



Spring Framework 시작하기 4

로깅 처리하기



로깅 [logging]

시스템을 작동할 때 시스템의 작동 상태의 기록과 보존, 이용자의 습성 조사 및 시스템 동작의 분석 등을 하기 위해 작동중의 각종 정보를 기록해둘 필요가 있다. 이 기록을 만드는 것을 로깅이라 한다. 즉 로그 시스템의 사용에 관계된 일련의 「사건」을 시간의 경과에 따라 기록하는 것이다.

[네이버 지식백과] [로깅 [logging]](<http://terms.naver.com/entry.nhn?docId=827902>) (컴퓨터인터넷IT용어대사전, 2011. 1. 20., 일진사)

Java Logging Framework

- java.util.logging
- Apache Commons logging
- Log4J

- Logback
- SLF4J(Simple Logging Facade for Java)

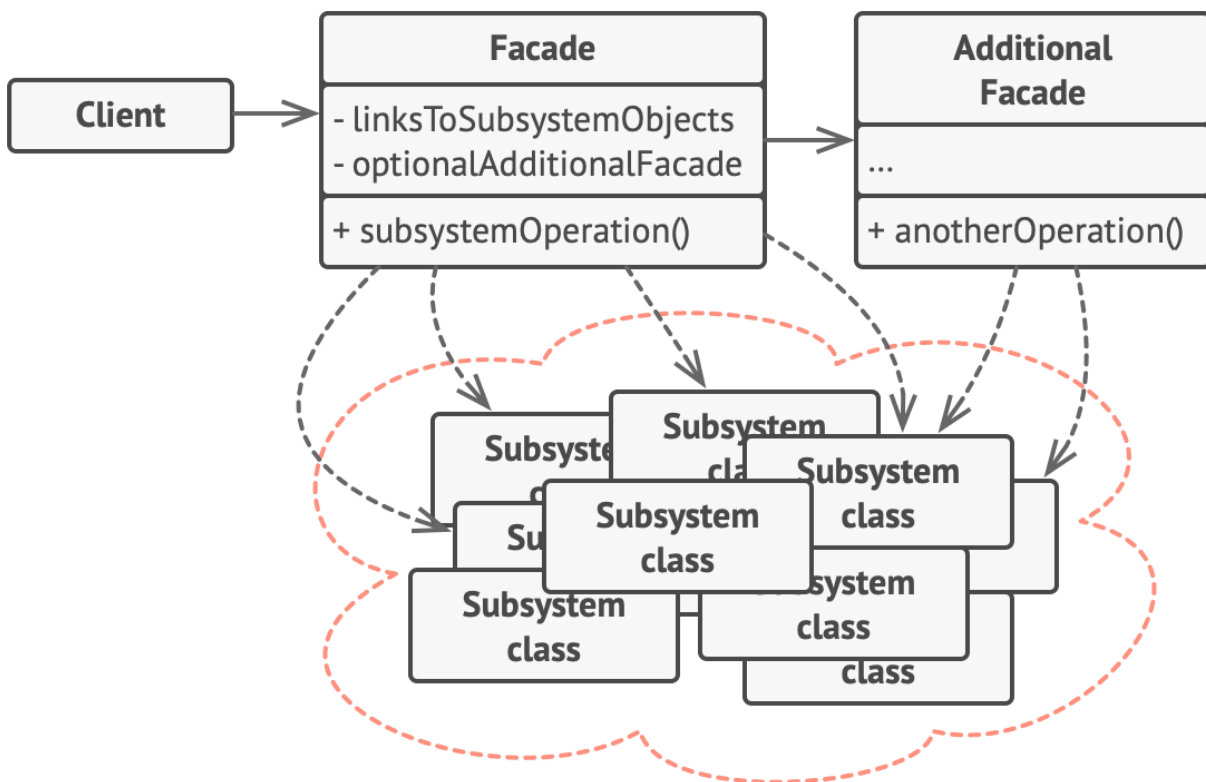
SLF4J

SLF4J(Simple Logging Façade For Java)란 Logging Framework들을 추상화해 놓은 것입니다. Facade Pattern을 이용한 Logging Framework입니다.



페사드란, 프랑스어 Façade 에서 유래된 단어로 건물의 외관이라는 뜻을 가지고 있습니다. 건물의 외벽에서 보면 안의 구조는 보이지 않습니다.

