



Hongyang

生命不息，奋斗不止，万事起于忽微，量变引起质变

目录视图

摘要视图

RSS 订阅

个人资料



鸿洋_

发私信



访问：15180515次

积分：48548

等级：BLOG > B

排名：第58名

原创：201篇 转载：0篇

译文：6篇 评论：14567条

我的微信公众号

点击直达推送文章汇总

长期为您推荐优秀博文、开源项目、视频等，进入还有好玩的等着您，欢迎扫一扫。



联系方式

新动态

给我写信

QQ群：

497438697

请勿重复加群，Thx

文章分类

【Android 5.x】 (11)

【Android 精彩案例】 (38)

【Android 源码解析】 (30)

浅谈 MVP in Android

标签： MVP Android

2015-06-23 09:11

182010人阅读

评论(232)

分类：

【android 进阶之路】 (74) ▾

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录(?)

[+]

转载请标明出处：

<http://blog.csdn.net/lmj623565791/article/details/46596109>；

本文出自：[【张鸿洋的博客】](#)

一、概述

对于MVP (Model View Presenter)，大多数人都能说出一二：“MVC的演化版本”，“让Model和View完全解耦”等等。本篇博文仅是为了做下记录，提出一些自己的看法，和帮助大家如何针对一个Activity页面去编写针对MVP风格的代码。

对于MVP，我的内心有一个问题：

为何这个模式出来后，就能被广大的Android的程序员接受呢？

问了些程序员，他们对于MVP的普遍的认识是：“代码很清晰，不过增加了很多类”。我在第一次看到MVP的时候，看了一个demo，看完以后觉得非常nice（但是回过头来，自己想个例子写，就头疼写不出来，当然这在后文会说）。nice的原因还是因为，这个模式的确让代码的清晰度有了很大的提升。

那么，提升一般都是对比出来的，回顾下，没有应用MVP的代码结构。很多人说明显是MVC么：

- View：对应于布局文件
- Model：业务逻辑和实体模型
- Controllor：对应于Activity

看起来的确像那么回事，但是细细的想想这个View对应于布局文件，其实能做的事情特别少，实际上关于该布局文件中的数据绑定的操作，事件处理的代码都在Activity中，造成了Activity既像View又像Controllor（当然了Data-Binder的出现，可能会让View更像View吧）。这可能也就是为何，在[该文](#)中有一句这样的话：

Most of the modern **android** applications just use View-Model architecture , everything is connected with Activity.

微信关注我的公众号

- 【Android 自定义控件实战】 (29)
- 【Android 自定义控件之起步】 (7)
- 【Android 快速开发】 (12)
- 【Android 原生开发游戏】 (3)
- 【Java 并发专题】 (15)
- 【android 进阶之路】 (75)
- 【Java 设计模式】 (10)
- 【Android 百度地图】 (4)
- 【html5 css3精彩案例】 (14)
- 【Android github 控件】 (10)
- 【Android 基础】 (16)
- 【Javascript 】 (9)
- 【rabbitMQ 用法】 (5)
- 【Android微知识点】 (4)

友情链接

- 郭霖的博客
- 夏安明的博客
- 任玉刚的博客
- 亓斌的博客
- Github开源项目的配套视频
- 敬佩的孔老师
- foruok的订阅号程序视界
- OpenCV大神shiter
- 专为Android程序员的导航
- 泡在网上的日子

博客专栏

- 

HTML5 & CSS3 实战
- 文章：11篇

阅读：203393
- 

设计模式融入生活
- 文章：10篇

阅读：117768
- 

Android 精彩案例
- 文章：67篇

阅读：5841187

阅读排行

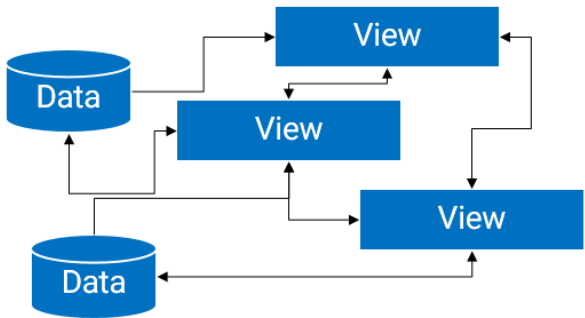
- Android Https相关完全解析 ... (1587745)
- Android Fragment 真正的完... (669220)
- Android RecyclerView 使用... (664194)
- Android OkHttp完全解析 是... (494095)
- Android 自定义View (一) (311783)
- Android 属性动画 (Property... (280615)
- Android 屏幕适配方案 (248599)
- Android Fragment 真正的完... (239884)
- Android 手把手教您自定义Vi... (222288)

而当将**架构**改为MVP以后，Presenter的出现，将Activity视为View层，Presenter负责完成View层与Model层的交互。现在是这样的：

