日志管理模块（Android）接口文档

一、导入方式

1.将 `.aar` 文件放入项目目录

1.创建一个 `libs` 文件夹（如果项目中还没有的话），通常位于 `app/libs`

2.将 `.aar` 文件复制到 `libs` 文件夹中 （需导入release包）

2.在 `build.gradle` 文件中添加依赖

1.打开模块的 `build.gradle` 文件（通常是 `app/build.gradle` ）

2.在 `dependencies` 块中添加以下内容 （ Groovy ）

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.aar'])
}
```

3.如果你使用的是 Kotlin DSL（`build.gradle.kts`），可以这样写

```
dependencies {
    implementation(fileTree("libs") { include("*.aar") })
}
```

3.同步项目

1. 在 Android Studio 中点击 **Sync Project with Gradle Files** 按钮，或者在终端运行 `./gradlew sync` 命令

二、生成aar包

1.在gradle任务模块xlog下执行assemble任务

2.在xlog\build\outputs\aar目录下存在debug和release包

三、日志级别

Android的日志等级通常分为五类，从低到高分别是Verbose、Debug、Info、Warning和Error。

1. **Verbose (V)**: 描述: 冗长啰嗦的日志，通常用于记录开发调试过程中的详细信息。

2. **Debug (D)**: 描述: 调试信息，记录对调试程序和分析问题有帮助的信息。
3. **Info (I)**: 描述: 一般信息，记录一些重要的数据和用户行为。
4. **Warning (W)**: 描述: 警告信息，表示可能出现的问题，但不一定会马上出现错误。
5. **Error (E)**: 描述: 错误信息，表示需要立即关注和解决的问题。

四、日志默认配置

配置内容包括：
- 日志级别，默认为DEBUG
- 日志的保留时间（单位为天，例如：保留三个月日志则设置为90），默认为30
- 日志的保存的路径，仅当日志存储在文件系统中适用，默认为设备主软件所在路径下的log目录

五、初始化

建议初始化的代码写在自定义的Application中。当然写在Activity里面也可以用，反正全局只需要初始化一次就好了！

```
/**
 * Initialize log system, should be called only once.
 *
 * @param context the context of application
 */
fun init(context: Context)
```

```
/**
 * Initialize log system, should be called only once.
 *
 * @param context                the context of application
 * @param configuration        the configuration of log
 */
fun init(context: Context, configuration: LogConfiguration)
```

参数解释：

context:当前应用上下文

configuration:日志配置

示例1：

```
class JMApplication : Application() {

    override fun onCreate() {
        super.onCreate()
        XLog.init(this)
    }
}
```

示例2：

```
class JMApplication : Application() {

    override fun onCreate() {
        super.onCreate()
        XLog.init(
            this, LogConfiguration(
                sLevel = LogLevel.VERBOSE,
                sDay = 30
            )
        )
    }
}
```

六、接口列表

1.打印对象、数组、字符串信息

（1）日志级别为Verbose

```
/**
 * Log an object with level [LogLevel.VERBOSE].
 *
 * @param object the object to log
 */
fun v(`object`: Any?)
```

（2）日志级别为DEBUG

```
/**
 * Log an object with level [LogLevel.DEBUG].
 *
 * @param object the object to log
 */
fun d(`object`: Any?)
```

（3）日志级别为INFO

```
/**
 * Log an object with level [LogLevel.INFO].
 *
 * @param object the object to log
 */
fun i(`object`: Any?)
```

（4）日志级别为WARN

```
/**
 * Log an object with level [LogLevel.WARN].
 *
 * @param object the object to log
 */
fun w(`object`: Any?)
```

（5）日志级别为ERROR

```
/**
 * Log an object with level [LogLevel.ERROR].
 *
 * @param object the object to log
 */
fun e(`object`: Any?)
```

1.1 打印对象信息

示例：

```
val stu = Student("王五",15,"男")
XLog.v(stu)

// 打印信息为:
[2025-06-18 01:56:48.498] [main] VERBOSE [MainActivity.kt] [33 line]
com.test.xlog.Student@6387ee0
```

这里就会发现，直接打印对象和用原生的打印对象.toString()的结果是一样的！实际上看过源码就知道，打印普通对象的时候他都会调用toString()返回的结果打印到控制台。所以这里我们把Student重写它的toString方法再试一次。

```
class Student(
    var name: String,
    var age: Int,
    var sex: String
) {
    override fun toString(): String {
        return "name:$name age:$age sex:$sex"
    }
}

val stu = Student("王五",15,"男")
XLog.v(stu)

// 打印信息为:
[2025-06-18 01:58:07.687] [main] VERBOSE [MainActivity.kt] [31 line] name:王五
age:15 sex:男
```

1.2 打印数组信息

示例:

```
val array = arrayOf(1, 2, 3, 4)
XLog.v(array)

// 打印信息为:
[2025-06-18 01:58:07.684] [main] VERBOSE [MainActivity.kt] [28 line] [1, 2, 3,
4]
```

1.3 打印字符串信息

示例:

```
XLog.v("Test XLog")

// 打印信息为:
[2025-06-18 01:59:56.924] [main] VERBOSE [MainActivity.kt] [22 line] Test XLog
```

2.打印格式化字符串信息

（1）日志级别为VERBOSE

