**[JAVA][log4j] 詳細設定**

**一、Log4j簡介**  
Log4j有三個主要的元件：Loggers、Appenders、Layouts。  
  
**1、Loggers(寫Log )**  
被分為五個級別：DEBUG < INFO < WARN < ERROR < FATAL  
只輸出級別不低於設定級別的訊息  
ex：Loggers為INFO，則INFO、WARN、ERROR和FATAL都會輸出 DEBUG則不會  
  
logger名稱階層關係  
ex：名稱為"com.foo"的 logger 是名稱為"com.foo.Bar"的 logger的parent  
程式中具名取得logger的方法是  
Logger.getLogger(String name)  //名稱相同取得的logger一定一樣,不同則產生新的  
通常我們用  Logger.getLogger(myApp.class) 取得loggger,名稱為程式的class  name,因此層級關係跟package層級關係吻合...  
  
有一個最上層的root logger,root logger 一定存在的,且只能以 Logger.getRootLogger()方式取得它的參考.不能以具名方式取得.  
  
**2、Appenders(log訊息的終點(目的地))**  
  
常使用的類如下：  
org.apache.log4j.ConsoleAppender（控制台）  
org.apache.log4j.FileAppender（文件）  
org.apache.log4j.DailyRollingFileAppender（每天產生一個日誌文件）  
org.apache.log4j.RollingFileAppender（檔大小到達指定size的時候產生一個新的檔）  
org.apache.log4j.WriterAppender（將日誌訊息以輸出流格式發送到任意指定的地方）  
  
如 System.out.println("Some message..."), 對Log4j 而言appenders 就是 console  
  
  
配置模式：  
log4j.appender.appenderName = className  
log4j.appender.appenderName.Option1 = value1  
…  
log4j.appender.appenderName.OptionN = valueN  
  
**3、Layouts( Log訊息的格式 )**  
  
常使用的類如下：  
  
org.apache.log4j.HTMLLayout（以HTML表格形式佈局）  
org.apache.log4j.PatternLayout（可以靈活地指定佈局模式）  
org.apache.log4j.SimpleLayout（包含日誌訊息的級別和訊息字串）  
org.apache.log4j.TTCCLayout（包含日誌產生的時間、執行緒、類別等訊息）  
  
配置模式：  
  
log4j.appender.appenderName.layout =className  
log4j.appender.appenderName.layout.Option1 = value1  
…  
log4j.appender.appenderName.layout.OptionN = valueN  
  
  
**二、簡單範例**  
  
log4j.rootLogger=Info, A1, A2  
  
  
# A1 is set to be a ConsoleAppender  
log4j.appender.A1=org.apache.log4j.ConsoleAppender  
log4j.appender.A1.layout=org.apache.log4j.PatternLayout  
log4j.appender.A1.layout.ConversionPattern=[%d{yy/MM/dd HH:mm:ss}][%p][%C-%L] %m%n  
  
# A2 is set to be a file  
log4j.appender.A2=org.apache.log4j.DailyRollingFileAppender  
log4j.appender.A2.layout=org.apache.log4j.PatternLayout  
log4j.appender.A2.layout.ConversionPattern=[%d{yy/MM/dd HH:mm:ss}][%p][%C-%L] %m%n  
log4j.appender.A2.File=./log/gpserver-${log4j.date}.log  
  
  
  
  
**三、配置詳解**  
  
設定檔事實上也就是對Logger、Appender及Layout進行相應設定。  
Log4j支援兩種設定檔格式，一種是XML格式的檔，一種是properties屬性檔。  
  
下麵以properties屬性文件為例介紹log4j.properties的配置。  
  
**1、配置根Logger：**  
log4j.rootLogger = [ level ] , appenderName1, appenderName2, …  
log4j.additivity.org.apache=false：表示Logger不會在父Logger的appender裡輸出，默認為true。  
  
level ：設定日誌記錄的最低級別，可設的值有OFF、FATAL、ERROR、WARN、INFO、DEBUG、ALL或者自定義的級別，  
ex：Loggers為INFO，則INFO、WARN、ERROR和FATAL都會輸出 DEBUG則不會  
  
appenderName：指定log訊息要輸出到哪裡。可以同時指定多個輸出目的地，用逗號隔開。  
例如：log4j.rootLogger＝INFO,A1,B2,C3  
  
