

P R E S E N T A T I O N

10주차 주제 로깅

~Spring Boot에서의 REST API 로깅~

by mun

Logging ?

로그파일

운영 체제나 다른 소프트웨어가 실행 중에 발생하는 이벤트나
각기 다른 사용자의 통신 소프트웨어 간의 메시지를 기록한 파일

Logging ?

로그파일

운영 체제나 다른 소프트웨어가 실행 중에 발생하는 이벤트나
각기 다른 사용자의 통신 소프트웨어 간의 메시지를 기록한 파일

로그를 기록하는 행위 → 로깅(logging)

어디서 오류났지?

출처 졸작

```
@RequiredArgsConstructor
public class JwtTokenFilter extends OncePerRequestFilter {

    private final String key;
    private final UserServiceImpl userService;

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) throws ServletException, IOException {

        final String header = request.getHeader(HttpHeaders.AUTHORIZATION);

        System.out.println(header);

        if(header == null || !header.startsWith("Bearer")){
            log.error("Error occurs while getting header. header is null or invalid");
            filterChain.doFilter(request, response);
            return;
        }
    }
}
```

맨날 System.out.println 침

어디서 오류났지?

출처 졸작

```
@RequiredArgsConstructor
public class JwtTokenFilter extends OncePerRequestFilter {

    private final String key;
    private final UserServiceImpl userService;

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) throws ServletException, IOException {

        final String header = request.getHeader(HttpHeaders.AUTHORIZATION);

        System.out.println(header);

        if(header == null || !header.startsWith("Bearer")){
            log.error("Error occurs while getting header. header is null or invalid");
            filterChain.doFilter(request, response);
            return;
        }
    }
}
```

오류와 성능 저하의 원인을
쉽게 파악할 수 있다

매날 System.out.println 침

Filter 와 interceptor

REST API의 요청 및 응답을 가로채고 수정할 수 있는 컴포넌트
다양한 로깅 기능을 구현 가능

- 요청 및 응답 세부 정보 기록
- REST API의 실행 시간 및 성능 메트릭 기록
- 발생하는 예외 및 오류 기록
- 인증 및 권한 정보 기록
- 비즈니스 로직 및 유효성 검사 기록

등등...

구현하는 방법은
사용하는 프레임워크나
라이브러리에 따라 다르다

우린 SpringBoot로!

HandlerInterceptor

```
@Component
public class ExecutionTimeInterceptor implements HandlerInterceptor {
    private long startTime;

    @Override //handler 메소드 실행 전 호출. execution chain 을 계속 할 지 여부를 return
    public boolean preHandle(HttpServletRequest request,
                             HttpServletResponse response,
                             Object handler) throws Exception {
        startTime = System.currentTimeMillis();
        return true;
    }

    @Override //handler 메소드 실행 후 호출, 뷰 렌더링 전 호출. (
    public void postHandle(HttpServletRequest request,
                           HttpServletResponse response,
                           Object handler,
                           ModelAndView modelAndView) throws Exception {
        long endTime = System.currentTimeMillis();
        long executionTime = endTime - startTime;
    }

    @Override //뷰 렌더링 후 호출. (모든 정리 작업 수행 가능)
    public void afterCompletion(HttpServletRequest request,
```

Dispatcher Servlet 수준에서
수신 및 발신 HTTP 메시지를
가로챌 수 있는 인터페이스

```
@Configuration
public class WebConfig implements WebMvcConfigurer {
    2 usages  siwon
    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(new ExecutionTimeInterceptor())
                .addPathPatterns("/*") // 해당 경로 접근 전 Interceptor 적용
                .excludePathPatterns("/exclude"); // Intercept 제외
    }
}
```

WebConfig에 추가

HandlerInterceptor

```
@Override //handler 메소드 실행 후 호출, 뷰 렌더링 전 호출. (ModelAndView 객체 수정 가능)
public void postHandle(HttpServletRequest request,
                        HttpServletResponse response,
                        Object handler,
                        ModelAndView modelAndView) throws Exception {
    long endTime = System.currentTimeMillis();
    long executionTime = endTime - startTime;
    log.info("executionTime : {}", executionTime);
}
```

```
com.example.logging.aop.LoggingAspect : Request: Test
com.example.logging.aop.LoggingAspect : Response: log test
c.e.l.i.ExecutionTimeInterceptor     : executionTime : 44
```

The screenshot shows a REST client interface. At the top, a dropdown menu is set to 'GET' and the URL is '{{host}}/api/test'. Below this, there are tabs for 'Params', 'Auth', 'Headers (6)', 'Body', 'Pre-req.', 'Tests', and 'Se'. The 'Body' tab is selected, and it shows a table with one row: '1' in the first column and 'log test' in the second column. At the bottom, there are buttons for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'Text'.

