Android 中的 MVP 模式（带实例）

原创 2016-06-05 伯乐专栏/Anthony 安卓应用频道

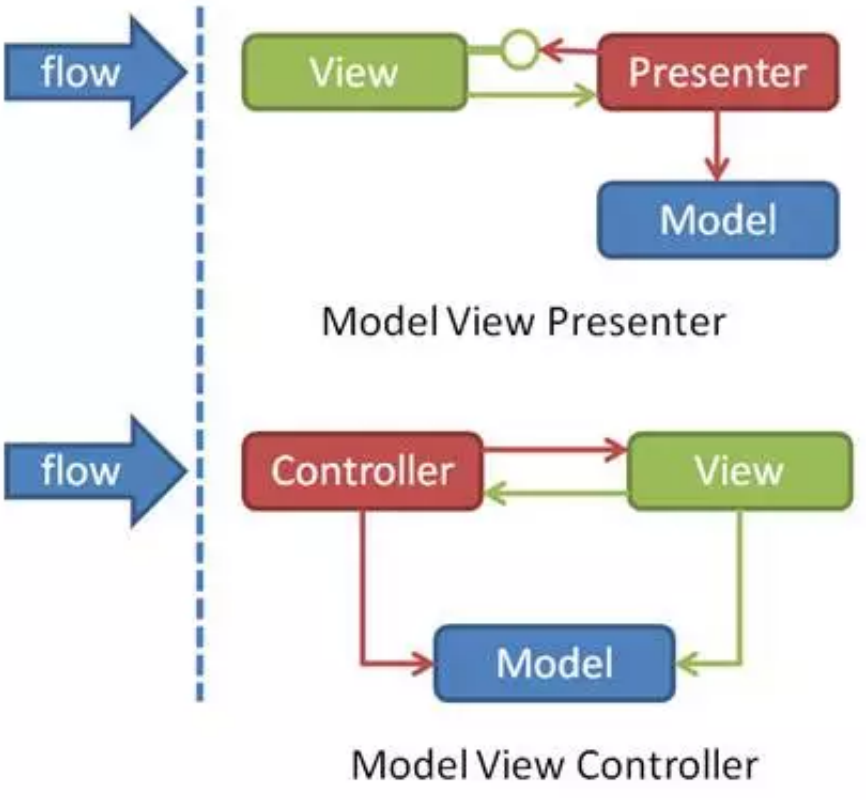
(点击上方公众号，可快速关注)

> 来源：伯乐在线专栏作者 - Anthony
> 链接：http://android.jobbole.com/83311/
> 点击 → 了解如何加入专栏作者

最近在利用工作闲暇时间学习各种网络的开源项目，也在搭建一个android开源框架，希望能够给对知识做一个总结。
这里利用一个简单的应用对MVP做一个讲解。后面也有很多github源码，都是特别经典的例子，可以学习一下。

（1）. MVP模式简介

相信大家对MVC都是比较熟悉了：M-Model-模型、V-View-视图、C-Controller-控制器，MVP作为MVC的演化版本，那么类似的MVP所对应的意义：M-Model-模型、V-View-视图、P-Presenter-表示器。 从MVC和MVP两者结合来看，Controlller/Presenter在MVC/MVP中都起着逻辑控制处理的角色，起着控制各业务流程的作用。而 MVP与MVC最不同的一点是M与V是不直接关联的也是就Model与View不存在直接关系，这两者之间间隔着的是Presenter层，其负责调控 View与Model之间的间接交互。在 Android中很重要的一点就是对UI的操作基本上需要异步进行也就是在MainThread中才能操作UI，所以对View与Model的切断分离是 合理的。此外Presenter与View、Model的交互使用接口定义交互操作可以进一步达到松耦合也可以通过接口更加方便地进行单元测试。所以也就有了这张图片（MVP和MVC的对比）



