### Log4j配置详解之log4j.properties

Log4J的配置文件(Configuration File)就是用来设置记录器的级别、存放器和布局的，它可接key=value格式的设置或xml格式的设置信息。通过配置，可以创建出Log4J的运行环境。

一、log4j.jar包下载地址：<http://logging.apache.org/log4j/1.2/download.html>

二、配置文件类型：log4j配置文件可以是log4j.xml也可以是log4j.properties

**log4j.properties配置说明**

1、配置文件

配置的基本格式

**Txt代码  [收藏代码](javascript:void())**

1. #配置根Logger
2. log4j.rootLogger = [ level ] , appenderName1 , appenderName2 , …
4. #配置日志信息输出目的地Appender
5. log4j.appender.appenderName = fully.qualified.name.of.appender.class
6. log4j.appender.appenderName.option1 = value1
7. …
8. log4j.appender.appenderName.optionN = valueN
10. #配置日志信息的格式（布局）
11. log4j.appender.appenderName.layout = fully.qualified.name.of.layout.class
12. log4j.appender.appenderName.layout.option1 = value1
13. …
14. log4j.appender.appenderName.layout.optionN = valueN

其中 [level]是日志输出级别，共有5级：

**Txt代码  [收藏代码](javascript:void())**

1. FATAL     0
2. ERROR     3
3. WARN      4
4. INFO      6
5. DEBUG     7

Appender 为日志输出目的地，Log4j提供的appender有以下几种：

**Txt代码  [收藏代码](javascript:void())**

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）

Layout：日志输出格式，Log4j提供的layout有以下几种：

**Txt代码  [收藏代码](javascript:void())**

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）

打印参数: Log4J采用类似C语言中的printf函数的打印格式格式化日志信息，如下:

**Txt代码  [收藏代码](javascript:void())**

1. %m   输出代码中指定的消息
2. %p   输出优先级，即DEBUG，INFO，WARN，ERROR，FATAL
3. %r   输出自应用启动到输出该log信息耗费的毫秒数
4. %c   输出所属的类目，通常就是所在类的全名
5. %t   输出产生该日志事件的线程名
6. %n   输出一个回车换行符，Windows平台为“\r\n”，Unix平台为“\n”
7. %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，比如：%d{yyy MMM dd HH:mm:ss , SSS}，输出类似：2002年10月18日 22 ： 10 ： 28 ， 921
8. %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：Testlog4.main(TestLog4.java: 10 )

2. 在代码中初始化Logger:

1）在程序中调用BasicConfigurator.configure()方法：给根记录器增加一个ConsoleAppender，输出格式通过PatternLayout设为"%-4r [%t] %-5p %c %x - %m%n"，还有根记录器的默认级别是Level.DEBUG.   
2）配置放在文件里，通过命令行参数传递文件名字，通过PropertyConfigurator.configure(args[x])解析并配置；  
3）配置放在文件里，通过环境变量传递文件名等信息，利用log4j默认的初始化过程解析并配置；  
4）配置放在文件里，通过应用服务器配置传递文件名等信息，利用一个特殊的servlet来完成配置。

3. 为不同的 Appender 设置日志输出级别：

当调试系统时，我们往往注意的只是异常级别的日志输出，但是通常所有级别的输出都是放在一个文件里的，如果日志输出的级别是BUG！？那就慢慢去找吧。  
这时我们也许会想要是能把异常信息单独输出到一个文件里该多好啊。当然可以，Log4j已经提供了这样的功能，我们只需要在配置中修改Appender的Threshold 就能实现,比如下面的例子：