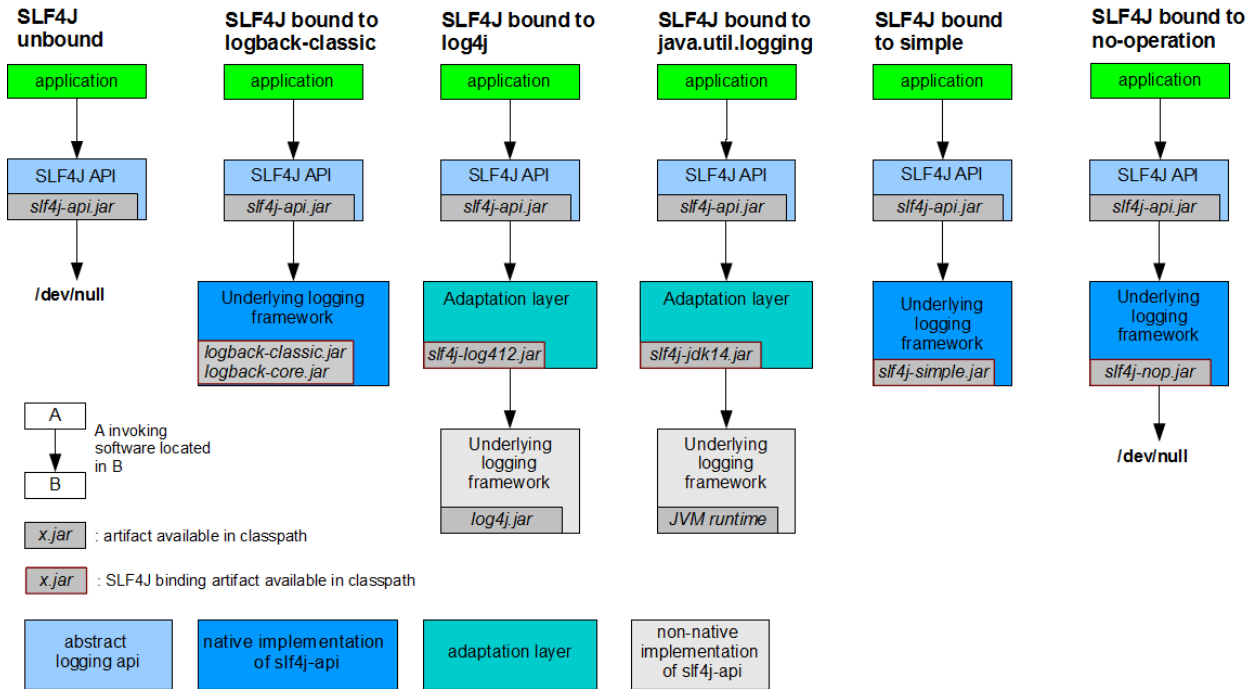
페사드 패턴은 많은 서브시스템(내부 구조)을 거대한 클래스(외벽)로 만들어 감싸서 편리한 인터페이스를 제공해 줍니다.



출처 <https://refactoring.guru/design-patterns/facade>

로깅 프레임워크의 Binding 모듈

SLF4J가 다양한 로깅 프레임워크를 지원하는데 이는 바인딩 모듈을 통해서 처리됩니다. 바인딩 모듈은 로깅 프레임워크를 연결하는 역할을 합니다. 대표적으로 logback-classic(logback), slf4j-log4j12(log4j2) 등등이 있습니다.



<http://www.slf4j.org/manual.html>

Log Level

1. trace
2. debug
3. info
4. warn
5. error

level of request p	effective level q					
	TRACE	DEBUG	INFO	WARN	ERROR	OFF
TRACE	YES	NO	NO	NO	NO	NO
DEBUG	YES	YES	NO	NO	NO	NO
INFO	YES	YES	YES	NO	NO	NO
WARN	YES	YES	YES	YES	NO	NO
ERROR	YES	YES	YES	YES	YES	NO

<http://logback.qos.ch/manual/architecture.html>

Logger

Logger들은 이름 기반으로 생성이 됩니다.



Let's write some code!

logback 설정하기

logback 설정파일 찾는 순서

1. logback-test.xml 파일을 먼저 찾습니다.
2. 없다면 logback.groovy 을 찾습니다.
3. 그래도 없다면 logback.xml을 찾습니다.
4. 모두 없다면 기본 설정 전략을 따릅니다. BasicConfiguration

로그 Appender 설정

대표적으로 ConsoleAppender, FileAppender, RollingFileAppender 등이 존재합니다.