MVP和MVC的对比

> 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)。

（2）. MVP模式的应用

2.1 model层描述和具体代码

提供我们想要展示在view层的数据和具体登陆业务逻辑处理的实现，

```
package com.nsu.edu.androidmvpdemo.login;

/**
 * Created by Anthony on 2016/2/15.
 * Class Note:模拟登陆的操作的接口，实现类为LoginModelImpl.相当于MVP模式中的Model层
 */
public interface LoginModel {
    void login(String username, String password, OnLoginFinishedListener listener);
}
```

```
package com.nsu.edu.androidmvpdemo.login;

import android.os.Handler;
import android.text.TextUtils;
/**
 * Created by Anthony on 2016/2/15.
 * Class Note:延时模拟登陆（2s），如果名字或者密码为空则登陆失败，否则登陆成功
 */
public class LoginModelImpl implements LoginModel {

    @Override
    public void login(final String username, final String password, final OnLoginFinishedListener listener) {

        new Handler().postDelayed(new Runnable() {
            @Override public void run() {
                boolean error = false;
                if (TextUtils.isEmpty(username)){
                    listener.onUsernameError();//model层里面回调listener
                    error = true;
                }
                if (TextUtils.isEmpty(password)){
                    listener.onPasswordError();
                    error = true;
                }
                if (!error){
                    listener.onSuccess();
                }
            }
        }, 2000);
    }
}
```

2.2 view层描述和具体代码

负责显示数据、提供友好界面跟用户交互就行。MVP下Activity和Fragment以及View的子类体现在了这一 层，Activity一般也就做加载UI视图、设置监听再交由Presenter处理的一些工作，所以也就需要持有相应Presenter的引用。本层所需要做的操作就是在每一次有相应交互的时候，调用presenter的相关方法就行。（比如说，button点击）

```
package com.nsu.edu.androidmvpdemo.login;

/**
 * Created by Anthony on 2016/2/15.
 * Class Note:登陆View的接口，实现类也就是登陆的activity
 */
public interface LoginView {
    void showProgress();

    void hideProgress();

    void setUsernameError();

    void setPasswordError();

    void navigateToHome();
}
```

```
package com.nsu.edu.androidmvpdemo.login;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.Toast;

import com.nsu.edu.androidmvpdemo.R;

/**
 * Created by Anthony on 2016/2/15.
 * Class Note:MVP模式中View层对应一个activity，这里是登陆的activity
 */
public class LoginActivity extends Activity implements LoginView, View.OnClickListener {

    private ProgressBar progressBar;
    private EditText username;
    private EditText password;
```

```
    private LoginPresenter presenter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        progressBar = (ProgressBar) findViewById(R.id.progress);
        username = (EditText) findViewById(R.id.username);
        password = (EditText) findViewById(R.id.password);
        findViewById(R.id.button).setOnClickListener(this);

        presenter = new LoginPresenterImpl(this);
    }

    @Override
    protected void onDestroy() {
        presenter.onDestroy();
        super.onDestroy();
    }

    @Override
    public void showProgress() {
        progressBar.setVisibility(View.VISIBLE);
    }

    @Override
    public void hideProgress() {
        progressBar.setVisibility(View.GONE);
    }

    @Override
    public void setUsernameError() {
        username.setError(getString(R.string.username_error));
    }

    @Override
    public void setPasswordError() {
        password.setError(getString(R.string.password_error));
```

```
    }

    @Override
    public void navigateToHome() {
// TODO        startActivity(new Intent(this, MainActivity.class));
        Toast.makeText(this,"login success",Toast.LENGTH_SHORT).show();
//        finish();
    }

    @Override
    public void onClick(View v) {
        presenter.validateCredentials(username.getText().toString(), password.getText().toString());
    }

}
```

2.3 presenter层描述和具体代码

Presenter扮演着view和model的中间层的角色。获取model层的数据之后构建view层；也可以收到view层UI上的反馈命令后分发处理逻辑，交给model层做业务操作。它也可以决定View层的各种操作。

```
package com.nsu.edu.androidmvpdemo.login;

/**
 * Created by Anthony on 2016/2/15.
 * Class Note:登陆的Presenter 的接口，实现类为LoginPresenterImpl，完成登陆的验证，以及销毁当前view
 */
public interface LoginPresenter {
    void validateCredentials(String username, String password);

    void onDestroy();
}
```

```
package com.nsu.edu.androidmvpdemo.login;

/**
 * Created by Anthony on 2016/2/15.
```

```
 * Class Note:
 * 1 完成presenter的实现。这里面主要是Model层和View层的交互和操作。
 * 2  presenter里面还有个OnLoginFinishedListener，
 * 其在Presenter层实现，给Model层回调，更改View层的状态，
 * 确保 Model层不直接操作View层。如果没有这一接口在LoginPresenterImpl实现的话，
 * LoginPresenterImpl只 有View和Model的引用那么Model怎么把结果告诉View呢？
 */
public class LoginPresenterImpl implements LoginPresenter, OnLoginFinishedListener {
    private LoginView loginView;
    private LoginModel loginModel;

    public LoginPresenterImpl(LoginView loginView) {
        this.loginView = loginView;
        this.loginModel = new LoginModelImpl();
    }

    @Override
    public void validateCredentials(String username, String password) {
        if (loginView != null) {
            loginView.showProgress();
        }

        loginModel.login(username, password, this);
    }

    @Override
    public void onDestroy() {
        loginView = null;
    }

    @Override
    public void onUsernameError() {
        if (loginView != null) {
            loginView.setUsernameError();
            loginView.hideProgress();
        }
    }
```

```
    @Override
    public void onPasswordError() {
        if (loginView != null) {
            loginView.setPasswordError();
            loginView.hideProgress();
        }
    }

    @Override
    public void onSuccess() {
        if (loginView != null) {
            loginView.navigateToHome();
        }
    }
}
```

