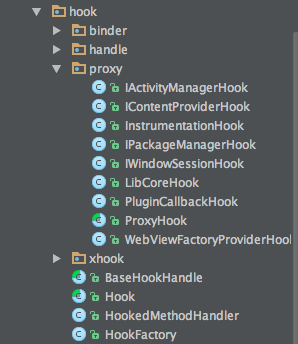
[插件开发之360 DroidPlugin源码分析（二）Hook机制](http://blog.csdn.net/hejjunlin/article/details/52124397)

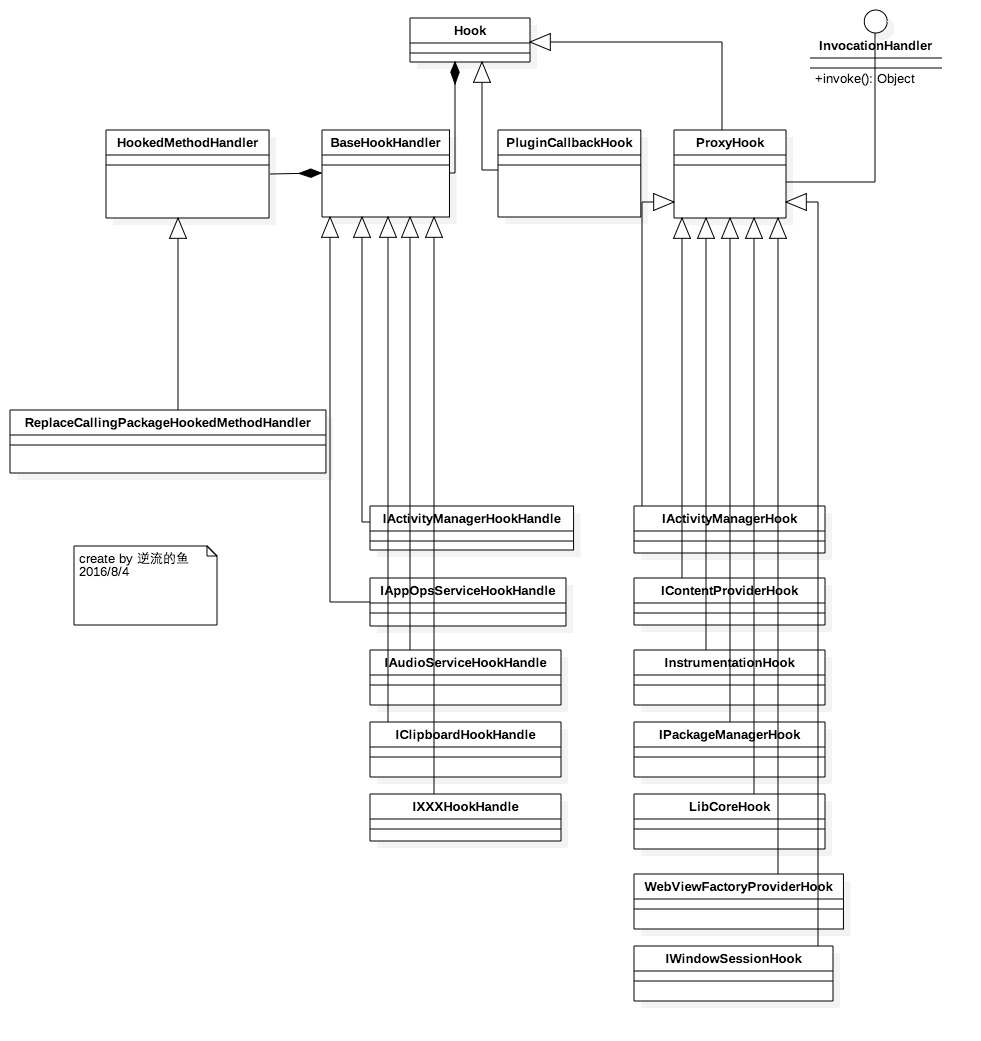
前言：新插件的开发，可以说是为插件开发者带来了福音，虽然还很多坑要填补，对于这款牛逼的插件机制，一直想找个时间分析和总结下它的code，话不多说，直接入正题，本文是分析../hook/handle及../hook/proxy下代码，../hook/binder单独分析

