链滴

史上最实用的 Android 切片应用库 XAOP 使用指南

作者：xuexiangjys

原文链接：https://ld246.com/article/1603644290565

来源网站：链滴

项目简介

一个轻量级的AOP(Android)应用框架，囊括了最实用的AOP应用。项目地址: https://github.com/xexiangjys/XAOP, 喜欢的话,欢迎star支持!

设计原由

在我们平时开发的过程中，一定会遇到权限申请、线程切换、数据缓存、异常捕获、埋点和方法执行间统计等问题。这些都是非常常见的问题，实现起来也不是很难，不过就是太麻烦了，还会让程序多很多重复性、模版化的代码。

设计思路

让我最初接触到AOP思想的是JakeWharton的hugo,通过阅读它的源码之后，让我对aspectj这项技的动态代码编织深深地着了迷。之后我详细研究了aspectj相关的技术，并不断搜集AOP在Android的典型应用场景，然后通过aspectj这项技术去逐一实现。最后就成就了XAOP这个库。

解决痛点

- 解决快速点击的问题
- 解决Android6.0以上动态权限申请的问题
- 线程自由切换的问题
- 日志埋点问题
- 缓存问题（磁盘缓存和内存缓存）
- 异常捕获处理
- 业务拦截（登陆验证、有效性验证等）

集成指南

添加Gradle依赖

1.先在项目根目录的 build.gradle 的 repositories 添加:

```
allprojects {
    repositories {
        ...
        maven { url "https://jitpack.io" }
    }
}
```

2.再在项目根目录的 build.gradle 的 dependencies 添加xaop插件:

```
buildscript {
    ...
    dependencies {
```

```
        ...
        classpath 'com.github.xuexiangjys.XAOP:xaop-plugin:1.1.0'
    }
}
```

3.在项目的 build.gradle 中增加依赖并引用xaop插件

```
apply plugin: 'com.xuexiang.xaop' //引用xaop插件

dependencies {
    ...
    //如果是androidx项目，使用1.1.0版本及以上
    implementation 'com.github.xuexiangjys.XAOP:xaop-runtime:1.1.0'
    //如果是support项目，请使用1.0.5版本
    implementation 'com.github.xuexiangjys.XAOP:xaop-runtime:1.0.5'
}
```

4.在 Application中进行初始化

```
XAOP.init(this); //初始化插件
XAOP.debug(true); //日志打印切片开启
XAOP.setPriority(Log.INFO); //设置日志打印的等级,默认为0

//设置动态申请权限切片 申请权限被拒绝的事件响应监听
XAOP.setOnPermissionDeniedListener(new PermissionUtils.OnPermissionDeniedListener() {
    @Override
    public void onDenied(List<String> permissionsDenied) {
      // 权限申请被拒绝的处理
    }

});

//设置自定义拦截切片的处理拦截器
XAOP.setInterceptor(new Interceptor() {
    @Override
    public boolean intercept(int type, JoinPoint joinPoint) throws Throwable {
        XLogger.d("正在进行拦截，拦截类型:" + type);
        switch(type) {
            case 1:
                //做你想要的拦截
                break;
            case 2:
                return true; //return true，直接拦截切片的执行
            default:
                break;
        }
        return false;
    }
});

//设置自动捕获异常的处理者
XAOP.setIThrowableHandler(new IThrowableHandler() {
```

```
    @Override
    public Object handleThrowable(String flag, Throwable throwable) {
        XLogger.d("捕获到异常，异常的flag:" + flag);
        if (flag.equals(TRY_CATCH_KEY)) {
            return 100;
        }
        return null;
    }
});
```

兼容Kotlin语法配置

1.在项目根目录的 build.gradle 的 dependencies 添加 aspectjx 插件：

```
buildscript {
    ...
    dependencies {
        ...
        classpath 'com.hujiang.aspectjx:gradle-android-plugin-aspectjx:2.0.10'
    }
}
```

2.在项目的 build.gradle 中增加依赖并引用 aspectjx 插件

```
apply plugin: 'android-aspectjx' //引用aspectjx插件

aspectjx {
    include '项目的applicationId'
}
```

详细使用可参见 kotlin-test 项目进行使用.

