# 스프링 부트와 로깅

백기선
whiteship2000@gmail.com

# 스프링 부트의
# 로깅 의존성 관리

# 스프링 부트와 로깅

- 스프링 부트는 JCL을 사용하여 로깅 코드를 작성한다.
- 스프링 부트 애플리케이션은 SLF4J를 사용한다.
- 스프링 부트 애플리케이션은 Logback을 사용한다.
- 원한다면 얼마든지 다른 로거를 사용할 수 있다.

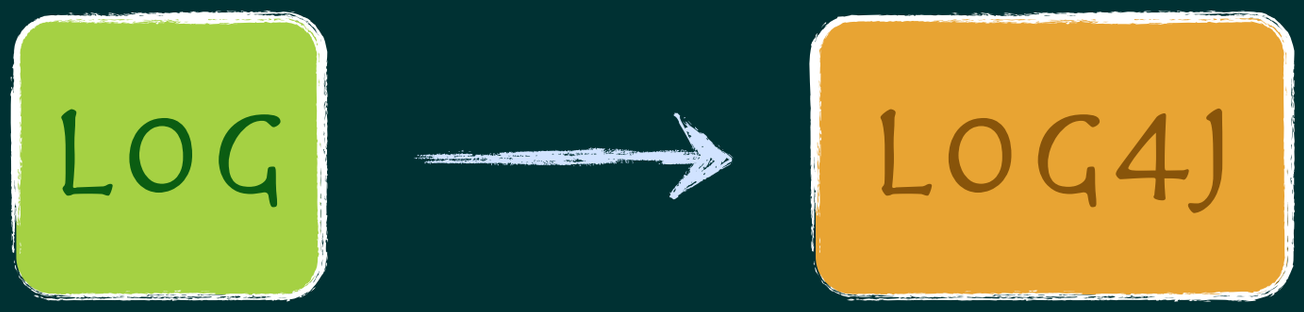
# 스프링 부트 소스 코드

```
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

...

private final Log log = LogFactory.getLog(getClass());

...

if (this.log.isDebugEnabled()) {
   this.log.debug("Loading source "
         + StringUtils.arrayToCommaDelimitedString(sources));
}
```

아파치(자카르타) 커먼스 로깅을 줄여서 JCL이라고 부릅니다.
http://commons.apache.org/proper/commons-logging/

# JCL (commons-logging)

- 로깅 라이브러리가 아니라 로깅 추상화 라이브러리다.
- 로깅 라이브러리 선택권은 애플리케이션 개발자의 것이다.
- 따라서 라이브러리나 프레임워크는 주로 로깅 추상화 라이브러리를 사용한다.



# JCL이 로깅 구현체를 찾는 방법

- 설정 파일에서 찾기
- 애플리케이션 **클래스패스에서** Log4J 구현체 찾아보기
- 애플리케이션이 JDK 1.4에서 구동중인지 확인하기
- 아무것도 못찾으면 기본 구현체 사용

로깅 구현체 찾는 방법
http://commons.apache.org/proper/commons-logging/guide.html#Configuration

# JCL을 꺼리는 이유

- 자바 클래스로더의 기본 동작 방식
- 서블릿 컨테이너의 클래스로더 동작 방식
- JCL은 클래스로더에 의존적인 방법으로 구현체를 찾는다.

JCL 사용시 겪을 수 있는 클래스로더 문제
http://articles.qos.ch/classloader.html

JCL 사용시 겪을 수 있는 메모리 누수 문제
http://wiki.apache.org/commons/Logging/UndeployMemoryLeak

스프링 개발자 Dave Syer, '과거로 돌아갈 수 있다면…'
http://spring.io/blog/2009/12/04/logging-dependencies-in-spring/

# SLF4J

- Simple Logging Facade For Java
- 로깅 라이브러리를 런타임이 아닌 **컴파일 타임**에 정한다.
- 세가지 모듈(Bridging, API, Binding) 제공

# SLF4J API

- 로깅 인터페이스
  - slf4j-api-{version}.jar
  - org.slf4j.Logger
  - org.slf4j.LoggerFactory

# SLF4J API

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

...