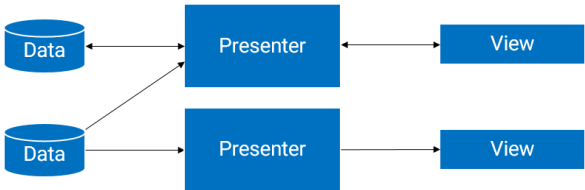
- View 对应于Activity，负责View的绘制以及与用户交互
- Model 依然是业务逻辑和实体模型
- Presenter 负责完成View于Model间的交互

ok，先简单了解下，文中会有例子到时候可以直观的感受下。

小总结下，也就是说，之所以让人觉得耳目一新，是因为这次的跳跃是从 并不标准的mvc到mvp的一个转变，减少了Activity的职责，简化了Activity中的代码，将复杂的逻辑代码提取到了Presenter中进行处理。与之对应的好处就是，耦合度更低，更方便的进行**测试**。借用两张图（出自[该文](#)），代表上述的转变：

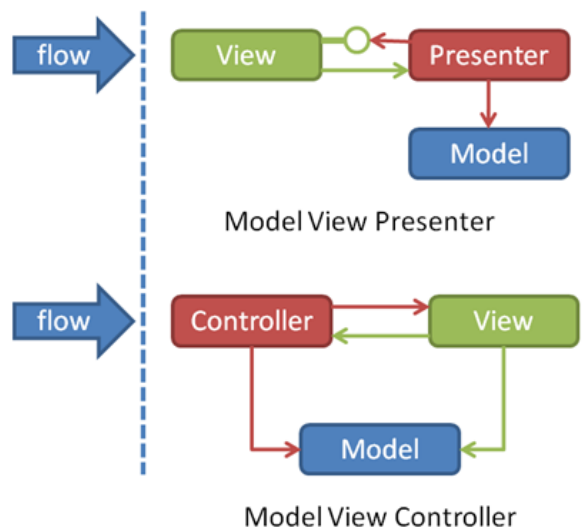


转变为：



二、MVP 与 MVC 区别

ok，上面说了一堆理论，下面我们还是需要看一看MVC与MVP的一个区别，请看下图（来自：[本文](#)）：



Android 沉浸式状态栏攻略 ...

(210841)

文章搜索

- 最新评论
- Android TagFlowLayout完全解析 一款...

qq_36499708 : chenliguan[reply]怎么设置单选中后, 再次点击选中的不让它消失呢, 兄弟, 你解决了吗
- Android TagFlowLayout完全解析 一款...

qq_36499708 : @lzy371043358:我也遇到这个问题, 感觉设置1后, 并不是实现真正的单选, 因为在此点击后会消...
- Android AutoLayout全新的适配方式 堪...

ieee_mingxing : 应该会遇到很多问题吧。如果能完美适配, google早就改写代码了。不过初衷是很好。
- Android AutoLayout全新的适配方式 堪...

ieee_mingxing : 应该会遇到很多问题吧。如果能完美适配, google早就改写代码了。不过初衷是很好。
- Android Fragment 真正的完全解析 (下)

Amandu1995 : 666666
- Android Fragment 真正的完全解析 (上)

Amandu1995 : @BigBigBai:哈哈被鸿洋大神偷懒了, 其实就是底部四个tab而已, 还有其实下面又继续偷懒, 他写...
- Android Fragment 真正的完全解析 (上)

Amandu1995 : @qq_38773678:源码都在上面了
- Android 自定义View (一)

74128796 : 留言支持
- Android 官方推荐: DialogFragment 创...

Amandu1995 : 这个效果其实AlertDialog.setView可以实现做出来了!
- Android 优雅的为RecyclerView添加Hea...

ideal_gn : 这个添加的头布局的宽没有铺满全屏, 不知道大神遇到过没, 能给个解决思路吗

统计

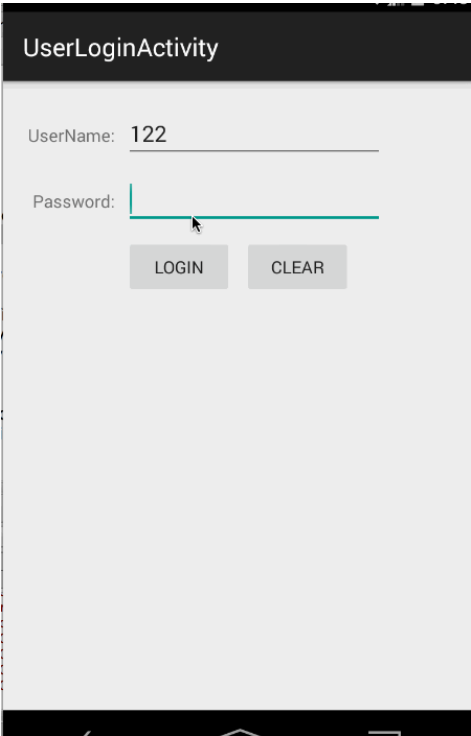
微信公众号

其实最明显的区别就是, MVC中是允许Model和View进行交互的, 而MVP中很明显, Model与View之间的交互由Presenter完成。还有一点就是Presenter与View之间的交互是通过接口的(代码中会体现)。

