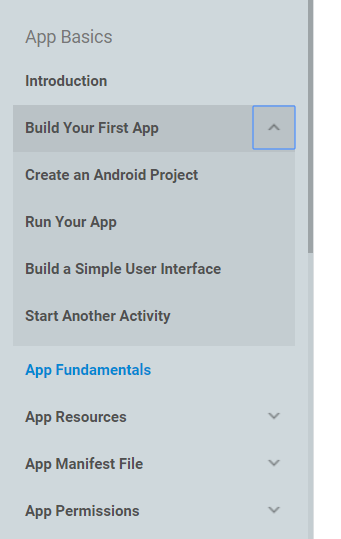
安卓入门笔记

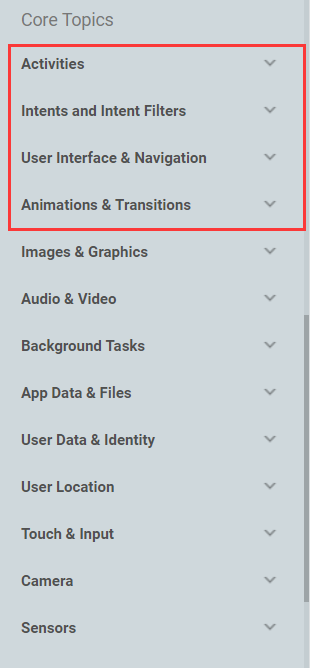
# 基础

## 基础入门

基础入门是一些概念、基本用法等，最好的资料就是官网的文档。

基础内容，包含了安卓的四大组件的介绍，组件调用，创建自定义组件，资源文件，动态权限等。





主要概念里面，详细介绍了各个组件的详细用法，用户的导航、布局、ui组件等。

可以加入组<https://groups.diigo.com/group/dhcjrbfeat>，里面有一些阅读笔记。

官方的指导网址<https://developer.android.com/guide/components/fundamentals.html>

基础里学习写的工程我放到github上了，https://github.com/Dickkk/activityGallery

## 界面设计到app页面

界面设计 一般使用sketch，sketch有丰富的插件，可以生成android代码，也可以切图。

墨刀，国内的一款app设计软件，包含很多app素材，使设计变得很简单。

怎么从界面设计到android studio中生成，可以参考这套视频，

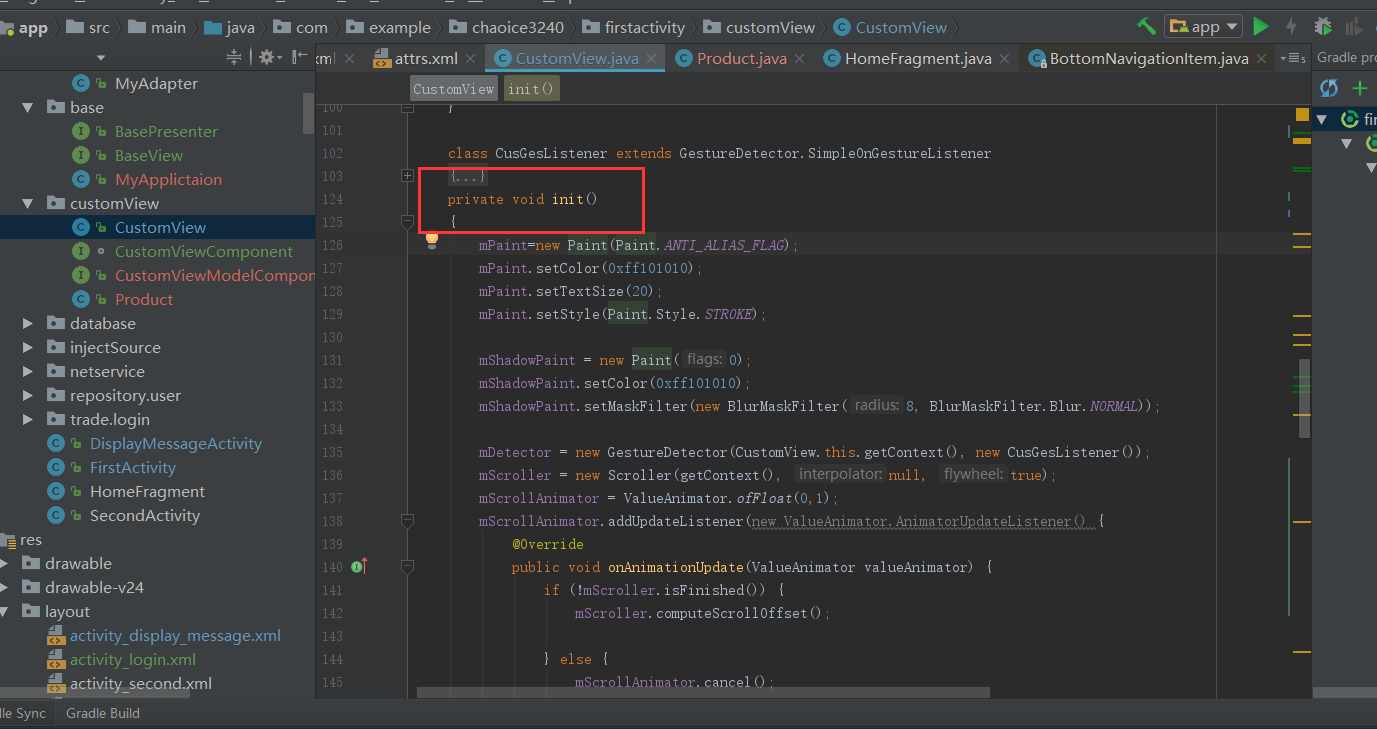
<https://www.youtube.com/playlist?list=PL6j62WXKkENq1pM1vqv1P9abb99ttMiIZ>

从UI原型设计，到app页面开发的流程。

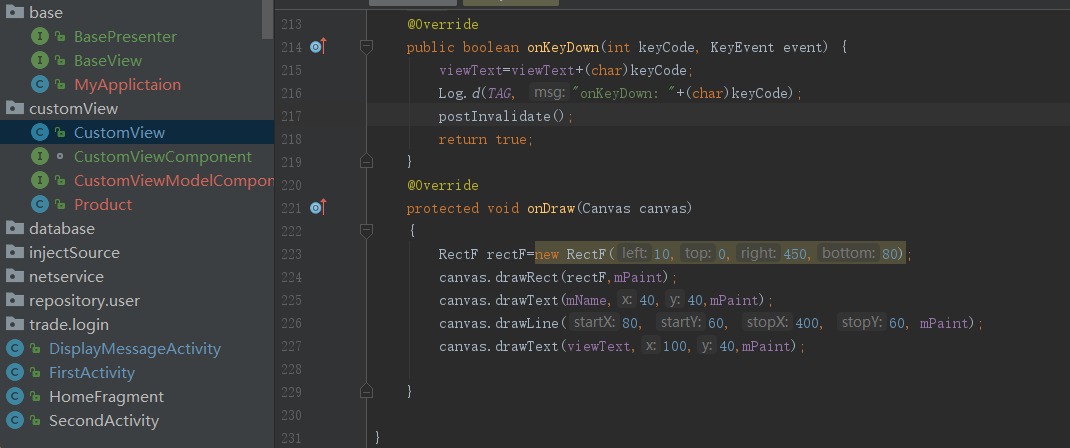
需求跟踪界面设计： https://modao.cc/app/1unRq2CyC0PXpw1wMJMgE14HqB54za6

