slf4j 설정-추천

목차

1. slf4j 란

System.out.println() 으로 로그를 찍는 방법과 로깅툴을 사용하는 방법의 성능은 천지차이이다.

System.out.println() 을 사용할때와 SLF4J 를 사용할때는 성능과 속도면에서 수십배의 차이가 생긴다.

간단한 시스템이라면 몰라도 조금만 규모가 커지면 서버가 감당해야 할 스트레스가 굉장히 커진다.

닷넷에서도 log4net 이 처음에는 많이 사용되다가, 속도, appender 문제로 NLog 가 더 많이 쓰이게 되었는데, Java 에서도 log4j 를 직접 사용하던가 common-logging 을 사용하던 방식에서 slf4j 로 facade 해서 사용하는 방식으로 바뀌었군요.

1.1 Logger

- 특정 대상에 대한 로그 출력
- appender-ref 를 통해 해당 appender 참조
- 애플리케이션 별로 정의하고 로그 레벨과 Appender 지정

1.2 Level

로그 레벨	설명
fatal	아주 심각한 에러가 발생한 상태
error	요청을 처리하는 중 문제가 발생한 상태
warn	처리 가능한 문제, 향후 에러의 원인이 될 수 있는 경고 메세지
info	로그인, 상태 변경과 같은 정보 메세지
debug	개발 시 디버그 용도로 사용한 메세지
trace	신규 추가된 레벨로 디버그 레벨이 넘 광범위한 것을 해결하기 위해서 좀 더 상세한 상태를 나타냄

1.3 Appender

- 출력 포맷 설정
- Threshold, File, Append
- DatePattern

형식	설명
'.'yyyy-MM	매달 시작 일에 로그 파일 변경
'.'yyyy-ww	매주 시작 일에 로그 파일 변경

'.'yyyy-MM-dd	매일 자정에 로그 파일 변경
'.'yyyy-MM-dd-a	자정과 정오에 로그 파일 변경
'.'yyyy-MM-dd-HH	매 시간마다 로그 파일 변경
'.'yyyy-MM-dd-HH-mm	매 분마다 로그 파일 변경

1.4 Layout

형식	설명
%P	debug, info, warn, error, fatal 등 priority 출력
%m	로그 내용 출력
%d	발생 시간 출력
%t	발생 쓰레드의 이름 출력
%n	개행 문자 출력
%c	패키지 출력 / {숫자} 를 이용하여 단계별 출력, ex) %x{5}
%C	클래스 명 출력 / {숫자} 를 이용하여 단계별 출력, ex) %x{5}
%F	프로그램 파일 명 출력
%I	로깅이 발생한 caller 의 정보 출력
%L	로깅이 발생한 caller 의 라인 수 출력
%M	로깅이 발생한 method 명 출력
%r	애플리케이션 시작 이후 로깅이 발생한 시점의 시간 출력(millisecond)
%x	NDC 에 저장된 내용 출력
%X	MDC 에 저장된 내용 출력