* **Hook机制的包结构关系**
* **Hook机制的类图关系**
* **Hook机制的时序图关系**
* **Manifest权制申请**
* **基类Hook做了什么？**
* **HookedMethodHandler**
* **基类BaseHookHandle和Hook有什么关系？**
* **ProxyHook能干什么？**
* **实例-如何hook IPackageManager**

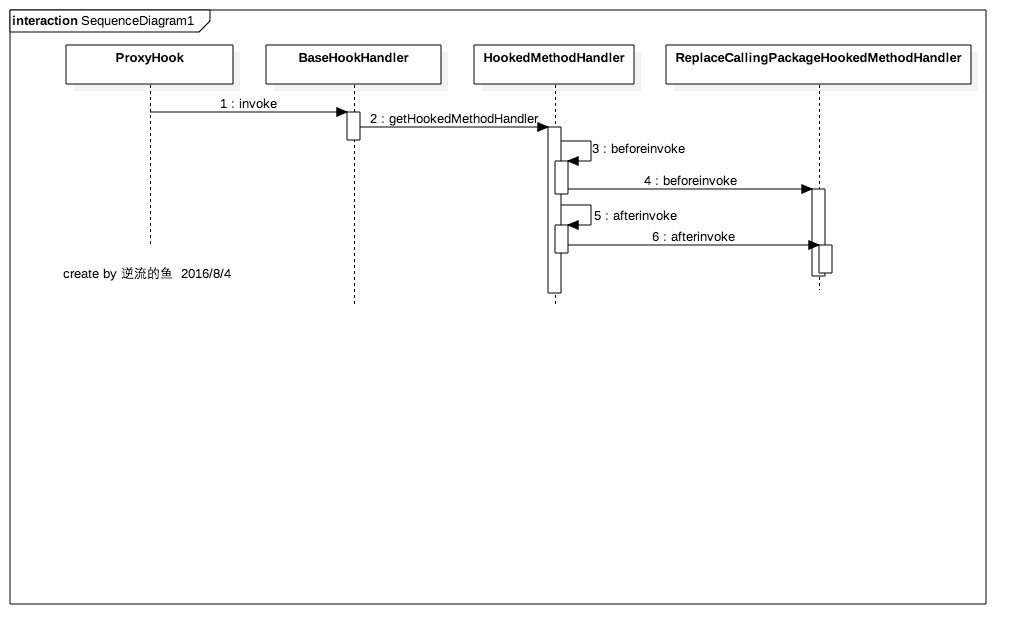
