Bison的博客

成功就是比你更天才的人，还比你更努力!

目录视图

Android MVP实战——MVP在Android中的简单应用

标签： android mvp

2015-08-12 16:02 43人阅读 评论(0)

分类： Android进阶（1）

目录(?) [+]

导读：上一章我们初探了Android MVP，但是只涉及到一些概念性的东西，这一章，我们将来一起来一步步实现一个简单的MVP的Dem

回顾

上章我们说到如何抽离MVP的层？还记得大致的步骤么？

- 创建实体JavaBean对象.
- 抽离出View的接口，即ViewInterface.
- 抽离Model的接口，即ModelInterface.
- 实现Presenter层.
- 实现ModelInterface与ViewInterface.

好了，我们就先按照这个步骤来吧。首先：

需求

想来想去，最后决定还是用登陆这个比较典型的栗子。如下：

- 用户界面接收用户名密码，点击登录按钮，出现进度条，进行接口请求.
- 登陆成功，进度条消失，跳转UI.
- 登陆失败，进度条消失，停留并显示失败原因

需求很简单，那么下面开始实战吧！

实现

创建项目MVPDemo，这里我选择的开发环境是AndroidStudio.

下面先看一下包结构：

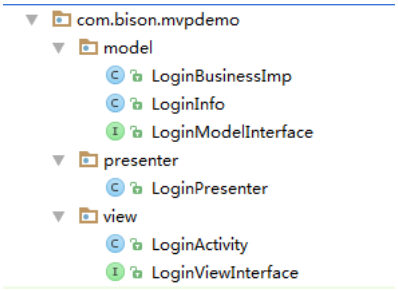
个人资料

bsince

关注 发私信

访问： 90次
积分： 20
等级： BLOG 1
排名： 千里之外

原创： 2篇 转载： 0篇
译文： 0篇 评论： 0条



按照MVP的通用结构，将其分为了三层，那么接下来是布局文件，这没什么好说的，大家大致过一下就好：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="@dimen/activity_horizontal_margin'
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">

<EditText
    android:layout_width="match_parent"
    android:id="@+id/userName"
    android:hint="userName"
    android:layout_height="wrap_content" />
<EditText
    android:layout_width="match_parent"
    android:id="@+id/passWord"
    android:hint="passWord"
    android:layout_height="wrap_content" />


<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:onClick="onAction"
    android:text="登录"/>

</LinearLayout>
```

就是简单的两个输入框，一个用于登录的Button.接下来，我们再来按照昨天说的步骤，开始MVP的应用。

- **创建JavaBean对象**.这个相信对大家没有什么难度。这里根据登陆的需求，我创建了一个**LoginInfo**实体。代码如下：

```
public class LoginInfo {
    private String userName;
    private String passWord;


    public LoginInfo(String userName, String passWord) {
        this.userName = userName;
        this.passWord = passWord;
    }

    public String getUserName() {

        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getPassWord() {
        return passWord;
    }

    public void setPassWord(String passWord) {
        this.passWord = passWord;
```

```
    }
}
```

- **抽离出View的接口.** 思考一下，昨天我们说了，ViewInterface是负责交互的,那么登录这个需求中涉及到UI交互的有哪些呢？界面交互需要提供哪些界面元素有哪些呢？这么一想，我们不难得出：

  1. 点击登录，即Button的点击监听事件.
  2. 进度条的显示与隐藏.
  3. 用户名和密码的获取（必要的界面元素）.

以下，贴出ViewInterface的抽离代码，取名为LoginViewInterface.

```
public interface LoginViewInterface {
    String getUserName();

    String getPassword();

    void showProgress();

    void dismissProgress();

    void showLoginFail(String error);

    void goToActivity();
}
```

再次回顾一下，我们可以发现，在LoginViewInterface中，出现的接口都是与交互，与UI，与界面元素有关的。

- **抽离Model的接口.**这里虽然说的是抽离model，但这里的Model层抽离出来的东西，其实是用于数据流的输入与输出。说得直白点我们的JavaBean来说的，是用来提供javaBean对象的业务层,怎么理解呢？可以这么想，它就是为Presenter层提供所操作javaBean的登录，那么在ModelInterface里面它只负责提供是否登录成功的数据流以及登录成功后的用户数据，登录失败的错误原因等。无论你是从网络获取还是从本地数据库读，你只要能提供这些数据就好。它并不包括对登录成功与失败的处理逻辑。那么一切就

```
public interface LoginModelInterface {

    void login(String userName,String password,LoginPresenter.CallBack callBack);
}
```

因为这里登录是属于耗时操作，所以我们在Presenter层定义了一个内部回调接口，用于在异步线程中回调。代码如下

```
public interface CallBack{

      void onSuccess(LoginInfo info);

      void onFail(Throwable error);
}
```

- **实现Presenter**,这个不难，我们的登录需求，处理逻辑其实就只有Login这一个。所以只需要实现login相关的逻辑处理就行了。则

