反射机制在创建Activity中相关应用的分析

Last edited by **caoquanli** 1 month ago

反射机制在创建Activity中相关应用的分析

Table of Contents

Android反射机制简述

Android中存在API中部分类或方法被隐藏、由于多态性产生基类变量引用不确定派生类实例等现象，此时就可能会用到Android的反射机制来获取类实例并调用其方法。

Android反射机制获取类对象并调用其方法的基本过程如下，
1. 通过ClassLoader按类的完整类名加载获得目标类并获取类实例；
2. 通过加载得到的类按方法名和参数类型列表获取到该方法对应的Method对象；
3. 通过此Method对象的invoke方法，将之前获取的类实例和方法需要的参数列表作为invoke方法的参数进行调用。

ActivityThread创建Activity的反射机制应用

1. ActivityThread创建Activity源码解读

Android源码的ActivityThread.java中，在启动Activity的performLaunchActivity方法里创建Activity对象，代码如下，

```
private Activity performLaunchActivity(ActivityClientRecord r, Intent customIntent) {
    ......
    ContextImpl appContext = createBaseContextForActivity(r);
    Activity activity = null;
    try {
        java.lang.ClassLoader cl = appContext.getClassLoader();
        activity = mInstrumentation.newActivity(
                cl, component.getClassName(), r.intent);
        StrictMode.incrementExpectedActivityCount(activity.getClass());
        r.intent.setExtrasClassLoader(cl);
        r.intent.prepareToEnterProcess();
        if (r.state != null) {
            r.state.setClassLoader(cl);
        }
    } catch (Exception e) {
        if (!mInstrumentation.onException(activity, e)) {
            throw new RuntimeException(
                "Unable to instantiate activity " + component
                + ": " + e.toString(), e);
        }
    }
    ......
}
```

此处先为Activity创建了上下文Context对象appContext，利用appContext获取用于加载目标Activity的类加载器ClassLoader对象，然后通过mInstrumentation对象的newActivity方法获取了一个Activity对象，传参为刚刚获取的ClassLoader对象、代表目标Activity类名的String和启动Activity的Intent对象。继续查看newActivity方法，方法内容如下，

```
public Activity newActivity(ClassLoader cl, String className,
        Intent intent)
        throws InstantiationException, IllegalAccessException,
        ClassNotFoundException {
    return (Activity)cl.loadClass(className).newInstance();
}
```

从newActivity方法的内容看来，先通过ClassLoader对象的loadClass方法，按传入的目标Activity类名String加载获得目标Activity的类，并用此类获取了目标Activity的实例。
此处加载获得目标Activity的类并获取目标Activity实例，就是使用了Android反射机制。

2. Android反射机制应用意义分析

Activity类是之后所有扩展的Activity的基类，目前Activity类的派生类很多，开发者也可自行继承派生；
由于多态性，目标Activity实例可以是各种各样的Activity派生类实例；
在ActivityThread.java的共通代码中无法确定目标Activity实例是哪个派生类实例，也就无法直接通过具体的派生类构造方法获取实例；
此时使用Android反射机制，由调用者提供完整类名，利用ClassLoader加载目标Activity类并获取其实例，可以达到以下效果，
1. 保持Activity的多态性，不用限制Activity实例所属的派生类；
2. 兼容之后不断扩展的Activity派生类，利用目标Activity完整的类名都可以通过Android反射机制来获取相应派生类实例。