

Server Solutions

陈磊

2017 年 5 月 18 日

目录

1 面向切面编程 (AOP)	1
1.1 AOP 统一记录 HTTP 请求日志	1
1.1.1 环境	1
1.1.2 配置	1
1.1.3 AOP 日志记录实现	2
2 maven 使用相关	4
2.1 不同环境 (开发, 上线) 配置切换	4
2.2 本地 JAR 文件引入	5

1 面向切面编程 (AOP)

1.1 AOP 统一记录 HTTP 请求日志

- 实现思路同样适用于非 HTTP 请求类型日志记录.
- 本文需求是: 通过日志记录 Controller 中的请求.
- 本文不对日志相关的配置作说明.
- 完整示例可以直接看参考.

1.1.1 环境

- apache-tomcat-8.5.11
- jdk1.8.0_121 (1.7 也可以)

1.1.2 配置

maven pom.xml 配置:

```
1 <dependency>
2   <groupId>org.aspectj</groupId>
3   <artifactId>aspectjrt</artifactId>
4   <version>1.8.4</version>
5 </dependency>
6 <dependency>
7   <groupId>org.aspectj</groupId>
8   <artifactId>aspectjweaver</artifactId>
9   <version>1.8.4</version>
10 </dependency>
11 <dependency>
12   <groupId>cglib</groupId>
13   <artifactId>cglib</artifactId>
14   <version>2.2</version>
15 </dependency>
```

dispatcher-servlet.xml 配置:

```
1 <context:component-scan base-package="com.xx.xxxx" />
2 <aop:aspectj-autoproxy />
```

1.1.3 AOP 日志记录实现

```
1 import org.apache.log4j.Logger;
2 import org.aspectj.lang.JoinPoint;
3 import org.aspectj.lang.annotation.Aspect;
4 import org.aspectj.lang.annotation.Pointcut;
5 import org.aspectj.lang.annotation.AfterReturning;
6 import org.aspectj.lang.annotation.Before;
7 import org.springframework.core.annotation.Order;
8 import org.springframework.stereotype.Component;
9 import org.springframework.web.context.request.RequestContextHolder;
10 import org.springframework.web.context.request.ServletRequestAttributes;
11
12 import javax.servlet.http.HttpServletRequest;
13 import java.util.Arrays;
14
15 /**
```

```
16 * Order(3) 制定 Aspect 处理顺序, 数值越小, 优先级越高
17 */
18 @Aspect()
19 @Order(3)
20 @Component()
21 public class HttpLogAspect {
22
23     private Logger logger = Logger.getLogger(getClass());
24     private ThreadLocal<Long> startTime = new ThreadLocal<Long>(); // 记
        ↳ 录请求与响应花费的时间
25
26     @Pointcut("within(@org.springframework.stereotype.Controller *)")
27     public void controller() {}
28
29     @Pointcut("execution(* *.*(..))")
30     protected void allMethod() {}
31
32     /**
33     * 执行前
34     * 记录 HTTP 请求详细
35     * @param joinPoint joinPoint
36     */
37     @Before("controller() && allMethod()")
38     public void logBefore(JoinPoint joinPoint) {
39         // 开始计时
40         startTime.set(System.currentTimeMillis());
41
42         logger.info("** START HTTP REQUEST **");
43
44         ServletRequestAttributes attributes = (ServletRequestAttributes)
        ↳ RequestContextHolder.getRequestAttributes();
45         HttpServletRequest request = attributes.getRequest();
46
47         // 记录类名及方法名
48         logger.info("HTTP_CLASS_METHOD : " + joinPoint.getSignature().
        ↳ getDeclaringTypeName() + "."
49             + joinPoint.getSignature().getName());
50         // 记录请求参数
51         logger.info("HTTP_ARGUMENTS : " + Arrays.toString(joinPoint.
        ↳ getArgs()));
```

```

52
53     if (null != request) {
54         // 记录请求地址
55         logger.info("HTTP_REQUEST_URL : " + request.getRequestURL().
                    ↪ toString());
56         // 记录请求方法
57         logger.info("HTTP_METHOD : " + request.getMethod());
58         // 记录请求 IP
59         logger.info("HTTP_REQUEST_IP : " + request.getRemoteAddr());
60     }
61 }
62
63 /**
64  * 执行后
65  * 请求结束，记录返回内容
66  * @param result 响应内容
67  */
68 @AfterReturning(pointcut = "controller() && allMethod()", returning =
                    ↪ "result")
69 public void logAfterReturning(Object result) {
70     logger.info("HTTP_RESPONSE : " + result);
71     // 结束计时
72     logger.info("HTTP_SPEND_TIME : " + (System.currentTimeMillis() -
                    ↪ startTime.get()) + " ms");
73     logger.info("** END HTTP REQUEST **");
74 }
75
76 }

