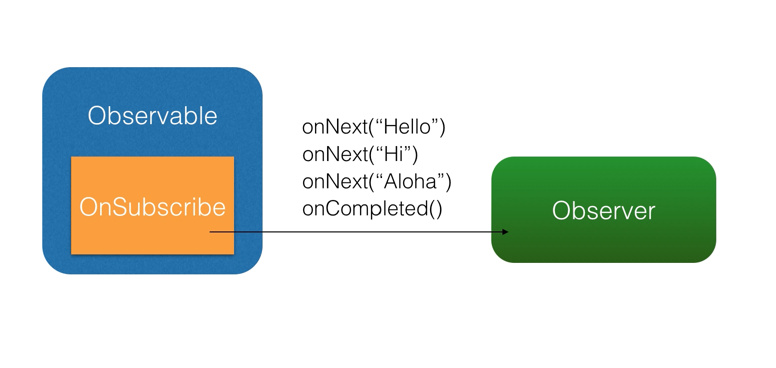
2.3 Rrd开发框架

Rrd开发框架是目前Android应用开发中，最主流的开发套件，包括Rxjava2,Retrofit2和Dagger2开发库，通过使用这套第三方API，能够优雅的解决开发中需要的异步问题，网络访问需求和为了实现各层次解耦而需要的依赖注入能力，这是快速，低耦合的开发Android程序最好的基础。

2.3.1 Rxjava2异步框架

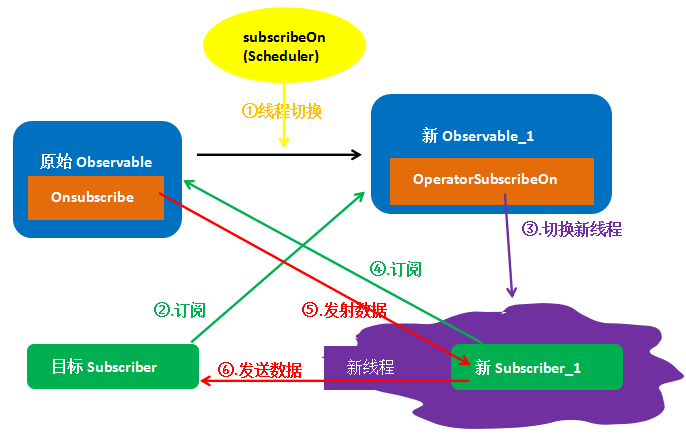
Rxjava2是一个响应式异步编程框架，通过链式语法极大的提高了java语言在处理异步问题时，线程调度困难和代码繁杂的问题。

RxJava2的异步实现，是通过一种扩展的观察者模式来实现的，RxJava2有四个基本概念：Observable (被观察者或者叫主题)、Observer (观察者)、subscribe (订阅器)和事件。Observable 和 Observer 通过 subscribe() 方法实现订阅关系，从而Observable可以在需要的时候发出事件来通知Observer。同时相比较传统意义上的观察者模式，RxJava的事件回调方法除了普通事件onNext()之外，还定义了两个特殊的事件：onCompleted() 和 onError()，以便在线程终止时，可以处理终了状态的业务逻辑。

 图xxx,Rxjava2异步事件处理流程图

Rxjava2最大的特色是可以在链式语法中，自然的实现线程调度，这在过去传统的JAVA代码中是一件无法想象的事情。

RxJava中使用observeOn和subscribeOn操作符，让Observable在一个特定的调度器上执行，observeOn指示一个Observable在一个特定的调度器上调用观察者的onNext, onError和onCompleted方法，subscribeOn则指示Observable将全部的处理过程（包括发射数据和通知）放在特定的调度器上执行。线程切换的大致流程如下图xxx



图xxx，Rxjava2线程切换示意图

Rxjava2在做线程切换时是依赖不同的调度器来实现的，Rxjava2中一共提供了6种不同的调度器来满足不同的场景

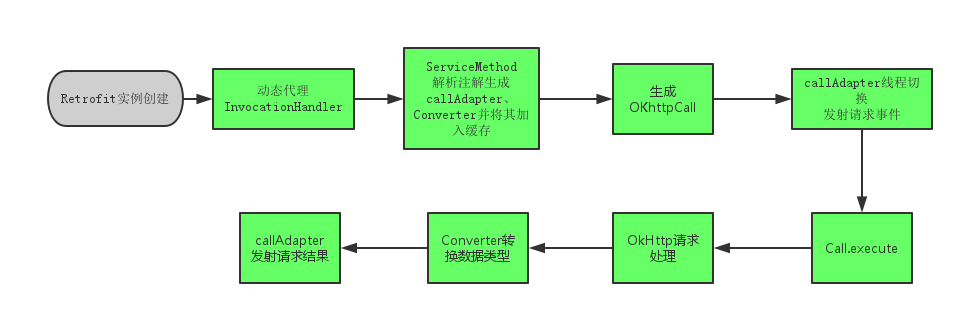


表xxx，Rxjava2中的调度器

2.3.2 Retrofit2

Retrofit2是一款支持Java和Android的RESTful风格的网络请求库。通过Retrofit2，可以把访问基于REST风格的Web服务变得相对容易且操作灵活。同时在Retrofit2中，可以配置任意用于数据序列化的转换器。一般情况下，通过JSON格式在客户端和服务端之间传递数据，但是也可以通过自定义转换器来处理XML或者其他任意格式的数据源。