Logger logger = LoggerFactory.getLogger(getClass());
logger.info("Loading source {}“, source);

...
```

SLF4J API FAQ

http://www.slf4j.org/faq.html#string_or_object

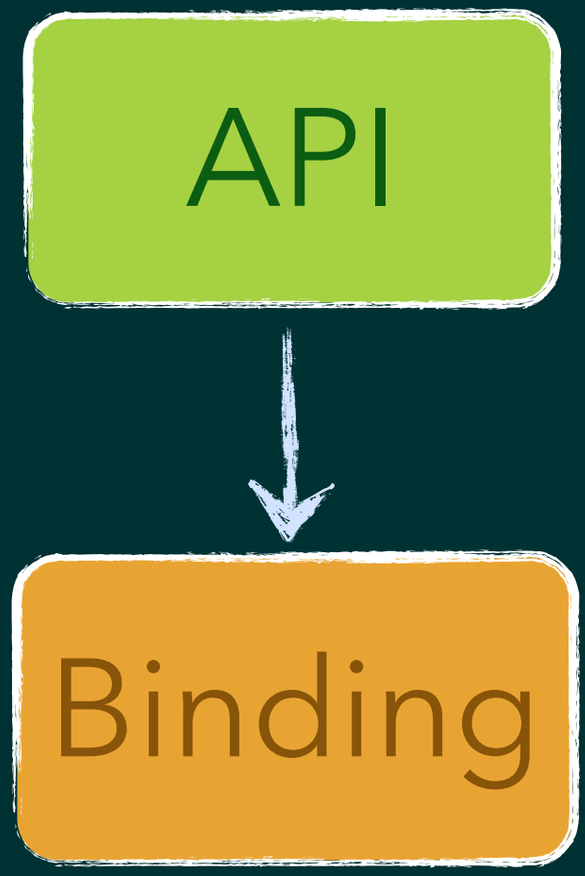
# SLF4J Binding

- SLF4J 인터페이스를 로깅 구현체로 연결
- 어댑터 역할.
- SLF4J 인터페이스를 직접 구현한 구현체도 있다.


- 라이브러리나 프레임워크는 사용하지 마세요.

# SLF4J Binding

- 여러 바인딩 중 반드시 한개만 사용할 것
  - slf4j-log4j12-{version}.jar
  - slf4j-jdk14-{version}.jar
  - slf4j-nop-{version}.jar
  - slf4j-jcl-{version}.jar
  - logback-classic-{logback-version}.jar



# SLF4J Binding

- 여러 바인딩을 클래스패스 두면?

