一、log4j是什么?

Log4j是一个开源的日志记录组件，其产品已经相当的成熟，且使用非常的广泛。在工程中以易用，方便等代替了 System.out 等打印语句。

Log4j的目标是：它允许开发人员以任意的精细程度控制哪些日志说明被输出。通过使用外部的配置文件，可以在运行时配置它。

Log4j的具体在 http://logging.apache.org/log4j/找到它（有使用的文档说明）。另外，log4j已经被转换成 C, C++, C#, Perl, Python, Ruby, 和 Eiffel 语言。

二、通常我们建立一个log4j.properties文件放在src下，编译之后就生成在../web-inf/class/当然也可以放到其它任何目录，只要该目录被包含到类路径中即可，中定位这个文件，并读入这个文件完成的配置。这个配置文件告 诉Log4J以什么样的格式、把什么样的信息、输出到什么地方。

　　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的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。下面对三个组件分别进行说明：

　　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 Log4j有一个规则：假设Loggers级别为P，如果在Loggers中发生了一个级别Q比P高，则可以启动，否则屏蔽掉。

假设你定义的级别是info，那么error和warn的日志可以显示而比他低的debug信息就不显示了。

　　其语法表示为：

　　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）

　　配置时使用方式为：

　　log4j.appender.appenderName = fully.qualified.name.of.appender.class

　　log4j.appender.appenderName.option1 = value1

　　…

log4j.appender.appenderName.option = valueN

　　这样就为日志的输出提供了相当大的便利。

　　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提供了 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、类别等信息的样式等等。

　　其语法表示为：

　　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）

　　配置时使用方式为：

　　log4j.appender.appenderName.layout =fully.qualified.name.of.layout.class

　　log4j.appender.appenderName.layout.option1 = value1

　　…

　　log4j.appender.appenderName.layout.option = valueN

4 . Log4j的配置

　　以上是从原理方面说明Log4j的使用方法，在具体Java编程使用Log4j可以参照以下示例：

　　1、 建立Logger实例：

　　语法表示：public static Logger getLogger( String name)

　　实际使用：static Logger logger = Logger.getLogger(ServerWithLog4j.class.getName ()) ;

　　2、 读取配置文件：

　　获得了Logger的实例之后，接下来将配置Log4j使用环境：

　　语法表示：

或者直接放在src下把log4j.properties文件

　　BasicConfigurator.configure()：自动快速地使用缺省Log4j环境。

　　PropertyConfigurator.configure(String configFilename)：读取使用Java的特性文件编写的配置文件。

　　DOMConfigurator.configure(String filename)：读取XML形式的配置文件。

　　实际使用：

PropertyConfigurator.configure("ServerWithLog4j.properties");

　　3、 插入日志信息

　　完成了以上连个步骤以后，下面就可以按日志的不同级别插入到你要记录日志的任何地方了。

　　语法表示：

　　Logger.debug(Object message);//调试信息

　　Logger.info(Object message);//一般信息

　　Logger.warn(Object message);//警告信息

　　Logger.error(Object message);//错误信息

　　Logger.fatal(Object message);//致命错误信息

　　实际使用：logger.info("ServerSocket before accept: " + server);

　 5. 配置过程 Log4j真正在系统中运行事先还要对配置文件进行定义。定义步骤就是对Logger、Appender及Layout的分别使用。

Log4j支持两种配置文件格式，一种是XML格式的文件，一种是java properties（key=value）【Java特性文件（键=值）】。下面我们介绍使用Java特性文件做为配置文件的方法

