 http://blog.csdn.net/z2wenfa/article/details/52873009

# [灵活利用泛型的MVP模式T-MVP(附带一个快速创建MVP接口的Android Studio插件-TMVPHelper)](http://blog.csdn.net/z2wenfa/article/details/52873009)

2016-10-21 10:29 3834人阅读 [评论](http://blog.csdn.net/z2wenfa/article/details/52873009" \l "comments)(1) [收藏](http://blog.csdn.net/z2wenfa/article/details/javascript:void(0);" \o "收藏" \t "http://blog.csdn.net/z2wenfa/article/details/_blank) [举报](http://blog.csdn.net/z2wenfa/article/details/52873009" \l "report" \o "举报)

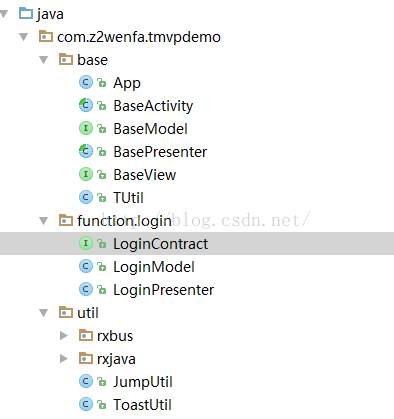
IMG_256 分类：

Android（23） IMG_257

版权声明：本文为博主原创文章，未经博主允许不得转载。

接触到MVP模式已经挺长时间了,好几个月前T-MVP我就看到了,觉得挺有想法的,于是直接就用到了这次项目里,用起来感觉还行,就花时间推荐一下吧。

大家都知道MVP模式都有的几个接口Model,View,Presenter,另外还有一个用于用于存放这几个接口Contract契约类。先看项目的结构吧:



差不多就是这样,Base包中有MVP的三个Base基类,不过没有看到login包中的View,因为这个View就是我们将要创建的Activity。

接着一步一步的的看Contract中的代码:

**[java]** [view plain](http://blog.csdn.net/z2wenfa/article/details/52873009" \o "view plain" \t "http://blog.csdn.net/z2wenfa/article/details/_blank) [copy](http://blog.csdn.net/z2wenfa/article/details/52873009" \o "copy" \t "http://blog.csdn.net/z2wenfa/article/details/_blank)

1. **package** com.z2wenfa.tmvpdemo.function.login;

4. **import** com.z2wenfa.tmvpdemo.base.BaseModel;
5. **import** com.z2wenfa.tmvpdemo.base.BasePresenter;
6. **import** com.z2wenfa.tmvpdemo.base.BaseView;
8. /\*\*
9. \* Created by z2wenfa on 2016/9/1.
10. \*/
11. **public** **interface** LoginContract {
13. **interface** View **extends** BaseView {
15. }
17. **interface** Model **extends** BaseModel {
19. }
21. **abstract** **static** **class** Presenter **extends** BasePresenter<Model, View> {
23. @Override
24. **public** **void** onStart() {
26. }
28. }
29. }

我们在Contract中声明MVP个接口所带的方法,这时我们看到BasePresenter带有2个泛型同时还是一个抽象类.

我们先来看下BasePresenter的代码:

**[java]** [view plain](http://blog.csdn.net/z2wenfa/article/details/52873009" \o "view plain" \t "http://blog.csdn.net/z2wenfa/article/details/_blank) [copy](http://blog.csdn.net/z2wenfa/article/details/52873009" \o "copy" \t "http://blog.csdn.net/z2wenfa/article/details/_blank)

1. **package** com.z2wenfa.tmvpdemo.base;
3. **import** android.content.Context;
5. **import** com.z2wenfa.tmvpdemo.util.rxbus.RxManager;

8. /\*\*
9. \* T-MVP Presenter基类
10. \* Created by baixiaokang on 16/4/22.
11. \*/
12. **public** **abstract** **class** BasePresenter<M, T> {
13. **public** Context context;
14. **public** M mModel;
15. **public** T mView;
16. **public** RxManager mRxManager = **new** RxManager();
18. **public** **void** setVM(T v, M m) {
19. **this**.mView = v;
20. **this**.mModel = m;
21. **this**.onStart();
23. }
25. **public** **abstract** **void** onStart();
27. **public** **void** onDestroy() {
28. mRxManager.clear();
29. }
30. }

其中有2个泛型变量M和T与一个set方法通过set方法将View与Model传入到Presenter中来。

但是怎么将Model与View实例化并且注入今Presenter的呢?

