**Cat 客户端埋点插件用户手册**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本 | 作者 | 日期 | 项目版本 | 修改内容 | 备注 |
| V0.1 | 李淑光 | 2017.12.14 | Agent:0.0.1-SNAPSHOT  Plugin: 0.0.1-SNAPSHOT | 初始化内容 |  |
| V0.2 | 李淑光 | 2019.3.5 | Agent:0.0.1-SNAPSHOT  Plugin:  0.0.2-SNAPSHOT | 增加Http调用 feign-okHttp调用埋点 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目录

1. [概述 3](#_Toc2700242)
2. [使用方法 3](#_Toc2700243)

[1) 说明 3](#_Toc2700244)

1. [埋点列表。 5](#_Toc2700245)

[1. Redis (jedis客户端） 5](#_Toc2700246)

[2. MongoDB(mongo-java-driver) 6](#_Toc2700247)

[3. 资源池(apache common-pool) 7](#_Toc2700248)

[4. Mysql (通过PreparedStatement 访问） 8](#_Toc2700249)

[5. dubbo调用 8](#_Toc2700250)

[6. Http远程调用 9](#_Toc2700251)

[7. Spring Controller 11](#_Toc2700252)

[8. Spring Service 11](#_Toc2700253)

[9. Web Filter 12](#_Toc2700254)

[10. Thread 12](#_Toc2700255)

[11. URL 13](#_Toc2700256)

1. [有关aop的几点思考 14](#_Toc2700257)

[AOP优势： 14](#_Toc2700258)

[AOP劣势： 14](#_Toc2700259)

1. [常见问题 15](#_Toc2700260)

# 概述

此项目包含两个应用，主要目的是实现对java应用无侵入式cat埋点。

1. cat-client-agent: 考虑到Cat埋点最好实现无侵入性，所需cat-client相关jar包与业务应用jar包不放在一起，因此实现了自定义类路径加载，仅此而已。 如果应用里面jar包已经包含了Cat-client相关jar包，可以不用此agent。

2. cat-client-plugin: 实现了一些常用组件的aop监控，没有什么高深的技术，主要是依靠AspectJ的LTW(load time weaver)功能，按照它要求的格式做了几个组件的Cat埋点，埋点的切入点不一定都符合所有项目需要，发的Cat消息也不一定符合项目需要，不过这些都可以比较方便的修改。

# 使用方法

启动命令行中加入

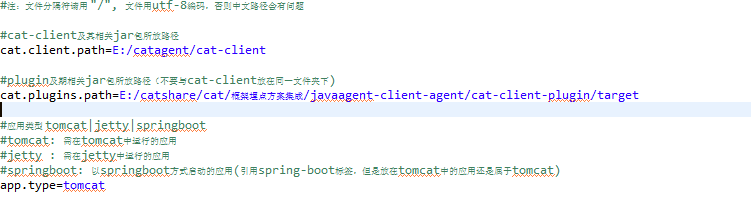
-javaagent:E:\catshare\cat\框架埋点方案集成\javaagent-client-agent\cat-client-agent\target\cat-client-agent-0.0.1-SNAPSHOT.jar=e:\catagent-conf.properties -javaagent:C:\Users\andersen\.m2\repository\org\aspectj\aspectjweaver\1.8.10\aspectjweaver-1.8.10.jar -Dorg.aspectj.weaver.loadtime.configuration=file:/E:/aop.xml -DCATPLUGIN\_CONF=e:\catplugin-conf.properties

## 1) 说明

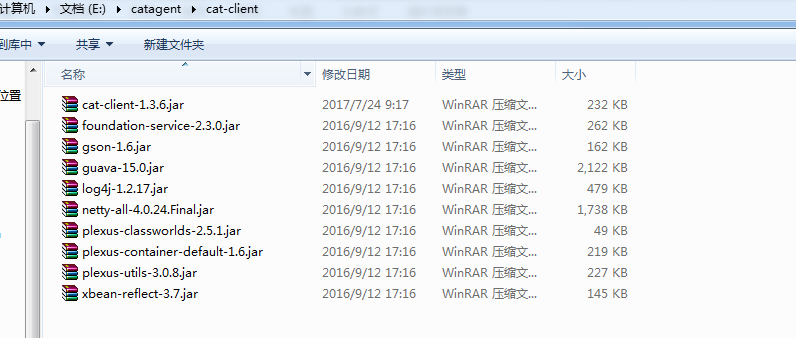
**1**. 第一个javaagent 主要是实现把Cat-client相关jar包和cat-client-plugin相关jar包加入到应用路径中，避免把这些jar包直接加入到应用的依赖或lib中。 如感觉把这些jar包和业务应用所需jar包放在一起也可以，可以不用写。

