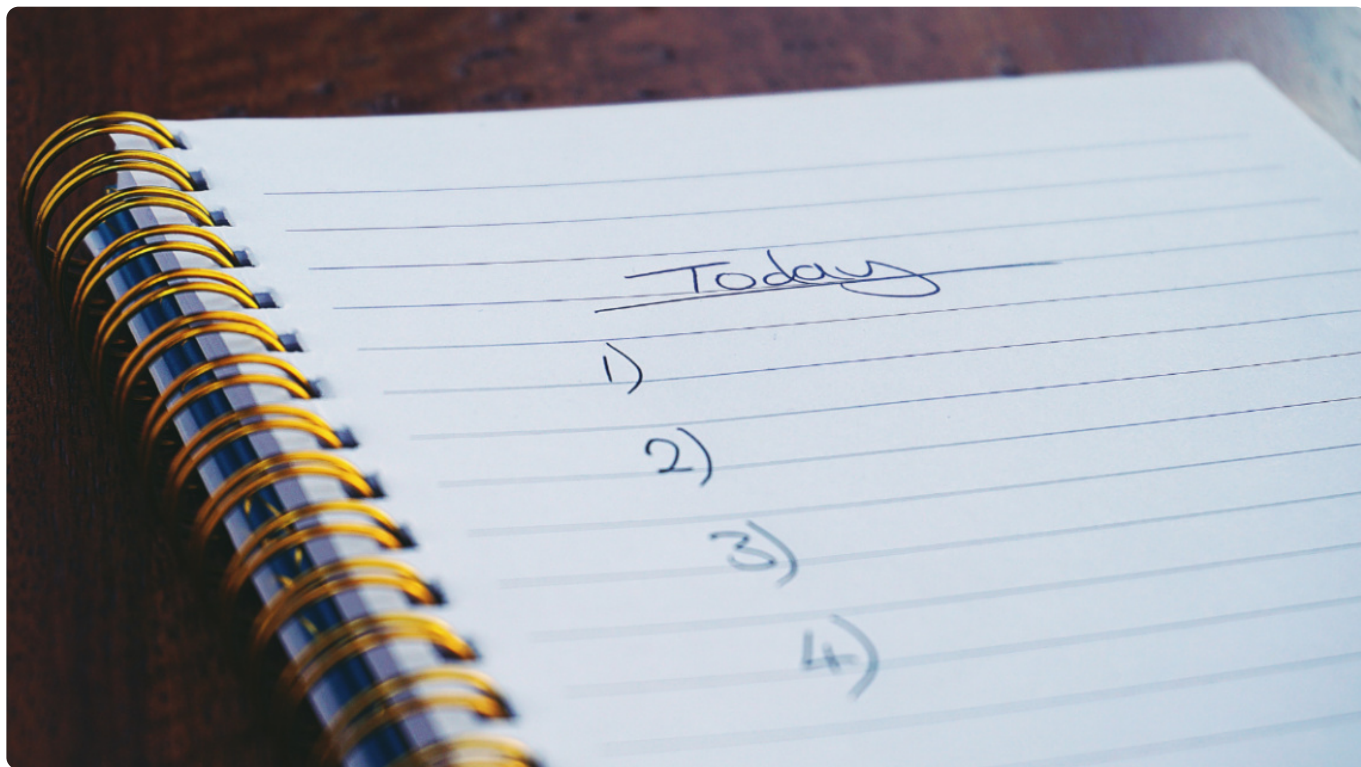


## 31 | Logger组件：Tomcat的日志框架及实战

2019-07-20 李号双

深入拆解Tomcat & Jetty

[进入课程 >](#)