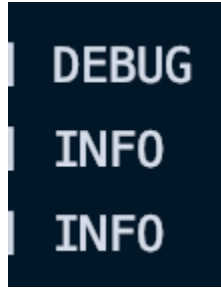
- **ConsoleAppender**
- **FileAppender**
- **RollingFileAppender**

Let's write some code!

PatternLayout

Logback이 기본적으로 제공하는 패턴레이아웃을 이용하면 우리가 로거에서 만든 로깅 이벤트를 가지고 문자열을 변환할 수 있습니다. 이때 문자열로 변환될때 `conversion specifiers` 라는걸 이용합니다.

1. %d - 로깅 이벤트의 날짜를 출력합니다. %date{ISO8601}, %date{HH:mm:ss.SSS} 이렇게 자바에서 제공하는 데이트 타임 패턴을 이용할 수 있습니다.
<https://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html>
2. %logger{length} - Logger name의 이름을 축약할 수 있습니다. {length}는 최대 차릿수입니다.
3. %thread - 현재 Thread name
4. %-5level - log level -5는 출력 고정폭 값
 - a. -를 주면 오른쪽에 스페이스로 주어진 숫자보다 작은 문자열이 오면 공백을 채웁니다
즉 INFO와 같이 4자리면 오른쪽에 1칸 공백 패딩처리가 됩니다.




5. %msg - log message %message은 alias

6. %n - new line

자세한 설명은 아래를 참조합니다.

Chapter 6: Layouts

和訳 (Japanese translation) TCP implementations will follow a general principle of robustness: be conservative in what you do, be liberal in what you accept from others. -JON POSTEL, RFC 793 In

 <http://logback.qos.ch/manual/layouts.html>

127.0.0.1	-	25-10-2009 18:48:14 +0200	GET /testname/700 HTTP/1.1	200	732
127.0.0.1	-	25-10-2009 18:48:15 +0200	POST /testname/ HTTP/1.1	302	0
127.0.0.1	-	25-10-2009 18:48:15 +0200	GET /testname/700 HTTP/1.1	200	734
127.0.0.1	-	25-10-2009 18:48:20 +0200	POST /testname/ HTTP/1.1	302	0
127.0.0.1	-	25-10-2009 18:48:20 +0200	GET /testname/700 HTTP/1.1	200	137
127.0.0.1	-	25-10-2009 18:48:40 +0200	GET /testbackstat.html HTTP/1.1	200	4564
127.0.0.1	-	25-10-2009 18:48:40 +0200	GET /testbackstat.p HTTP/1.1	200	670
127.0.0.1	-	25-10-2009 18:48:40 +0200	GET /testbackstat.p HTTP/1.1	200	47603
127.0.0.1	-	25-10-2009 18:48:40 +0200	GET /testbackstat.p HTTP/1.1	200	7894
127.0.0.1	-	25-10-2009 18:48:40 +0200	GET /testbackstat.p HTTP/1.1	200	4705
127.0.0.1	-	25-10-2009 18:48:40 +0200	GET /testbackstat.p HTTP/1.1	200	3884
127.0.0.1	-	25-10-2009 18:48:40 +0200	GET /testbackstat.co HTTP/1.1	200	688
127.0.0.1	-	25-10-2009 18:48:40 +0200	GET /testbackstat.co HTTP/1.1	200	35
127.0.0.1	-	25-10-2009 18:48:42 +0200	GET /testbackstat.co HTTP/1.1	200	219
127.0.0.1	-	25-10-2009 18:48:42 +0200	POST /testback HTTP/1.1	200	115
127.0.0.1	-	25-10-2009 18:48:42 +0200	POST /testback HTTP/1.1	200	97
127.0.0.1	-	25-10-2009 18:48:43 +0200	GET /testbackstat.co HTTP/1.1	200	35
127.0.0.1	-	25-10-2009 18:48:51 +0200	GET /testbackstat.co HTTP/1.1	200	35
127.0.0.1	-	25-10-2009 18:48:51 +0200	GET /testbackstat.co HTTP/1.1	200	35
127.0.0.1	-	25-10-2009 18:48:51 +0200	GET /testbackstat.co HTTP/1.1	200	35
127.0.0.1	-	25-10-2009 18:48:51 +0200	GET /testbackstat.co HTTP/1.1	200	35
127.0.0.1	-	25-10-2009 18:48:51 +0200	GET /testbackstat.co HTTP/1.1	200	25091
127.0.0.1	-	25-10-2009 18:48:51 +0200	GET /testbackstat.co HTTP/1.1	200	35
127.0.0.1	-	25-10-2009 18:48:51 +0200	GET /testbackstat.co HTTP/1.1	302	0
127.0.0.1	-	25-10-2009 18:48:51 +0200	GET /testbackstat.co HTTP/1.1	200	1411
127.0.0.1	-	25-10-2009 18:48:51 +0200	GET /testbackstat.co HTTP/1.1	200	7364
127.0.0.1	-	25-10-2009 18:48:51 +0200	GET /testbackstat.co HTTP/1.1	302	14465