在Retrofit2库中，默认使用okHttp这个第三方库来完成http请求和接受服务端的响应，同时还支持Rxjava2的响应式编程方式，让网络请求的代码变得优雅而灵活，一次通过Retrofit2框架发起的网络请求的流程如下图。



图xxx，Retrofit2网络请求流程

Retrofit2的核心三要素是

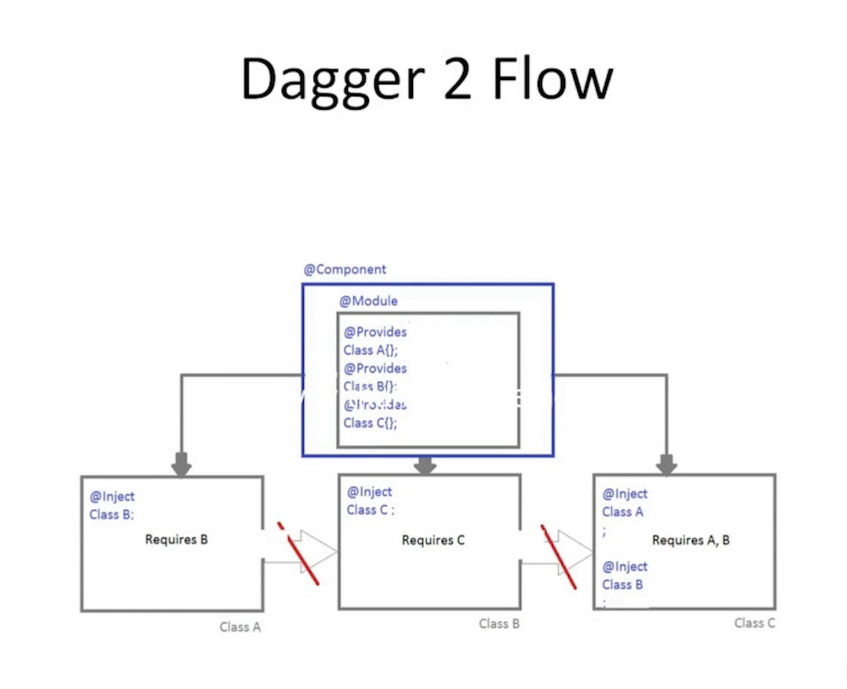
1. 数据对象，用来映射网络请求返回的JSON数据。
2. http接口，用来对应基于RESTful的网络请求。
3. Retrofit.Builder类，基于Retrofit2的注解解析器生成的请求接口的实例，在这里真正完成接口的请求和响应回调。

除了方便的网络请求方式以外，Retrofit2还支持Rxjava2的响应式编程范式，通过在定义http接口时定义返回值为Rxjava2的Observable对象，就可以方便的完成异步网络请求。另外因为支持嵌入请求拦截器，可以通过统一的请求拦截器完成诸如无网络检测，403鉴权，token校验失败等网络基础功能。

2.3.3 Dagger2

Dagger2是一款基于Java注解来实现的完全在编译阶段完成依赖注入的开源库，主要用于模块间解耦、提高代码的健壮性和可维护性。Dagger2在编译阶段通过apt利用Java注解自动生成Java代码，然后结合手写的代码来自动帮我们完成依赖注入的工作。

我们之所是要依赖注入，最重要的就是为了解耦，达到高内聚低耦合的目的，保证代码的健壮性、灵活性和可维护性。当在对象A中通过new操作符实例化对象B时，我们就说对象A依赖对象B，这样编码会带来两个个问题，面向对象的设计模式告诉我们，应该遵守依赖倒置原则，暨应该面向接口编程而不是面向实例，另外还应该遵守迪米特法则，暨A应该只知道需要调用B的什么行为而不是知道B的全部，基于此Dagger2这个依赖注入框架可以很好的完成这个任务。Dagger2的依赖注入用例如下图。



Dagger2是基于注解的方式来标记需要被注入和引用的对象，然后通过Android的apt（Annotation Processing Tool）编译时注解工具，在代码编译过程中自动生成被注解标记的依赖注入对象，来达到对象间脱偶的目的。Dagger2主要提供了如下几个注解。

