AOP在移动开发中的应用

WeGene移动开发团队—邓积艺

1.如果我想在所有界面（Activity，ViewController）销毁的时候输出log怎么实现?

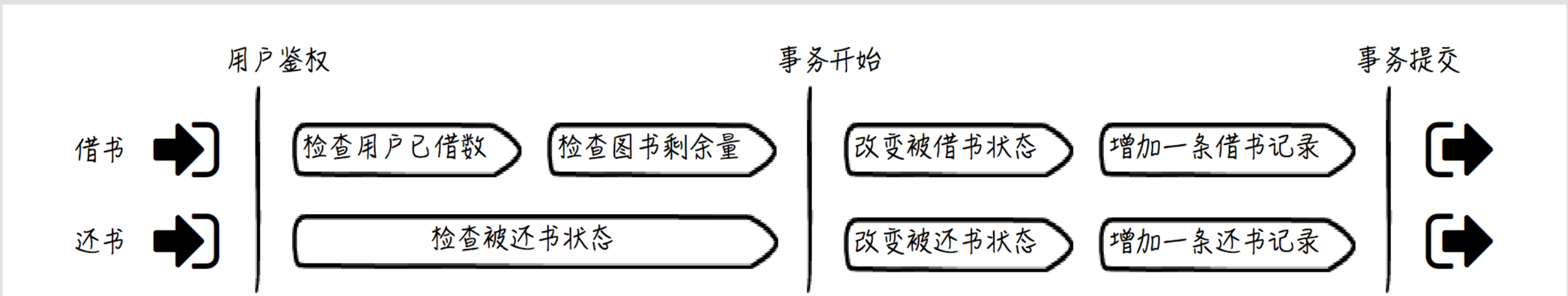
2.GrowingIO的无埋点统计是怎么实现的?

目录

AOP简介

- AOP，Aspect Oriented Programming，面向切面编程
- 面向切面编程是一种通过横切关注点（Cross-cutting Concerns）分离来增强代码模块性的方法，它能够在不修改业务主体代码的情况下，对它添加额外的行为。
- 是对OOP的一种补充,是一种解耦的重要手段

