slf4j 설정-추천

목차

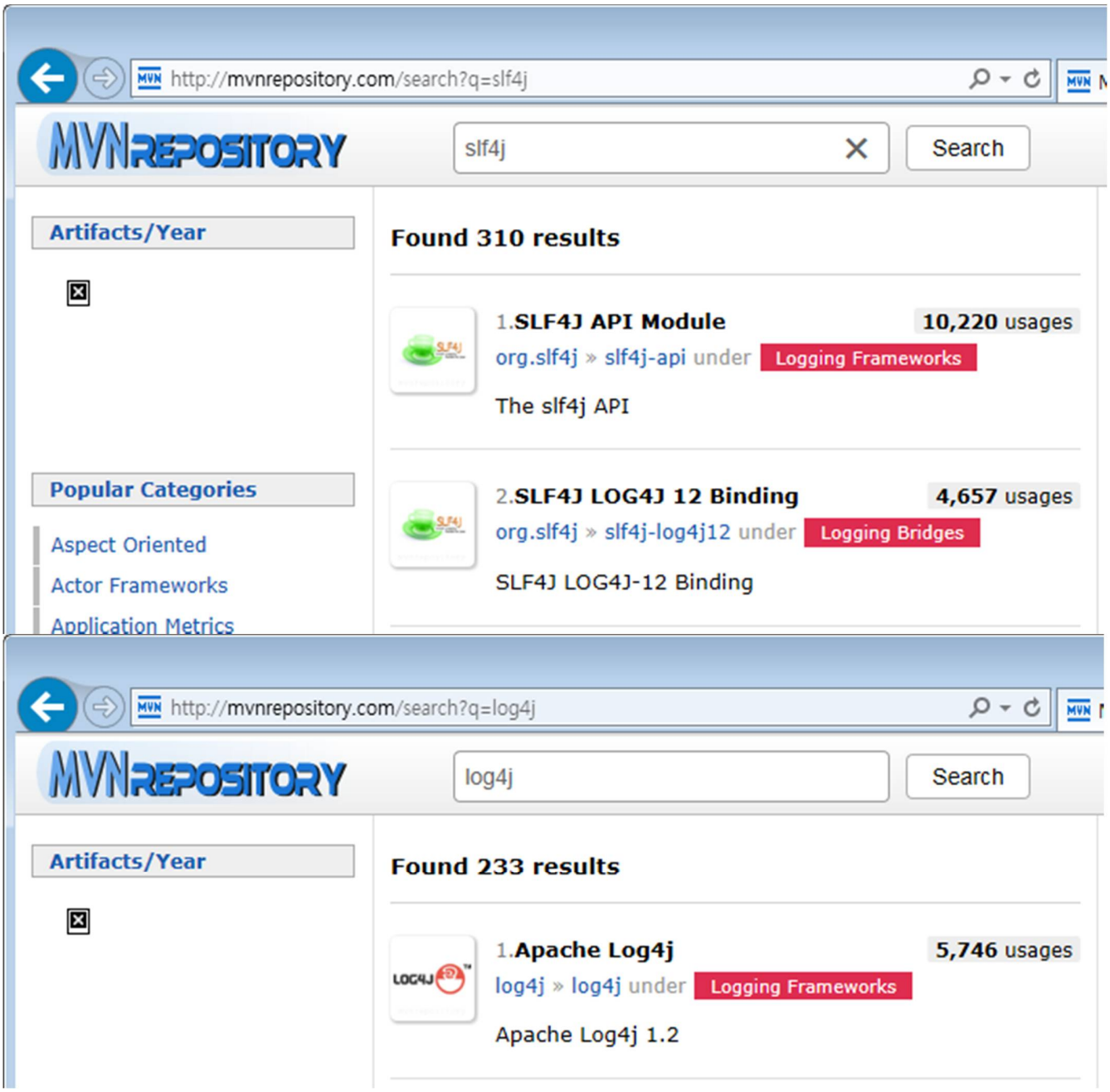
1. slf4j 란

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.

2. 라이브러리 추가

- slf4j-api-1.6.1.jar
- slf4j-log4j12-1.6.1.jar
- log4j-1.2.16.jar
-



3. build.gradle 수정

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

sourceCompatibility = 1.7
```

slf4j 설정-추천

```
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:1.7.12'
    compile 'org.slf4j:slf4j-log4j12:1.7.12'
    compile 'log4j:log4j:1.2.17'

    testCompile 'junit:junit:4.+'
}

test {
    systemProperties 'property': 'value'
}

uploadArchives {
    repositories {
       flatDir {
           dirs 'repos'
       }
    }
}
```

4. 소스에 log 코드 추가

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

public class EmployeeList{

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

    logger.info("hello~!");
    ..........


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

    ..........
}
```

를 추가한다.