## 自定义view

所有的ui组件都继承自view，init中进行画笔啥的一些初始化：



Ondraw进行一些自定义的绘制。



Ontouchevent进行一些拖动，以及一些动画处理。



## 自定义fragment

Fragment如果要传参数，不能改造构造函数，必须使用静态方法，通过bundle来初始化。



# 架构

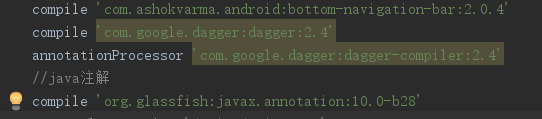
## Dragger2

### Dragger2是什么

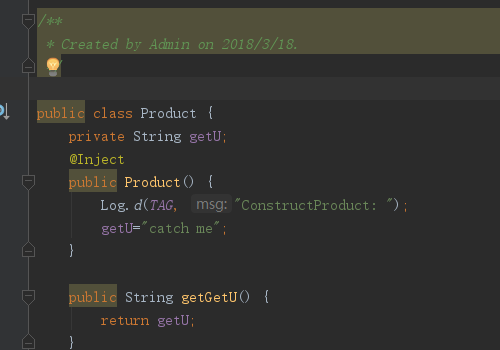
Dragger2解决依赖注入问题， 两个类强依赖导致的违背单一职责问题。Dragger2通过自动生成中间代码实现，我们通过注解标注之后，重新编译项目，会在out里生成中间类，在代码中就可以使用这些中间类。

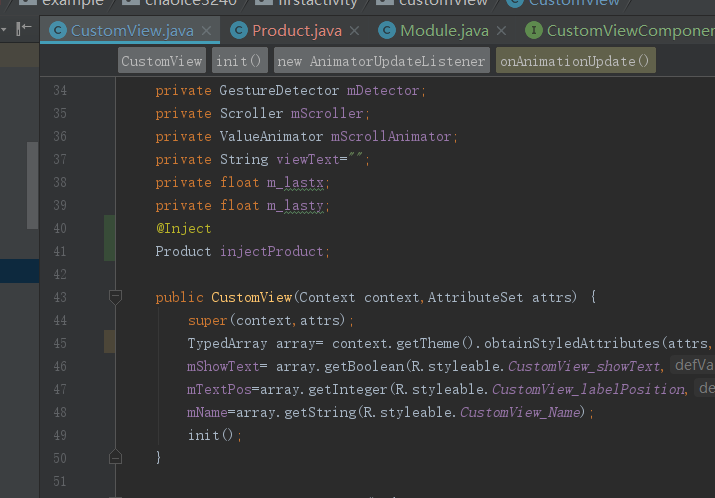
### Dragger2的使用

添加引用

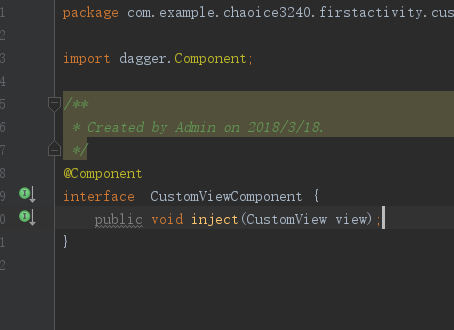


1. 通过标注@inject标注 依赖类与目标类，依赖类通过注解构造器提供。

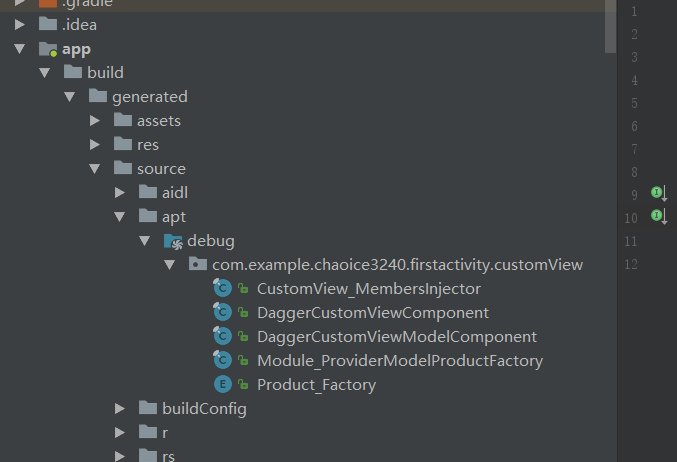




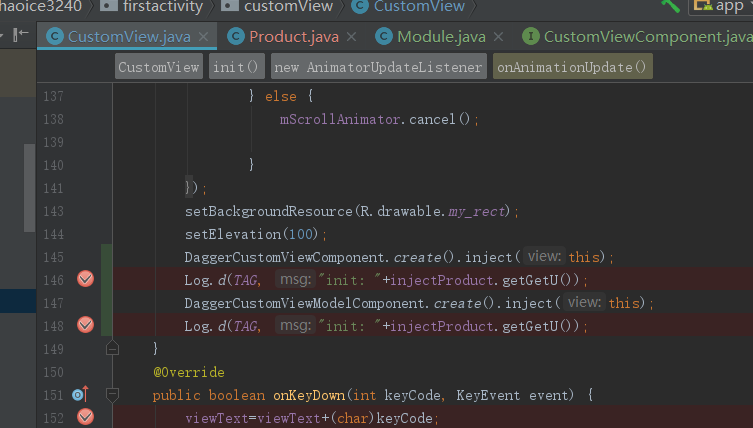
1. 编写注解的xxcomponent。声明目标注入方法。



1. 重新编译项目生成中间component，Draggerxxcomponent

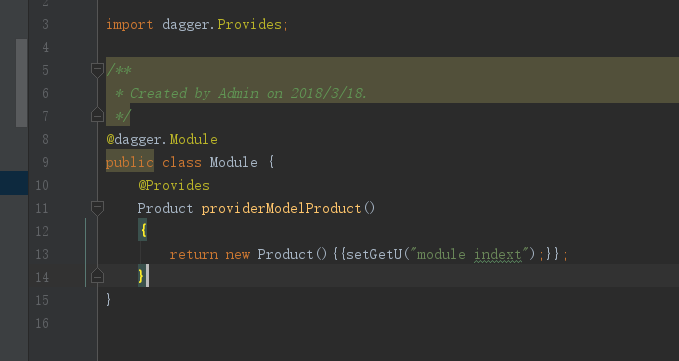


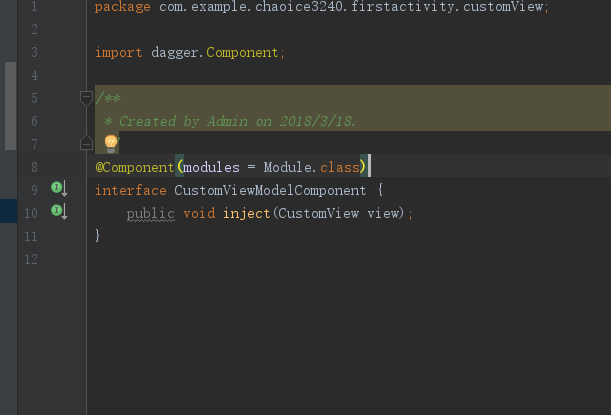
1. 在需要注入的类中调用draggerxxcomponent的inject方法。