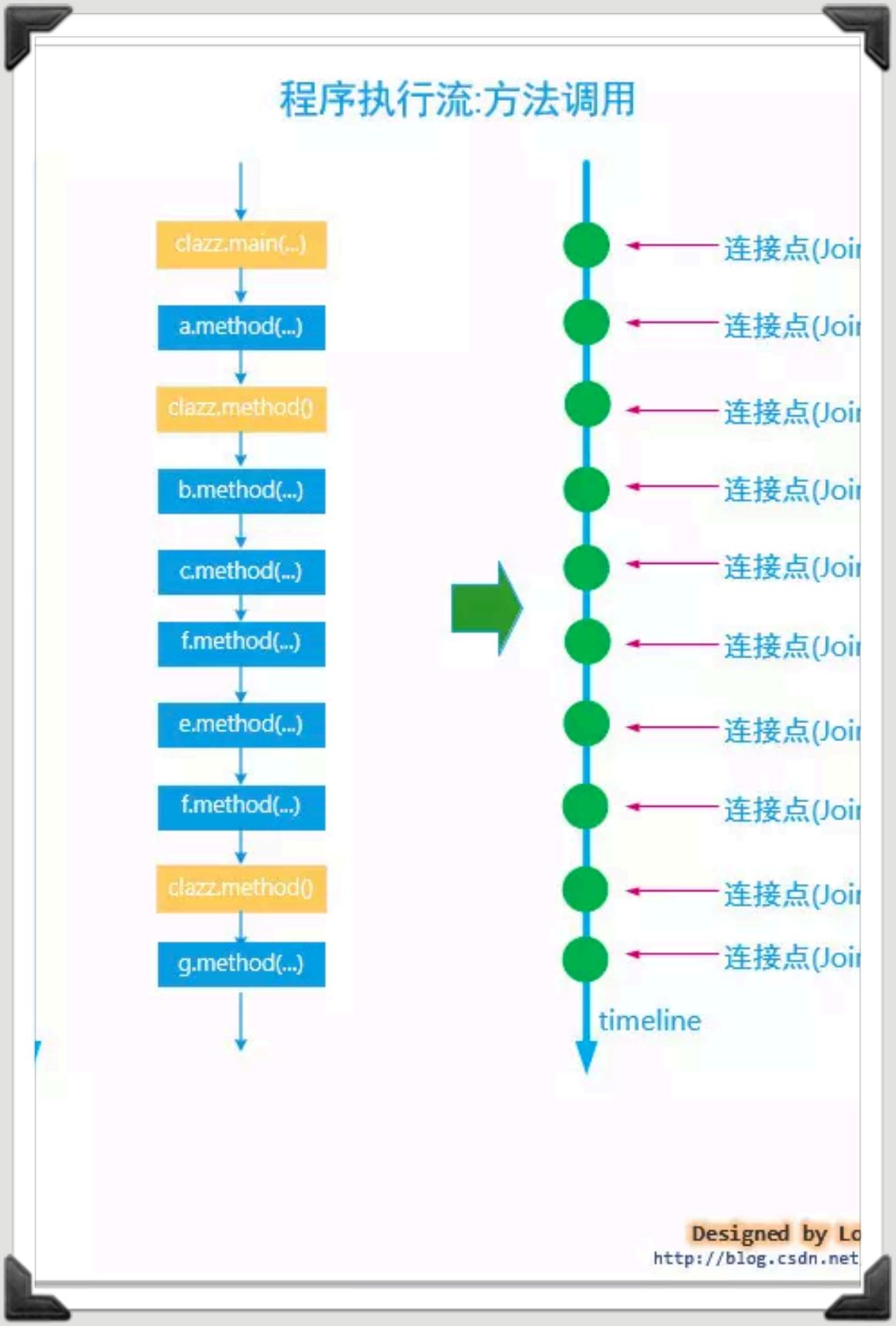
AOP典型案例

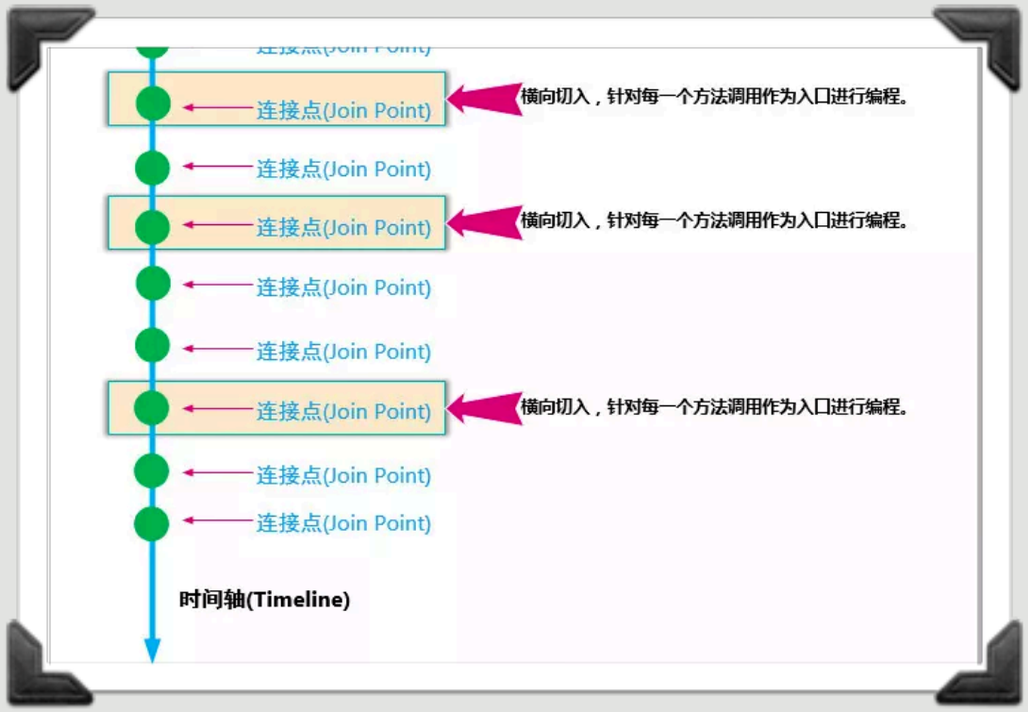


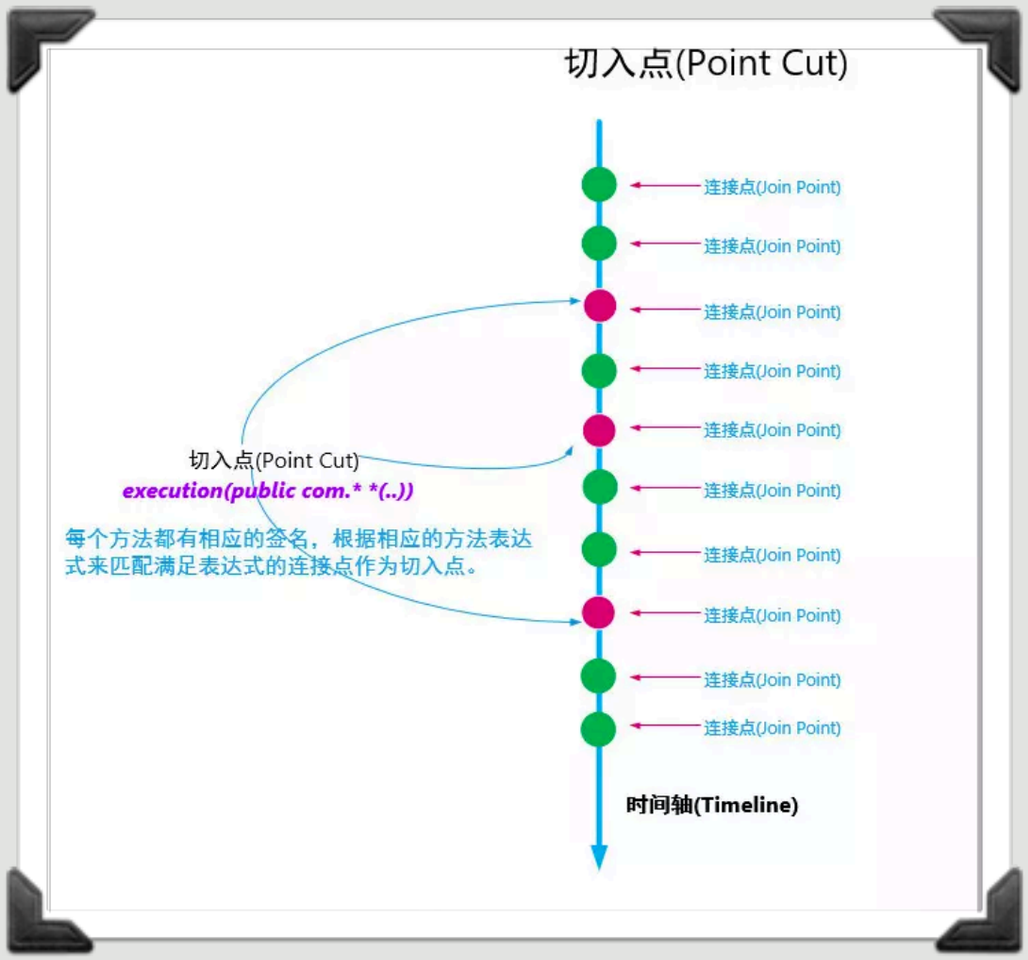
对于图中横向的业务流程，我们能够保持它们独立不变，而把鉴权、事务这样的公共功能，彻底拿出去，放到单独的地方，这样整个业务流程就变得纯粹和干净，没有任何代码残留的痕迹，就好像武林高手彻底隐形了一般，但是，功能却没有任何丢失。就好比面条一般顺下来的业务流程，水平地切了几刀，每一刀，都是一个 AOP 的功能实现。

