Logback

이미지출처 : https://www.jorgeacetozi.com/single-post/changing-logback-log-level-at-runtime-using-etcd-watch-feature

1. 개요

- Logback은 Java 애플리케이션을 위한 로깅 프레임워크로, 성능이 뛰어나고 유연한 설정이 가능하여 다양한 로깅 요구 사항을 충족한다.

- Logback은 세 가지 모듈로 구성되어 있다.
  logback-core :모든 Logback 모듈의 기본 코어 모듈
  logback-classic : SLF4J API와 통합되는 구현체
  logback-access : 서블릿 컨테이너에서 HTTP 접근 로깅을 제공

- 설정 파일을 통해 다양한 로깅 설정을 할 수 있으며, 여러 종류의 Appender와 Encoder를 사용하여 원하는 형식으로 로그를 사용할 수 있다.

- 레벨별로 체계적인 로그관리가 가능하다.

- SQL , File등 외부 기능과 연동하여 부가적인 기능 구현이 가능하다.

- 여러개발자가 협업을 하는 경우 Logback을 사용하여 로깅방법을 통일할 수 있다.

- Logger객체를 통해 애플리케이션의 다양한 부분에서 로깅을 수행할 수 있다.

2. Logback 설정

Logback 설정은 logback.xml 파일을 통해 이루어진다.
이 파일은 애플리케이션의 클래스패스(src/main/resources)에 위치해야 하며, Logback은 설정 파일을 통해 로깅 출력 형식, 레벨, 대상 등을 정의할 수 있다.
> logback(기본예시코드).xml , logback(고급예시코드).xml 참조

3. Logback 주요 개념

Logger
- 로깅 요청을 담당하는 객체로, 이름을 가지며 계층 구조를 형성한다. LoggerFactory를 통해 인스턴스를 생성한다.

Appender
- 로깅된 메시지를 특정 출력 대상으로 보내는 역할을 한다. 콘솔, 파일, 네트워크 등이 될 수 있다.

Encoder
- 로그 메시지를 문자열로 변환하는 역할을 한다.

Layout
- 로그 메시지의 형식을 정의한다.

Level
- 로그 메시지의 중요도를 나타내며, TRACE, DEBUG, INFO, WARN, ERROR 등이 있다.
- 로깅 레벨을 설정하면 그 이상 레벨을 로깅한다.
- Spring Boot는 기본적으로 INFO 레벨로 설정되어 있다.

level 6) FATAL
몇몇 로깅 프레임워크에서만 지원하는 레벨로 치명적인 오류를 나타낸다.

level 5) ERROR
치명적인 오류 메시지를 기록한다. 애플리케이션은 계속 실행될 수 있지만 심각한 문제가 발생했음을 나타낸다.

level 4) WARN
경고 메시지를 기록한다. 어떤 문제가 발생했지만 애플리케이션은 계속 실행될 수 있다.

level 3) INFO

일반적인 실행에 대한 정보를 기록한다. 애플리케이션의 주요 이벤트 및 상태 변경에 대한 정보를 보여준다.

level 2) DEBUG : 디버깅에 유용한 로그를 기록한다. 개발 중에 자세한 정보를 확인할 때 사용된다.

level 1) TRACE : 가장 낮은 로깅 레벨로 매우 상세한 로그를 기록한다.