1.5 Root

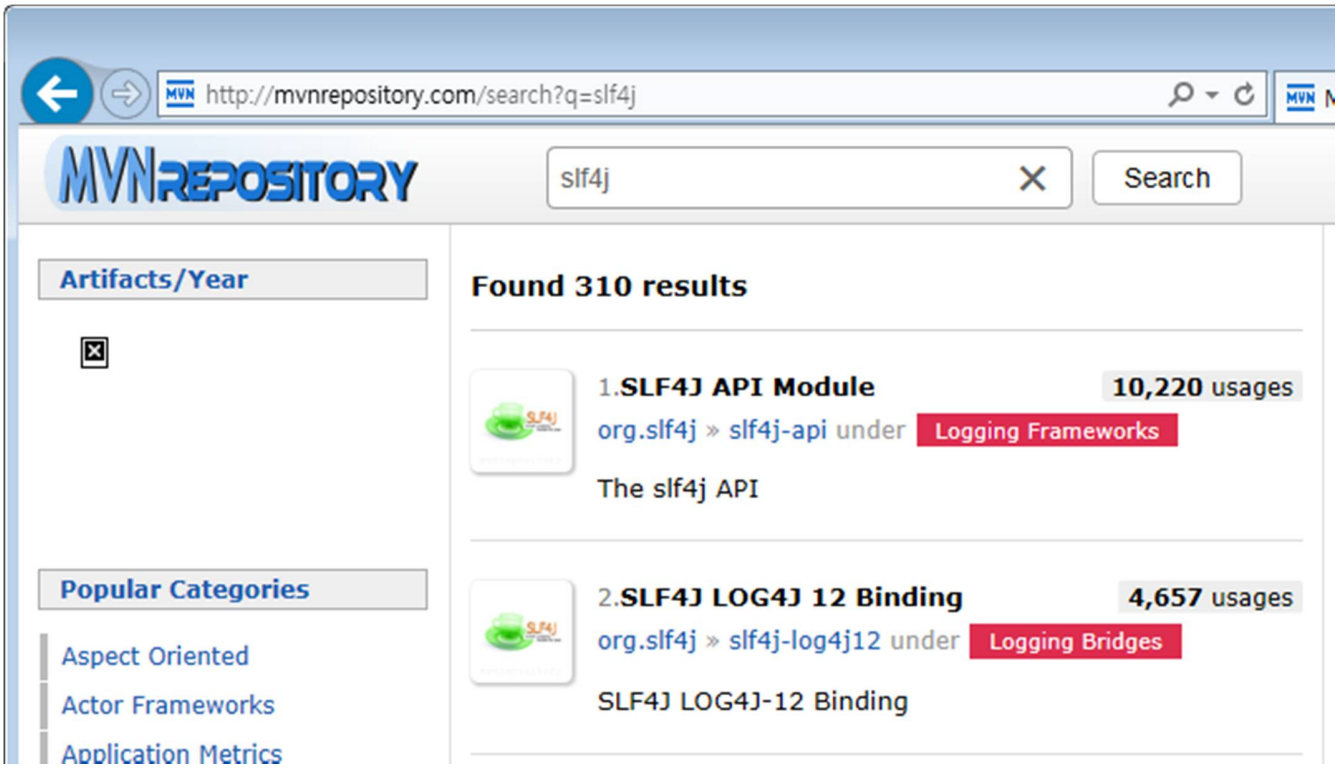
- 기본적으로 출력 될 로그 출력
- appender-ref 를 통해 해당 appender 참조

