Advances in Computer Science Research, volume 71
4th International Conference on Machinery, Materials and Information Technology Applications (ICMMITA 2016)

The Exploration and Practice of MVVM Pattern on Android Platform

Wei Sun[1, a], Haohui Chen[1, b] and Wen Yu[1, c]

[1] College of Information Technology, Beijing Normal University Zhuhai, Zhuhai, Guangdong 519085, China

[a]asunwei@bnuz.edu.cn, [b]moon4chen@163.com,[c]wenyu@bnuz.edu.cn,

Keywords: MVVM; RSS; Data Binding; Android

Abstract. How UI designers and business engineers collaborate to development apps on Android platform conveniently and effectively is a difficult problem to tackle when projects get more complicate and need more effort. This paper compared the commonality and variability of MVVM with the commonly used MVC and MVP patterns. An RSS subscription app was designed and implemented by using Data Binding and Rome.jar techniques and the use of MVVM pattern on Android platform was explored. Three key points to solve the problem of bidirectional binding of views and view models were described. Decoupling of Model and View further separated data, logic and view and satisfied the requirements of different format of views for the same model. Therefore, duplicated code was reduced. The more important is that the coupling level of code was decreased for multiple developers. The software design objective of "high cohesion and low cupping" was achieved and efficiently collaborative development was accomplished.

Introduction

With the increase of the scale and the complexity of software to be developed, the modular design principle of "high cohesion and low coupling" [1] has been the goal for researchers and developers to explore and put into practice. The MVC, MVP and MVVM are three representative models of the MV+X model. These models are common design patterns of the application system and they realize the different degrees of separation of data, logic and view [2]. This paper analyzed and compared the different working mechanisms of the above three models and explored the use of MVVM model under the Android Operating System (OS) by illustrating the key design points of a RSS subscription app.

Driven by the commercial interests, the environment of getting information has become harsher now. More and more domestic and foreign websites have provided the corresponding RSS services. Users can access the "clean" and targeted information by subscribing the RSS service of website.

Recently, mobile internet industry has gradually penetrated into every corner of our life, which is considered to be a competition in the Red Sea. Because mobile terminals are flow inlets, the data flow is particularly precious. RSS subscription app can greatly reduce the time and data traffic by avoiding users to browse and download a lot of useless information. By using the RSS subscription app service in the mobile terminal, one can get information more convenient and more efficient [3].

In addition to the traditional functions, an RSS subscription app on the Android platform, with RSS parsing, rich text reading, cloud storage and other functions, will bring users an exciting experience. The main innovative points are as follows:

- Optimizing functions of the RSS subscription app, highlighting the two attributes of the Internet thinking, that is the usability and personalized customizing;
- Adding cloud data storage capabilities;
- Exploring the application of MVVM mode on Android platform in the framework design, which makes the collaboration of interface design engineers and business engineers more convenient and more;
- Following the idea of Google Material Design in the prototype design and studying realization methods of the Design Material style;

- Using the network request framework (Volley), which supports queue requests and priority request processes and provides the caching mechanism;
- Using the lightweight event bus (TinyBus) and simplifying the communication between activities by the thread pool, the interaction between the background and the main thread;
- Using MBaaS services as a support for back-end services and reducing server development costs, so that the design and development are focused on the front end of the app.

Related technology analysis

RSS parsing. RSS（Really Simple Syndication）is a format for describing and synchronizing the web content [4]. All or part of the information released by the website can be integrated into an RSS file. This file is called the feed and data in the file is in the XML format [5]. RSS treats websites in different formats as the combination of channel. The structure of the feed document in the standard RSS2.0 specification is shown in Fig 1.

