**课程概要：**

1. 埋点插桩基础知识
2. Servlet处理埋点
3. Redis 调用埋点

一、埋点插桩基础知识

**什么是埋点：**

埋点就是在方法前后插入代码或指令，去获取方法的执行时间或参数等信息。

常用办法有：

1. 硬编码
2. AOP拦截
3. 动态字节码插桩

.java ====> class ===>系统指令

翻译 成我们人类可以阅读的系统指令

**什么是插桩：**

JAVA源代码要先编译成Class文件，文件的内容就由若干条JVM指令组成的集合（即代码逻辑）。插桩的过程就是将这这些指令，拆开来，然后在插入监控所需指令，最后进行重新组装生成新的Class字节。

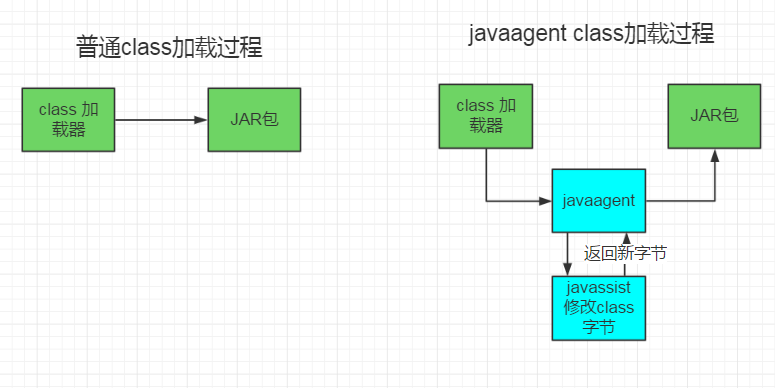
* 演示将Class转换成可读的指令码

**字节码插桩所用到的技术：**

1. javaagent  代理拦截（插桩的入口）
2. javassist 字节码修改工具 (怎么插)

**Java Agent**

javaagent 是java1.5之后引入的特性，其主要作用是在class 被加载之前对其拦截，已插入我们的监听字节码



**javassit介绍**

Javassist是一个开源的分析、编辑和创建Java字节码的类库。其主要的优点，在于简单，而且快速。直接使用java编码的形式，而不需要了解虚拟机指令，就能动态改变类的结构，或者动态生成

**注**：也可以使用ASM实现，但需要会操作字节码指令，学习使用成本高

* javaagent 快速演示
* javassist 快速演示

二、Servlet处理埋点

**Servlet埋点目的**

1. 生成TraceId
2. 开启关闭监控会话
3. 捕捉Http请求（url、客户端IP、参数、响应时长、响应状态码）

**埋点埋在哪？**

* 1. 每一个Control方法
  2. DispatcherServlet.doDispatch方法
  3. HttpServlet.service 方法

**方案对比：**

|  |  |  |
| --- | --- | --- |
| 方案 | 优点 | 缺点 |
| 应用层Control类 | 简单，风险因素低 | 判别成本高，有局限性，只能根据 HttpServlet 子类或@RequestMapping进行识别。 |
| DispatcherServlet.doDispatch | 简单，适应性强 | 1、只能针对spring mvc 项目 2、spring boot 项目不支持 |
| HttpServlet.service | 适应性强，与应用层和框架无关 | 1、不同的容器ClassPath不一样，存在兼容性问题。 2、存在风险，几乎所有请求都会经过此方法 3、业务异常无法捕获 |

