Logging

원문 : https://docs.spring.io/spring-framework/docs/5.0.0.RC3/spring-framework-reference/overview.html#overview-logging

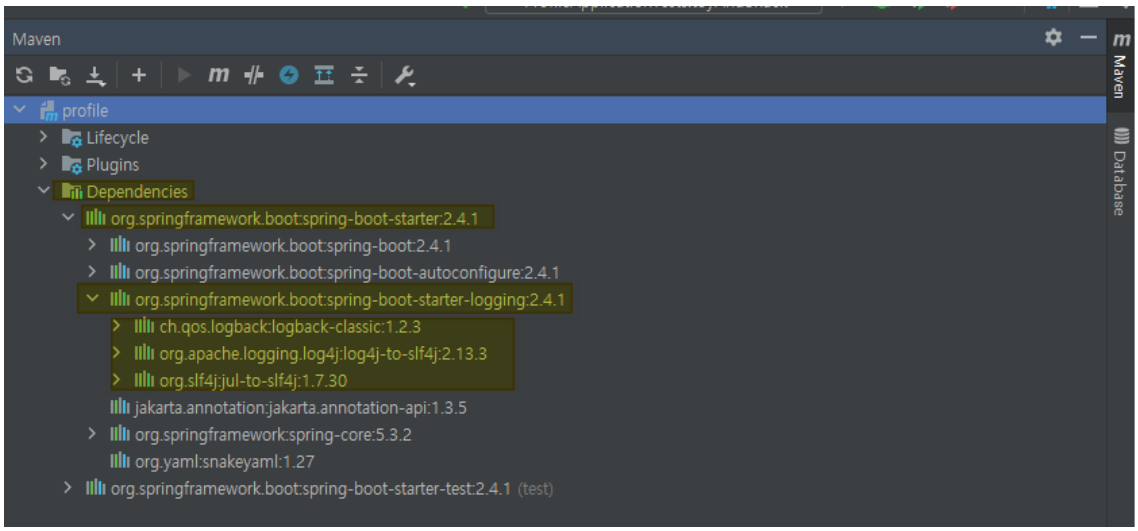
로깅의 예

스프링 부트의 내장 로깅 라이브러리

- 로깅 facade : `Commons Logging`, `SFL4j`
- 로깅 facade 의 구현체 : `java.util.logger (JUL)`, `Log4J2`, `Logback`

> Logging Facade : 로깅 인터페이스 역할
>
> Logging Facade 를 사용하는 이유 : Facade 구현체(JUL, Log4j, Logback) 의존성을 자유롭게 갈아 끼울 수 있도록

spring-boot-starter 의 내장 의존성



스프링 부트는 로그를 남길때 `Commons Logging` 파사드 -> `SFL4J` 파사드 -> `Logback` 구현체 순으로 실행된다.

결국 로그를 찍는 실체는 `Logback`이다.

로그 출력하기

MyRunner.class

```
package com.megait.logging;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.ApplicationArguments;
import org.springframework.boot.ApplicationRunner;
import org.springframework.stereotype.Component;

@Component
public class MyRunner implements ApplicationRunner {

    private Logger logger = LoggerFactory.getLogger(getClass());

    @Override
    public void run(ApplicationArguments args) throws Exception {

        logger.info("MyRunner.run() 시작합니당");
        // logger.debug("MyRunner.run() 시작합니당");
        // logger.warn("MyRunner.run() 시작합니당");
        // logger.error("MyRunner.run() 시작합니당");
        // logger.trace("MyRunner.run() 시작합니당");


    }
}
```

Service, Controller 같은 부분에도 로그를 남겨야 한다.

결과

```
2021-01-03 17:15:03.284  INFO 15300 --- [           main] com.megait.logging.LoggingApplication    : Starting LoggingApplication on DE
2021-01-03 17:15:03.287 DEBUG 15300 --- [           main] com.megait.logging.LoggingApplication    : Running with Spring Boot v2.3.7.R
2021-01-03 17:15:03.287  INFO 15300 --- [           main] com.megait.logging.LoggingApplication    : No active profile set, falling ba
2021-01-03 17:15:03.920  INFO 15300 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s):
2021-01-03 17:15:03.927  INFO 15300 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2021-01-03 17:15:03.927  INFO 15300 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache
2021-01-03 17:15:03.996  INFO 15300 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebA
2021-01-03 17:15:03.996  INFO 15300 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initi
2021-01-03 17:15:04.111  INFO 15300 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'app
2021-01-03 17:15:04.222  INFO 15300 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (
2021-01-03 17:15:04.230  INFO 15300 --- [           main] com.megait.logging.LoggingApplication    : Started LoggingApplication in 1.2
2021-01-03 17:15:04.231  INFO 15300 --- [           main] com.megait.logging.MyRunner              : MyRunner.run() 시작합니당
```

Logging level의 포함관계

error < warn < info < debug < trace

로깅 포맷

```
2019-03-05 10:57:51.112  INFO 45469 --- [           main]
org.apache.catalina.core.StandardEngine  : Starting Servlet Engine: Apache
Tomcat/7.0.52
2019-03-05 10:57:51.253  INFO 45469 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].
[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2019-03-05 10:57:51.253  INFO 45469 --- [ost-startStop-1]
o.s.web.context.ContextLoader            : Root WebApplicationContext:
initialization completed in 1358 ms
2019-03-05 10:57:51.698  INFO 45469 --- [ost-startStop-1]
o.s.b.c.e.ServletRegistrationBean        : Mapping servlet: 'dispatcherServlet'
to [/]
2019-03-05 10:57:51.702  INFO 45469 --- [ost-startStop-1]
o.s.b.c.embedded.FilterRegistrationBean  : Mapping filter:
'hiddenHttpMethodFilter' to: [/*]
```

The following items are output:

- Date and Time: Millisecond precision and easily sortable.
- Log Level: `ERROR`, `WARN`, `INFO`, `DEBUG`, or `TRACE`.
- Process ID.
- A `---` separator to distinguish the start of actual log messages.
- Thread name: Enclosed in square brackets (may be truncated for console output).
- Logger name: This is usually the source class name (often abbreviated).
- The log message.