**2、配置appender：**  
log4j.appender.appenderName = className  
appenderName：自定義appderName，在log4j.rootLogger設置中使用；  
className：可設值如下：  
   (1)org.apache.log4j.ConsoleAppender（控制台）  
   (2)org.apache.log4j.FileAppender（文件）  
   (3)org.apache.log4j.DailyRollingFileAppender（每天產生一個日誌文件）  
   (4)org.apache.log4j.RollingFileAppender（檔大小到達指定大小的時候產生一個新的檔）  
   (5)org.apache.log4j.WriterAppender（將日誌訊息以輸出流格式發送到任意指定的地方）  
  
  
Appender 的設定亦會反映在 Logger 的階層中。  
當 Logger 輸出一筆 log message 時,父階層的 Appender 和自己的 Appender(如果有的話)都會記錄到這筆 log message;  
  
例如"com.foo" Logger 有一個 Appender 將 log message 輸出到 console,  
而 "com.foo.bar" 有一個 Appender 將 log message 輸出到檔案;  
當 "com.foo.bar" Logger 輸出一筆 log message 時, console 和檔案都會出現這筆 log message。  
  
而最簡單的例子,就是當 root logger 擁有一個輸出到 console 的 Appender 時,  
則程式中所有的 logger 所產生的 log message 都會輸出到 console。  
唯一個例外的情況,是當某個 logger 將自己的 additivity 屬性設為 false(Logger.setAdditivity(false)),  
則此 logger 與隸屬於它的子 logger 都不會將 log message 寫到 console。  
  
  
(1)ConsoleAppender選項：  
Threshold=WARN：指定日誌訊息的最低輸出級別，預設為DEBUG。  
ImmediateFlush=true：表示所有訊息都會被立即輸出，設為false則不輸出，預設值是true。  
Target=System.err：預設值是System.out。  
  
(2)FileAppender選項：  
Threshold=WARN：指定日誌訊息的最低輸出級別，默認為DEBUG。  
ImmediateFlush=true：表示所有訊息都會被立即輸出，設為false則不輸出，預設值是true。  
Append=false：true表示訊息增加到指定檔中，false則將消息覆蓋指定的檔內容，預設值是true。  
File=D:/logs/logging.log4j：指定訊息輸出到logging.log4j文件中。  
  
(3)DailyRollingFileAppender選項：  
Threshold=WARN：指定日誌訊息的最低輸出級別，預設為DEBUG。  
ImmediateFlush=true：表示所有訊息都會被立即輸出，設為false則不輸出，預設值是true。  
Append=false：true表示訊息增加到指定檔中，false則將消息覆蓋指定的檔內容，預設值是true。  
File=D:/logs/logging.log4j：指定訊息輸出到logging.log4j文件中。  
DatePattern='.'yyyy-MM：即每月產生一個新的日誌檔。  
若目前月的日誌檔案名為logging.log4j，前一個月的日誌檔案名會為為logging.log4j.yyyy-MM  
另外，也可以指定按周、天、時、分等來滾動日誌檔，對應的格式如下：  
1)'.'yyyy-MM：每月  
2)'.'yyyy-ww：每週  
3)'.'yyyy-MM-dd：每天  
4)'.'yyyy-MM-dd-a：每天兩次  
5)'.'yyyy-MM-dd-HH：每小時  
6)'.'yyyy-MM-dd-HH-mm：每分鐘  
  
若要有副檔名：  
'\_'yyyy-MM-dd'.log'  
  
  
(4)RollingFileAppender選項：  
Threshold=WARN：指定日誌訊息的最低輸出級別，預設為DEBUG。  
ImmediateFlush=true：表示所有訊息都會被立即輸出，設為false則不輸出，預設值是true。  
Append=false：true表示訊息增加到指定檔中，false則將消息覆蓋指定的檔內容，預設值是true。  
File=D:/logs/logging.log4j：指定訊息輸出到logging.log4j文件中。  
MaxFileSize=100KB：單位可以是KB, MB 或者GB。在日誌文件到達該大小時，將會自動變更，即將原來的內容移到logging.log4j.1文件中。  
MaxBackupIndex=2：指定可以產生的變動檔的最大數，例如，設為2則可以產生logging.log4j.1，logging.log4j.2兩個變動文件和一個logging.log4j文件。  
  
**3、配置日誌訊息的輸出格式（Layout）：**  
log4j.appender.appenderName.layout=className  
className：可設值如下：  
(1)org.apache.log4j.HTMLLayout（以HTML表格形式佈局）  
(2)org.apache.log4j.PatternLayout（可以靈活地指定佈局模式）  
(3)org.apache.log4j.SimpleLayout（包含日誌訊息的級別和訊息字串）  
(4)org.apache.log4j.TTCCLayout（包含日誌產生的時間、執行緒、類別等等訊息）  
  