混淆配置

```
-keep @com.xuexiang.xaop.annotation.* class * {*;}
-keep class * {
    @com.xuexiang.xaop.annotation.* <fields>;
}
-keepclassmembers class * {
    @com.xuexiang.xaop.annotation.* <methods>;
}
```

基础使用

快速点击切片

- SingleClick属性表

属性名	类型	默认值	备注
value	long	1000	快速点击的间隔（ms）

1.使用 @SingleClick标注点击的方法。注意点击的方法中一定要有点击控件View作为方法参数，否将不起作用。

2.可以设置快速点击的时间间隔，单位:ms。不设置的话默认是1000ms。

```
@SingleClick(5000)
public void handleOnClick(View v) {
    XLogger.e("点击响应！");
    ToastUtil.get().toast("点击响应！");
    hello("xuexiangjys", "666666");
}
```

动态申请权限切片

- Permission属性表

属性名	类型	默认值	备注
value	String[]	/	需要申权限的集合

1.使用 @Permission标注需要申请权限执行的方法。可设置申请一个或多个权限。

2.使用 @Permission标注的方法，在执行时会自动判断是否需要申请权限。

```
@SingleClick
@Permission({PermissionConsts.CALENDAR, PermissionConsts.CAMERA, PermissionConsts.LCATION})
private void handleRequestPermission(View v) {

}
```

主线程切片

1.使用 @MainThread标注需要在主线程中执行的方法。

2.使用 @MainThread标注的方法，在执行时会自动切换至主线程。

```
@MainThread
private void doInMainThread(View v) {
    mTvHello.setText("工作在主线程");
}
```

IO线程切片

- IOThread属性表

属性名	类型	默认值	备注
value	ThreadType	ThreadType.Fixed	线程的类型

1.使用 @IOThread标注需要在io线程中执行的方法。可设置线程池的类型 ThreadType，不设置的默认是Fixed类型。

线程池的类型如下:

- Single:单线程池
- Fixed:多线程池
- Disk:磁盘读写线程池(本质上是单线程池)
- Network:网络请求线程池(本质上是多线程池)

2.使用 @IOThread标注的方法，在执行时会自动切换至指定类型的io线程。

```
@IOThread(ThreadType.Single)
private String doInIOThread(View v) {
    return "io线程名:" + Thread.currentThread().getName();
}
```

日志打印切片

- DebugLog属性表

属性名	类型	默认值	备注
priority	int	0	日志的优级

1.使用 @DebugLog标注需要打印的方法和类。可设置打印的优先级，不设置的话默认优先级为0。意：如果打印的优先级比 XAOP.setPriority设置的优先级小的话，将不会进行打印。

2.使用 @DebugLog标注的类和方法在执行的过程中，方法名、参数、执行的时间以及结果都将会被印。

3.可调用 XAOP.setISerializer设置打印时序列化参数对象的序列化器。

4.可调用 XAOP.setLogger设置打印的实现接口。默认提供的是突破4000限制的logcat日志打印。

```
@DebugLog(priority = Log.ERROR)
private String hello(String name, String cardId) {
    return "hello, " + name + "! Your CardId is " + cardId + ".";
}
```

内存缓存切片

- MemoryCache属性表

属性名	类型	默认值	备注

value	String	""	内存缓存key
enableEmpty	boolean	true	于String、数组和集合等，是否允许缓存为空

1.使用 @MemoryCache标注需要内存缓存的方法。可设置缓存的key，不设置的话默认key为 方法名参数1名=参数1值|参数2名=参数2值|...),当然你也可以修改key的自动生成规则，你只需要调用 XAOP.etICacheKeyCreator即可。

2.标注的方法一定要有返回值，否则内存缓存切片将不起作用。

3.使用 @MemoryCache标注的方法，可自动实现缓存策略。默认使用的内存缓存是 LruCache。

4.可调用 XAOP.initMemoryCache设置内存缓存的最大数量。默认是 Runtime.getRuntime().maxMemory() / 1024) / 8

```
@MemoryCache
private String hello(String name, String cardId) {
    return "hello, " + name + "! Your CardId is " + cardId + ".";
}
```

磁盘缓存切片

- DiskCache属性表

属性名	类型	默认值	备注
value	String	""	内存缓的key
cacheTime	long	-1	缓时间【单位：s】，默认是永久有效
enableEmpty	boolean	true	于String、数组和集合等，是否允许缓存为空

1.使用 @DiskCache标注需要磁盘缓存的方法。可设置缓存的key，不设置的话默认key为 方法名(参1名=参数1值|参数2名=参数2值|...),当然你也可以修改key的自动生成规则，你只需要调用 XAOP.setIacheKeyCreator即可。

2.可设置磁盘缓存的有效期，单位:s。不设置的话默认永久有效。

3.标注的方法一定要有返回值，否则磁盘缓存切片将不起作用。

4.使用 @DiskCache标注的方法，可自动实现缓存策略。默认使用的磁盘缓存是JakeWharton的 DiskruCache。

5.可调用 XAOP.initDiskCache设置磁盘缓存的属性,包括磁盘序列化器 IDiskConverter，磁盘缓存的目录，磁盘缓存的最大空间等。

```
@DiskCache
private String hello(String name, String cardId) {
    return "hello, " + name + "! Your CardId is " + cardId + ".";
}
```