**2**. 第一个javaagent后面跟的catagent-conf.properties里主要定义了cat-client相关jar包路径和cat-client-plugin相关jar包路径，之所以要区别开来，是因为cat-client相关jar包最好保持一个JVM一个只有一个classloader加载(否则可能会出现两个MessageIdFactory类，出现id重复），而且cat-client这几个jar包不再依赖业务应用jar包； cat-agent-plugin相关jar包不一样，里面做埋点时，一般会用到业务应用中用到组件的类（如mysql-connector-java.jar包中的com.mysql.jdbc.PreparedStatement），所以必须设置和业务应用（被监控应用）一样的classpath。 第三个参数 app.type 非常关键，它决定了jvm用哪个classloader加载cat-client和cat-agent-plugin 相关jar包。

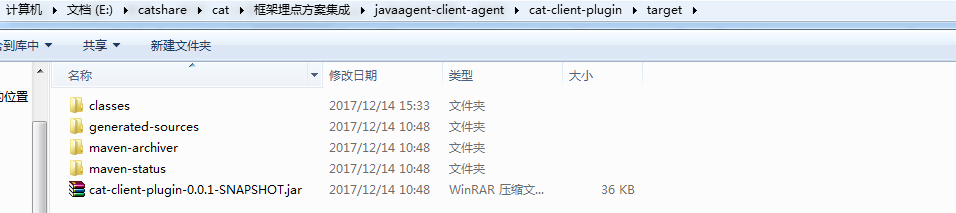
附：catagent-conf.properteis内容截图



cat-client相关jar包



cat-agent-plugin相关jar包



**3**. 第二个javaagent定义了用aspectj做aop，这是aspectj的规定写法，没什么妥协的余地。 注意： 1. 如果应用里已经用了aspectjweaver.jar包，此处最好用应用里的aspectjweaver包。 2. aspectjweaver与jdk有版本对应关系，本地测试出来是aspectjweaver-1.6.9支持jdk1.7，aspectjweaver-1.8.10支持jdk1.8。 **如果应用中有aspectjweaver 且不支持运行时jdk，则需要调整应用中的aspectjweaver版本，这是一个非常大的限制**

**4.** -Dorg.aspectj.weaver.loadtime.configuration 参数定义了aop.xml文件的位置，如果不定义，则aspectj默认会汇总所有classpath上META-INF/aop.xml， META-INF/aop-ajc.xml, org/aspectj/aop.xml文件

aop.xml文件的规范请参考aspectjweaver.jar包里面的aspectj\_\*.\*.\*.dtd

注： 使用时在weaver部分，尽可能缩窄include的范围， 只要包含那些expression里面用到的类即可，这样可以节省其它类加载时间

**5**. -DCATPLUGIN\_CONF=e:\catplugin-conf.properties 定义了cat-agent-plugin里面使用的配置（如给mysql埋点时，可以配置是否把填充过参数的sql打出来）

附： 文件内容截图



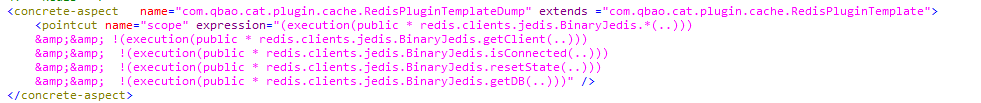
# 埋点列表。

### 1. Redis (jedis客户端）

本地测试版本：jedis 2.7.2

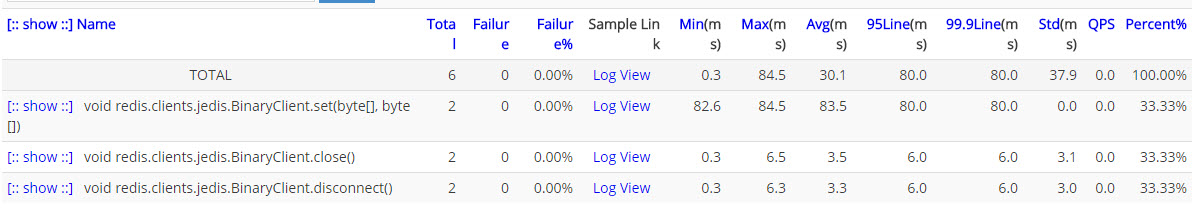
实现类：com.qbao.cat.plugin.cache.RedisPluginTemplate

