Log4j

Log4j란

- Java application에서 로깅을 할 수 있도록 도와주는 Open source Project
- 관련사이트
  - **메인사이트 :** http://logging.apache.org/log4j/
  - **다운로드** : http://logging.apache.org/log4j/1.2/download.html
  - **매뉴얼** : http://logging.apache.org/log4j/1.2/manual.html
  - **API** : http://logging.apache.org/log4j/1.2/apidocs/index.html
- 설치
  - 위 다운로드 사이트에서 최신 패키지를 다운 받는다.
  - 압축을 푼뒤 api인 jar파일을 classpath안에 저장한다.

Log4j 구성요소

- Logger (Category): Log4j의 핵심 클래스로 로그파일을 **작성**하고 관리하는 역할.
- Appender : Logger로 부터 전달된 Logging 메세지를 **어디에** 출력 또는 저장할 것인지 결정하는 역할
- Layout : Logging 메시지를 어떤 형식으로 **출력**할 것인지 출력 Layout을 결정한다.

Log4j 로그의 레벨

- 로그 레벨 (Priority)
  - FATAL : 가장 심각한 상황의 에러가 났을 경우 사용한다.
  - ERROR : 일반적인 에러가 났을 때 사용한다.
  - WARN : 에러는 아니나 주의가 필요한 경우 사용한다.
  - INFO : 일반 정보를 나타낼 때 사용한다.
  - DEBUG : 개발 시 프로그램 디버깅용 메시지를 출력해야 하는 경우 사용한다.
  - TRACE : DEBUG 보다 낮은 레벨의 메시지 출력시 사용
- 우선순위는 FATAL이 가장 높고 TRACE가 가장 낮다.
- 로그레벨로 선택된 것 이상의 로그레벨 메시지가 모두 출력된다.
  - Ex : INFO로 설정한 경우 -> FATAL, ERROR, WARN, INFO로 설정된 메시지가 모두 출력된다.

Log4j 로그 출력 설정

- **출력 설정**
  - 출력패턴, 로그 레벨등의 사항을 정의한다.
  - 설정파일을 이용, 프로그램 내에 직접 명시하는 두 가지 방법이 있다.
  - 파일명을 log4j.properties 로 하여 classpath내에 저장한다.
  - Web application의 경우 WEB-INF/classes 밑에 저장한다.
  - 작성형식은 자바 프로퍼티 형식으로 name=value 형식으로 작성한다.

Log4j 설정파일의 예

```
log4j.rootLogger=INFO, stdout, rolling

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d %-5p [%t] %-17c{2}
(%13F:%L) %3x - %m%n

log4j.appender.rolling=org.apache.log4j.DailyRollingFileAppender
log4j.appender.rolling.File=logfile.log
log4j.appender.rolling.Append=true
log4j.appender.rolling.DatePattern='.'yyyy-MM-dd
log4j.appender.rolling.layout=org.apache.log4j.PatternLayout
log4j.appender.rolling.layout.ConversionPattern=%d %-5p [%t] %-17c{2}
(%13F:%L) %3x - %m%n
```

Log4j 설정파일

- log4j.rootLogger=DEBUG,stdout, rolling
  - 로그의 레벨과 아래에서 설정할 Appender들의 이름을 지정한다.
  - 레벨은 DEBUG이고 Appender는 stdout, rolling 두개를 지정했다. 이 두 값을 이용해 각각의 Appender에 대한 설정을 하게된다.
- log4j.appender.stdout=org.apache.log4j.ConsoleAppender
  - stdout의 Appender종류를 세팅 한다.
  - **ConsoleAppender** :표준 출력 Appender로 System.out 이나 System.err을 이용해 출력한다.
  - **DailyRollingFileAppender** : FileAppender의 하위로 날짜별로 로그 파일을 만들어 그 파일에 출력한다.
  - 그외 Appender : RollingFileAppender, WriterAppender, AsyncAppender, FileAppender
- log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
  - 출력 Layout을 설정한다.
  - PatternLayout : C의 printf에서 사용하는 전환 패턴(conversion pattern)을 사용하여 출력 Layout을 정의 한다. (다음 페이지 참고)
  - 그외 Layout : HTMLLayout, SimpleLayout, XMLLayout등

