로깅

Tags	
스터디 일자	@2023/12/21

로깅이 머에요?

서비스를 배포 및 운영하고 있는 상황이라면 운영상에서의 **예외상황**, **에러**를 확인하고 싶은 상황이 자주 있습니다!

콘솔에 `print`해 확인하는 방법도 있지만, 지속적으로 관리하고 운영하기에 콘솔은 *불편한 부분이 많답니다..* 그렇기 때문에 이를 **Log 형태** + **파일 형태**로 저장해 관리하는 것이 일반적이라고 하네요!!

이것이 로깅..

- 시스템 운영에 대한 기록
- 디버깅, 시스템 에러 추적, 성능, 문제점 향상 등의 목적으로 사용
- 어느 정도까지 로그를 남길 것인가?
  - 너무 적은 로그? → 정확한 시스템의 상황을 파악하기 어려움
  - 너무 많은 로그? → 빈번한 file I/O의 오버헤드와 로그 파일의 백업 문제 등 파생 문제 발생 가능성

자바에서 로깅은?

자바에서는 다음과 같은 로깅 관련 라이브러리들을 제공하고 있어요!

- java.util.logging.Logger
- Log4J, log4j2
- Logback

- SLF4J → 추상체

스프링부트에서는 기본적으로 Logback(구현체)과 SLF4J(추상체)를 채택하고 있기 때문에 요걸로 로깅을 해봅시다!

(그래서 spring-boot-starter-web 안에 Logback이 포함돼 있음 → 따로 의존성 추가 안해도됨 👍)

간단한 로깅 미리보기

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class LogTestController {

        // Lombok의 @Slf4j 어노테이션을 사용하면 이 코드 생략 가능
    private final Logger log = LoggerFactory.getLogger(getClass

    @RequestMapping("log-test")
    public String logTest() {

        String name = "Spring";

        System.out.println("name = " + name);

        log.info(" info log={}", name);

        return "ok";
    }
}
```

> **호출 결과**
>
> ```
> name = Spring
> 2023-12-19T17:18:58.084+09:00  INFO 6944 --- [nio-8080-exec-2] hello.hellospring.LogTestController      :  info log=Spring
> ```

보이시나요?

```
name = Spring
```

단순히 `System.out.println` 으로 출력한 결과와

```
2023-12-19T17:18:58.084+09:00  INFO 6944 --- [nio-8080-exec-2] hello.hellospring.LogTestController      :  info log=Spring
```

로깅을 한 결과입니다!

차이가 느껴지시죠 ~v~

로그 레벨

(회사 내규에 따라 다를 수도 있다고 함)

SLF4J의 로그 레벨은 다음과 같습니다!

- **Trace**
  - 가장 상세한 로그 레벨로, 코드의 흐름을 따라가며 디버깅 정보를 기록한다.
- **Debug**
  - 디버깅을 위한 로그 레벨로, 프로그램의 상태 및 실행 중에 발생하는 중요한 이벤트를 기록한다.
- **Info**
  - 일반적인 정보를 기록하는 로그 레벨로, 프로그램의 주요 이벤트 및 상태 변경을 기록한다.
  - **에러는 아니지만 주시해야할 것**

- **Warning**
  - 예외적인 상황을 기록하는 로그 레벨로, 잠재적인 문제 또는 예상치 못한 동작을 알린다.
  - **예외 상황이긴 했지만 에러는 아닌 것**
- **Error**
  - 심각한 에러를 기록하는 로그 레벨로, 예외 상황 또는 잘못된 동작을 나타낸다.
  - **에러가 맞고 대응 해야할 것**

- **로그 레벨을 활용하면?**