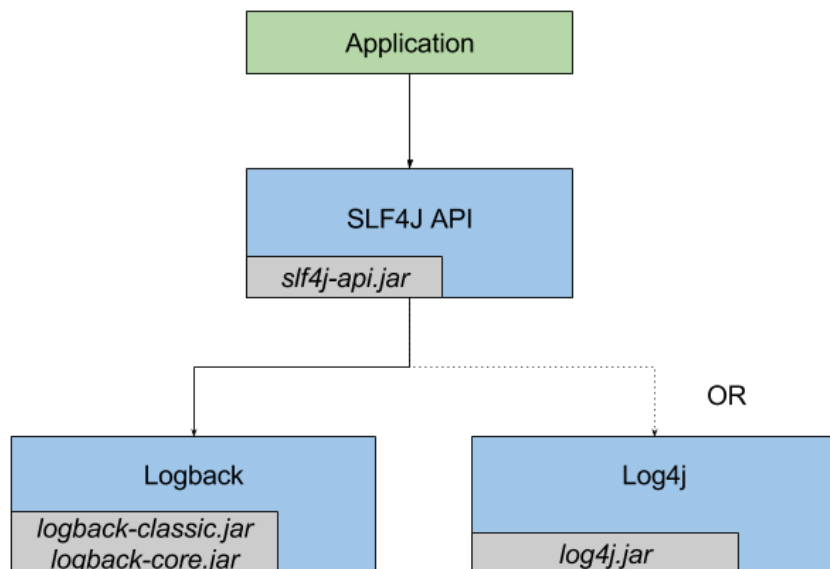
讲述：李号双

时长 07:29 大小 6.87M



每一个系统都有一些通用的模块，比如日志模块、异常处理模块、工具类等，对于 Tomcat 来说，比较重要的通用模块有日志、Session 管理和集群管理。从今天开始我会分三期来介绍通用模块，今天这一期先来讲日志模块。

日志模块作为一个通用的功能，在系统里通常会使用第三方的日志框架。Java 的日志框架有很多，比如：JUL ( Java Util Logging )、Log4j、Logback、Log4j2、Tinylog 等。除此之外，还有 JCL ( Apache Commons Logging ) 和 SLF4J 这样的“门面日志”。下面是 SLF4J 与日志框架 Logback、Log4j 的关系图：



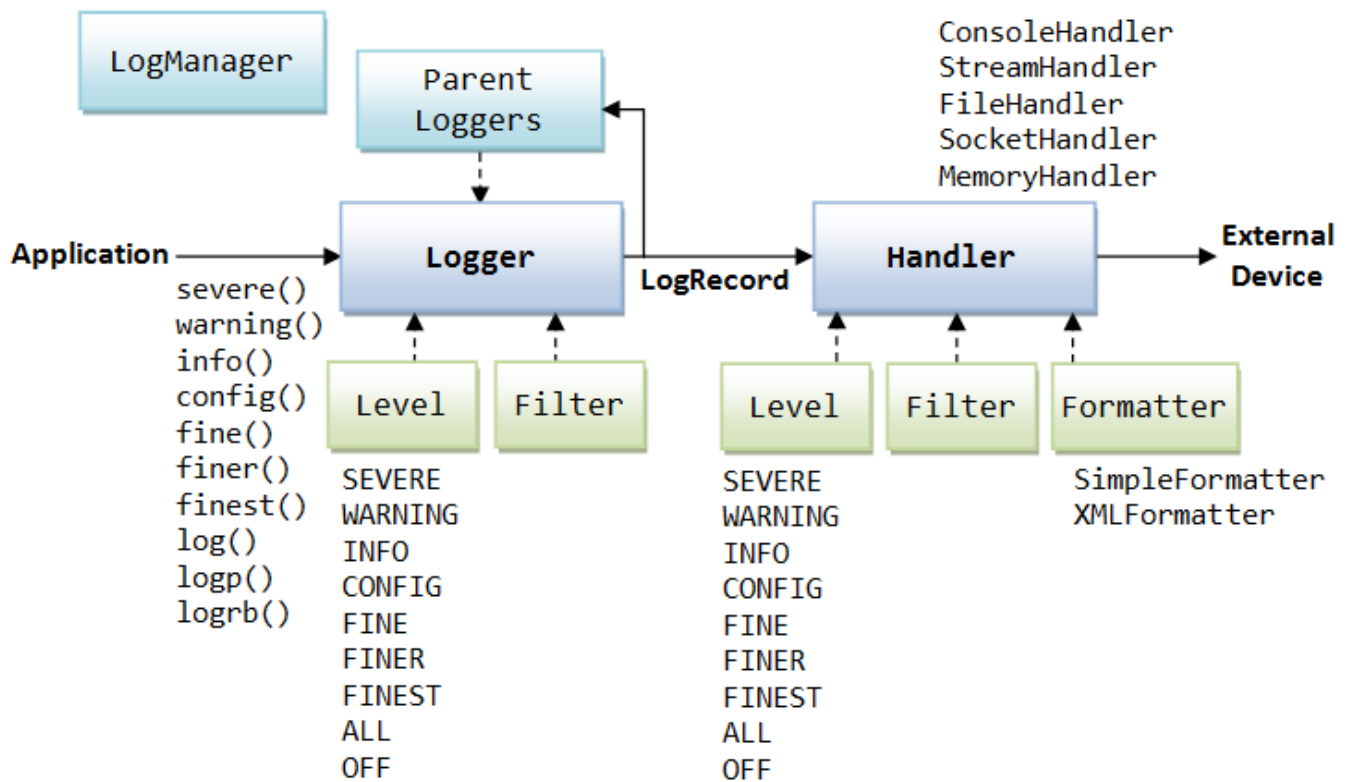
我先来解释一下什么是“门面日志”。“门面日志”利用了设计模式中的门面模式思想，对外提供一套通用的日志记录的 API，而不提供具体的日志输出服务，如果要实现日志输出，需要集成其他的日志框架，比如 Log4j、Logback、Log4j2 等。