## Hook机制的包结构关系



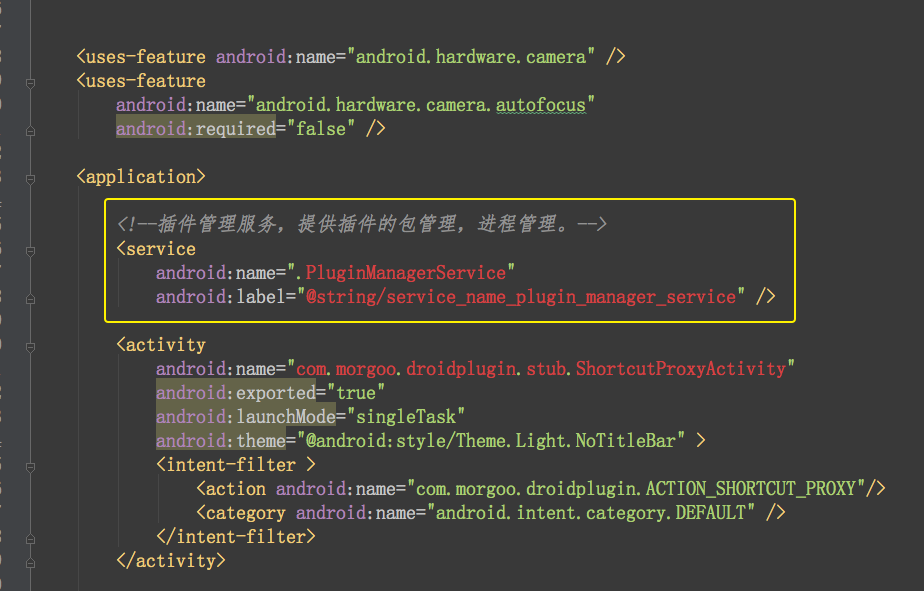
## Hook机制类图关系

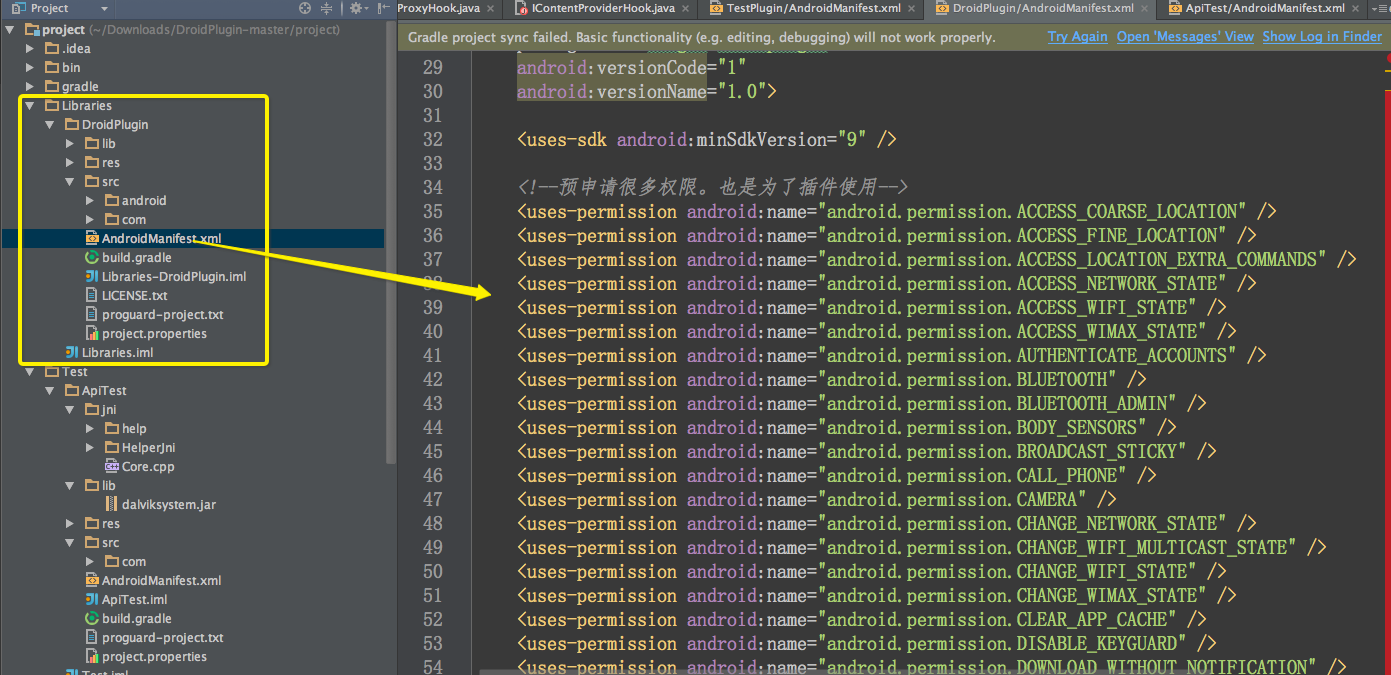
  
首先定义了一个基类Hook，抽象类，外部可以通过setEnable()方法来使能否hook。声明了onInstall和onUnInstall及相关的方法，子类可以覆盖这些方法完成相应的车间机床,这里相当于提供一个车间，机床上的具体操作什么由子类去自己实现。