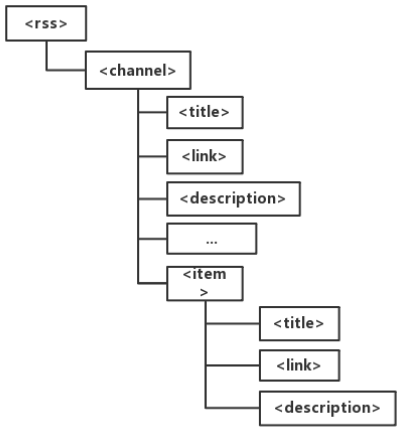

Figure 1 The structure of feed document in the RSS 2.0

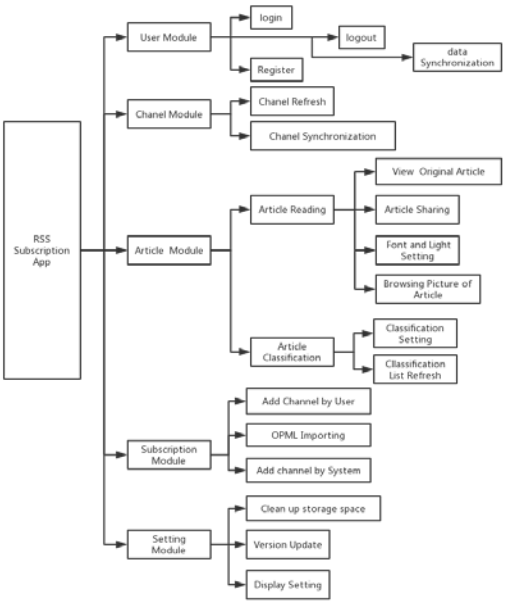

Figure 2 Function structure diagram of RSS subscription App

There are three commonly used XML parsing methods: DOM, SAX and PULL[6]. Due to the limitation of the mobile terminal's memory capacity, the parsing type of the SAX or PULL should be used to resolve the XML in the mobile terminal app[7]. Rome.jar is an excellent RSS parsing library under the Java platform. It uses the SAX to parse XML files [8]. Rome.jar parsing steps are described as follows. Scan the document in a sequential order. When the node is scanned, notify the event handler function to handle the event. The above process repeat until the parsing process is finished. [9] .

By using the method retrieveFeed (url) provided by Rome.jar, one can get the object of the SyndFeed class： *feed = getRetriever( ).retrieveFeed(url);*

Since local saved data may not be in the format of SyndFeed provided by the Rome.jar class, other properties, such as ID, users of the data, subscription time and update time need to be added. Therefore, it is necessary to convert the SyndFeed again to obtain the final Feed class. The structure of the Feed class is shown in Table 2. The conversion function for converting SyndFeed into Feed objects is shown in Table 3.

Table 1 The structure of SyndFeed Class

```
private ObjectBean _objBean;
private String _encoding;
private String _uri;
private SyndContent _title;
private SyndContent _description;
private String _feedType;
private String _link;
private List _links;
private SyndImage _image;
private List _entries;
private List _modules;
private List _authors;
private List _foreignMarkup;
private WireFeed wireFeed;
private boolean preserveWireFeed;
```

The structure of SyndFeed class is shown in Table 1。

Table 2 The structure of the Feed class

```
private String title;
private String url;
private Integer status;
    …
private String uri;
private String creator;
private long user_id;
```

Articles are in the SyndFeed List<SyndEntry>. It can be obtained by calling the syndFeed.getEntries() method. As the local data is more adequate, we also need to traverse the SyndEntrys and convert each SyndEntry to an Article object. The conversion function is shown in table 4.

Table 3 the conversion function for converting SyndFeed into Feed

```
public static Feed feedConert(SyndFeed syndFeed, long userID)
{
Feed feed = new Feed(
syndFeed.getTitle(),
…
//get icon
HtmlHelper.getIconUrlString(syndFeed.getLink()),
   syndFeed.getPublishedDate(),
   …
   ((DCModule) syndFeed.getModules().get(0)).getCreator(),
   userID);
   return feed;
}
```

MVC, MVP and MVVM. Software developments on the Android platform often use MVC (Model-View-Controller) or MVP (Model-View-Presenter) and other framework model [10]. The working mechanism of both MVC and MVP is shown in Fig 2.