```
public class LoginPresenter {

    private final LoginModelInterface loginBusiness;

    private final LoginViewInterface loginViewInterface;

    public LoginPresenter(LoginViewInterface viewInterface){
        this.loginBusiness = new LoginBusinessImp();
        this.loginViewInterface = viewInterface;
    }

    public void login(){
        String userName = loginViewInterface.getUserName();
```

```
        String password  = loginViewInterface.getPassword();
        if(TextUtils.isEmpty(userName)||TextUtils.isEmpty(password)){
            loginViewInterface.showLoginFail("请完善登录信息");
            return;
        }
        loginViewInterface.showProgress();
        loginBusiness.login(userName, password, new CallBack() {
            @Override
            public void onSuccess(LoginInfo info) {
                loginViewInterface.dismissProgress();
                loginViewInterface.goToActivity();
            }

            @Override
            public void onFail(final Throwable error) {
                loginViewInterface.dismissProgress();
                loginViewInterface.showLoginFail(error.getMessage());
            }
        });
    }

    public interface CallBack{

        void onSuccess(LoginInfo info);

        void onFail(Throwable error);
    }
}
```

- **实现ModelInterface与ViewInterface.**这一步就更较简单了。既然接口都抽离出来了，实现还会难吗？直接贴代码了

```
public class LoginBusinessImp implements LoginModelInterface {
    @Override
    public void login(final String userName, final String password, final LoginPresenter.Ca
        new Thread(new Runnable() {
            @Override
            public void run() {
                //进行耗时操作，进行网络请求
                try {
                    Thread.sleep(3000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                //模拟验证。用户名密码相同，我们定为校验失败，否则校验成功
                if(!userName.equals(password)){
                    LoginInfo info = new LoginInfo(userName,password);
                    callBack.onSuccess(info);
                }else{
                    callBack.onFail(new Exception("用户名或密码错误"));
                }
            }
        }).start();
    }
}
```

这里说明一下，有些朋友可能会疑惑，你不是说ModelInterface只需要提供需要的数据就行了吗？登录业务的话，那这里只需要在登录

个成功的状态码，再加上一个LoginInfo给Presenter层，登录失败返回错误异常就行了，不需要进行逻辑判断。但你这里明显是用if else

好吧，这是因为我们这里是模拟登录。也就是说在实际开发中，判断登录成功与失败是在服务器中进行的，我们这儿的逻辑也是模拟

而且，MVP的根本目的在于减轻Activity或Fragment的当量，即使有一些简单的逻辑处理混在里面倒也无需计较。

最后还有我们的Activity:

```
public class LoginActivity extends AppCompatActivity implements LoginViewInterface {

    private EditText userNameEdit;
    private EditText pwdEdit;
    private ProgressDialog dialog;
```

```
    private LoginPresenter loginPresenter = new LoginPresenter(this);

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

    private void initViews() {

        userNameEdit = (EditText) findViewById(R.id.userName);
        pwdEdit = (EditText) findViewById(R.id.passWord);
    }

    public void onAction(View v){
        loginPresenter.login();
    }


    @Override
    public String getUserName() {
        return userNameEdit.getText().toString().trim();
    }

    @Override
    public String getPassword() {
        return pwdEdit.getText().toString().trim();
    }

    @Override
    public void showProgress() {
        if(dialog==null){
            dialog = ProgressDialog.show(this,"","loading...");
        }else{
            if(!dialog.isShowing()){
                dialog.show();
            }
        }
    }

    @Override
    public void dismissProgress() {
        if(dialog!=null&&dialog.isShowing()){
            dialog.dismiss();
        }
    }

    @Override
    public void showLoginFail(final String error) {
       runOnUiThread(new Runnable() {
           @Override
           public void run() {
               Toast.makeText(LoginActivity.this, error, Toast.LENGTH_SHORT).show();
           }
       });

    }

    @Override
    public void goToActivity() {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                Toast.makeText(LoginActivity.this, "登录成功,跳转HomeActivity", Toast.LENGTH_S
            }
        });

    }
}
```

这里跳转与错误信息提示，我都偷懒直接用的Toast了。另外为防止子线程刷新UI崩溃，这里作了一下处理。当然也可以Activity中不做

Presenter层通过Har ;UI更新，我这里是只是图个便捷。

好了，到这里这么一个简单的Demo便完成了，功能比较单一，也就不贴效果图了。回顾一下，有的朋友可能会说，这么一个简单的登
么多类，而实际开发中一个界面涉及的业务何其之多，使用MVP真的现实吗？其实，这是可以的，并不是虚妄，而且实质上，在熟练
增加多少工作量，反而会让你在以后的维护工作中轻松自如。有兴趣的同学，可以看一下 chrisbanes的开源项目philm,其整体架构就是
下面附上本文中的Demo源码

源码下载

- 上一篇 Android MVP 探究与总结

我的同类文章

Android进阶（1）

- Android MVP 探究与总结 2015-08-11 阅读 49

猜你在找

查看评论

暂无评论

核心技术类目