```
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/Users/naver/.m2/
repository/ch/qos/logback/logback-classic/1.1.2/logback-
classic-1.1.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/Users/naver/.m2/
repository/org/slf4j/slf4j-jdk14/1.7.8/slf4j-
jdk14-1.7.8.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings
for an explanation.
SLF4J: Actual binding is of type
[ch.qos.logback.classic.util.ContextSelectorStaticBinder]
```

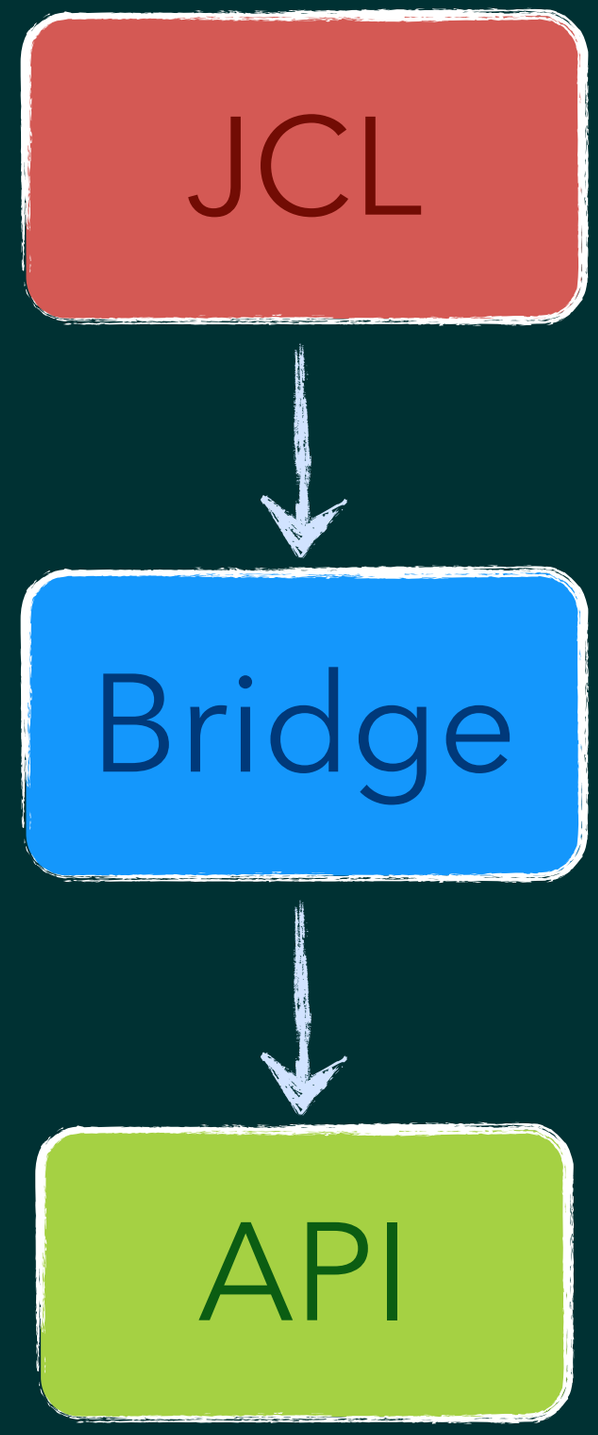
http://www.slf4j.org/codes.html#multiple_bindings

# SLF4J Bridge

- 로거 호출을 SLF4J 인터페이스로 연결
  - 예) Log4J -> SLF4J API
- 어떤 컴포넌트가 구체적인 로거에 의존하면서 고쳐줄 기미가 보이지 않는다면…
- 해당 로거 호출을 대신 받아서 SLF4J API를 호출해준다.

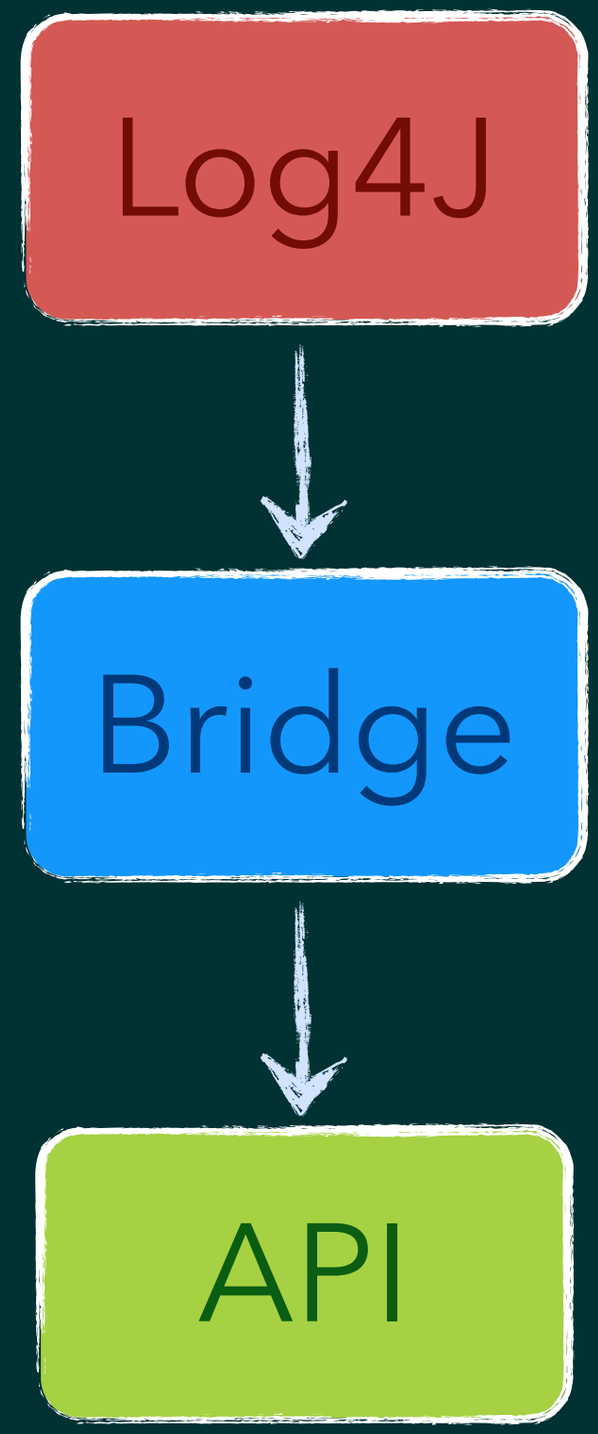
# JCL 호출을 SLF4J로 연결하기

- 의존성에 jcl-over-slf4j.jar 추가
  - JCL 호출을 받아서 SLF4J API 호출
  - JCL 인터페이스를 구현하고 있다.
- 의존성에서 commons-logging.jar 제거.
- slf4j-jcl.jar랑 같이 쓰면 안돼요.