Conversion

conversionRule 을 이용하면 PatternLayout 에서 본것처럼 d, date, msg와 같은 컨버전 문자를 추가할 수 있습니다.

스프링 부트 이해하기 1

This section dives into the details of Spring Boot. Here you can learn about the key features that you may want to use and customize. If you have not already done so, you might want to read the ["getting-started.html"](#) and ["using.html"](#) sections, so that you have a good grounding of the basics.

The `SpringApplication` class provides a convenient way to bootstrap a Spring application that is started from a `main()` method. In many situations, you can delegate to the static `SpringApplication.run` method, as shown in the following example:

When your application starts, you should see something similar to the following output:

```

2021-02-03 10:33:25.224 INFO 17321 -- [main] o.s.b.d.s.s.SpringApplicationExample : Starting S
2021-02-03 10:33:25.226 INFO 17900 -- [main] o.s.b.d.s.s.SpringApplicationExample : No active f
2021-02-03 10:33:26.046 INFO 17321 -- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat ini
2021-02-03 10:33:26.054 INFO 17900 -- [main] o.apache.catalina.core.StandardService : Starting S
2021-02-03 10:33:26.055 INFO 17900 -- [main] org.apache.catalina.core.StandardEngine : Starting S
2021-02-03 10:33:26.097 INFO 17900 -- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializ
2021-02-03 10:33:26.097 INFO 17900 -- [main] w.s.c.ServletWebServerApplicationContext : Root WebApp
2021-02-03 10:33:26.144 INFO 17900 -- [main] s.tomcat.SampleTomcatApplication : ServletCont
2021-02-03 10:33:26.376 INFO 17900 -- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat star
2021-02-03 10:33:26.384 INFO 17900 -- [main] o.s.b.d.s.s.SpringApplicationExample : Started San

```

By default, `TN50` logging messages are shown, including some relevant startup details, such as the user that launched the application. If you

- spring-boot-starter (Starter POMs)
- SpringApplication를 통한 손쉬운 실행
- Auto Configuration
- 쉬운 외부 환경 설정 - Properties, YAML, Command line 설정 등
- 프로파일을 통한 실행환경 관리
- Packaging Executable Jar
- Developer Tools



Let's write some code!

SpringBoot Banner 제너레이터

Text to ASCII Art Generator (TAAG)

Main Controls - *FIGlet and AOL Macro Fonts Supported*

<https://patorjk.com/software/taag/#p=display&f=Graffiti&t=Type%20Something%20>

Spring Boot Logging 기능

5.4. Logging

Spring Boot uses [Commons Logging](#) for all internal logging but leaves the underlying log implementation open. Default configurations are provided for [Java Util Logging](#), [Log4J2](#), and [Logback](#). In each case, loggers are pre-configured to use console output with optional file output also available.

By default, if you use the “Starters”, Logback is used for logging. Appropriate Logback routing is also included to ensure that dependent libraries that use Java Util Logging, Commons Logging, Log4J, or [SLF4J](#) all work correctly.

Tip

There are a lot of logging frameworks available for Java. Do not worry if the above list seems confusing. Generally, you do not need to change your logging dependencies and the Spring Boot defaults work just fine.

Tip

When you deploy your application to a servlet container or application server, logging performed via the Java Util Logging API is not routed into your application's logs. This prevents logging performed by the container or other applications that have been deployed to it from appearing in your application's logs.

<https://docs.spring.io/spring-boot/docs/2.5.1/reference/htmlsingle/#features.logging>

스프링부트를 사용하면 기본적으로 Logback이 사용되고 SLF4J가 이용이 가능합니다.

