链滴

spring-boot aop 打印 http 日志

作者：crick77

原文链接：https://ld246.com/article/1511023202197

来源网站：链滴

aop打印http日志

本文项目已发布到github，后续学习项目也会添加到此工程下，欢迎fork点赞。

https://github.com/wangyuheng/spring-boot-sample

需要具备少量aop基础，通过springboot构建项目方便演示。

AOP-面向切面编程

一句话描述，在java对象增加切点，在不改变对象的前提下通过代理扩展功能。

http日志打印拦截器

restful api

通过springboot快速搭建一个RestController接口。

```
import org.springframework.web.bind.annotation.*;
import wang.crick.study.httplog.annotation.HttpLog;
import wang.crick.study.httplog.domain.User;

import java.util.Random;

@RestController
@RequestMapping("/user")
public class UserApi {

    @GetMapping("/log/{id}")
    public RestApiResponse<User> getInfo(@PathVariable("id") int id,
                                         @RequestParam("age") int age){
        User user = new User();
        user.setId(id);
        user.setUsername(String.valueOf(new Random().nextLong()));
        user.setAge(age);
        return RestApiResponse.success(user);
    }

    @GetMapping("/log/pwd/{id}")
    public RestApiResponse<User> getInfoWithPwd(@PathVariable("id") int id,
                                                @RequestHeader("username") String username,
                                                @RequestHeader("password") String password){
        User user = new User();
        user.setId(id);
        user.setUsername(username);
        user.setPassword(password);
        return RestApiResponse.success(user);
    }

    @GetMapping("/log/pwdExcludeResponse/{id}")
    public RestApiResponse<User> getInfoWithPwd(@PathVariable("id") int id,
                                                @RequestParam("age") int age,
```

```
                                        @RequestHeader("password") String password){
        User user = new User();
        user.setId(id);
        user.setPassword(password);
        user.setAge(age);
        return RestApiResponse.success(user);
    }

}
```

切面选择

一般教程会选择拦截所有http请求，并打印request.parameters。但是存在问题：

1. 不够灵活，部分参数不想打印，如文件数据（过大）、敏感数据（身份证）等。
2. 显式的标注日志输出，避免给维护人员造成疑惑。
3. 部分参数通过header传输

因此，自定义日志输出@annotation **HttpLog**

```
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface HttpLog {
    /**
     * 忽略参数，避免文件or无意义参数打印
     *
     * @return 忽略参数数组
     */
    String[] exclude() default {};

    /**
     * 需要打印的header参数
     *
     * @return header参数名数组
     */
    String[] headerParams() default {};

    boolean ignoreResponse() default false;
}
```

获取HttpServletRequest

spring通过ThreadLocal持有request参数。

```
private HttpServletRequest getRequest() {
    RequestAttributes ra = RequestContextHolder.getRequestAttributes();
    ServletRequestAttributes sra = (ServletRequestAttributes) ra;
    return sra.getRequest();
}
```

获取uri

根据拦截规则不同，getServletPath()和request.getPathInfo()可能为空，简单的做一次健壮性判断。

```
private String getRequestPath(HttpServletRequest request) {
  return (null != request.getServletPath() && request.getServletPath().length() > 0)
        ? request.getServletPath() : request.getPathInfo();

}
```

aop日志输出

1. Pointcut自定义@annotation **HttpLog**
2. 拿到@annotation，读取自定义属性，如忽略response、打印headers等
3. 遍历request & headers中需打印参数
4. 定制日志格式并打印log
5. 拦截返回值并打印log