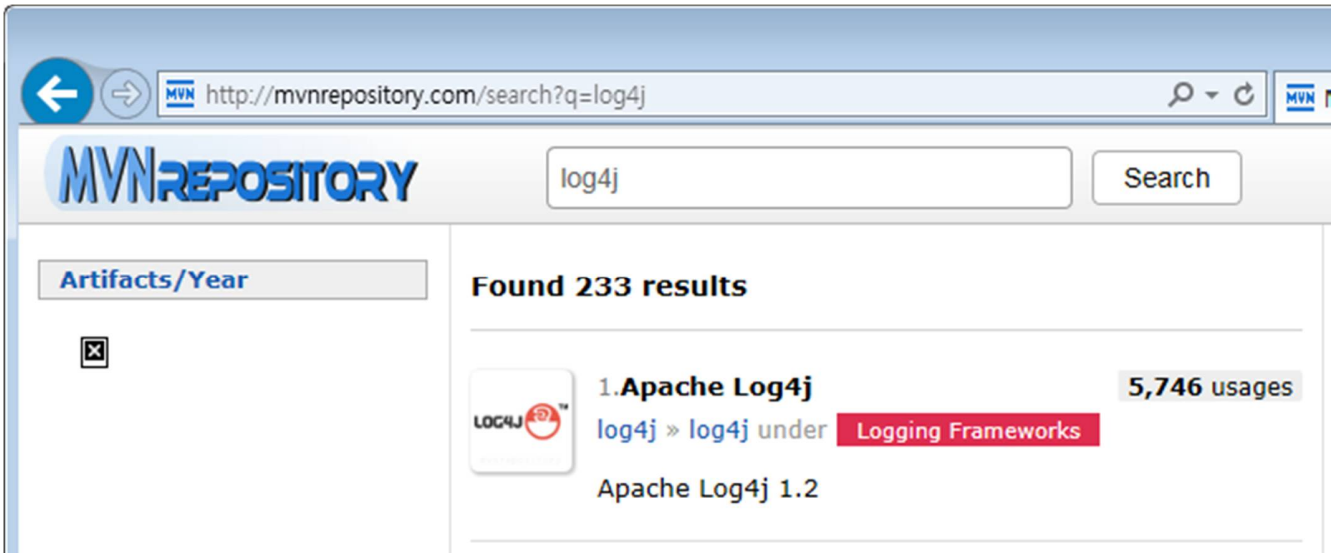
2. slf4j 라이브러리 검색

slf4j-api-1.7.21.jar

slf4j-log4j12-1.7.21.jar

log4j-1.2.16.jar





3. build.gradle 에 라이브러리 추가

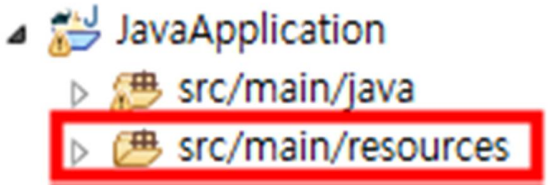
```
apply plugin: 'java'
apply plugin: 'eclipse'

sourceCompatibility = 1.8

version = '1.0'

jar {
    manifest {
        attributes 'Implementation-Title': 'Gradle Quickstart',  'Implementation-
Version': version
    }
}

repositories {
    mavenCentral()
}

dependencies {
    compile 'commons-collections:commons-collections:3.2'

    // log 관련 라이브러리 추가
    compile 'org.slf4j:slf4j-api:1.7.21'
    compile 'org.slf4j:slf4j-log4j12:1.7.21'
    compile 'log4j:log4j:1.2.17'

    testCompile 'junit:junit:4.+'
}

test {
    systemProperties 'property': 'value'
}
```

4. log4j.xml 설정

log4j 에 대한 설정은 **src/main/resources/log4j.xml** 파일에 있다.

log4j.xml 이 없으면 인터넷에서 파일을 찾아서 붙여 넣으면 된다.

주의사항. 위치가 바뀌면 설정이 적용되지 않는다. 반드시 **src/main/resources** 에 위치해야 한다.




```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration PUBLIC "-//APACHE//DTD LOG4J 1.2//EN" "log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">

    <!-- 콘솔 로그 -->
    <appender name="console" class="org.apache.log4j.ConsoleAppender">
        <param name="Target" value="System.out" />
        <param name="Threshold" value="DEBUG" />
        <!-- 출력 패턴 설정 -->
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="[%p] [%d{HH:mm:ss.SSS}] %m%n" />
        </layout>
    </appender>

    <!-- 일별 파일 로그 : 날짜별로  로그를 남김. Log file 에 날짜를 붙여 백업하는 방식 -->
    <appender name="rolling" class="org.apache.log4j.DailyRollingFileAppender">

        <!-- Tomcat Restart 시 새로쓸껀지 말껀지 : True 기존파일에 추가, False 새로씀 -->
        <param name="Append" value="true"/>

        <!-- Log File 뒤에 날짜 패턴 추가 -->
        <param name="DatePattern" value="'.'yyyyMMdd"/>

        <!-- Log File 위치 -->
        <param name="File" value="logs/kiwi.log"/>

        <!-- 출력 패턴 설정 -->
```

```
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="%t> [%d{yyyy-MM-dd HH:mm:ss}] [%c{1}]
[%L] [%p] %m %n"/>
        </layout>
    </appender>

    <appender name="file" class="org.apache.log4j.FileAppender">
        <param name="File" value="logs/test.log" />
        <param name="Append" value="false" />
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="[%p] [%d{HH:mm:ss.SSS}] %m%n" />
        </layout>
    </appender>


    <!-- Root Logger -->
    <root>
        <!-- level 에는 레벨에는 error, fatal, info, warn, debug 등이 있다. -->
        <level value="debug" />
        <appender-ref ref="console" />
        <appender-ref ref="rolling" />
    </root>

</log4j:configuration>
```

