MVVM 设计模式及其应用研究*

陈 涛
(中船重工集团公司第 722 研究所 武汉 430205)

摘 要 MVVM 即 Model-View-ViewModel,是微软 WPF 和 Silverlight 应用特有的一种界面设计模式。使用 MVVM 设计模式可以帮助我们分离业务逻辑,显示逻辑和用户界面,使得我们的程序代码结构清晰,容易被阅读、测试、维护、替换、扩展和改进。论文意在通过逐层分解对 MVVM 及其应用进行研究。

关键词 MVVM; 设计模式; 用户界面
中图分类号 TP311 **DOI**:10.3969/j.issn1672-9722.2014.10.055

MVVM Design Pattern and Its Application

CHEN Tao
(No. 722 Research Institute of CSIC, Wuhan 430205)

Abstract MVVM is equivalent to the Model-View-View Model, is a kind of UI design pattern For Microsoft's WPF and Silverlight application. Using MVVM can help us separate the business logic, present logic and UI, making our code structure clear, easy to read, test, maintenance, replace, expand and improve. This article is intended to study MVVM and its application.

Key Words MVVM, design pattern, UI
Class Number TP311

1 MVVM 概述

随着程序功能的拓展与扩张,代码量也会相应地增加。程序也将会变得十分复杂。设计模式可以帮助解决用户界面不断变更的难题,并且能够使代码层次分明,易于阅读,条理清晰,并且便于后期维护和改进。另一方面,如果开发人员更换,新的开发人员可以以更高的效率理解代码并参与后续的开发和维护。

在 WPF 以及 Silverlight 应用程序设计中使用 MVVM 设计模式能够实现业务逻辑,显示逻辑和用户界面的分离。带来的好处是使得我们的程序代码结构清晰,易于阅读、测试、维护、替换、扩展和改进[1]。

在程序开发人员与界面设计人员分工明确的情况下,使用 MVVM,解除了 View 和 ViewModel 之间紧耦合的弊端,使得程序开发人员和界面设计人员可以各司其职,分别专注于创建更加完善的 ViewModel 处理逻辑和优雅友好的 View 界面[2]。

2 MVVM 模式剖析

2.1 MVVM 模式结构

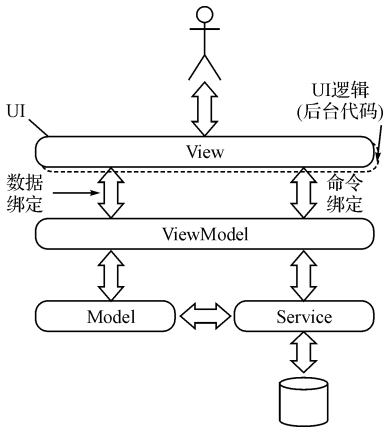

图 1 MVVM 模式结构

* 收稿日期:2014 年 4 月 3 日,修回日期:2014 年 5 月 24 日
作者简介:陈涛,男,工程师;研究方向:信息网络通信。

MVVM 模式下，用户与 View 交互，View 与 ViewModel 通过数据和命令绑定交互，ViewModel 与 Model 以及其他服务交互，Model 同样可以与其他服务交互。可以用图 1 简要表示[3]。