这种门面模式的好处在于，记录日志的 API 和日志输出的服务分离开，代码里面只需要关注记录日志的 API，通过 SLF4J 指定的接口记录日志；而日志输出通过引入 JAR 包的方式即可指定其他的日志框架。当我们需要改变系统的日志输出服务时，不用修改代码，只需要改变引入日志输出框架 JAR 包。

今天我们就来看看 Tomcat 的日志模块是如何实现的。默认情况下，Tomcat 使用自身的 JULI 作为 Tomcat 内部的日志处理系统。JULI 的日志门面采用了 JCL；而 JULI 的具体实现是构建在 Java 原生的日志系统 `java.util.logging` 之上的，所以在看 JULI 的日志系统之前，我先简单介绍一下 Java 的日志系统。

## Java 日志系统

Java 的日志包在 `java.util.logging` 路径下，包含了几个比较重要的组件，我们通过一张图来理解一下：



从图上我们看到这样几个重要的组件：

Logger：用来记录日志的类。

Handler：规定了日志的输出方式，如控制台输出、写入文件。

Level：定义了日志的不同等级。

Formatter：将日志信息格式化，比如纯文本、XML。

我们可以通过下面的代码来使用这些组件：

复制代码

```

1 public static void main(String[] args) {
2     Logger logger = Logger.getLogger("com.mycompany.myapp");
3     logger.setLevel(Level.FINE);
4     logger.setUseParentHandlers(false);
5     Handler hd = new ConsoleHandler();
6     hd.setLevel(Level.FINE);
7     logger.addHandler(hd);
8     logger.info("start log");
9 }
  
```

JULI 对日志的处理方式与 Java 自带的基本一致，但是 Tomcat 中可以包含多个应用，而每个应用的日志系统应该相互独立。Java 的原生日志系统是每个 JVM 有一份日志的配置文件，这不符合 Tomcat 多应用的场景，所以 JULI 重新实现了一些日志接口。

## DirectJDKLog

Log 的基础实现类是 DirectJDKLog，这个类相对简单，就包装了一下 Java 的 Logger 类。但是它也在原来的基础上进行了一些修改，比如修改默认的格式化方式。