自动捕获异常切片

- Safe属性表

属性名	类型	默认值	备注
value	String	""	捕获异的标志

1.使用 @Safe标注需要进行异常捕获的方法。可设置一个异常捕获的标志Flag，默认的Flag为当前名.方法名。

2.调用 XAOP.setIThrowableHandler设置捕获异常的自定义处理者，可实现对异常的弥补处理。如不设置的话，将只打印异常的堆栈信息。

3.使用 @Safe标注的方法,可自动进行异常捕获，并统一进行异常处理，保证方法平稳执行。

```
@Safe(TRY_CATCH_KEY)
private int getNumber() {
    return 100 / 0;
}
```

自定义拦截切片

- Intercept属性表

属性名	类型	默认值	备注
value	int[]	/	拦截类型

1.使用 @Intercept标注需要进行拦截的方法和类。可设置申请一个或多个拦截类型。

2.如果不调用 XAOP.setInterceptor设置切片拦截的拦截器的话，自定义拦截切片将不起作用。

3.使用 @Intercept标注的类和方法，在执行时将自动调用 XAOP设置的拦截器进行拦截处理。如果截器处理返回true的话，该类或方法的执行将被拦截，不执行。

4.使用 @Intercept可以灵活地进行切片拦截。比如用户登录权限等。

```
@SingleClick(5000)
@DebugLog(priority = Log.ERROR)
@Intercept(3)
public void handleOnClick(View v) {
    XLogger.e("点击响应！");
    ToastUtil.get().toast("点击响应！");
    hello("xuexiangjys", "666666");
}

@DebugLog(priority = Log.ERROR)
@Intercept({1,2,3})
private String hello(String name, String cardId) {
```

```
    return "hello, " + name + "! Your CardId is " + cardId + ".";
}
```

【注意】：当有多个切片注解修饰时，一般是从上至下依次顺序执行。

进阶使用

登陆验证

在应用中，对于部分功能，如：个人中心、钱包、收藏等需要我们验证登录的功能，我们都可以通过Intercept业务拦截切片来实现。

1. 定义业务拦截类型

```
// 登录校验拦截类型
public static final int INTERCEPT_LOGIN = 10;
```

2. 定义拦截处理逻辑

```
XAOP.setInterceptor(new Interceptor() {
    @Override
    public boolean intercept(int type, JoinPoint joinPoint) throws Throwable {
        switch(type) {
            case INTERCEPT_LOGIN:
                if (!LoginActivity.sIsLogined) { //没登录,进行拦截
                    ToastUtils.toast("请先进行登陆！");
                    ActivityUtils.startActivity(LoginActivity.class);
                    return true; //return true，直接拦截切片的执行
                }
                break;
            default:
                break;
        }
        return false;
    }
});
```

3. 在需要拦截的地方增加 @Intercept标注

```
@Intercept(INTERCEPT_LOGIN)
public void doSomeThing() {
    ToastUtils.toast("已登陆过啦～～");
}
```

常见问题

接入的问题

使用前，请一定要仔细阅读集成指南，只要你每一步都参照文档上写的来接入，是不会有任何问题的！

1.问：我的项目是kotlin项目，我该怎么使用？

答：kotlin项目的配置，只需要在原先项目的基础上加上aspectjx 插件即可，详情请参考兼容Kotlin法配置 。

2.问：为什么我每次运行编译时，一直报错 Invalid byte tag in constant pool 而且会自动生成一个 ajcore.xxxxxxxxx.txt文件？

答：这里很有可能你的项目目前还是使用的androidx版本，但是你使用的XAOP版本是support版本导致编译失败。这里需要强调的是，如果你的项目是support版本，请使用1.0.5版本；如果你的项目androidx版本，请使用1.1.0及以上版本。

3.问：为什么我编译都通过了，但是使用任何一个切片都没有起任何作用？

答：这里可能的原因有两个。

- 1.你使用的XAOP版本和你的项目版本不匹配导致。比如你的项目是androidx版本，但是你却使用XOP的support版本，这样瞎配的话，切片是不会起任何作用的。
- 2.你忘记在项目的 build.gradle 中增加xaop插件的引用了。

```
apply plugin: 'com.xuexiang.xaop' //引用xaop插件
```

使用的问题

1.问：为什么我使用 @SingleClick标注点击的方法不起作用？

答：被 @SingleClick标注的方法中，一定要有点击控件View作为方法参数，否则将不起作用。

2.问：为什么我使用 @Permission标注的方法，返回值失效了？

答：由于动态申请权限是一个异步的操作，所以被 @Permission标注的方法是不能有返回值的。

微信公众号

更多资讯内容，欢迎微信搜索公众号:「我的Android开源之旅」

原文链接：史上最实用的 Android 切片应用库 XAOP 使用指南