

Silverlight 下的 MVVM 模式的应用

李龙澍, 华晓飞

(安徽大学 计算机科学与技术学院, 安徽 合肥 230601)

摘要: 为了改善传统开发方式和应用 RIA (Rich Internet Application, 富互联网应用程序) 开发技术, 文中应用了一种新的设计模式-MVVM, 提出了一套基于 MVVM 模式的系统开发方案。介绍了 MVVM 模式设计思想, 阐述了系统的设计, 以实际流程为例, 在 Silverlight 开发环境下给出了具体的实现以及代码比对分析。实践表明, 将该模式应用于系统开发, 可以成功地将数据、表示和业务逻辑分离, 有效地改善系统开发, 使系统具有结构清晰、可拓展、易测试的特点。

关键词: 富互联网应用程序; Silverlight; 设计模式; 模型-视图-视图模型

中图分类号: TP302.1

文献标识码: A

文章编号: 1673-629X(2013)12-0203-05

doi:10.3969/j.issn.1673-629X.2013.12.049

Application of MVVM Pattern in Silverlight

LI Long-shu, HUA Xiao-fei

(College of Computer Science and Technology, Anhui University, Hefei 230601, China)

Abstract: In order to improve the way of traditional development and apply the RIA development technologies, a new design pattern, MV-VM, is used, a development plan based on it is presented. Introduce the design idea of this pattern, and then give an implementation in Silverlight and comparative analysis of the code by using an actual process as example. Practice shows that by using this pattern in the system development, the data, display and logical transaction can be successfully divided, system can be improved effectively, and has the characteristics of clear structure, stability and ease to be tested.

Key words: RIA; Silverlight; design pattern; MVVM

0 引言

新一代的 RIA 具有快速、低成本以及丰富用户体验的特点, 正在发展成为 Web 应用领域开发的核心^[1]。与传统的 Web 应用程序相比, RIA 的客户端更“Rich”: 大部分的处理任务被从用户界面移植到客户端, 从而提高了响应速度, 可以提供更多的数据模型来操作数据, 使用更丰富的元素来呈现界面。RIA 的这一特点对于系统开发提出了以下要求: 首先, 需要一个层次来包含分离出来的业务逻辑, 同时要保证业务逻辑模块的可拓展性和独立性来满足项目更改的需求和保证开发的质量; 其次, RIA 技术更加追求实现丰富用户体验的用户界面^[2], 视觉设计被重视, 项目团队需要划分出视觉设计团队, 让其与开发团队进行分工协作; 最后, RIA 程序的逻辑复杂于传统互联网应用程序, 需要进行大量的测试, 这就要求编写出来的代码要

易于测试。

针对以上几点要求, 项目开发的系统需要具有松散耦合、界面功能分离和可测试性的特点。一些设计模式, 如 MVC^[3]、MVP^[4]、MVVM 等被提出并应用于系统开发过程中, 这些设计模式的实现虽然要花费额外的工作, 但设计后的系统的结构层次化, 系统功能模块化, 整体系统结构清晰, 功能明确^[5-6]。

文中主要讨论新型设计模式——MVVM 在使用 RIA 富客户端开发技术 Silverlight 开发的系统的设计与应用。下面将论述这种新型模式及其应用系统的设计与实现。

1 MVVM 模式

MVVM (Model-View-ViewModel, 模型-视图-视图模型) 模式是一种 .NET 框架^[7]下的设计模式, 其出现

收稿日期: 2013-02-25

修回日期: 2013-05-28

网络出版时间: 2013-09-29

基金项目: 安徽省农林科研项目 (06090736); 安徽省大学研究生学术创新项目 (yqh100135)

作者简介: 李龙澍 (1956-), 男, 教授, 博士生导师, 研究方向为软件设计技术、智能信息处理和 Agent 应用技术等; 华晓飞 (1990-), 男, 硕士研究生, 研究方向为软件测试。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20130929.1521.009.html>