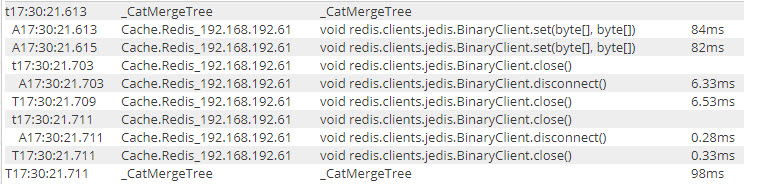
埋点配置（注： name只是经过aspectj后，新生成类的名字，一般是我们的模板名+Dump，对应用没什么用，以下同)：



cat输出：

redis-simple-type.jpg





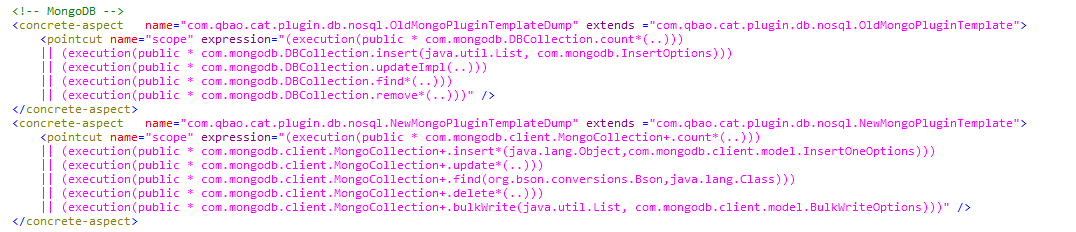
### 2. MongoDB(mongo-java-driver)

版本：3.2.2

实现类： com.qbao.cat.plugin.db.nosql.OldMongoPluginTemplate

com.qbao.cat.plugin.db.nosql.NewMongoPluginTemplate

埋点配置：

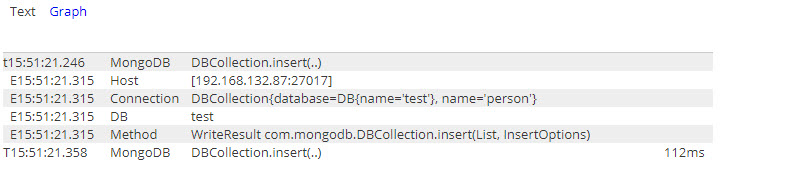


cat输出：

mongo-old

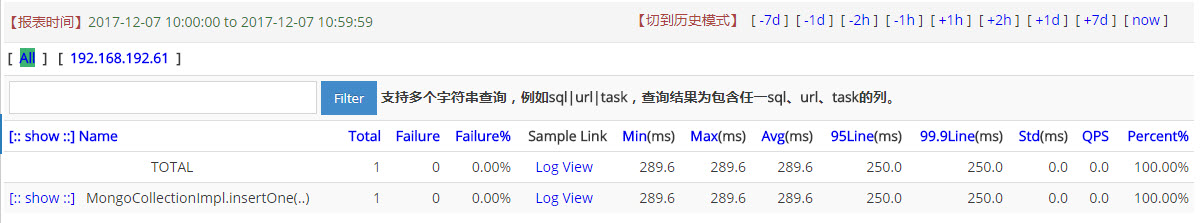


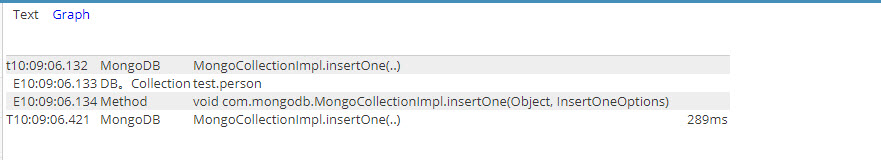




mongo-new







### 3. 资源池(apache common-pool)

版本：commons-pool 1.6

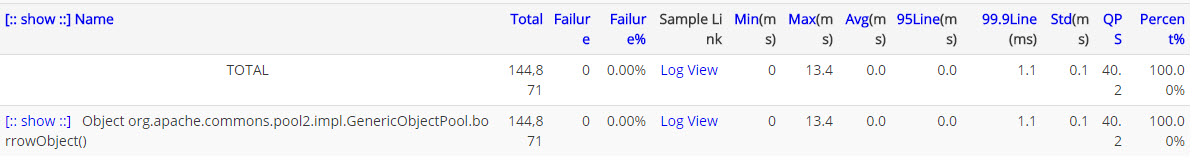
实现类：com.qbao.cat.plugin.common.CommonPluginTemplate