Aspect代码如下

```
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.Signature;
import org.aspectj.lang.annotation.AfterReturning;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.aspectj.lang.annotation.Pointcut;
import org.aspectj.lang.reflect.MethodSignature;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.aop.aspectj.MethodInvocationProceedingJoinPoint;
import org.springframework.web.context.request.RequestAttributes;
import org.springframework.web.context.request.RequestContextHolder;
import org.springframework.web.context.request.ServletRequestAttributes;
import wang.crick.study.httplog.annotation.HttpLog;
import wang.crick.study.httplog.api.RestApiResponse;

import javax.servlet.http.HttpServletRequest;
import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Optional;

@Aspect
public class HttpLogAspect {

    private Logger log = LoggerFactory.getLogger(HttpLogAspect.class);

    @Pointcut("@annotation(wang.crick.study.httplog.annotation.HttpLog)")
    public void logAnnotation() {
    }

    private Optional<HttpLog> getLogAnnotation(JoinPoint joinPoint) {
        if (joinPoint instanceof MethodInvocationProceedingJoinPoint) {
```

```
        Signature signature = joinPoint.getSignature();
        if (signature instanceof MethodSignature) {
            MethodSignature methodSignature = (MethodSignature) signature;
            Method method = methodSignature.getMethod();
            if (method.isAnnotationPresent(HttpLog.class)) {
                return Optional.of(method.getAnnotation(HttpLog.class));
            }
        }
    }
    return Optional.empty();
}

private HttpServletRequest getRequest() {
    RequestAttributes ra = RequestContextHolder.getRequestAttributes();
    ServletRequestAttributes sra = (ServletRequestAttributes) ra;
    return sra.getRequest();
}

private String getRequestPath(HttpServletRequest request) {
    return (null != request.getServletPath() && request.getServletPath().length() > 0)
            ? request.getServletPath() : request.getPathInfo();

}

@Before("logAnnotation()")
public void requestLog(JoinPoint joinPoint) {
    try {
        Optional<HttpLog> httpLog = getLogAnnotation(joinPoint);
        httpLog.ifPresent(anno -> {
            HttpServletRequest request = getRequest();
            List<String> excludes = Arrays.asList(anno.exclude());
            List<Object> params = new ArrayList<>();
            StringBuilder logMsg = new StringBuilder();
            logMsg.append("REQUEST_LOG. sessionId:{}. ")
                    .append("requestUrl: ")
                    .append(getRequestPath(request))
                    .append(" -PARAMS- ");
            params.add(request.getSession().getId());
            request.getParameterMap().forEach((k, v) -> {
                if (!excludes.contains(k)) {
                    logMsg.append(k).append(": {}, ");
                    params.add(v);
                }
            });
            if (anno.headerParams().length > 0) {
                logMsg.append(" -HEADER_PARAMS- ");
                Arrays.asList(anno.headerParams()).forEach(param -> {
                    logMsg.append(param).append(": {}, ");
                    params.add(request.getHeader(param));
                });
            }
            log.info(logMsg.toString(), params.toArray());
        });
    } catch (Exception ignore) {
```

```
            log.warn("print request log fail!", ignore);
        }
    }


    @AfterReturning(returning = "restApiResponse", pointcut = "logAnnotation()")
    public void response(JoinPoint joinPoint, RestApiResponse restApiResponse) {
        try {
            Optional<HttpLog> httpLog = getLogAnnotation(joinPoint);
            httpLog.ifPresent(anno -> {
                if (!anno.ignoreResponse()) {
                    log.info("RESPONSE_LOG. sessionId:{}. result:{}", getRequest().getSession().getId
), restApiResponse);
                }
            });
        } catch (Exception ignore) {
            log.warn("print response log fail!", ignore);
        }
    }

}
```

使用

在RestController中增加自定义注解HttpLog

```
import org.springframework.web.bind.annotation.*;
import wang.crick.study.httplog.annotation.HttpLog;
import wang.crick.study.httplog.domain.User;

import java.util.Random;

@RestController
@RequestMapping("/user")
public class UserApi {


    /**
     * curl -H 'username:12b4' -H 'password:34ndd' -v 'http://localhost:8080/user/log/123?a
e=32'
     */
    @GetMapping("/log/{id}")
    @HttpLog()
    public RestApiResponse<User> getInfo(@PathVariable("id") int id,
                                         @RequestParam("age") int age){
        User user = new User();
        user.setId(id);
        user.setUsername(String.valueOf(new Random().nextLong()));
        user.setAge(age);
        return RestApiResponse.success(user);
    }

    /**
```

```
     * curl -H 'username:12b4' -H 'password:34ndd' -v 'http://localhost:8080/user/log/pwd/13?age=32'
     */
    @GetMapping("/log/pwd/{id}")
    @HttpLog(headerParams="password")
    public RestApiResponse<User> getInfoWithPwd(@PathVariable("id") int id,
                                        @RequestHeader("username") String username,
                                        @RequestHeader("password") String password){
        User user = new User();
        user.setId(id);
        user.setUsername(username);
        user.setPassword(password);
        return RestApiResponse.success(user);
    }

    /**
     * curl -H 'username:12b4' -H 'password:34ndd' -v 'http://localhost:8080/user/log/pwdExludeResponse/123?age=32'
     */
    @GetMapping("/log/pwdExcludeResponse/{id}")
    @HttpLog(headerParams="username", ignoreResponse = true)
    public RestApiResponse<User> getInfoWithPwd(@PathVariable("id") int id,
                                        @RequestParam("age") int age,
                                        @RequestHeader("password") String password){
        User user = new User();
        user.setId(id);
        user.setPassword(password);
        user.setAge(age);
        return RestApiResponse.success(user);
    }

}
```