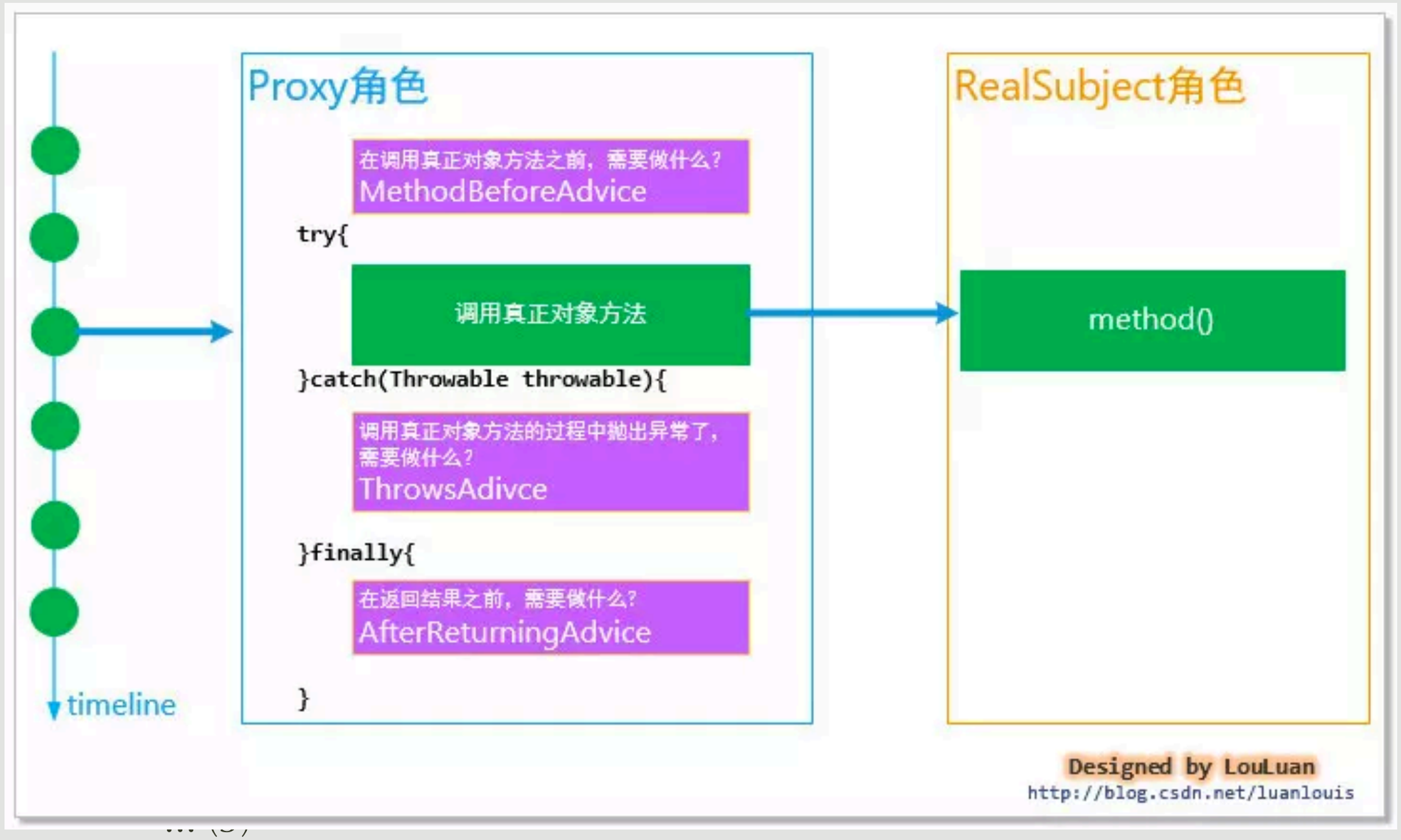
AOP常用概念

- Join point：程序执行期间的一个点,表示方法的执行
- Pointcut: 切入点实际上也是从所有的连接点(Join point)挑选自己感兴趣的连接点的过程
- Aspect: 程序横向切割成若干的面，即Aspect.每个面被称为切面
- Advice: 某个特定连接点的某个方面采取的行动。不同类型的建议包括“周围(Around)”，“之前(Before)”和“之后(After)”建议


程序执行流:方法调用
clazz.main(...)
a.method(...)
clazz.method()
b.method(...)
c.method(...)
f.method(...)
e.method(...)
f.method(...)
clazz.method()
g.method(...)
连接点(Join
timeline
Designed by Lo
http://blog.csdn.net


连接点(Join Point)
横向切入，针对每一个方法调用作为入口进行编程。
时间轴(Timeline)


切入点(Point Cut)
连接点(Join Point)
切入点(Point Cut)
execution(public com.* *(..))
每个方法都有相应的签名，根据相应的方法表达式来匹配满足表达式的连接点作为切入点。
时间轴(Timeline)


Proxy角色
在调用真正对象方法之前，需要做什么?
MethodBeforeAdvice
try{
调用真正对象方法
}catch(Throwable throwable){
调用真正对象方法的过程中抛出异常了，
需要做什么?
ThrowsAdivce
}finally{
在返回结果之前，需要做什么?
AfterReturningAdvice
}
RealSubject角色
method()
timeline
Designed by LouLuan
http://blog.csdn.net/luanlouis