Layout 亦會反映在 Logger 的階層上。  
  
(1)HTMLLayout選項：  
LocationInfo=true：輸出java檔案名稱和行號，預設值是false。  
Title=My Logging： 預設值是Log4J Log Messages。  
  
(2)PatternLayout選項：  
ConversionPattern=%m%n：設定以怎樣的格式顯示訊息。  
  
例如,為 Logger 加入一個 conversion pattern 為 "%r [%t] %-5p %c - %m%n" 的 PatternLayout,則輸出的 log message 就可能會像下列這樣：  
  
*176 [main] INFO org.foo.Bar - Located nearest gas station.*  
  
  
格式字元說明：  
# %c 輸出日誌訊息所屬的類別的全名  
# %d 輸出日誌時間點的日期或時間，指定格式的方式：%d{yyy-MM-dd HH:mm:ss }。  
# %l 輸出日誌事件的發生位置，即輸出日誌訊息的語句處於它所在的類別的第幾行。相當於%c.%M(%F:%L)的組合包括類別全名、method、檔案名以及行數例如：test.TestLog4j.main(TestLog4j.java:10)  
# %m 輸出的訊息，如log(message)中的message。  
# %n 輸出一個換行符號。  
# %p 輸出優先階層，即DEBUG，INFO，WARN，ERROR，FATAL。如果是調用debug()輸出的，則為DEBUG，依此類推。  
# %r 輸出自應用啟動到輸出該日誌訊息所耗費的毫秒數。  
# %t 輸出產生該日誌事件的執行緒名。  
# %r 輸出自應用啟動到輸出該日誌訊息所耗費的毫秒數。  
# %f 輸出日誌訊息所屬的類別的類別名。  
# %M 輸出產生日誌的方法名。  
# %F 輸出日誌時所在的檔案名稱。  
# %L 輸出行號。  
# %x 輸出和當前執行緒相關聯的NDC(嵌套診斷環境)，尤其用到像java servlets這樣的多客戶多執行緒的應用中。  
# %% 輸出一個"%"字元。  
  
另外，也可以在%與格式字元之間加上修飾符來控制其最小長度、最大長度、對齊方式。如：  
1) c：指定輸出category的名稱，最小的長度是20，如果category的名稱長度小於20的話，預設的情況下右對齊。  
2)%-20c："-"號表示左對齊。  
3)%.30c：指定輸出category的名稱，最大的長度是30，如果category的名稱長度大於30的話，就會將左邊多出的字元截掉，但小於30的話也不會補空格。  
  
  
  
範例  
  
Log4j設定檔實現了輸出到控制台(console)、logFile、rollingFile、發送日誌郵件、輸出到資料庫日誌表、自定義標籤等  
  
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  
  
# 日誌文件(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  
  
# 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  
  
# 定期變更檔案(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  
  
# 應用於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  
  
# 發送日誌到指定郵件  
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  
  
# 應用於資料庫  
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  
  
# 自定義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](http://logging.apache.org/) 是目前開發 java 時很常用到的 log 套件，  
以下整理一些我目前用到的功能，並附上更多更深入的教學。  
  
首先在使用  [Log4j](http://logging.apache.org/) 時我們要先將一些 jar 檔匯入，  
如果有出現意外的 Exception 記得先找看看是不是有 jar 檔沒有匯入，  
類似的問題將 Exception 丟到 google 就會有解答，所以在這邊就不詳細說明。  
只列出官方的使用前須知：<http://logging.apache.org/log4j/1.2/faq.html#1.3>  
  
(2011.10.04更新)  
若您只是要快速產生或下載 Log4j 的設定檔，請參考：  
[Logback, Log4j 設定檔自動產生器](http://werdna1222coldcodes.blogspot.com/2011/10/logback-log4j.html)  
  
使用 [Log4j](http://logging.apache.org/) 時我們可以建立一個 Log4j.properties 檔來描述我們要記錄的項目，  
這個設定檔長得像下麵這樣：  
--------------------------------------------------------------------------------------------------------------  
log4j.rootLogger=Info, A1, A2

# A1 is set to be a ConsoleAppender

log4j.appender.A1=org.apache.log4j.ConsoleAppender  
log4j.appender.A1.layout=org.apache.log4j.PatternLayout  
log4j.appender.A1.layout.ConversionPattern=[%d{yy/MM/dd HH:mm:ss}][%p][%C-%L] %m%n

# A2 is set to be a file

log4j.appender.A2=org.apache.log4j.DailyRollingFileAppender  
log4j.appender.A2.layout=org.apache.log4j.PatternLayout  
log4j.appender.A2.layout.ConversionPattern=[%d{yy/MM/dd HH:mm:ss}][%p][%C-%L] %m%n  
log4j.appender.A2.File=./log/Log4j.log  
--------------------------------------------------------------------------------------------------------------  
關於設定檔的詳細設定說明可以參考[史帝芬心得筆記](http://my.so-net.net.tw/idealist/Java/Log4j.html)，重要部份節錄如下：

