Server Solutions

陈磊

March 15, 2017

Contents

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 环境

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

1.2 配置

maven `pom.xml` 配置:

```
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjrt</artifactId>
    <version>1.8.4</version>
```

```
</dependency>
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>1.8.4</version>
</dependency>
<dependency>
    <groupId>cglib</groupId>
    <artifactId>cglib</artifactId>
    <version>2.2</version>
</dependency>
```

dispatcher-servlet.xml 配置:

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

1.3 AOP 日志记录实现

```
import org.apache.log4j.Logger;
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;
import org.aspectj.lang.annotation.AfterReturning;
import org.aspectj.lang.annotation.Before;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;
import org.springframework.web.context.request.RequestContextHolder;
import org.springframework.web.context.request.ServletRequestAttributes;

import javax.servlet.http.HttpServletRequest;
import java.util.Arrays;

/**
* Order(3) 制定 Aspect 处理顺序, 数值越小, 优先级越高
*/
@Aspect()
@Order(3)
@Component()
public class HttpLogAspect {
```

```
 private Logger logger = Logger.getLogger(getClass());
private ThreadLocal<Long> startTime = new ThreadLocal<Long>(); // 记录请求与响应花费的时间

 @Pointcut("within(@org.springframework.stereotype.Controller *)")
 public void controller() {}

 @Pointcut("execution(* *.*(..))")
 protected void allMethod() {}

 /**
 * 执行前
 * 记录 HTTP 请求详细
 * @param joinPoint joinPoint
 */
 @Before("controller() && allMethod()")
 public void logBefore(JoinPoint joinPoint) {
     // 开始计时
     startTime.set(System.currentTimeMillis());

     logger.info("** START HTTP REQUEST **");

  ServletRequestAttributes attributes = (ServletRequestAttributes) RequestContextHolder.getRequestAt
     HttpServletRequest request = attributes.getRequest();

     // 记录类名及方法名
  logger.info("HTTP_CLASS_METHOD : " + joinPoint.getSignature().getDeclaringTypeName() + "."
          + joinPoint.getSignature().getName());
     // 记录请求参数
  logger.info("HTTP_ARGUMENTS :  " + Arrays.toString(joinPoint.getArgs()));

     if (null != request) {
         // 记录请求地址
     logger.info("HTTP_REQUEST_URL : " + request.getRequestURL().toString());
         // 记录请求方法
         logger.info("HTTP_METHOD : " + request.getMethod());
         // 记录请求 IP
         logger.info("HTTP_REQUEST_IP : " + request.getRemoteAddr());
     }
 }
```

```
    /**
    * 执行后
    * 请求结束，记录返回内容
    * @param result 响应内容
    */
   @AfterReturning(pointcut = "controller() && allMethod()", returning = "result")
    public void logAfterReturning(Object result) {
        logger.info("HTTP_RESPONSE : " + result);
        // 结束计时
     logger.info("HTTP_SPEND_TIME : " + (System.currentTimeMillis() - startTime.get()) + " ms");
        logger.info("** END HTTP REQUEST **");
    }

}
```

2 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.

```
<build>
    <resources>
        <resource>
            <directory>src/main/resources</directory>
            <!-- 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 -->
            <excludes>
                <exclude>prod/*</exclude>
                <exclude>dev/*</exclude>
            </excludes>
        </resource>
        <resource>
            <directory>src/main/resources/${profiles.active}</directory>
        </resource>
    </resources>
</build>
<profiles>
    <profile>
        <!-- 开发环境 -->
        <id>dev</id>
```

```
        <properties>
            <profiles.active>dev</profiles.active>
        </properties>
        <activation>
            <activeByDefault>true</activeByDefault>
        </activation>
    </profile>
    <profile>
        <!-- 生产环境 -->
        <id>prod</id>
        <properties>
            <profiles.active>prod</profiles.active>
        </properties>
    </profile>
</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.