全面掌握JAVA日志体系

概要

1. log4j2漏洞演示
2. JAVA日志体系概述
3. SLF4j 适配方案
4. Spring Boot日志应用

一、JAVA日志体系概述

提问?

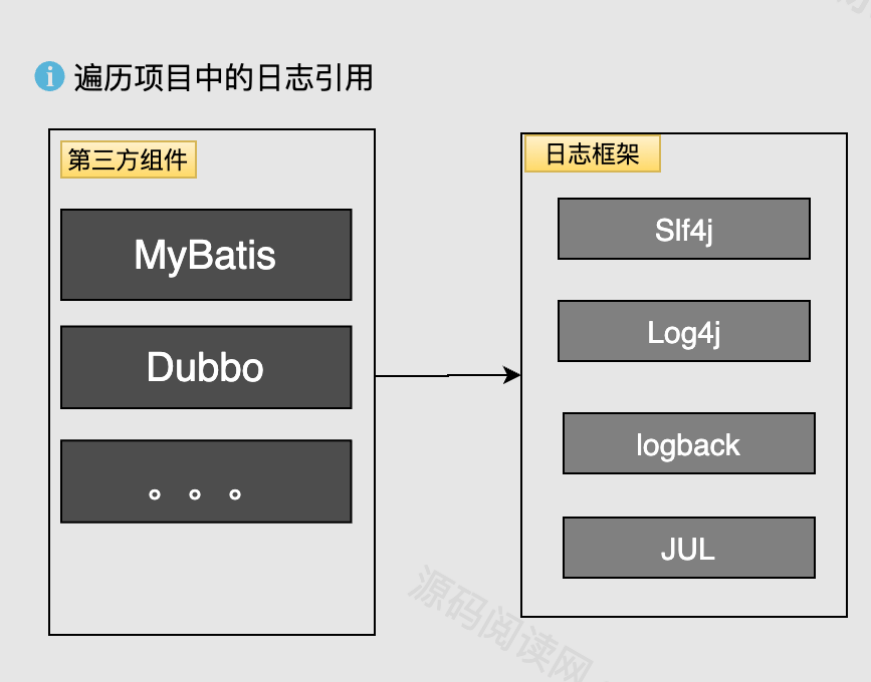
常用的日志框架有哪些?
大家目前正在使用哪些日志组件?

常用日志组件:

名称	JAR包	描述
log4j	log4j-1.2.17.jar	早期常用日志组件
logback	logback-core、logback-classic、logback-access	一套日志组件的实现,性能优于log4j(slf4j阵营)。
log4j2	log4j、log4j-api、log4j-core	apache开发的一款Log4j的升级产品
java.util.logging	jdk(JUL)	Java1.4以来的官方日志实现。无需第三方依懒

