链滴

Log4j 真的超简单

作者：pleaseok

原文链接：https://ld246.com/article/1544358701580

来源网站：链滴

首先，这里并没有非常详细的来介绍log4j。只是按照步骤、流程来明白怎么去使用。更多细节还是阅读log4j官网。

什么是Log4j?

log4j是一款功能强大的日志组件，来源于遵守开源精神的apache组织。链接地址。

它有三个重要的组件

- 记录器(Loggers)：指定日志输出级别。如果代码中使用的级别小于记录器配置的级别则不会被输。顺序：ALL <DEBUG<INFO<WARN<ERROR<FATAL<OFF
- 依附器或者叫输出源(Appenders)：为什么叫输出源呢？因为这个组件的作用是日志信息输出到控台、文件甚至数据库中。那我为什么又叫它依附器呢？因为它的实现是要依附于(=)某个实现类的。下

```
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
```

- 布局(Layouts)

用户可以根据自己的喜好格式化自己的日志信息。在appender后加上layout就可以完成这个功能。可以是HTML样式(org.apache.log4j.HTMLLayout)、自由定制的样式(org.apache.log4j.PatternLayout)、简单样式(org.apache.log4j.SimpleLayout - 输出日志的级别和信息字符串信息)、TTCC样式(or.apache.log4j.TTCCLayout - 用于输出日志产生时间类别或者线程登信息的时候)

示例如下

```
log4j.appender.youName.layout=layoutClassName
```

log4j到底该怎么用?

Step1.导jar包

==Maven:==

```
<!-- https://mvnrepository.com/artifact/log4j/log4j -->
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
</dependency>
```

Step2.书写配置文件(log4j.properties)

该文件在你项目的Source Folder下创建

```
<!-- 配置记录器(logger)，名称可以自定义（默认级别debug） -->
log4j.rootLogger = INFO,stdout,code666

<!-- 输出源,这里是输出到控制台 -->
log4j.appender.stdout = org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target = System.out
<!-- 布局,自由定制的样式PatternLayout -->
log4j.appender.stdout.layout = org.apache.log4j.PatternLayout
log4j.appender.stdout.ConversionPattern = [%-5p] %d{yyyy-MM-dd HH:mm:ss,SSS} method:
```

```
l%n%m%n

<!-- code666,输出到文件 -->
log4j.appender.code666 = org.apache.log4j.DailyRollingFileAppender
log4j.appender.code666.File =E:\error.log
log4j.appender.code666.Append = true
log4j.appender.code666.Threshold = ERROR
log4j.appender.code666.layout = org.apache.log4j.PatternLayout
log4j.appender.code666.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss}  [ %t:%r ] -
 %p ]  %m%n
```

Step3.测试类(Test.class)

```
import org.apache.log4j.Logger;

public class Test {
    private static final Logger log = Logger.getLogger(Test.class);
    public static void main(String[] args) {
        log.debug("i am debug message");
        log.info("i am info message");
    }
}
```

最终输出结果为"i am info message",因为默认为info级别，而debug是小于info级别的，所以不显
。

扩展之slf4j的使用

- slf4j是什么?

JAVA简易日志门面，是一套包装Logging 框架的界面程式，以外观模式实现。可以在软件部署的时
决定要使用的Logging 框架，目前主要支援的有Java Logging API、log4j及logback等框架。以MIT
授权方式发布。==维基百科==

slf4j内部并没有任何日志的实现类，它只是提供了一些接口。也就是一套接口支持logging、logbac
、log4j等日志框架。

- slf4j怎么搭配log4j使用?

1. maven

需要导入slf4j-api与slf4j-log4j12，由于slf4j-log4j12里包含了log4j，所以不必要再写log4j的依赖

```
<!-- <dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
</dependency> -->
  <!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-api -->
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.25</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
```

```
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.25</version>
</dependency>
```

2. Test类

这里是用Logger.getLogger的工厂方式来获取slf4j的Logger对象

```
//import org.apache.log4j.Logger;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Test {
        //private static final Logger log = Logger.getLogger(Test.class);
        private static final Logger log = LoggerFactory.getLogger(Test.class);
        public static void main(String[] args) {
            log.debug("i am debug message");
            log.info("i am info message");
        }
    }
```

运行后显示与上一样。

log4j输出源的其它配置(搬运互联网)

- 控制台console日志输出源

```
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.Threshold=DEBUG
log4j.appender.console.ImmediateFlush=true
log4j.appender.console.Target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} [%p] %m%n
```

- 文件logFile日志输出源

```
log4j.appender.logFile=org.apache.log4j.FileAppender
log4j.appender.logFile.Threshold=DEBUG
log4j.appender.logFile.ImmediateFlush=true
log4j.appender.logFile.Append=true
log4j.appender.logFile.File=D:/logs/log.log4j
log4j.appender.logFile.layout=org.apache.log4j.PatternLayout
log4j.appender.logFile.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} [%p] %m%n
```

- 回滚文件rollingFile日志输出源

```
log4j.appender.rollingFile=org.apache.log4j.RollingFileAppender
log4j.appender.rollingFile.Threshold=DEBUG
log4j.appender.rollingFile.ImmediateFlush=true
log4j.appender.rollingFile.Append=true
log4j.appender.rollingFile.File=D:/logs/log.log4j
log4j.appender.rollingFile.MaxFileSize=200KB
log4j.appender.rollingFile.MaxBackupIndex=50
log4j.appender.rollingFile.layout=org.apache.log4j.PatternLayout
```

```
log4j.appender.rollingFile.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} [%p] %m%n
```

- 定期回滚文件dailyFile日志输出源

```
log4j.appender.dailyFile=org.apache.log4j.DailyRollingFileAppender
log4j.appender.dailyFile.Threshold=DEBUG
log4j.appender.dailyFile.ImmediateFlush=true
log4j.appender.dailyFile.Append=true
log4j.appender.dailyFile.File=D:/logs/log.log4j
log4j.appender.dailyFile.DatePattern='.'yyyy-MM-dd
log4j.appender.dailyFile.layout=org.apache.log4j.PatternLayout
log4j.appender.dailyFile.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} [%p] %m%n
```

- 应用于socket日志输出源

```
log4j.appender.socket=org.apache.log4j.RollingFileAppender
log4j.appender.socket.RemoteHost=localhost
log4j.appender.socket.Port=5001
log4j.appender.socket.LocationInfo=true
```

- 发送日志到指定邮件

```
log4j.appender.mail=org.apache.log4j.net.SMTPAppender
log4j.appender.mail.Threshold=FATAL
log4j.appender.mail.BufferSize=10
log4j.appender.mail.From = xxx@mail.com
log4j.appender.mail.SMTPHost=mail.com
log4j.appender.mail.Subject=Log4J Message
log4j.appender.mail.To= xxx@mail.com
log4j.appender.mail.layout=org.apache.log4j.PatternLayout
log4j.appender.mail.layout.ConversionPattern=[%-5p] %d(%r) --> [%t] %l: %m %x %n
```

- 应用到数据库

```
log4j.appender.database=org.apache.log4j.jdbc.JDBCAppender
log4j.appender.database.URL=jdbc:mysql://localhost:3306/test
log4j.appender.database.driver=com.mysql.jdbc.Driver
log4j.appender.database.user=root
log4j.appender.database.password=
log4j.appender.database.sql=INSERT INTO LOG4J (Message) VALUES('=[%-5p] %d(%r) --> [%
] %l: %m %x %n')
log4j.appender.database.layout=org.apache.log4j.PatternLayout
log4j.appender.database.layout.ConversionPattern=[%-5p] %d(%r) --> [%t] %l: %m %x %n
```