# Log4j的配置和使用

Log4j的简介：

log4j是一个很强大的日志记录工具，可以在系统中记录一些操作信息，帮助管理人员后台分析与管理相关的数据。可以将你需要记录的数据控制台、文件、接口服务器、GUI组件。用户可以通过配置，也可以通过自己的配置，灵活的记录日志格式，从而达到数据分析、系统分析的效果。好了，废话少说，收先配置一个简单的log4j的日志系统。

Linux公社（LinuxIDC.com）于2006年9月25日注册并开通网站，Linux现在已经成为一种广受关注和支持的一种操作系统，IDC是互联网数据中心，LinuxIDC就是关于Linux的数据中心。

Linux公社是专业的Linux系统门户网站，实时发布最新Linux资讯，包括Linux、Ubuntu、Fedora、RedHat、红旗Linux、Linux教程、Linux认证、SUSE Linux、Android、Oracle、Hadoop等技术。

并被收录到Google 网页目录-计算机 > 软件 > 操作系统 > Linux 目录下。

Linux公社（LinuxIDC.com）设置了有一定影响力的Linux专题栏目。

包括：

[Ubuntu专题](http://www.linuxidc.com/topicnews.aspx?tid=2)

[Fedora专题](http://www.linuxidc.com/topicnews.aspx?tid=5)

[Android专题](http://www.linuxidc.com/topicnews.aspx?tid=11)

[Oracle专题](http://www.linuxidc.com/topicnews.aspx?tid=12)

[Hadoop专题](http://www.linuxidc.com/topicnews.aspx?tid=13)

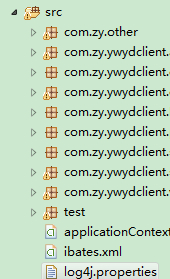
[RedHat专题](http://www.linuxidc.com/topicnews.aspx?tid=10)

[SUSE专题](http://www.linuxidc.com/topicnews.aspx?tid=3)

[红旗Linux专题](http://www.linuxidc.com/topicnews.aspx?tid=1)

## Log4j的简单使用：

1. 首先，导入log4j的jar包：log4j-1.2.16.jar 包，放到lib下面就好了。
2. 配置log4j的文件，名字为log4j.properties，将这个文件放到src目录下面就好，如下图：



1. 配置log4j文件，具体代码如下：

log4j.rootLogger=debug,B

log4j.appender.B = org.apache.log4j.DailyRollingFileAppender

log4j.appender.B.File = G:/test/error.log

log4j.appender.B.DatePattern='.'yyyy-MM-dd

log4j.appender.B.Append = true

log4j.appender.B.Threshold = ERROR

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

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

log4j.appender.B.MaxFileSize= 1024KB

注释：

log4j.rootLogger=debug,B //这是一个全局变量，是必须要配置的 B是这个别名，很明显，在下面可以看到这个别名的用法。

log4j.appender.B = org.apache.log4j.DailyRollingFileAppender//表示输出方式，表示每天产生一个日志文件

log4j.appender.B.File = G:/test/error.log //表示将日志输出到G盘符下面的test文件夹中。这样，再利用的时候，就会在G盘的test文件夹下面，有log日志了。

log4j.appender.B.DatePattern='.'yyyy-MM-dd//表示在文件的后面添加一个日期格式的名字，这样，每天生成的日志，都会在后面有一个日期的名字了。

log4j.appender.B.Append = true //表示是追加的，不是重写的，如果不设置的话，一个log日志中，可能就会只出现一条日志记录。

log4j.appender.B.Threshold = ERROR //表示日志的级别，log4j一共有5个级别，这个后面有注释。

log4j.appender.B.layout = org.apache.log4j.PatternLayout //表示可以灵活的制定布局模式

log4j.appender.B.layout.ConversionPattern = %n%d%p[%c]-%m //布局模式，日志以种种格式数出来的。2013-12-17 19:29:34,984ERROR[errorlog]-msg

log4j.appender.B.MaxFileSize= 1024KB //表示最大的日志文件时多少，当超过这个限制的时候，就创建一个新的文件

1. 编写java

Java代码如下：

**package** com.zy.other;

**import** org.apache.log4j.Logger;

**import** org.apache.log4j.PropertyConfigurator;

**publicclass** Log {

//Logger实例

**public** Logger loger;

//将Log类封装成单实例的模式，独立于其他类。以后要用到日志的地方只要获得Log的实例就可以方便使用

**privatestatic** Log *errorlog*;

**privatestatic** String *str*;

//构造函数，用于初始化Logger配置需要的属性

**private** Log()

{

//获得当前目录路径

String filePath=**this**.getClass().getResource("/").getPath();