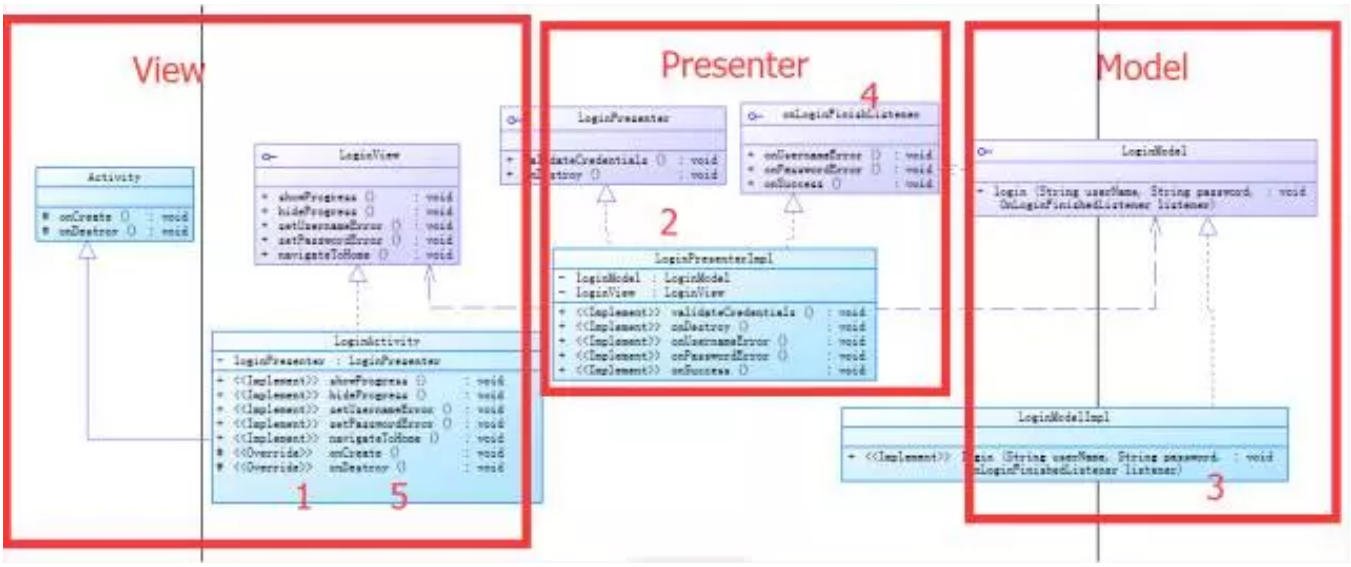
2.4 登陆的回调接口

```
package com.nsu.edu.androidmvpdemo.login;

/**
 * Created by Anthony on 2016/2/15.
 * Class Note:登陆事件监听
 */
public interface OnLoginFinishedListener {

    void onUsernameError();

    void onPasswordError();

    void onSuccess();
}
```

demo的代码流程：（请参考下面的类图）

1 Activity做了一些UI初始化的东西并需要实例化对应LoginPresenter的引用和实现LoginView的接口，监听界面动作
2 登陆按钮按下后即接收到登陆的事件，在onClick里接收到即通过LoginPresenter的引用把它交给LoginPresenter处理。LoginPresenter接收到了登陆的逻辑就知道要登陆了

