Log日志的使用方法

1. 支持环境

经过测试确认，目前发现此Log日志无SDK与JDK要求，Android可完美运行，eclipse只要能支持Maven即可。

1. 引用方法：

这里主要以Android Studio为例。

第一步：引入Maven的URL

需要在我们项目的build.gradle中引用：具体代码如下：

repositories {  
 jcenter()  
 maven {  
 url "https://jitpack.io"  
 }  
}

第二步：在我们APP下的bulid.gradle中需要进行远程依赖：具体代码如下：

dependencies {  
  
 compile 'com.github.orhanobut:logger:1.12'  
}

配置以上两步后，重新加载gradle即可引入Log日志。

此Log日志与传统Log日志的优势：

1. 相对于传统Log日志输出更多样化，更加美观。
2. 可打印当前Log日志输出的多层级方法。
3. 打印代码更简洁，如：Logger.i(“value”),而传统的每一个Log都需要对应的一个key,如：Log.i(“key”,”value”).
4. 过滤层级可很好控制。
5. 可区分正式版本与测试版本，可通过Logger.logLevel(LogLevel.NONE) ; 与 Logger.logLevel(LogLevel.FILL) ;来控制，NONE为正式版本，不会输出任何一个日志，FILL为测试版本，及会输出Log日志。
6. 源码可查看，也可自定义输出格式。
7. 显示格式很灵活，可根据项目需求来输出内容。
8. 对json.xml等有特殊、美观的输出格式，可很方便的进行过滤。
9. 有很好的维护性。
10. 对严重级的错误可用标红处理，更加显眼。

使用方法介绍：

初始化：

//日志打印初始化，注意LogLevel.NONE表示为版本发布

Logger

.init(MY\_TAG) // default PRETTYLOGGER or use just init()

.methodCount(0) // default 2

.hideThreadInfo() // default shown

.logLevel(LogLevel.NONE) ; // default LogLevel.FULL

//.methodOffset(0) // default 0

//.logTool(new AndroidLogTool()); // custom log tool, optional

Log日志提供的方法如下：

Log.i(); //打印info信息

Log.d(); // 打印debug信息

Log.e(); // 打印Error 信息

Log.v(); //打印Verbose信息

Log.w(); //打印warn信息

Log.t(“”); //添加标记

Log.t(0); // 打印方法数

Log.wtf(); //输出wtf格式的信息

Log.json(); //输出json格式的信息

Log.xml(); //输出xml个数的信息

注意事项：

1. 使用前必须进行初始化，否则会出现空指针，导致程序崩溃。
2. 发布正式版本时。一定要把LOG版本设为正式版本调用如下代码：Logger.logLevel(LogLevel.NONE);