日志框架的选择

如果让你开发一个类似Spring 框架，你会采用上述哪个组件?
发现哪个都不能选，只能基于应用实际使用的日志组件来。不然就会日志打印会多份。

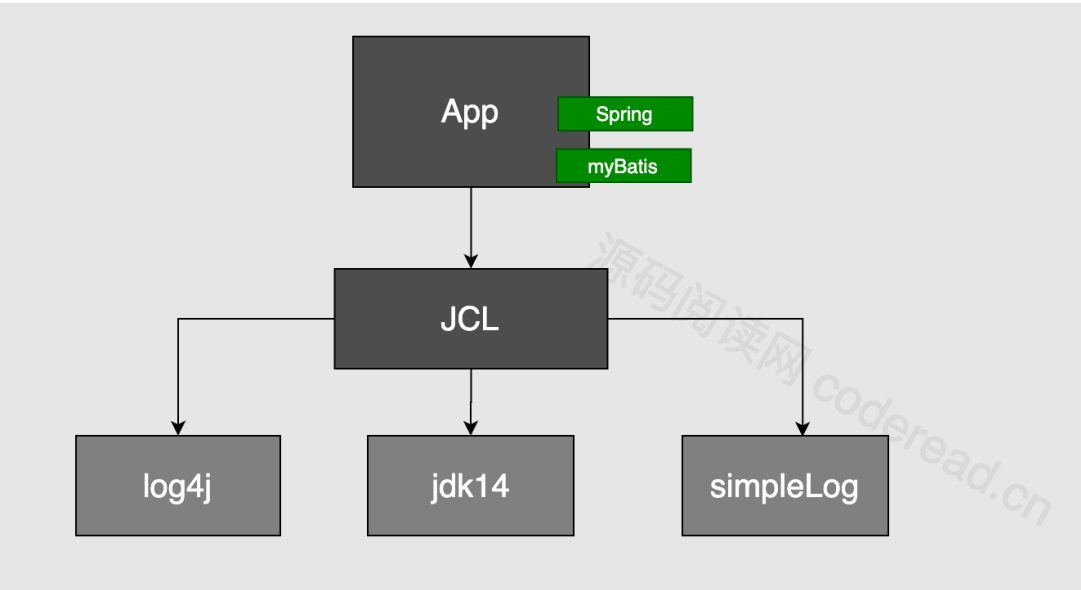


查找应用实际使用的日志组件，并适配打印正是 JCL（Apache Commons Loging）干的事情。

JCL日志

Apache Commons Loging （JCL）

Commons Loging 本身只提供日志接口，具体实现在运行时动态寻找对应组件？比如：log4j、jdk14looger 等。



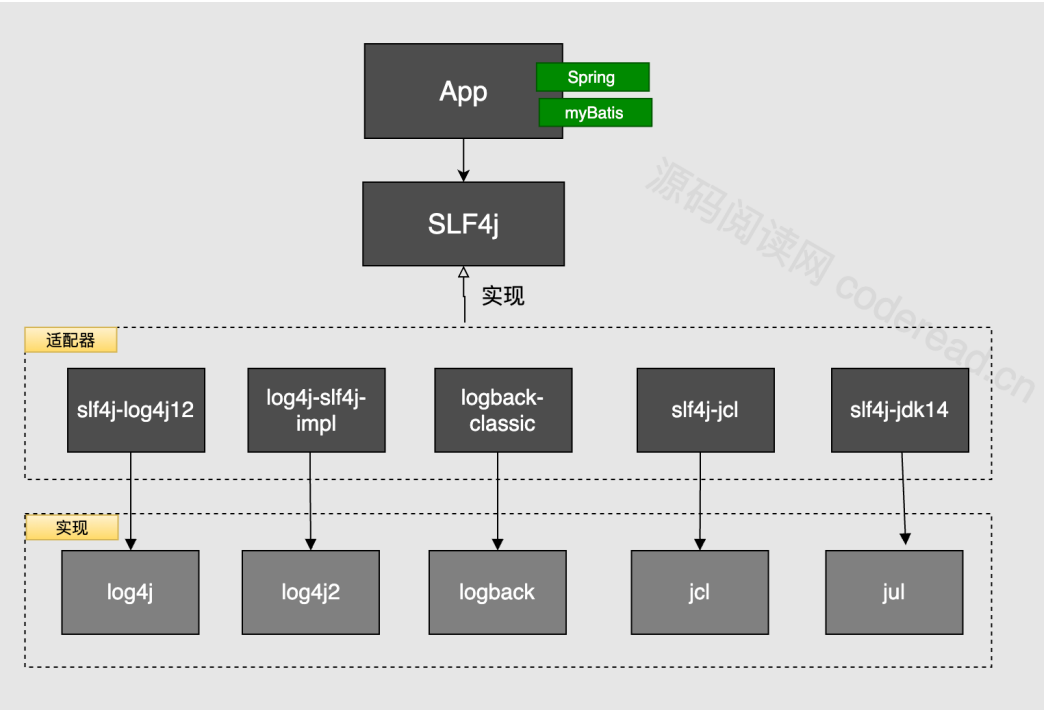
JCL实现逻辑

通过jcl 的源码可看出，jcl为每一种日志实现采用了一个适配器，具体采用哪个 是根据 动态的根据指定顺序查找classPath 是否存在相应的实现。如果一个应用当中有多个classLoader 。比如OSGI 规定了每个模块都有其独立的ClassLoader 。这种机制保证了插件互相独立，同时也限制了JCL在OSGi中的正常使用。这时出现了slf4j 基于静态绑定的方式解决了这个问题。

SLF4j 组成方案

全称 Simple Logging Facade for Java（简单日志门面），与jcl 类似本身不替供日志具体实现，只对外提供接口或门面。与commons loging 不同的是其采用在classPath 加入适配器ajar包来表示具体采用哪种实现：