//找到log4j.properties配置文件所在的目录(已经创建好)

filePath=filePath.substring(1).replace("bin", "src");

//获得日志类loger的实例

loger=Logger.*getLogger*(**this**.getClass());

//loger所需的配置文件路径

PropertyConfigurator.*configure*("/"+filePath+"log4j.properties");

*str* = filePath;

}

**publicstatic** Log getLoger()

{

**if**(*errorlog*!=**null**)

**return***errorlog*;

**else**

**returnnew** Log();

}

//测试函数

**publicstaticvoid** main(String args[])

{

Log log=Log.*getLoger*();

System.*out*.println(*str*);

log.loger.error("msg");

}

**publicstaticvoid** writelog(Exception e){

Log log=Log.*getLoger*();

System.*out*.println(*str*);

log.loger.error(e.getMessage(), e);

}

}

运行一下主函数，就会在你的对应的文件创建一个log日志了。

注意：上面的每天生成一个备份文件，当天的是没有日期的，只有到了第二天，才会出现上一天的日志文件。

## Log4j的多文件配置

有的时候，日志不可能只是记录一个事情，可能还要记录其他的义务数据，但是又不能记录到一个文件中，如果这样的话，就太乱了，最好就是分类记录，这样的话，想看什么数据，就看什么文件就好了啊。Log4j这么使用也很简单：

首先，看一下log4j的配置文件

log4j.rootLogger=INFO

log4j.logger.errorlog= ERROR,error

log4j.logger.paylog = INFO,pay

log4j.appender.error = org.apache.log4j.DailyRollingFileAppender

log4j.appender.error.File = G:/test/error.log

log4j.appender.error.DatePattern='.'yyyy-MM-dd

log4j.appender.error.Append = true

log4j.appender.error.Threshold = ERROR

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

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

log4j.appender.errorlog.MaxFileSize=10240KB

log4j.appender.pay = org.apache.log4j.DailyRollingFileAppender

log4j.appender.pay.File = G:/test/paylog.log

log4j.appender.pay.DatePattern='.'yyyy-MM-dd

log4j.appender.pay.Append = true

log4j.appender.pay.Threshold = INFO

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

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

log4j.appender.paylog.MaxFileSize=1024000KB

上面的配置，跟一开始的配置差不多，就是多定义了两个 logger，在后台记录的时候，根据这个logger获取记录日志源，然后分别取记录到不同的文件中

下面是java代码：

**package** com.zy.ywyd.gameoperators.until;

**import**org.apache.commons.logging.LogFactory;

**import** org.apache.log4j.Logger;

**import** org.apache.log4j.PropertyConfigurator;

**publicclass** Log {

//Logger实例

**public** Logger loger;

**public** Logger errrorloger;

**public** Logger paylogger;

//将Log类封装成单实例的模式，独立于其他类。以后要用到日志的地方只要获得Log的实例就可以方便使用

**privatestatic** Log *log*;

**privatestatic** String *str*;

//构造函数，用于初始化Logger配置需要的属性

**private** Log()

{

//获得当前目录路径

String filePath=**this**.getClass().getResource("/").getPath();

//找到log4j.properties配置文件所在的目录(已经创建好)

filePath=filePath.substring(1).replace("bin", "src");

//获得日志类loger的实例

loger=Logger.*getLogger*(**this**.getClass());

errrorloger = Logger.*getLogger*("errorlog");

paylogger = Logger.*getLogger*("paylog");

//loger所需的配置文件路径

PropertyConfigurator.*configure*("/"+filePath+"log4j.properties");

*str* = filePath;