4.1 MyBatis 로그를 남기고 싶을 때는 아래 내용을 추가

```
<category name="java.sql.Connection">
    <priority value="debug" />
    <appender-ref ref="console" />
</category>
<category name="java.sql.Statment">
    <priority value="debug" />
    <appender-ref ref="console" />
</category>
<category name="java.sql.PreparedStatment">
    <priority value="debug" />
    <appender-ref ref="console" />
</category>
<category name="java.sql.ResultSet">
    <priority value="debug" />
    <appender-ref ref="console" />
</category>
```

나같은 경우는 소스상에서 정의한 info 이외에 시스템에서 발생하는 DEBUG 급의 메시지도 모두 로깅으로 남긴다. 언제 JDBC connection 을 가져왔으며, 어떤 메소드에서 transaction 을 발생시켰는지, 호출된 쿼리문은 어떤것인지 등등 시스템의 상황을 한눈에 파악할 수 있다.

4.2 Spring 로그를 남기고 싶을 때는 아래 내용을 추가

```
<!-- Application Loggers -->
<logger name="com.spring67.upload">
    <level value="info" />
</logger>

<!-- 3rdparty Loggers -->
<logger name="org.springframework.core">
    <level value="info" />
</logger>

<logger name="org.springframework.beans">
    <level value="info" />
</logger>

<logger name="org.springframework.context">
    <level value="info" />
</logger>

<logger name="org.springframework.web">
    <level value="info" />
</logger>
```

5. 소스에 log 코드 추가

로그를 쓰는 레벨에는 error, fatal, info, warn, debug 와 같이 다양하게 있으며 주로 사용하는 것은 info, debug 이다. 원하는 위치에서 아래와 같이 호출하여 사용하면 된다.

```
log.info("적을 로그");
log.debug("적을 로그");
log.warn("적을 로그");
```

5.1 Eclipse 에서 테스트 할 때

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Log4jTest {
    private static Logger log = LoggerFactory.getLogger(Log4jTest.class);

    public static void main(String[] args) {

        log.info("hello~!");
        if (log.isDebugEnabled()) {
            log.info("hello~!");
        }
    }
}
```

5.2 Android Studio 에서 테스트 할 때

```
public class Log4jTest {
    private static Logger log = LoggerFactory.getLogger( Log4jTest.class );

    public static void main( String[] args){

        java.io.File log4jfile = new java.io.File("src/main/resources/log4j.properties");
        org.apache.log4j.PropertyConfigurator.configure( log4jfile.getAbsolutePath() );

        log.info("hello~!");
        if (log.isDebugEnabled()) {
            log.debug("hello~!");
        }
    }
}
```

6. How to store logs in tables.

Log4j create simple log files, html log files or xml log files also. It also insert log statements into database.

6.1 Step 1) Create a maven java project and update log4j dependencies

Follow the steps given in this post related to configuring log4j with maven.

6.2 Step 2) Create the table in database and test the application

Create the database table LOGS, in schema test.