1. 階層：

Log4j將輸出訊息分成五個等級，分別為DEBUG、INFO、WARN、ERROR、FATAL，在上面設定檔將輸出等級 設為INFO，就表示INFO~FATAL等級的訊息將會輸出，因此DEBUG等級的訊息就會被忽略，如果將設定檔的 第一行改為log4j.rootLogger=DEBUG, A1, A2，就會兩個訊息都輸出了。除了這五個等級外，也可以設為ALL或OFF， ALL很示全部訊息都輸出，OFF則表示不輸出訊息。

1. 輸出格式：

# %c 輸出日誌訊息所屬的類別的全名 # %d 輸出日誌時間點的日期或時間，指定格式的方式：%d{yyy-MM-dd HH:mm:ss }。 # %l 輸出日誌事件的發生位置，即輸出日誌訊息的語句處於它所在的類別的第幾行。 # %m 輸出訊息，如log(message)中的message。 # %n 輸出一個列尾符號。 # %p 輸出優先階層，即DEBUG，INFO，WARN，ERROR，FATAL。如果是調用debug()輸出的，則為DEBUG，依此類推。 # %r 輸出自應用啟動到輸出該日誌訊息所耗費的毫秒數。 # %t 輸出產生該日誌事件的線程名。 # %r 輸出自應用啟動到輸出該日誌訊息所耗費的毫秒數。 # %f 輸出日誌訊息所屬的類別的類別名。 值得注意的是A2的appender被設為org.apache.log4j.DailyRollingFileAppender表示Log4j會將日誌檔依日期分開。

完成設定檔後可以開始寫程式，程式的內容非常簡單：

1 **package** demo.log4j;

2

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

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

5

6 **public class** Log4jDemo {

7

8 **static public** **void** main(**String** args[]) {

9

10 // 純 java project 中，若將 Log4j.properties 和 java 檔放在一起，

11 // 則 Eclipse 會把他複製到 .class 檔的位置，(project路徑/build/classes/package路徑/檔名)

12 // 若是 web-application 的話，會在 (project路徑/WEB-INF/classes/package路徑/檔名)

13 PropertyConfigurator.configure("build/classes/demo/log4j/Log4j.properties");

14 **Logger** logger = **Logger**.getLogger(Log4jDemo.**class**);

15

16 // 對應的 Log4j.properties 設定要在等級 Info 之上才會顯示，所以logger.debug 不會出現

17 logger.debug("Hello Log4j, this is debug message");

18

19 // 以下的訊息會出現在 console 和 log file 中

20 logger.info("Hi Log4j, this will appear in console and log file");

21 logger.error("This is error message!!!");

22 }

23 }

其結果會在 console 和 log file 中出現這些log ：

[09/11/19 18:44:39][INFO][demo.log4j.Log4jDemo-20] Hi Log4j, this will appear in console and log file

[09/11/19 18:44:39][ERROR][demo.log4j.Log4jDemo-21] This is error message!!!

除了將 log 寫在 console 和 log file 外，[Log4j](http://logging.apache.org/) 還可以寫入其他更多地方，如 DB、Email等等，  
這些應用可以參考：[利用log4j實作console log + file log + mail log + db log](http://www.javaworld.com.tw/jute/post/view?bid=11&id=186896&sty=1&tpg=1&age=0)

[**Log4j詳細介紹(一)----Log4j快速入門**](http://www.cnblogs.com/ArtsCrafts/archive/2013/06/06/Log4j%e5%bf%ab%e9%80%9f%e5%85%a5%e9%97%a8.html)

            Log4j是目前運用最廣泛的日誌控制項,它把日誌分為ALL,TRACE(跟蹤),DEBUG( 調試),INFO(資訊),WARNING(警告),ERROR(錯誤),FITAL(致命),OFF等幾個級別,級別依次升高.級別高的Level會遮罩級別低的信息.如果設置為WARNING,則TRACE,DEBUG,INFO都不會輸出。Log4j的使用非常簡單

[复制代码](javascript:void(0);)

1 import org.apache.log4j.Logger;

2

3 public class Log4jTest {

4 public static Logger log = Logger.getLogger(Log4jTest.class);

5

6 public static void main(String[] args) {

7 log.trace("trace信息");

8 log.debug("debug信息");

9 log.info("info信息");

10 log.warn("warn信息");

11 log.error("errof信息");

