Chapter 6-2 Logging

```
@GetMapping("test-log")
public void testLog(){
  logger.trace("TRACE Log Message");
  logger.debug("DEBUG Log Message");
  logger.info("INFO Log Message");
  logger.warn("WARN Log Message");
  logger.error("ERROR Log Message");
}
```

logger

애플리케이션이 진행되는데 있어서 무슨 일이 일어났는지를 보는

개발적인 측면에서 필요한 메시지를 남김

서비스를 어떻게 사용했는지

이력을 남기로 log를 남김

프로그램의 진행을 확인하기 위해 사용

영문으로 남기는 것이 좋음

의미 없는 메시지를 남기는 것은 X

어떤일이 있었는지에 대한 메시지를 남기기

trace 가장 사소한 메시지

error로 갈수록 좀 더 위험한 수준의 메시지

info 진행상황에 대한 메시지

warn 미래에 문제가 생길수도 있다는 수준의 메시지

error 발생함으로서 서비스가 정상적으로 진행되지 않았다.

debug ex) 어떤 행위가 이루어졌다.

trace ex) 변수에 값을 할당했다. 상용서비스에서는 메시지를 제거해주는 것이 좋음 (git에 올리지 않는 ..)

```java
@RequestMapping("/")
public String index() {
    logger.trace("A TRACE Message");
    logger.debug("A DEBUG Message");
    logger.info("An INFO Message");
    logger.warn("A WARN Message");
    logger.error("An ERROR Message");

    return "index";
}
```

Log Level: Debug

Debug 이상의 Log만 출력된다.

어플리케이션에 설정된 레벨보다 중요한 메시지만 출력된다.

```
java -jar spring-boot.jar --trace
java -jar spring-boot.jar --debug
```

trace, debug: 실행인자로 직접 전달

```
logging.level.root=warn
```

```
logging:
  level:
    root: warn
```

logging.level.root

어플리케이션 전체 기본 로그 레벨 설정

```yaml
logging:
  level:
    root: warn
    org.springframework: info
```

개별 패키지별 레벨 설정 가능

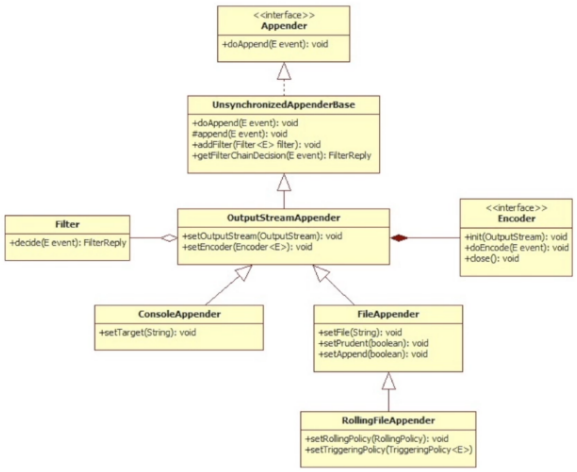
```
java -Dlogging.level.org.hibernate=info -jar spring-boot.jar
```

JVM 실행시 인자 전달 가능

Logback

Spring Boot에서 기본으로 사용하는 Logging Framework

상세한 logging 설정 → logback 라이브러리 사용



- Appender라는 Interface를 통해 logger의 동작을 정의한다.
  - 출력 위치, 파일 위치
  - 파일 생성 주기 등
- XML을 통해 log level 및 appender 등을 정의할 수 있다.

Appender을 통해 어떤 식으로 출력하고 저장할지 설정할 수 있는 인터페이스의 일부분

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<property name="LOGS" value="./logs" />
<appender name="Console"
          class="ch.qos.logback.core.ConsoleAppender">
    <layout class="ch.qos.logback.classic.PatternLayout">
        <Pattern>
            %black(%d{ISO8601}) %highlight(%-5level) [%blue(%t)] %yellow(%C{0}): %msg%
n%throwable
        </Pattern>
    </layout>
</appender>

<appender name="RollingFile"
          class="ch.qos.logback.core.rolling.RollingFileAppender">
    <file>${LOGS}/spring-boot-logger.log</file>
    <encoder
            class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
        <Pattern>%d %p %C{0} [%t] %m%n</Pattern>
    </encoder>

    <rollingPolicy
            class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
        <fileNamePattern>${LOGS}/archived/spring-boot-logger-%d{yyyy-MM-dd-HH-mm}.%i.l
og
        </fileNamePattern>
        <maxFileSize>100MB</maxFileSize>
        <maxHistory>5</maxHistory>
        <totalSizeCap>20GB</totalSizeCap>
    </rollingPolicy>
</appender>

<root level="warn">
```

```
    <appender-ref ref="RollingFile" />
    <appender-ref ref="Console" />
</root>

<logger name="dev.aquashdw" level="trace" additivity="false">
    <appender-ref ref="RollingFile" />
    <appender-ref ref="Console" />
</logger>
</configuration>
```

<property name="LOGS" value="./logs" />
: xml에서 사용하기 위한 특정 값을 선언하는 것

`rollingPolicy`

: 특정한 제약사항을 두고 그 만큼의 log 파일만 남아 있을 수 있도록 ~