的目的是解决 WPF 应用系统的架构模式的设计问题^[8],可应用于 WPF 和 Silverlight 项目开发中。

MVVM 模式由 Model(模型)、View(视图)和 View-Model(视图模型)三个部分组成。Model,模式架构的底层,实现对数据的封装;View,与用户交互,提供给用户控件及视觉呈现;ViewModel,MVVM 模式的核心层,介于 Model 层和 View 层之间,主要任务包括封装 Model 的数据类和完成对 View 中控件事件对应的业务逻辑处理,展示了 Model 中的数据和 View 的特定状态。

运行过程:用户的一次操作就可实现各层之间的通信,比如当用户点击 View 对象中的 Button 控件(Button 上的属性 Command 已绑定到 ViewModel 类的某条命令上)时,通过绑定机制,ViewModel 对象中的某条命令将会运行来执行请求的操作。进而完成对封装的 Model 类中数据的修改操作。修改过程中,如果 ViewModel 对象上的某个属性值发生了改变,同样通过访问数据绑定,该新值将被传送到 View 对象中更新呈现。

MVVM 模式是由 MVP 模式演变而来^[9],其与 MVC、MVP 模式的区别在于 View 类中没有复杂的代码逻辑,MVVM 模式主张整个开发过程中的 View 设计与其后台代码编写完全分开,这样的特性使得 MVVM 模式具有以下优势:

具有复用性。MVVM 模式中,ViewModel 被看作是 View 的数据上下文,多个 View 对象中的不同控件属性可以绑定一个 ViewModel 对象中的属性,其中有属性值发生更改时,绑定机制可以访问和更新所有参与绑定的 View 和 ViewModel 对象中的属性值。

各层之间通过 Silverlight 和 WPF 中独特的数据绑定和模板建立联系,彼此之间互相透明^[10]:View 类并不知晓 Model 类的存在,同时 ViewModel 和 Model 也不了解 View 的详细情况,甚至 Model 完全不了解 View-Model 和 View 存在的事实。

业务逻辑易于测试。MVVM 模式中,一个应用程序的交互逻辑都被包含在 ViewModel 类中,你可以轻松地编写测试它们的代码。View 对象和单元测试代码是访问 ViewModel 对象的两种不同形式。可以独立地为应用程序的 ViewModel 类设计测试套件,进行广泛快速的回归测试,从而降低维护程序的开支。

有助于改进视觉设计。MVVM 模式中,将一个 View 对象移出工作空间,用另一个 View 对象来呈现 ViewModel 对象并不需要对 ViewModel 类中代码进行大幅度的修改,这个优势使得快速设计原型来改进 View 变成了可能。

团队分工明确。由于彼此工作域是互相不可见的,因此开发团队和视觉团队可以各司其职;开发团队

专注于创建健壮的 ViewModel 类,视觉设计团队专注于建立用户友好的 View。随后只需编写正确的绑定代码就可以将两个团队的产品联系起来。

2 系统设计

文中开发的信息系统是基于 Web 的使用 Silverlight 技术开发的省公益林信息系统,负责管理全省范围内的公益林及相关信息。系统功能图如图 1 所示。

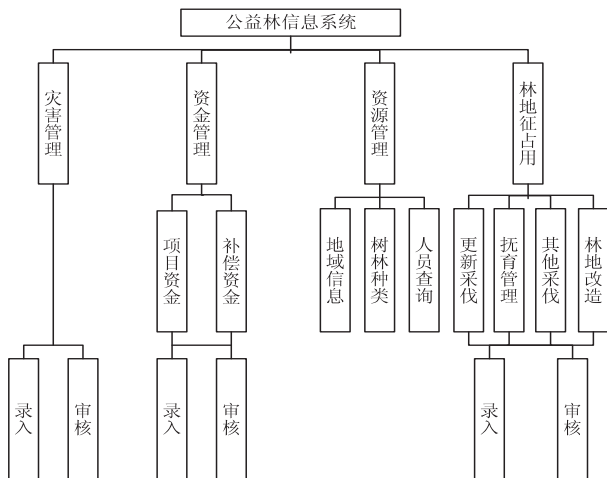


图 1 公益林信息系统功能图

从系统功能图中可以看出该系统具有以下特点:模块众多。该系统包含灾害、资金、资源和林地征占用四个主模块。其中,资金、资源、林地征占用又会细分为不同种类子模块。再按照对信息处理的方式(录入或者审核)进行进一步的功能划分。最终得到了 17 个独立的功能模块。

