12_Logback

로그(log)란?

- 사전적 의미 : 통나무, 향해 일지, 배의 속력이나 항해한 거리를 계측하는 장치의 총칭.
- 실질적 의미 : 기록을 남기는 것.

로그 사용 이유

- 애플리케이션 운영 시 로그의 효율적인 관리가 가능하다.(콘솔 또는 특정파일)
- 콘솔 로그를 위해 System.out.print를 사용하는 건 성능저하를 야기함.

로그의 장점 VS 단점

- 장점
    - 프로그램의 문제 파악에 용이
    - 빠르고 효율적인 디버깅 가능
    - 수행내역 파악이 쉬움
    - 로그 이력을 파일, DB 등으로 남길 수 있음

- 단점
    - 로그에 대한 디바이스(파일) 입출력으로 인해 런타임 오버헤드 발생
    - 로깅을 위한 추가 코드로 인해 전체 코드 사이즈 증가
    - 심하게 생성되는 로그는 혼란을 야기하거나 어플리케이션 성능에 영향을 미침
    - 개발 중간에 로깅 코드를 추가하기 어려움

Logback

- Java용으로 구현된 강력한 **Logging Framework**
- log4j의 후속 프레임워크
- https://logback.qos.ch/manual/introduction.html

logging level 지원

1. **TRACE:** 추적 정보 제공
2. **DEBUG:** 디버깅
3. **INFO:** 수행된 기능, 관련 정보 제공

4. **WARN:** 경고 사항 표시
5. **ERROR:** 심각한 오류

- 로그 출력 시 로깅 레벨을 하나 지정, 선택된 레벨 이상 레벨의 내용이 모두 출력 됨
  ex) INFO 레벨 선택 시 INFO, WARN, ERROR 레벨도 출력 될 수 있음
  TRACE, DEBUG는 출력 되지 않음

Appender

- 로그를 출력하는 객체
- **ConsoleAppender:** 콘솔에 로그 출력
- **FileAppender:** 파일에 로그 출력
- **SmtpAppender:** 이메일로 로그 전송
- **JDBCAppender:** 데이터베이스에 로그 저장
- **RollingFileAppender:** 파일에 로그 출력, 일정 크기 이상 저장되면 새 파일을 생성

SLF4J (Simple Logging Facade for Java)

- Java에서 로그를 간단히 사용할 수 있게 하기 위한 Logging API
- 추상화 계층을 제공하여 다양한 Logging Framework 코드를 작성할 수 있음