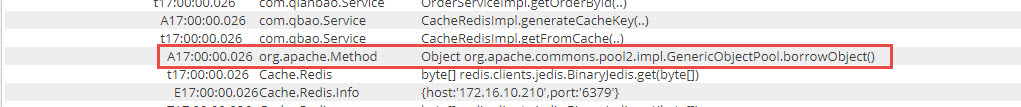
埋点配置：



cat输出：

pool-type.jpg



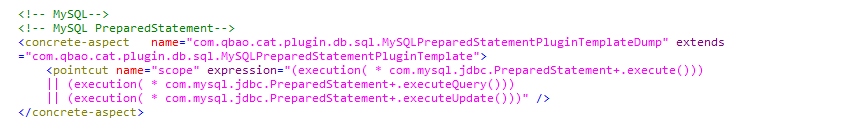


### 4. Mysql (通过PreparedStatement 访问）

版本：mysql-connector-java 5.1.8

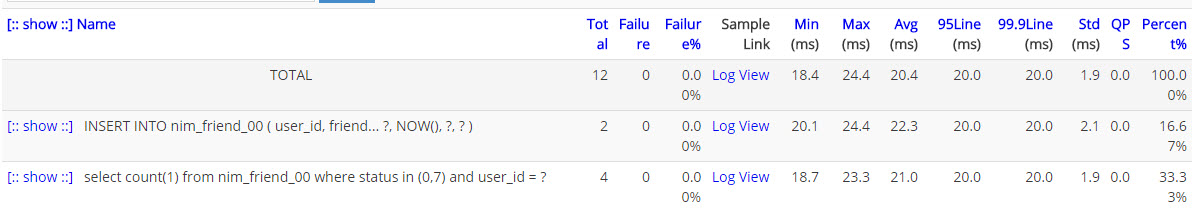
实现类：com.qbao.cat.plugin.db.sql.MySQLPreparedStatementPluginTemplate

埋点配置(可能会有遗漏，请根据需要添加）：



cat输出：

mysqlps-type.jpg





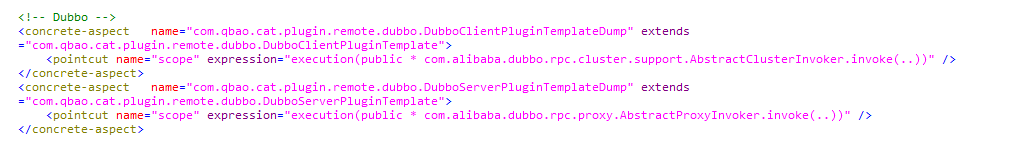
### 5. dubbo调用

版本： dubbo 2.8.4

实现类：com.qbao.cat.plugin.remote.dubbo.DubboClientPluginTemplate

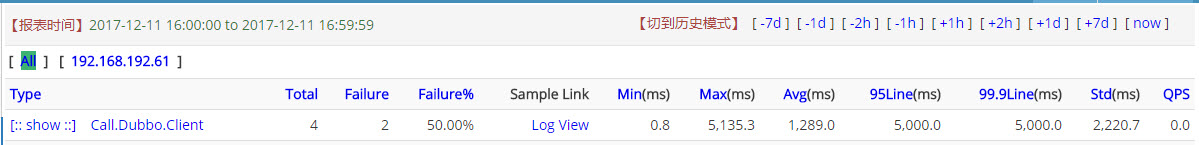
com.qbao.cat.plugin.remote.dubbo.DubboServerPluginTemplate