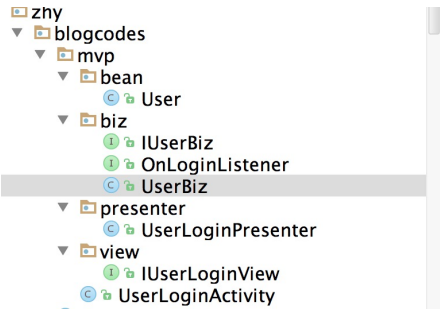
还有一堆概念性的东西, 以及优点就略了, 有兴趣自行百度。下面还是通过一些简单的需求来展示如何编写MVP的demo。

三、Simple Login Demo

效果图:



看到这样的效果, 先看下完工后的项目结构:



ok, 接下来开始一步一步的编写思路。

(一) Model

首先实体类User不用考虑这个肯定有, 其次从效果图可以看到至少有一个业务方法login(), 这两点没什么难度, 我们首先完成:

```
1 package com.zhy.blogcodes.mvp.bean;
2
3 /**
4  * Created by zhy on 15/6/18.
5  */
6 public class User
7 {
```

微信关注我的公众号

```
8     private String username ;
9     private String password ;
10
11     public String getUsername()
12     {
13         return username;
14     }
15
16     public void setUsername(String username)
17     {
18         this.username = username;
19     }
20
21     public String getPassword()
22     {
23         return password;
24     }
25
26     public void setPassword(String password)
27     {
28         this.password = password;
29     }
30 }
...
1
2 package com.zhy.blogcodes.mvp.biz;
3
4 /**
5  * Created by zhy on 15/6/19.
6  */
7 public interface IUserBiz
8 {
9     public void login(String username, String password, OnLoginListener loginListener)
10 }
11
```

```
1
2 package com.zhy.blogcodes.mvp.biz;
3
4 import com.zhy.blogcodes.mvp.bean.User;
5
6 /**
7  * Created by zhy on 15/6/19.
8  */
9 public class UserBiz implements IUserBiz
10 {
11
12     @Override
13     public void login(final String username, final String password, final OnLoginListe
14     {
15         //模拟子线程耗时操作
16         new Thread()
17         {
18             @Override
19             public void run()
20             {
21                 try
22                 {
23                     Thread.sleep(2000);
24                 } catch (InterruptedException e)
25                 {
26                     e.printStackTrace();
27                 }
28                 //模拟登录成功
29                 if ("zhy".equals(username) && "123".equals(password))
30                 {
31                     User user = new User();
32                     user.setUsername(username);
33                     user.setPassword(password);
34                     loginListener.loginSuccess(user);
35                 }
36             }
37         }
38     }
39 }
```

```

35         } else
36         {
37             loginListener.loginFailed();
38         }
39     }
40     }.start();
41 }
42 }

```

```

1
2 package com.zhy.blogcodes.mvp.biz;
3
4 import com.zhy.blogcodes.mvp.bean.User;
5
6 /**
7  * Created by zhy on 15/6/19.
8  */
9 public interface OnLoginListener
10 {
11     void loginSuccess(User user);
12
13     void loginFailed();
14 }
15

```

实体类不用说，至于业务类，我们抽取了一个接口，一个实现类这也常见~~login方法，一般肯定是连接服务器的，是个耗时操作，所以我们开辟了子线程，Thread.sleep(2000)模拟了耗时，由于是耗时操作，所以我们通过一个回调接口来通知登录的状态。

其实这里还是比较好写的，因为和传统写法没区别。

(二) View

上面我们说过，Presenter与View交互是通过接口。所以我们这里需要定义一个 ILoginView，难点就在于应该有哪些方法，我们看一眼效果图：

可以看到我们有两个按钮，一个是login，一个是clear；

login说明了要有用户名、密码，那么对应两个方法：

```

1
2     String getUsername();
3
4     String getPassword();

```

再者login是个耗时操作，我们需要给用户一个友好的提示，一般就是操作ProgressBar，所以再两个：

```

1     void showLoading();
2
3     void hideLoading();

```

login当然存在登录成功与失败的处理，我们主要看成功我们是跳转Activity，而失败可能是去给个提醒：

```

1     void toMainActivity(User user);
2
3     void showFailedError();

```

ok，login这个方法我们分析完了~~还剩个clear那就简单了：

```
1      void clearUserName();
2
3      void clearPassword();
```

综上，接口完整为：

```
1  package com.zhy.blogcodes.mvp.view;
2
3  import com.zhy.blogcodes.mvp.bean.User;
4
5  /**
6   * Created by zhy on 15/6/19.
7   */
8  public interface IUserLoginView
9  {
10     String getUsername();
11
12     String getPassword();
13
14     void clearUserName();
15
16     void clearPassword();
17
18     void showLoading();
19
20     void hideLoading();
21
22     void toMainActivity(User user);
23
24     void showFailedError();
25
26 }
```

有了接口，实现就太好写了~~~

总结下，对于View的接口，去观察功能上的操作，然后考虑：

- 该操作需要什么？（getUsername, getPassword）
- 该操作的结果，对应的反馈？(toMainActivity, showFailedError)
- 该操作过程中对应的友好的交互？(showLoading, hideLoading)