2.2 View 模块分析

View 即视图，是 MVVM 设计模式中用来定义用户界面和存放其逻辑的部分。

理想状况下，所有用户界面和 UI 逻辑都在 XAML 文件中定义，View 的后台代码中只有一个默认的构造函数，如下所示：

```
using System. Windows;
namespace WPFApplication
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

这种情况下，我们甚至可以删除 View 的后台代码文件。但是，一些比较复杂的 UI 逻辑很难在 XAML 文件中表示，这时我们可以将 UI 逻辑放入 View 的后台代码中。

在 MVVM 中，ViewModel 被放入 View 的 DataContext 中，这样 View 就可以绑定到 ViewModel 中的属性和命令。View 还可以对绑定的 ViewModel 的数据进行一些转换后显示。

一般一个 View 对应一个 ViewModel[4]。

2.3 ViewModel 模块分析

ViewModel 视图模型是 MVVM 设计模式中用来存放显示逻辑和状态的类，它为视图封装了展示逻辑。

在.NET 开发中，另一种比较常用的界面设计模式是 MVP（Model-View-Presenter）。ViewModel 和 MVP 中的 Presenter 的作用相似，但是不同于 Presenter，ViewModel 没有 View 的引用，也就是说 ViewModel 并不知道 View 的存在。ViewModel 与 Model 之间的通信使用 Data Binding 和 Command 技术，ViewModel 通过事件通知 View 其状态的改变。

ViewModel 负责协调 View 与 Model 之间的数据传递，ViewModel 可以选择直接将 Model 传递给 View，或者对 Model 中的数据传递给 View。

ViewModel 的另一个功能是存放状态。例如，ViewModel 可以保存一个正在下载的状态，View 可以使用进度条绑定到正在下载状态，显示一个正在下载动画。

ViewModel 可以通过实现 INotifyPropertyChanged 或 INotifyCollectionChanged 接口来支持属性或集合改变事件通知，通过实现 IDataErrorInfo 或 INotifyDataErrorInfo 接口来支持数据验证。

2.4 Model 模块分析

Model 模型是 MVVM 设计模式中用来存放数据和业务逻辑的类。业务逻辑定义了一系列业务规则以确保数据的持久有效，最大化实现了代码重用。

Model 不能包含任何特定的情况，通常是现实对象的抽象，例如一个用户，一个产品，或者一个订单。

同 ViewModel 一样，Model 也可以通过实现 INotifyPropertyChanged 或 INotifyCollectionChanged 接口来支持属性或集合改变事件通知，通过实现 IDataErrorInfo 或 INotifyDataErrorInfo 接口来支持数据验证。

模型也能通过 IDateErrorInfo 接口支持数据验证以及错误报告。这些接口允许当数值改变时的的数据绑定被通知到，从而完成界面的更新。

3 实现 MVVM 的其他相关技术

3.1 DataBinding 技术

WPF 中的 Data Binding 技术，使得 MVVM 成为非常优秀的界面设计模式。通过绑定，我们解耦了 View 和 ViewModel，并且不用在 ViewModel 中直接更新 View。

View 中的 UI 控件绑定到 ViewModel 的属性上，通过 ViewModel 得到 Model 里的数据。当 ViewModel 属性的值改变时，新值将通过绑定传递给 View。

Data Binding 支持多种模式，其中单向绑定使 UI 控件从 ViewModel 的属性上得到数据，而双向绑定除完成单向绑定的功能外，还会在用户通过 UI 控件修改数据后，自动更新 ViewModel 的属性值。

ViewModel 需要实现 INotifyPropertyChanged 接口来通知 View 属性值的改变，如下所示：

```
public class MainViewModel : INotifyPropertyChanged
{
    private int myVar;
    public int MyProperty
    {
```

```
        get { return myVar; }
        set
        {
            myVar = value;
            OnPropertyChanged("MyProperty");
        }
    }
    public event PropertyChangedEventHandler PropertyChanged;
    private void OnPropertyChanged(string propertyName)
    {
        PropertyChangedEventHandler handler = this.PropertyChanged;
        if (this.PropertyChanged != null)
        {
            handler(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

如果 ViewModel 是一个集合类型,则需要实现 INotifyCollectionChanged 接口,或者使用更简单的方法,继承 ObservableCollection〈T〉类。另外,通常可以使用一个 ObservableCollection〈T〉类型的属性来支持 Data Binding。