- slfj-log4j12.jar (表适配接 log4j)
- slf4j-jdk14.jar(表示适配jdk Looging)
- slf4j-jcl.jar(表适配接jcl)
- log4j-slf4j-impl(表示适配log4j2)
- logback-classic(表示适配logback)




```
<dependency>
     <groupId>commons-logging</groupId>
     <artifactId>commons-logging</artifactId>
     <version>1.2</version>
 </dependency>
 <dependency>
     <groupId>org.slf4j</groupId>
     <artifactId>slf4j-api</artifactId>
     <version>1.7.30</version>
 </dependency>
 <dependency>
     <groupId>com.lmax</groupId>
     <artifactId>disruptor</artifactId>
     <version>3.4.2</version>
 </dependency>
<!-- <dependency>
     <groupId>org.slf4j</groupId>
     <artifactId>slf4j-log4j12</artifactId>
```

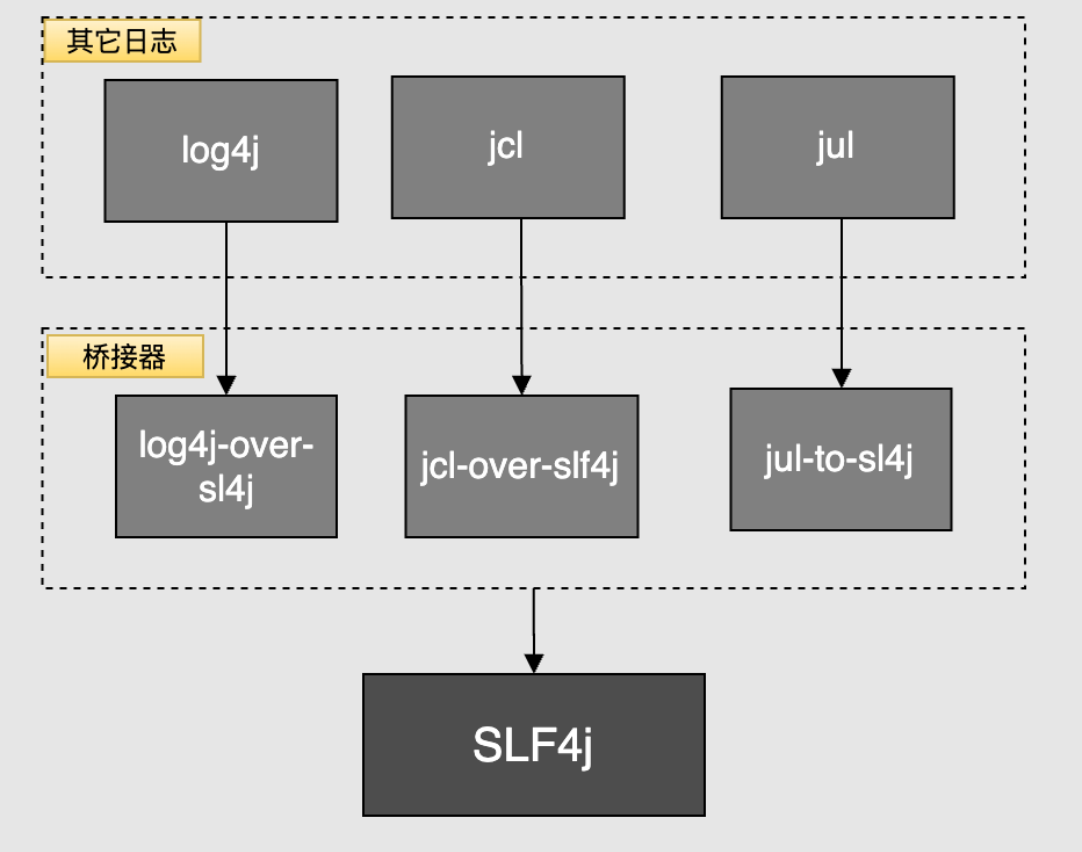
```
      <version>1.7.30</version>
  </dependecy>-->

<!-- <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-jdk14</artifactId>
      <version>1.7.25</version>
  </dependency>-->

  <!--<dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-jcl</artifactId>
      <version>1.7.30</version>
  </dependency>-->

  <dependency>
      <groupId>org.apache.logging.log4j</groupId>
      <artifactId>log4j-slf4j-impl</artifactId>
      <version>2.13.3</version>
  </dependency>
  <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.17</version>
  </dependency>
```

桥接方案

假设你们系统当中之前在用 JCL 打印日志，但这时想加入slf4j来打印日志，就会出现两类日志输出如何解决?

