**Log4j 정의**

Log4j : Log for Java  
로그문의 출력을 다양한 대상으로 할 수 있도록 도와주는 도구(오픈소스)

**Log4j 특징**

- log4j는 속도에 최적화  
- log4j는 이름있는 로그 계층에 기반  
- log4j는 fail-stop이지만 신뢰성은 없음  
- log4j는 thread-safe(멀티스레드 환경에서 사용해도 안전하다:역주).  
- log4j는 융통성이 풍부  
- 설정 파일은 property 파일과 XML 형식으로 실행 중 수정 적용 가능  
- log4j는 처음부터 자바의 예외를 처리하기 위해 디자인  
- log4j는 출력을 파일, 콘솔, java.io.OutputStream, java.io.Writer, TCP를 사용하는 원격서버, 원격 Unix Syslog 데몬, - 원격 JMS 구독자, 윈도우NT EventLog로 보낼 수 있고, 심지어는 e-mail로 보낼 수도 있음  
- log4j는 다음 6단계의 장애레벨을 사용. < TRACE(추가), DEBUG, INFO, WARN, ERROR, FATAL >  
- 로그 출력의 형식은 Layout 클래스를 확장함으로써 쉽게 바꿀 수 있음  
- 로그가 출력될 대상과 출력 방법은 Appender 인터페이스로 할 수 있음  
- log4j는 로거 하나에 다수의, 출력을 담당하는 appender를 할당할 수 있음  
- log4j는 국제화를 지원

**Log4j 구조**

|  |  |
| --- | --- |
| **Logger(Category)** | 로깅 메세지를 Appender에 전달  log4J의 심장부에 위치  개발자가 직접 로그출력 여부를 런타임에 조정  logger는 로그레벨을 가지고 있으며, 로그의 출력 여부는 로그문의 레벨과 로거의 레벨을 가지고 결정 |
| **Appender** | 로그의 출력위치를 결정(파일, 콘솔, DB 등)  log4J API문서의 XXXAppender로 끝나는 클래스들의 이름을 보면, 출력위치를 어느정도 짐작가능 |
| **Layout** | Appender가 어디에 출력할 것인지 결정했다면 어떤 형식으로 출력할 것이지 출력 layout을 결정 |

**Log4j 레벨**

|  |  |
| --- | --- |
| **FATAL** | 아주 심각한 에러가 발생한 상태. 시스템적으로 심각한 문제가 발생해서 어플리케이션작동이 불가능할 경우가 해당하는데, 일반적으로는 어플리케이션에서는 사용할 일이 없음 |
| **ERROR** | 요청을 처리하는 중 문제가 발생한 상태를 나타냄 |
| **WARN** | 처리 가능한 문제이지만, 향후 시스템 에러의 원인이 될 수 있는 경고성 메시지를 나타냄 |
| **INFO** | 로그인, 상태변경과 같은 정보성 메시지를 나타냄 |
| **DEBUG** | 개발시 디버그 용도로 사용한 메시지를 나타냄 |
| **TRACE** | log4j1.2.12에서 신규 추가된 레벨로서, DEBUG 레벨이 너무 광범위한 것을 해결하기 위해서 좀 더 상세한 상태를 나타냄 |
| FATAL > ERROR > WARN > INFO > DEBUG > TRACE DEBUG 레벨로 했다면 INFO~FATAL까지 모두 logging이 되어집니다 | |

**Log4j Pattern Option**  
    일반적으로 PatternLayout을 사용하는 것이 디버깅에 가장 적합함

**\* 붉은 색으로 표시된 항목은 실행속도에 영향이 있음**

|  |  |
| --- | --- |
| **%p** | debug, info, warn, error, fatal 등의 priority 출력 |
| **%m** | 로그내용 출력 |
| **%d** | 로깅 이벤트가 발생한 시간을 출력   ex)포맷은 %d{HH:mm:ss} 같은 형태의 SimpleDateFormat |
| **%t** | 로그이벤트가 발생된 쓰레드의 이름 출력 |
| **%F** | 로깅이 발생한 프로그램 파일명 출력 |
| **%l** | 로깅이 발생한 caller의 정보 출력 |
| **%L** | 로깅이 발생한 caller의 라인수 출력 |
| **%M** | 로깅이 발생한 method 이름 출력 |
| **%** | % 표시 출력 |
| **%n** | 플랫폼 종속적인 개행문자 출력 |
| **%c** | 카테고리 출력   ex)카테고리가 a.b.c 처럼 되어있다면 %c{2}는 b.c 출력 |
| **%C** | 클래스명 출력   ex)클래스구조가 org.apache.xyz.SomeClass 처럼 되어있다면 %C{2}는 xyz.SomeClass 출력 |
| **%r** | 어플리케이션 시작 이후 부터 로깅이 발생한 시점의 시간(milliseconds) 출력 |
| **%x** | 로깅이 발생한 thread와 관련된 NDC(nested diagnostic context) 출력 |
| **%X** | 로깅이 발생한 thread와 관련된 MDC(mapped diagnostic context) 출력 |

**Log4j 주요 클래스**

|  |  |
| --- | --- |
| **ConsoleAppender** | org.apache.log4j.ConsoleAppender   콘솔에 로그 메시지 출력 |
| **FileAppender** | org.apache.log4j.FileAppender   파일에 로그 메시지 기록 |
| **RollingFileAppender** | org.apache.log4j.rolling.RollingFileAppender   파일 크기가 일정 수준 이상이 되면 기존 파일을 백업파일로 바꾸고 처음부터 기록 |
| **DailyRollingFileAppender** | org.apache.log4j.DailyRollingFileAppender     일정 기간  단위로 로그 파일을 생성하고 기록 |
| **JDBCAppender** | org.apache.log4j.jdbc.JDBCAppender   DB에 로그를 출력. 하위에 Driver, URL, User, Password, Sql과 같은 parameter를 정의할 수 있음 |
| **SMTPAppender** | 로그 메시지를 이메일로 전송 |
| **NTEventAppender** | 윈도우 시스템 이벤트 로그로 메시지 전송 |

출처: <https://cofs.tistory.com/354> [CofS]