对于不是我们写的类，我们不能通过在构造器里添加注解的方式将我们的类暴露给依赖类。

这时可以使用module注解到某个类，添加provider注解的函数暴露，同时需要告诉component我依赖的models是啥。





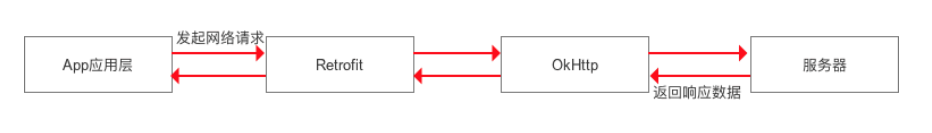
### 总结：

Dragger2是一个依赖注入的工具，他解决了创建对象的时机问题，类似spring里的容器，区别是spring创建的东西都在spring容器，是在启动的时候都建好，在需要注入的时候从容器找。 而dragger是通过component手动注入，需要我们写注入接口，指定modules。

## Retrofit2

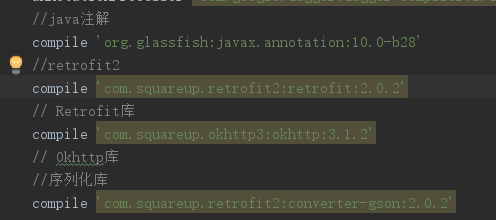
### 什么是retrofit2

网络框架，支持okhttp3 、 restfull、Gson序列化、rxjava

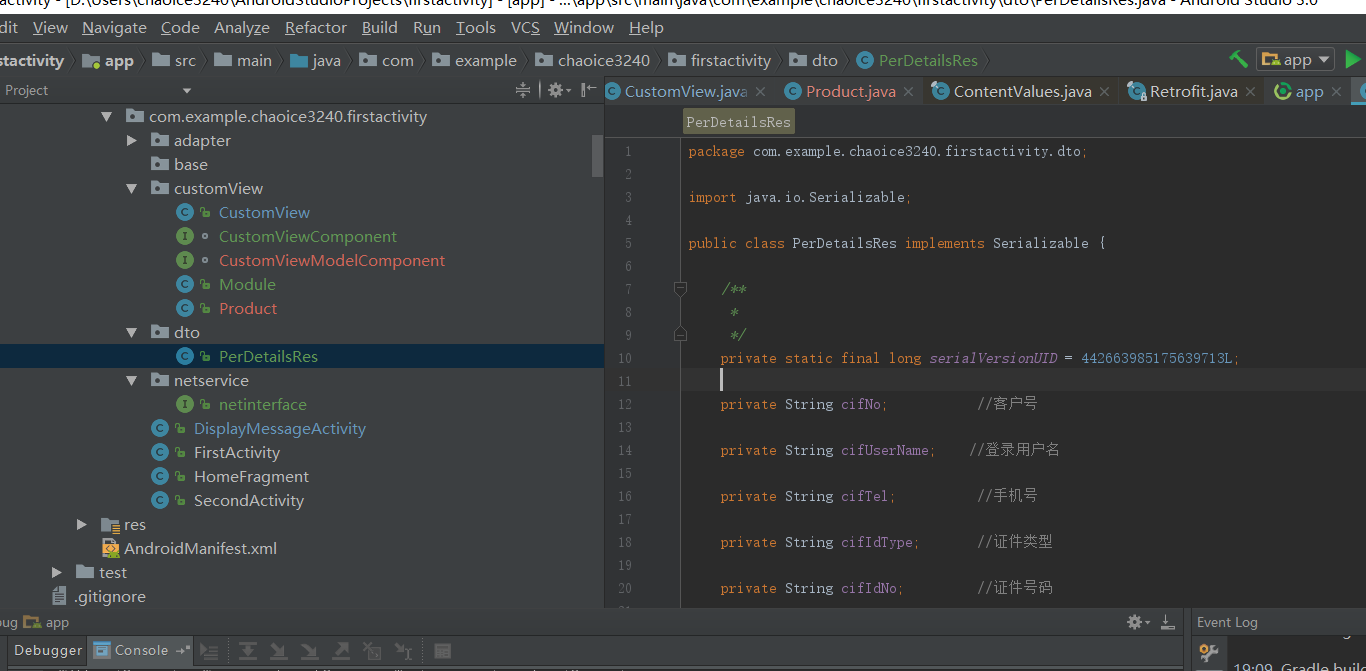


### Retrofit2使用

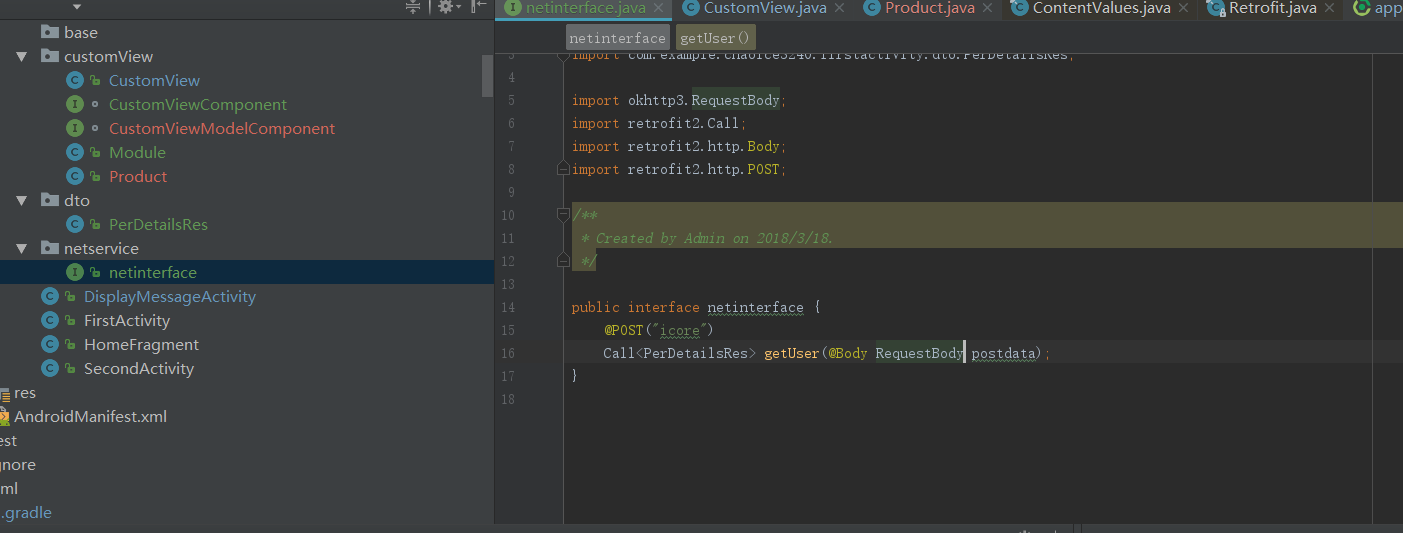
1. 添加依赖



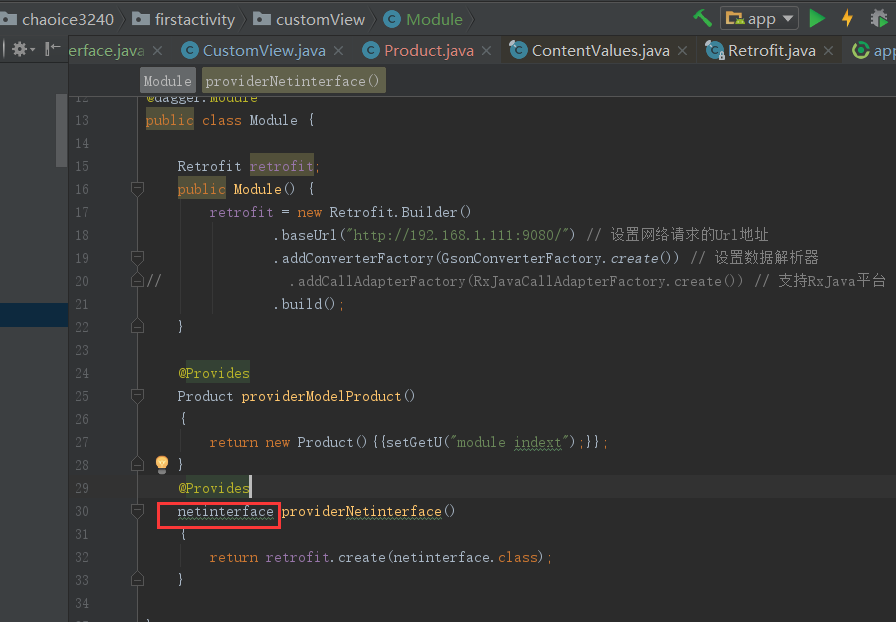
1. 创建返回实体



