日志

1、日志框架

一、日志框架

市场上存在非常多的日志框架。JUL（java.util.logging），JCL（Apache Commons Logging），Log4j，Log4j2，Logback、SLF4j、jboss-logging等。Spring Boot在框架内容部使用JCL，spring-boot-starter-logging采用了slf4j+logback的形式，Spring Boot也能自动适配（jul、log4j2、logback）并简化配置

日志门面	日志实现
JCL（Jakarta Commons Logging）	Log4j JUL（java.util.logging）
SLF4j（Simple Logging Facade for Java）	Log4j2 Logback
jboss-logging	

log4j没有logback先进

市面上的日志框架；

JUL、JCL、Jboss-logging、logback、log4j、log4j2、slf4j....

日志门面 （日志的抽象层）	日志实现
~~JCL（Jakarta Commons Logging）~~ SLF4j（Simple Logging Facade for Java） **~~jboss-logging~~**	Log4j JUL（java.util.logging） Log4j2 Logback

左边选一个门面（抽象层）、右边来选一个实现；

日志门面： SLF4J；

日志实现：Logback；

SpringBoot：底层是Spring框架，Spring框架默认使用JCL；

SpringBoot选用SLF4j和logback

2、SLF4j使用

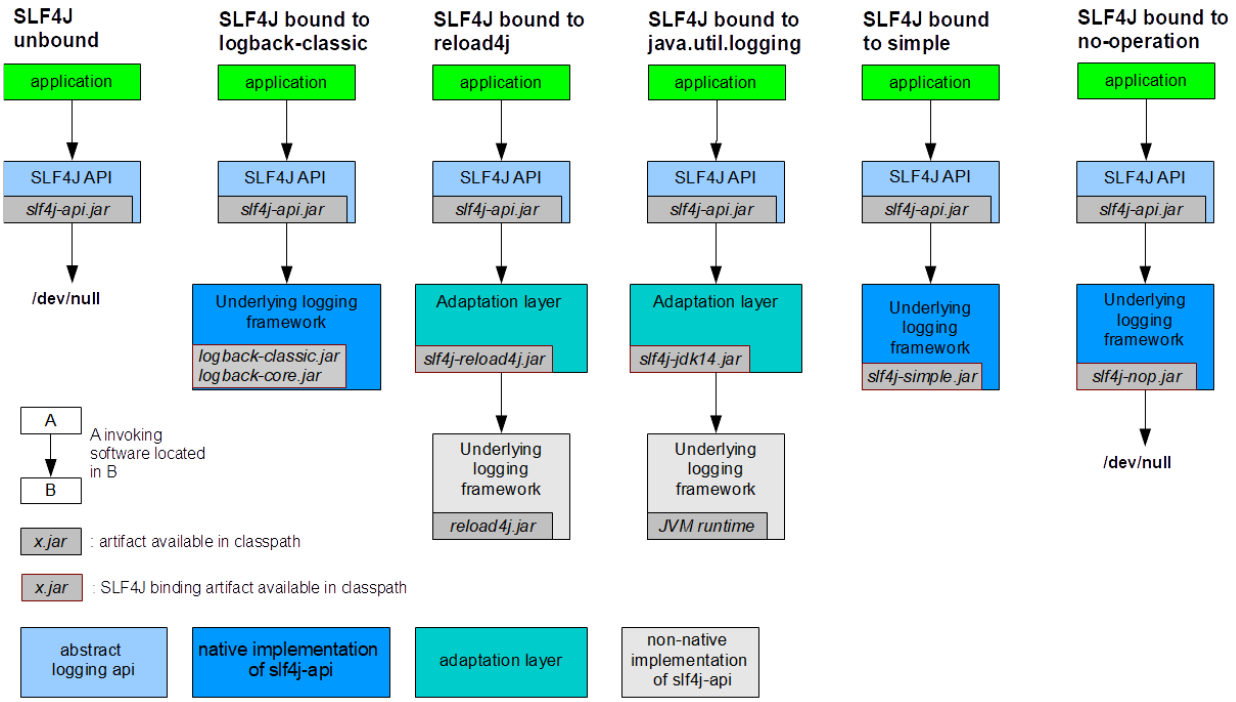
1. 如何在系统中使用SLF4j

以后开发的时候，日志记录方法的调用，不应该直接调用日志的实现类，而是调用日志抽象层里面的方法；

给系统里面导入slf4j的jar和logback的实现jar

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

```
public class HelloWorld {
  public static void main(String[] args) {
    Logger logger = LoggerFactory.getLogger(HelloWorld.class);
    logger.info("Hello World");
  }
}
```

