로깅툴

로깅이란?

로깅(Logging)이란 프로그램을 개발하거나 운영할 때 생기는 문제점을 관리하고 모니터링 할 수 있는 데이터를 말합니다.

1. 로킹툴을 사용하는 이유

System.out.println() 명령어는 IO 리소스를 많이 사용하여 시스템이 느려질 수 있음
로그를 파일로 저장하여 분석할 필요가 있음

2. 로깅툴의 종류

commons-logging : 스프링 3에서 사용하던 로깅툴

log4j : 효율적인 메모리 관리로 그동안 많이 사용되었음

logback : log4j 보다 성능이 더 우수하여 최근에 많이 사용되고 있음

SLF4J : logback을 사용하기 위한 인터페이스

3. SLF4J 설정방법

ㄱ) pom.xml 의 slf4j - version을 1.7.25 로 설정

```
<properties>
        <java-version>1.8</java-version>
        <!-- 2019년 1월 현재 최신 버전 5.1.4, 에러가 날 경우 호환성을 위해 버전을 내려야 함 -->
        <org.springframework-version>5.1.3.RELEASE</org.springframework-version>
        <org.aspectj-version>1.9.2</org.aspectj-version>
        <org.slf4j-version>1.7.25</org.slf4j-version>
</properties>
```

ㄴ) pom.xml 에 라이브러리 추가
<dependency>태그를 이용해 추가를 하면 메이븐이 자동으로 다운로드 받아준다.

```
<dependency>
            <groupId>ch.qos.logback</groupId>
            <artifactId>logback-classic</artifactId>
            <version>1.2.3</version>
        </dependency>

        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>${org.slf4j-version}</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>jcl-over-slf4j</artifactId>
            <version>${org.slf4j-version}</version>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>${org.slf4j-version}</version>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.17</version>
            <exclusions>
                <exclusion>
                    <groupId>javax.mail</groupId>
                    <artifactId>mail</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>javax.jms</groupId>
                    <artifactId>jms</artifactId>
                </exclusion>
```

```
                <exclusion>
                    <groupId>com.sun.jdmk</groupId>
                    <artifactId>jmxtools</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>com.sun.jmx</groupId>
                    <artifactId>jmxri</artifactId>
                </exclusion>
            </exclusions>
            <scope>runtime</scope>
</dependency>
```

ㄷ) src/main/resources 에 logback.xml 파일을 작성

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <!-- log4jdbc-log4j2 -->
    <logger name="jdbc.sqlonly"        level="DEBUG"/>
    <logger name="jdbc.sqltiming"      level="INFO"/>
    <logger name="jdbc.audit"          level="WARN"/>
    <logger name="jdbc.resultset"      level="ERROR"/>
    <logger name="jdbc.resultsettable" level="ERROR"/>
    <logger name="jdbc.connection"     level="INFO"/>

    <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
        <layout class="ch.qos.logback.classic.PatternLayout">
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-4level [%logger.%method:%line]-
                %msg%n</pattern>
        </layout>
    </appender>

    <appender name="LOGFILE"
        class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>/WEB-INF/logback.log</file>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>logback.%d{yyyy-MM-dd}.log</fileNamePattern>
            <!-- 30일 지난 파일은 삭제한다. -->
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %-4level [%logger.%method:%line]
                - %msg %n</pattern>
        </encoder>
    </appender>

    <!-- 로그의 레벨( 지정된 로그 레벨 이상만 수집 ) : DEBUG < INFO < WARN < ERROR < FATAL -->
    <logger name="myweb" additivity="false">
        <level value="INFO" />
        <appender-ref ref="LOGFILE" />
        <appender-ref ref="CONSOLE" />
    </logger>

    <root>
        <level value="INFO" />
        <appender-ref ref="CONSOLE" />
    </root>

</configuration>
```

ㄹ) 로그를 수집할 클래스에 변수 선언

private static final Logger logger = LoggerFactory.getLogger (클래스 이름.class);

HomeController.java (파일중 일부)

```
@Controller
public class HomeController {

    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
    //로그를 수집할 클래스의 변수 선언

    /**
     * Simply selects the home view to render by returning its name.
     */
    @RequestMapping(value = "/", method = RequestMethod.GET)
```

private : 외부에서 로그를 가로채지 못하도록 하기 위해서
static final : 로그 내용이 바뀌지 않으므로

ㅁ) 로그를 수집할 method에서 로그 수집 명령어 호출

logger.info ("로그 타이틀", 출력할 값);

HomeController.java (파일중 일부)

```
@RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
        logger.info("Welcome home! The client locale is {}.", locale);
        //home 메소드에서 로그 수집 명령어를 호출한다.
```