```
private final Logger log = LoggerFactory.getLogger(getClass());

log.trace("trace 로깅이야!!!");
log.debug("debug 로깅이야!!!");
log.info("info 로깅이야!!!");
log.warn("warn 로깅이야!!!");
log.error("error 로깅이야!!!");
```

> ## 실행 결과
>
> ```
> 2023-12-19T18:21:48.324+09:00 TRACE 5528 --- [nio-8080-exec-1] hello.hellospring.LogTestController      : trace 로깅이야!!!
> 2023-12-19T18:21:48.324+09:00 DEBUG 5528 --- [nio-8080-exec-1] hello.hellospring.LogTestController      : debug 로깅이야!!!
> 2023-12-19T18:21:48.324+09:00  INFO 5528 --- [nio-8080-exec-1] hello.hellospring.LogTestController      : info 로깅이야!!!
> 2023-12-19T18:21:48.325+09:00  WARN 5528 --- [nio-8080-exec-1] hello.hellospring.LogTestController      : warn 로깅이야!!!
> 2023-12-19T18:21:48.326+09:00 ERROR 5528 --- [nio-8080-exec-1] hello.hellospring.LogTestController      : error 로깅이야!!!
> ```

▼ 로그를 찍을 때는 문자열 연산을 사용하면 안돼요!

로그 특성 상 특정 레벨의 로그만 출력하는 상황이 많은데,

`log.trace("trace log = " + name)` 이런식으로 사용하면 trace 레벨이 출력되지 않는 상황에서도 `+` 연산을 수행하기 때문에 심한 자원 낭비가 된담니다...!

똑같은 결과이지만 `log.trace("trace log = {}", name)` 이렇게 사용하면 연산을 수행하지 않기 때문에 자원 낭비가 없습니다 ^~^

- **로그 레벨 제한하기?**
  - 설정 파일 작성! `resources/logback.xml`