}

**publicstatic** Log getLoger()

{

**if**(*log*!=**null**)

**return***log*;

**else**

**returnnew** Log();

}

//测试函数

**publicstaticvoid** main(String args[])

{

Log logc = Log.*getLoger*();

logc.errrorloger.error("error,error,errorqqq");

logc.paylogger.info("info info info wwww");

}

//记录错误日志

**publicstaticvoid** errorLog(Exception e){

Log log=Log.*getLoger*();

System.*out*.println(*str*);

log.errrorloger.error(e.getMessage(), e);

}

//记录回复日志

**publicstaticvoid** payLog(String msg){

Log log=Log.*getLoger*();

System.*out*.println(*str*);

log.paylogger.error(msg);

}

}

这样的话，直接调用main主函数应该就会生成相应的文件了。

备注：

#appender名字定义  
log4j.rootLogger=INFO,FILE,consoleAppender,RollingFile,MAIL,DailyRollingFile,DATABASE  
log4j.addivity.org.apache=true

#配置日志信息输出目的地  
#log4j.appender.appenderName = fully.qualified.name.of.appender.class  
#(如：log4j.appender.RollingFile = org.apache.log4j.RollingFileAppender)  
# 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  
#(如：log4j.appender.RollingFile.layout = org.apache.log4j.HTMLLayout)  
# 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）

#日记记录的优先级：priority，优先级由高到低分为   
# OFF ,FATAL ,ERROR ,WARN ,INFO ,DEBUG ,ALL。  
# Log4j建议只使用FATAL ,ERROR ,WARN ,INFO ,DEBUG这五个级别。  
# 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

#输出格式：layout中的参数都以%开始，后面不同的参数代表不同的格式化信息（参数按字母表顺序列出）：  
# %c 输出所属类的全名，可在修改为 %d{Num} ,Num类名输出的维（如："org.apache.elathen.ClassName",%C{2}将输出elathen.ClassName）  
# %d 输出日志时间其格式为 %d{yyyy-MM-dd HH:mm:ss,SSS}，可指定格式 如 %d{HH:mm:ss}  
# %l 输出日志事件发生位置，包括类目名、发生线程，在代码中的行数  
# %n 换行符  
# %m 输出代码指定信息，如info(“message”),输出message  
# %p 输出优先级，即 FATAL ,ERROR 等  
# %r 输出从启动到显示该log信息所耗费的毫秒数  
# %t 输出产生该日志事件的线程名

#ConsoleAppender，控制台输出  
#FileAppender，文件日志输出  
#SMTPAppender，发邮件输出日志  
#SocketAppender，Socket日志  
#NTEventLogAppender，Window NT日志  
#SyslogAppender，  
#JMSAppender，  
#AsyncAppender，  
#NullAppender

############################  
# 文件输出：FileAppender   
############################   
log4j.appender.FILE=org.apache.log4j.FileAppender   
log4j.appender.FILE.File=../logs/file.log   
log4j.appender.FILE.Append=false   
log4j.appender.FILE.layout=org.apache.log4j.PatternLayout   
log4j.appender.FILE.layout.ConversionPattern=[framework] %d - %c -%-4r [%t] %-5p %c %x - %m%n   
# Use this layout for LogFactor 5 analysis