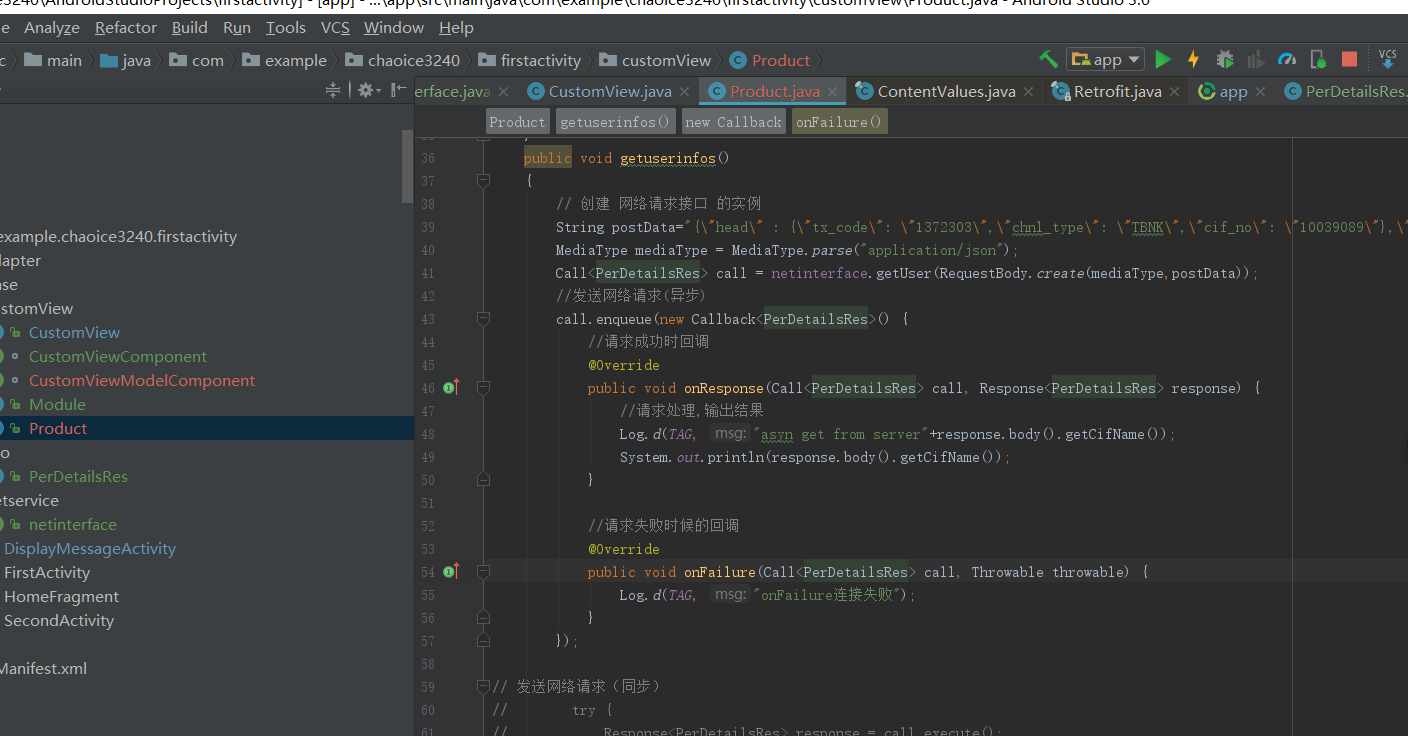
1. 创建网络请求接口



1. 创建retrofit2和网路请求接口实例



1. 发送网路请求，并处理返回数据



### 总结

Retrofit2是一个网络库，结合了okhttp3、gson，提供网络通信以及序列化、反序列化的工作。同时可以很好的融入rxjava，是网络通信库中比较快速、轻便的一个。

## 观察者模式

### 概念

观察者 （消费数据）

被观察者（数据提供，订阅者清单（观察者列表））

被观察者.订阅（观察者）完成 被观察者中的观察者列表的维护。

## Reactive programming

### 什么是reactive programming

反应式编程（Reactive Programming）是一种面向数据流和变化传播的编程范式，我的理解是面向stream的多线程调度模型。具体在rxjava中进行分析。

## Rxjava

### 什么是rxjava

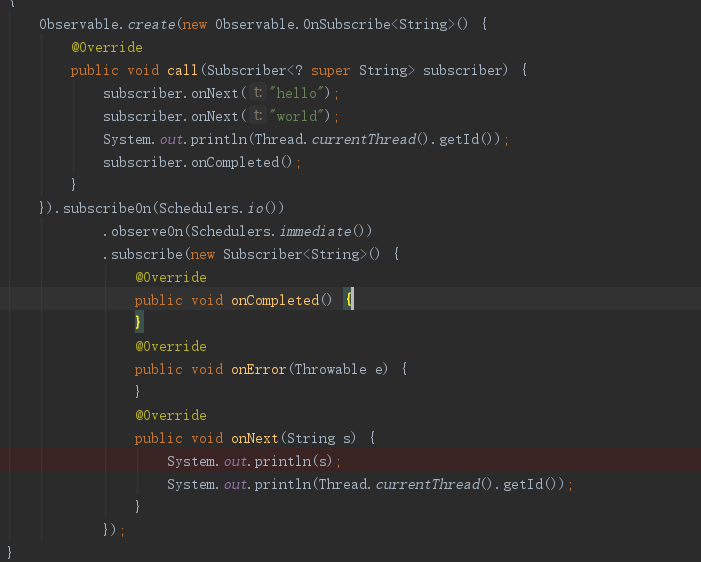
Rxjava是一个异步线程执行库，在进行一些耗时操作的时候放在rxjava的io线程进行处理，完成处理后放在主线程接受结果进行后续异步处理。因为android的主线程不能进行耗时操作，理论上主线程有5s没响应就会造成anr。