## Hook机制的时序图关系

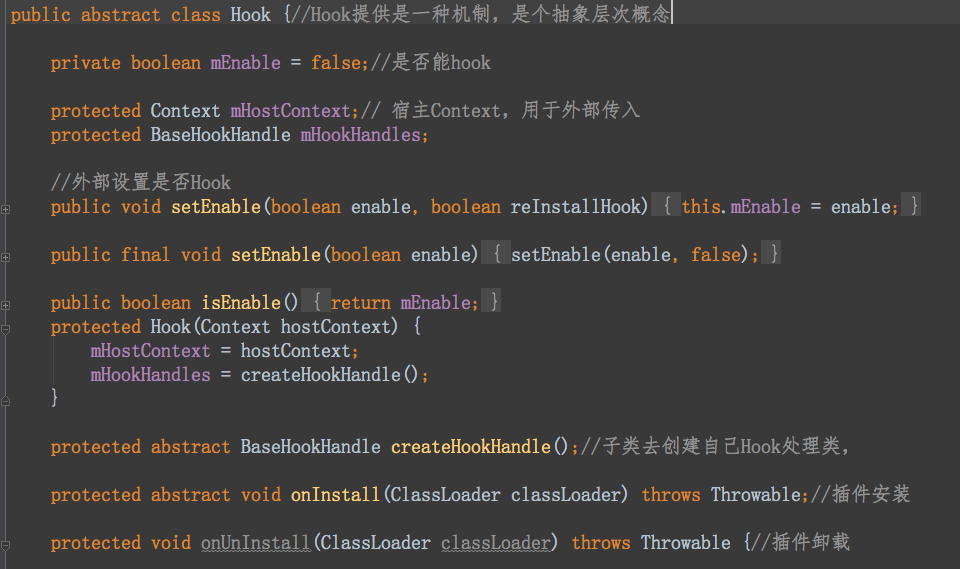


## Manifest权限申请

插件管理服务类声明：   


权限申请：   


## 基类Hook做了什么？

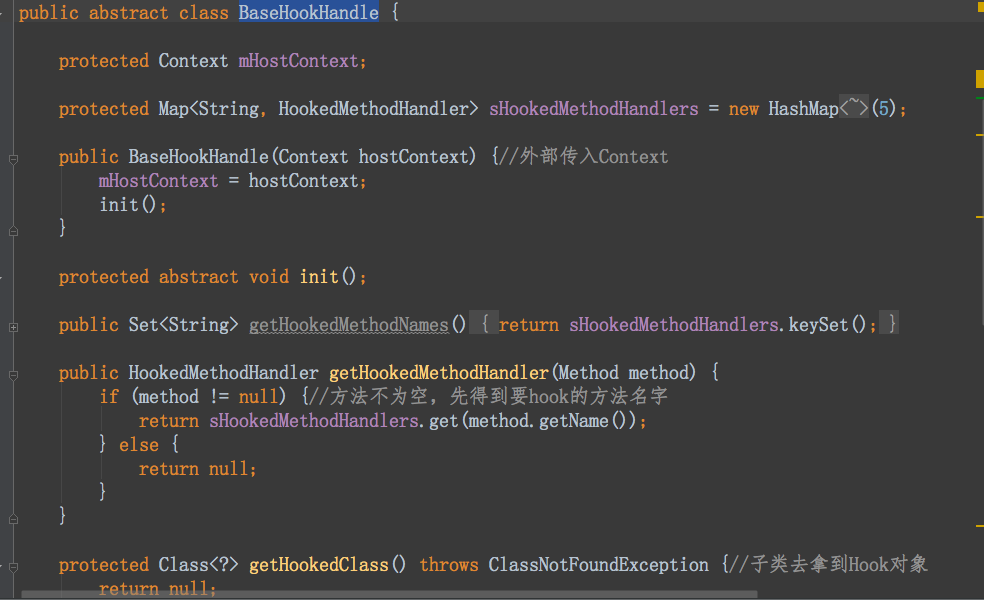