下面贴一下我们的View的实现类，哈，其实就是Activity，文章开始就说过，MVP中的View其实就是Activity。

```
1  package com.zhy.blogcodes.mvp;
2
3  import android.os.Bundle;
4  import android.support.v7.app.ActionBarActivity;
5  import android.view.View;
6  import android.widget.Button;
7  import android.widget.EditText;
8  import android.widget.ProgressBar;
9  import android.widget.Toast;
10
11  import com.zhy.blogcodes.R;
12  import com.zhy.blogcodes.mvp.bean.User;
13  import com.zhy.blogcodes.mvp.presenter.UserLoginPresenter;
14  import com.zhy.blogcodes.mvp.view.IUserLoginView;
15
16  public class UserLoginActivity extends ActionBarActivity implements IUserLoginView
17  {
18
19
20     private EditText mEtUsername, mEtPassword;
21     private Button mBtnLogin, mBtnClear;
```

```
22     private ProgressBar mPbLoading;
23
24     private UserLoginPresenter mUserLoginPresenter = new UserLoginPresenter(this);
25
26     @Override
27     protected void onCreate(Bundle savedInstanceState)
28     {
29         super.onCreate(savedInstanceState);
30         setContentView(R.layout.activity_user_login);
31
32         initView();
33     }
34
35     private void initView()
36     {
37         mEtUsername = (EditText) findViewById(R.id.id_et_username);
38         mEtPassword = (EditText) findViewById(R.id.id_et_password);
39
40         mBtnClear = (Button) findViewById(R.id.id_btn_clear);
41         mBtnLogin = (Button) findViewById(R.id.id_btn_login);
42
43         mPbLoading = (ProgressBar) findViewById(R.id.id_pb_loading);
44
45         mBtnLogin.setOnClickListener(new View.OnClickListener()
46         {
47             @Override
48             public void onClick(View v)
49             {
50                 mUserLoginPresenter.login();
51             }
52         });
53
54         mBtnClear.setOnClickListener(new View.OnClickListener()
55         {
56             @Override
57             public void onClick(View v)
58             {
59                 mUserLoginPresenter.clear();
60             }
61         });
62     }
63
64
65     @Override
66     public String getUser_name()
67     {
68         return mEtUsername.getText().toString();
69     }
70
71     @Override
72     public String getPassword()
73     {
74         return mEtPassword.getText().toString();
75     }
76
77     @Override
78     public void clearUser_name()
79     {
80         mEtUsername.setText("");
81     }
82
83     @Override
84     public void clearPassword()
85     {
86         mEtPassword.setText("");
87     }
88
89     @Override
90     public void showLoading()
91     {
92         mPbLoading.setVisibility(View.VISIBLE);
```

```

93     }
94
95     @Override
96     public void hideLoading()
97     {
98         mPbLoading.setVisibility(View.GONE);
99     }
100
101     @Override
102     public void toMainActivity(User user)
103     {
104         Toast.makeText(this, user.getUsername() +
105             " login success , to MainActivity", Toast.LENGTH_SHOR
106     }
107
108     @Override
109     public void showFailedError()
110     {
111         Toast.makeText(this,
112             "login failed", Toast.LENGTH_SHORT).show();
113     }
114 }

```

对于在Activity中实现我们上述定义的接口，是一件很容易的事，毕竟接口引导我们去完成。

最后看我们的Presenter。

(三) Presenter

Presenter是用作Model和View之间交互的桥梁，那么应该有什么方法呢？

其实也是主要看该功能有什么操作，比如本例，两个操作:login和clear。

```

1  package com.zhy.blogcodes.mvp.presenter;
2
3  import android.os.Handler;
4
5  import com.zhy.blogcodes.mvp.bean.User;
6  import com.zhy.blogcodes.mvp.biz.IUserBiz;
7  import com.zhy.blogcodes.mvp.biz.OnLoginListener;
8  import com.zhy.blogcodes.mvp.biz.UserBiz;
9  import com.zhy.blogcodes.mvp.view.IUserLoginView;
10
11
12  /**
13   * Created by zhy on 15/6/19.
14   */
15  public class UserLoginPresenter
16  {
17      private IUserBiz userBiz;
18      private IUserLoginView userLoginView;
19      private Handler mHandler = new Handler();
20
21      public UserLoginPresenter(IUserLoginView userLoginView)
22      {
23          this.userLoginView = userLoginView;
24          this.userBiz = new UserBiz();
25      }
26
27      public void login()
28      {
29          userLoginView.showLoading();
30          userBiz.login(userLoginView.getUserName(), userLoginView.getPassword(), new On
31          {
32              @Override
33              public void loginSuccess(final User user)
34              {
35                  //需要在UI线程执行
36                  mHandler.post(new Runnable()

```

微信关注我的公众号


```
37         {
38             @Override
39             public void run()
40             {
41                 userLoginView.toMainActivity(user);
42                 userLoginView.hideLoading();
43             }
44         });
45     }
46 }
47
48 @Override
49 public void loginFailed()
50 {
51     //需要在UI线程执行
52     mHandler.post(new Runnable()
53     {
54         @Override
55         public void run()
56         {
57             userLoginView.showFailedError();
58             userLoginView.hideLoading();
59         }
60     });
61 }
62 }
63 });
64 }
65
66 public void clear()
67 {
68     userLoginView.clearUserName();
69     userLoginView.clearPassword();
70 }
71
72
73
74 }
```