3 然后LoginPresenter显示进度条并且把逻辑交给我们的Model去处理，也就是这里面的LoginModel，（LoginModel的实现类LoginModelImpl），同时会把OnLoginFinishedListener也就是LoginPresenter自身传递给我们的Model（LoginModel）。
4 LoginModel处理完逻辑之后，结果通过OnLoginFinishedListener回调通知LoginPresenter
5 LoginPresenter再把结果返回给view层的Activity，最后activity显示结果
请参考这张类图：



本项目类图

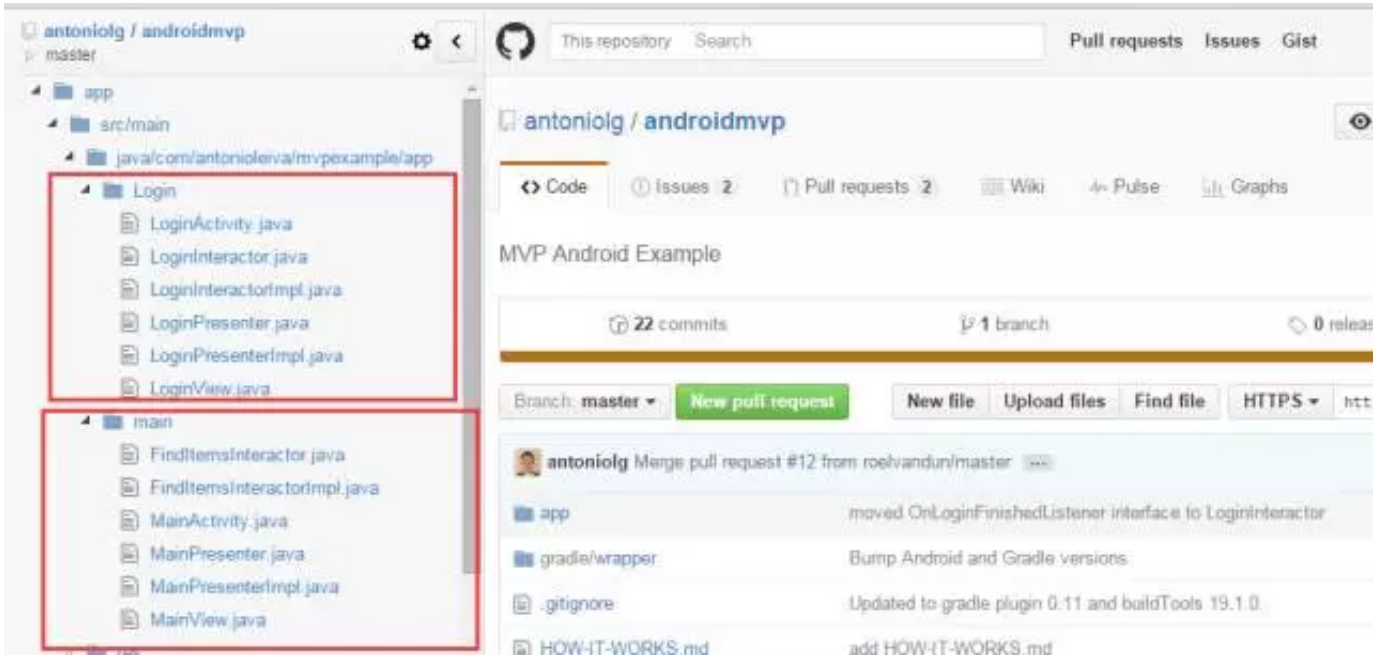
（3）注意：

3.1 presenter里面还有个OnLoginFinishedListener，其在Presenter层实现，给Model层回调，更改View层的状态，确保 Model层不直接操作View层。
3.2 在一个好的架构中，model层可能只是一个领域层和业务逻辑层的入口，如果我们参考网上比较火的Uncle Bob clean architecture model层可能是一个实现业务用例的交互者，在后续的文章中应该会涉及到这方面的问题，目前能力有限。暂时讲解到这里

（4）MVP经典参考资料

请直接参考文章，这里面有很多的mvp模式的学习资料：

- android架构合集（请关注github，后续会不断更新）
- android mvp github地址（本篇博客正是参考这个项目进行讲解的。这个项目也很简单，分为login和main两个模块，总共十个类，思路非常清晰。学习的朋友可以直接clone查看源码。）



androidmvp 的src代码分为login和main两个模块

本项目为了简单操作,只添加了login模块

本项目github地址：
https://github.com/CameloeAnthony/AndroidMVPDemo

阅读原文