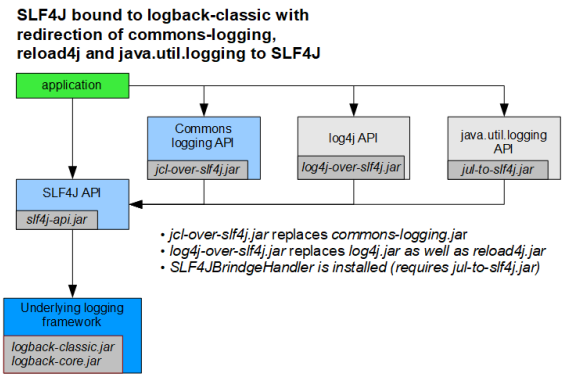


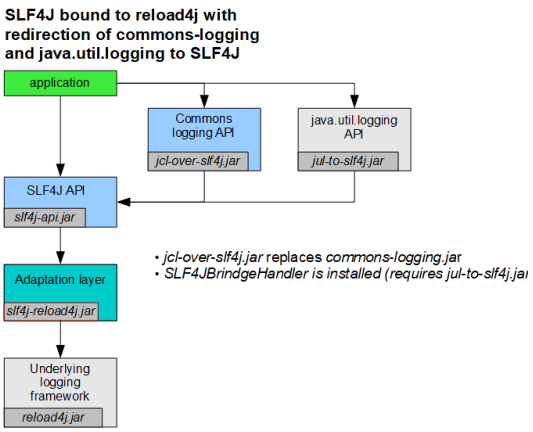
每个日志的实现框架都有自己的配置文件，使用slf4j之后，**配置文件要使用日志实现框架自己本身的配置文件。**

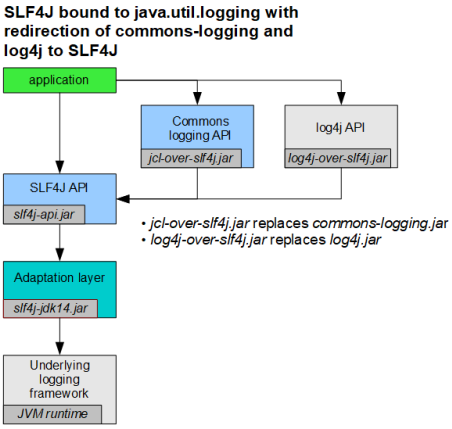
2、遗留问题

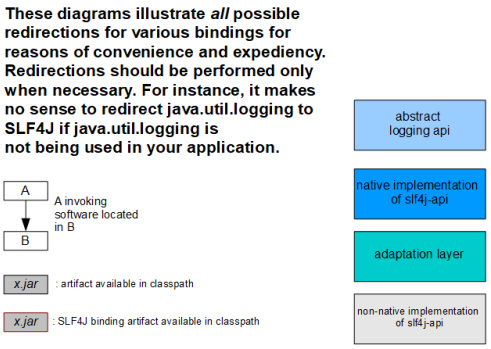
a (slf4j+logback) : Spring (commons-logging) 、Hibernate (jboss-logging) 、MyBatis、xxxx