最关键的代码来了:

我们先看一个工具类TUtil:

**[java]** [view plain](http://blog.csdn.net/z2wenfa/article/details/52873009" \o "view plain" \t "http://blog.csdn.net/z2wenfa/article/details/_blank) [copy](http://blog.csdn.net/z2wenfa/article/details/52873009" \o "copy" \t "http://blog.csdn.net/z2wenfa/article/details/_blank)

1. **package** com.innostic.application.util;
3. **import** java.lang.reflect.ParameterizedType;
5. /\*\*
6. \* 泛型实例化工具类 (MVP模式)
7. \* Created by baixiaokang on 16/4/30.
8. \*/
9. **public** **class** TUtil {
10. **public** **static** <T> T getT(Object o, **int** i) {
11. **try** {
12. **return** ((Class<T>) ((ParameterizedType) (o.getClass()
13. .getGenericSuperclass())).getActualTypeArguments()[i])
14. .newInstance();
15. } **catch** (InstantiationException e) {
16. e.printStackTrace();
17. } **catch** (IllegalAccessException e) {
18. e.printStackTrace();
19. } **catch** (ClassCastException e) {
20. e.printStackTrace();
21. }
22. **return** **null**;
23. }
25. **public** **static** Class<?> forName(String className) {
26. **try** {
27. **return** Class.forName(className);
28. } **catch** (ClassNotFoundException e) {
29. e.printStackTrace();
30. }
31. **return** **null**;
32. }
33. }

通过这个类我们可以传入一个对象通过这个对象与泛型所在位置实例化出一个泛型的对象。

再看BaseActivity类:

**[java]** [view plain](http://blog.csdn.net/z2wenfa/article/details/52873009" \o "view plain" \t "http://blog.csdn.net/z2wenfa/article/details/_blank) [copy](http://blog.csdn.net/z2wenfa/article/details/52873009" \o "copy" \t "http://blog.csdn.net/z2wenfa/article/details/_blank)

1. /\*\*
2. \* Activity基类
3. \* Created by z2wenfa on 2016/8/25.
4. \*/
5. **public** **abstract** **class** BaseActivity<T **extends** BasePresenter, E **extends** BaseModel> **extends** Activity **implements**  JumpUtil.JumpInterface {
6. **public** T mPresenter;
7. **public** E mModel;
8. **private** ProgressDialog progressDialog;
9. **private** ConfirmDialog confirmDialog;
11. @Override
12. **protected** **void** onCreate(Bundle savedInstanceState) {
13. **super**.onCreate(savedInstanceState);
14. init();
15. }
17. **protected** **void** init() {
18. getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT\_INPUT\_ADJUST\_PAN);
19. setContentView(getLayoutResID());
20. mPresenter = TUtil.getT(**this**, 0);
21. mModel = TUtil.getT(**this**, 1);
22. **if** (**this** **instanceof** BaseView) mPresenter.setVM(**this**, mModel);
23. }
25. /\*\*
26. \* 获得Layout文件id
27. \*
28. \* @return
29. \*/
30. **protected** **abstract** **int** getLayoutResID();

