| log4j.xml

먼저 log4j.xml 파일을 살펴보자.
"Project Name"\src\main\resources 경로에 log4j.xml 파일이 있다.

<appender> 태그에는 로그를 어떻게 출력할지를 등록할 수 있다.
name="console" 로 설정하고 console에 로그를 출력해보자.
ConversionPattern 으로 로그를 어떻게 남길지 설정할 수도 있다.

```
<appender name="console" class="org.apache.log4j.ConsoleAppender">
    <param name="Target" value="System.out" />
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern" value="%d{HH:mm:ss} %-5p: %c >> %m%n" />
    </layout>
</appender>
```

먼저 Application Loggers의 level을 설정할 수 있다.
Log level은 TRACE -> DEBUG -> INFO -> WARN -> ERROR -> FATAL 순서로 이루어져 있다.
> TRACE : Debug보다 좀더 상세한 정보

> DEBUG : 프로그램을 디버깅하기 위한 정보 지정

> INFO : 상태변경과 같은 정보성 메시지

> WARN : 처리 가능한 문제, 향후 시스템 에러의 원인이 될 수 있는 경고성 메시지

> ERROR : 요청을 처리하는 중 문제가 발생한 경우

> FATAL : 아주 심각한 에러가 발생한 상태, 시스템적으로 심각한 문제가 발생해서 어플리케이션 작동이 불가능할 경우

```
<logger name="com.cristoval.web">
    <level value="trace" />
</logger>
```

다음은 Spring 자체 로그다.

Spring 내부적으로 발생하는 로그를 의미한다.
사실상 Spring 내부적인 문제는 봐도 처리할 수 없는 경우가 많으므로..
INFO level이 default 이다.

```
<!-- 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>
```

위에서 해당되지 않는 로그는 Root Logger에 설정한다.
마찬가지로 console에 출력해주기 위해 appender-ref를 console로 설정하였다.

```
<!-- Root Logger -->
<root>
    <priority value="info" />
    <appender-ref ref="console" />
</root>
```

log4j.xml 전체 코드

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:config<logger name="com.cristoval.web">
        <level value="trace" />
    </logger>uration PUBLIC "-//APACHE//DTD LOG4J 1.2//EN" "log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">

    <appender name="console" class="org.apache.log4j.ConsoleAppender">
        <param name="Target" value="System.out" />
```

```
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="%d{HH:mm:ss} %-5p: %c >> %m%n" />
        </layout>
    </appender>

    <!-- Application Loggers -->
    <logger name="com.cristoval.web">
        <level value="trace" />
    </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>

    <!-- Root Logger -->
    <root>
        <priority value="info" />
        <appender-ref ref="console" />
    </root>

</log4j:configuration>
```

|| 로그 확인하기

Java 파일에서 System.out.println으로 log를 확인하는 것 보다 훨씬 편리하다.
Java 파일 어디에서나 logger만 등록되어있다면 사용 가능하다.

test

```
@Controller
public class HomeController {

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

    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {

```

```
9           logger.trace("> trace : {}, {}.", locale, "Hello");
10          logger.debug("> debug: {}.", locale);
11          logger.info("> info : {}.", locale);
12          logger.warn("> warn : {}.", locale);
13          logger.error("> error : {}.", locale);
14
15          return "home";
16      }
17  }
18
```

console

```
18:41:33 TRACE: com.cristoval.web.controller.HomeController >> > trace : ko_KR, Hello.
18:41:33 DEBUG: com.cristoval.web.controller.HomeController >> > debug: ko_KR.

18:41:33 INFO : com.cristoval.web.controller.HomeController >> > info : ko_KR.

18:41:33 WARN : com.cristoval.web.controller.HomeController >> > warn : ko_KR.

18:41:33 ERROR: com.cristoval.web.controller.HomeController >> > error : ko_KR.
```

로그 레벨	설명
FATAL	아주 심각한 에러가 발생한 상태를 나타낸다.
ERROR	어떠한 요청을 처리하는 중 문제가 발생한 상태를 나타낸다.
WARN	프로그램의 실행에는 문제가 없지만, 향후 시스템 에러의 원인이 될 수 있는 경고성 메시지를 나타낸다.
INFO	어떠한 상태변경과 같은 정보성 메시지를 나타낸다.
DEBUG	개발시 디버그 용도로 사용하는 메시지를 나타낸다.
TRACE	디버그 레벨이 너무 광범위한것을 해결하기 위해 좀 더 상세한 이벤트를 나타낸다.