同一主模块的功能子模块的业务逻辑具有相似性。如资金管理下项目管理、资金发放;林地征占用下的更新采伐、其他采伐等,都包含录入、审核的业务流程,而且同一主模块下的功能子模块对于数据的处理方式(比如该录入什么样的信息作为记录,如何对录入的记录进行审核修改,如何对审核后的记录进行批准回复)大致相同。

针对以上特点,该系统的基于 MVVM 模式的架构如图 2 所示。

系统架构图的说明:

View 层:按照系统功能架构显示的层次构建模块化的 View 类,同层次 View 类相互独立,上下层次的 View 类之间通过两种方式进行通信:事件响应(如主 View 对象通过 Button 控件的 Click 事件打开子 View 对象的窗体)和嵌入为自身的元素(如在主 View 类的 XAML 代码中,添加一个类型为 UserControl 的子 View 对象)。

用层次模块化的方法构建 View 层的优势在于,同层次的 View 类之间的独立性保证不同的功能模块可

以由不同的团队开发、测试和维护。另一方面,层次性保证当系统需求发生变化,需要添加新的模块时,只需要将该模块的 View 类挂接到所属的主 View 类下,减少了添加代码的工作量和避免了同层之间的冲突。

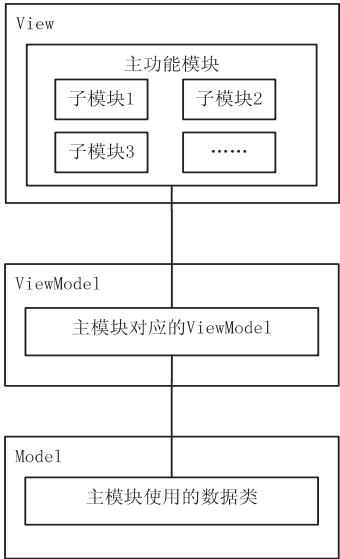


图 2 基于 MVVM 模式设计的系统架构图

ViewModel 和 Model 层: ViewModel 类中封装了 Model 中的数据类,同时包含了对 View 的业务逻辑的处理代码。系统中,同一主模块下子功能模块的业务逻辑具有相似性,因此在 ViewModel 层中,建立了四个与 View 层中四个主模块类对应的 ViewModel 类。至于建立了四个 ViewModel 类而不只使用一个的原因也是很明显的,Model 层中各个主要功能模块所使用的数据类不相同,如果只使用一个 ViewModel 类来完成对所有数据类的封装,会使得这个类很臃肿,同时不同的业务逻辑也被包含进来,影响了代码的可读性,有可能造成编写代码混乱的情况。

3 实现及示例应用

3.1 实现与示例

以下因素是 MVVM 模式实现的重要条件:

数据绑定机制。数据绑定机制保障了层次之间的通信。通过 View 中用 XAML 描述的绑定语句和数据上下文语句建立 View 层和 ViewModel 层的联系,绑定系统将会构建和实现 View 层和 ViewModel 层的通信。

命令(Command)。按照 MVVM 模式的设计原则,View 层中不应该包含业务逻辑,因此每一个 View 类的后台代码文件中,除了类构造器中样板代码 InitializeComponent 以外,基本没有什么处理事件的方法。但当用户与 View 对象上的控件交互,比如点击 Button 控件时,程序要能够反应和满足用户需求。这些控件功能实现的保证归功于建立于 View 类中控件(主要有 Hyperlink, Button, MenuItem)上 Command 属性的绑定。