  - 무한 루프



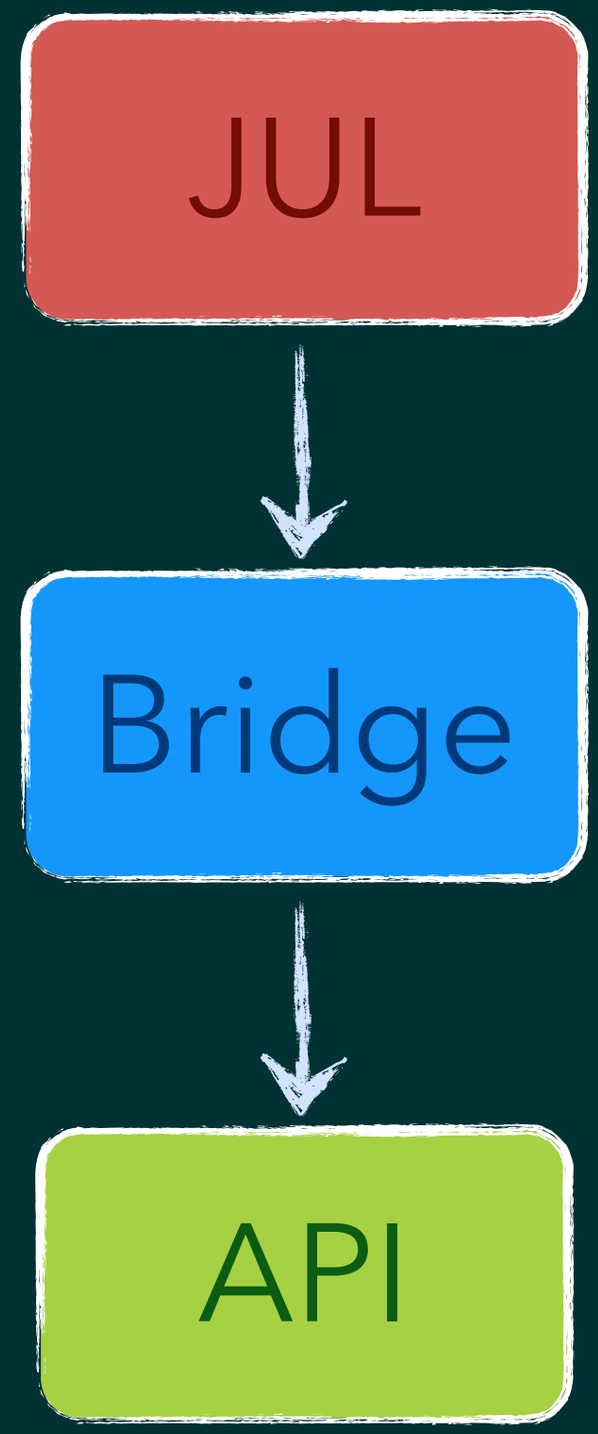
# Log4J 호출을 SLF4J로 연결하기

- log4j-over-slf4j.jar 추가
  - Log4J 호출을 받아서 SLF4J API 호출
  - Log4J 인터페이스를 구현하고 있다.
- 의존성에서 log4j.jar 제거.
- slf4j-log4j12.jar랑 같이 쓰면 안돼요.
  - 무한 루프

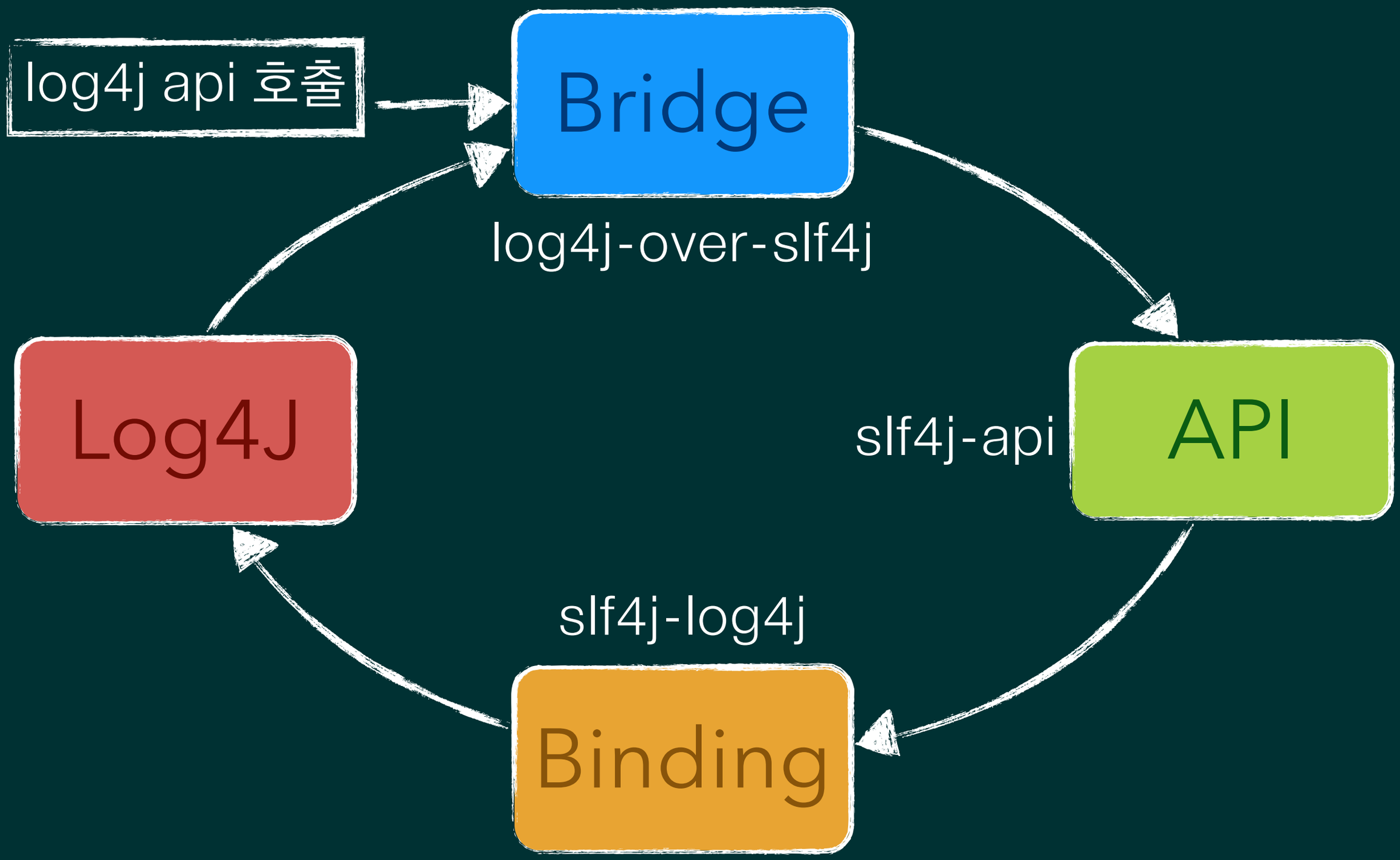


# JUL 호출을 SLF4J로 연결하기

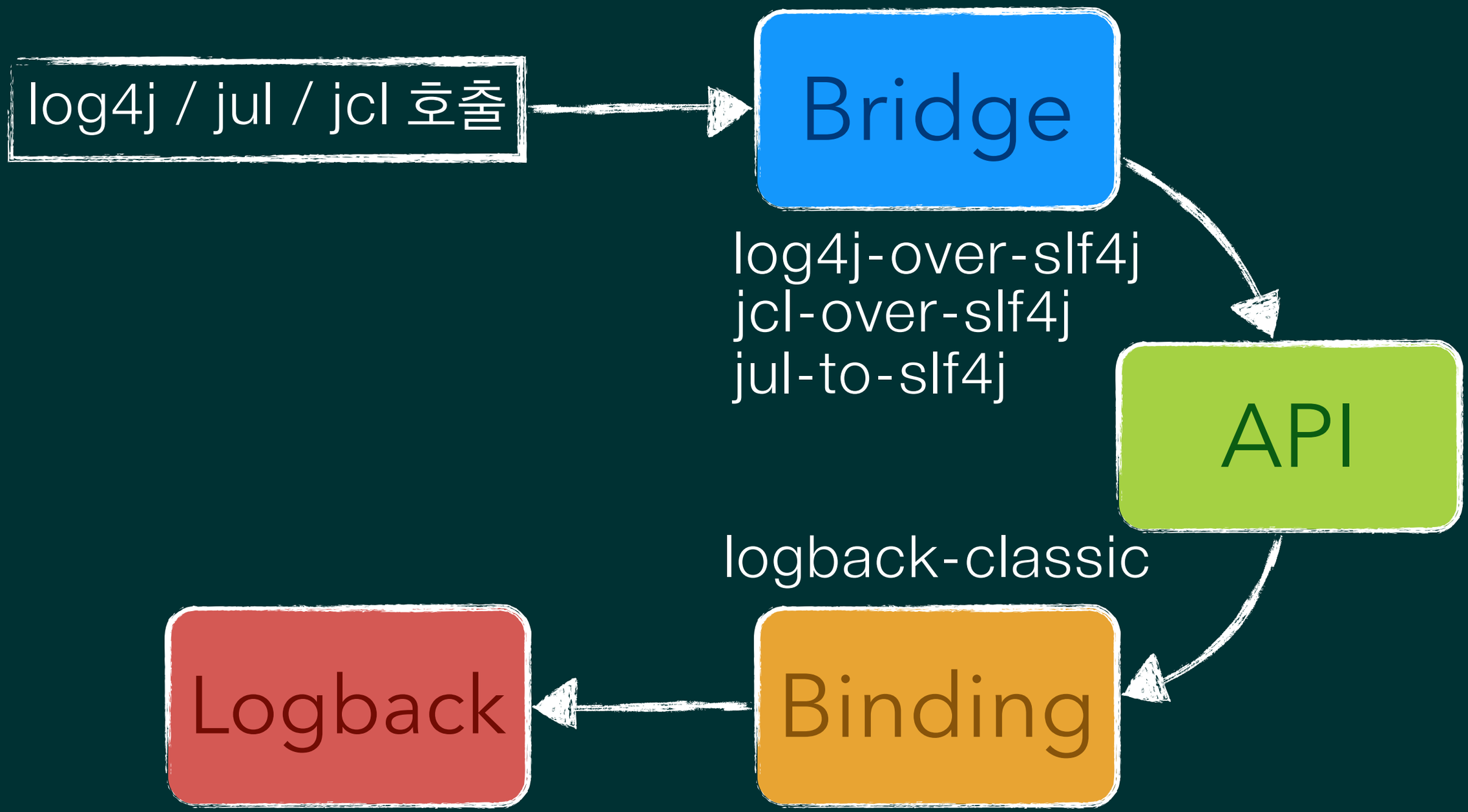
- jul-to-slf4j.jar 추가
  - java.util.logging은 교체 불가능
  - LogRecord 객체를 사용해서 위임.
- slf4j-jdk14.jar랑 같이 쓰면 안돼요.
  - 무한 루프



# 동종 브릿지와 바인딩을 같이 쓰면 어떻게 되나?



# 모든 로깅을 Logback으로 하려면?



# 스프링 부트 로깅

- spring-boot-starter-web

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-logging</artifactId>
</dependency>
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-core</artifactId>
   <exclusions>
      <exclusion>
         <groupId>commons-logging</groupId>
         <artifactId>commons-logging</artifactId>
      </exclusion>
   </exclusions>
</dependency>
```

