转到主页屏幕，并单击箭头，选择“Settings | theme”在仿真器中选择主题。

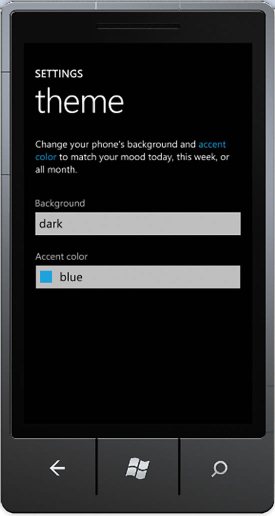


图14-7 Windows Phone主题设置屏幕

在SquareView.xaml中的控件使用系统提供的字体和颜色样式。在SquareView.xaml中的网格内，放置Border 和TextBlock。

Silverlight Project: SudukoArticle File: Views\SquareView.xaml

<UserControl x:Class="SudukoArticle.Views.SquareView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:utility="clr-namespace:SudukoArticle.Utility"

mc:Ignorable="d"

FontFamily="{StaticResource PhoneFontFamilyNormal}"

FontSize="{StaticResource PhoneFontSizeNormal}"

Foreground="{StaticResource PhoneForegroundBrush}"

d:DesignHeight="40" d:DesignWidth="40">

<Grid x:Name="LayoutRoot" MouseLeftButtonDown="LayoutRoot\_MouseLeftButtonDown">

<Border x:Name="BoxGridBorder" BorderBrush="{StaticResource PhoneForegroundBrush}" BorderThickness="{Binding Path=BorderThickness}">

<TextBlock x:Name="MainText" VerticalAlignment="Center" Margin="0" Padding="0" TextAlignment="Center" Text="{Binding Path=StringValue}"></TextBlock>

</Border>

</Grid>

</UserControl>

### 实现GameBoard

本节讲述GameBoard 的视图（View）和视图模式（ViewModel）。GameBoard视图是由9\*9的网格构成。

ViewModel包含用于在用户输入后验证游戏板的方法，用于显示答案的方法，以及用于保存和加载游戏的方法。在保存时会将游戏板序列化为 XML，并在独立存储空间保存此文件存储代码是最有趣的。

Silverlight Project: SudukoArticle File: ViewModels\GameBoardViewModel.cs

public void SaveToDisk()

{

using (IsolatedStorageFile store = IsolatedStorageFile.GetUserStoreForApplication())

{

if (store.FileExists(FileName))

{

store.DeleteFile(FileName);

}

using (IsolatedStorageFileStream stream = store.CreateFile(FileName))

{

using (StreamWriter writer = new StreamWriter(stream))

{

List<SquareViewModel> s = new List<SquareViewModel>();

foreach (SquareViewModel item in GameArray)

s.Add(item);

XmlSerializer serializer = new XmlSerializer(s.GetType());

serializer.Serialize(writer, s);

}

}

}

}

public static GameBoardViewModel LoadFromDisk()

{

GameBoardViewModel result = null;

using (IsolatedStorageFile store = IsolatedStorageFile.GetUserStoreForApplication())

{

if (store.FileExists(FileName))

{

using (IsolatedStorageFileStream stream = store.OpenFile(FileName, FileMode.Open))

{

using (StreamReader reader = new StreamReader(stream))

{

List<SquareViewModel> s = new List<SquareViewModel>();

XmlSerializer serializer = new XmlSerializer(s.GetType());

s = (List<SquareViewModel>)serializer.Deserialize(new StringReader(reader.ReadToEnd()));

result = new GameBoardViewModel();

result.GameArray = LoadFromSquareList(s);

}

}

}

}

return result;

}

应用程序栏的“New Game”按钮单击事件中调用LoadNewPuzzle方法新建游戏。

Silverlight Project: SudukoArticle File: ViewModels\GameBoardViewModel.cs

public static GameBoardViewModel LoadNewPuzzle()

{

GameBoardViewModel result = new GameBoardViewModel();

Random random = new Random();

string easyPuzzle = SavedBoards.EasyGames[random.Next(0, SavedBoards.EasyGames.Length - 1)];

List<SquareViewModel> squares = new List<SquareViewModel>();

foreach (char s in easyPuzzle.ToCharArray())

{

SquareViewModel square = new SquareViewModel();

if (s != '.')