Key	Value

Body

Key	Value
1	log test

Filter

```
@Component
public class LoggingFilter implements Filter {
    private final Logger log = LoggerFactory.getLogger(this.getClass());

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        //Initialization code
    }

    @Override
    public void doFilter(ServletRequest request,
                        ServletResponse response,
                        FilterChain chain) throws IOException, ServletException {
        var req = (HttpServletRequest) request;
        log.info("Logging request: " + req.getMethod() + " " + req.getRequestURI());
        chain.doFilter(request, response);
        log.info("Logging response: " + response.getContentType());
    }

    @Override
    public void destroy() {
        //Cleanup code
    }
}
```

서블릿 컨테이너 수준에서
수신 및 발신 HTTP 메시지를
가로챌 수 있는 컴포넌트

예제

Filter

```
@Override
public void doFilter(ServletRequest request,
                    ServletResponse response,
                    FilterChain chain) throws IOException, ServletException {
    var req = (HttpServletRequest) request;
    log.info("Logging request: " + req.getMethod() + " " + req.getRequestURI());
    chain.doFilter(request, response);
    log.info("Logging response: " + response.getContentType());
}
```

GET ⌵ | {{host}}/api/test

Params Auth Headers (6) Body Pre-req. Tests Se

Query Params

	Key	Value
--	-----	-------

Body ⌵

Pretty Raw Preview Visualize Text ⌵

1 log test

```
c.example.logging.filter.LoggingFilter : Logging request: GET /api/test
c.example.logging.filter.LoggingFilter : Logging response: text/plain;charset=UTF-8
```

AOP

```
@Aspect
@Component
public class LoggingAspect {
    private final Logger log = LoggerFactory.getLogger(this.getClass());

    @Pointcut("within(@org.springframework.web.bind.annotation.RestController *)")
    public void restController(){};

    @Around("restController()")
    public Object logRequest(ProceedingJoinPoint joinPoint) throws Throwable {
        log.info("Request: {}", joinPoint.getSignature().getName());
        Object result = joinPoint.proceed();
        log.info("Response: {}", result.toString());
        return result;
    }
}
```

코드를 수정하지 않고도
횡단 관심사(예: 로깅)를
코드에 주입할 수 있는 기술

예제

AOP

1개 사용 위치 siwon

```
@Pointcut("within(@org.springframework.web.bind.annotation.RestController *)")
public void restController();
```

siwon

```
@Around("restController()")
public Object logRequest(ProceedingJoinPoint joinPoint) throws Throwable {
    log.info("Request: {}", joinPoint.getSignature().getName());
    Object result = joinPoint.proceed();
    log.info("Response: {}", result.toString());
    return result;
}
```

GET {{host}}/api/test

Params Auth Headers (6) Body Pre-req. Tests Se

Query Params

	Key	Value
--	-----	-------

Body

Pretty Raw Preview Visualize Text

1 log test

```
com.example.logging.aop.LoggingAspect : Request: Test
com.example.logging.aop.LoggingAspect : Response: log test
```

```
public String Test(){
    return "log test";
}
```

ControllerAdvice

REST API에서 전역 예외 및 오류를
처리할 수 있는 어노테이션

```
@ControllerAdvice
public class RequestLoggingInterceptor {
    private final Logger log = LoggerFactory.getLogger(this.getClass());

    @Autowired
    private HttpServletRequest request;

    @Autowired
    private HttpServletResponse response;

    @Around("@annotation(org.springframework.web.bind.annotation.RequestMapping)")
    public Object logRequest(ProceedingJoinPoint joinPoint) throws Throwable {
        log.info("Request: {} {}", request.getMethod(), request.getRequestURI());
        Object result = joinPoint.proceed();
        log.info("Response: {}", response.getStatus());
        return result;
    }
}
```

예제

ControllerAdvice

우리 졸업작품 코드^^

```
@S1f4j
@RestControllerAdvice
public class GlobalControllerAdvice {

    @ExceptionHandler(TluApplicationException.class)
    public ResponseEntity<?> applicationHandler(TluApplicationException e){
        log.error("Error occurs {}", e.toString());
        return ResponseEntity.status(e.getErrorCode().getStatus())
            .body(Response.error(e.getErrorCode().name()));
    }

    @ExceptionHandler(RuntimeException.class)
    public ResponseEntity<?> applicationHandler(RuntimeException e){
        log.error("Error occurs {}", e.toString());
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
            .body(Response.error(ErrorCode.INTERNAL_SERVER_ERROR.name()));
    }
}
```

RestControllerAdvice

작동 방식을 이해하기 위해

로깅은 필수!

~로그 찍읍시다~

감사합니다



참고 : <https://romanglushach.medium.com/java-rest-api-logging-best-practices-and-guidelines-bf5982ee4180>