############################################  
# 滚动文件输出：RollingFileAppender  
############################################  
log4j.appender.RollingFile = org.apache.log4j.RollingFileAppender  
# 输出以上的INFO信息   
log4j.appender.RollingFile.Threshold = DEBUG   
#保存log文件路径   
log4j.appender.RollingFile.File = ../logs/chenya.html   
#默认为true，添加到末尾，false在每次启动时进行覆盖   
log4j.appender.RollingFile.Append = true   
#一个log文件的大小，KB、MB、GB表示. 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chenya.html.1文件。  
log4j.appender.RollingFile.MaxFileSize = 1MB   
#最多保存3个滚动文件备份（chenya.html.1、chenya.html.2、chenya.html.3）  
log4j.appender.RollingFile.MaxBackupIndex = 3   
#输出文件的格式   
log4j.appender.RollingFile.layout = org.apache.log4j.HTMLLayout   
#是否显示类名和行数  
log4j.appender.RollingFile.layout.LocationInfo = true   
#html页面的 < title >   
log4j.appender.RollingFile.layout.Title =title:\u63d0\u9192\u60a8\uff1a\u7cfb\u7edf\u53d1\u751f\u4e86\u4e25\u91cd\u9519\u8bef   
############################## SampleLayout ####################################  
# log4j.appender.RollingFile.layout = org.apache.log4j.SampleLayout  
############################## PatternLayout ###################################  
# log4j.appender.RollingFile.layout = org.apache.log4j.PatternLayout  
# log4j.appender.RollingFile.layout.ConversionPattern =% d % p [ % c] - % m % n % d  
############################## XMLLayout #######################################  
# log4j.appender.RollingFile.layout = org.apache.log4j.XMLLayout  
# log4j.appender.RollingFile.layout.LocationInfo = true #是否显示类名和行数  
############################## TTCCLayout ######################################  
# log4j.appender.RollingFile.layout = org.apache.log4j.TTCCLayout  
# log4j.appender.RollingFile.layout.DateFormat = ISO8601  
#NULL, RELATIVE, ABSOLUTE, DATE or ISO8601.  
# log4j.appender.RollingFile.layout.TimeZoneID = GMT - 8 : 00   
# log4j.appender.RollingFile.layout.CategoryPrefixing = false ##默认为true 打印类别名  
# log4j.appender.RollingFile.layout.ContextPrinting = false ##默认为true 打印上下文信息  
# log4j.appender.RollingFile.layout.ThreadPrinting = false ##默认为true 打印线程名  
# 打印信息如下：  
#2007 - 09 - 13 14 : 45 : 39 , 765 [http - 8080 - 1 ] ERROR com.poxool.test.test - error成功关闭链接  
###############################################################################

############################################  
# 每天文件的输出：DailyRollingFileAppender  
############################################  
log4j.appender.DailyRollingFile = org.apache.log4j.DailyRollingFileAppender  
log4j.appender.DailyRollingFile.Threshold = DEBUG  
log4j.appender.DailyRollingFile.File = ../logs/cheny.log  
log4j.appender.DailyRollingFile.Append = true   
#默认为true，添加到末尾，false在每次启动时进行覆盖  
log4j.appender.DailyRollingFile.ImmediateFlush = true   
#直接输出，不进行缓存  
# ' . ' yyyy - MM: 每个月更新一个log日志  
# ' . ' yyyy - ww: 每个星期更新一个log日志  
# ' . ' yyyy - MM - dd: 每天更新一个log日志  
# ' . ' yyyy - MM - dd - a: 每天的午夜和正午更新一个log日志  
# ' . ' yyyy - MM - dd - HH: 每小时更新一个log日志  
# ' . ' yyyy - MM - dd - HH - mm: 每分钟更新一个log日志  
log4j.appender.DailyRollingFile.DatePattern = ' . ' yyyy - MM - dd ' .log '   
#文件名称的格式  
log4j.appender.DailyRollingFile.layout = org.apache.log4j.PatternLayout  
log4j.appender.DailyRollingFile.layout.ConversionPattern =%d %p [ %c] - %m %n %d

##############################  
# 控制台输出：ConsoleAppender  
##############################  
log4j.appender.consoleAppender = org.apache.log4j.ConsoleAppender  
log4j.appender.consoleAppender.Threshold = DEBUG  
log4j.appender.consoleAppender.layout = org.apache.log4j.PatternLayout  
log4j.appender.consoleAppender.layout.ConversionPattern =%d %-5p %m %n  
log4j.appender.consoleAppender.ImmediateFlush = true  
# 直接输出，不进行缓存   
log4j.appender.consoleAppender.Target = System.err   
# 默认是System.out方式输出