当需要编程对集合进行筛选、排序,或者需要知道用户选中了哪条记录时,ObservableCollection〈T〉类型就不能满足要求了。WPF 提供了 ListCollectionView 类来完成以上工作。例如,可以使用以下代码,在用户选中的项目改变时,进行相关操作:

```
public class MainViewModel : INotifyPropertyChanged
{
    private ICollectionView MyModels { get; private set; }
    public MainViewModel(List〈MyModel〉 models)
    {
        MyModels = new ListCollectionView(models);
        MyModels.CurrentChanged += new System.EventHandler(MyModels_CurrentChanged);
    }
    void MyModels_CurrentChanged(object sender, System.EventArgs e)
    {
        var model = MyModels.CurrentItem as MyModel;
        //相关操作
        …
    }
    …
}
```

3.2 Command 技术

Command 是 MVVM 模式中另一个非常重要的技术,用来执行用户的操作。在 MVP 模式中,用户操作是通过 UI 事件来响应的,这样耦合度较高,而 Command 解耦了 View 和 ViewModel。

Command 对象需要实现 ICommand 接口。在 ICommand 接口中定义了两个方法,Execute 和 CanExecute。Execute 方法执行 Command 的操作,而 CanExecute 方法判断 Command 是否可执行,并可根据结果控制绑定的控件是否可用。通常我们不用自己去实现 ICommand 接口,而可以使用一些已有的 Command 类,例如 Epression Blend SDK 中提供的 ActionCommand,或 Prism 中提供的 DelegateCommand。ActionCommand 和 DelegateCommand 的具体使用方法超出本文的范围,这里不做介绍。

在 ViewModel 中定义好 Command 对象后,可以使用如下代码在 View 的控件中绑定 Command:

```
〈Button Content="保存" Command="{Binding Path=SaveCommand}" /〉
```

上例中将保存按键的 Command 绑定到 SaveCommand,当用户单击按键时,Command 将执行。

对于实现了 ICommandSource 接口的 UI 控件,都可以使用 Command 属性进行绑定。但是,当需要绑定 Command 到没有实现 ICommandSource 接口的 UI 控件,或者需要绑定到默认事件以外的其它事件上时,则需要使用 Expression Blend 的 Interaction Triggers 和 InvokeCommandAction Behavior[5]。

3.3 View 与 ViewModel 的调用关系

ViewModel 作为 View 的抽象,负责 View 与 Model 之间信息转换,将 View 的 Command 传送至 Model 实现数据访问。其调用关系如图 2 所示。

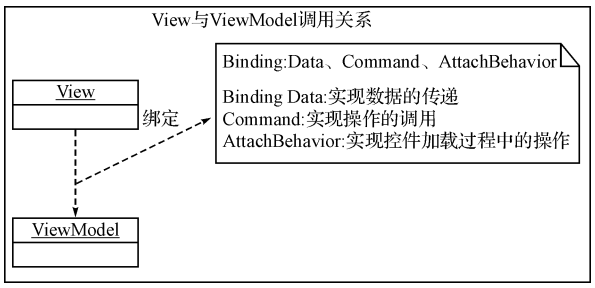

图 2 View 与 View Model 调用关系

4 结语

本文对 MMVM 设计模式的概念及使用方法进行了介绍。可以看出，MVVM 设计模式非常容易使用，但需要指出的是，并不是所有 WPF、Silverlight 程序都必须使用 MVVM。MVVM 比较适合在具有一定规模的企业应用中使用。我们完全没有必要在一个非常简单，只有一个或几个功能的小程序中使用 MVVM 设计模式。在实际应用中，需要根据需求来决定是否使用 MVVM。

最后，我们应该明白，所有的设计模式都只是对程序开发的指导，是对特定问题的一些好的解决方法，但是没有任何一种设计模式能够解决所有问题。因此在实际应用中我们需要根据实际情况来选择是否使用以及如何使用设计模式，从而创造出更加优美、易于维护的应用程序[6]。

参 考 文 献

[1] 曾蔚. 基于 Silverlight 的下一代可视化商业智能系统研究[J]. 电脑知识与技术，2010(19).
ZENG Wei. The Study of the Next Generation Business Intelligence System Based on Sliverlight[J]. Comuter Knowledge and Technology，2010(19).

