1.

一、Log4j简介

Log4j有三个主要的组件：Loggers(记录器)，Appenders (输出源)和Layouts(布局)。这里可简单理解为日志类别，日志要输出的地方和日志以何种形式输出。综合使用这三个组件可以轻松地记录信息的类型和级别，并可以在运行时控制日志输出的样式和位置。

1、Loggers

Loggers组件在此系统中被分为五个级别：DEBUG、INFO、WARN、ERROR和FATAL。这五个级别是有顺序的，DEBUG < INFO < WARN < ERROR < FATAL，分别用来指定这条日志信息的重要程度，明白这一点很重要，Log4j有一个规则：只输出级别不低于设定级别的日志信息，假设Loggers级别设定为INFO，则INFO、WARN、ERROR和FATAL级别的日志信息都会输出，而级别比INFO低的DEBUG则不会输出。

2、Appenders

禁用和使用日志请求只是Log4j的基本功能，Log4j日志系统还提供许多强大的功能，比如允许把日志输出到不同的地方，如控制台（Console）、文件（Files）等，可以根据天数或者文件大小产生新的文件，可以以流的形式发送到其它地方等等。

常使用的类如下：

org.apache.log4j.ConsoleAppender（控制台）

org.apache.log4j.FileAppender（文件）

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

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）

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

配置模式：

log4j.appender.appenderName = className

log4j.appender.appenderName.Option1 = value1

…

log4j.appender.appenderName.OptionN = valueN

3、Layouts

有时用户希望根据自己的喜好格式化自己的日志输出，Log4j可以在Appenders的后面附加Layouts来完成这个功能。Layouts提供四种日志输出样式，如根据HTML样式、自由指定样式、包含日志级别与信息的样式和包含日志时间、线程、类别等信息的样式。

常使用的类如下：

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、类别等信息）

配置模式：

log4j.appender.appenderName.layout =className

log4j.appender.appenderName.layout.Option1 = value1

…

log4j.appender.appenderName.layout.OptionN = valueN

二、配置详解

在实际应用中，要使Log4j在系统中运行须事先设定配置文件。配置文件事实上也就是对Logger、Appender及Layout进行相应设定。Log4j支持两种配置文件格式，一种是XML格式的文件，一种是properties属性文件。下面以properties属性文件为例介绍log4j.properties的配置。

1、配置根Logger：

log4j.rootLogger = [level] , appenderName1, appenderName2, …

log4j.additivity.org.apache=false：表示Logger不会在父Logger的appender里输出，默认为true。

level ：设定日志记录的最低级别，可设的值有OFF、FATAL、ERROR、WARN、INFO、DEBUG、ALL或者自定义的级别，Log4j建议只使用中间四个级别。通过在这里设定级别，您可以控制应用程序中相应级别的日志信息的开关，比如在这里设定了INFO级别，则应用程序中所有DEBUG级别的日志信息将不会被打印出来。