注意上述代码，我们的presenter完成二者的交互，那么肯定需要二者的实现类。大致就是从View中获取需要的参数，交给Model去执行业务方法，执行的过程中需要的反馈，以及结果，再让View进行做对应的显示。

ok，拿到一个例子经过上述的分解基本就能完成。以上纯属个人见解，欢迎讨论和吐槽~ have a nice day ~。

[源码点击下载](#)

微信公众号：hongyangAndroid
(欢迎关注，第一时间推送博文信息)



参考资料

- <https://github.com/zhengxiaopeng/Rocko-Android-Demos/tree/master/android-mvp>

微信关注我的公众号

- <https://github.com/antonioig/androidmvp>
- <https://github.com/pedrovgs/EffectiveAndroidUI>
- <http://zhengxiaopeng.com/2015/02/06/Android%E4%B8%AD%E7%9A%84MVP/>
- <http://magenic.com/Blog/Post/6/An-MVP-Pattern-for-Android>
- <http://antonioleiva.com/mvp-android/>
- <http://konmik.github.io/introduction-to-model-view-presenter-on-an>

顶 踩
161 40

- [上一篇](#) Android 不规则封闭区域填充 手指秒变油漆桶
- [下一篇](#) FloatingActionButton 完全解析[Design Support Library(2)]

相关文章推荐

- android中MVP模式 (一)
- 浅谈Android中的MVP
- 在Android开发中使用MVP模式
- Android官方MVP架构解读
- Android开发MVP模式解析
- Android中的MVP架构初探
- Android中的MVP模式使用
- 浅谈安卓MVP模式
- Android中的MVP
- 浅谈 MVP in Android

猜你在找

- 深度学习基础与TensorFlow实践
- 【在线峰会】一天掌握物联网全栈开发之道
- 机器学习40天精英计划
- 微信小程序开发实战
- 备战2017软考 系统集成项目管理工程师 学习套餐
- 【在线峰会】前端开发重点难点技术剖析与创新实践
- 【在线峰会】如何高质高效的进行Android技术开发
- Python数据挖掘与分析速成班
- JFinal极速开发企业实战
- Python大型网络爬虫项目开发实战 (全套)

查看评论



gongliwin

150楼 2017-05-23 00:14发表

结构清晰，但是完全看不出好处在哪里，web端mvc架构的好处就是解耦，既然android要解耦，activity中做的事明显多了，xml布局就是view层的话，activity里就该通过接口把所有控件和activity本身传给p层去做处理，当然我们不会去传activity本身和控件本身，我们需要传他的接口。看到mvp模式和我想的不一样，很失望。如果按楼主所说，去提前写好每个按钮可能的方法，接口里面提前写上，感觉开发起来特别费力。



俞志云求职ing安卓实习大三985本科

149楼 2017-05-11 15:56发表

<http://blog.csdn.net/yuzhiyun3536/article/details/71634026> mvp实现的下载文件demo,带进度条，(AsyncTask HttpURLConnection实现)，欢迎点评

微信关注我的公众号



SunnyCoffee

148楼 2017-05-11 10:34发表

文中关于mvc的理解表示质疑，在其他的文中MVC表示成：

$M \Leftarrow \Rightarrow C(\text{Activity}) \Leftarrow \Rightarrow V(\text{View})$

而MVP表示成：

$M \Leftarrow \Rightarrow P \Leftarrow \Rightarrow V(\text{Activity} \& \text{View})$

不存在View和Mode交互的问题。



flfCoder

147楼 2017-04-27

厉害了我的洋神



qq_26261937

146楼 2017-04-02

这有个不错的项目，UI模仿网易云音乐，基于Material Design + Rxjava + Retrofit + dagger2 + MVP构架的项目：项目地址<https://github.com/laotan7237/EasyReader>

如果哪里不好可以给作者issue



井一流

145楼 2017-03-30 16:05发表

感觉代码多了好几倍，文件多了10多个，没啥意思啊



wyc7556342

Re: 2017-04-28 16:06发表

引用“android_001_download”的评论：

感觉代码多了好几倍，文件多了10多个，没啥意思啊

感觉这个mvp的设计模式，很明显的一个缺点代码量带创建的文件多，它所谓的条理清晰的优点，也表现的不是很抢眼。



RxWorld

144楼 2017-03-29 15:55发表

还有，Model层为什么还要做数据检验呢？不是应该放到Presenter层吗？求大神指点



鲁哈

Re: 2017-04-21 10:00发表

回复RxWorld：小伙子，这里有说明只是模拟登录，这些应该是你传账号密码服务器判断的。这里只是模拟一下。



RxWorld

143楼 2017-03-29 15:03发表

View接口里面直接传了个user????那样不行吧？



qq_17025411

142楼 2017-03-22 00:52发表

条理清晰，点个赞



浏冰

141楼 2017-03-10 23:31发表

撤回评论，没好好看0,0



浏冰

140楼 2017-03-10 23:21发表

BIZ业务逻辑算在P层还是M层？



由此及彼

139楼 2017-03-08 15:07发表

请教一下，UML图中，P和V的关系应该是相互依赖，而不是相互关联吧。

微信关注我的公众号

**坚持不放弃**

138楼 2017-03-07 22:32发表

鸿神思维很流弊

**安卓美女**

137楼 2017-02-28 15:59发表

鸿阳大哥,你用什么写的博客啊,排版很好.