统一日志记录，即使是别的框架和我一起统一使用slf4j进行输出？









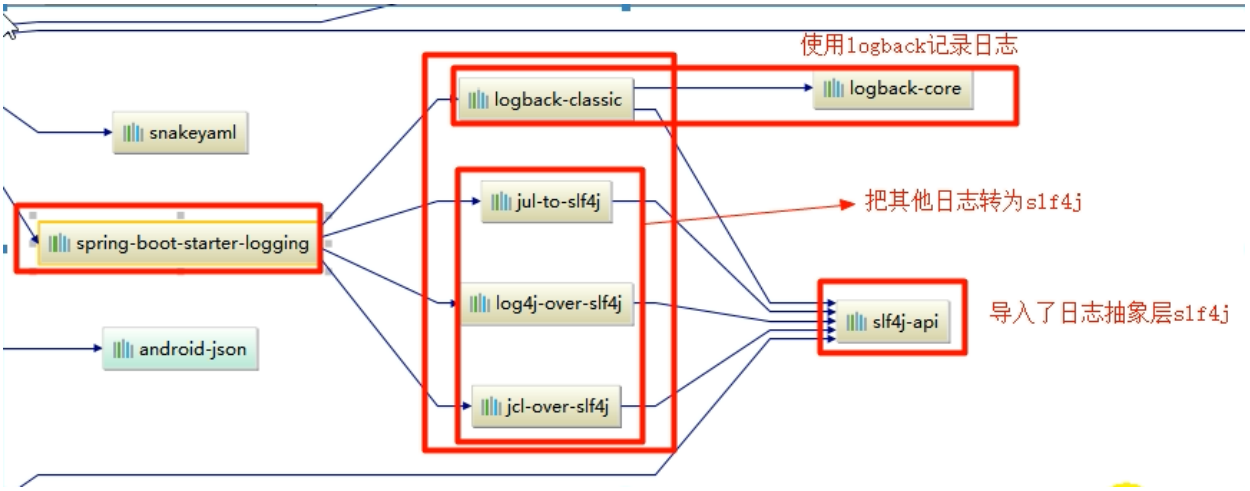
如何让系统中所有的日志都统一到slf4j；

1、将系统中其他日志框架先排除出去；

2、用中间包来替换原有的日志框架；

3、我们导入slf4j其他的实现

3、SpringBoot日志关系



总结：

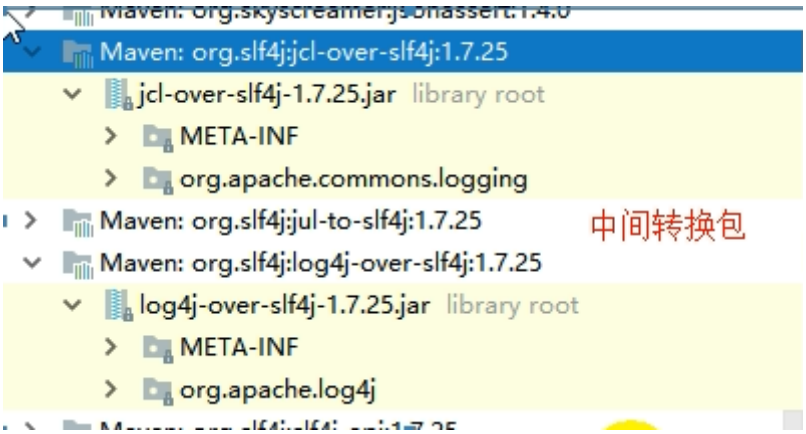
1. SpringBoot底层也是使用slf4j+logback的方式进行日志记录
2. SpringBoot也把其它的日志都替换成了slf4j

3. 中间替换包 log4j-over-slf4j...

```java
@SuppressWarnings("rawtypes")
public abstract class LogFactory {

    static String UNSUPPORTED_OPERATION_IN_JCL_OVER_SLF4J =
"http://www.slf4j.org/codes.html#unsupported_operation_in_jcl_over_slf4j";

    static LogFactory logFactory = new SLF4JLogFactory();
```



4. 如果要引入其他框架，一定要把这个框架的默认日志依赖移除掉，防止jar包冲突

Spring框架用的是commons-logging；

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

==SpringBoot能自动适配所有的日志，而且底层使用slf4j+logback的方式记录日志，引入其他框架的时候，只需要把这个框架依赖的日志框架排除掉；==

logback 读取的配置文件类型

logback会依次读取以下类型配置文件

- logback.groovy
- logback-test.xml
- logback.xml

如果均不存在会采用默认配置

logback组件

- Logger

日志的记录器，把它关联到应用的对应的context上后，主要用于存放日志对象，也可以定义 日志类型、级别。

- Appender

用于指定日志输出的目的地，目的地可以是控制台、文件、数据库等等。

- Layout

负责把事件转换成字符串，格式化的日志信息的输出。在logback中Layout对象被封装在encoder中。

log4j日志的级别

「ALL」: 最低等级的，用于打开所有日志记录。

「TRACE」 : designates finer-grained informational events than the DEBUG.Since:1.2.12，很低的日志级别，一般不会使用。

「DEBUG」: 指出细粒度信息事件对调试应用程序是非常有帮助的，主要用于开发过程中打印一些运行信息。

「INFO」: 消息在粗粒度级别上突出强调应用程序的运行过程。打印一些你感兴趣的或者重要的信息，这个可以用于生产环境中输出程序运行的一些重要信息，但是不能滥用，避免打印过多的日志。

「WARN」: 表明会出现潜在错误的情形，有些信息不是错误信息，但是也要给程序员的一些提示。

「ERROR」: 指出虽然发生错误事件，但仍然不影响系统的继续运行。打印错误和异常信息，如果不想输出太多的日志，可以使用这个级别。

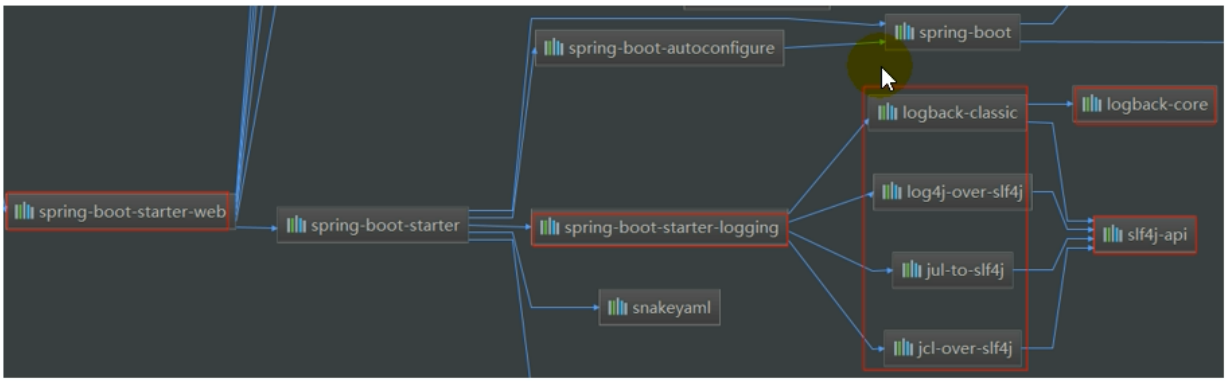
「FATAL」: 指出每个严重的错误事件将会导致应用程序的退出。这个级别比较高了。重大错误，这种级别你可以直接停止程序了。