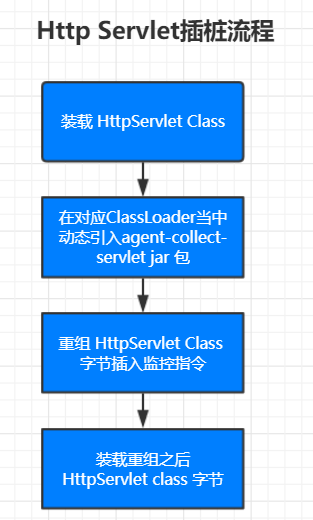
总合比较还是选择 HttpServlet.service 会更好些。

HttpServlet.service 埋点需要做的工作：

1. 字节码插桩
2. 请求拦截并获取请求信息

**字节码插桩流程：**

字节码插是指在数据装载前在HttpServlet.service 插入监控指令，以拦截Http请求，其插桩的过程如下图。

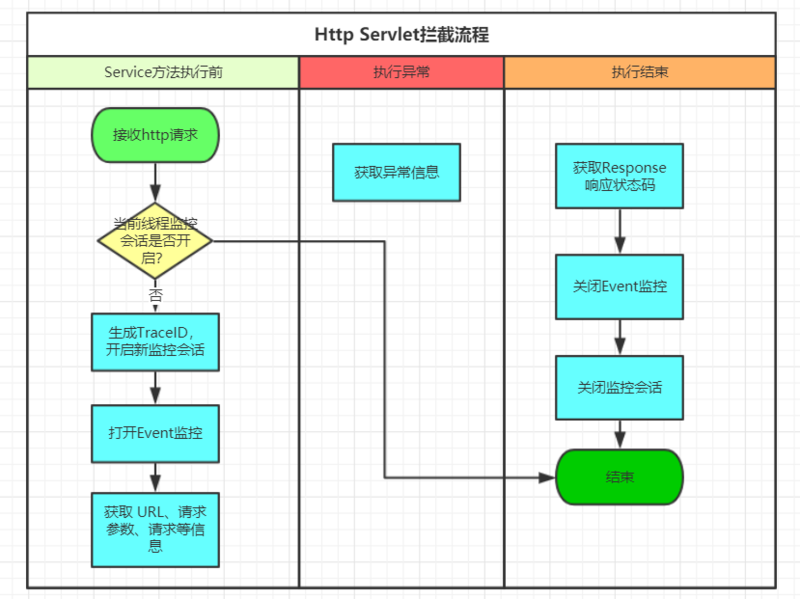


* 代码演示插桩过程

**请求拦截流程：**

请求拦截是指具体Http请求过来时进行拦截过滤，这么做主要是为了完成两个目的：

* 1. 开启监控会话
  2. 开启对Servlet响应过程的监控

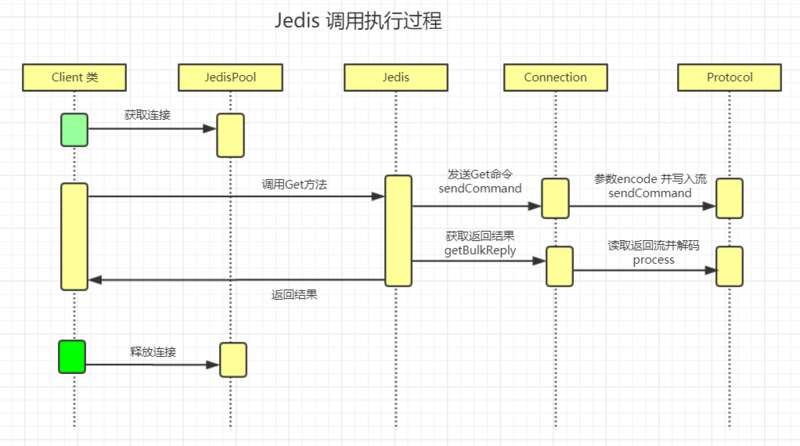


**代码演示追踪过程：**

* 方法开始 开启监控会话
* 采集请求信息
* 采集响应信息
* 关闭监控会话

三、Redis 调用埋点

**Redis调用过程：**



**埋点可选方案：**

|  |  |  |
| --- | --- | --- |
| **方案名称** | **优点** | **缺点** |
| 埋点jedis 类Get、Set等API方法 | 简单直接 | 工作量大，方法较多、需要了解每个方法特性 |
| 埋点 Connection sendCommand方法 | 全面、所有命令都会经过此方法 | 存在未知风险、不方便计算执行时间、和返回结果 |
| 埋点 Protocol | 全面、所有命令都会经过此方法 | 存在未知风险、不方便计算执行时间、和返回结果 |

目前采用的办法是 对Protocol 的sendCommand 和 process方法

进行埋点，两个方法分别采集 参数信息和返回结果，也可以基于时间差计算执行用时。虽然两个方法都属于静态方法，但并不影响执行效果。

**采集流程：**

基于sendCommand 采集参数，同时屏蔽 exists、client、ping等命令

1. exists

* 代码演示Redis采集过程