**随心_搬砖客**

136楼 2017-02-16

哎

**ganshenml**

135楼 2017-02-06

感觉这样的架构方法不是太好：

如果是简单业务逻辑的Activity，直接用mv就可以了；如果是负责业务逻辑的Activity，拆分出来了好多接口方法，而且随着Activity或者fragment过多，接口方法也变得越来越来多。。。

**qq_27426665**

Re: 2017-03-22 09:38发表

回复ganshenml：你可以用MVVM框架

**乐活青年**

134楼 2017-02-04 18:11发表

mvp仍然不是很好用

**demon_die**

133楼 2017-01-19 17:49发表

按照这个思路搞了一下，我的view接口中方法达到10个的时候，就不敢再往下搞了，还遥遥无期呢

**Juzphome**

132楼 2017-01-14 16:48发表

在文章MVC模式的图，数据走向是否有问题？View怎么会直接对Model操作

**Juzphome**

Re: 2017-01-14 16:53发表

回复Juzphome：在上面MVC模式图中，如果model数据更新了，应该告诉Controller层，更新View界面，可上面的图数据走向，不是这样的

**qq_36423673**

131楼 2016-11-11 15:31发表

洪神我有个问题想请教你，我最看重 MVP两个重要优点：

- 1.更方便团队开发的分工职责，代码好维护，团队人员之间的代码易读性更强。
- 2.方便需求更改时的拓展和修改。

但是相对小团队来说，一般Android一个中小项目最多2个人来开发，使用MVP无疑要在构架上花不少时间来搭建模型，在实际项目上，该如何度量“时间”，“后期维护成本”的价值呢？

**qq_36423673**

130楼 2016-11-11 15:31发表

洪神我有个问题想请教你，我最看重 MVP两个重要优点：

- 1.更方便团队开发的分工职责，代码好维护，团队人员之间的代码易读性更强。
- 2.方便需求更改时的拓展和修改。

但是相对小团队来说，一般Android一个中小项目最多2个人来开发，使用MVP无疑要在构架上花不少时间来搭建模型，在实际项目上，该如何度量“时间”，“后期维护成本”的价值呢？

**Android小工**

129楼 2016-11-11 14:00:24发表

[微信关注我的公众号](#)

可以下载我的MVP项目看一下，相信对初学者帮助很大<https://github.com/qiaoyhh/MvpPlus>



Luyifei666

Re: 2017-03-18 09:38发表

回复Android小工：打不开啊



寥寥过客

Re: 2016-11-16

一直想找份较完整的小项目看看是怎么搭建框架的,受教了



魔法少年康思奇

128楼 2016-11-09 11:24发表

看了数十个MVP介绍，终于在看到这里的时候，稍微的看懂了一些



u011142996

Re: 2016-11-29 17:45发表

回复魔法少年康思奇：我也是



ziyouzhifeng007

127楼 2016-10-30 00:14发表

把View单独抽出来实现ILoginView接口呢



SEU_Calvin

126楼 2016-10-20 21:16发表

接口有点太多了



u010670174

125楼 2016-10-20 16:15发表

一百个接口,写一百个这样的,累死你



Developmc

124楼 2016-10-17 14:49发表

学习了



hunk_lh

123楼 2016-09-29 13:36发表

感谢楼主 不错



Mr_ggx

122楼 2016-09-23 23:24发表

这篇文章的只能是一个毛皮，问题还有很多，比如内存泄漏，而且很容易造成代码冗余



做一名优秀的Android工程师

Re: 2016-11-13 14:29发表

回复Mr_ggx：不懂内存泄漏,希望有懂得朋友分享一下.



ayzhiaho

Re: 2016-10-10 15:35发表

回复Mr_ggx：浅谈.....



过...路鬼

121楼 2016-09-19 16:55发表

先Mark，在阅读学习。^_^



三刀飞

120楼 2016-09-18 09:57发表

微信关注我的公众号

根据图示，View跟Model是不会有交互的，那么在您的View层的接口中，有一个toMainActivity(User user)方法，这个方法引用到了Model层的User实体，这是不是不可避免的问题呢？



SEU_Calvin

Re: 2016-10-21 10:01发表

回复三刀飞：个人感觉Model层主要是提供数据的存取功能。Presenter通过它来存储、获取数据。对应于项目中封装了数据库DAO或者网络获取数据的角色。像User类这种基本的实体类严格来说不属于Model层。



tangmilon

119楼 2016-09-17 10:00发表

如果activity的界面被干掉了，引用还在，那presster回调此activity，在回调方法中有使用界面元素，则会出现NullPointerException的问题。