```
-- LogsDB 데이터베이스 구조 내보내기
DROP DATABASE IF EXISTS LogsDB;
CREATE  database LogsDB  CHARACTER SET utf8 COLLATE utf8_unicode_ci;

-- 데이터베이스 변경
USE LogsDB;

CREATE TABLE LOGS
(
    USER_ID VARCHAR(20) NOT NULL,
    DATED   DATETIME NOT NULL,
    LOGGER  VARCHAR(50) NOT NULL,
    LEVEL   VARCHAR(10) NOT NULL,
    MESSAGE VARCHAR(1000) NOT NULL
);


-- 사용자 추가
grant all on LogsDB.* to LogsDBUser01@'%' identified by 'LogsDBUser01';
flush privileges;


-- 사용자 추가 여부 확인
USE mysql;
select host, user, password from user;
```

6.3 Step 3) Configure JDBCAppender in log4j.properties file

The JDBCAppender provides mechanism for sending log events to a database tables. Each append call adds to an ArrayListbuffer. When the buffer is filled each log event is placed in a sql statement (configurable) and executed. BufferSize, db URL, User, &Password are configurable options in the standard log4j ways.

WARNING: This version of JDBCAppender is very likely to be completely replaced in the future. Moreoever, it does not log exceptions.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<log4j:configuration xmlns:log4j='http://jakarta.apache.org/log4j/'>

    <appender name="JDBC" class="org.apache.log4j.jdbc.JDBCAppender">
        <param name="URL" value="jdbc:mysql://localhost:3306/test" />
        <param name="user" value="LogsDBUser01" />
        <param name="password" value="LogsDBUser01" />
        <param name="driver" value="com.mysql.jdbc.Driver" />
        <param name="sql" value="INSERT INTO LOGS VALUES ('%x', now() ,'%C','%p','%m')" />
    </appender>

    <root>
        <level value="DEBUG" />
        <appender-ref ref="JDBC" />
    </root>
</log4j:configuration>
```

Property	Description
bufferSize	Sets the size of log messages wait in buffer. The Default size is 1.
driver	Sets the driver class of database. The default driver is sun.jdbc.odbc.JdbcOdbcDriver
layout	Set the layout used with logger. The Default layout is org.apache.log4j.PatternLayout
password	Set the password of database
user	Set the user name of database
URL	Set the database URL
sql	The sql statement to inset log into database for every logging event

6.4 Step 4) make a test code.

Now, configure the log4j.properties file using PropertyConfigurator and call some log events.

```
package com.lecture.log;

import java.io.File;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import org.apache.log4j.PropertyConfigurator;


public class Log4jJDBCExample {

    // SLF4J Logging
    private static Logger logger = LoggerFactory.getLogger(Log4jJDBCExample.class);


    public static void main(String[] args)
    {
        File log4jfile = new File("./src/main/resources/table.log4j.properties");
        PropertyConfigurator.configure(log4jfile.getAbsolutePath());

        logger.debug("Sample debug message");
        logger.info("Sample info message");
        logger.error("Sample error message");
        logger.trace("Sample trace message");
    }
}
```

Step 5) Log statements will be inserted in database using sql statement.

USER_ID	DATED	LOGGER	LEVEL	MESSAGE
	2015-10-13 12:25:39	com.lecture.log.Log4jJDBCExample	DEBUG	Sample debug message
	2015-10-13 12:25:39	com.lecture.log.Log4jJDBCExample	INFO	Sample info message
	2015-10-13 12:25:39	com.lecture.log.Log4jJDBCExample	ERROR	Sample error message

Let me know if any question.

http://howtodoinjava.com/2013/04/08/how-to-create-logs-in-database-using-jdbcappender-in-log4j/

http://www.javatutorialcorner.com/2013/09/log4j-jdbcappender-using-xml.html

slf4j 설정-추천

https://coderanch.com/t/529904/databases/log-jdbcplus-JDBCAppender-log-xml

7. Reference

http://sidekick.tistory.com/465
http://blog.daum.net/_blog/BlogTypeView.do?blogid=0LFt6&articleno=7861280
http://linuxism.tistory.com/591
http://blog.naver.com/PostView.nhn?blogId=kjwoo97&logNo=90080865451
http://logging.apache.org/

http://devofhwb.tistory.com/20