## LogFactory

Log 使用了工厂模式来向外提供实例，LogFactory 是一个单例，可以通过 SeviceLoader 为 Log 提供自定义的实现版本，如果没有配置，就默认使用 DirectJDKLog。

 复制代码

```
1 private LogFactory() {
2     // 通过 ServiceLoader 尝试加载 Log 的实现类
3     ServiceLoader<Log> logLoader = ServiceLoader.load(Log.class);
4     Constructor<? extends Log> m=null;
5
6     for (Log log: logLoader) {
7         Class<? extends Log> c=log.getClass();
8         try {
9             m=c.getConstructor(String.class);
10            break;
11        }
12        catch (NoSuchMethodException | SecurityException e) {
13            throw new Error(e);
14        }
15    }
16
17    // 如何没有定义 Log 的实现类，discoveredLogConstructor 为 null
18    discoveredLogConstructor = m;
19 }
```

下面的代码是 LogFactory 的 getInstance 方法：

 复制代码


```
1 public Log getInstance(String name) throws LogConfigurationException {
2     // 如果 discoveredLogConstructor 为 null，也就没有定义 Log 类，默认用 DirectJDKLog
```

```
3     if (discoveredLogConstructor == null) {
4         return DirectJDKLog.getInstance(name);
5     }
6
7     try {
8         return discoveredLogConstructor.newInstance(name);
9     } catch (ReflectiveOperationException | IllegalArgumentException e) {
10        throw new LogConfigurationException(e);
11    }
12 }
```

## Handler

在 JULI 中就自定义了两个 Handler : `FileHandler` 和 `AsyncFileHandler`。 `FileHandler` 可以简单地理解为一个在特定位置写文件的工具类，有一些写操作常用的方法，如 `open`、`write(publish)`、`close`、`flush` 等，使用了读写锁。其中的日志信息通过 `Formatter` 来格式化。

`AsyncFileHandler` 继承自 `FileHandler`，实现了异步的写操作。其中缓存存储是通过阻塞双端队列 `LinkedBlockingDeque` 来实现的。当应用要通过这个 Handler 来记录一条消息时，消息会先被存储到队列中，而在后台会有一个专门的线程来处理队列中的消息，取出的消息会通过父类的 `publish` 方法写入相应文件内。这样就可以在大量日志需要写入的时候起到缓冲作用，防止都阻塞在写日志这个动作上。需要注意的是，我们可以为阻塞双端队列设置不同的模式，在不同模式下，对新进入的消息有不同的处理方式，有些模式下会直接丢弃一些日志：

 复制代码

- 1 `OVERFLOW_DROP_LAST`: 丢弃栈顶的元素
- 2 `OVERFLOW_DROP_FIRSH`: 丢弃栈底的元素
- 3 `OVERFLOW_DROP_FLUSH`: 等待一定时间并重试，不会丢失元素
- 4 `OVERFLOW_DROP_CURRENT`: 丢弃放入的元素

## Formatter

`Formatter` 通过一个 `format` 方法将日志记录 `LogRecord` 转化成格式化的字符串，JULI 提供了三个新的 `Formatter`。


OnlineFormatter：基本与 Java 自带的 SimpleFormatter 格式相同，不过把所有内容都写到了一行中。

VerbatimFormatter：只记录了日志信息，没有任何额外的信息。

JdkLoggerFormatter：格式化了一个轻量级的日志信息。

## 日志配置

Tomcat 的日志配置文件为 Tomcat 文件夹下 `conf/logging.properties`。我来拆解一下这个配置文件，首先可以看到各种 Handler 的配置：

 复制代码

```
1 handlers = 1catalina.org.apache.juli.AsyncFileHandler, 2localhost.org.apache.juli.Async
2
3 .handlers = 1catalina.org.apache.juli.AsyncFileHandler, java.util.logging.ConsoleHandle
```

以 `1catalina.org.apache.juli.AsyncFileHandler` 为例，数字是为了区分同一个类的不同实例；`catalina`、`localhost`、`manager` 和 `host-manager` 是 Tomcat 用来区分不同系统日志的标志；后面的字符串表示了 Handler 具体类型，如果要添加 Tomcat 服务器的自定义 Handler，需要在字符串里添加。