**[配置文件]**

**Txt代码  [收藏代码](javascript:void())**

1. ### set log levels ###
2. log4j.rootLogger = debug , stdout , D , E
4. ### 输出到控制台 ###
5. log4j.appender.stdout = org.apache.log4j.ConsoleAppender
6. log4j.appender.stdout.Target = System.out
7. log4j.appender.stdout.layout = org.apache.log4j.PatternLayout
8. log4j.appender.stdout.layout.ConversionPattern = %d{ABSOLUTE} %5p %c{ 1 }:%L - %m%n
10. ### 输出到日志文件 ###
11. log4j.appender.D = org.apache.log4j.DailyRollingFileAppender
12. log4j.appender.D.File = logs/log.log
13. log4j.appender.D.Append = true
14. log4j.appender.D.Threshold = DEBUG ## 输出DEBUG级别以上的日志
15. log4j.appender.D.layout = org.apache.log4j.PatternLayout
16. log4j.appender.D.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss} [ %t:%r ] - [ %p ] %m%n
18. ### 保存异常信息到单独文件 ###
19. log4j.appender.D = org.apache.log4j.DailyRollingFileAppender
20. log4j.appender.D.File = logs/error.log ## 异常日志文件名
21. log4j.appender.D.Append = true
22. log4j.appender.D.Threshold = ERROR ## 只输出ERROR级别以上的日志!!!
23. log4j.appender.D.layout = org.apache.log4j.PatternLayout
24. log4j.appender.D.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss} [ %t:%r ] - [ %p ] %m%n

**[代码中使用]**

**Java代码  [收藏代码](javascript:void())**

1. **public** **class** TestLog4j {
2. **public** **static** **void** main(String[] args) {
3. PropertyConfigurator.configure( " D:/Code/conf/log4j.properties " );
4. Logger logger = Logger.getLogger(TestLog4j. **class** );
5. logger.debug( " debug " );
6. logger.error( " error " );
7. }

 运行一下，看看异常信息是不是保存在了一个单独的文件error.log中

**log4j.properties 使用**

一.参数意义说明  
输出级别的种类  
ERROR、WARN、INFO、DEBUG  
ERROR 为严重错误 主要是程序的错误  
WARN 为一般警告，比如session丢失  
INFO 为一般要显示的信息，比如登录登出  
DEBUG 为程序的调试信息

配置日志信息输出目的地  
log4j.appender.appenderName = fully.qualified.name.of.appender.class  
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）

配置日志信息的格式  
log4j.appender.appenderName.layout = fully.qualified.name.of.layout.class  
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）

控制台选项  
Threshold=DEBUG:指定日志消息的输出最低层次。  
ImmediateFlush=true:默认值是true,意谓着所有的消息都会被立即输出。  
Target=System.err：默认情况下是：System.out,指定输出控制台  
FileAppender 选项  
Threshold=DEBUF:指定日志消息的输出最低层次。  
ImmediateFlush=true:默认值是true,意谓着所有的消息都会被立即输出。  
File=mylog.txt:指定消息输出到mylog.txt文件。  
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  
Threshold=DEBUG:指定日志消息的输出最低层次。  
ImmediateFlush=true:默认值是true,意谓着所有的消息都会被立即输出。  
File=mylog.txt:指定消息输出到mylog.txt文件。  
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。  
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。  
MaxBackupIndex=2:指定可以产生的滚动文件的最大数。  
log4j.appender.A1.layout.ConversionPattern=%-4r %-5p %d{yyyy-MM-dd HH:mm:ssS} %c %m%n

日志信息格式中几个符号所代表的含义：  
-X号: X信息输出时左对齐；  
%p: 输出日志信息优先级，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，比如：%d{yyy MMM dd HH:mm:ss,SSS}，输出类似：2002年10月18日 22：10：28，921  
%r: 输出自应用启动到输出该log信息耗费的毫秒数  
%c: 输出日志信息所属的类目，通常就是所在类的全名  
%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,包括类目名、发生的线程，以及在代码中的行数。举例：Testlog4.main (TestLog4.java:10)  
%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  
%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: 输出代码中指定的消息,产生的日志具体信息  
%n: 输出一个回车换行符，Windows平台为"\r\n"，Unix平台为"\n"输出日志信息换行  
可以在%与模式字符之间加上修饰符来控制其最小宽度、最大宽度、和文本的对齐方式。如：  
1)%20c：指定输出category的名称，最小的宽度是20，如果category的名称小于20的话，默认的情况下右对齐。  
2)%-20c:指定输出category的名称，最小的宽度是20，如果category的名称小于20的话，"-"号指定左对齐。  
3)%.30c:指定输出category的名称，最大的宽度是30，如果category的名称大于30的话，就会将左边多出的字符截掉，但小于30的话也不会有空格。  
4)%20.30c:如果category的名称小于20就补空格，并且右对齐，如果其名称长于30字符，就从左边较远输出的字符截掉。

