로깅 라이브러리(Logging Library)

- 로깅(logging)이란 시스템 동작시 시스템의 상태와 작동 정보를 시간의 경과에 따라 기록하는 것
- System.out.println()을 사용하지 말고 log.XXX() 이용하면 좋은 점

1. 로그 출력위치(클래스, 메서드, 라인번호)를 알수 있고
2. 출력 형식을 지정할 수 있으며
3. 성능면에서 훨씬 뛰어남.

로깅 라이브러리

- 등장 순서

```
java.util.logging  ==>  Log4j  ==>  Logback  ==>  Log4j2
                        spring      springboot       최신
```

로그 설정 위치

Spring이나 일반 java 프로그램의 경우 logback.xml 파일을 resources 디렉터리에 만들어서 참조하지만 Spring Boot의 경우에는 아래 3가지 중 한 가지 방법을 선택합니다.

- application.properties에 설정

```
logging.level.root=INFO
logging.level.org.springframework=DEBUG
logging.level.com.example.demo=DEBUG
logging.pattern.console=[%d{HH:mm:ss}] %-5level %logger{36} - %msg%n
```

- resources/logback-spring.xml에 설정
- resources/logback.xml에 설정

```
<configuration>

  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <!-- encoders are assigned the type
         ch.qos.logback.classic.encoder.PatternLayoutEncoder by default -->
    <encoder>
      <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} -%kvp-
%msg%n</pattern>
    </encoder>
  </appender>

  <root level="debug">
    <appender-ref ref="STDOUT" />
  </root>
</configuration>
```

log에 대한 세부적인 설정을 하려면 logback.xml 파일을 통한 설정이 필요하고, 간단하게 콘솔 로그만 확인하기 위해서는 application.properties (또는 .yml) 파일에 로그 레벨을 설정하는 것만으로 로그를 확인할 수 있습니다.

- logback : https://logback.qos.ch/manual/configuration.html

로그레벨

```
TRACE > DEBUG > INFO > WARN > ERROR > FATAL > OFF
```

- trace : 아주 자세한 로그 출력
- debug : DEV 환경에서 사용
- fatal : 심각한 오류 로그 출력
- off : 로그 출력 안함

로깅

💡 Spring Boot Logging

- Spring Boot에는 내장으로 'jul(java.util.logging)' 로깅 프레임워크가 내장되어 있어서 이를 즉시 사용할 수 있습니다.

```
Logger log = LoggerFactory.getLogger(SampleController.class);
```

💡 Spring Boot Lombok

- Spring Boot Lombok 내에 Slf4j가 포함되어 있어서 기존의 Logger를 선언해서 사용하는 방식외에 `@Slf4J` `@log4j2` Annotaion을 이용하여서 간편하게 사용이 가능합니다.

logback을 log4j2로 변경

spring boot에서 log4j2를 사용하기 위해서는 dependency에서 logback을 제거해주는 작업이 필요합니다. 스프링 부트는 기본 설정으로 logback을 사용하기 때문에 `@log4j2` 의존성을 추가하더라도 기본 설정으로 잡힌 logback이 동작하게 됩니다.

스프링 로깅 동작과정

spring framework(Log4j)

- Log4j 구현체 사용 JCL(Jakarta Common Logging 인터페이스) ==> Log4j(JCL 구현체)
- log4j2 구현체 사용 JCL(Jakarta Common Logging 인터페이스) ==> jcl-over-slf4j(어댑터) ==> SLF4J(인터페이스) ==> slf4j-log4j(브릿지) ==> log4j2 (slf4j 구현체)

spring boot (Logback)

-> log4j-to-slf4j(어댑터) -> SLF4J(인터페이스) -> Logback(slf4j 구현체)

spring-boot starter-logging logback-classic log4j-to-slf4j jul-to-slf4j(브릿지)

slf4j (Simple Logging Facade For Java)

slf4j는 자바 로깅 시스템을 쉽게 사용할 수 있도록 해주는 라이브러리이며, 다양한 자바 로깅 시스템을 사용할 수 있도록 파사드 패턴(facade pattern) 및 추상화를 통해 로깅 기능을 제공합니다. slf4j를 사용함으로서 얻을 수 있는 이점은 기존에 사용하는 로깅 시스템을 교체하고 싶을 때, 소스 코드를 수정하지 않고도 maven이나 gradle의 의존성 설정만 바꾸면 손쉽게 적용할 수 있습니다.

파사드(Facade pattern)

복잡한 서브 시스템을 단순한 인터페이스를 제공하여 제어하는 방식. 클라이언트는 복잡한 서브 시스템에 대해 알 필요가 없으며, Facade를 통해 일괄적으로 제어가능하다. 단순한고 일관된 통합 인터페이스르 제공