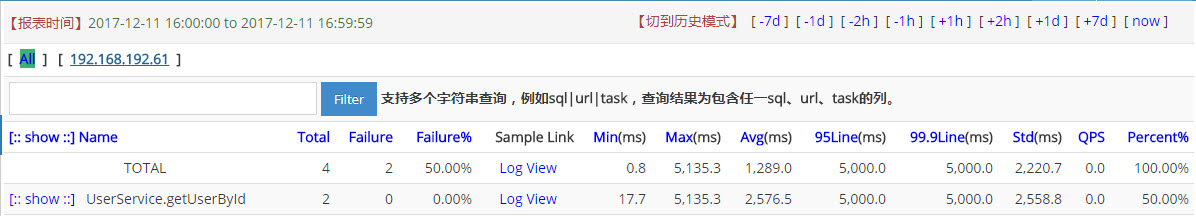
埋点配置

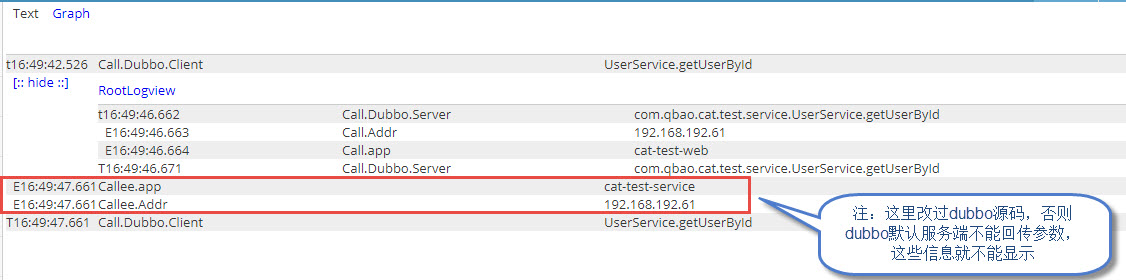


cat输出：

客户端



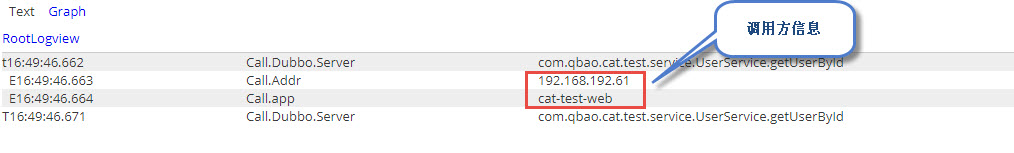




服务端

dubbo-server-type.jpg





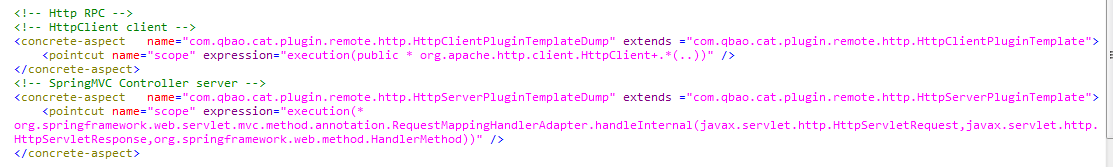
### 6. Http远程调用

版本： HttpClient 4.4.1 Spring 3.2.9

实现类： com.qbao.cat.plugin.remote.http.HttpClientPluginTemplate

com.qbao.cat.plugin.remote.http.HttpServerPluginTemplate

埋点配置（注：用其它客户端或服务端的需另外埋点及写对应template)：

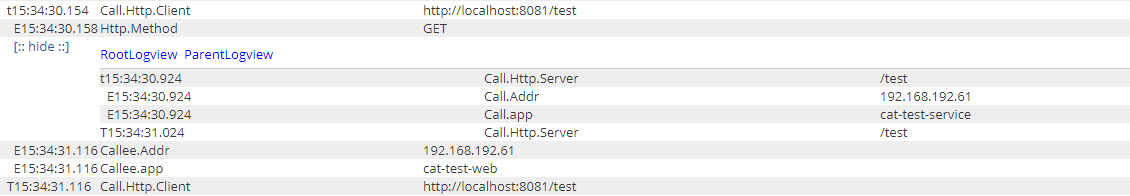


cat输出：

客户端：

http-client-type.jpg

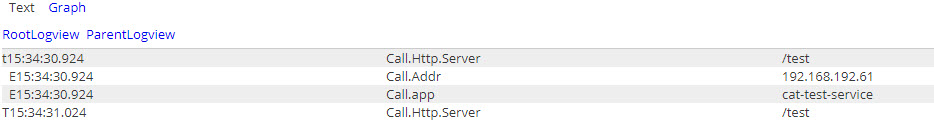




服务端：

http-server-type.jpg



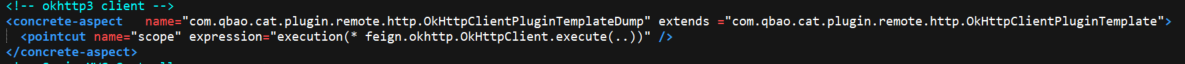