# 스프링 부트 로깅

- spring-boot-starter-logging

```
<dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>jcl-over-slf4j</artifactId>
</dependency>
<dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>jul-to-slf4j</artifactId>
</dependency>
<dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>log4j-over-slf4j</artifactId>
</dependency>
<dependency>
   <groupId>ch.qos.logback</groupId>
   <artifactId>logback-classic</artifactId>
</dependency>
```

# 스프링 부트 로깅

- 스붓 애플리케이션에서 JCL, JUL, Log4J를 사용해도
- JCL, JUL, Log4J용 브릿지가 있으니까
- 모든 로깅 메서드 호출은 SLF4J-API를 호출하게 되고
- Logback용 바인딩에 있으니까
- 결국 Logback을 사용해서 로깅한다.

# 스프링 부트 로깅
# 기본 로깅 포맷

# 스프링 부트 로그 포맷

```
2015-04-09 21:28:24.475  INFO 37267 --- [           main]
me.whiteship.Application                  : Started
Application in 4.178 seconds (JVM running for 5.396)
```

- 날짜와 시간: 밀리초 단위까지 출력하며 이 정보를 가지고 로그를 정렬할 수 있다.
- 로그 레벨: ERROR, WARN, INFO, DEBUG, TRACE 중 하나.
- 프로세스 ID
- ---: 로그 메시지 시작 구분자
- 쓰레드 이름
- 로거 이름: 보통 클래스 이름인데 패키지 이름은 축약해서 출력하기도 한다.
- 로그 메시지

# 스프링 부트 로깅
# 콘솔 출력

# 콘솔 출력

- DEBUG 레벨로 출력하고 싶다면
  - java -jar app.jar --debug
- spring.output.ansi.enabled 설정
  - ANSI를 지원하는 터미널에서는 콘솔에 색을 적용할 수 있다.
  - ALWAYS, DETECT, NEVER 중에 선택해서 사용

# 스프링 부트 로깅
# 파일 출력

# 파일 출력

- logging.file, logging.path 설정을 사용해야 파일 출력을 사용할 수 있다.
- 10M 단위로 로그 파일을 새로 만든다.

# logging 설정

| logging.path | logging.file | 예 | 설명 |
|---|---|---|---|
| 없음 | 없음 | | 콘솔 출력만 사용 |
| 있음 | 없음 | ./logs/ | 현재 디렉토리 아래 logs라는 디렉토리 안에 spring.log라는 파일에 로그를 남긴다. |
| 없음 | 있음 | ./logs/spring-boot-ri.log | 현재 디렉토리 아래 logs라는 디렉토리 안에 spring-boot-ri.log라는 파일에 로그를 남긴다. |
| 있음 | 있음 | | logging.path 설정은 무시하고 logging.file 설정을 사용한다. |

# 스프링 부트 로깅
# 로그 레벨

# 로그 레벨

- logging.level.{패키지} = {로그 레벨}
  - 패키지 별로 로그 레벨을 설정할 수 있다.
- TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF 지원
- 예) 스프링 MVC 관련 로그만 DEBUG 수준으로 보고 다른 로그는 보고 싶지 않다면 다음과 같이 설정할 수 있다.
  - logging.level.org.springframework = OFF
  - logging.level.org.springframework.web = DEBUG

# 스프링 부트 로깅
# 커스텀 로그 설정

# 커스텀 로그 설정

- 직접 만든 로그 설정 파일을 클래스패스 루트에 두거나
- logging.config로 로그 설정 파일 위치를 설정한다.

# 커스텀 로그 설정

| 로깅 시스템 | 설정 파일 |
| --- | --- |
| Logback | logback.xml 또는 logback.groovy |
| Log4j | log4j.properties 또는 log4j.xml |
| Log4j2 | log4j2.xml |
| JDK (Java Util Logging) | logging.properties |

# 스프링 부트 로깅
# 기본 로그 설정 파일

# 기본 로그 설정

- 스프링 부트는 지금까지 살펴본 로깅 기능을 기본 설정으로 제공한다.
- org.springframework.boot.logging  패키지 코드 참고.

# 스프링 부트 로깅
# Log4j(2) 사용하기

# Log4j(2) 사용하기

- spring-boot-starter-logging 의존성 제거
- spring-boot-starter-log4j(2) 의존성 추가

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter</artifactId>
    <exclusions>
        <exclusion>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-logging</
artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-log4j</artifactId>
</dependency>
```

# 참고

- http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-logging.html
- http://docs.spring.io/spring-boot/docs/current/reference/html/howto-logging.html
- http://www.slf4j.org/legacy.html
- http://wiki.apache.org/commons/Logging/UndeployMemoryLeak

감사합니다.