appenderName：就是指定日志信息要输出到哪里。可以同时指定多个输出目的地，用逗号隔开。

例如：log4j.rootLogger＝INFO,A1,B2,C3

2、配置日志信息输出目的地（appender）：

log4j.appender.appenderName = className

appenderName：自定义appderName，在log4j.rootLogger设置中使用；

className：可设值如下：

(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）

(1)ConsoleAppender选项：

Threshold=WARN：指定日志信息的最低输出级别，默认为DEBUG。

ImmediateFlush=true：表示所有消息都会被立即输出，设为false则不输出，默认值是true。

Target=System.err：默认值是System.out。

(2)FileAppender选项：

Threshold=WARN：指定日志信息的最低输出级别，默认为DEBUG。

ImmediateFlush=true：表示所有消息都会被立即输出，设为false则不输出，默认值是true。

Append=false：true表示消息增加到指定文件中，false则将消息覆盖指定的文件内容，默认值是true。

File=D:/logs/logging.log4j：指定消息输出到logging.log4j文件中。

(3)DailyRollingFileAppender选项：

Threshold=WARN：指定日志信息的最低输出级别，默认为DEBUG。

ImmediateFlush=true：表示所有消息都会被立即输出，设为false则不输出，默认值是true。

Append=false：true表示消息增加到指定文件中，false则将消息覆盖指定的文件内容，默认值是true。

File=D:/logs/logging.log4j：指定当前消息输出到logging.log4j文件中。

DatePattern='.'yyyy-MM：每月滚动一次日志文件，即每月产生一个新的日志文件。当前月的日志文

件名为logging.log4j，前一个月的日志文件名为logging.log4j.yyyy-MM。
另外，也可以指定按周、天、时、分等来滚动日志文件，对应的格式如下：
1)'.'yyyy-MM：每月
2)'.'yyyy-ww：每周
3)'.'yyyy-MM-dd：每天
4)'.'yyyy-MM-dd-a：每天两次
5)'.'yyyy-MM-dd-HH：每小时
6)'.'yyyy-MM-dd-HH-mm：每分钟
(4)RollingFileAppender选项：
Threshold=WARN：指定日志信息的最低输出级别，默认为DEBUG。
ImmediateFlush=true：表示所有消息都会被立即输出，设为false则不输出，默认值是true。
Append=false：true表示消息增加到指定文件中，false则将消息覆盖指定的文件内容，默认值是true。
File=D:/logs/logging.log4j：指定消息输出到logging.log4j文件中。
MaxFileSize=100KB：后缀可以是KB, MB 或者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logging.log4j.1文件中。
MaxBackupIndex=2：指定可以产生的滚动文件的最大数，例如，设为2则可以产生
logging.log4j.1，logging.log4j.2两个滚动文件和一个logging.log4j文件。
3、配置日志信息的输出格式（Layout）：
log4j.appender.appenderName.layout=className
className：可设值如下：
(1)org.apache.log4j.HTMLLayout（以HTML表格形式布局）
(2)org.apache.log4j.PatternLayout（可以灵活地指定布局模式）
(3)org.apache.log4j.SimpleLayout（包含日志信息的级别和信息字符串）
(4)org.apache.log4j.TTCCLayout（包含日志产生的时间、线程、类别等等信息）
(1)HTMLLayout选项：
LocationInfo=true：输出java文件名称和行号，默认值是false。
Title=My Logging： 默认值是Log4J Log Messages。
(2)PatternLayout选项：
ConversionPattern=%m%n：设定以怎样的格式显示消息。
格式化符号说明：
%p：输出日志信息的优先级，即DEBUG，INFO，WARN，ERROR，FATAL。
%d：输出日志时间点的日期或时间，默认格式为ISO8601，也可以在其后指定格式，如
：%d{yyyy/MM/dd HH:mm:ss,SSS}。
%r：输出自应用程序启动到输出该log信息耗费的毫秒数。
%t：输出产生该日志事件的线程名。
%l：输出日志事件的发生位置，相当于%c.%M(%F:%L)的组合，包括类全名、方法、文件名以及在代码中的行数。例如：test.TestLog4j.main(TestLog4j.java:10)。
%c：输出日志信息所属的类目，通常就是所在类的全名。
%M：输出产生日志信息的方法名。
%F：输出日志消息产生时所在的文件名称。
%L:：输出代码中的行号。

%m:：输出代码中指定的具体日志信息。
%n：输出一个回车换行符，Windows平台为"rn"，Unix平台为"n"。
%x：输出和当前线程相关联的NDC(嵌套诊断环境)，尤其用到像java servlets这样的多客户多线程的应用中。
%%：输出一个"%"字符。
另外，还可以在%与格式字符之间加上修饰符来控制其最小长度、最大长度、和文本的对齐方式。如：
1) c：指定输出category的名称，最小的长度是20，如果category的名称长度小于20的话，默认的情况下右对齐。
2)%-20c："-"号表示左对齐。
3)%.30c：指定输出category的名称，最大的长度是30，如果category的名称长度大于30的话，就会将左边多出的字符截掉，但小于30的话也不会补空格。