「OFF」: 最高等级的，用于关闭所有日志记录。

一般只使用4个级别，优先级从高到低为 ERROR > WARN > INFO > DEBUG

在类路径下给出每个每个日志框架自己的配置文件即可，SpringBoot就不会使用默认配置了

Logging System	Customization
Logback	`logback-spring.xml`, `logback-spring.groovy`, `logback.xml` or `logback.groovy`
Log4j2	`log4j2-spring.xml` or `log4j2.xml`
JDK (Java Util Logging)	`logging.properties`



日志使用

1、默认配置

二、默认配置

1、全局常规设置（格式、路径、级别）
2、指定日志配置文件位置
3、切换日志框架
二选一

`spring-boot-starter-log4j2`	Starter for using Log4j2 for logging. An alternative to `spring-boot-starter-logging`
`spring-boot-starter-logging`	Starter for logging using Logback. Default logging starter

logging.file	logging.path	Example	Description
(none)	*(none)*		只在控制台输出
指定文件名	*(none)*	my.log	输出日志到my.log文件
(none)	指定目录	/var/log	输出到指定目录的 spring.log 文件中

```
    日志输出格式：
        %d表示日期时间，
        %thread表示线程名，
        %-5level：级别从左显示5个字符宽度
        %logger{50} 表示logger名字最长50个字符，否则按照句点分割。
        %msg：日志消息，
        %n是换行符
    -->
    %d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n
```

application.properties修改日志的默认配置

```
logging.level.com=debug

# 不指定路径的情况下，默认在项目路径下生成myLog.log日志文件
# 可以指定日志文件的完整路径
logging.file.path=C:/Users/Xiangtai/IdeaProjects/springboot-atguigu/spring-boot-03-logging/log
# 在当前项目所在磁盘的根路径下创建log文件夹，并创建spring.log文件作为默认日志文件
# logging.file.path=/log

# 设置控制台输出的日志格式 %d表示日期时间，%thread表示线程名，%-5level左对齐显示5个字符宽度的日记级别，%msg日志消息，%n换行
logging.pattern.console=%d{yyyy-MM-dd HH:mm} [%thread] %-5level %logger{50} - %msg%n
# 设置文件中日志的输出格式
logging.pattern.file=%d{yyyy-MM-dd} === [%thread] %-5level === %logger{50} === - %msg%n
```

logback.xml：直接就被日志框架识别了；

logback-spring.xml：日志框架就不直接加载日志的配置项，由SpringBoot解析日志配置，可以使用SpringBoot的高级Profile功能

```
<springProfile name="staging">
    <!-- configuration to be enabled when the "staging" profile is active -->
    可以指定某段配置只在某个环境下生效
</springProfile>

```

在主配置文件中配置：spring.profiles.active=dev 让logback-spring.xml文件的<springProfile name="dev">配置生效，如果日志配置文件名命名为logback.xml则不能配置springProfile，则无法根据项目环境切换日志配置。

```
<appender>
   <layout class="ch.qos.logback.classic.PatternLayout">
      <springProfile name="dev">
         <pattern>%d{yyyy-MM-dd} ---> [%thread] %-5level ---> %logger{50} ---> - %msg%n</pattern>
      </springProfile>
      <springProfile name="!dev">
         <pattern>%d{yyyy-MM-dd} === [%thread] %-5level === %logger{50} === - %msg%n</pattern>
      </springProfile>
   </layout>
</appender>
```

切换日志框架

将日志实现切换成log4j（不推荐）

可以按照slf4j的日志适配图，进行相关的切换；

slf4j+log4j的方式；

```xml
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  <exclusions>
    <exclusion>
      <artifactId>logback-classic</artifactId>
      <groupId>ch.qos.logback</groupId>
    </exclusion>
    <exclusion>
      <artifactId>log4j-over-slf4j</artifactId>
      <groupId>org.slf4j</groupId>
    </exclusion>
  </exclusions>
</dependency>

<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-log4j12</artifactId>
</dependency>

```

切换为log4j2

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

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-log4j2</artifactId>
        </dependency>
```

xml