{

square.Value = int.Parse(s.ToString());

square.IsEditable = false;

}

squares.Add(square);

}

result.GameArray = LoadFromSquareList(squares);

return result;

}

### 实现输入视图

输入视图也非常简单，它只是在面板堆栈中嵌入几个按钮。下面的代码公开了按钮的单击事件处理函数，当单击事件发生时向应用程序发送已单击按钮的值。

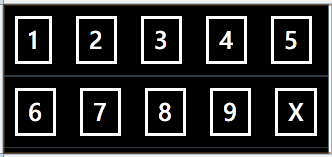


图14-8 输入视图

Silverlight Project: SudukoArticle File: Views\InputView.xaml.cs

public event EventHandler SendInput;

private void UserInput\_Click(object sender, RoutedEventArgs e)

{

int inputValue = int.Parse(((Button)sender).Tag.ToString());

if (SendInput != null)

SendInput(inputValue, null);

}

用于帮助使该游戏能在纵向模式或横向模式的方法。

Silverlight Project: SudukoArticle File: Views\InputView.xaml.cs

public void RotateVertical()

{

TopRow.Orientation = Orientation.Vertical;

BottomRow.Orientation = Orientation.Vertical;

OuterPanel.Orientation = Orientation.Horizontal;

}

public void RotateHorizontal()

{

TopRow.Orientation = Orientation.Horizontal;

BottomRow.Orientation = Orientation.Horizontal;

OuterPanel.Orientation = Orientation.Vertical;

}

### 整合视图

将GameBoard输入视图集成到MainPage.xaml中，即将输入视图和游戏板视图置于一个网格（Grid）中。由于此应用程序需要所有可用的屏幕空间，因此将默认的标题面板（TitlePanel）的Visibility属性设置为“Collapsed”。

设置应用程序栏。通过使用此应用程序栏，可使应用程序具有高集成性的用户体验，并将为数独游戏应用程序提供一个很好的接口，为用户解答、重置和开始新游戏。

Silverlight Project: SudukoArticle File: MainPage.xaml

<phone:PhoneApplicationPage.ApplicationBar>

<shell:ApplicationBar IsVisible="True" IsMenuEnabled="True" Opacity="0">

<shell:ApplicationBarIconButton x:Name="NewGame" IconUri="/Images/appbar.favs.rest.png" Text="New Game" Click="NewGame\_Click"></shell:ApplicationBarIconButton>

<shell:ApplicationBarIconButton x:Name="Solve" IconUri="/Images/appbar.share.rest.png" Text="Solve" Click="Solve\_Click"></shell:ApplicationBarIconButton>

<shell:ApplicationBarIconButton x:Name="Clear" IconUri="/Images/appbar.refresh.rest.png" Text="Clear" Click="Clear\_Click"></shell:ApplicationBarIconButton>

</shell:ApplicationBar>

</phone:PhoneApplicationPage.ApplicationBar>

Windows Phone提供默认图标，与工具一起安装到 C:\Program Files\Microsoft SDKs\Windows Phone\v7.1\Icons。这些图标确实值得选择，因为它们与手机的外观相匹配。在将图像导入项目后，选择图像属性，并将“生成操作”从“资源”更改为“内容”，然后将“复制到输出目录”从“不复制”更改为“如果较新则复制”。

在MainPage构造函数中，设置SupportedOrientations属性以允许应用程序在用户旋转手机时随之旋转。另外，处理InputView的SendInput事件，并将输入值传送到GameBoard。

Silverlight Project: SudukoArticle File: MainPage.xaml.cs

// Constructor

public MainPage()

{

InitializeComponent();

SupportedOrientations = SupportedPageOrientation.Portrait | SupportedPageOrientation.Landscape;

InputControl.SendInput += new EventHandler(InputControl\_SendInput);

}

void InputControl\_SendInput(object sender, EventArgs e)

{

MainBoard.GameBoard.SendInput((int)sender);

}

实现OnNavigatedTo和OnNavigatedFrom以便在应用程序进入逻辑删除以及重新唤醒后，加载和保存游戏。

Silverlight Project: SudukoArticle File: MainPage.xaml.cs

protected override void OnNavigatedTo(NavigationEventArgs e)

{

GameBoardViewModel board = GameBoardViewModel.LoadFromDisk();

if (board == null)

board = GameBoardViewModel.LoadNewPuzzle();

MainBoard.GameBoard = board;

base.OnNavigatedTo(e);

}

protected override void OnNavigatedFrom(NavigationEventArgs e)

{

MainBoard.GameBoard.SaveToDisk();

base.OnNavigatedFrom(e);