Spring Boot에서 logback 사용하기

1. src/main/resources 폴더에 logback-spring.xml 생성
2. 아래 내용 붙여넣기

```
<configuration scan="true" scanPeriod="60 seconds">
    <!-- 1분(60초) 마다 설정 변경 사항을 파악한 후 갱신-->

    <!-- config.properties에 아래 내용 추가

        log.config.path(로그 파일 저장 경로)
        log.config.fileName(로그 파일 이름)
    -->

    <!-- 설정 값을 읽어들일 외부 파일 지정-->
    <property resource="config.properties" />

    <!-- logback-spring.xml에서 사용할 변수 선언 -->
    <property name="logPath"        value="${log.config.path}" /> <!-- config.properties에서 얻어온 값-->
    <property name="fileName"       value="${log.config.fileName}" />  <!-- config.properties에서 얻어온
값-->
    <property name="maxHistory"     value="30" />
    <property name="maxFileSize"    value="10MB" />


    <!--============================ 스프링 부트 기본 로그 설정 ============================-->
    <conversionRule conversionWord="clr" converterClass="org.springframework.boot.logging.logback.ColorC
onverter" />
    <conversionRule conversionWord="wex" converterClass="org.springframework.boot.logging.logback.Whites
paceThrowableProxyConverter" />
    <conversionRule conversionWord="wEx" converterClass="org.springframework.boot.logging.logback.Extend
edWhitespaceThrowableProxyConverter" />
```

```
    <property name="CONSOLE_LOG_PATTERN" value="${CONSOLE_LOG_PATTERN:-%clr(%d{${LOG_DATEFORMAT_PATTERN:-yyyy-MM-dd HH:mm:ss.SSS}}){green} %clr(${LOG_LEVEL_PATTERN:-%5p}) %clr(${PID:- }){magenta} %clr(---){faint} %clr([%15.15t]){faint} %clr(%-40.40logger{39}){cyan} %clr(:){faint} %m%n${LOG_EXCEPTION_CONVERSION_WORD:-%wEx}}"/>
    <property name="CONSOLE_LOG_CHARSET" value="${CONSOLE_LOG_CHARSET:-default}"/>

    <property name="FILE_LOG_PATTERN" value="${FILE_LOG_PATTERN:-%d{${LOG_DATEFORMAT_PATTERN:-yyyy-MM-dd HH:mm:ss.SSS}} ${LOG_LEVEL_PATTERN:-%5p} ${PID:- } --- [%t] %-40.40logger{39} : %m%n${LOG_EXCEPTION_CONVERSION_WORD:-%wEx}}"/>
    <property name="FILE_LOG_CHARSET" value="${FILE_LOG_CHARSET:-default}"/>

    <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>${CONSOLE_LOG_PATTERN}</pattern>
            <charset>${CONSOLE_LOG_CHARSET}</charset>
        </encoder>
    </appender>

    <include resource="org/springframework/boot/logging/logback/file-appender.xml" />
    <logger name="org.apache.catalina.startup.DigesterFactory" level="ERROR"/>
    <logger name="org.apache.catalina.util.LifecycleBase" level="ERROR"/>
    <logger name="org.apache.coyote.http11.Http11NioProtocol" level="WARN"/>
    <logger name="org.apache.sshd.common.util.SecurityUtils" level="WARN"/>
    <logger name="org.apache.tomcat.util.net.NioSelectorPool" level="WARN"/>
    <logger name="org.eclipse.jetty.util.component.AbstractLifeCycle" level="ERROR"/>
    <logger name="org.hibernate.validator.internal.util.Version" level="WARN"/>
    <logger name="org.springframework.boot.actuate.endpoint.jmx" level="WARN"/>
    <!--============================== 스프링 부트 기본 로그 설정 ==============================-->



    <!-- 로그를 파일로 저장 -->
    <appender name="ROLLING" class="ch.qos.logback.core.rolling.RollingFileAppender">

        <!-- Rolling 정책 -->
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

            <!-- 로그 파일이 저장될 위치, 파일명 패턴 지정-->
            <fileNamePattern>${logPath}${fileName}.%d{yyyy-MM-dd}_%i.log</fileNamePattern>

            <!-- 로그파일 최대 보관일 -->
            <maxHistory>${maxHistory}</maxHistory>

            <!-- 로그 파일당 최고 용량 kb, mb, gb -->
            <timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
                <maxFileSize>${maxFileSize}</maxFileSize>
            </timeBasedFileNamingAndTriggeringPolicy>
        </rollingPolicy>

        <!-- 출력되는 로그의 패턴, 인코딩-->
        <encoder>
            <pattern>${FILE_LOG_PATTERN}</pattern>
            <charset>${FILE_LOG_CHARSET}</charset>
        </encoder>

    </appender>


    <!--
        * 로그 레벨 : TRACE < DEBUIG < INFO < WARN < ERROR
        (로그 레벨 지정 시 지정된 레벨 + 상위 레벨만 출력됨)
    -->
```

```
    <!-- 기본 로그 출력 레벨, 로그 작성 객체(appender) 지정-->
    <root level="INFO">
        <appender-ref ref="CONSOLE" />
        <appender-ref ref="ROLLING" />
    </root>

    <!-- 별로 로그 출력 설정 시 작성하는 태그 -->
    <!--
    <logger name="edu.kh.project" level="DEBUG">
        <appender-ref ref="ROLLING" />
    </logger>
    -->

</configuration>
```