In the MVC framework, the request procedure is shown as follows:

- The view accepts the user's request
- The view transfer the request to Controller
- The controller operation Model for data updates
- The model notice View and let it change
- The view updates the display information according to the updated data.
- The request process flow of the MVP framework is show as follows:
- The view accepts the user’s request
- The view transfer the request to Presenter
- The Presenter to complete the logic processing, and then modify the Model

- The model notice the presenter and let it change
- The Presenter update the view

Table 4 The conversion function for converting the SyndEntry to the Article

```
public static Article articleConvert(SyndEntry entry)
{
Article article = new Article(
        entry.getTitle(),
        …
        entry.getPublishedDate(),
        null,//recent reading time
        0,//state
        ((DCModule) entry.getModules().get(0)).getUri(),
        …);
       return article;
}
```

Compared to MVC, MVP uses Presenter to remove the coupling between View and Presenter. The presenter will be returned to the Model changes to View.

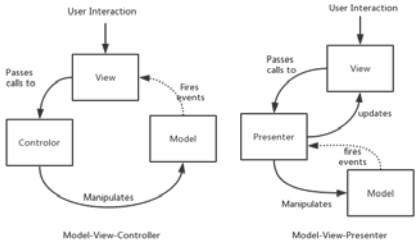

Figure 3 The working mechanism of MVC and MVP

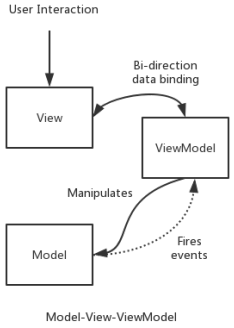

Figure 4 The working mechanism of MVVM

In the 2005, Microsoft's architect Gossman John proposed the MVVM (Model Model-View-View) model[11]. MVVM framework is a new framework for the formation of MVP (Model-View-Presenter) model and WPF combination. It is based on the original MVP framework, and the WPF of the new features are fused into, in order to cope with the growing complexity of the customer needs change. The working mechanism of MVVM is shown in 3, One of the most critical innovation lies in the Data Binding Technology. In 2015 Google I/O Developer Conference launched a data binding framework (Data Binding Library) [12]. The framework can realize the bidirectional binding of view and data, and complete the automatic binding between view and view data, so it can reduce the coupling degree between code quantity and role in the development. The final advantage of the framework to solve a major pain points in the AndroidUI development, and no longer need to write code like *findViewById*. More important is a greater degree of reduction in the degree of coupling between the view and the control module.

In the MVVM framework, the request processing process is similar to MVP, but in the last step, the interaction between View and ViewModel is done by Biding Data. Thus, the coupling of the view and the control module is reduced, and the pressure of the view is reduced.

Material Design. In 2014, at the Google I/O conference, the original design (Material Design) was launched [13].Material has 3 dimensional space, so each object has a x, y, Z 3D coordinate properties, and follow the physical characteristics of the object Material (thickness, shadow, cannot be penetrated, 2 objects cannot occupy the same space at the same time point, can be moved and retractable, etc.). This is a new visual design language follows the classic rule of good design, at the

same time with the concept of innovation and new technology, so as to solve the consistency problem of the previous Android platform design style [14]. Material Design combined with card design, and combined with the metaphor of the paper in the real world, in the end, unified the expression of Google in the design, so as to show a clear style.

BaaS（Backend as a Service）

A mobile application cannot be separated from the support of the back end of the service. If the number of mobile development team is too small, then in addition to consider the platform adaptation (Android, IOS) also must take into account the back-end erection, development, maintenance and other work, which increase the workload of developers. BaaS (as a Service Backend) is to help developers solve this problem. The MBaaS (Mobile Backend as a Service) provides integrated cloud services for application developers at the rear end of the border, such as data storage, account management, message or file push, social module, geographic location, advertising etc.[15]. Now BaaS has arisen, such as StackMob and Parse, Bmob, Talking Data, Baidu and Sina open platform, etc. they have provided BaaS services.

Network Request Framework（Volley）In 2013, the network communication framework that is, Volley was presented at the I/O Google conference[16]. Network requests can be similar to archery, the Volley can handle a large number of requests, similar to the "fire" in a short period of time, so Volley is suitable for fast, simple, a large number of requests, such as access to JSON data and network pictures, etc. [17].

Lightweight event bus(TinyBus) TinyBus is a lightweight, simple event bus TinyBus [18]. Activity and Fragment can submit events and respond to them. TinyBus can simplify the communication cost between Activities through the thread pool, and simplify the interaction between the background and the main thread.