}

当手机发生旋转时，应用程序将收到一个通知。InputView 会从该位置开始从游戏板下方移动到其右侧并进行旋转。

Silverlight Project: SudukoArticle File: MainPage.xaml.cs

protected override void OnOrientationChanged(OrientationChangedEventArgs e)

{

switch (e.Orientation)

{

case PageOrientation.Landscape:

case PageOrientation.LandscapeLeft:

case PageOrientation.LandscapeRight:

TitlePanel.Visibility = Visibility.Collapsed;

Grid.SetColumn(InputControl, 1);

Grid.SetRow(InputControl, 0);

InputControl.RotateVertical();

break;

case PageOrientation.Portrait:

case PageOrientation.PortraitUp:

case PageOrientation.PortraitDown:

TitlePanel.Visibility = Visibility.Collapsed;

Grid.SetColumn(InputControl, 0);

Grid.SetRow(InputControl, 1);

InputControl.RotateHorizontal();

break;

default:

break;

}

base.OnOrientationChanged(e);

}

应用程序栏的新建游戏按钮、游戏解答按钮和清除游戏按钮的单击事件处理。

Silverlight Project: SudukoArticle File: MainPage.xaml.cs

private void NewGame\_Click(object sender, EventArgs e)

{

MainBoard.GameBoard = GameBoardViewModel.LoadNewPuzzle();

}

private void Solve\_Click(object sender, EventArgs e)

{

MainBoard.GameBoard.Solve();

}

private void Clear\_Click(object sender, EventArgs e)

{

MainBoard.GameBoard.Clear();

}

此时，该游戏已完成，可以开始玩了。

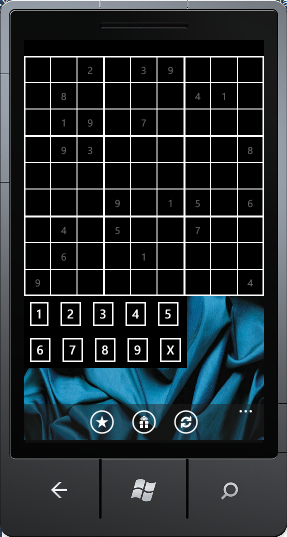
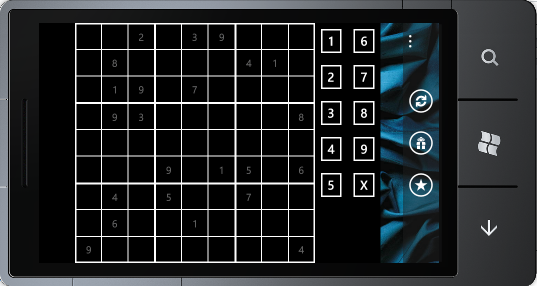
 

图14-9 纵向模式和横向模式的数独游戏

## 第三方MVVM框架

在[www.CodePlex.com](http://www.CodePlex.com)上有许多不错的第三方MVVM框架可供我们借鉴，其中比较推崇的有MVVM Light Toolkit、Ultra Light MVVM for Windows Phone和Simple MVVM Toolkit都适合Windows Phone的开发。

### MVVM Light Toolkit

MVVM Light Toolkit是帮助人们在Silverlight和WPF中使用MVVM设计模式的一套组件。It is a light and pragmatic framework that contains only the essential components needed.它是一个轻量级的、务实的框架，只包含所需的必要组成部分。

下载地址： <http://mvvmlight.codeplex.com/> 。

1. **Get started**在XAML中绑定DataContext

ViewModel文件夹添加了两个附加的类：ViewModelLocator.cs和MainViewModel.cs。ViewModelLocator声明在 App.xaml中可用作源的 DataContext 绑定；MainViewModel 也已添加到项目中 ViewModelLocator 作为公开属性。

**注意**

如果在项目中不需要使用MainViewModel，可以删除此类。

为了使用MainViewModel作为MainPage.xaml的DataContext，需要在MainPage.xaml的开始标记中添加以下代码：

DataContext="{Binding Main, Source={StaticResource Locator}}"

1. 在Blend中绑定DataContext

DataContext 也可以直观地在Expression Blend中使用以下步骤中绑定。

1. 在Expression Blend中打开Windows Phone工程；
2. 编译应用程序；
3. 打开MainPage.xaml；
4. 在Objects和Timeline panel选择UserControl；
5. 在属性面板中，选择DataContext 属性(在Common Properties部分)，在Advanced属性的下拉菜单中选择数据绑定。如图 Blend的数据绑定。