这些绑定的 Command 属性确保当用户点击控件时,ViewModel 类对应的继承 ICommand 接口的对象将执行其封装的业务逻辑的语句。

结合以上因素,ViewModel 层中包含的类图如图 3 所示。

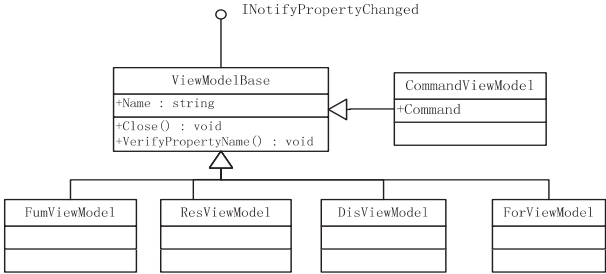


图 3 ViewModel 层 UML 类图

类图说明:

共同基类。ViewModel 层中并不只定义了四个主模块的 ViewModel 类,这四个类都继承自基类 ViewModelBase,ViewModelBase 类中包含了一些 ViewModel 类的基本功能代码:实现通知属性更改的 INotifyPropertyChanged 接口和移除对应 View 对象的 Close 方法。

CommandViewModel 类:另外还添加了一个 CommandViewModel 类,该类封装了一个继承 ICommand 接口的 Command 对象,用于给其他的 ViewModel 类提供一组命令。

下面以灾害审核流程为例,描述模式的具体实现。灾害审核实现的功能是当灾害信息录入后,工作人员通过对市、县、灾害类型的选择来筛选灾害记录,并勾选标记来完成对灾害记录的审核。

下面按照 MVVM 的层次来阐述各层相关工作:

Model 层:建立了 3 个相关的数据类,代码中包含了代表数据字段的属性和操作数据的方法(见图 4)。

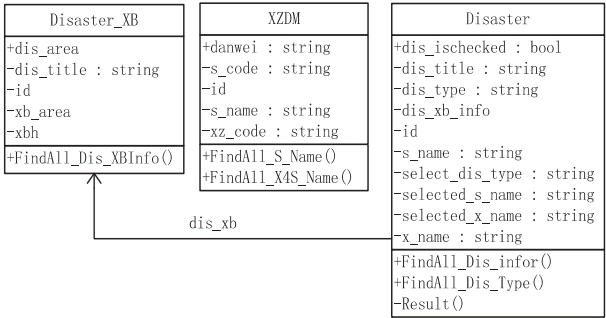


图 4 Model 层 UML 类图

View 层:

Dis_CheckView 类主要工作:

- (1) 实例化 DisViewModel 类,作为数据上下文(DataContext)。
- (2) 绑定控件属性并设定绑定模式,绑定模式有 OneTime、OneWay 和 TwoWay 三种,设定的依据为数据流方向和触发源更新的原因。

参与绑定的控件属性信息如表 1。

表 1 绑定控件属性信息表

控件类型	参与绑定属性和模式	绑定对象
ComboBox	SelectedItem (TwoWay)	selected_s_name, selected_x_name, selected_distype
	ItemSource (OneWay)	s_name, x_name, dis_type
DataGrid	ItemSource (TwoWay)	disinforlist, dis_xb
DataGridTextColumn	Binding (TwoWay)	dis_title, dis_type, xbh, dis_area, xb_area
DataGridCheckBoxColumn		dis_ischecked
Button	Click 事件 (TwoWay)	makesureCommand 命令, closeCommand 命令

其中,用户通过对 DataGridCheckBoxColumn (表示一个 DataGrid 列,该列在其单元格中承载 CheckBox 控件)列进行勾选,实现对灾害记录的审核。

ViewModel 层:

其中包含的 DisViewModel 类的主要工作包括:

- (1)封装了 Model 中的数据类的对象。
- (2)对 OnPropertyChanged 方法进行重载:

```
// DisViewModel.cs
///重载该函数做必要工作
protected override void OnPropertyChanged ( string propertyName ) {
    base.OnPropertyChanged ( propertyName );
    switch ( propertyName ) {
    case "selected_s_name" :
        //通知相关属性的更新
        OnPropertyChanged ( "x_name" );
        OnPropertyChanged ( "dis_inforlist" );
        break;
        ..... } }
```