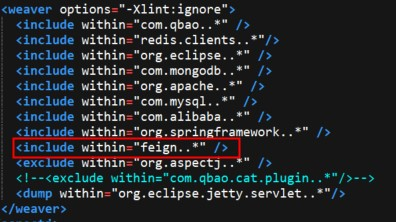
##### 6.2 Feign-Okhttp调用

版本：feign-okhtt 10.1.0

实现类：com.qbao.cat.plugin.remote.http.OkHttpClientPluginTemplate

埋点配置：





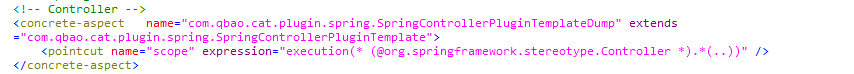
Cat输出：同HttpClient相同

### 7. Spring Controller

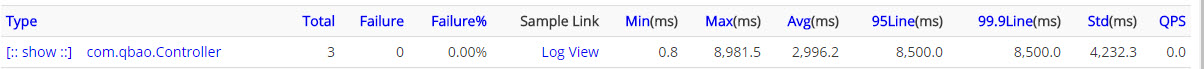
版本： Spring 3.2.9

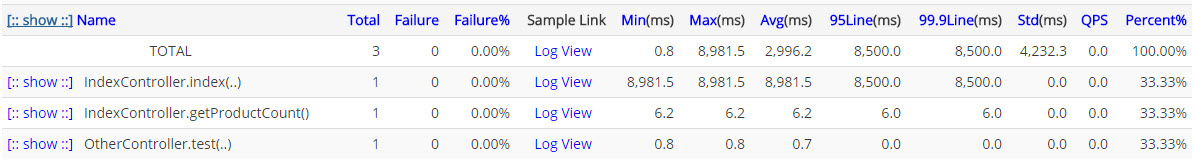
实现类：com.qbao.cat.plugin.spring.SpringControllerPluginTemplate

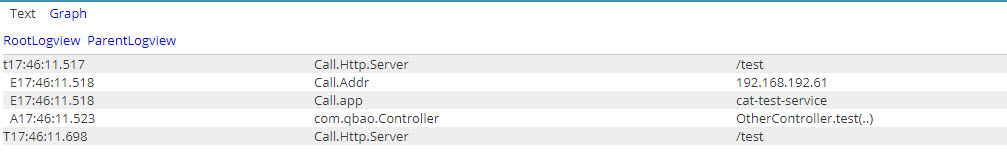
埋点配置：



cat输出：





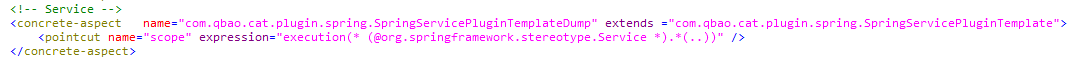


### 8. Spring Service

版本：Spring 3.2.9

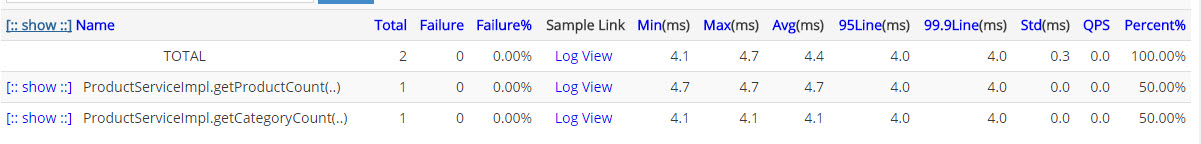
实现类：com.qbao.cat.plugin.spring.SpringServicePluginTemplate