二.文件配置

文件配置Sample1

**Txt代码  [收藏代码](javascript:void())**

1. log4j.rootLogger=DEBUG,A1,R
2. #log4j.rootLogger=INFO,A1,R
3. # ConsoleAppender 输出
4. log4j.appender.A1=org.apache.log4j.ConsoleAppender
5. log4j.appender.A1.layout=org.apache.log4j.PatternLayout
6. log4j.appender.A1.layout.ConversionPattern=%-d{yyyy-MM-dd HH:mm:ss,SSS} [%c]-[%p] %m%n
7. # File 输出 一天一个文件,输出路径可以定制,一般在根路径下
8. log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
9. log4j.appender.R.File=blog\_log.txt
10. log4j.appender.R.MaxFileSize=500KB
11. log4j.appender.R.MaxBackupIndex=10
12. log4j.appender.R.layout=org.apache.log4j.PatternLayout
13. log4j.appender.R.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss,SSS} [%t] [%c] [%p] - %m%n

文件配置Sample2

下面给出的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。

**Txt代码  [收藏代码](javascript:void())**

1. log4j.rootLogger=DEBUG,CONSOLE,A1,im
2. #DEBUG,CONSOLE,FILE,ROLLING\_FILE,MAIL,DATABASE
3. log4j.addivity.org.apache=true
4. ###################
5. # Console Appender
6. ###################
7. log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
8. log4j.appender.Threshold=DEBUG
9. log4j.appender.CONSOLE.Target=System.out
10. log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
11. log4j.appender.CONSOLE.layout.ConversionPattern=[framework] %d - %c -%-4r [%t] %-5p %c %x - %m%n
12. #log4j.appender.CONSOLE.layout.ConversionPattern=[start]%d{DATE}[DATE]%n%p[PRIORITY]%n%x[NDC]%n%t[THREAD] n%c[CATEGORY]%n%m[MESSAGE]%n%n
13. #####################
14. # File Appender
15. #####################
16. log4j.appender.FILE=org.apache.log4j.FileAppender
17. log4j.appender.FILE.File=file.log
18. log4j.appender.FILE.Append=false
19. log4j.appender.FILE.layout=org.apache.log4j.PatternLayout
20. log4j.appender.FILE.layout.ConversionPattern=[framework] %d - %c -%-4r [%t] %-5p %c %x - %m%n
21. # Use this layout for LogFactor 5 analysis
22. ########################
23. # Rolling File
24. ########################
25. log4j.appender.ROLLING\_FILE=org.apache.log4j.RollingFileAppender
26. log4j.appender.ROLLING\_FILE.Threshold=ERROR
27. log4j.appender.ROLLING\_FILE.File=rolling.log
28. log4j.appender.ROLLING\_FILE.Append=true
29. log4j.appender.ROLLING\_FILE.MaxFileSize=10KB
30. log4j.appender.ROLLING\_FILE.MaxBackupIndex=1
31. log4j.appender.ROLLING\_FILE.layout=org.apache.log4j.PatternLayout
32. log4j.appender.ROLLING\_FILE.layout.ConversionPattern=[framework] %d - %c -%-4r [%t] %-5p %c %x - %m%n
33. ####################
34. # Socket Appender
35. ####################
36. log4j.appender.SOCKET=org.apache.log4j.RollingFileAppender
37. log4j.appender.SOCKET.RemoteHost=localhost
38. log4j.appender.SOCKET.Port=5001
39. log4j.appender.SOCKET.LocationInfo=true
40. # Set up for Log Facter 5
41. log4j.appender.SOCKET.layout=org.apache.log4j.PatternLayout
42. log4j.appender.SOCET.layout.ConversionPattern=[start]%d{DATE}[DATE]%n%p[PRIORITY]%n%x[NDC]%n%t[THREAD]%n%c[CATEGORY]%n%m[MESSAGE]%n%n
43. ########################
44. # Log Factor 5 Appender
45. ########################
46. log4j.appender.LF5\_APPENDER=org.apache.log4j.lf5.LF5Appender
47. log4j.appender.LF5\_APPENDER.MaxNumberOfRecords=2000
48. ########################
49. # SMTP Appender
50. #######################
51. log4j.appender.MAIL=org.apache.log4j.net.SMTPAppender
52. log4j.appender.MAIL.Threshold=FATAL
53. log4j.appender.MAIL.BufferSize=10
54. log4j.appender.MAIL.From=chenyl@yeqiangwei.com
55. log4j.appender.MAIL.SMTPHost=mail.hollycrm.com
56. log4j.appender.MAIL.Subject=Log4J Message
57. log4j.appender.MAIL.To=chenyl@yeqiangwei.com
58. log4j.appender.MAIL.layout=org.apache.log4j.PatternLayout
59. log4j.appender.MAIL.layout.ConversionPattern=[framework] %d - %c -%-4r [%t] %-5p %c %x - %m%n
60. ########################
61. # JDBC Appender
62. #######################
63. log4j.appender.DATABASE=org.apache.log4j.jdbc.JDBCAppender
64. log4j.appender.DATABASE.URL=jdbc:mysql://localhost:3306/test
65. log4j.appender.DATABASE.driver=com.mysql.jdbc.Driver
66. log4j.appender.DATABASE.user=root
67. log4j.appender.DATABASE.password=
68. log4j.appender.DATABASE.sql=INSERT INTO LOG4J (Message) VALUES ('[framework] %d - %c -%-4r [%t] %-5p %c %x - %m%n')
69. log4j.appender.DATABASE.layout=org.apache.log4j.PatternLayout
70. log4j.appender.DATABASE.layout.ConversionPattern=[framework] %d - %c -%-4r [%t] %-5p %c %x - %m%n
71. log4j.appender.A1=org.apache.log4j.DailyRollingFileAppender
72. log4j.appender.A1.File=SampleMessages.log4j
73. log4j.appender.A1.DatePattern=yyyyMMdd-HH'.log4j'
74. log4j.appender.A1.layout=org.apache.log4j.xml.XMLLayout
75. ###################
76. #自定义Appender
77. ###################
78. log4j.appender.im = net.cybercorlin.util.logger.appender.IMAppender
79. log4j.appender.im.host = mail.cybercorlin.net
80. log4j.appender.im.username = username
81. log4j.appender.im.password = password
82. log4j.appender.im.recipient = corlin@yeqiangwei.com
83. log4j.appender.im.layout=org.apache.log4j.PatternLayout
84. log4j.appender.im.layout.ConversionPattern =[framework] %d - %c -%-4r [%t] %-5p %c %x - %m%n