...(3)

- }
- }

AOP实现原理

- 编译期间的静态织入，又称为编译时增强
- 运行期间的动态代理，又称为运行时增强

运行时AOP

- 程序运行时，依靠预先创建或运行时创建的代理类来完成切面功能。这种方式依赖编程语言的动态能力

- JDK 基于接口的动态代理技术（Spring AOP），基于Java反射特性实现

- Objective-C的Method Swizzling和KVO，基于OC的Runtime实现

JDK 基于接口的动态代理技术

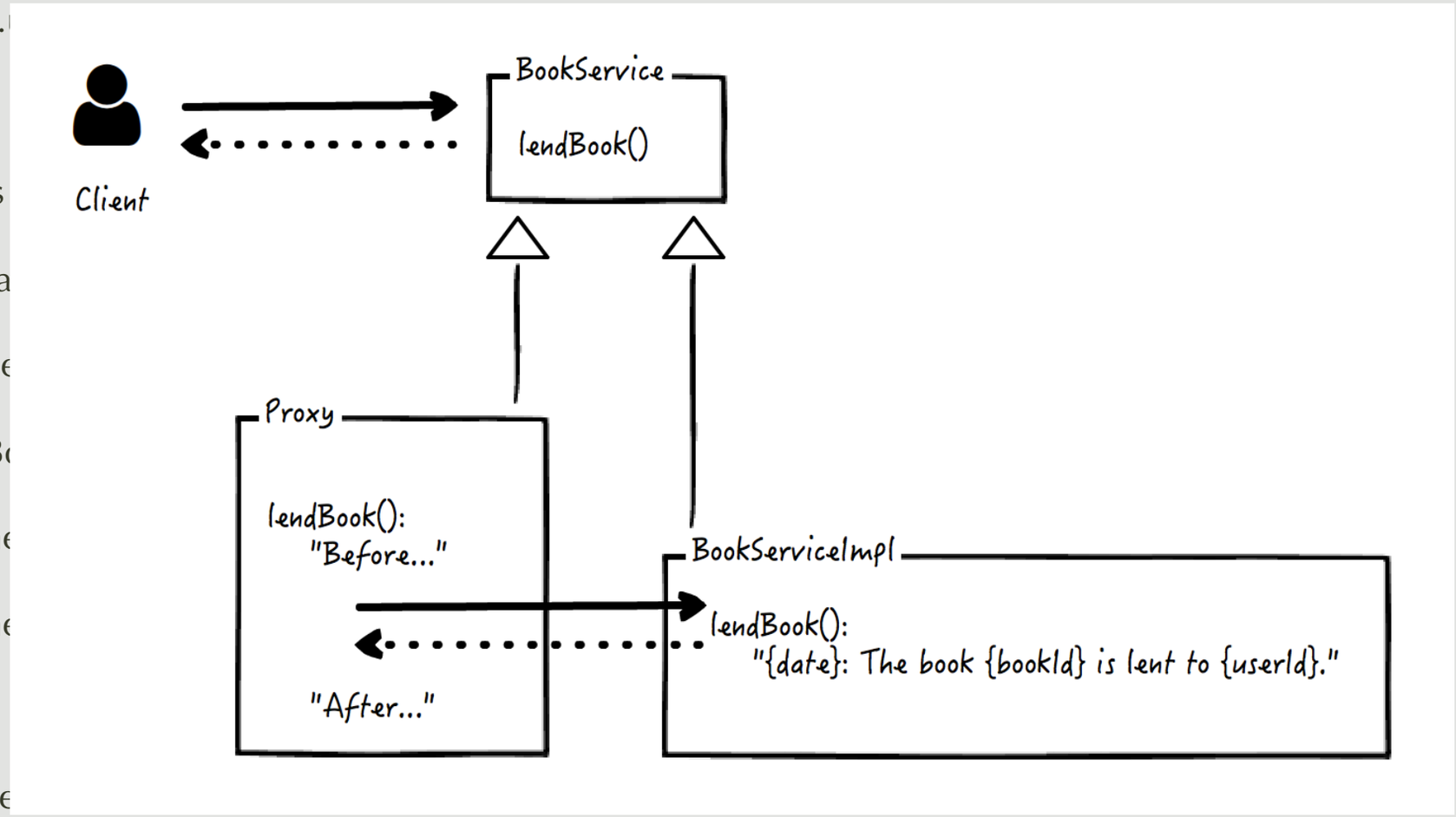
```
import java.text.MessageFormat;
import java.util.Date;


interface BookService {
    void lendOut(String bookId, String userId, Date date);
}
```

```
class BookServiceImpl implements BookService {
    @Override
    public void lendOut(String bookId, String userId, Date date) {
        System.out.println(MessageFormat.format("{0}: The book {1} is lent to {2}.", date, bookId, userId));
    }
}
```

```
import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Method;
class ServiceInvocationHandler implements InvocationHandler {
    private Object target;
    public ServiceInvocationHandler(Object target) {
        this.target = target;
    }
    @Override
    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
        System.out.println("Before...");
        Object result = method.invoke(this.target, args);
        System.out.println("After...");
        return result;
    }
}
```