### Rxjava的使用

Rxjava的观察者分析， 怎么异步执行的。

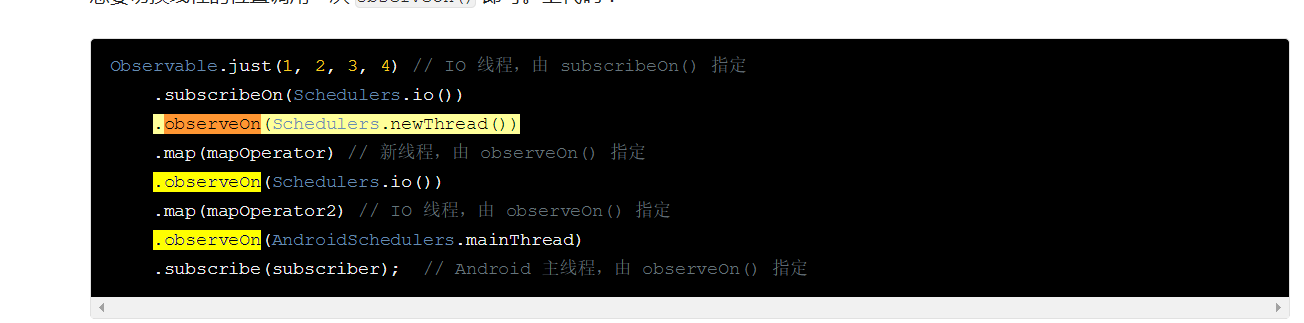
步骤

### 创建出 Observable 和 Subscriber ，再用 subscribe() 将它们串起来。



#### 使用异步调度器scheduler

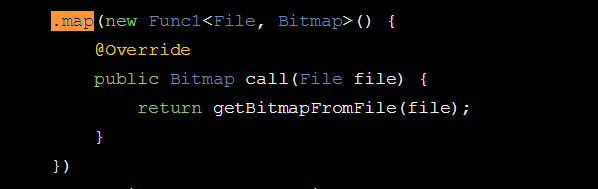
使用scheduleon 消息产生使用的线程 和 subscribeon函数制定消息消费使用的线程，这样将耗时的需要异步执行的放在io线程。



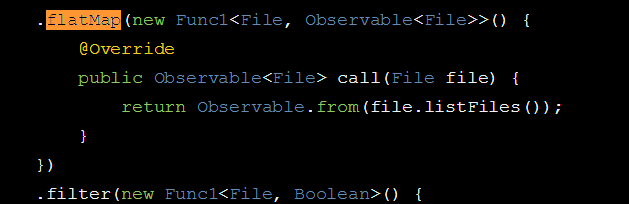
Subscribeon指定的是准备事件所运行的线程。 Observeon指定的是从当前到下面的所有操作的线程。

#### 使用事件变换map与flatmap

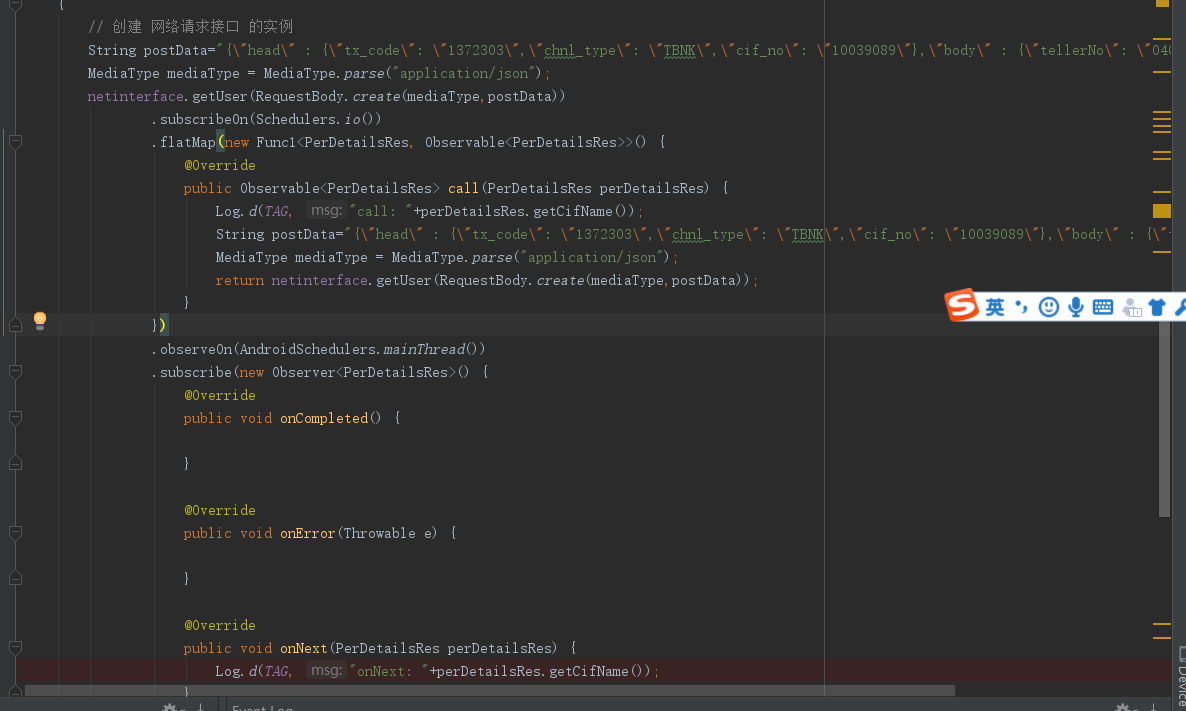
map将原有observable进行变换从类型A转变为类型B 处理1对1关系



flatmap将原有的observable进行变换从类型A经过处理转为observalbe<T> 处理1对1或者1对多关系。



#### Rxjava集成retrofit2



### 总结

Rxjava在安卓开发中流行，是因为它的简洁代码结构、能解决秘之缩进痛点、耗时操作异步处理。很好的被很多开发库支持，比如retrofit2,、rxbinding。

## Handler消息机制

### 消息机制是什么

为什么会有消息机制，因为activity这种ui控件是线程不安全的，必须由ui线程（主线程）来更新界面，这就牵扯到线程间通信。Message handler messagequeue，looper的概念。

Message是消息，消息中包含消息内容，包含handler的引用。

Messagequeue是消息队列，存放消息。

Handler是消息处理类，负责消息的发送（send、post）以及消息的处理（dispatch）重写handler的时候需要重写handlemessage函数。包含looper的引用。

Looper 是循环器，负责循环messagequeue，取出message，根据message的handler调用handler的dispatch，handler进行handlemessage处理。

Looper类提供了很多静态方法，同时内部有个线程变量，存放threadlocal的looper对象。也就是说每个线程只有一个looper对象。当调用Looper.prepare的时候才创建线程变量Looper对象实例，同时looper中新增mq，并设置当前线程。

### 使用

Looper.prepare();

mHandler = new Handler()

{

public void handleMessage(Message msg)

{

// process incoming messages here

}

};

Looper.loop();