三.高级使用

实验目的：

1.把FATAL级错误写入2000NT日志  
2. WARN，ERROR，FATAL级错误发送email通知管理员  
3.其他级别的错误直接在后台输出

实验步骤：  
一）输出到2000NT日志

1.把Log4j压缩包里的NTEventLogAppender.dll拷到WINNT\SYSTEM32目录下  
2.写配置文件log4j.properties

**Txt代码  [收藏代码](javascript:void())**

1. # 在2000系统日志输出
2. log4j.logger.NTlog=FATAL, A8
3. # APPENDER A8
4. log4j.appender.A8=org.apache.log4j.nt.NTEventLogAppender
5. log4j.appender.A8.Source=JavaTest
6. log4j.appender.A8.layout=org.apache.log4j.PatternLayout
7. log4j.appender.A8.layout.ConversionPattern=%-4r %-5p [%t] %37c %3x - %m%n

 3.调用代码：

**Java代码  [收藏代码](javascript:void())**

1. Logger logger2 = Logger.getLogger("NTlog"); //要和配置文件中设置的名字相同
2. logger2.debug("debug!!!");
3. logger2.info("info!!!");
4. logger2.warn("warn!!!");
5. logger2.error("error!!!");
6. //只有这个错误才会写入2000日志
7. logger2.fatal("fatal!!!");

二）发送email通知管理员：

1. 首先下载JavaMail和JAF,

<http://java.sun.com/j2ee/ja/javamail/index.html>

<http://java.sun.com/beans/glasgow/jaf.html>

在项目中引用mail.jar和activation.jar。

2. 写配置文件

**Txt代码  [收藏代码](javascript:void())**

1. # 将日志发送到email
2. log4j.logger.MailLog=WARN,A5
3. #  APPENDER A5
4. log4j.appender.A5=org.apache.log4j.net.SMTPAppender
5. log4j.appender.A5.BufferSize=5
6. log4j.appender.A5.To=chunjie@yeqiangwei.com
7. log4j.appender.A5.From=error@yeqiangwei.com
8. log4j.appender.A5.Subject=ErrorLog
9. log4j.appender.A5.SMTPHost=smtp.263.net
10. log4j.appender.A5.layout=org.apache.log4j.PatternLayout
11. log4j.appender.A5.layout.ConversionPattern=%-4r %-5p [%t] %37c %3x - %m%n