之所以重载 OnPropertyChanged 方法,是因为 View 中控件属性之间存在联系,有时一个控件属性被用户或者系统更改,其他的一个或几个属性值也需要更新。比如,当用户通过 S_NameCombo 下拉框选择了某个市名,OnPropertyChanged (selected_s_name) 方法会执行 (因为 selected_s_name 值更改), 同时, X_NameCombo 应该要显示该市下的所有县名, Dis_InforDataGrid 要显示该市的所有灾害记录,因此这两个控件的数据源 (ItemSource) 绑定的 x_name 和 dis_inforlist 都需要执行 OnPropertyChanged 方法来更新值。

(3)封装了 ICommand 对象:

```
// DisViewModel.cs
///保存命令
private ICommand _makesureCommand;
public ICommand makesureCommand {
    get {
        if ( _makesureCommand == null )
```

```
//通过实例化继承自 ICommand 接口的 DelegateCommand 类来指定封装的包含业务逻辑的私有方法作为委派[11]
```

```
{ _makesureCommand = new DelegateCommand ( e => this.
Save(), e => true ); }
return _makesureCommand; } }
///关闭命令
private ICommand _closeCommand;
public ICommand closeCommand {
    get {
        if ( _closeCommand == null )
            { _closeCommand = new DelegateCommand ( e => this. Close
            ( ), e => true ); }
        return _closeCommand; } }
```

封装的私有方法代码如下:

```
private void Save() {
    _disaster. SaveChange(); }
public override void Close() {
    base. Close(); }
```

Save 方法中调用了保存更改记录审核状态 (dis_ischecked 属性值) 并写回数据库的数据操作方法,完成了最后审核完成的业务逻辑。

Close 方法调用基类的方法来实现关闭 Dis_CheckView 的功能。

3.2 与传统开发方式下所得代码的比较分析

传统开发方式对界面控件事件进行响应,在后台代码文件在挂接的方法中编写代码来实现业务逻辑,因此传统开发方式下 Dis_Check. xaml. cs 文件会添加相关的方法代码。

将使用 MVVM 设计模式开发的代码与不使用 MVVM 模式的传统开发方式得到的代码进行比较,如表 2 所示。

表 2 使用 MVVM 设计模式开发的代码与传统模式开发代码比较表

开发方式	层次名	涉及文件	添加的方法个数
MVVM	Model	Disaster. cs	3
		Disaster_XB. cs	1
		XZDM. cs	2
	ViewModel	DisViewModel. cs	3
传统	界面与业务逻辑	Dis_Check. xaml. cs	9

相比于传统开发方式,使用 MVVM 模式开发得到的文件数虽然较多,但是每个文件所包含的方法数比较适中,单个文件不会显得很臃肿,文件易于阅读,有利于控制代码更改。同时使用不同的层次中的类封装不同用途的方法,彼此之间通过对象方法调用的方式进行联系,也体现了封装和高内聚、低耦合的特性,易于新模块的拓展。

测试方面, MVVM 模式开发的 ViewModel 类具有可测试性,可独立参与到测试中 (代码如下), 相比于

传统开发方式,测试代码中不需要包含涉及界面的代码,减少了测试的工作量。

```
//DisViewModelTest.cs
[TestMethod]
public void TestDisViewModel() {
//只实例化 DisViewModel
DisViewModel target = new DisViewModel();
Assert.AreEqual(0, target.s_name.Count, "市名列表不能为空");
Assert.AreEqual(0, target.dis_type.Count, "灾害类型列表不能为空");
..... }}
```

4 结束语

文中依托新型 RIA 富客户端开发技术 Silverlight,将新型的 MVVM 设计模式应用到系统开发中,并且以实际流程为例进行了实现和分析。应用结果表明,MVVM 模式将系统中的表示和业务逻辑进行了层次划分和强分割,有效地提高了开发和维护效率,避免了开发时的混乱情况,值得应用和推广。

参考文献:

[1] Lawton G. New ways to build rich internet applications[J]. IEEE computer,2008,41(8):10-12.
[2] 孙超,钟路.基于 Silverlight 的富界面应用研究[J].武

(上接第 202 页)

参考文献:

[1] Pecora L M, Carroll T L. Synchronization in chaotic systems [J]. Phys rev lett,1990,64:821-830.
[2] 方锦清.非线性系统中混沌控制方法、同步原理及其应用前景(二)[J].物理学报,1996,16(2):137-159.
[3] 刘崇新.分数阶混沌电路理论及应用[M].西安:西安交通大学出版社,2011.
[4] Li Changping, Peng Guojun. Chaos in Chen's system with a fractional order[J]. Chaos, solitons and fractals,2004,22(2):443-450.
[5] Grigorenko I, Grigorenko E. Chaotic dynamics of the fractional Lorenz system [J]. Physical review letters, 2003, 91(3):034101.
[6] Lu Junguo. Chaotic dynamics of the fractional order Lü system and its synchronization[J]. Physics letters A,2006,354(4):305-311.
[7] 曹鹤飞,张若洵.基于单驱动变量分数阶混沌同步的参数调制数字通信及硬件实现[J].物理学报,2012,61(2):123-130.
[8] 李志军,孙克辉,任健.分数阶统一混沌系统的耦合同步研究[J].新疆大学学报(自然科学版),2011,28(2):127-

汉理工大学学报,2008,30(12):95-97.
[3] Leff A, Rayfield J T. Web-application development using the model/view/controller design pattern [C]//Proceedings of fifth IEEE international conference on enterprise distributed object computing. [s. l.]:[s. n.],2001:118-127.
[4] Potel M. MVP: Model-view-presenter the Taligent programming model for C++ and Java[R]. [s. l.]: Taligent Inc., 1996.
[5] 张晓丽,路杨.基于 MVC 模式的 Web OA 系统的设计与实现[J].计算机技术与发展,2012,22(8):63-66.
[6] 邓志宏,张智,李建奇,等.基于 MVP 模式的进销存系统的软件架构设计[J].计算机与数字工程,2012,38(12):96-99.
[7] 艾迪明. .NET 框架体系结构[J].计算机工程与应用,2003(2):174-176.
[8] 程国雄,胡世清.基于 Silverlight 的 RIA 系统架构与设计模式研究[J].计算机工程与设计,2010,31(8):1706-1709.
[9] 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,2010,27(8):27-30.
[10] Smith J. WPF apps with Model-View-ViewModel design pattern[EB/OL]. 2008. <http://msdn.microsoft.com/en-us/magazine/dd419663.aspx>.
[11] Noyes B. Understanding routed events and commands in WPF [EB/OL]. 2008. <http://msdn.microsoft.com/en-us/magazine/cc785480.aspx>.

131.
[9] 张若洵,杨世平,刘永利.基于线性控制的分数阶统一混沌系统的同步[J].物理学报,2010,59(3):1549-1553.
[10] 马铁东,江伟波,浮洁,等.一类分数阶混沌系统的自适应同步[J].物理学报,2012,61(16):90-95.
[11] 朱少平,钱富才.超混沌系统同步非线性反馈控制[J].计算机工程与应用,2011,47(1):50-52.
[12] Caputo M. Linear models of dissipation whose Q is almost frequency independent-II[J]. Geophysical journal of the royal astronomical society,1967,13(5):529-539.
[13] 胡建兵,韩焱,赵灵冬.分数阶系统的一种稳定性判定定理及在分数阶统一混沌系统同步中的应用[J].物理学报,2009,58(7):39-44.
[14] 何万生,杨丽新,刘晓君.一类分数阶混沌系统的耦合同步[J].河北师范大学学报:自然科学版,2012,36(6):569-672.
[15] 章婷芳,姚洪兴,耿霞. Chen 混沌系统的反馈控制方法与分析[J].系统工程理论与实践,2005(8):97-102.
[16] 谢丽君.基于相干 CSK 技术的数字通信系统的分析研究[D].南京:南京理工大学,2012.

Silverlight下的MVVM模式的应用

作者：[李龙澍](#)，[华骁飞](#)，[LI Long-shu](#)，[HUA Xiao-fei](#)
作者单位：[安徽大学 计算机科学与技术学院, 安徽 合肥, 230601](#)
刊名：[计算机技术与发展](#)

英文刊名：[Computer Technology and Development](#)

年，卷(期)：2013(12)

本文链接：http://d.g.wanfangdata.com.cn/Periodical_wjfz201312049.aspx