分为三步，prepare来创建looper， 在线程中创建handler（会设置自己的queue以及looper），调用handler 发送消息到消息队列。

在activity中不需要创建looper线程变量，因为UI主线程中已经有looper了。之间创建handler就可以。

### 总结

消息机制是线程间通信的基础，需要对binder机制有了解。

## Repository 仓库模式

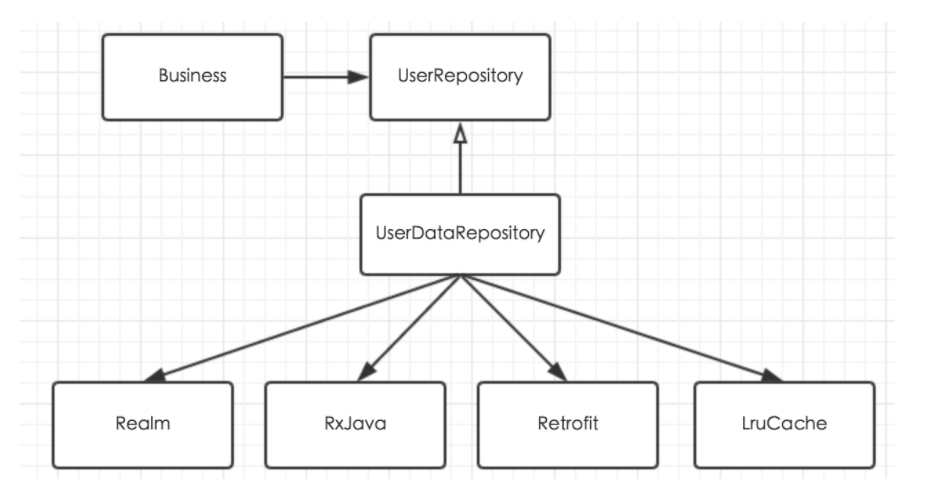
### 什么是仓库模式

屏蔽底层细节，面向接口注入。什么来自于网络、数据库，亦或是内存等等。如此，一来上层可以不用关心细节，二来底层可以根据需求修改，不会影响上层，两者的分离用可以帮助协同开发。是仓库模式。Present 或者usecase直接调用repository接口的方法。以后需要修改内容来源，只改repository的实现即可。将数据相关的逻辑使用接口进行依赖解耦，让数据repository自己负责数据，不会侵入present或者view。例如如下的场景

- 当现在是无网状态，我希望列表能直接显示上一次的数据，而不会是空页面。

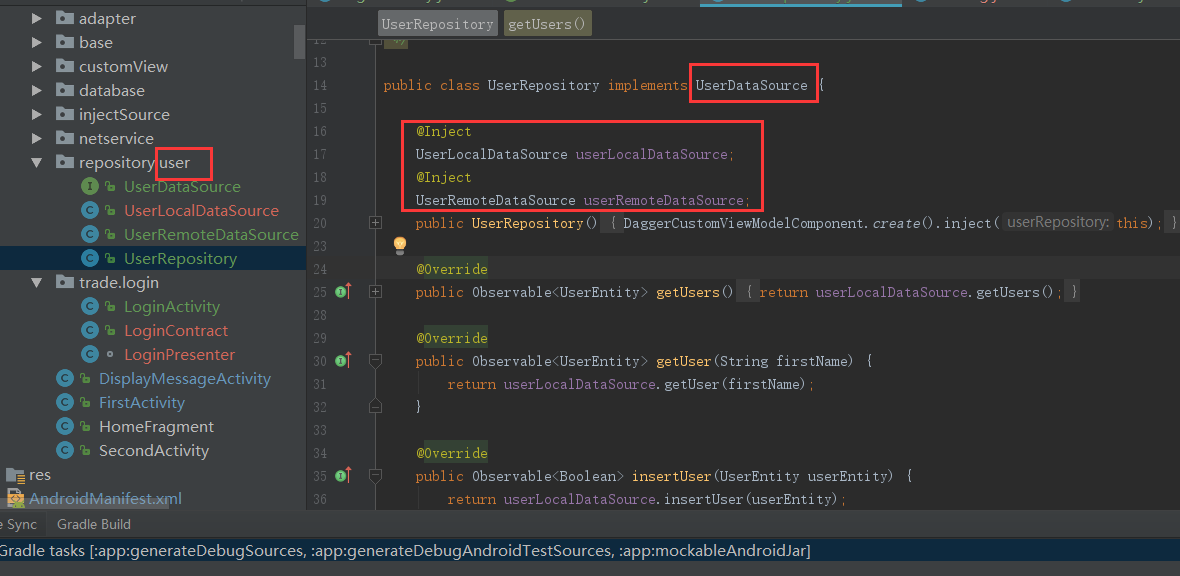
- 除非好友的用户数据过期（比如超过一天），否则希望直接使用本地缓存中的，但如果缓存没有，或者过期，则需要拉取并更新。