[2] 李猛坤，陈明. 一种基于扩展 MVVM 模式的面向服务软构件模型[J]. 科学技术与工程，2011(10).
LI Mengkun，CHEN Ming. A Model of Service-oriented SoftWare Component Based on Extended MVVM Pattern. Science Technology and Engineering，2011(10).

[3] 陈明，李猛坤，张强. 一种基于扩展 MVVM 模式的 SaaS 面向服务计算模型[J]. 微电子学与计算机，2010(8).
CHEN Ming，LI Mengkun，ZHANG Qiang. Service-oriented Computing Model for SaaS with Extended MVVM Pattern，2010(8).

[4] Charles PetZold. WPF 程序设计指南[M]. 北京：电子工业出版社，2008：455-487.
Charles PetZold. Windows Presentation Foundation [M]. Beijing：Publishing House of Electronics Industry，2008：455-487.

[5] 温昱. 软件架构设计[M]. 北京：电子工业出版社，2007：217-246.
WEN Yu. Software Architecture Design[M]. Beijing：Publishing House of Electronics Industry，2007：217-246.

[6] 沙洛韦，特罗特. 设计模式精解[M]. 北京：清华大学出版社，2004：143-155.
Alan Shalloway，James R. Trott. Design Patterns Explained[M]. Beijing：Tsinghua University Press，2004：143-155.

(上接第 1948 页)

[4] 陈全，邓倩妮. 云计算及其关键技术[J]. 计算机应用，2009，29(9)：2562-2567.
CHEN Quan，DENG Qianni. Cloud computing and its key techniques[J]. Journal of Computer Applications，2009，29(9)：2562-2567.

[5] 王德文，宋亚奇，朱永利. 基于云计算的智能电网信息平台[J]. 电力系统自动化，2010，34(22)：7-12.
WANG Dewen，SONG Yaqi，ZHU Yongli. Information platform of smart grid based on cloud computing [J]. Automation of Electric Power Systems，2010，34(22)：7-12.

[6] 畅广辉，镐俊杰，刘涤尘，等. 基于多代理技术的电力控制中心综合数据平台设计[J]. 电力系统自动化，2008，32(1)：85-89.
CHANG Guanghui，GAO Junjie，LIU Dichen，et al. Design of the integrated data platform in electric power control center based on mutilagent technology[J]. Automation of Electric Power Systems，2008，32(1)：85-89.

[7] 许丞，刘洪，谭良. Hadoop 云平台的一种新的任务调度和监控机制[J]. 计算机科学，2013，40(1)：112-117.
XU Cheng，LIU Hong，TAN Liang. New Mechanism of Monitoring on Hadoop Cloud Platform[J]. Computer Science，2013，40(1)：112-117.

[8] 郑湃，崔立真，王海洋，等. 云计算环境下面向数据密集型应用的数据布局策略与方法[J]. 计算机学报，2010，33(8)：1472-1480.
ZHENG Pai，CUI Lizhen，WANG Haiyang，et al. A Data Placement Strategy for Data-Intensive Applications in Cloud[J]. Chinese Journal of Computers，2010，33(8)：1472-1480.

[9] 尹芳，冯敏，诸云强，等. 基于开源 Hadoop 的矢量空间数据分布式处理研究[J]. 计算机工程与应用，2013，49(16)：25-29.
YIN Fang，FENG Min，ZHU Yunqiang，et al. Research on vector spatial data distributed computing using Hadoop projects[J]. Computer Engineering and Applications，2013，49(16)：25-29.

[10] 毕睿华，杨志超，王玉忠. 基于多智能体 SOA 模型的电力系统信息集成的应用研究[J]. 电力系统保护与控制，2010，38(7)：63-68.
BI Ruihua，YANG Zhichao，WANG Yuzhong. Studies on information integration of MAS-based SOA model in power system[J]. Power System Protection and Control，2010，38(7)：63-68.