Logging

*** 로그(log)란?**

- 사전적 의미 : 통나무, 향해 일지, 배의 속력이나 항주한 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

*** Logging framework (log4j / logback / slf4j)**

- 스프링 프로젝트에는 기본적으로 **log4j 라이브러리**가 추가되어있고
  log4j와 관련된 설정을 담는 **log4j.xml** 파일을 서버 구동과 동시에 로딩하게 되어 있음.

- 하지만 log4j를 이용하여 많은양의 로그를 출력하는 경우 성능 저하가 심해
  최근에는 **logback** 이라는 라이브러리를 사용함**.(10배 빠름)**

- 스프링 프로젝트에서 기존 사용하던 log4j를 새로운 logback로 변경을 해야 하는데
  이를 일일이 변경하기에는 비효율적이므로
  log4j <-> logback 서로 변환을 해줄 수 있는 **slf4j 라이브러리가** 존재
  (STS 사용 시 pom.xml에 자동으로 slf4h 라이브러리가 추가되어 있음
  ➔ pom.xml에 <!—Logging → 부분)

- **slf4j(Simple logging Facade for Java)** 란 Facade(추상체)역할을 하여,
  Logging framework간 호환성을 보장하는 역할을 함

*** log4j.xml (로그 관련 설정 파일) 구조**

- Appender : 전달된 로그를 어디에 출력할지 결정(콘솔출력, 파일기록, DB저장)

- Logger/root : 출력할 메세지를 Appender에 전달한다. 로그 주체

  # name 속성 : 로그 주체, 패키지 작성

  # additivity 속성 : 로그가 상위로 전달할 지 여부. 기본값 true

  # appender-ref 자식태그 : ref 속성값으로 appeder태그 name 값을 지정.

  # level 자식태그 : 로그레벨을 설정함. 설정된 값 이상의 priority일 경우, 로깅출력

  (DEBUG < INFO < WARN < ERROR < FATAL)

*** Appenders 의 종류**

- **ConsoleAppender** : 로그를 콘솔에 출력하기 위한 Appender
- JDBCAppender : 로그를 RDB에 출력하기 위한 Appender
- FileAppender : 로그를 파일에 출력하기 위한 Appender

  단, 지정한 파일에 로그가 계속 남기때문에 크기가 지나치게 커질 수 있고

  지속적인 로그관리 불가능해짐

- **RollingFileAppender** : FileAppender를 보완한 개념

  일정한 조건 후에 기존파일을 백업파일로 바꾸고 다시 처음부터 로깅 시작

  대표적인 예로 **DailyRollingFileAppender**가 있음.

*** Layout : 로그를 어떤 형식으로 출력할 지 결정**

- Layout의 종류

  PatternLayout(가장 디버깅에 적합) / DateLayout / HTMLLayout

  SimpleLayout / XMLLayout

*** PatternLayout 패턴 지정 방법**

%p : debug, info, warn, error, fatal 등의 priority가 출력

%m : 로그 내용이 출력

%d : 로깅 이벤트가 발생한 시간을 출력

포맷을 %d{HH:mm:ss}, %d{yyyy-MM-dd HH:mm:ss} 같은 형태로 사용하면

SimpleDateFormat에 따른 포맷팅이 진행됨.

%t : 로그이벤트가 발생된 쓰레드의 이름을 출력

%% : % 표시를 출력.

%n : 플랫폼 종속적인 개행문자가 출력된다. rn 또는 n 일 것이다.

%c : 카테고리를 표시

-> 예) 카테고리가 a.b.c 처럼 되어있다면 %c{2}는 b.c가 출력된다.

%C : 클래스명을 표시

-> 예) 클래스 구조가 org.apache.xyz.SomeClass 처럼 되어있다면

%C{2}는 xyz.SomeClass가 출력

%F : 로깅이 발생한 프로그램 파일명을 출력

%l : 로깅이 발생한 caller의 정보를 출력

%L : 로깅이 발생한 caller의 라인수를 출력

%M : 로깅이 발생한 method 이름 출력

%r : 어플리케이션 시작 이후 부터 로깅이 발생한 시점의 시간(milliseconds)

%x : 로깅이 발생한 thread와 관련된 NDC(nested diagnostic context)를 출력

*** log4j의 DTD 파일 경로가 변경됨**

➔ log4j.xml 2번째 줄 내용을 아래 내용으로 변경

<!DOCTYPE log4j:configuration SYSTEM

"http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/xml/doc-files/log4j.dtd">