dagger2 让你爱不释手：终结篇

2016-04-10 安卓应用频道

(点击上方公众号，可快速关注)

> 来源：伯乐在线 - 牛犇
> 链接：http://android.jobbole.com/82705/

前言

如果您对dagger2的概念，整个依赖注入框架还不清楚，可以先了解下我的前2篇文章：

《dagger2让你爱不释手：基础依赖注入框架篇》

《dagger2让你爱不释手：重点概念讲解、融合篇》

这2篇文章也收到好多网友的好评和提问，谢谢大家的支持。我大概总结了下提的问题：

- dagger2到底能带来哪些好处？
- dagger2怎么使用？

因此我将结合这2点来进行本文的讲解。并且会有具体的sample。

dagger2到底有哪些好处？

咱们直奔主题：

增加开发效率、省去重复的简单体力劳动

首先new一个实例的过程是一个重复的简单体力劳动，dagger2完全可以把new一个实例的工作做了，因此我们把主要精力集中在关键业务上、同时也能增加开发效率上。

省去写单例的方法，并且也不需要担心自己写的单例方法是否线程安全，自己写的单例是懒汉模式还是饿汉模式。因为dagger2都可以把这些工作做了。

更好的管理类实例

每个app中的ApplicationComponent管理整个app的全局类实例，所有的全局类实例都统一交给ApplicationComponent管理，并且它们的生命周期与app的生命周期一样。

每个页面对应自己的Component，页面Component管理着自己页面所依赖的所有类实例。

因为Component，Module，整个app的类实例结构变的很清晰。

解耦

假如不用dagger2的话，一个类的new代码是非常可能充斥在app的多个类中的，假如该类的构造函数发生变化，那这些涉及到的类都得进行修改。设计模式中提倡把容易变化的部分封装起来。

我们用了dagger2后。

假如是通过用Inject注解标注的构造函数创建类实例，则即使构造函数变的天花乱坠，我们基本上都不需要修改任何代码。

假如是通过工厂模式Module创建类实例，Module其实就是把new类实例的代码封装起来，这样即使类的构造函数发生变化，只需要修改Module即可。

有个网友问过一个这样的问题，Module的构造函数也会发生变化，发生变化后，相应的new Module的类也发生变化，这就没有达到解耦的效果。首先解耦不是说让类之间或模块之间真的一点关系都没有了，解耦达到的目的是让一个类或一个模块对与自己有关联的类或模块的影响降到最低，不是说这种影响就完全没有了，这是不可能的。

解耦还有个好处，就是方便测试，若需要替换为网络测试类，只需要修改相应的Module即可。

项目中使用dagger2注意点

具体的代码就不讲了，dagger2 sample地址，大家自行下载。这里重点说下dagger2对目标类进行依赖注入的过程，现在假设要初始化目标类中的其中一个依赖类的实例，那具体步骤就在下面：

> 步骤1：查找Module中是否存在创建该类的方法。
>
> 步骤2：若存在创建类方法，查看该方法是否存在参数

> 步骤2.1：若存在参数，则按从**步骤1**开始依次初始化每个参数
>
> 步骤2.2：若不存在参数，则直接初始化该类实例，一次依赖注入到此结束
>
> 步骤3：若不存在创建类方法，则查找Inject注解的构造函数，看构造函数是否存在参数
>
> 步骤3.1：若存在参数，则从**步骤1**开始依次初始化每个参数
>
> 步骤3.2：若不存在参数，则直接初始化该类实例，一次依赖注入到此结束

以上是dagger2进行的一次依赖注入的步骤，其实这个步骤是一个递归的过程，并且在查找类的实例的过程中Module的级别要高于Inject，这概念在上一篇讲过。

下面在说下注意的几点

- 一个app必须要有一个Component（名字可以是ApplicationComponent）用来管理app的整个全局类实例
- 多个页面可以共享一个Component
- 不是说Component就一定要对应一个或多个Module，Component也可以不包含Module
- 自定义Scope注解最好使用上，虽然不使用也是可以让项目运行起来的，但是加上好处多多。

总结

好了关于dagger2的所有的概念知识点到此终于结束了，希望能帮助大家，与大家共勉，有问题可以随时与我沟通。