加载HttpLogAspect对象

可以在@SpringBootApplication类下直接加在，也可以在HttpLogAspect中增加@Component注，推荐前者，更清晰。不需要增加@EnableAspectJAutoProxy类 **(注:1)**

```
@Bean
public HttpLogAspect httpLogAspect(){
    return new HttpLogAspect();
}
```

启动容器并访问

header参数可以通过curl验证。**(注:2)**

测试

可测试的代码才是好代码。所以其实我没写过几行好代码。。。

日志输出到控制台，可以通过获取控制台中内容进行contains验证

获取控制台输出字符

```
ByteArrayOutputStream outContent = new ByteArrayOutputStream();
System.setOut(new PrintStream(outContent));
```

执行request请求

基于spring test提供的mockMvc方法，测试代码如下：

```
import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.http.MediaType;
import org.springframework.test.context.junit4.SpringRunner;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.RequestBuilder;
import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
import org.springframework.test.web.servlet.result.MockMvcResultHandlers;
import org.springframework.test.web.servlet.result.MockMvcResultMatchers;
import org.springframework.test.web.servlet.setup.MockMvcBuilders;
import org.springframework.web.context.WebApplicationContext;

import java.io.ByteArrayOutputStream;
import java.io.PrintStream;
import java.util.Random;

import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;

@RunWith(SpringRunner.class)
@SpringBootTest
public class UserApiTest {

    private MockMvc mockMvc;

    @Autowired
    private UserApi userApi;

    @Autowired
    private WebApplicationContext context;

    private ByteArrayOutputStream outContent;
    private int userId = new Random().nextInt(10);
    private int age = new Random().nextInt(10);
    private long username = new Random().nextLong();
    private long password = new Random().nextLong();

    @Before
    public void setup() {
        // 坚挺控制台输出
        outContent = new ByteArrayOutputStream();
```

```
    System.setOut(new PrintStream(outContent));

    //项目拦截器有效
    mockMvc = MockMvcBuilders.webAppContextSetup(context).build();
    //单个类，拦截器无效
    // mockMvc = MockMvcBuilders.standaloneSteup(userApi).build();
}


@Test
public void test_log() throws Exception {
    String path = "/user/log/" + userId;
    String uri = path + "?age=" + age;
    RequestBuilder request = MockMvcRequestBuilders.get(uri)
            .contentType(MediaType.APPLICATION_JSON_UTF8)
            .accept(MediaType.APPLICATION_JSON);

    mockMvc.perform(request).andExpect(MockMvcResultMatchers.status().isOk());
    String console = outContent.toString();
    assertTrue(console.contains("REQUEST_LOG"));
    assertFalse(console.contains("HEADER_PARAMS"));
    assertTrue(console.contains("RESPONSE_LOG"));
    assertTrue(console.contains(path));
    assertTrue(console.contains(String.valueOf(age)));
    assertFalse(console.contains(String.valueOf(username)));
    assertFalse(console.contains(String.valueOf(password)));
}

@Test
public void test_log_header() throws Exception {
    String path = "/user/log/pwd/" + userId;
    String uri = path + "?age=" + age;
    RequestBuilder request = MockMvcRequestBuilders.get(uri)
            .header("username", username)
            .header("password", password)
            .contentType(MediaType.APPLICATION_JSON_UTF8)
            .accept(MediaType.APPLICATION_JSON);

    mockMvc.perform(request).andExpect(MockMvcResultMatchers.status().isOk());

    String console = outContent.toString();
    assertTrue(console.contains("REQUEST_LOG"));
    assertTrue(console.contains("HEADER_PARAMS"));
    assertTrue(console.contains("RESPONSE_LOG"));
    assertTrue(console.contains(path));
    assertTrue(console.contains(String.valueOf(age)));
    assertFalse(console.contains(String.valueOf(username)));
    assertTrue(console.contains(String.valueOf(password)));
}

@Test
public void test_log_header_excludeResponse() throws Exception {
    String path = "/user/log/pwdExcludeResponse/" + userId;
    String uri = path + "?age=" + age;
```

```
        RequestBuilder request = MockMvcRequestBuilders.get(uri)
                .header("username", username)
                .header("password", password)
                .contentType(MediaType.APPLICATION_JSON_UTF8)
                .accept(MediaType.APPLICATION_JSON);

        mockMvc.perform(request).andExpect(MockMvcResultMatchers.status().isOk());

        String console = outContent.toString();
        assertTrue(console.contains("REQUEST_LOG"));
        assertTrue(console.contains("HEADER_PARAMS"));
        assertFalse(console.contains("RESPONSE_LOG"));
        assertTrue(console.contains(path));
        assertTrue(console.contains(String.valueOf(age)));
        assertTrue(console.contains(String.valueOf(username)));
        assertFalse(console.contains(String.valueOf(password)));
    }

}
```

