8253jang@daum.net
장희정

log4j란?

- **Java 애플리케이션에서 빠르고 효과적으로 로그를 남길 수 있게 해주는 오픈소스 라이브러리이다.**
- System.out.println()으로 디버깅 메시지를 찍는 대신, 로그를 **레벨별로 관리**하고 **출력 위치와 형식**을 설정할 수 있다.

log4j의 구조

log4j는 크게 **3가지 컴포넌트**로 이루어져 있다.

1. **Logger**
   - 애플리케이션 코드에서 로그를 남기는 주체.
   - 메시지를 Appender로 전달한다.
2. **Appender**
   - 전달받은 로그 메시지를 어디에 기록할지 지정.
   - 예: 콘솔, 파일, DB, 소켓 등
3. **Layout**
   - 로그 출력 형식을 정의.
   - 예: 날짜, 로그레벨, 클래스명, 메시지 등을 조합해 출력

Layout 옵션 예시

- %d : 로그 발생 시각
- %p : 로그 레벨 (INFO, DEBUG 등)
- %c : 카테고리(보통 클래스 이름)
- %m : 실제 로그 메시지
- %n : 개행문자
- %t :로깅 이벤트를 생성한 스레드 이름
- %F :로깅요청을 일으킨 파일 이름
- %L :로깅요청을 일으킨 파일의 행번호
- %C :로깅요청을 일으킨 호출자의 완전한 클래스이름

log4j 로그 레벨

로그는 심각도에 따라 다음과 같은 레벨을 가진다:

TRACE < DEBUG < INFO < WARN < ERROR < FATAL

- **FATAL** : 치명적인 에러 (애플리케이션 중단 수준)
- **ERROR** : 일반적인 에러
- **WARN** : 경고, 주의 필요
- **INFO** : 실행 과정의 일반 정보

8253jang@daum.net
장희정

- **DEBUG** : 상세한 실행 정보 (개발 시 주로 사용)
- **TRACE** : 가장 상세한 수준의 추적 로그
-

👉 예: 설정을 DEBUG로 두면 **DEBUG~FATAL** 로그가 모두 출력됨.
👉 운영 모드에서는 보통 **INFO 이상**만 출력하도록 설정

설정 파일 (log4j.properties)

log4j는 보통 log4j.properties 또는 log4j.xml 설정 파일을 통해 사용한다.
이 파일은 classpath(Eclipse: src, Tomcat: WEB-INF/classes)에 위치해야 한다.

예시: log4j.properties

```
# Root Logger 설정
log4j.rootLogger=DEBUG, Console, File


# 콘솔에 출력하는 Appender
log4j.appender.Console=org.apache.log4j.ConsoleAppender
log4j.appender.Console.layout=org.apache.log4j.PatternLayout
log4j.appender.Console.layout.ConversionPattern=%d [%t] %-5p %c - %m%n


# 파일에 출력하는 Appender
log4j.appender.File=org.apache.log4j.FileAppender
log4j.appender.File.File=app.log
log4j.appender.File.Append=true
log4j.appender.File.layout=org.apache.log4j.PatternLayout
log4j.appender.File.layout.ConversionPattern=%d %-5p %c - %m%n
```

사용 예제 (Java 코드)

8253jang@daum.net
장희정

```java
import org.apache.log4j.Logger;

public class Log4jExample {
    // Logger 생성
    private static final Logger logger = Logger.getLogger(Log4jExample.class);

    public static void main(String[] args) {
        logger.trace("Trace Message!");
        logger.debug("Debug Message!");
        logger.info("Info Message!");
        logger.warn("Warn Message!");
        logger.error("Error Message!");
        logger.fatal("Fatal Message!");
    }
}
```

실행 결과 (예시)

```
2025-09-23 14:00:01 [main] INFO  Log4jExample - Info Message!
2025-09-23 14:00:01 [main] WARN  Log4jExample - Warn Message!
2025-09-23 14:00:01 [main] ERROR Log4jExample - Error Message!
2025-09-23 14:00:01 [main] FATAL Log4jExample - Fatal Message!
```

정리

- **log4j는 Logger, Appender, Layout 구조로 동작**
- **로그 레벨을 통해 개발/운영 환경에 맞게 로그를 관리**
- **설정 파일(log4j.properties)로 로그 출력 형식과 위치 지정 가능**
- **실제 코드에서는 Logger 객체를 사용하여 로그 출력**