The requirements of RSS subscription App

The main modules of RSS subscription App include: user module, channel module, article module, subscription module and setting module. Detailed functional structure is shown in Fig 4.

The application of MVVM model

Using Data Binding Library to realize the bidirectional binding of view and view model is the key point of using MVVM model in Android platform. Which ViewModel is responsible for the logic processing and control of the state of View, Model is responsible for the encapsulation of business data; View is only responsible for the display of dada. Below the Model layer includes a service layer, which is used for supporting the service. Realizing the bidirectional binding of the view and view model need to be further implemented in the following three key parts:

- View bound view model（View data Binding）
- The view sends commands to the view model（Commands）
- view model notification view to update(VM notify)

View Bound View Model(View data Binding). Use Data Binding dynamically generated a xxxBinding class (hereinafter referred to as the Binding class) in the XML file, the Binding class is a subclass of ViewDataBinding which is used to implement the view and data binding.

Through using setContentView (activity, layoutId) method provided by DataBindingUtil of Data Binding, that XML files can be converted into a View object, and set to activity ContentView, and returns an object of class Binding.For example, in a articleViewModelBinding object, by calling setArticleViewModel (articleViewModel) and writing to the ArticleViewModel. Through this method can complete the work of the generation and setting of View. Table 5 shows the use of ArticleViewModel objects in activity_article.xml.

Table 5 The Setting of the XML file

```
<layout.>
   <data><variable name="articleViewModel"
type=" ArticleViewModel" />
   </data>
...
<include layout="@layout/ bottom_bar"
bind:articleViewModel="@{articleViewModel}"/>..
</layout>
```

View Model Notification View to Update(VM notify). Data Binding provides three ways (Observable, ObservableFields, and collections observable) to notify the View to update the data when the bound data has changed, this article uses the Observable method.

In fact, ViewModel is not a pure POJO (Plain old Java Object), which contains a variety of additional operations, such as the View click event handling, database read and write network request etc.. The benefit is that the UI can be handed over to the ViewModel, which is similar to the implementation in the MVP mode, that is, the Action agent to Presenter.ViewModel

A variety of Model in the ViewModel is a typical POJO objects. All ViewModel inherited from the BaseObservable base class. BaseObservable provides two methods, namely (notifyChange) and notifyPropertyChanged (int fieldId), Notice all data is updated by using the two methods in the Binding view, and provides the Bindable annotation to achieve registered listeners.Table 6 shows the binding and notification of objects in Article View Model through Bindable.

Table 6 the binding and notification of objects in View Model

```
public class ArticleViewModel
extends BaseObservable{
   private Article article;
   @Bindable
   public Article getArticle() {
      return article;
   }
   public void setArticle(Article article) {
      this.article = article;
      notifyPropertyChanged(BR.article);
}
….
```

The View Sends Commands to The View Model(Commands). In general, Commands is generated by the user operation at the interface,however in the MVVM mode, Commands is passed to the View by VM, such as Click (click), sliding (Slide) and other operations.

Binding Data also provides a Event binding feature that can bind many of the events common in the system. As shown in Table 7, you can bind the onClick event of TextView to the corresponding onClickWebArticle method of VM in the XML.

Table 7 The binding events in the View

```
<IconTextView …
android:onClick="@{articleViewModel.onClickWebArticle}"
   …/>
```

Then write the response method in the corresponding articleViewModel, as shown in table 8.

Table 8 Writing the response method in the View Model

```
public void onClickWebArticle(View view) {
        //do some thing
}
```

The Application of MVVM Model in The Article List Module. Taking the design of the article list module as an example to demonstrate the application of MVVM design pattern, that is, feedActivity as View, FeedVM as ViewModel, ArticleAdapter, Feed Ariticle and Feed as Model. As shown in Fig 5.

