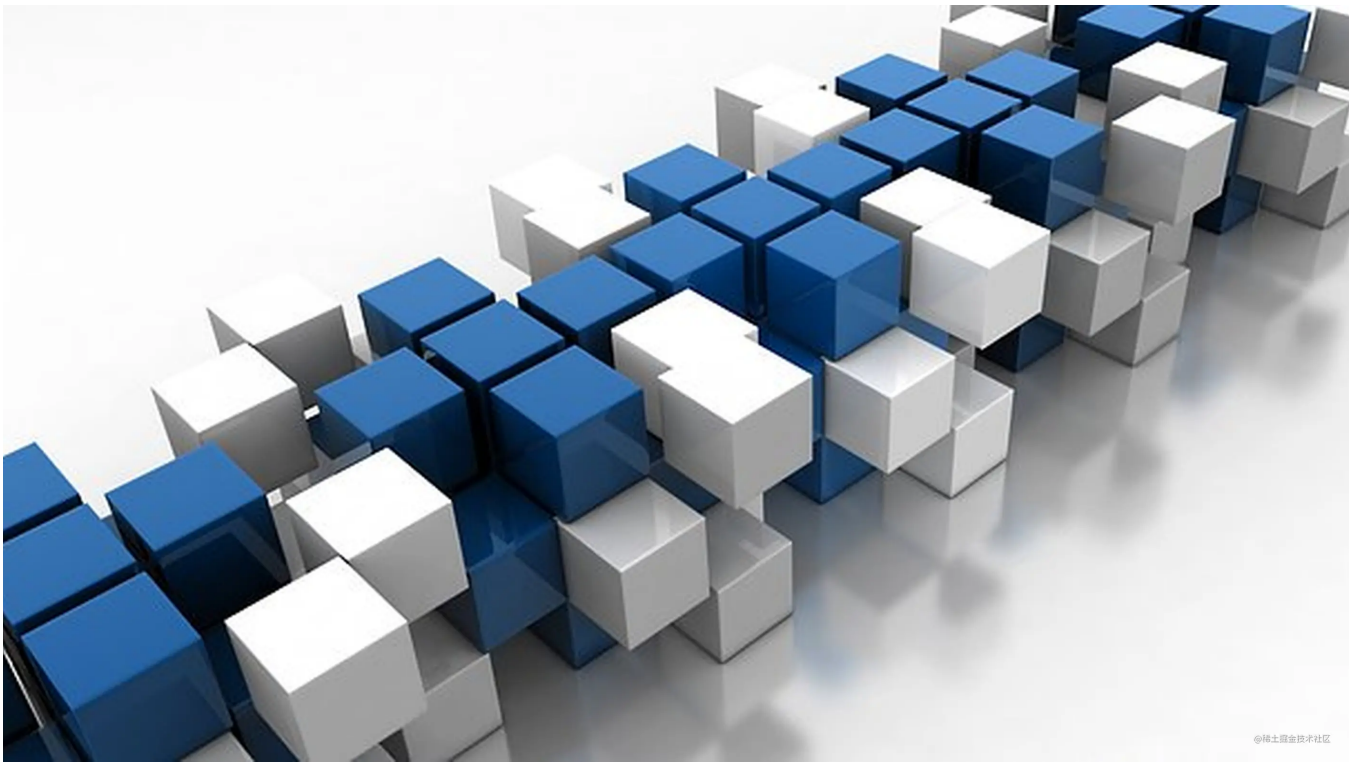
Android 正确的 Log 打印姿势—中级篇

JasonThink LV.4
2017年03月21日 09:53 · 阅读 3420

关注



以前写过『Android正确的Log打印姿势』， 今天我们来谈谈怎么控制 Log 的输出?

第一种：自定义常量， 例如将配置文件放到 asset 下，里面配置是否需要打开 log，Debug 设为 true，Release 设为 false。

第二种：通过 Log.isLogable 去判断，每次需要手动开启。

第三种：Gradle 中 BuildConfig.DEBUG 判断是否是 Debug 模式， 从而做到一些在 Debug 模式下开启， 这样好处是不用再发布前去主动修改。

第一种类似最原始的方式，每次生产环境和开发环境下切换，容易出错，另外在给其他模块提供服务时也不会通用。人类是个不靠谱的东西，机器才是我们目前为止更应该相信的。例如我们经常说的：

> 作为研发不要相信产品、作为测试不要相信研发、作为运维谁都不要相信、作为产品谁都不要相信...