附：Log4j比较全面的配置
Log4j配置文件实现了输出到控制台、文件、回滚文件、发送日志邮件、输出到数据库日志表、自定义标签等全套功能。

```
log4j.rootLogger=DEBUG,console,dailyFile,im
log4j.additivity.org.apache=true
# 控制台(console)
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=[%-5p] %d(%r) --> [%t] %l: %m %x %n
# 日志文件(logFile)
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=[%-5p] %d(%r) --> [%t] %l: %m %x %n
# 回滚文件(rollingFile)
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=[%-5p] %d(%r) --> [%t] %l: %m %x %n
# 定期回滚日志文件(dailyFile)
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=[%-5p] %d(%r) --> [%t] %l: %m %x %n
# 应用于socket
log4j.appender.socket=org.apache.log4j.RollingFileAppender
log4j.appender.socket.RemoteHost=localhost
log4j.appender.socket.Port=5001
log4j.appender.socket.LocationInfo=true
# Set up for Log Factor 5
log4j.appender.socket.layout=org.apache.log4j.PatternLayout
log4j.appender.socket.layout.ConversionPattern=[%-5p] %d(%r) --> [%t] %l: %m %x %n
# Log Factor 5 Appender
log4j.appender.LF5_APPENDER=org.apache.log4j.lf5.LF5Appender
log4j.appender.LF5_APPENDER.MaxNumberOfRecords=2000
# 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
# 应用于数据库
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) --> [%t] %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
# 自定义Appender
log4j.appender.im = net.cybercorlin.util.logger.appender.IMAppender
log4j.appender.im.host = mail.cybercorlin.net
log4j.appender.im.username = username
log4j.appender.im.password = password
log4j.appender.im.recipient = corlin@cybercorlin.net
log4j.appender.im.layout=org.apache.log4j.PatternLayout
log4j.appender.im.layout.ConversionPattern=[%-5p] %d(%r) --> [%t] %l: %m %x %n

log4j的强大功能无可置疑，但实际应用中免不了遇到某个功能需要输出独立的日志文件的情况，怎样才能把所需的内容从原有日志中分离，形成单独的日志文件呢？其实只要在现有的log4j基础上稍加配置即可轻松实现这一功能。

先看一个常见的log4j.properties文件，它是在控制台和myweb.log文件中记录日志：

log4j.rootLogger=DEBUG, stdout, logfile

log4j.category.org.springframework=ERROR

log4j.category.org.apache=INFO

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

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

log4j.appender.stdout.layout.ConversionPattern=%d %p [%c] - %m%n

log4j.appender.logfile=org.apache.log4j.RollingFileAppender

log4j.appender.logfile.File=${myweb.root}/WEB-INF/log/myweb.log

log4j.appender.logfile.MaxFileSize=512KB

log4j.appender.logfile.MaxBackupIndex=5

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

log4j.appender.logfile.layout.ConversionPattern=%d %p [%c] - %m%n

如果想对不同的类输出不同的文件(以cn.com.Test为例)，先要在Test.java中定义:

private static Log logger = LogFactory.getLog(Test.class);

然后在log4j.properties中加入:

log4j.logger.cn.com.Test= DEBUG, test

log4j.appender.test=org.apache.log4j.FileAppender

log4j.appender.test.File=${myweb.root}/WEB-INF/log/test.log

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

log4j.appender.test.layout.ConversionPattern=%d %p [%c] - %m%n

也就是让cn.com.Test中的logger使用log4j.appender.test所做的配置。

但是，如果在同一类中需要输出多个日志文件呢？其实道理是一样的，先在Test.java中定义:

```
private static Log logger1 = LogFactory.getLog("myTest1");
private static Log logger2 = LogFactory.getLog("myTest2");
```

然后在log4j.properties中加入:

```
log4j.logger.myTest1= DEBUG,    test1
log4j.appender.test1=org.apache.log4j.FileAppender
log4j.appender.test1.File=${myweb.root}/WEB-INF/log/test1.log
log4j.appender.test1.layout=org.apache.log4j.PatternLayout
log4j.appender.test1.layout.ConversionPattern=%d %p [%c] - %m%n

log4j.logger.myTest2= DEBUG,    test2
log4j.appender.test2=org.apache.log4j.FileAppender
log4j.appender.test2.File=${myweb.root}/WEB-INF/log/test2.log
log4j.appender.test2.layout=org.apache.log4j.PatternLayout
log4j.appender.test2.layout.ConversionPattern=%d %p [%c] - %m%n
```

也就是在用logger时给它一个自定义的名字(如这里的"myTest1")，然后在log4j.properties中做出相应配置即可。别忘了不同日志要使用不同的logger(如输出到test1.log的要用logger1.info("abc"))。

还有一个问题，就是这些自定义的日志默认是同时输出到log4j.rootLogger所配置的日志中的，如何能只让它们输出到自己指定的日志中呢？别急，这里有个开关：

log4j.additivity.myTest1 = false

它用来设置是否同时输出到log4j.rootLogger所配置的日志中，设为false就不会输出到其它地方啦！注意这里的"myTest1"是你在程序中给logger起的那个自定义的名字！

如果你说，我只是不想同时输出这个日志到log4j.rootLogger所配置的logfile中，stdout里我还想同时输出呢！那也好办，把你的log4j.logger.myTest1 = DEBUG, test1改为下式就OK啦！

log4j.logger.myTest1=DEBUG, test1

下面是文件上传时记录文件类型的log日志，并输出到指定文件的配置

```
1 log4j.rootLogger=INFO, stdout
2 ######################### logger ##############################
3
4 log4j.appender.stdout = org.apache.log4j.ConsoleAppender
5 log4j.appender.stdout.layout = org.apache.log4j.PatternLayout
6 log4j.appender.stdout.layout.conversionPattern = %d [%t] %-5p %c - %m%n
7 log4j.logger.extProfile=INFO, extProfile#日志级别是INFO,标签是extProfile
8 log4j.additivity.extProfile=false;#输出到指定文件extProfile.log中
9
10 #userProfile
log\uff08\u8bb0\u5f55\u4fee\u6539\u5bc6\u7801\uff0c\u627e\u56de\u5bc6\u7801\uff0c\u4fee\u6539\u90ae\u7bb1\uff0c\u4fee\u6539\u624b\u673a\u53f7\uff09
11 log4j.appender.extProfile=org.apache.log4j.RollingFileAppender
12 log4j.appender.extProfile.File=logs/extProfile.log#输出到resin根目录的logs文件夹,log4j会自动生成目录和文件
```

```
13 log4j.appender.extProfile.MaxFileSize=20480KB#超过20M就重新创建一个文件
14 log4j.appender.extProfile.MaxBackupIndex=10
15 log4j.appender.extProfile.layout=org.apache.log4j.PatternLayout
16 log4j.appender.extProfile.layout.ConversionPattern=%d [%t] %-5p %c - %m%n
```

Java端控制代码

```
<%@page contentType="text/html" session="false" pageEncoding="UTF-8"%><%@page
...
org.apache.commons.logging.Log,
org.apache.commons.logging.LogFactory
"%>
...
Log extProfile  = LogFactory.getLog("extProfile");
...
if (!item.isFormField()) {
    String fileExt = StringUtils.substringAfterLast(item.getName(), ".").toLowerCase();
    extProfile.info("upfile type is : [ "+fileExt +" ]");
}
```

参考：

http://www.cnblogs.com/ITEagle/archive/2010/04/23/1718365.html

http://blog.csdn.net/anlina_1984/article/details/5313023

原文：http://www.cnblogs.com/ITtangtang/p/3926665.html