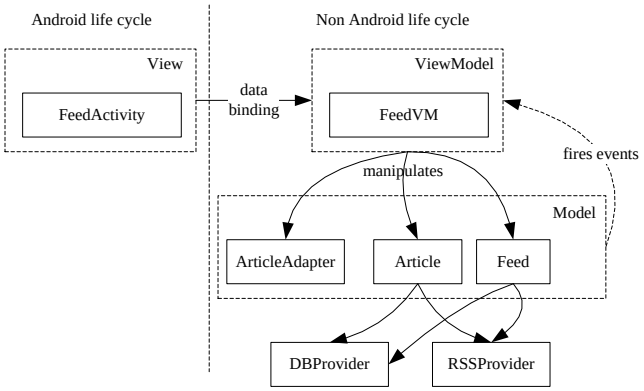

Figure 5 the application of MVVM model in the article list module

The conclusion

The development of a variety of MV+X mode, in essence, is to realize the logic control between Model and View, in a low coupling, low cost way. Using the MVVM model on the Android platform has the following advantages:

- Achieve a low coupling, which is easy for collaborative developments. The View and Model are completely decoupled, so when the project becomes bigger and developers increase, the interface design engineers and business engineers can easily collaborate. They only need to use the binding technology to facilitate the combination.
- Unit testing becomes easy. The introduction of the ViewModel is independent of the Android's life cycle, so it can be convenient to write test cases
- Reusable: ViewModel can be reused. When the same model needs to be displayed in different views, it can be bound with a variety of views. Therefore, how to define ViewModel flexibly and enhance its reusability is very important.
- In summary, the use of MVVM model under the Android platform can reduce the number of code coupling development, reducing a large number of duplicate codes, and realizing the unit test conveniently.
- Although the Google Data Binding technology under the development is not satisfactory, such as no code completion tips and the unstable of the updated version, with the constantly updating and improvement of Google Data Binding technology, MVVM mode will be widely used in Android platform.

Acknowledgements

This work was financially supported by
Integrating NSF/IEEE-TCPP Curriculum Initiative on PDC to Software Engineering Course System at Beijing Normal University Zhuhai（No:EduPar-16）

Reference

[1] Grady Booch. Object-Oriented Analysis and Design with Applications[M]. Addison-Wesley Professional,2006.5.

[2] John Gossman Introduction to Model/View/ViewModel pattern for building WPF apps http://blogs.msdn.com/b/johngossman/archive/2005/10/08/478683.aspx 2005.10

[3] HuJingjing ZhengZhiyun. Research on Personalized Information Service Based on RSS [J]. Computer Applications and Software,2009,26(05) :1

[4] Holzner, Steven. Secrets of RSS [M]. Addison-Wesley .2005:3-16

[5] W3C.RSS 2.0 Specification. http://validator.w3.org/feed/docs/rss2.html . 2010

[6] Dave Johnson.RSS and Atom in action [M].American.Manning Publications,2006

[7] Brett D.McLaughlin,Justin Edelson.Java&XML[M].0'Reilly Media,2007.6.

[8] Rome.jar [EB/OL]. http://rometools.github.io/rome/index.html.2016

[9] How Rome Works [EB/OL]. http://rometools.github.io/rome/HowRomeWorks/index.html .2009

[10] nirajrules. MVC vs. MVP vs. MVVM [EB/OL]. https://nirajrules.wordpress.com/2009/07/18/mvc-vs-mvp-vs-mvvm/，2009.

[11] John Gossman Introduction to Model/View/ViewModel pattern for building WPF APPs http://blogs.msdn.com/b/johngossman/archive/2005/10/08/478683.aspx 2005.10

[12] Google. Data Binding Guide [EB/OL]. http://developer.android.com/intl/zh-cn/tools/data-binding/guide.html. 2015.7

[13] Google. Material design [EB/OL]. http://www.google.com/design/spec/material-design/introduction.html

[14] Google. What is Material? [EB/OL]. http://www.google.com/design/spec/what-is-material/environment.html#environment-3d-world

[15] http://www.infoq.com/cn/articles/the-definition-development-and-future-of-baas-services.

[16] android-volley [EB/OL]. https://github.com/mcxiaoke/android-volley.

[17] Google Developers Google I/O 2013 - Volley: Easy, Fast Networking for Android [EB/OL]. https://www.youtube.com/watch?v=yhv8l9F44qo&feature=player_embedded

[18] TinyBus [EB/OL]. https://github.com/beworker/tinybus.