今天我们就不讨论第一种模式，着重说说第二种和第三种情况， 并讨论他们的优缺点。

第二种 Log.isLoagable

直接上代码：

arduino 复制代码

```
public class LogUtils {
    private static final int LEVEL = Log.DEBUG;

    private static boolean isDebug(String tag, int level) {
        return level >= LogUtils.LEVEL && Log.isLoggable(tag, level);
    }

    public static void println(String tag, int level, String msg) {
        if (isDebug(tag, level)) {
            Log.println(level, tag, msg);
        }
    }

    public static void d(String tag, String msg) {
        println(tag, Log.DEBUG, msg);
    }
}
```

上面的代码主要作用，判断当前 level 是否大于 Log.DEBUG 并且判断 Log.isLoggable() 是否为 true，如果条件成立调用 println()。最后问题就回归到怎么通过改变 isLoggable() 状态呢?

最后我们发现可以通过 adb shell 进行设置，方法如下：

> adb shell setprop log.tag.x D

第三种 BuildConfig.DEBUG

我们直到 Debug 模式下 BuildConfig.DEBUG 会始终为 true。我们将上面的代码改造一下， 只需要修改 isDubug() 函数：

```
private static boolean isDebug(int level) {
    return level >= LEVEL && BuildConfig.DEBUG;  //===>修改
}
```

实际开发中我们一般将 LogUtils 放到 Utils Module 中， 而不是放到 app Module 里， 我们会发现 BuildConfig.DEBUG 永远为 false。 Why？ 难道遇到假的系统吗?

WHY

原来 BUildConfig.java 是编译时自动生成的， 每个 Module 都会自己生成一份。所以如果你的应用拥有多个 Module 就会有多个 BuildConfig.java, 而上面的 LogUtils 被放到 Utils, 而 app Module 又依赖 Utils，编译时被依赖的 Utils Module 默认会提供 Release 版给其他 Module 或工程，这就导致该 BuildConfig.DEBUG 会始终为 false。那么如何解决了?

HOW

为了解决上面的问题，我们反编译 Debug 包和 Release 包发现，在AndroidManifest.xml 中 application 节点的 android:debuggable 值是不同的。Debug 版本值为 true，Release 版本值为 false。所以我们可以通过 ApplicationInfo 的这个属性来判断是否是 Debug 版本，代码如下：

typescript 复制代码

```
public class AppUtils {
    private static Boolean isDebug = false;
    public static boolean isDebug() {
        return isDebug == null ? false : isDebug;
    }
    public static void debugMode(Context context) {
        if (isDebug == null) {
            isDebug = context.getApplicationInfo() != null &&
                    (context.getApplicationInfo().flags & ApplicationInfo.FLAG_DEBUGGABLE)
        }
    }
}
```

在自己的 Application 内调用：

> AppUtils.debugMode(getApplicationContext());

修改 LogUtils 的 isDubug() :

arduino 复制代码

```
private static boolean isDebug(int level) {
    return level >= LEVEL && AppUtils.isDebug();
}
```

好了， 这样就解决上面的问题了。

注意：

上面的解决方案，不能再 app Module 中主动设置 android:debuggable，否则无论 Debug 还是 Release 版会始终是设置的值。

优缺点

第二种

优点：

- 不用手动修改， 从而减少出错的概率
- 灵活，能在 Release 也能动态控制输出结果

缺点：

- 当我们重启手机以后上面的设置就不会起作用了，每次重启需要重新设置

第三种

优点：

- 不用手动修改， 从而减少出错的概率

缺点：

- 不够灵活，不能在 Release 也能动态控制输出结果

写在最后

上面的方案可以根据开发中的使用场景来确定，没有最好，只有更好，寻找一个属于自己的就好了，实现项目中持续优化。

PS:

最近在思考一个问题， 到底写什么都你们有用？大家可以留言或写评论告诉我。

更多精彩内容请关注：

分类： Android 标签： Android Android Studio