# SLF4j-log4j使用总结

## 一、Maven配置

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.7.12</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.7.12</version>

</dependency>

## 二、log4j.properties配置说明

appenderName：以下示例：R统一代表

### log4j.rootLogger

配置根Logger

### log4j.appender.R

配置日志信息输出目的地Appender

Appender分为以下5类：

1、org.apache.log4j.ConsoleAppender（控制台）

2、org.apache.log4j.FileAppender（文件）

3、org.apache.log4j.DailyRollingFileAppender（每天产生一个日志文件）

4、org.apache.log4j.RollingFileAppender（文件大小到达指定尺寸的时候产生一个新的文件）

5、org.apache.log4j.WriterAppender（将日志信息以流格式发送到任意指定的地方）

### log4j.appender.R.layout

配置日志信息的格式（布局）

org.apache.log4j.HTMLLayout（以HTML表格形式布局）

org.apache.log4j.PatternLayout（可以灵活地指定布局模式）

org.apache.log4j.SimpleLayout（包含日志信息的级别和信息字符串）

org.apache.log4j.TTCCLayout（包含日志产生的时间、线程、类别等等信息）

### log4j.appender.R.layout.ConversionPattern

**输出打印参数:**

%d   输出日志时间点的日期或时间，%d{yyyy-MM-dd HH\:mm\:ss,SSS}

%p   输出优先级，即DEBUG，INFO，WARN，ERROR，FATAL

%l   输出日志事件的发生位置，包括类目名、发生的线程，以及在代码中的行数。举例：test.main(test.java: 10 )

%m   输出代码中指定的消息

%n   输出一个回车换行符，Windows平台为“\r\n”，Unix平台为“\n”

%c   输出所属的类目，通常就是所在类的全名

%r   输出自应用启动到输出该log信息耗费的毫秒数

%t   输出产生该日志事件的线程名

一个常用的：%d{yyyy-MM-dd HH\:mm\:ss,SSS} %p %l %m%n

### ConsoleAppender选项

1、Threshold=DEBUG:指定日志消息的输出最低层次。

2、ImmediateFlush=true:默认值是true,意谓着所有的消息都会被立即输出。

3、Target=System.err：默认情况下是：System.out,指定输出控制台

### FileAppender 选项

1、Threshold=DEBUF:指定日志消息的输出最低层次。

2、ImmediateFlush=true:默认值是true,意谓着所有的消息都会被立即输出。

3、File=mylog.txt:指定消息输出到mylog.txt文件。

4、Append=false:默认值是true,即将消息增加到指定文件中，false指将消息覆盖指定的文件内容。

### RollingFileAppender 选项

1、Threshold=DEBUG:指定日志消息的输出最低层次。

2、ImmediateFlush=true:默认值是true,意谓着所有的消息都会被立即输出。

3、File=mylog.txt:指定消息输出到mylog.txt文件。

4、Append=false:默认值是true,即将消息增加到指定文件中，false指将消息覆盖指定的文件内容。

5、MaxFileSize=100KB: 后缀可以是KB, MB 或者是 GB. 在日志文件到达该大小时，将会自动滚动，即将原来的内容移到mylog.log.1文件。

6、MaxBackupIndex=2:指定可以产生的滚动文件的最大数。

## 三、SLF4j好处

1、使用SLF4J【Simple Logging Facade for Java】使得你的程序独立于特定的日志类库。

2、SLF4J API特性的占位符(place holder)，在代码中表示为“{}”的特性。降低了你代码中字符串连接次数，节省了新建String对象的开销。因为String对象是不可修改并且它们建立在一个String池中，它们消耗堆内存( heap memory)而且大多数时间他们是不被需要的，使用SLF4J,你可以在运行时延迟字符串的建立，这意味着只有需要的String对象才被建立。

3、使用SLF4J日志方法的代码，来自于slf4j-log4j12-1.6.1.jar中的Log4j的适配器类Log4jLoggerAdapter。

1、在你的开源或内部类库中使用SLF4J会使得它独立于任何一个特定的日志实现，这意味着不需要管理多个日志配置或者多个日志类库，你的客户端会很感激这点。

2、SLF4J提供了基于占位符的日志方法，这通过去除检查isDebugEnabled(), isInfoEnabled()等等，提高了代码可读性。

3、通过使用SLF4J的日志方法，你可以延迟构建日志信息（Srting）的开销，直到你真正需要，这对于内存和CPU都是高效的。

4、更少的暂时的字符串意味着垃圾回收器（Garbage Collector）需要做更好的工作，这意味着你的应用程序有为更好的吞吐量和性能。

5、这些好处只是冰山一角，你将在开始使用SL4J和阅读其中代码的时候知道更多的好处。我强烈建议，任何一个新的Java程序员，都应该使用SLF4J做日志而不是使用包括Log4J在内的其他日志API。

## 四、类单独日志

为类：com.afeng.utils.FileUtils单独配置一个日志文件：

log4j.logger.com.afeng.utils.FileUtils=

不写入主日志文件

log4j.additivity.com.afeng.utils.FileUtils=false

## 五、log4j.properties示例