注:1 @EnableAspectJAutoProxy

注:2 curl增加header参数

```
curl -H 'username:123' -H 'password:345'
```

升级版

1. 增加了耗时统计
2. traceId用于追踪，避免参数日志重复打印
3. 增加异常捕获log

```
@Aspect
public class HttpLogAspect {

    private Logger log = LoggerFactory.getLogger(HttpLogAspect.class);

    @Pointcut("@annotation(wang.crick.study.httplog.annotation.HttpLog)")
    public void logAnnotation() {
    }

    private ThreadLocal<Long> startTimeThreadLocal = new ThreadLocal<>();
    private ThreadLocal<String> traceIdThreadLocal = new ThreadLocal<>();

    private Optional<HttpLog> getLogAnnotation(JoinPoint joinPoint) {
        if (joinPoint instanceof MethodInvocationProceedingJoinPoint) {
            Signature signature = joinPoint.getSignature();
            if (signature instanceof MethodSignature) {
                MethodSignature methodSignature = (MethodSignature) signature;
                Method method = methodSignature.getMethod();
                if (method.isAnnotationPresent(HttpLog.class)) {
```

```
                return Optional.of(method.getAnnotation(HttpLog.class));
            }
        }
    }
    return Optional.empty();
}

private HttpServletRequest getRequest() {
    RequestAttributes ra = RequestContextHolder.getRequestAttributes();
    ServletRequestAttributes sra = (ServletRequestAttributes) ra;
    return sra.getRequest();
}

private String getRequestPath(HttpServletRequest request) {
    return (null != request.getServletPath() && request.getServletPath().length() > 0)
            ? request.getServletPath() : request.getPathInfo();

}

@Before("logAnnotation()")
public void requestLog(JoinPoint joinPoint) {
    try {
        Optional<HttpLog> httpLog = getLogAnnotation(joinPoint);
        httpLog.ifPresent(anno -> {
            HttpServletRequest request = getRequest();
            traceIdThreadLocal.set(UUID.randomUUID().toString());
            startTimeThreadLocal.set(System.currentTimeMillis());
            List<String> excludes = Arrays.asList(anno.exclude());
            List<Object> params = new ArrayList<>();
            StringBuilder logMsg = new StringBuilder();
            logMsg.append("REQUEST_LOG. traceId:{}. ")
                    .append("requestUrl: ")
                    .append(getRequestPath(request))
                    .append(" -PARAMS- ");
            params.add(traceIdThreadLocal.get());
            request.getParameterMap().forEach((k, v) -> {
                if (!excludes.contains(k)) {
                    logMsg.append(k).append(": {}, ");
                    params.add(v);
                }
            });
            if (anno.headerParams().length > 0) {
                logMsg.append(" -HEADER_PARAMS- ");
                Arrays.asList(anno.headerParams()).forEach(param -> {
                    logMsg.append(param).append(": {}, ");
                    params.add(request.getHeader(param));
                });
            }
            log.info(logMsg.toString(), params.toArray());
        });
    } catch (Exception ignore) {
        log.warn("print request log fail!", ignore);
    }
}
```

```
    @AfterReturning(returning = "restApiResponse", pointcut = "logAnnotation()")
    public void response(JoinPoint joinPoint, RestApiResponse restApiResponse) {
        try {
            Optional<HttpLog> httpLog = getLogAnnotation(joinPoint);
            httpLog.ifPresent(anno -> {
                if (!anno.ignoreResponse()) {
                    log.info("RESPONSE_LOG. traceId:{}, result:{}, cost:{}",
                            traceIdThreadLocal.get(), restApiResponse, System.currentTimeMillis() - st
rtTimeThreadLocal.get());
                }
            });
        } catch (Exception ignore) {
            log.warn("print response log fail!", ignore);
        }
    }

    @AfterThrowing(throwing = "e", pointcut = "logAnnotation()")
    public void throwing(JoinPoint joinPoint, Exception e) {
        try {
            Optional<HttpLog> httpLog = getLogAnnotation(joinPoint);
            httpLog.ifPresent(anno -> {
                if (!anno.ignoreResponse()) {
                    log.info("ERROR_LOG. traceId:{}, cost:{}",
                            traceIdThreadLocal.get(), System.currentTimeMillis() - startTimeThreadLoc
l.get(), e);
                }
            });
        } catch (Exception ignore) {
            log.warn("print error log fail!", ignore);
        }
    }

}
```