为i存在

Re: 2016-10-06 19:39发表

回复tangmilon：那样的话activity就不会被干掉，因为垃圾回收器不会回收一个被持有强引用的内存，这就是内存泄漏



做一名优秀的Android工程师

Re: 2016-11-13 14:44发表

回复为i存在：可以告知一下解决办法吗,谢谢~!



kim_小金

Re: 2016-09-19 18:00发表

回复tangmilon：activity 被干掉，对应的persenter 也会被干掉



一念三千相

118楼 2016-09-08 11:50发表

修改数据，以及调用adapter的notifyDataSetChanged()在封装成方法放在Activity里。

例如:这是在activity里

```
void setData(String result){
    mList.add(result);
    mAdapter.notifyDataSetChanged();
}
```

然后在presenter里

获取到activity对象之后，去model层拿到数据mData。然后在调用 m Activity.setData(mData);

以上一种解决方案。



漠北之夜

Re: 2016-09-14 17:24发表

回复一念三千相：网上介绍的这个MVP都是简单的模式，没有考虑你说的问题，你可以去GitHub上找一些开源项目看一下就知道，好像是抽取了BaseView通过Bind的方式去处理就解决了你说的这个问题，具体你可以去搜一下，我之前看那些项目别人这么用一直不理解，后面在某培训机构的公开课看了这一课才明白用意何在，可惜最后的优化他留着不讲，沃日



一念三千相

117楼 2016-09-08 11:49发表

修改数据，以及调用adapter的notifyDataSetChanged()在封装成方法放在Activity里。

例如:这是在activity里

```
void setData(String result){
    mList.add(result);
    mAdapter.notifyDataSetChanged();
}
```

然后在presenter里

微信关注我的公众号

获取到activity对象之后，去model层拿到数据mData。然后在调用 m
Activity.setData(mData);



一念三千相

116楼 2016-09-08 11:44发表

修改数据，以及调用adapter的notifyDataSetChanged()在封装成方法
放在Activity里。

例如:这是在activity里

```
void setData(String result){  
    mList.add(result);  
    mAdapter.notifyDataSetChanged();  
}
```

然后在presenter里

获取到activity对象之后，去model层拿到数据mData。然后在调用 m
Activity.setData(mData);

以上一种解决方案。



一念三千相

115楼 2016-09-08 11:35发表

Presenter经常需要执行耗时操作，时间会比较长。这种写法 Present
er 持有IUserLoginView（也就是Activity）是强引用,如果请求结束之
前Activity被销毁了，那么Presenter操作还未完成。导致Presenter一
直持有该对象，使得无法回收，就会发生内存泄漏。



乐活青年

Re: 2017-02-04 18:10发表

回复一念三千相：我想，可以在P中增加一个方法destor
y，activity销毁时调用destory，在这里面解除对V的引
用，当然在P里面使用V时需要先判空了。或者使用softref
erence持有V。个人见解。



CurseMundi

114楼 2016-08-30 16:14发表

不懂MVP,但是原先碰到个一个Android项目,涉及复杂的业务逻辑,后来
为了实现业务与Activity分离,就直接添加一个service层,没有使用接口,
以activity的属性存在,只是有时候会使用回调来更新UI线程.MVP有
机会试试看吧



繁华未至

113楼 2016-08-25 18:13发表

这个P层越看越像三层架构的bll层。



游鑫

112楼 2016-08-16 12:47发表

https://github.com/youxin11544/mvp_hybride_framework This is an A
ndroid MVP model good architecture design , Which also inherited t
he Android architecture and HTML 5 interaction。（这是一个Androi
d MVP模型良好的架构设计,同时也做了Android和HTML 5交互架构。



Ivan_mo

111楼 2016-08-10 00:00发表

UserLoginActivity extends ActionBarActivity implements IUserLogin
View 这种做法有点不妥吧，就单单一个登录UserLoginActivity就已经
Override这么多方法了，复杂一点的会不会导致Activity很臃肿，我自
己也写了个MVP Demo，把业务抽取了出来，Activity变得简洁多了，
和大家分享一下，希望共同进步：[https://github.com/moqingliang/an
droid-mvp-demo](https://github.com/moqingliang/android-mvp-demo)



KumuiP

110楼 2016-08-08 10:19发表

在稍显复杂的业务逻辑中，Presenter多接口回调的实现模式有倾向模
板设计的嫌疑，频繁的数据与UI的变化会使抽象的合理性变得更加困
难，不能很好的拥抱变化。我认为这是Presenter早已提出但并未广泛
标准化使用的重要原因之一。

我认为应该采用代理的思想（非模式），即业务工具类（Presente
r）。将一个Activity（View）中的核心逻辑“委托”给该工具类（View
持有）。我们可以在View中实现针对View的动作的处理，比如数据刷
新等，在工具类中对业务进行真正的处理。这样在工具类中可以对业