 3.调用代码：

**Java代码  [收藏代码](javascript:void())**

1. //把日志发送到mail
2. Logger logger3 = Logger.getLogger("MailLog");
3. logger3.warn("warn!!!");
4. logger3.error("error!!!");
5. logger3.fatal("fatal!!!");

三）在后台输出所有类别的错误：

1. 写配置文件

**Txt代码  [收藏代码](javascript:void())**

1. # 在后台输出
2. log4j.logger.console=DEBUG, A1
3. # APPENDER A1
4. log4j.appender.A1=org.apache.log4j.ConsoleAppender
5. log4j.appender.A1.layout=org.apache.log4j.PatternLayout
6. log4j.appender.A1.layout.ConversionPattern=%-4r %-5p [%t] %37c %3x - %m%n

2．调用代码

**Java代码  [收藏代码](javascript:void())**

1. Logger logger1 = Logger.getLogger("console");
2. logger1.debug("debug!!!");
3. logger1.info("info!!!");
4. logger1.warn("warn!!!");
5. logger1.error("error!!!");
6. logger1.fatal("fatal!!!");

--------------------------------------------------------------------  
全部配置文件：log4j.properties

**Txt代码  [收藏代码](javascript:void())**

1. # 在后台输出
2. log4j.logger.console=DEBUG, A1
3. # APPENDER A1
4. log4j.appender.A1=org.apache.log4j.ConsoleAppender
5. log4j.appender.A1.layout=org.apache.log4j.PatternLayout
6. log4j.appender.A1.layout.ConversionPattern=%-4r %-5p [%t] %37c %3x - %m%n
7. # 在2000系统日志输出
8. log4j.logger.NTlog=FATAL, A8
9. # APPENDER A8
10. log4j.appender.A8=org.apache.log4j.nt.NTEventLogAppender
11. log4j.appender.A8.Source=JavaTest
12. log4j.appender.A8.layout=org.apache.log4j.PatternLayout
13. log4j.appender.A8.layout.ConversionPattern=%-4r %-5p [%t] %37c %3x - %m%n
14. # 将日志发送到email
15. log4j.logger.MailLog=WARN,A5
16. #  APPENDER A5
17. log4j.appender.A5=org.apache.log4j.net.SMTPAppender
18. log4j.appender.A5.BufferSize=5
19. log4j.appender.A5.To=chunjie@yeqiangwei.com
20. log4j.appender.A5.From=error@yeqiangwei.com
21. log4j.appender.A5.Subject=ErrorLog
22. log4j.appender.A5.SMTPHost=smtp.263.net
23. log4j.appender.A5.layout=org.apache.log4j.PatternLayout
24. log4j.appender.A5.layout.ConversionPattern=%-4r %-5p [%t] %37c %3x - %m%n

全部代码：Log4jTest.java

**Java代码  [收藏代码](javascript:void())**

1. /\*
2. \* 创建日期 2003-11-13
3. \*/
4. **package** edu.bcu.Bean;
5. **import** org.apache.log4j.\*;
6. //import org.apache.log4j.nt.\*;
7. //import org.apache.log4j.net.\*;
8. /\*\*
9. \* @author yanxu
10. \*/
11. **public** **class** Log4jTest
12. {
13. **public** **static** **void** main(String args[])
14. {
15. PropertyConfigurator.configure("log4j.properties");
16. //在后台输出
17. Logger logger1 = Logger.getLogger("console");
18. logger1.debug("debug!!!");
19. logger1.info("info!!!");
20. logger1.warn("warn!!!");
21. logger1.error("error!!!");
22. logger1.fatal("fatal!!!");
23. //在NT系统日志输出
24. Logger logger2 = Logger.getLogger("NTlog");
25. //NTEventLogAppender nla = new NTEventLogAppender();
26. logger2.debug("debug!!!");
27. logger2.info("info!!!");
28. logger2.warn("warn!!!");
29. logger2.error("error!!!");
30. //只有这个错误才会写入2000日志
31. logger2.fatal("fatal!!!");
32. //把日志发送到mail
33. Logger logger3 = Logger.getLogger("MailLog");
34. //SMTPAppender sa = new SMTPAppender();
35. logger3.warn("warn!!!");
36. logger3.error("error!!!");
37. logger3.fatal("fatal!!!");
38. }
39. }