특정 로그 레벨만 출력하기

방법1. cmd 아규먼트 사용

```
$ java -jar myapp.jar --debug
```

info, trace 등의 다른 레벨을 출력하고 싶다면 `--info`, `--trace` 등의 아규먼트 사용

> trace 레벨이 가장 자세한 로그레벨이다.

방법2. application.properties 사용

```
debug=true
```

info, trace 등의 다른 레벨을 출력하고 싶다면 `info=true`, `trace=true` 등의 프로퍼티 사용

잊지 말자. 우리는 이 외에도 더 많은 프로퍼티 설정 방법을 알고 있다. (`외부 설정` 챕터 참고)

주의 ! `debug` 모드는 모든 debug 레벨을 출력하는 것은 아니다. 코어 로거(Spring boot, Hibernate 등에 내장된 로거)의 debug 레벨만 출력된다.

우리가 만든 debug 레벨 로그는 출력되지 않는다는 뜻이다.

로그레벨을 패키지별로 따로 설정하기

application.properties

```
logging.level.root=info
logging.level.com.megait.logging=debug
```

`logging.level.root` : 모든 패키지에 대한 로그 레벨

`logging.level.패키지` : 해당 패키지에 대한 로그 레벨. 내가 만든 패키지가 아닌 기본 제공 패키지도 가능.

파일에 로깅 저장하기

`logging.file.name`	`logging.file.path`	Example	Description
(none)	*(none)*		Console only logging.
Specific file	*(none)*	`my.log`	Writes to the specified log file. Names can be an exact location or relative to the current directory.
(none)	Specific directory	`/var/log`	Writes `spring.log` to the specified directory. Names can be an exact location or relative to the current directory.

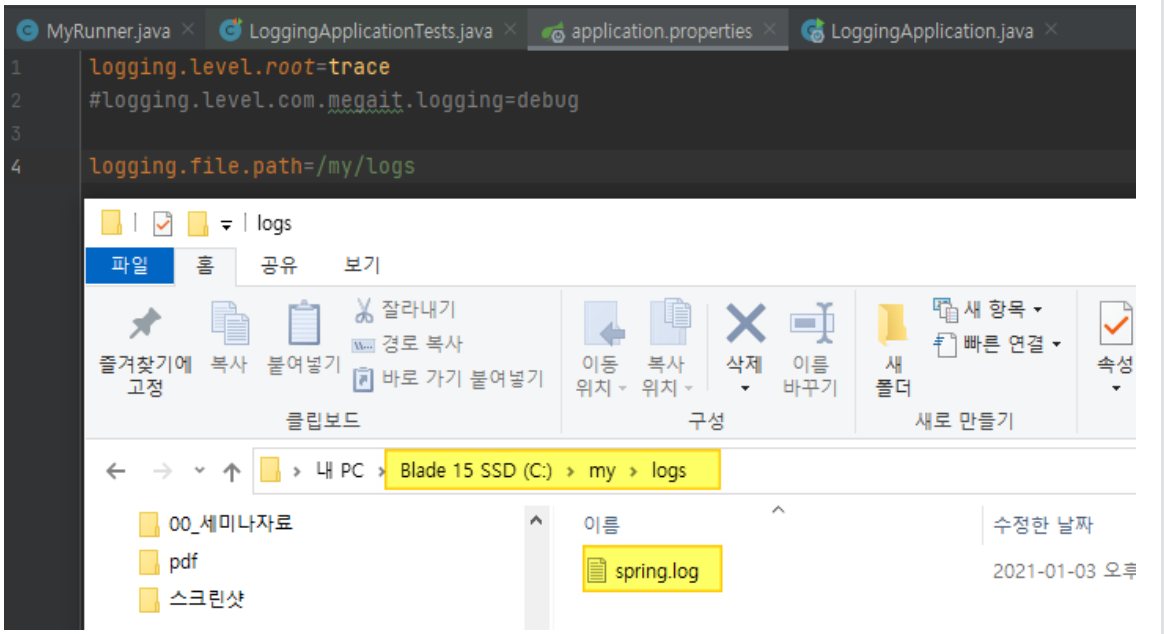
파일은 최대 10mb 까지 저장가능하며 이후는 rotation이 실행된다.

> rotation 설정: https://docs.spring.io/spring-boot/docs/current/reference/html/spring-boot-features.html#boot-features-logging-file-rotation

application.properties

```
logging.file.path=/logs  # 디렉토리만 지정하고 싶을 때. 절대 경로, 상대 경로 상관 없음. 기본 파일명 'spring.log' 가 지정됨.
logging.file.name=/logs/mylog_files # 파일 이름까지 지정하고 싶을 때. 절대 경로, 상대 경로 상관 없음.
```

결과



로그 커스터마이징하기

https://docs.spring.io/spring-boot/docs/current/reference/html/howto.html#howto-logging