- 点赞后，即便请求还没发送或者没有收到response，仍然希望显示点赞后的状态。

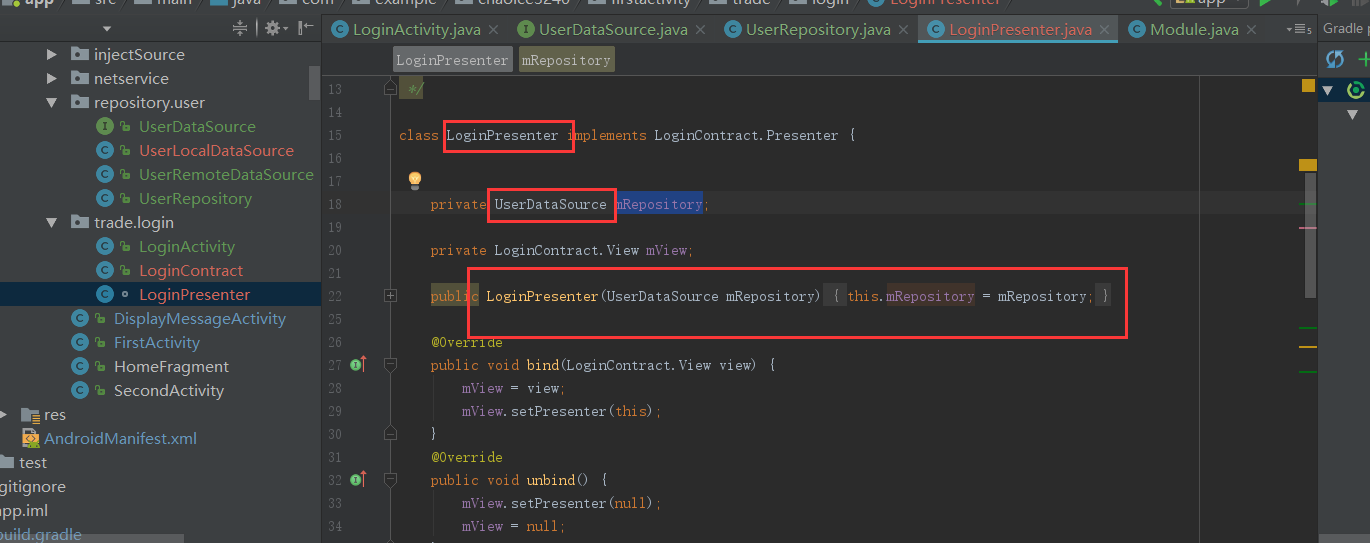


### 使用

1. 具体repository实现数据处理，调用不同的实现进行处理。



2.presenter 通过构造器注入repository接口。



### 总结

仓库模式，屏蔽了底层model的获取细节，repository会根据业务逻辑去网络、本地数据库、缓存中取数据。

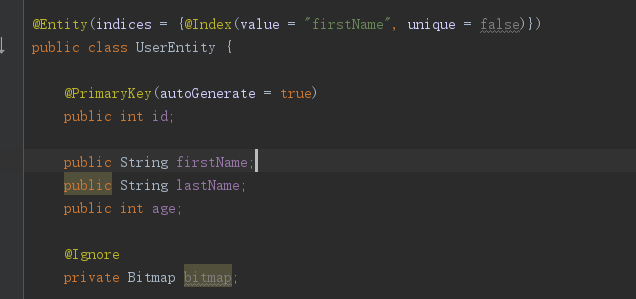
## Room库

### 什么是room库

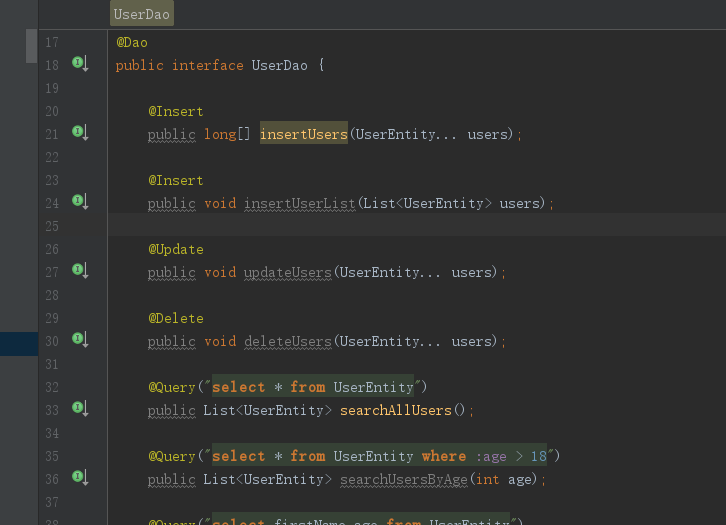
对本地sqlite数据库的api封装

### 使用

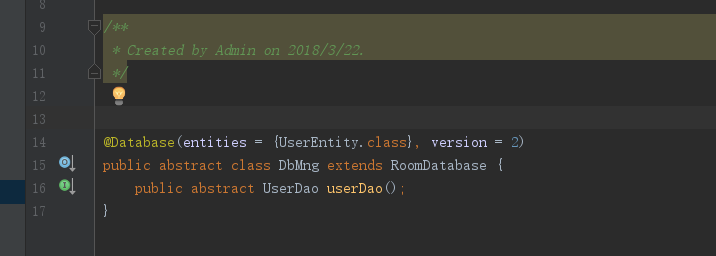
1、



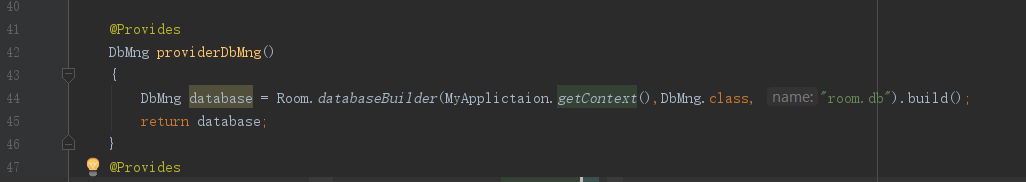
2、



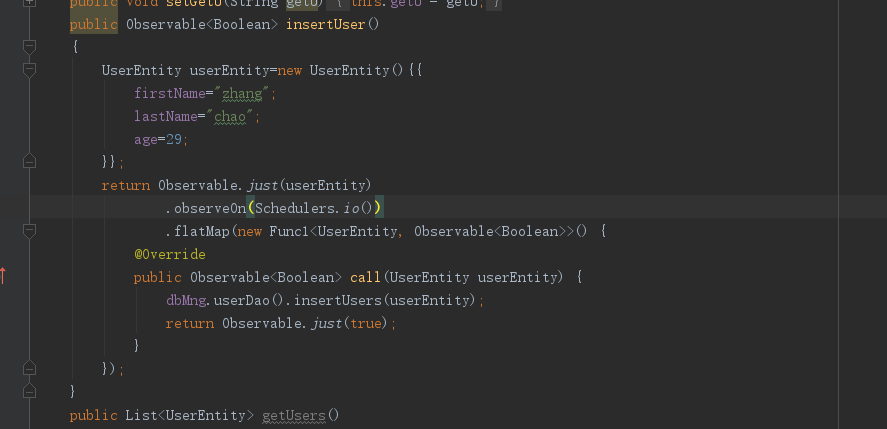
3、



4、



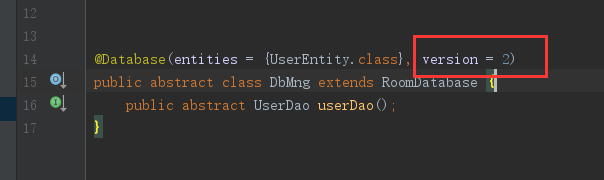
1. 在repository中写获取函数

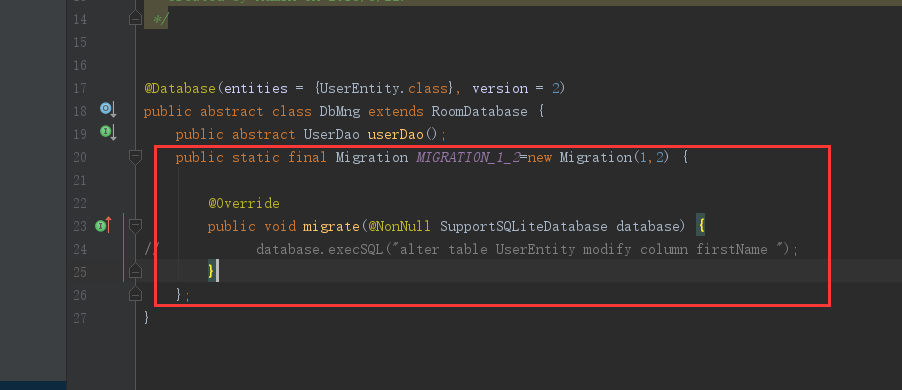


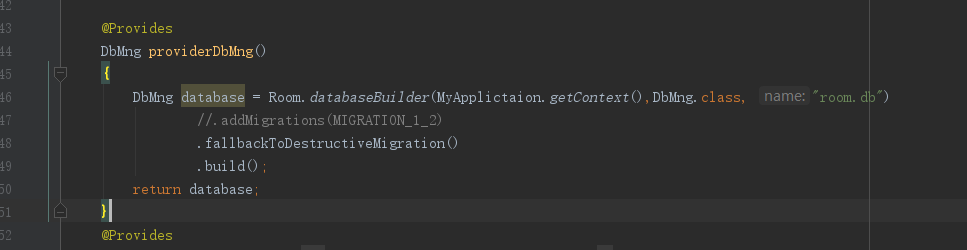
1. 在presenter层写处理函数



1. 当实体发生变更后，更新数据，需要修改数据库的版本，以及添加migration或者调用fallback







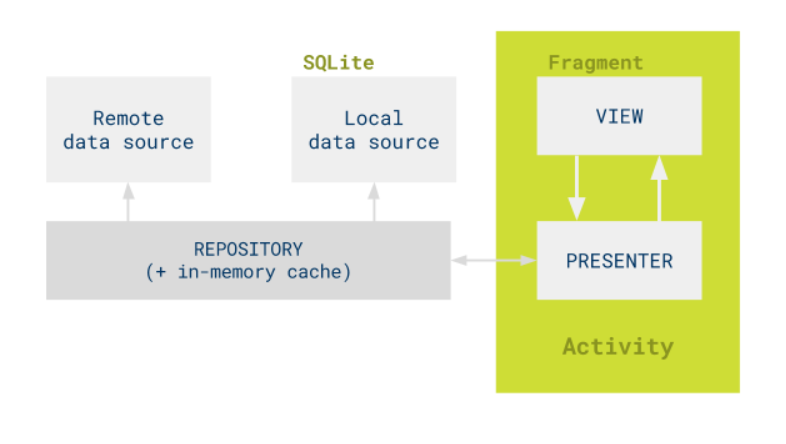
### 总结

Room是sqlite的门面工具。

## Mvp

### 什么是mvp

Mvp 模式 clean模式 repository模式。



与mvc相比，将所有的逻辑放到presenter中。 所有的数据都放在repository中。View只负责显示。

流程如下：

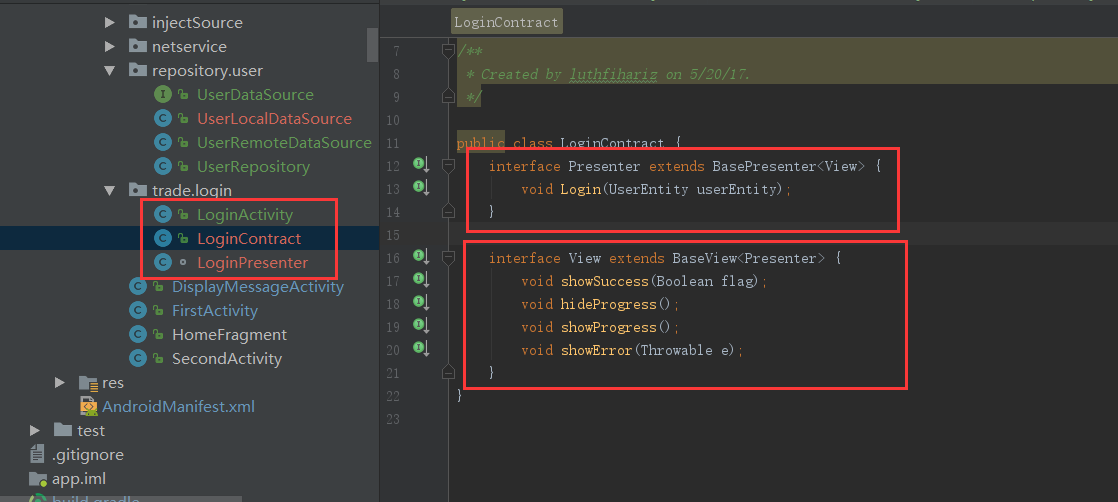
1、事件的触发发生在view，当某个按钮点击后，调用presenter的逻辑处理。

2、presenter会从repository获取数据，然后将数据返回给view。

3、view做展示或者跳转。

### 使用

1、项目结构，Mvp 将view presenter的接口定义到契约类logincontract中。View 与 presenter实现好契约接口方法。

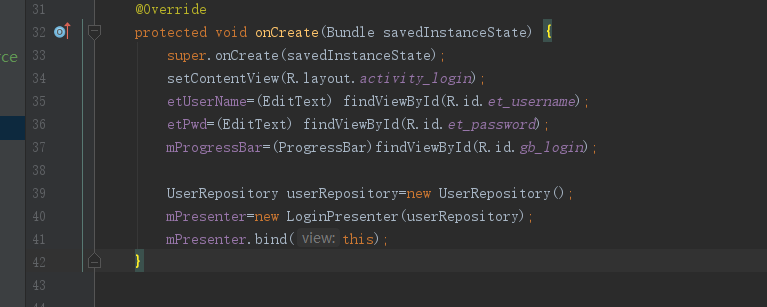


2、注入过程，

1、loginactivity实现view接口，负责view的展示。