```
/**
 * Log a message with level [LogLevel.VERBOSE].
 *
 * @param format the format of the message to log
 * @param args   the arguments of the message to log
 */
fun v(format: String?, vararg args: Any?)
```

（2）日志级别为DEBUG

```
/**
 * Log a message with level [LogLevel.DEBUG].
 *
 * @param format the format of the message to log
 * @param args   the arguments of the message to log
 */
fun d(format: String?, vararg args: Any?)
```

（3）日志级别为INFO

```
/**
 * Log a message with level [LogLevel.INFO].
 *
 * @param format the format of the message to log
 * @param args   the arguments of the message to log
 */
fun i(format: String?, vararg args: Any?)
```

（4）日志级别为WARN

```
/**
 * Log a message with level [LogLevel.WARN].
 *
 * @param format the format of the message to log
 * @param args   the arguments of the message to log
 */
fun w(format: String?, vararg args: Any?)
```

（5）日志级别为ERROR

```
/**
 * Log a message with level [LogLevel.ERROR].
 *
 * @param format the format of the message to log
 * @param args   the arguments of the message to log
 */
fun e(format: String?, vararg args: Any?)
```

示例：

```
XLog.v("Hello %s, I am %d", "Elvis", 20)

// 打印信息为:
[2025-06-18 05:07:29.213] [main] VERBOSE [XLog.kt] [157 line] Hello Elvis, I am
20
```

3.打印字符串和异常信息

（1）日志级别为VERBOSE

```
/**
 * Log a message and a throwable with level [LogLevel.VERBOSE].
 *
 * @param msg the message to log
 * @param tr  the throwable to be log
 */
fun vt(msg: String?, tr: Throwable?)
```

（2）日志级别为DEBUG

```
/**
 * Log a message and a throwable with level [LogLevel.DEBUG].
 *
 * @param msg the message to log
 * @param tr  the throwable to be log
 */
fun dt(msg: String?, tr: Throwable?)
```

（3）日志级别为INFO

```
/**
 * Log a message and a throwable with level [LogLevel.INFO].
 *
 * @param msg the message to log
 * @param tr  the throwable to be log
 */
fun it(msg: String?, tr: Throwable?)
```

（4）日志级别为WARN

```
/**
 * Log a message and a throwable with level [LogLevel.WARN].
 *
 * @param msg the message to log
 * @param tr  the throwable to be log
 */
fun wt(msg: String?, tr: Throwable?)
```

（5）日志级别为ERROR

```
/**
 * Log a message and a throwable with level [LogLevel.ERROR].
 *
 * @param msg the message to log
 * @param tr  the throwable to be log
 */
fun et(msg: String?, tr: Throwable?)
```

示例：

```
try {
    1/0
}catch (e:Exception){
    XLog.vt("出现错误",e)
}

// 打印信息为:
[2025-06-18 02:02:59.560] [main] VERBOSE [MainActivity.kt] [36 line] 出现错误
java.lang.ArithmeticException: divide by zero
at com.test.xlog.MainActivity.onCreate(MainActivity.kt:34)
at android.app.Activity.performCreate(Activity.java:7802)
at android.app.Activity.performCreate(Activity.java:7791)
at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1299)
at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:3245)
at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:3409)
at
android.app.servertransaction.LaunchActivityItem.execute(LaunchActivityItem.jav
a:83)
at
android.app.servertransaction.TransactionExecutor.executeCallbacks(TransactionE
xecutor.java:135)
at
android.app.servertransaction.TransactionExecutor.execute(TransactionExecutor.j
ava:95)
at android.app.ActivityThread$H.handleMessage(ActivityThread.java:2016)
at android.os.Handler.dispatchMessage(Handler.java:107)
at android.os.Looper.loop(Looper.java:214)
at android.app.ActivityThread.main(ActivityThread.java:7356)
at java.lang.reflect.Method.invoke(Native Method)
at
com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:49
2)
at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:930)
```

4.打印json信息

```
/**
 * Log a JSON string, with level [LogLevel.DEBUG] by default.
 *
 * @param json the JSON string to log
 */
fun json(json: String?)
```

示例:

```
val jsonObject = JSONObject()
        jsonObject.put("name", "张三")
        jsonObject.put("age", "21")
        jsonObject.put("sex", "女")
XLog.json(jsonObject.toString(2))

// 打印信息为:
[2025-06-17 10:11:59.912] [main] DEBUG [Activity.java] [7802 line]
{
    "name": "张三",
    "age": "21",
    "sex": "女"
}
```

5.打印xml信息

```
/**
 * Log a XML string, with level [LogLevel.DEBUG] by default.
 *
 * @param xml the XML string to log
 */
fun xml(xml: String?)
```

示例:

```
val xmlString = "<team>" +
                "<member name='Elvis'/>" +
                "<member name='Leon'/>" +
                "</team>"
XLog.xml(xmlString)

// 打印信息为:
[2025-06-17 10:11:59.949] [main] DEBUG [Activity.java] [7802 line]
<?xml version="1.0" encoding="UTF-8"?>
<team>
    <member name="Elvis"/>
    <member name="Leon"/>
</team>
```

5.日志导出

```
/**
 * Export log from file system or database
 *
 * @param path The export target path
 */
fun exportLog(path: String): Boolean
```

示例：

```
XLog.exportLog("/storage/emulated/0/Android/data/com.test.xlog")

// 导出文件名称为：XLog_2025_07_18_08_34_25.zip
```