只要classPath 当中指定 sfl4j 适配器 包即可无缝将 原日志输出转移动slf4j上来。

- jcl-over-slf4j ： 转移jcl 日志至slf4j
- log4j-over-sl4j:转移log4j 日志至 slf4j
- jul-over-sl4j :转移jul 日志至 slf4j



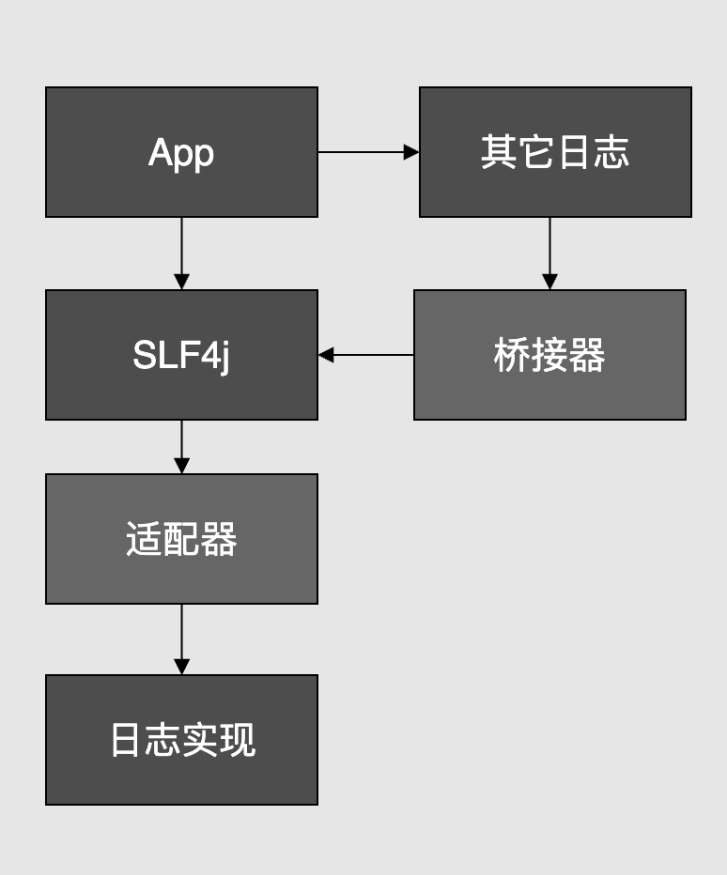

```
<dependency>
    <groupId>commons-logging</groupId>
    <artifactId>commons-logging</artifactId>
    <version>1.2</version>
</dependency>
<!--
jcl 桥接包
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>jcl-over-slf4j</artifactId>
    <version>1.7.28</version>
</dependency>-->
<!-- <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>log4j-over-slf4j</artifactId>
    <version>1.7.25</version>
</dependency>-->
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>jul-to-slf4j</artifactId>
    <version>1.7.30</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.30</version>
</dependency>
```

```
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-slf4j-impl</artifactId>
    <version>2.13.3</version>
</dependency>
<dependency>
    <groupId>com.lmax</groupId>
    <artifactId>disruptor</artifactId>
    <version>3.4.2</version>
</dependency>
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13</version>
</dependency>
```