33. **protected** **abstract** **void** initView();
35. }

38. }

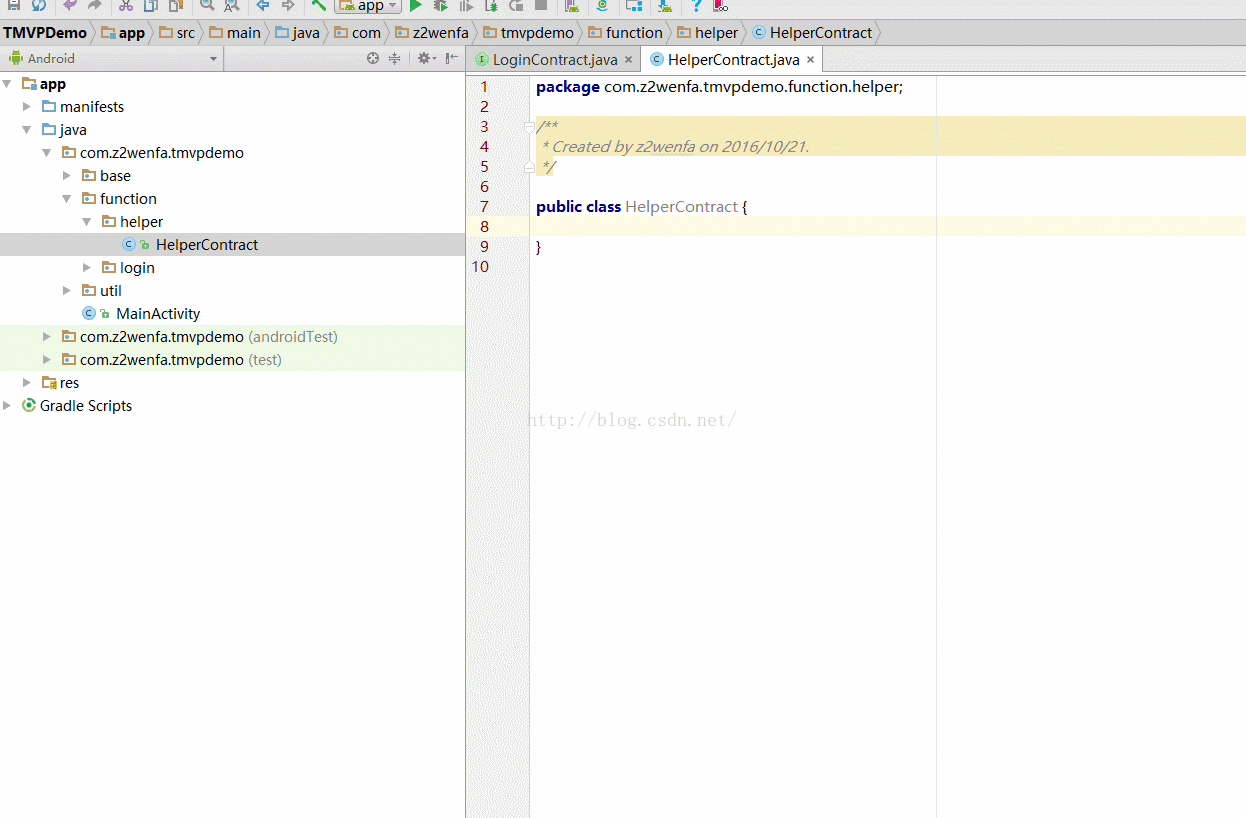
BaseActivity带有两个泛型,分别限制是继承BasePresenter与BaseModel,通过TUtil生成两个Model与Presenter中,同时调用Presenter的setVM方传入View与Model。

此时我们边能够再Activity中通过mPresenter调用Presenter,在Presenter中通过mView与mModel分别调用View与Model了。

[Demo代码](https://git.oschina.net/z2wenfa/TMVPDemo.git" \t "http://blog.csdn.net/z2wenfa/article/details/_blank)

https://gitee.com/z2wenfa/TMVPDemo

最后再给大家提供一个Android Studio的插件用于自动生成MVP模式的接口,这是我根据别人的MVPHepler改的更适合T-MVP。只需要创建在Contract类中使用,边能够自动生成通用的代码，与对应的Model和Presenter类。大家也可以根据自己的需要制作最适合自己的插件。



[TMVPHelper插件下载地址](http://download.csdn.net/detail/z2wenfa/9659697" \t "http://blog.csdn.net/z2wenfa/article/details/_blank) http://download.csdn.net/download/z2wenfa/9659697

关于这种模式的几种想法:

1.因为是通过Activity的泛型生成的Model与与Presenter,所以Activity中便有了Model的实例能够直接调用Model的方法。但是尽量不要这么做，这样的话就更MVC没什么区别了,Model和View直接就耦合了。

2.因为MVP模式Model不能调用Presenter所以Presenter如何获得Model耗时操作的结果就成了一个问题,T-MVP使用的是Rxjava来处理这个问题，但是如果对Rxjava没那么熟的话那就只能使用一个接口回调的方式了,大家可以自己选择。

T-MVP博客地址:[点击打开链接](http://www.jianshu.com/p/b49958e1889d" \t "http://blog.csdn.net/z2wenfa/article/details/_blank) http://www.jianshu.com/p/b49958e1889d

MVPHelper地址:[点击打开链接](http://blog.csdn.net/wingichoy/article/details/52148695" \t "http://blog.csdn.net/z2wenfa/article/details/_blank) http://blog.csdn.net/wingichoy/article/details/52148695

Android Studio插件编写:[点击打开链接](http://blog.csdn.net/lmj623565791/article/details/51548272" \t "http://blog.csdn.net/z2wenfa/article/details/_blank) http://blog.csdn.net/lmj623565791/article/details/51548272

TMVPHelper源码地址:[点击打开链接](https://git.oschina.net/z2wenfa/TMVPHelper" \t "http://blog.csdn.net/z2wenfa/article/details/_blank) https://gitee.com/z2wenfa/TMVPHelper