Log Level 설정하기

Properties **Yaml**

```
logging.level.org.springframework.web=debug
logging.level.org.hibernate=error
```

PROPERTIES

You can also set the location of a file to which to write the log (in addition to the console) by using

```
logging.file.name .
```

To configure the more fine-grained settings of a logging system, you need to use the native configuration format supported by the `LoggingSystem` in question. By default, Spring Boot picks up the native configuration from its default location for the system (such as `classpath:logback.xml` for Logback), but you can set the location of the config file by using the `logging.config` property.

<https://docs.spring.io/spring-boot/docs/2.5.1/reference/htmlsingle/#howto.logging>

Spring Boot에서 외부에서 설정 가져오기

24. Externalized Configuration

Spring Boot lets you externalize your configuration so that you can work with the same application code in different environments. You can use properties files, YAML files, environment variables, and command-line arguments to externalize configuration. Property values can be injected directly into your beans by using the `@Value` annotation, accessed through Spring's `Environment` abstraction, or be bound to structured objects through `@ConfigurationProperties`.

Spring Boot uses a very particular `PropertySource` order that is designed to allow sensible overriding of values. Properties are considered in the following order:

1. Devtools global settings properties on your home directory (`~/.spring-boot-devtools.properties` when devtools is active).
2. `@TestPropertySource` annotations on your tests.
3. `properties` attribute on your tests. Available on `@SpringBootTest` and the [test annotations for testing a particular slice of your application](#).
4. Command line arguments.
5. Properties from `SPRING_APPLICATION_JSON` (inline JSON embedded in an environment variable or system property).
6. `ServletConfig` init parameters.
7. `ServletContext` init parameters.
8. JNDI attributes from `java:comp/env`.
9. Java System properties (`System.getProperties()`).
10. OS environment variables.
11. A `RandomValuePropertySource` that has properties only in `random.*`.
12. [Profile-specific application properties](#) outside of your packaged jar (`application-{profile}.properties` and YAML variants).
13. [Profile-specific application properties](#) packaged inside your jar (`application-{profile}.properties` and YAML variants).
14. Application properties outside of your packaged jar (`application.properties` and YAML variants).
15. Application properties packaged inside your jar (`application.properties` and YAML variants).
16. `@PropertySource` annotations on your `@Configuration` classes.
17. Default properties (specified by setting `SpringApplication.setDefaultProperties`).

<https://docs.spring.io/spring-boot/docs/2.1.8.RELEASE/reference/html/boot-features-external-config.html>

▼ 출처: 번역 <https://www.latera.kr/reference/java/2019-09-29-spring-boot-config-externalize/>

1. 홈 디렉터리(개발 도구가 활성화된 경우 `~/.spring-boot-devtools.properties`)의 개발 도구 전역 설정 프로퍼티
2. 테스트의 `@TestPropertySource` 어노테이션.
3. 테스트의 `properties` 애트리뷰트. `@SpringBootTest` 와 애플리케이션의 특정 부분을 테스트하기 위한 테스트 어노테이션에서 사용 가능.
4. 커맨드 라인 인자.
5. `SPRING_APPLICATION_JSON` 의 프로퍼티(환경 변수나 시스템 프로퍼티에 삽입된 인라인 JSON).
6. `ServletConfig` 초기 파라미터.
7. `ServletContext` 초기 파라미터.
8. `java:comp/env` 의 JNDI 애트리뷰트.

9. Java 시스템 프로퍼티(`System.getProperties()`).
10. OS 환경 변수
11. `random.*` 에 프로퍼티를 가진 `RandomValuePropertySource` .
12. 패키징된 jar 외부의 프로파일 지정 애플리케이션 프로퍼티(`application-{profile}.properties` 와 YAML 형식).
13. 패키징된 jar 내부의 프로파일 지정 애플리케이션 프로퍼티(`application-{profile}.properties` 와 YAML 형식).
14. 패키징된 jar 외부의 애플리케이션 프로퍼티(`application-{profile}.properties` 와 YAML 형식).
15. 패키징된 jar 내부의 애플리케이션 프로퍼티(`application-{profile}.properties` 와 YAML 형식).
16. `@Configuration` 클래스의 `@PropertySource` 어노테이션
17. (`SpringApplication.setDefaultProperties` 에 의해 명시된) 기본 프로퍼티.

실행 가능한 jar파일 생성하기

Spring Boot의 Maven 플러그인을 이용하면 jar / war 패키지를 손쉽게 할 수 있습니다.