1）@Inject：@Inject有两个作用，一是用来标记需要依赖的变量，以此告诉Dagger2为它提供依赖；二是用来标记构造函数，Dagger2通过@Inject注解可以在需要这个类实例的时候来找到这个构造函数并把相关实例构造出来，以此来为被@Inject标记了的变量提供依赖。

2）@Module：@Module用于标注提供依赖的类。即真正的实现业务的类，最终通过APT编译的方式，注入到目标对象中。

3）@Provides：@Provides用于标注Module所标注的类中的方法，该方法在需要提供依赖时被调用，从而把预先提供好的对象当做依赖给标注了@Inject的变量赋值。

4）@Component：@Component用于标注接口，是依赖需求方和依赖提供方之间的桥梁。被Component标注的接口在编译时会生成该接口的实现类（如果@Component标注的接口为CarComponent，则编译期生成的实现类为DaggerCarComponent）,我们通过调用这个实现类的方法完成注入。

5）@Qulifier：@Qulifier用于自定义注解，简单的说它是注解的注解，当被标记的依赖对象出现同比对象或者同名方法时，可以通过这个注解来定义如何区分它们。

6）@Scope：@Scope同样用于自定义注解，我能可以通过@Scope自定义的注解来限定注解作用域，实现局部的单例；

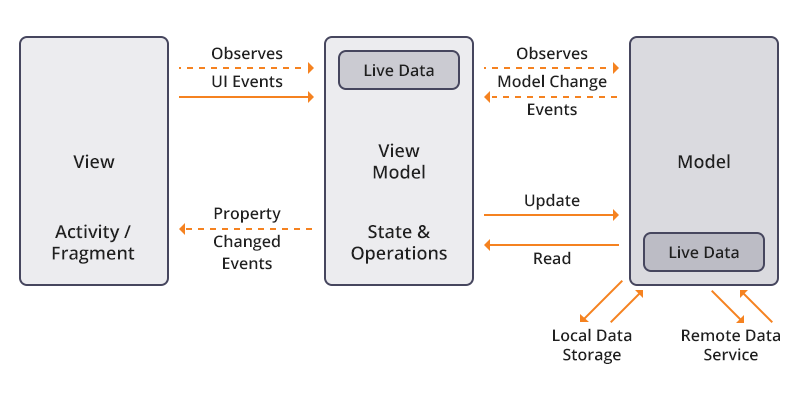
7）@Singleton：@Singleton是一个通过@Scope定义的注解，我们一般通过它来实现全局单例。如果被标注的对象并非单例则可能发生运行时问题。

通过引入Dagger2依赖注入框架可以为我们提供一个简单的对象间脱偶方案，通过注解的方式也可以显著的降低使用和学习的门槛，不过同时因为这些是编译时发生的，所以会增加编译的时间，同时因为会额外生成多个注解对象，也会降低程序的性能。

2.4 DataBinding

DataBinding是谷歌官方发布的一个框架，顾名思义即为数据绑定，是 MVVM 模式在 Android 上的一种实现，用于降低布局和逻辑的耦合性，使代码逻辑更加清晰。MVVM 相对于 MVP，其实就是将 Presenter 层替换成了 ViewModel 层。DataBinding 能够省去我们一直以来的 findViewById() 步骤，大量减少Activity内的代码，数据能够单向或双向绑定到layout文件中，有助于防止内存泄漏，而且能自动进行空检测以避免空指针等异常情况.

在MVVM模式中，能够架起Model数据层和View视图层之间的桥梁，同时又能避免直接引用实例这种高耦合的方式，就是依赖DataBinding技术。DataBinding技术的实现原理如下图，图xxx所示。



图xxx，DataBinding的实现原理

如上图所示，整个数据绑定过程包括，通过实现观察者模式，利用Observes给getter加上@Bindable注解，并在setter中实现域的变动通知。当数据发送变化时变动通知可以用PropertyChangeEvents来传递，在过程中还会设定state脏标记，来达到只有数据真正发送变化时才会触发属性变更事件。

数据绑定实现了数据和识图的分离，结合响应式编程框架RxJava等，编码效率可以得到进一步提高。

2.5 本章小结

本章主要介绍了Android系统的相关知识以及本课题中主要设计模式MVVM，然后结合整个应用开发中最基础的Rrd开发框架，包括异步框架Rxjava2，网络请求库Retrofit2，依赖注入框架Dagger2和Android数据绑定技术进行了重点介绍。

引用

2.3.1

<http://gank.io/post/560e15be2dca930e00da1083>

2.3.2

<https://square.github.io/retrofit/>

<http://www.voidcn.com/article/p-ochwoldp-brh.html>

2.3.3

<https://zhuanlan.zhihu.com/p/24454466>

<https://guides.codepath.com/android/dependency-injection-with-dagger-2>

2.4

<https://developer.android.com/topic/libraries/data-binding>

单词表

RESTful

JSON

依赖倒置

迪米特法则

APT