# 什么是Dagger2?

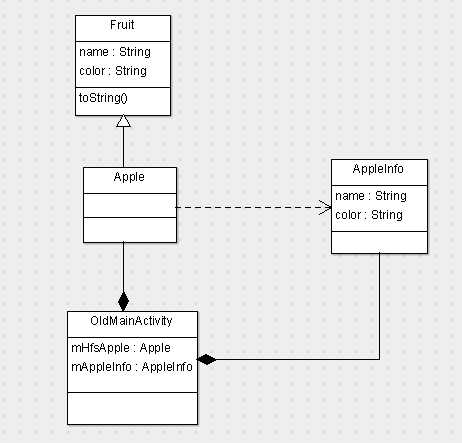
Dagger 2是一种依赖注入的框架，能够在编译时自动生成出一些代码，这些代码可以帮助对应的实例初始化。

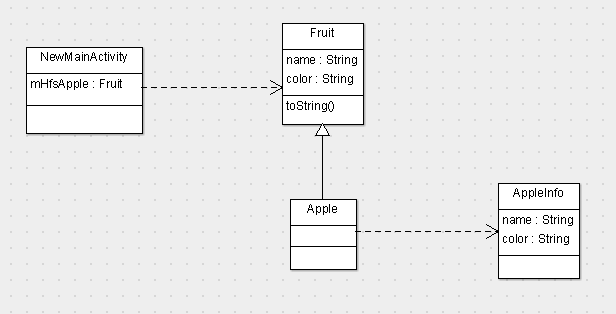
目前 Dagger 有两个分支，一个由 [Square](https://github.com/square/dagger" \t "http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2016/0309/_blank) 维护，一个为 Google 在前者的基础上开出的分支，即 [Dagger 2](https://github.com/google/dagger" \t "http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2016/0309/_blank)

# 为什么要用Dagger2?

例子: 跟Apple类和AppleInfo类产生依赖！

如果业务上变化需对Apple和AppleInfo进行修改，则会影响所有使用了Apple和AppleInfo的类！





# 优势：

1. 构造/使用的分离
2. 单元测试
3. 独立并行开发

# 劣势：

1. 产生较大的模块代码
2. 学习使用框架时间曲线长

# 3.怎么用Dagger2?

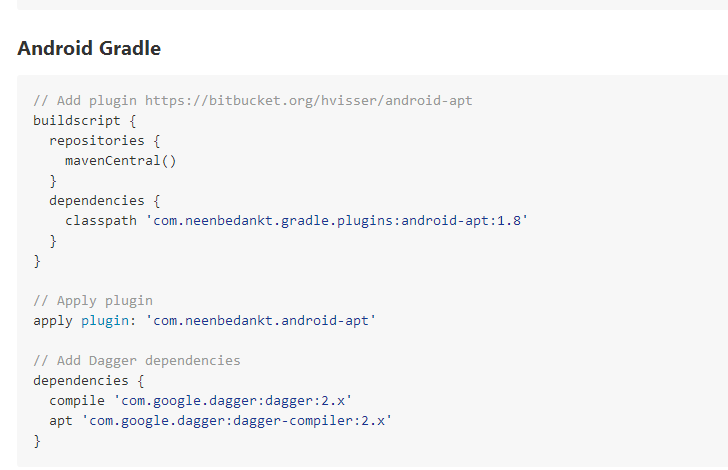
在引入 Dagger 之前，我们需要了解一些基础概念。Dagger 主要分三块：

@Inject：需要注入依赖的地方，Dagger 会构造一个该类的实例并满足它所需要的依赖；

@Module：依赖的提供者，Module 类中的方法专门提供依赖，并用 @Provides 注解标记；

@Component：依赖的注入者，是 @Inject 和 @Module 的桥梁，它从 @Module 中获取依赖并注入给@Inject。

对于以上关系，一句话解释就是：**模块（Module）负责提供依赖，组件（Component）负责注入依赖。**

****

# 其他注解：

**@Scope：**Dagger可以通过自定义注解限定注解作用域。

**@Qualifier：**限定符，当类的类型不足以鉴别一个依赖的时候，我们就可以使用这个注解来区分。

**@Singleton：**单例模式，依赖的对象只会被初始化一次。

**@Subcomponent:** Subcomponent用于拓展原有component。同时，这将产生一种副作用——子component同时具备两种不同生命周期的scope。子Component具备了父Component拥有的Scope，也具备了自己的Scope。

# 参考：

<http://www.cnblogs.com/tiantianbyconan/p/5092083.html>

<http://blog.csdn.net/duo2005duo/article/details/50618171>