日志系统

学习内容：

1. java历史
2. HelloWorld 程序

# log4j 简介

Log4j是[Apache](https://baike.baidu.com/item/Apache/8512995" \t "https://baike.baidu.com/item/log4j/_blank)的一个开源项目，通过使用Log4j，我们可以控制日志信息输送的目的地是[控制台](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E5%8F%B0/2438626" \t "https://baike.baidu.com/item/log4j/_blank)、文件、[GUI](https://baike.baidu.com/item/GUI" \t "https://baike.baidu.com/item/log4j/_blank)组件，甚至是套接口服务器、[NT](https://baike.baidu.com/item/NT/3443842" \t "https://baike.baidu.com/item/log4j/_blank)的事件记录器、[UNIX](https://baike.baidu.com/item/UNIX" \t "https://baike.baidu.com/item/log4j/_blank) [Syslog](https://baike.baidu.com/item/Syslog" \t "https://baike.baidu.com/item/log4j/_blank)[守护进程](https://baike.baidu.com/item/%E5%AE%88%E6%8A%A4%E8%BF%9B%E7%A8%8B/966835" \t "https://baike.baidu.com/item/log4j/_blank)等；我们也可以控制每一条日志的输出格式；通过定义每一条日志信息的级别，我们能够更加细致地控制日志的生成过程。最令人感兴趣的就是，这些可以通过一个[配置文件](https://baike.baidu.com/item/%E9%85%8D%E7%BD%AE%E6%96%87%E4%BB%B6/286550" \t "https://baike.baidu.com/item/log4j/_blank)来灵活地进行配置，而不需要修改应用的代码。 System.out.println()

日志:程序开发中的一些信息.

常用信息输出:System.out.println(“”);

\* 这种方式不好:

\* 如果输出内容比较多,项目已经开发完毕,不想使用输出.需要打开每个类,将输出的代码注释.

日志:Hibernate中使用slf4j技术.

slf4j:SLF4J，即简单日志门面（Simple Logging Facade for Java），不是具体的日志解决方案，它只服务于各种各样的日志系统。

\* 用于整合其他日志系统.

在企业中常用的日志记录:log4j

\* 是具体的日志记录方案.

Log4J的日志级别:

OFF,

\* fatal(致命的)、error(普通错误)、warn(警告)、info(信息)、debug(调试)、trace(堆栈信息).

Logger.error(“错误信息”);

\* 通过配置文件的形式,显示错误信息.

\* 配置的级别是info.

日志记录器（Logger）的行为是分等级的。如下表所示：

分 为OFF、FATAL、ERROR、WARN、INFO、DEBUG、ALL或者您定义的级别，这些级别是从高到低的级别。Log4j建议只使用四个级别，优先级从高到低分别是 ERROR、WARN、INFO、DEBUG。通过在这里定义的级别，您可以控制到应用程序中相应级别的日志信息的开关。比如在这里定义了INFO级别， 则应用程序中所有DEBUG级别的日志信息将不被打印出来

优先级从高到低分别是ERROR、WARN、INFO、DEBUG。通过在这里定义的级别，您可以控制到应用程序中相应级别的日志信息的开关。比如在这里定义了INFO级别，则应用程序中所有DEBUG级别的日志信息将不被打印出来。程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。如果设置级别为INFO，则优先级高于等于INFO级别（如：INFO、WARN、ERROR）的日志信息将可以被输出,小于该级别的如DEBUG将不会被输出。

所以从上面来说我们最好设置的日志级别为INFO ，或者是DEBUG级别。推荐使用DEBUG级别，可以显示所有的日志。

Log4J的三个组件:

记录器（Loggers）

\* 格式:记录器 = 级别, 输出源1 , 输出源2

\* log4j.rootLogger=info, stdout

输出源（Appenders）

\* log4j.appender.stdout=org.apache.log4j.ConsoleAppender :控制台进行输出.

\* log4j.appender.file=org.apache.log4j.FileAppender :向文件进行输出.

布局（Layouts）

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n

# slf4j

## 几个系统

眼下java应用日志收集都是採用日志框架(slf4j、apache commons logging)+日志系统（log4j、log4j2、LogBack、JUL等）的方式。而针对在分布式环境须要实时分析统计的日志，一般採用apache flume、facebook scribe等分布式日志收集系统。

### ****日志框架****：

提供日志调用的接口，实际的日志输出托付给日志系统实现。（apache commons logging)

* JCL(Jakarta Commons Logging)：比較流行的日志框架，非常多框架都依赖JCL，比如Spring等。
* SLF4j：提供新的API，初衷是配合Logback使用，但同一时候兼容Log4j。

### ****日志系统****：

负责输出日志

* Log4j：经典的一种日志解决方式。内部把日志系统抽象封装成Logger 、appender 、pattern 等实现。我们能够通过配置文件轻松的实现日志系统的管理和多样化配置。
* Log4j2：Log4j的2.0版本号。对Log4j进行了优化。比方支持參数API、支持异步appender、插件式架构等
* Logback：Log4j的替代产品。须要配合日志框架SLF4j使用
* JUL(java.util.logging)：JDK提供的日志系统。较混乱，不经常使用

眼下我们的应用大部分都是使用了SLF4j作为门面，然后搭配log4j或者log4j2日志系统。

## 简介



## Slf4j 是 什么 ？

slf4j 日志的统一接口（ common-loggin ） (jdbc) (interface)

log4j

logback

类

不管怎样，让我们回到这个话题，SLF4J不同于其他日志类库，与其它有很大的不同。SLF4J(Simple logging Facade for Java)不是一个真正的日志实现，而是一个抽象层（ abstraction layer），它允许你在后台使用任意一个日志类库。如果是在编写供内外部都可以使用的API或者通用类库，那么你真不会希望使用你类库的客户端必须使用你选择的日志类库。

总的来说，SLF4J使你的代码独立于任意一个特定的日志API，这是一个对于开发API的开发者很好的思想。虽然抽象日志类库的思想已经不是新鲜的事物而且Apache commons logging也已经在使用这种思想了，但现在SLF4J正迅速成为Java世界的日志标准。让我们再看看几个使用SLF4J而不是log4j、logback或者java.util.logging的理由。