接下来是每个 Handler 设置日志等级、目录和文件前缀，自定义的 Handler 也要在这里配置详细信息：

 复制代码

```
1 1catalina.org.apache.juli.AsyncFileHandler.level = FINE
2 1catalina.org.apache.juli.AsyncFileHandler.directory = ${catalina.base}/logs
3 1catalina.org.apache.juli.AsyncFileHandler.prefix = catalina.
4 1catalina.org.apache.juli.AsyncFileHandler.maxDays = 90
5 1catalina.org.apache.juli.AsyncFileHandler.encoding = UTF-8
```

## Tomcat + SLF4J + Logback

在今天文章开头我提到，SLF4J 和 JCL 都是日志门面，那它们有什么区别呢？它们的区别主要体现在日志服务类的绑定机制上。JCL 采用运行时动态绑定的机制，在运行时动态寻找

和加载日志框架实现。

SLF4J 日志输出服务绑定则相对简单很多，在编译时就静态绑定日志框架，只需要提前引入需要的日志框架。另外 Logback 可以说 Log4j 的进化版，在性能和可用性方面都有所提升。你可以参考官网上这篇[文章](#)来了解 Logback 的优势。

基于此我们来实战一下如何将 Tomcat 默认的日志框架切换成为 “SLF4J + Logback” 。具体的步骤是：

1. 根据你的 Tomcat 版本，从[这里](#)下载所需要文件。解压后你会看到一个类似于 Tomcat 目录结构的文件夹。
2. 替换或拷贝下列这些文件到 Tomcat 的安装目录：

bin/tomcat-juli.jar	-->	<Tomcat>/bin/tomcat-juli.jar
bin/setenv.sh	-->	<Tomcat>/bin/setenv.sh
bin/setenv.bat	-->	<Tomcat>/bin/setenv.bat
conf/logback.xml	-->	<Tomcat>/conf/logback.xml
conf/logback-access.xml	-->	<Tomcat>/conf/logback-access.xml
conf/server.xml	-->	<Tomcat>/conf/server.xml
lib/logback-core-1.2.1.jar	-->	<Tomcat>/lib
lib/logback-access-1.2.1.jar	-->	<Tomcat>/lib

3. 删除<Tomcat>/conf/logging.properties

4. 启动 Tomcat

## 本期精华

今天我们谈了日志框架与日志门面的区别，以及 Tomcat 的日志模块是如何实现的。默认情况下，Tomcat 的日志模板叫作 JULI，JULI 的日志门面采用了 JCL，而具体实现是基于 Java 默认的日志框架 Java Util Logging，Tomcat 在 Java Util Logging 基础上进行了改造，使得它自身的日志框架不会影响 Web 应用，并且可以分模板配置日志的输出文件和格式。最后我分享了如何将 Tomcat 的日志模块切换到时下流行的 “SLF4J + Logback”，希望对你有所帮助。

## 课后思考



Tomcat 独立部署时，各种日志都输出到了相应的日志文件，假如 Spring Boot 以内嵌式的方式运行 Tomcat，这种情况下 Tomcat 的日志都输出到哪里去了？

不知道今天的内容你消化得如何？如果还有疑问，请大胆的在留言区提问，也欢迎你把你的课后思考和心得记录下来，与我和其他同学一起讨论。如果你觉得今天有所收获，欢迎你把它分享给你的朋友。



## 深入拆解 Tomcat & Jetty

从源码角度深度探索 Java 中间件

李号双

eBay 技术主管



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 30 | 热点问题答疑（3）：Spring框架中的设计模式

### 精选留言 (2)

写留言



QQ怪

2019-07-20

默认应该是输出到控制台吧，配置目录可以输出到文件

展开 ∨



1





**z.l**

2019-07-20

springboot可以配置切换tomcat默认的日志模块吗？

展开 ▾