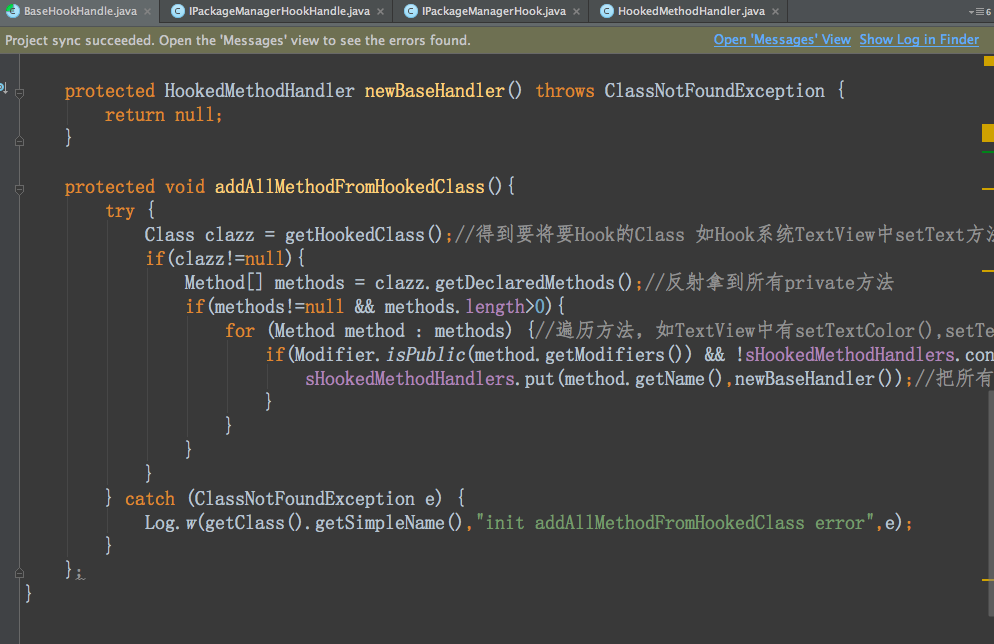
## ProxyHook



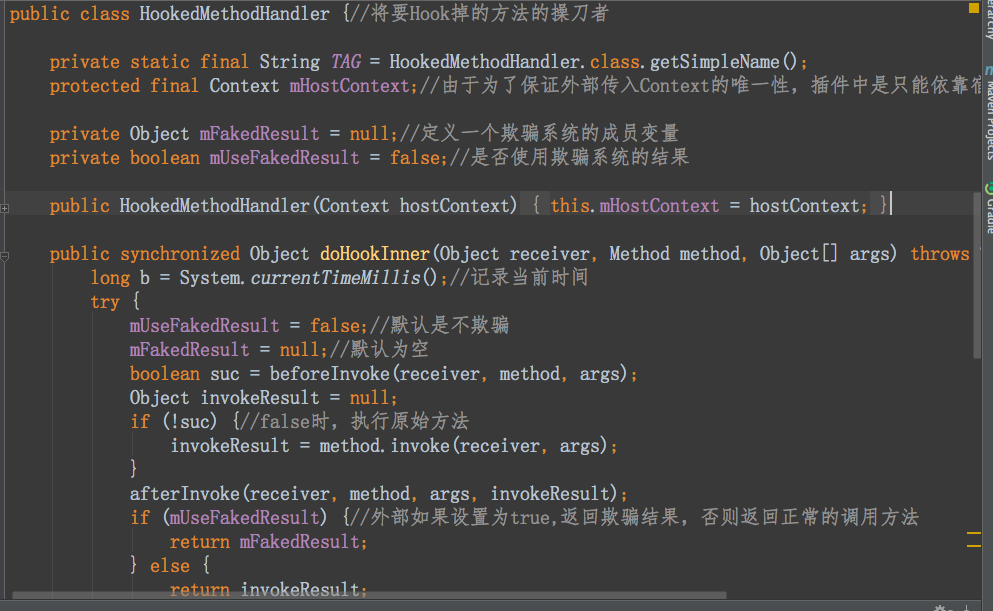
ProxyHook继承自Hook，实现了InvocationHandler接口。它有一个setOldObj()方法，用来保存原始对象。新的代理对象可以看到在代码中是如何实现的（动态代理）

