Log4j知识

1. **入门实例**

**第一步：只能在src目录新建一个名称为log4j2.xml的文件，有两种配置文件，一种是XML另外一种是properties文件**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<configuration status=*"OFF"*>

<appenders>

<!--这个输出控制台的配置-->

<!-- <Console name="Console" target="SYSTEM\_OUT"> -->

<!-- 这个都知道是输出日志的格式 -->

<!-- <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"/> -->

<!-- </Console> -->

<!--文件会打印出所有信息，这个log每次运行程序会自动清空，由append属性决定，这个也挺有用的，适合临时测试用-->

<File name=*"file"* fileName=*"log/test.log"* append=*"false"*>

<!--控制台只输出level及以上级别的信息（error），其他的直接拒绝-->

<ThresholdFilter level=*"error"* onMatch=*"ACCEPT"* onMismatch=*"DENY"*/>

<PatternLayout pattern=*"%d{yyyy-MM-dd HH:mm:ss.SSS} %-5level %class{36} %L %M - %msg%xEx%n"*/>

</File>

<!--这个会打印出所有的信息，每次大小超过size，则这size大小的日志会自动存入按年份-月份建立的文件夹下面并进行压缩，作为存档-->

<!-- <RollingFile name="RollingFile" fileName="app.log"

filePattern="log/$${date:yyyy-MM}/app-%d{MM-dd-yyyy}-%i.log.gz">

<PatternLayout

pattern="%d{yyyy-MM-dd 'at' HH:mm:ss z} %-5level %class{36} %L %M - %msg%xEx%n" />

<SizeBasedTriggeringPolicy size="500MB" />

</RollingFile> -->

</appenders>

<!--然后定义logger，只有定义了logger并引入的appender，appender才会生效-->

<loggers>

<logger name=*"com.gh.log4j.TestLog4j"* level=*"error"* additivity=*"false"*>

<!-- <appender-ref ref="Console"/> -->

<appender-ref ref=*"file"*/>

</logger>

<root level=*"error"*>

<!--声明日志输出到控制台-->

<!-- <appender-ref ref="Console"/> -->

<!--声明日志输出的位置，这里输出到文件，这里的file与<File>标签里面定义的name属性值相同。-->

<appender-ref ref=*"file"*/>

</root>

</loggers>

</configuration>

第二种：properties文件的配置

# 设置日志的输出级别 等级可分为OFF、 FATAL、ERROR、WARN、INFO、DEBUG、ALL

# 如果设置info 则会打印info及其级别以上的信息，而debug信息则不会打印，如果配置OFF则不打出任何信息

# 设置完日志打印等级后，设置输出的方式，这里console是输出到控制台，file是输出到文件

log4j.rootLogger = debug,console,file

#其中，Log4j提供的appender有以下几种：

#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），

# 可通过 log4j.appender.R.MaxFileSize=100KB设置文件大小，还可通过 log4j.appender.R.MaxBackupIndex=1设置为保存一个备份文件。

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

log4j.appender.console=org.apache.log4j.ConsoleAppender

#设置日志输出信息包含的内容，在设置日志输出格式后，我们还可以在这基础上控制日志输出的内容，比如日志输出格式设置了输出日期的格式，在这里，如果我们设置

#为SimpleLayout，则只会输出简单的日志信息，而不会包含日期。如下提供了四种信息内容布局

#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.console.layout=org.apache.log4j.PatternLayout

#设置控制台日志的格式

log4j.appender.console.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss.SSS} %l %M - %m%n

#表示文件大小到达指定尺寸的时候产生一个新的文件，这里设置一个文件为100MB，达到100MB后产生一个新文件

log4j.appender.file=org.apache.log4j.RollingFileAppender

#设置日志文件打印的路径

log4j.appender.file.File=log/test.log

#设置每个文件的大小

log4j.appender.file.MaxFileSize=100MB

#设置日志文件的数量，这里设置10个日志文件，如果日志文件都满了，则不会继续打印日志信息

log4j.appender.file.MaxBackupIndex=10

#设置日志文件的输出格式，这里只是设置日志文件输出的格式，不会影响控制台日志的格式

#%m 输出代码中指定的消息

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

#%r 输出自应用启动到输出该log信息耗费的毫秒数

#%c 输出所属的类目，通常就是所在类的全名

#%t 输出产生该日志事件的线程名

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

#%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{yyyy MMM dd HH:mm:ss,SSS}，输出类似：2002年10月18日 22：10：28，921

#[QC]是log信息的开头，可以为任意字符，一般为项目简称。

#2015-04-17 15:17:23.378 com.yp.log.TestLog.main(TestLog.java:13) com.yp.log.TestLoglass 13 main - Didn't do it.sgEx

#d{yyyy-MM-dd HH:mm:ss.SSS}对应日期

#%-5l或者%l 对应 com.yp.log.TestLog.main(TestLog.java:13)

# %class对应 输出日志的类 com.yp.log.TestLoglass

# %L代表输出日志信息是该类的第几行

# %M 代表输出日志信息的是来自该类的哪个方法 这里是main函数

log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss.SSS} %l %M - %m%n

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

**第二步：导入log4j的jar文件**

必须的jar包log4j-api-2.1.jar和log4j-core-2.1.jar

**第三步：使用log4j**

**package** com.gh.log4j;

**import** org.apache.logging.log4j.LogManager;

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

**public** **class** TestLog4j {

**private** **static** **final** Logger *logger* = LogManager.*getLogger*(TestLog4j.**class**);

**public** **static** **void** main(**final** String... args) {

// Set up a simple configuration that logs on the console.

*logger*.trace("Entering application.");

// Bar bar = new Bar();

// if (!bar.doIt()) {

*logger*.error("Didn't do it.");

// }

*logger*.trace("Exiting application.");

}

}

1. **输出error日志格式**

输出error信息，并且该信息包含出现错误的代码行

**try** {

}**catch** (Exception e) {

e.printStackTrace();

*logger*.error("",e);

}

输出结果：

2014-11-18 11:44:09.673 [main] ERROR com.gh.log4j.TestLog4j - java.lang.ArithmeticException: / by zero

at com.gh.log4j.TestLog4j.main(TestLog4j.java:21) [bin/:?]

另外一种写法：

**try** {

}**catch** (Exception e) {

e.printStackTrace();

*logger*.error(e);

}

输出结果：

2014-11-18 11:44:09.673 [main] ERROR com.gh.log4j.TestLog4j - java.lang.ArithmeticException: / by zero

1. **在引用某些框架时，会影响log4j找寻配置文件**

例子1：

问题描述：正常情况下，在maven工程下，只需要把log4j.properties文件放到resources文件下，然后在eclipse环境可以直接运行，如果打成jar需要把log4j.properties放置到jar包同级目录下，这样也可以正常运行。

注：在java工程下，log4j.properties要放到src文件下