```
<configuration>
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleA
        <encoder>
                        <!--로그 출력 형식을 지정할 수 있다.-->
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logge
        </encoder>
    </appender>
    <!--에러 이상 레벨만 출력-->
    <root level="error">
        <appender-ref ref="STDOUT" />
    </root>
</configuration>
```

▼ **코드 설명**

%d{HH:mm:ss.SSS} : 로그가 출력되는 시간이 출력된다. 중괄호{ } 안은 이 시간의 포맷이다.

%thread : 실행 쓰레드 이름이다.

%-5level : 로그 레벨을 5의 고정폭 값으로 출력하라는 것을 의미한다.

%logger : 패키지를 포함 클래스 정보다.

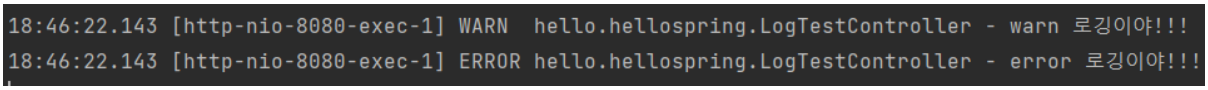
%msg : %message의 약자로, 사용자가 출력한 메시지가 출력된다.

%n : 줄바꿈이다.

<root level=""> : 설정한 레벨 이상의 로그만 출력된다.

> ## 실행 결과

지정한 패턴으로, **에러 로깅만** 찍히는 것을 확인할 수 있습니다~!
만약
<root level="warn"> 으로 설정했다면? ~~(warn, error만 나오겠죠?)~~



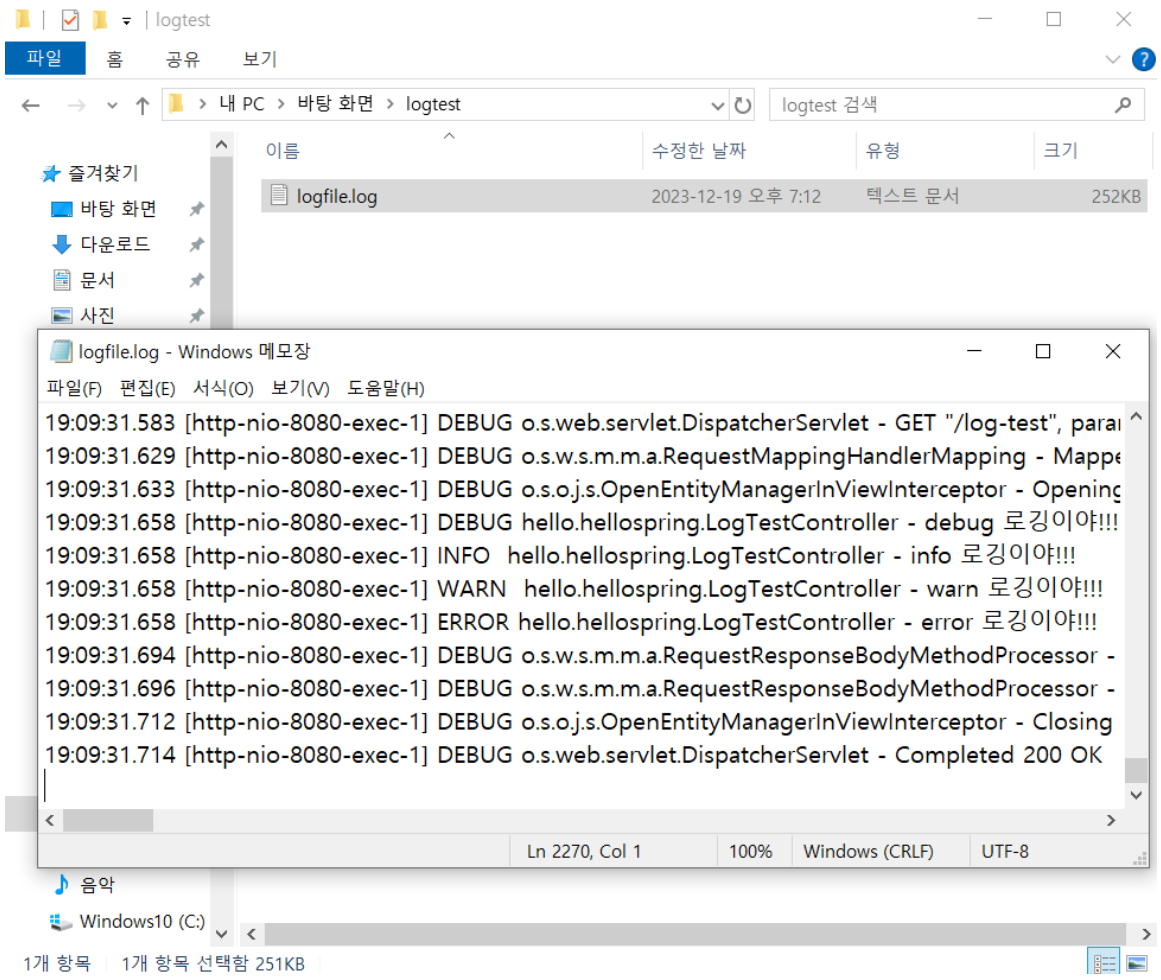
로그를 파일로 저장하고 싶다면?

- `resources/logback.xml` 에서 **설정만 추가**해주면 됩니다!

```
<configuration>
<appender name="FILE" class="ch.qos.logback.core.FileAppender">
    <file>C:\Users\chaen\Desktop\logtest/logfile.log</file>
    <encoder>
        <!--로그 출력 형식을 지정할 수 있다.-->
        <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36}
    </encoder>
</appender>
    <!--디버그 이상 레벨만 출력-->
    <root level="debug">
```

```
        <appender-ref ref="FILE" />
    </root>
</configuration>
```

실행 결과



지정한 경로에 로그가 **차곡차곡 저장된다!**

🥳 로그를 파일로 저장할 때 다양한 옵션들을 줘서
몇분마다 로그를 다른 파일로 저장한다던지, 파일 최대 개수를 지정해서 초과하면 이전 로그 파일은 삭제하는 등
다양한 커스텀이 가능하다고 합니다~~!!
잘 알아봐서 실무에서 쓰면 유용하겠죠 ?

참고한 사이트

https://livenow14.tistory.com/63

https://ksabs.tistory.com/105

https://colinch4.github.io/2023-11-20/12-08-24-233175-slf4j를-사용하여-로깅-메시지를-파일로-저장하는-방법은-무엇인가요/