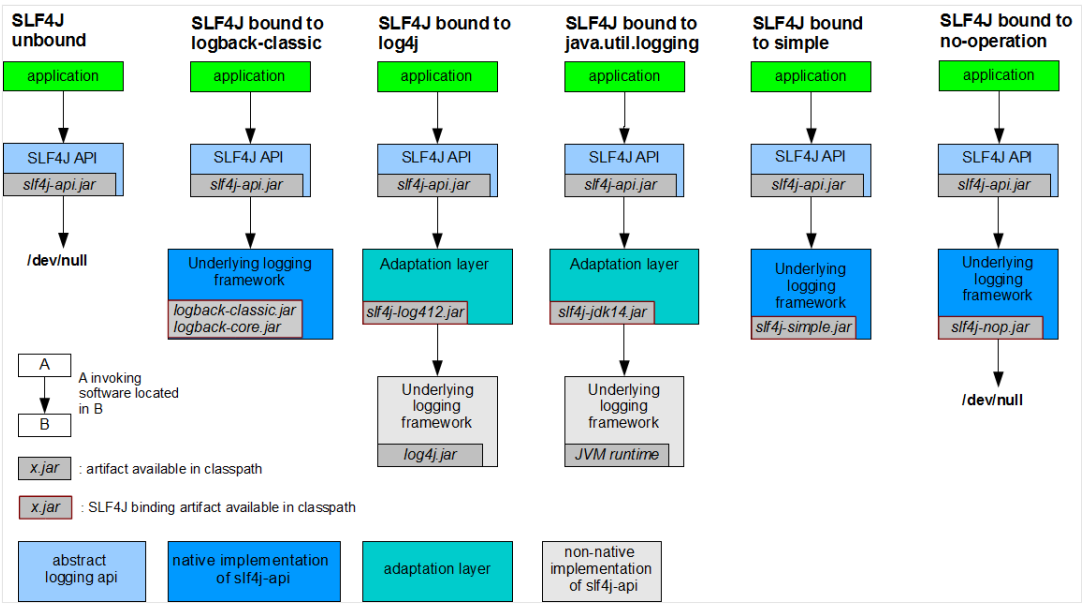
SLF4j各组件汇总

名称	描述	JAR包
门面：	slf4j API接口	slf4j-api.jar
适配器	用于slf4j 连接对应日志实现	slfj-log4j12.jar 、slf4j-jdk14.jar、log4j-slf4j-impl、logback-classic、slf4j-jcl.jar
桥接器	用于将原日志输出无缝转移到slf4j	jcl-over-slf4j、log4j-over-sl4j、jul-over-sl4j
实现	日志的具体实现	log4j、logback、log4j2、java.util.looging

SLF4j各部件结构关系图：

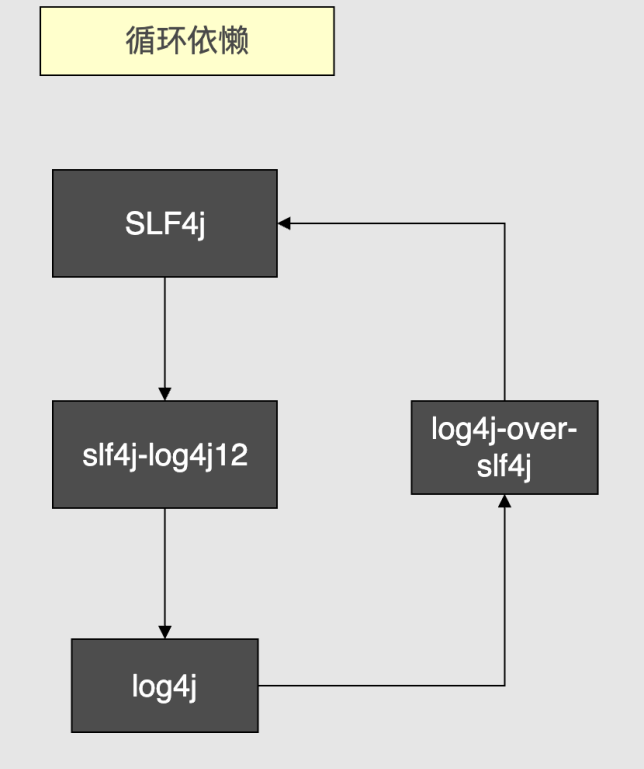


SLF4j 各框架集成方案



循环依懒的问题

log4j 同时桥接又适配到 slf4j 会有什么问题发生?



Spring Boot日志应用

默认日志实现

boot的默认日志实现是logback,它通过spring-boot-starter-logging 引入，可直接在默认配置文件配置。

```
#开启Debug
debug=true
#指定日志目录
logging.file.path=/Users/tommy/git/coderead-spring-boot/
#指定日志文件
logging.file.name=hello.log
#日志输出表达示
logging.pattern.console=%-4relative [%thread] %-5level %logger{30} ==== %msg
#指定包日志级别
logging.level.coderead=debug
#指定整个WEB项目的 日志输出级别
logging.level.web=info
```

具体配置参照官网

也可通过独立的配置文件，进行更精准的配置，默认情况下用lobback的配置文件，可选名称有：
logback.xml
logback-spring.xml

log4j2实现

boot 共有两个日志启动器？另一个是spring-boot-starter-log4j2，只能同时引入一个，所以得排除另一个。

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <exclusions>
      <!--排除默认选择器 -->
        <exclusion>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-logging</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<!-- 引入log4j2-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-log4j2</artifactId>
</dependency>
```

默认配置不变，但是要更精准控制，就需要引入log4j2.xml或log4j2-spring.xml.

其它实现

如果要引入其它日志，请参见Slf4j 集成方案