## BaseHookHandle

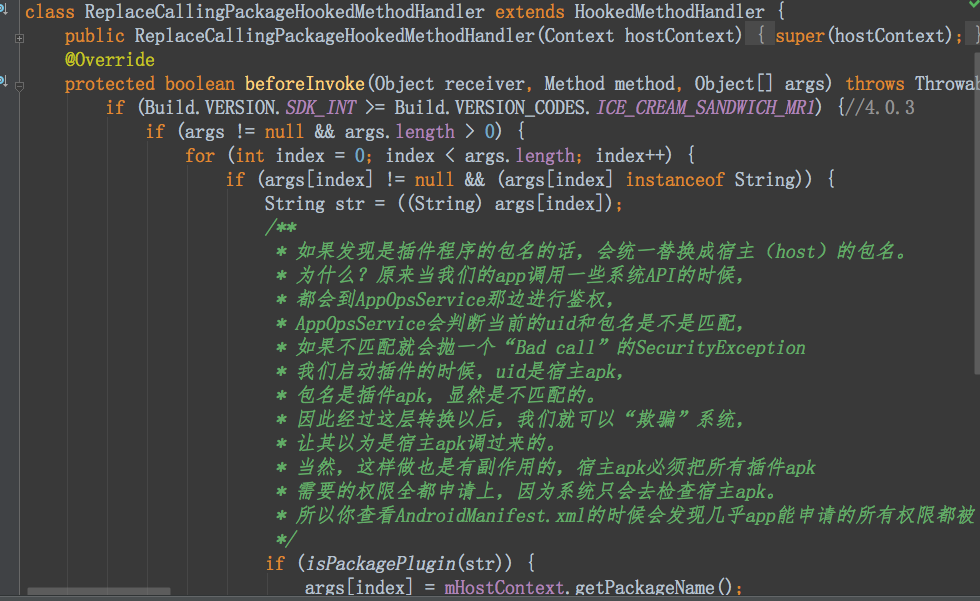


  
接上面ProxyHook中的invoke()方法，mHookHandles是一个BaseHookHandle对象，内部包含了一个Map，可以根据API名映射到对应对应的HookedMethodHandler对象。这个Map由其子类IXXXHookHandle在初始化的时候进行填充。   
紧接着调用HookedMethodHandler的doHookInner()方法：

## HookedMethodHandler



## ReplaceCallingPackageHookedMethodHandler



## IO重定向



## 递归遍历



## 以IPackageManager为例

IPackageManagerHook：Hook所有IPackageManager的方法   
IActivityManagerHookHandle：安装所有被Hook的方法的处理对象，加入到Map中   
IPackageManagerHandle.checkSignatures：这是一个内部类，继承HookedMethodHandler, 专门校验签名的。以此，还有各种各样的PackageManger原生中的方法，在这都变成了一个内部类继承了HookedMethodHandler.上图：   
IPackageManagerHookHandle：