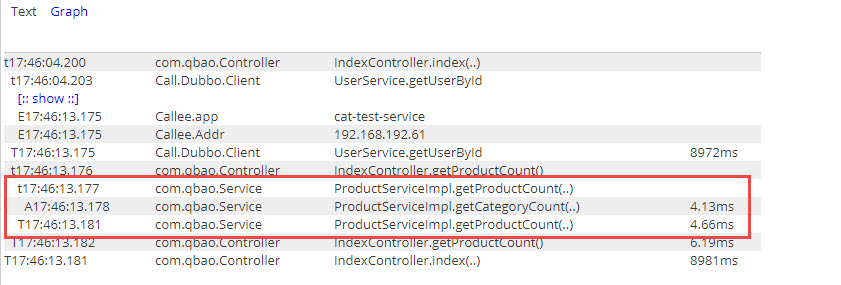
埋点配置：



cat输出：

spring-service-type.jpg





### 9. Web Filter

版本：Servlet 2.5

实现类：com.qbao.cat.plugin.web.FilterPluginTemplate

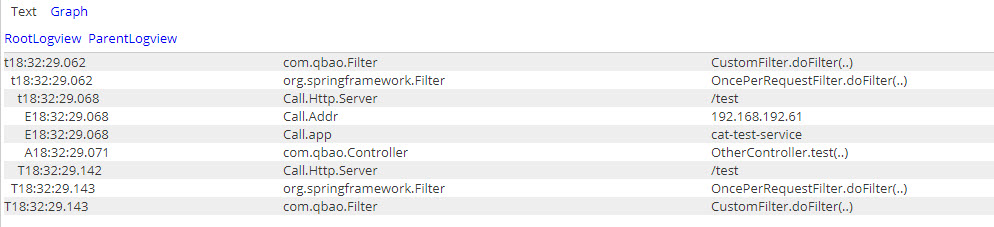
埋点配置：



cat输出：

web-filter-type.jpg



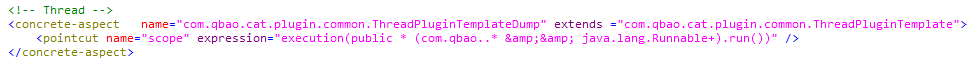


### 10. Thread

版本： 暂无限制

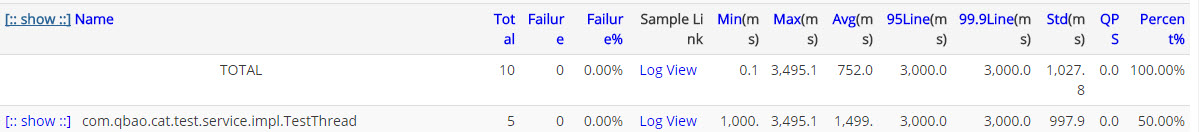
实现类： com.qbao.cat.plugin.common.ThreadPluginTemplate

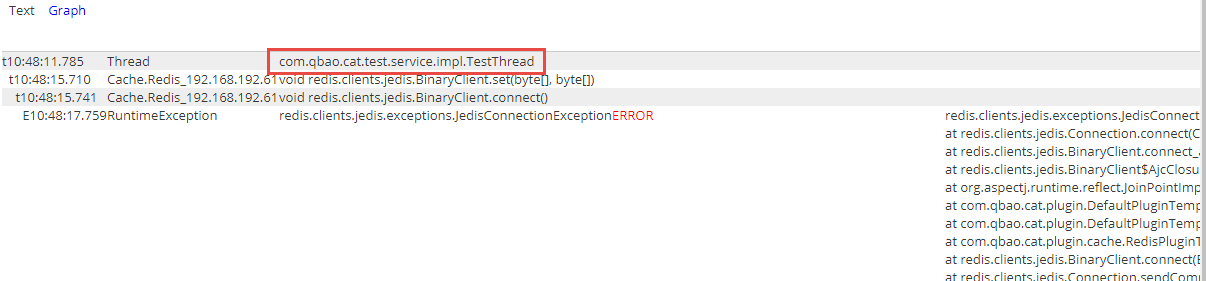
埋点配置：



cat输出：







### 11. URL

版本：servlet 2.3

实现类： com.qbao.cat.plugin.web.TomcatUrlPluginTemplate （针对tomcat和内嵌了tomcat的springboot应用）

com.qbao.cat.plugin.web. JettyUrlPluginTemplate （针对使用eclipse jetty插件启动的web应用）

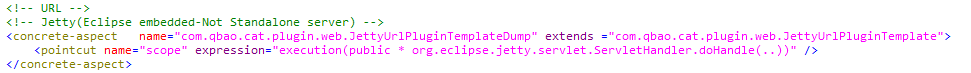
注： 因url的埋点是在容器类中埋点，而容器类可能与应用类不一个classloader, 因此如果启用此处埋点，最好启动时要加 -javaagent:/\*\*\*/cat-client-agent.jar命令来调节下classloader。