2、activity在create的时候，创建repository与presenter，并将repository注入到presenter中。

3、presenter绑定到当前activity（view）。



3、运行流程，以登录为例。

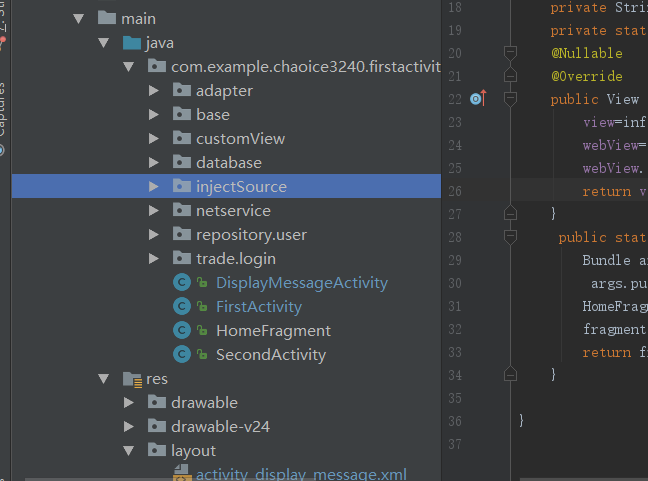
从登录按钮的事件处理函数往后看。

### 总结

Mvp 将展示、逻辑、数据获取分离开来， 可以让开发者在没有view的时候测试逻辑与数据。符合测试驱动开发的规范。

代码地址：

# 整体代码结构，开发交易流程



1. 在netservice中添加对后端的接口。
2. 在repository中添加需要的repository模型层。
3. 在trade中添加交易对应的contract、view、presenter。
4. 对presenter进行测试。

# 趋势

以往是h5+原生的混合开发方式，这种方式在内容多的时候就会造成程序卡顿等性能问题。

现在流行的方式是react native+原生的混合开发方式，虽然相比原生的性能也是比价弱但是比webview要强很多。

React native的优势

React native的劣势

# 质量组需求跟踪项目介绍

## 需求

1、 登录模块，两种用户银行需求人员，指定给总经理，总经理分配给产品经理，产品经理分配项目经理， 项目经理评工作量，项目经理制定技术人员，技术人员制定 需求开发时间表（设计时间，开发时间，自测时间，客户测试，上线时间）， 需求进入准备状态，不同阶段上传设计文档。提醒各阶段人员的任务完成状态。

2、项目模块，质量组超级用户，新建项目，从属于某个银行，包含哪些模块。

3、银行客户视图：

银行客户人员提需求，查看行内需求的进度。

4、需求模块：

需求提出：银行客户新增需求， 默认从属于当前银行客户的银行客户， 选择此银行对应的项目，模块。同时上传需求文档，需求描述，dealline等。

总经理分配阶段：分配给产品经理

产品经理分配阶段：分配给项目经理

项目经理分配技术人员：分配给多个开发人员

开发人员确认需求：确认需求阶段，开发人员确认设计、开发、自测、验收测试、上线的时间段。需求进入确认阶段

需求进行阶段：需求进行阶段，不同阶段都有响应的文档提交。领导可以进行查看，开发人员可以进行状态变动。有倒计时，提醒。同时可以推送app预警消息给项目经理、质量组人员、开发人员。

5、需求生命周期

A 后台会维护项目数据，包含

6、视图

根据银行客户的视图，项目个数。

根据银行项目的视图，需求数量、需求推迟情况、项目人员视图。

根据人的视图，参加的需求数量、参与的人月数、项目经理评分。

7、评价

项目完成开发人员自评，项目经理评分与意见。

## 设计页面

<https://modao.cc/app/1unRq2CyC0PXpw1wMJMgE14HqB54za6>

## demo地址

https://github.com/Dickkk/activityGallery