微信关注我的公众号

务的优化整合更为集中，也免去了频繁的数据UI变动对二次接口的调用流程。部分在Activity中的额外监听等也可以通过工具类的public方法来进行处理。

进行委托之后，View持有工具类模拟MVC，工具类持有子View，也可以自包含网络的数据获取功能。对于同一View更复杂的逻辑，比如相当多相对比较独立的模块，可以采用中介者模式来处理，让Activity做中介者或者另外新建类。

当然，工具类中还是数据和UI混在一起，但是这仅仅可以当做是业务模块了，这样的话可以多做方法进行整理，或者对代码的编写风格进行规范，这样就可以在可读性和结构上可以有所优化。对于习惯了Activity中大量代码的我们来说，这样毫无压力。

说来说去好像比较乱，其实主要就是想用MVP的设计思想还原到MVC模式，目的是对较为混乱的Activity进行分离。这样算不算Presenter我也有点迷糊了。请抽时间大神点评一下最好了，非常感谢！



KumuiP

Re: 2016-08-08 10:25发表

回复KumuiP：或者说是MVVM模式



蔡CJ

109楼 2016-07-27 14:50发表

请教大神们，M层的接口意义何在？



叶国豪

Re: 2016-07-29 14:28发表

回复蔡CJ：M层是逻辑操作层，P从V（activity）拿到数据交给M处理



SimpleIsTrue

108楼 2016-07-06 06:50发表

MVP的目的主要是进行解耦，减少模块之间的直接调用。Presenter和View之间，view可以完全调用presenter，而presenter调用View是通过接口调用的，这就基于View经常改变（各种产品需求改动）而presenter改动较少的情况设计的，不对的地方，希望大家指出。



forecheng

107楼 2016-06-27 09:37发表

这个例子很有代表性 啊



azui007

106楼 2016-06-25 18:26发表

UserLoginPresenter出现了直接控制view的代码，这样做是否有不妥。Presenter是否应该只是对数据的设置麻烦解答一下



SEU_Calvin

Re: 2016-10-21 11:02发表

回复azui007：View真正的操作在Activity里已经写好了啊，我感觉Presenter层只是起到控制的作用。Presenter和View层交互挺正常的啊。



mr_same

Re: 2016-08-11 17:53发表

回复azui007：同问



Jon小北

105楼 2016-06-15 17:54发表

这里presenter持有activity的引用而又处理耗时操作，是否有内存泄露的风险？



soft_zhang_

104楼 2016-06-04 13:59发表

发现鸿洋大神的代码格式。。可能是习惯吧。



酒伴晚风

103楼 2016-05-30 18:56发表

感谢老乡分享！关注你很久了。



qhung

102楼 2016-05-28 16:24发表

这样一个view和一个presenter对应一个界面，要是完整的项目，界面肯定不少，那和以view和presenter会暴增，这样怎么办呢？



wlcw16

Re: 2016-06-01

回复qhung：利用组合思想。V和P是一一对应的，但是，我们可以把通用的VP提取出来。一个Activity implements 多个View，然后利用组合包含几个P。举个例子。有个LoginPresenter。我们现在要在登陆页面中用到，另外在一个回复页面，也需要做个快速登陆功能。那么我们可能需要在LoginActivity和ReplyActivity中都包含这个LoginPresenter，两个Activity都各自去实现LoginView。可能ReplyActivity还有其他功能，他还要包含自己的ReplyPresenter，并实现自己的ReplyView。利用组合来实现Presenter的复用，这个是MVP的优雅之一。但是别忘了不要持有View实例，记得detach。



qhung

Re: 2016-06-13 19:29发表

回复wlcw16：嗯，谢谢解答，这个思想不错



米心语

Re: 2016-06-01 16:22发表

回复qhung：我也想知道 因为真正开发中逻辑都会比demo复杂，相应的view接口方法就会很多，感觉整个项目还更大了



qq_34312797

101楼 2016-05-12 16:04发表

学习了.3q



zcmain

100楼 2016-05-06 18:13发表

mHandler.post 很不起眼却很有用我开始直接new Thread 结果在View中更新UI结果提示 not called Looper.prepare (=. =)



qq_18602035

Re: 2016-05-27 17:27发表

回复zcmain：缺少

```
Looper.getMainLooper()
```

应该这样写

```
Handler handler = new Handler(Looper.getMainLooper());
```



非花非雾--

99楼 2016-04-26 18:29发表

我想问下，MVP模式中的Presenter接口实现类，View接口的实现类，model的实现类都是只有一个吗？如果是只有一个他又如何处理不同View的逻辑处理啊。如果我现在在一个界面现在不是登录，而是注册的话，是不是需要在view接口中在添加几个方法呢



花-开-花-谢

98楼 2016-04-12 11:14发表

那不就是封装逻辑，然后接口回调吗，可参考：<http://blog.csdn.net/gaolei1201/article/details/47084111>




zw1127266710

97楼 2016-03-30 12:12发表

换头像了？

微信关注我的公众号

 **mocaris**
洋哥，我先收藏了

96楼 2016-03-24 14:18发表

[查看更多评论](#)

发表评论

用户名：

xfhy_

评论内容：




提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

[网站客服](#) [杂志客服](#) [微博客服](#) webmaster@csdn.net 400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved 

微信关注我的公众号