```

2 maven 使用相关

2.1 不同环境 (开发, 上线) 配置切换

在编译时使用 maven 命令参数打包不同环境下的配置文件, 比如 `src/` ↪ `main/resources/prod` 和 `src/main/resources/dev` 文件夹下分别是线上环境和开发环境的配置文件. maven `pom.xml` 配置文件部分配置如下.

```

1 <build>
2     <resources>
3         <resource>

```

```

4         <directory>src/main/resources</directory>
5         <!-- 资源根目录排除各环境的配置，使用单独的资源目录来指定 -->
6         <excludes>
7             <exclude>prod/*</exclude>
8             <exclude>dev/*</exclude>
9         </excludes>
10        </resource>
11        <resource>
12            <directory>src/main/resources/${profiles.active}</directory>
13        </resource>
14    </resources>
15</build>
16<profiles>
17    <profile>
18        <!-- 开发环境 -->
19        <id>dev</id>
20        <properties>
21            <profiles.active>dev</profiles.active>
22        </properties>
23        <activation>
24            <activeByDefault>true</activeByDefault>
25        </activation>
26    </profile>
27    <profile>
28        <!-- 生产环境 -->
29        <id>prod</id>
30        <properties>
31            <profiles.active>prod</profiles.active>
32        </properties>
33    </profile>
34</profiles>

```

打包时通过 `mvn clean package -P prod` 实现线上环境配置打包, `mvn`
`clean package -P dev` 实现开发环境配置打包.

2.2 本地 JAR 文件引入

本地 JAR 文件加入到本地 maven 库

```

1 <!--build>plugins 下添加-->
2 <!--安装本地 jar 包-->

```

```

3 <!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-
   ↪ install-plugin -->
4 <plugin>
5   <groupId>org.apache.maven.plugins</groupId>
6   <artifactId>maven-install-plugin</artifactId>
7   <version>2.5.2</version>
8   <configuration>
9     <groupId>com.mycompany</groupId>
10    <artifactId>myproject</artifactId>
11    <version>1.0</version>
12    <packaging>jar</packaging>
13    <generatePom>true</generatePom>
14    <file>${basedir}/src/main/webapp/WEB-INF/lib/myjar.jar</file>
15  </configuration>
16  <executions>
17    <execution>
18      <id>install-jar-lib</id>
19      <goals>
20        <goal>install-file</goal>
21      </goals>
22      <phase>validate</phase>
23    </execution>
24  </executions>
25 </plugin>

```

依赖添加

```

1 <!--dependencies 下添加-->
2 <dependency>
3   <groupId>com.mycompany</groupId>
4   <artifactId>myproject</artifactId>
5   <version>1.0</version>
6 </dependency>

```

打包前先执行 `mvn install:install-file` 或者 `mvn validate`. 或者一行命令执行 `mvn validate & mvn clean package -P dev`.

- Maven 安装 jar 文件到本地仓库
- Install local jar dependency as part of the lifecycle, before Maven attempts to resolve it