- import java.lang.reflect.Proxy;
- import java.
- public class
- public sta
- BookSe
- B
- ne
- ne
-);
- bookSe
- }
- }



Objective-C的Method Swizzling

```
- (void)validStart {
    @try {
        if (self.isCancelled || self.isFinished || self.isExecuting) {
            NSLog(@"should not start");
        } else {
            [self validStart];  //[self start];
        }
    } @catch (NSException *exception) {
        NSLog(@"exception");
    } @finally {
        NSLog(@"@finally");
    }

```

- + (void)load {
- [super load];
- Class class = [self class];
- Method oldMethod = class_getInstanceMethod(class, NSSelectorFromString(@"start"));
- Method newMethod = class_getInstanceMethod(class, NSSelectorFromString(@"validStart"));
- method_exchangeImplementations(oldMethod, newMethod);
- }

- 编译期AOP

- 编译期就把切面代码和业务代码链接起来，需要编译器或编译工具支持

- 有些编程语言编译期生成中间代码的话，较容易支持这种AOP实现

- Java的AspectJ，Flutter的Dart，Android的Gradle插件Transform API

移动端语言对于AOP的支持

- Java：支持运行时AOP和编译时AOP
- Objective-C支持运行时AOP
- kotlin支持编译期AOP（它编译成字节码）
- swift支持运行时AOP（基于Objective-C的特性，局限于OC代码）
- Flutter Dart支持编译期AOP，编译中间产物是dill（修改编译工具）

