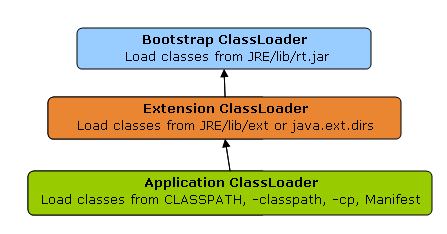
Android插件开发-动态加载

1. android类加载器介绍ClassLoader

双亲委托（ClassLoader.java） 目的是加载路径不一致

作用：相关类只会加载一次。避免核心类被重复加载，防止核心类被替换。

Java中：AppClassLoader->ExtClassLoader->Bootstrp loader



Android：（PathClassLoader/DexClassLoader）->BootClassLoader

PathClassLoader:不可以定义加载的dex文件路径，只能加载dex文件，android系统本身默认使用。即：加载系统已安装的dex文件（已安装路径system/app, data/app），dex路径参见：/data/dalvik-cache



DexClassLoader：可以指定路径，可以加载apk，jar文件（内部包含dex文件）

1. 加载Activity，3种方法

**1）、**静态代理，插件->宿主

宿主程序中声明全4种启动模式的Activity，这些Activity都是代理。插件中的activity都是实现activity生命周期方法的普通java类。当调用启动activity时，实际启动的是宿主中的代理。

主要实现的时候：在宿主中通过声明接口，接口中定义Activity的生命周期。插件中的Activity实现这个接口。将接口抽出来打包成jar，在eclipse中以Library的形式导入，在AndroidStudio中Module的Dependencies选项中Scope属性为Provided。

**2）、**动态创建（动态代理）宿主->插件 开源项目：[android-pluginmgr](https://github.com/houkx/android-pluginmgr/)

代码运行时加载插件中的activity，在运行时动态创建activity，这个activity继承自插件中的指定activity。宿主中需要占坑，即声明子activity。

动态创建使用[DexMaker](https://github.com/crittercism/dexmaker)

3）、修改ClassLoader

反射修改系统类加载器：目的是为了加载的Activity有自己的启动流程生命周期，被AMS管理

修改ClassLoader：（具体代码参见Demo）

方法1：替换ClassLoader

1）、替换LoadApk中的私有ClassLoader成员变量。将PathClassLoader替换成DexClassLoader。

关键代码：



注意这里的getClassLoader：获取当前的activity的PathClassLoader，作为DexClassLoader的父ClassLoader，利用双亲委托机制，保证ClassLoader执行了PathClassLoader加载流程（主要是根ClassLoader执行了加载代码）。

2）、替换ClassLoader：通过反射，替换当前进程的ClassLoader

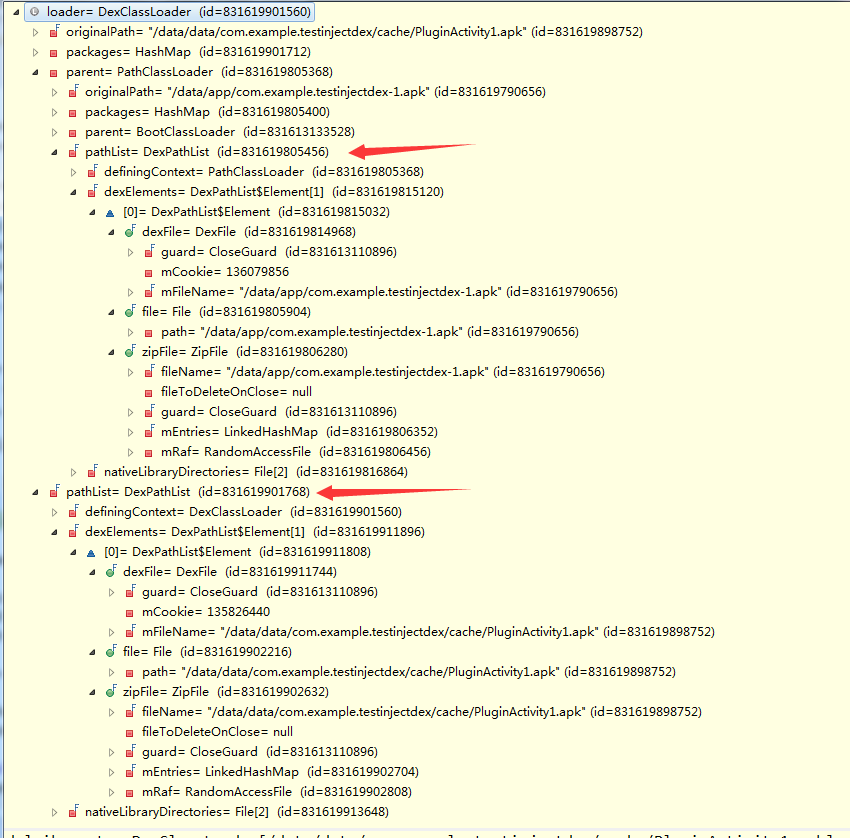


注意：ActivityThread中的mPackages成员，在4.4之前是HashMap，4.4中就改成了ArrayMap了。运行demo的时候需要根据自己实际运行的机器版本号来更改代码。

方法2：添加PathClassLoader中加载路径

如图是Demo运行中创建的DexClassLoader相关信息。

Parent就是系统默认加载器PathClassLoader，注意pathList中的dexElements指向不同。



将DexClassLoader的PathList中的dexElements数组添加到PathClassLoader的PathList中的dexElements中。

开源项目：

Dynamic-load-apk：<https://github.com/singwhatiwanna/dynamic-load-apk>

Android-pluginmgr：<https://github.com/houkx/android-pluginmgr>

360DroidPlugin：<https://github.com/Qihoo360/DroidPlugin>

来源：

<http://segmentfault.com/a/1190000004086213>

<http://blog.csdn.net/jiangwei0910410003/article/details/48104581>