Log4J란?

정의

Log4j는 JAVA를 위한 신뢰할만한 빠르고 유연한 로깅 프레임워크

WHY Logging?

로그메시지를 소스 코드 안에 삽입하는 것은 디버깅을 위한 가장 단순 무식한 로깅방법이다. 별도의 디버깅 툴이 제공되지 않는다면 이 방법이 최선의 방법일 수도 있다.

WHY Log4j?

Log4j를 이용하면, 프로그램 실행 시, 실행 코드의 수정 없이 설정파일을 통해서 로깅 작업을 컨트롤 할 수 있다. Log4J의 특별한 기능중 하나는 로거의 상속 개념의 사용이다. Logger 계층구조를 이용하면 어떤 로그문을 출력할지 상세하게 컨트롤하기가 무척 쉬워진다.

Log4j 주요 구성 요소

- **Logger** : 로깅 정보를 캡쳐
  - ✓ 로깅 메시지를 Appender에 전달
  - ✓ log4J의 심장부에 위치
  - ✓ 개발자가 직접 로그 출력 여부를 런타임에 조정
  - ✓ logger는 로그 레벨을 가지고 있으며, 로그의 출력 여부는 로그문의 레벨과 로거의 레벨을 가지고 결정

Log4j 주요 구성 요소

- **Appender** : 다양한 목적지로 로깅 정보를 출력
  - ✓ 로그의 출력 위치를 결정(파일, 콘솔, DB 등)
  - ✓ log4J API문서의 XXXAppender로 끝나는 클래스들의 이름을 보면, 출력 위치를 어느 정도 짐작 가능하다.
  - ✓ ConsoleAppender, FileAppender, JDBCAppender, JMSAppender, SMTPAppender, SocketAppender, SyslogAppender

ConsoleAppender	org.apache.log4j.ConsoleAppender - 콘솔에 로그 메시지 출력
FileAppender	org.apache.log4j.FileAppender - 파일에 로그 메시지 기록
RollingFileAppender	org.apache.log4j.RollingFileAppender - 파일 크기가 일정 수준 이상이 되면 기존 파일을 백업 파일로 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, SqI과 같은 parameter를 정의할 수 있음
SMTPAppender	- 로그 메시지를 이메일로 전송
NTEventAppender	- 윈도우 시스템 이벤트 로그로 메시지 전송

Log4j 주요 구성 요소

❖**layouts** : 로깅 정보를 위한 다양한 출력 포맷 구성

- ✓ Appender가 어디에 출력할 것인지 결정했다면 어떤 형식으로 출력할 것이지 출력 layout을 결정

Pattern Conversion Characters

변환문자	설명
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
d	로깅 이벤트가 발생한 시간을 출력 ex)포맷은 %d{HH:mm:ss} 같은 형태의 SimpleDateFormat
F	로깅이 발생한 프로그램 파일명 출력
l	로깅이 발생한 caller의 정보 출력
L	로깅이 발생한 caller의 라인수 출력
m	로그내용 출력

Pattern Conversion Characters(계속)

M	로깅이 발생한 method 이름 출력
n	플랫폼 종속적인 개행문자 출력
p	debug, info, warn, error, fatal 등의 priority 출력
r	어플리케이션 시작 이후부터 로깅이 발생한 시점의 시간(milliseconds) 출력
t	로그이벤트가 발생된 쓰레드의 이름 출력
x	로깅이 발생한 thread와 관련된 NDC(nested diagnostic context) 출력
X	로깅이 발생한 thread와 관련된 MDC(mapped diagnostic context) 출력
%	% 표시 출력

Format Modifiers

Format modifier	왼쪽 정렬	최소 너비	최대 너비
%20c	false	20	none
%-20c	true	20	none
%.30c	NA	none	30
%20.30c	false	20	30
%-20.30c	true	20	30

DatePattern

실행주기	날짜패턴	생성되는 로그파일이름
Minutely	'.'yyyy-MM-dd-HH-mm	sample.log.2017-11-09-21-54
Hourly	'.'yyyy-MM-dd-HH	sample.log.2017-11-09-22
Half-daily	'.'yyyy-MM-dd-a	sample.log.2017-11-09-AM sample.log.2017-11-09-PM
Daily	'.'yyyy-MM-dd	sample.log.2017-11-09
Weekly	'.'yyyy-ww	sample.log.2017-45 sample.log.2017-46
Monthly	'.'yyyy-MM	sample.log.2017-10 sample.log.2017-11

Log4j 로그레벨

로그레벨	설명
TRACE	log4j1.2.12에서 신규 추가된 레벨로서, DEBUG 레벨이 너무 광범위한 것을 해결하기 위해서 좀 더 상세한 상태를 나타냄
DEBUG	개발 시 디버그 용도로 사용한 메시지
INFO	로그인, 상태 변경과 같은 정보성 메시지 출력 모드로 일반적으로 많이 사용함.
WARN	잠재적인 위험(경고) 메시지 출력을 위한 모드
ERROR	애플리케이션 실행 중 발생하는 에러메시지 출력 모드
FATAL	아주 심각한 에러가 발생한 상태. 시스템적으로 심각한 문제가 발생해서 어플리케이션작동이 불가능할 경우가 해당하는데, 일반적으로는 어플리케이션에서는 사용할 일이 없음.

❖ 우선순위 : TRACE < DEBUG < INFO < WARN < ERROR < FATAL
DEBUG 레벨로 했다면 INFO~FATAL까지 모두 logging이 된다.

Log4j 설정방법

❖ 해당사이트에서 라이브러리파일(.jar) 다운로드 및 설치

❖ 설정파일 (log4j.properties 또는 log4j.xml) 을 클래스 패스 위치에 생성

❖ 설정파일의 설정항목을 통해서 로깅 레벨 및 로깅 처리 정보 설정