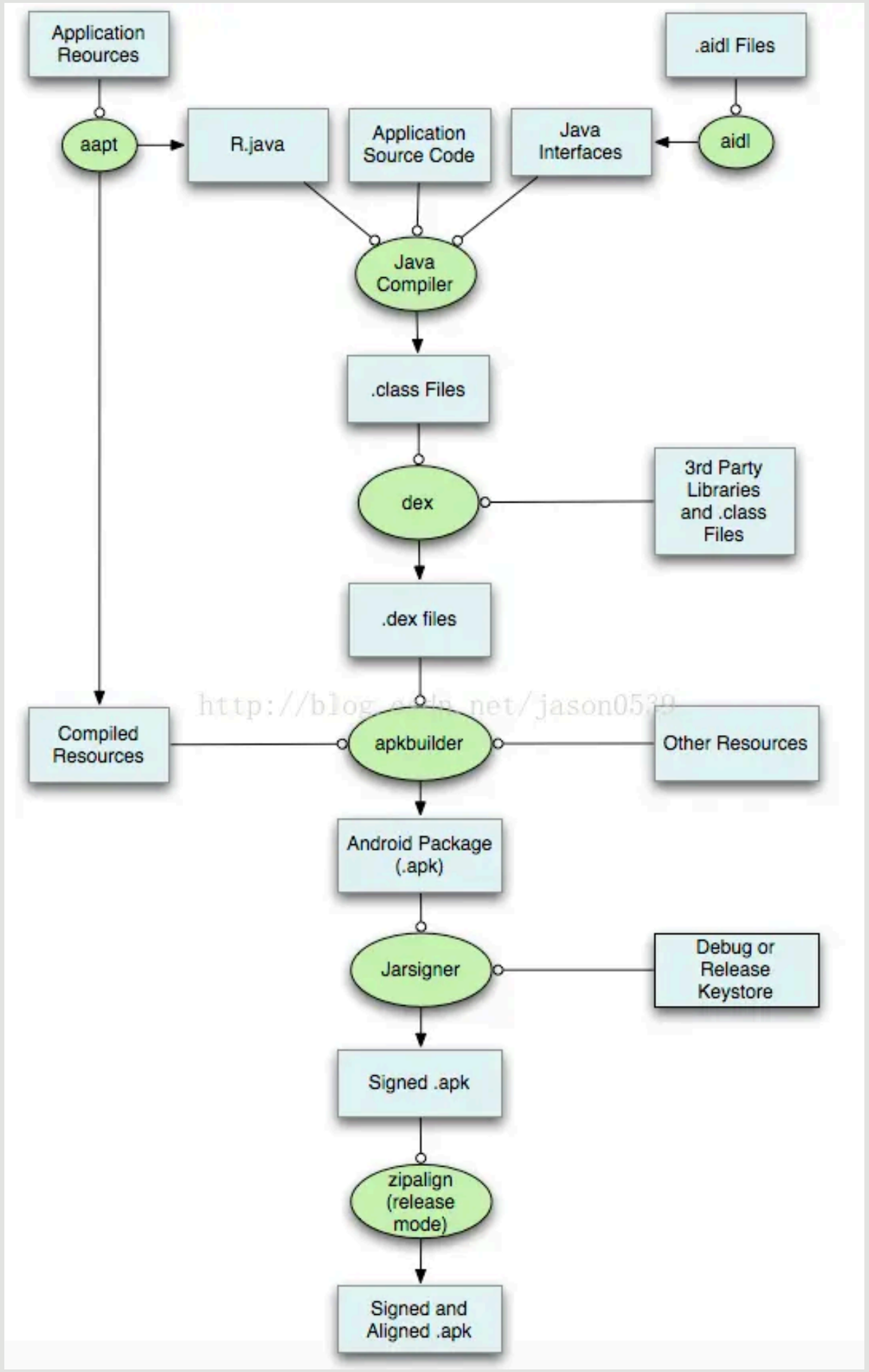
AOP在移动端应用：无埋点统计

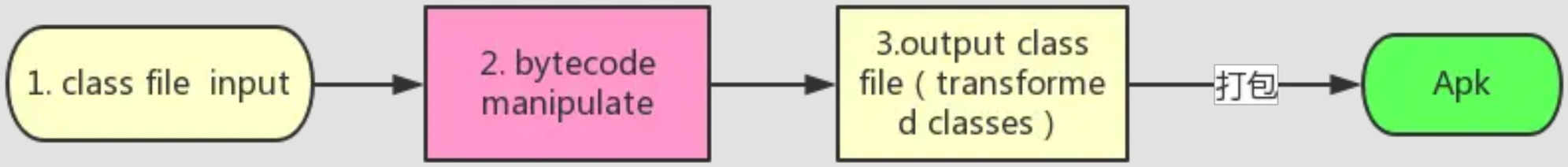
- 传统埋点代码涉及各个模块，有点繁琐又没有太大技术含量
- 无埋点采用AOP实现，解耦代码，且不容易漏统计。业务开发人员只需关注自己的业务代码

Android AOP实现

javac：将源文件编译成class格式的文件

dex：将class格式的文件汇总到dex格式的文件中





- 遍历所有要编译的class文件并对其中符合条件的方法进行修改，注入我们要调用的SDK数据搜集代码，从而实现自动埋点的目的

- Android Gradle 工具在 1.5.0 版本后提供了 Transfrom API, 允许第三方 Plugin 在打包 dex 文件之前的编译过程中操作 .class 文件

参考文献

AOP设计原理：https://www.jianshu.com/p/9f0a98ce8a8f

Spring 框架简介: https://www.ibm.com/developerworks/cn/java/wa-spring1/

Android面向切面编程（AOP）:https://www.jianshu.com/p/aa1112dbebc7

Android AOP之字节码插桩: https://www.jianshu.com/p/c202853059b4

应用于Android无埋点的Gradle插件解析: https://www.jianshu.com/p/250c83449dc0

AOP for Flutter: AspectD(闲鱼团队开源框架): https://yq.aliyun.com/articles/705751

58无埋点数据采集技术在Android端实践: https://juejin.im/entry/5b2400bc51882574b55e4fc7

如何理解 Transform API: https://www.jianshu.com/p/37df81365edf