형식	설명
%p	debug, info, warn, error, fatal 등의 priority 가 출력된다.
%m	로그내용이 출력됩니다
%d	로깅 이벤트가 발생한 시간을 기록합니다. 포맷은 %d{HH:mm:ss, SSS}, %d{yyyy MMM dd HH:mm:ss, SSS}같은 형태로 사용하며 SimpleDateFormat에 따른 포맷팅을 하면 된다
%t	로그이벤트가 발생된 쓰레드의 이름을 출력합니다.
%%	% 표시를 출력하기 위해 사용한다.
%n	플랫폼 종속적인 개행문자가 출력된다. ₩r₩n 또는 ₩n 일것이다.
%c	Logger를 표시합니다 예)Logger가 a.b.c 처럼 되 있다면 %c{2}는 b.c가 출력됩니다. %-10c 이면 10글자까지만 보여줍니다.
%C	클래스명을 포시합니다. 예) 클래스구조가 org.apache.xyz.SomeClass 처럼 되어있다면 %C{2}는 xyz.SomeClass 가 출력됩니다
%F	로깅이 발생한 프로그램 파일명을 나타냅니다.
%l	로깅이 발생한 caller의 정보를 나타냅니다
%L	로깅이 발생한 caller의 라인수를 나타냅니다
%M	로깅이 발생한 method 이름을 나타냅니다.
%r	어플리케이션 시작 이후 부터 로깅이 발생한 시점의 시간(milliseconds)
%x	로깅이 발생한 thread와 관련된 NDC(nested diagnostic context)를 출력합니다.
%X	로깅이 발생한 thread와 관련된 MDC(mapped diagnostic context)를 출력합니다.

Log4j 설정파일

- log4j.appender.stdout.layout.ConversionPattern=%d %-5p [%t] %-17c{2} (%13F:%L) - %m%n
  - PatternLayout일 경우 출력 될 패턴을 정의 한다.
  - 시간 로그레벨 [발생쓰레드] 클래스명 (파일명:라인) – 내용
- log4j.appender.rolling=org.apache.log4j.DailyRollingFileAppender
  - log4j.appender.rolling.File=logfile.log : 로그를 남길 파일명(절대, 상대경로 다 됨)
  - log4j.appender.rolling.Append=true : 서버의 종료 여부와 상관없이 파일이 리셋되지 않는다.
  - log4j.appender.rolling.DatePattern='.'yyyy-MM-dd : 파일명 포맷 (당일이 지나면 백업파일을 만드는데 그때의 파일명 뒤에 날짜가 붙는다.)

날짜 형식

형식	설명
'.'yyyy-MM	매달 첫번째날에 로그파일을 변경합니다
'.'yyyy-ww	매주의 시작시 로그파일을 변경합니다.
'.'yyyy-MM-dd	매일 자정에 로그파일을 변경합니다.
'.'yyyy-MM-dd-a	자정과 정오에 로그파일을 변경합니다.
'.'yyyy-MM-dd-HH	매 시간의 시작마다 로그파일을 변경합니다.
'.'yyyy-MM-dd-HH-mm	매분마다 로그파일을 변경합니다.

로깅 프로그램 작성

1. org.apache.log4j.Logger.getLogger(String name 또는 Class clazz) 메소드를 통해 Logger객체를 가져 온다.
   Ex) Logger logger = Logger.getLogger(Abc.class);
   -> Abc.class는 로거의 이름이 된다.
2. Logger객체의 로깅 메소드를 통해 로그메세지를 출력한다.

Logging 메소드

Message만 출력	메시지와 오류객체 출력
logger.fatal(Object message)	logger.fatal(Object message, Throwable t)
logger.error(Object message)	logger.error(Object message, Throwable t)
logger.warn(Object message)	logger.warn(Object message, Throwable t)
logger.info(Object message)	logger.info(Object message, Throwable t)
logger.debug(Object message)	logger.debug(Object message, Throwable t)

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

public class Abc{
    private static Logger logger =
                        Logger.getLogger(Abc.class);
    public static void main(String[] args) {
        logger.debug("[DEBUG]모드 로그^^");
        logger.info ("[INFO] 모드 로그^^");
        logger.warn ("[WARN] 모드 로그^^", new
          Exception("aaa"));
        logger.error("[ERROR] 모드 로그^^");
        logger.fatal("[FATAL] 모드 로그^^");
    }
}
```