5. log4j 설정

웹 프로젝트 인 경우 을 생성한 뒤 WEB-INF/classes 디렉토리 밑에 둔다.

log4j.properties

```
########################################
### Rules reminder:
### DEBUG < INFO < WARN < ERROR < FATAL
########################################



########################################
### Root Logger로 stout, rolling으로 셋팅
### 최상위 카테고리에 DEBUG로 레벨 설정 및 appender로 stdout, rolling을 정의
########################################
log4j.rootLogger=DEBUG, stout, rolling



########################################
### console 설정
########################################

#### 콘솔에 뿌려줌
log4j.appender.stout=org.apache.log4j.ConsoleAppender

#### DEBUG 이상 레벨에서만 찍는다.
log4j.appender.stout.Threshold=DEBUG

#### 출력 패턴 설정
log4j.appender.stout.layout=org.apache.log4j.PatternLayout
log4j.appender.stout.layout.ConversionPattern=%-5p at %C{3}.%M(%13F:%L) %3x - %m%n



########################################
### file 설정
########################################

#### Log File 설정
#### DailyRollingAppender 매일매일 Log file 을 날짜를 붙여서 백업하는 방식이다.
log4j.appender.rolling=org.apache.log4j.DailyRollingFileAppender

#### Log File 위치
```

slf4j 설정-추천

```
#log4j.appender.rolling.File=C:/log/cron.log
log4j.appender.rolling.File=./logs/cron.log  # 계정 홈 디렉토리에 저장됨.

#### Log File 뒤에 날짜 패턴 추가
log4j.appender.rolling.DatePattern='.'yyyy-MM-dd

#### Log File의 최대 사이즈
#### 300KB 넘을경우 뒤에 _1, _2 숫자가 붙는다.
#log4j.appender.rolling.MaxFileSize=300KB

#### Tomcat Restart시 새로쓸껀지 말껀지
#### True 기존파일에 추가
#### False 새로씀
log4j.appender.rolling.Append=true

#### 출력 패턴 설정
log4j.appender.rolling.layout=org.apache.log4j.PatternLayout
log4j.appender.rolling.layout.ConversionPattern=[%d] %-5p at %C{3}.%M(%13F:%L) %3x -
 %m%n



########################################
### mybatis 설정
### http://www.okjsp.net/seq/199384#1422521740338
########################################
log4j.logger.org.apache=DEBUG
log4j.logger.com.ibatis=DEBUG
log4j.logger.com.ibatis.common.jdbc.SimpleDataSource=DEBUG
log4j.logger.com.ibatis.common.jdbc.ScriptRunner=DEBUG
log4j.logger.com.ibatis.sqlmap.engine.impl.SqlMapClientDelegate=DEBUG

log4j.logger.java.sql.Connection=DEBUG
log4j.logger.java.sql.Statement=DEBUG
log4j.logger.java.sql.PreparedStatement=DEBUG
log4j.logger.java.sql.ResultSet=DEBUG
```

log4j.xml 설정

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

    <appender name="console" class="org.apache.log4j.ConsoleAppender">
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="%d %5p [%c] %m%n" />
        </layout>
    </appender>

    <logger name="java.sql">
     <level value="DEBUG"/>
        <appender-ref ref="console"/>
    </logger>

    <logger name="javada.board" additivity="false">
        <level value="INFO"/>
        <appender-ref ref="console"/>
    </logger>

 <!-- log SQL with timing information, post execution -->
 <logger name="jdbc.sqltiming" additivity="false">
  <level value="ERROR" />
  <appender-ref ref="console" />
 </logger>

 <logger name="org.springframework" additivity="false">
  <level value="ERROR" />
  <appender-ref ref="console" />
 </logger>

 <root>
  <level value="DEBUG" />
  <appender-ref ref="console" />
 </root>

</log4j:configuration>
```

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.

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.

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

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

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@localhost identified by 'LogsDBUser01';
flush privileges;

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


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

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.

```
# Define the root logger with file appender
log4j.rootLogger = DEBUG, sql


########################################
### sql appender  설정
### http://howtodoinjava.com/2013/04/08/how-to-create-logs-in-database-using-jdbcappender-in-log4j/
########################################
log4j.appender.sql=org.apache.log4j.jdbc.JDBCAppender
log4j.appender.sql.URL=jdbc:mysql://localhost:3306/LogsDB
log4j.appender.sql.driver=com.mysql.jdbc.Driver

# Set database user name and password
log4j.appender.sql.user=LogsDBUser01
log4j.appender.sql.password=LogsDBUser01

# Set the SQL statement to be executed.
log4j.appender.sql.sql=INSERT INTO LOGS VALUES ('%x', now() ,'%C','%p','%m')

# Define the xml layout for file appender
log4j.appender.sql.layout=org.apache.log4j.PatternLayout
```

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.cubrid.org/store_java_logs_to_databdase_using_log4j

7. Reference

http://sidekick.tistory.com/465

http://blog.daum.net/_blog/BlogTypeView.do?blogid=0LFt6&articleno=7861280

http://linuxism.tistory.com/591