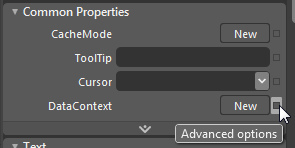
[](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http://www.flickr.com/photos/36917929@N06/5696690247)

图14-2 Blend的数据绑定

More information about the MVVM Light Toolkit can be found on [http://www.galasoft.ch/mvvm/getstarted](http://translate.googleusercontent.com/translate_c?hl=zh-CN&ie=UTF8&prev=_t&rurl=translate.google.co.jp&sl=en&tl=zh-CN&u=http://www.galasoft.ch/mvvm/getstarted&usg=ALkJrhgY8UIueXYXmHnyvCSlmLXQ2DuB0g)更多关于MVVM Light Toolkit的帮助请登录[http://www.galasoft.ch/mvvm/getstarted](http://translate.googleusercontent.com/translate_c?hl=zh-CN&ie=UTF8&prev=_t&rurl=translate.google.co.jp&sl=en&tl=zh-CN&u=http://www.galasoft.ch/mvvm/getstarted&usg=ALkJrhgY8UIueXYXmHnyvCSlmLXQ2DuB0g) .。

### Ultra Light MVVM for Windows Phone 7

Ultra Light MVVM是轻量级的MVVM Silverlight应用程序，支持Windows Phone的逻辑删除（tombstoning）。Ultra Light MVVM支持Windows Phone的特性：

1. 命令
2. 命令按钮（使用参数）绑定
3. 支持将命令绑定到应用程序栏上的按钮
4. 对话框，通知和确认
5. 位置服务
6. 友好的设计时视图模式
7. 具有逻辑删除事件控制钩的逻辑删除友好视图模型
8. 从视图模型解耦的导航支持
9. 从视图模型解耦的视觉状态支持
10. 在视图模型拦截后退键
11. 用户界面线程访问的分配器帮助

下载地址： <http://ultralightmvvm.codeplex.com/> 。

### Simple MVVM Toolkit

Simple MVVM Toolkit使开发应用MVVM设计模式Widnows Phone的应用程序变得更容易，为基于 MVVM设计模式的应用程序提供一个简单的框架和工具集。Simple MVVM Toolkit的特点是简单，但它包含执行 MVVM 设计模式的应用程序所需的一切。

下载地址： <http://simplemvvmtoolkit.codeplex.com/> 。

来自Tony Sneed博客的Simple MVVM Toolkit和MVVM Light Toolkit的比较结果。

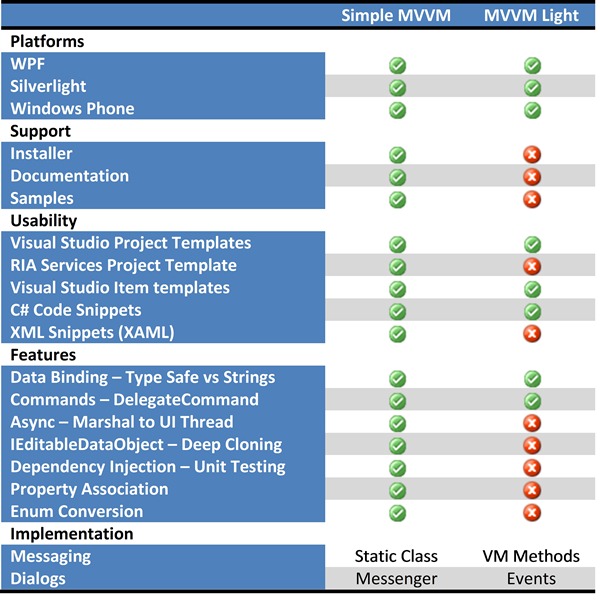
[](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http://blog.tonysneed.com/2011/04/23/simple-mvvm-toolkit-versus-mvvm-light-toolkit/)

图14-3 Simple MVVM Toolkit和MVVM Light Toolkit的比较

Simple MVVM Toolkit安装后的MVVM应用程序如图 Simple MVVM Toolkit。

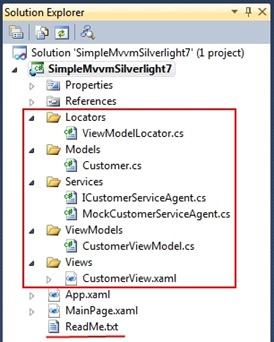


图14-4 Simple MVVM Toolkit