具体如下：

三、配置根Logger，其语法为：  
　　log4j.rootLogger = [ level ] , appenderName1, appenderName2, …  
level : 是日志记录的优先级，分为OFF、FATAL、ERROR、WARN、INFO、DEBUG、ALL或者您定义的级别。Log4j建议只使用四个级别，优先级从高到低分别是ERROR、WARN、INFO、DEBUG。通过在这里定义的级别，您可以控制到应用程序中相应级别的日志信息的开关。比如在这里定 义了INFO级别，则应用程序中所有DEBUG级别的日志信息将不被打印出来。  
　　 appenderName:就是指定日志信息输出到哪个地方。您可以同时指定多个输出目的地。  
例如：log4j.rootLogger＝info,A1,B2,C3  
　　  
　　2、配置日志信息输出目的地，其语法为：  
　　log4j.appender.appenderName = fully.qualified.name.of.appender.class //  
　　 "fully.qualified.name.of.appender.class" 可以指定下面五个目的地中的一个：  
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  
2.FileAppender 选项  
Threshold=WARN: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。  
3.DailyRollingFileAppender 选项  
Threshold=WARN: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。  
DatePattern='.'yyyy-ww:每周滚动一次文件，即每周产生一个新的文件。当然也可以指定按月、周、天、时和分。即对应的格式如下：  
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。  
实际应用：  
　　log4j.appender.A1=org.apache.log4j.ConsoleAppender //这里指定了日志输出的第一个位置A1是控制台ConsoleAppender  
　　  
　　3、配置日志信息的格式，其语法为：  
　　A. log4j.appender.appenderName.layout = fully.qualified.name.of.layout.class  
"fully.qualified.name.of.layout.class" 可以指定下面4个格式中的一个：  
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:默认值是false,输出java文件名称和行号  
Title=my app file: 默认值是 Log4J Log Messages.  
2.PatternLayout 选项  
ConversionPattern=%m%n :指定怎样格式化指定的消息。  
3.XMLLayout 选项  
LocationInfo=true:默认值是false,输出java文件和行号  
实际应用：  
　　log4j.appender.A1.layout=org.apache.log4j.PatternLayout  
B. log4j.appender.A1.layout.ConversionPattern=%-4r %-5p %d{yyyy-MM-dd HH:mm:ssS} %c %m%n  
这里需要说明的就是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，就从左边交远销出的字符截掉。  
　　这里上面三个步骤是对前面Log4j组件说明的一个简化；下面给出一个具体配置例子，在程序中可以参照执行：  
　　log4j.rootLogger=INFO,A1，B2  
　　log4j.appender.A1=org.apache.log4j.ConsoleAppender  
　　log4j.appender.A1.layout=org.apache.log4j.PatternLayout  
　　log4j.appender.A1.layout.ConversionPattern=%-4r %-5p %d{yyyy-MM-dd HH:mm:ssS} %c %m%n  
　　根据上面的日志格式，某一个程序的输出结果如下：  
　　0　　INFO　2003-06-13 13:23:46968 ClientWithLog4j Client socket: Socket[addr=localhost/127.0.0.1,port=8002,localport=2014]  
16　 DEBUG 2003-06-13 13:23:46984 ClientWithLog4j Server says: 'Java server with log4j, Fri Jun 13 13:23:46 CST 2003'  
　　16　 DEBUG 2003-06-13 13:23:46984 ClientWithLog4j GOOD  
　　16　 DEBUG 2003-06-13 13:23:46984 ClientWithLog4j Server responds: 'Command 'HELLO' not understood.'  
　　16　 DEBUG 2003-06-13 13:23:46984 ClientWithLog4j HELP  
　　16　 DEBUG 2003-06-13 13:23:46984 ClientWithLog4j Server responds: 'Vocabulary: HELP QUIT'  
　　16　 DEBUG 2003-06-13 13:23:46984 ClientWithLog4j QUIT

4. # 当输出信息于回滚文件时  
log4j.appender.ROLLING\_FILE=org.apache.log4j.RollingFileAppender //指定以文件的方式输出日志  
log4j.appender.ROLLING\_FILE.Threshold=ERROR   
log4j.appender.ROLLING\_FILE.File=rolling.log //文件位置,也可以用变量${java.home}、rolling.log  
log4j.appender.ROLLING\_FILE.Append=true   
log4j.appender.ROLLING\_FILE.MaxFileSize=10KB //文件最大尺寸  
log4j.appender.ROLLING\_FILE.MaxBackupIndex=1 //备份数  
log4j.appender.ROLLING\_FILE.layout=org.apache.log4j.PatternLayout   
log4j.appender.ROLLING\_FILE.layout.ConversionPattern=[framework] %d - %c -%-4r [%t] %-5p %c %x - %m%n

四、具体应用文件配置举例

#所有包下的日志信息都输出到下面的日志路径下

log4j.rootLogger=INFO,trycatch

log4j.appender.trycatch=org.apache.log4j.RollingFileAppender

log4j.appender.trycatch.File=d:/loginfo/trycatch.log

log4j.appender.trycatch.MaxFileSize=500KB

log4j.appender.trycatch.MaxBackupIndex=1

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

log4j.appender.trycatch.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss,SSS} [%t] [%c] [%p] - %m%n

#指定目录下包下的日志信息都存放到该目录下

log4j.logger.cn.com.cuigq.Theard=DEBUG,R1

log4j.appender.R1=org.apache.log4j.RollingFileAppender

log4j.appender.R1.File=d:/loginfo/iamSystem1.log

log4j.appender.R1.MaxFileSize=500KB

log4j.appender.R1.MaxBackupIndex=1

log4j.appender.R1.layout=org.apache.log4j.PatternLayout

log4j.appender.R1.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss,SSS} [%t] [%c] [%p] - %m%n

#指定目录下的java类的日志信息都存放到该目录下

log4j.logger.com.cuigq.TestError=DEBUG,R2

log4j.appender.R2=org.apache.log4j.DailyRollingFileAppender

log4j.appender.R2.File=d:/loginfo/html.html

#log4j.appender.R2.MaxFileSize=500KB

#log4j.appender.R2.MaxBackupIndex=1

log4j.appender.R2.layout=org.apache.log4j.HTMLLayout

#log4j.appender.R2.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss,SSS} [%t] [%c] [%p] - %m%n

#指定目录下包下的日志信息都存放到数据库

log4j.logger.database=DEBUG,oracledb

log4j.appender.oracledb=org.apache.log4j.jdbc.JDBCAppender

log4j.appender.oracledb.Driver=oracle.jdbc.driver.OracleDriver

log4j.appender.oracledb.URL=jdbc:oracle:thin:@192.168.20.92:1521:chshsid（切忌此处为sid而不是数据库名）

log4j.appender.oracledb.user=scott

log4j.appender.oracledb.password=tiger

#log4j.appender.encoding=utf-8

log4j.appender.oracledb.sql=INSERT INTO enter\_ora\_log4j VALUES ('%d{yyyy-MM-dd HH:mm:ss}', '%t', '%p', '%l', '%m')

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