########################  
# 发送邮件：SMTPAppender  
########################  
# log4j的邮件发送appender，如果有必要你可以写自己的appender  
log4j.appender.MAIL=org.apache.log4j.net.SMTPAppender  
#设置发送邮件的门槛，仅当等于或高于ERROR（比如FATAL）时，邮件才被发送  
log4j.appender.MAIL.Threshold=DEBUG  
#邮件缓冲区大小 日志达到10K时发送Email  
log4j.appender.MAIL.BufferSize=10  
#发送邮件的邮箱帐号  
[log4j.appender.MAIL.From=mail@163.com](mailto:log4j.appender.MAIL.From=mail@163.com)  
#SMTP邮件发送服务器地址  
log4j.appender.MAIL.SMTPHost=smtp.163.com  
#SMTP发送认证的帐号名  
[log4j.appender.MAIL.SMTPUsername=mail@163.com](mailto:log4j.appender.MAIL.SMTPUsername=pesport@163.com)  
#SMTP发送认证帐号的密码  
log4j.appender.MAIL.SMTPPassword=password  
#是否打印调试信息，如果选true，则会输出和SMTP之间的握手等详细信息  
log4j.appender.MAIL.SMTPDebug=true  
#邮件主题  
log4j.appender.MAIL.Subject= 警告 log4J Message  
#发送到什么邮箱，如果要发送给多个邮箱，则用逗号分隔；  
#如果需要bcc给某人，则加入下列行：  
[log4j.appender.MAIL.Bcc=mail@163.com](mailto:log4j.appender.MAIL.Bcc=pesport@163.com)  
[log4j.appender.MAIL.To=mail@163.com](mailto:log4j.appender.MAIL.To=pesport@163.com)  
#设置输出样式和内容  
log4j.appender.MAIL.layout=org.apache.log4j.PatternLayout  
log4j.appender.MAIL.layout.ConversionPattern =%d - %c -%-4r [%t] %-5p %c %x - %m %n

######################  
#数据库：JDBCAppender  
######################  
log4j.appender.DATABASE.Threshold = ERROR  
log4j.appender.DATABASE = org.apache.log4j.jdbc.JDBCAppender  
log4j.appender.DATABASE.URL = jdbc:mysql://localhost:3306/db\_oa  
log4j.appender.DATABASE.driver = com.mysql.jdbc.Driver  
log4j.appender.DATABASE.user = root  
log4j.appender.DATABASE.password = 123456  
#log4j.appender.DATABASE.sql = INSERT INTO tb\_log(text) VALUES( ' %d - %c %-5p %c %x - %m%n ' )  
log4j.appender.DATABASE.sql = INSERT INTO tb\_log(text) VALUES('%d,%-5p,%C,%L,%x')   
log4j.appender.DATABASE.layout = org.apache.log4j.PatternLayout  
log4j.appender.DATABASE.layout.ConversionPattern =% d - % c -%- 4r [ % t] %- 5p % c % x - % m % n  
#数据库的链接会有问题，可以重写org.apache.log4j.jdbc.JDBCAppender的getConnection() 使用数据库链接池去得链接，可以避免insert一条就链接一次数据库  
#log4j官方建议把其提供的JDBCAppender作为基类来使用，然后Override三个父类的方法：getConnection(),closeConnection(Connection con)和getLogStatement(LoggingEvent event)。  
#原来如此，那就写一个子类JDBCPoolAppender来替代这个JDBCAppender

####################   
# Socket Appender   
####################   
#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 Facter 5   
#log4j.appender.SOCKET.layout=org.apache.log4j.PatternLayout   
#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

level指大于该级别的日志才显示， error>warn>info>debug  
threshold起过滤作用，即低于过滤条件的日志不显示

---------------------------------------------------------------------------------------

my

log4j.logger.com.tuan.partner.service.api=info,apiFile

log4j.appender.apiFile=org.apache.log4j.DailyRollingFileAppender  
log4j.appender.apiFile.Append=true  
#log4j.appender.apiFile.File=c:/partner-api.log  
log4j.appender.apiFile.File=/data/application/logs/partner/api/partner-api.log  
log4j.appender.apiFile.ImmediateFlush=true  
log4j.appender.apiFile.layout = org.apache.log4j.PatternLayout  
log4j.appender.apiFile.DatePattern  = '\_'yyyy-MM-dd-HH'.log'  
log4j.appender.apiFile.layout.ConversionPattern = %d{yyyy-MM-dd HH\:mm\:ss} [%t] %-5p (%c\:%L) - %m%n

附录资源来自：<http://my.oschina.net/heguangdong/blog/13679> 感谢朋友的支持