埋点配置：

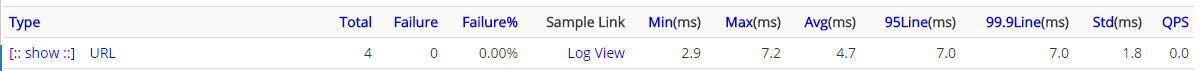
tomcat(springboot)

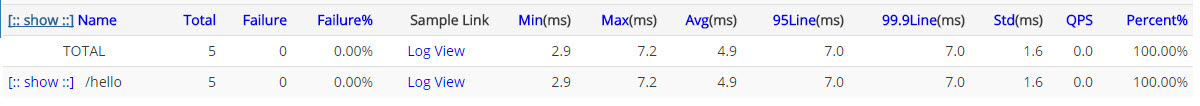


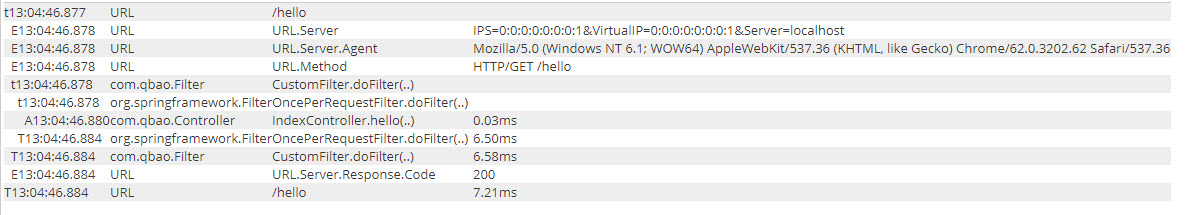
jetty



cat输出：







# 有关aop的几点思考

AOP是一把双刃剑，用得好事半功倍，用得不好事倍功半！！！

### AOP优势：

1. 对业务透明。尤其是用javaagent的方式，比用spring aop更透明，不用修改业务应用的代码和配置。当然还有一种更纯粹的agentlib方式，独立于JVM,可以用其它语言写的，不过我不会其它语言。

2. 埋点方便。 针对某个类或方法的访问，一个切入点就能选择了，如果在业务代码里面埋点，可能需要在多个地方修改代码。

3. 埋点不会遗漏。 如果切入点选的好，肯定不会有遗漏场景的情况，如果在业务代码里面埋点，有可能遗漏某个场景或分支。

4. 不需要源代码。 aop只需要class文件，不需要源文件，在没有源代码的情况下只能使用aop。

### AOP劣势：

1. AOP框架一般会生成代理类,造成JVM加载更多的类，更容易造成PermGen space 的OOM

2. AOP框架一般会在类加载时做些逻辑处理，造成类加载时间变长，对于负载非常大的系统，需要类做预加载

3. 选择切入点是aop的关键，特别是要监控第三方组件的时候，需要对第三方组件非常熟悉，否则可能会达不到目的，反而会浪费资源（如级联调用方法一般只监控最后的方法，如果所有的方法都监控，会造成资源浪费）

4. 在应用出错打印调用堆栈时，aop会打印出一些奇怪的信息，干扰日志的可读性。

5. 现在所用的aspectj框架只适用于java语言，对于其它语言暂时还无能为力。

# 常见问题

1。 启动时报错： Invalid byte tag in constant pool: \*\*

主要是因为aspectj包不支持当前项目所用jdk的原因，如用aspectweaver-1.6.9来监控jdk8的项目，就会出现此种错误。 可以在项目中或监控时改用更高版本的aspectjweaver

2. 启动时报错： error Cannot find parent aspect for: <concrete-aspect name='com.qbao.cat.plugin.cache.RedisPluginTemplateDump' extends='com.qbao.cat.plugin.cache.RedisPluginTemplate' perclause='null'/>

主要是因为aspectj尝试在每一个classloader里面加载截面类（即aop.xml里面的extends类），如果找不到，就会报这个错，不影响使用（因为我们的应用可能就只在一个classloader中使用，截面类在这个classloader中能找到就可以），如果实在不想显示，可以启动时加上参数-Daj.weaving.loadersToSkip=sun.misc.Launcher$ExtClassLoader,\*.\*.\*ClassLoader 来避免