12 log.fatal("fatal信息");

13 try {

14 String s = null;

15 s.length();

16 } catch (Exception e) {

17 log.trace("trace一個異常", e);

18 log.debug("debug一個異常", e);

19 log.info("info一個異常", e);

20 log.warn("warn一個異常", e);

21 log.error("error一個異常", e);

22 log.fatal("fatal一個異常", e);

23 }

24 }

25

26 }

[复制代码](javascript:void(0);)

日誌配置如輸出級別,輸出到哪兒,輸出什麼附加資訊,輸出格式等,一般都寫在設定檔log4j.properties中。Log4j默認的設定檔為log4j.properties.啟動時,會載入classpath下的log4j.properties初始化log4j。如果檔不存在,Log4j會在控制台列印如下資訊,提示沒有找到Log4j配置

1 log4j:WARN No appenders could be found for logger (com.log4j.test2.Log4jTest).

2 log4j:WARN Please initialize the log4j system properly.

下麵是log4j.properties的配置

[复制代码](javascript:void(0);)

1 # ERROR級別,輸出到A1

2 log4j.rootLogger=DEBUG, A1

3

4 # 定義A1為控制台輸出

5 log4j.appender.A1=org.apache.log4j.ConsoleAppender

6 log4j.appender.A1.layout=org.apache.log4j.PatternLayout

7 #定義輸出格式

8 log4j.appender.A1.layout.ConversionPattern=%-d{yyyy-MM-dd HH:mm:ss,SSS} **[**%C**]**-**[**%p**]** %m%n

[复制代码](javascript:void(0);)

 如果不適用默認的檔案名log4j.properties，可用使用PropertyConfigurator指定設定檔路徑,例如:

[复制代码](javascript:void(0);)

1 import org.apache.log4j.Logger;

2 import org.apache.log4j.PropertyConfigurator;

3

4 public class Log4jConfigTest {

5

6 public static Logger log = Logger.getLogger(Log4jConfigTest.class);

7

8 public static void main(String[] args) {

9 PropertyConfigurator.configure("src/log4jConfigTest.properties");

10 for (int i = 0; i < 2; i++) {

11 for (int j = 0; j < 2; j++) {

12

13 if (log.isDebugEnabled())

14 log.debug("開始計算 " + i + " \* " + j + " 次運算");

15 // 或者

16 // if (log.isEnabledFor(Priority.WARN))

17 // log.warn("開始計算 " + i + " \* " + j + " 次運算");

18 }

19 }

20

21 new Thread(new Runnable() {

22 public void run() {

23 // log.error("debug 級別的輸出");

24 log.error("ERROR 級別的輸出");

25 }

26 }).start();

27

28 // log.error("debug 級別的輸出");

29 log.info("INFO 級別的輸出");

30 log.warn("WARN 級別的輸出");

31 log.error("ERROR 級別的輸出");

32

33 try {

34 String s = null;

35 s.length();

36 } catch (Exception e) {

37 log.error(e, e);

38 }

39

40 }

41

42 }

[复制代码](javascript:void(0);)

 Log4j也可以使用XML檔配置,例如:

[复制代码](javascript:void(0);)

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

2 <!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

3 <log4j:configuration xmlns:log4j='http://jakarta.apache.org/log4j/'>

4 <!-- 根logger的設置 -->

5 <root>

6 <priority value="all" />

7 <appender-ref ref="console" />

8 </root>

9 <appender name="console" class="org.apache.log4j.ConsoleAppender">

10 <layout class="org.apache.log4j.PatternLayout">

11 <param name="ConversionPattern" value="%-d{yyyy-MM-dd HH\:mm\:ss,SSS} [%C]-[%p] %m%n" />

12 </layout>

13 </appender>

14 </log4j:configuration>

[复制代码](javascript:void(0);)

輸出結果

[复制代码](javascript:void(0);)

1 2013-06-06 17:40:36,042 **[**com.log4j.test2.Log4jTest**]**-**[**DEBUG**]** debug信息

2 2013-06-06 17:40:36,045 **[**com.log4j.test2.Log4jTest**]**-**[**INFO**]** info信息

3 2013-06-06 17:40:36,045 **[**com.log4j.test2.Log4jTest**]**-**[**WARN**]** warn信息

4 2013-06-06 17:40:36,046 **[**com.log4j.test2.Log4jTest**]**-**[**ERROR**]** errof信息

5 2013-06-06 17:40:36,046 **[**com.log4j.test2.Log4jTest**]**-**[**FATAL**]** fatal信息

6 2013-06-06 17:40:36,048 **[**com.log4j.test2.Log4jTest**]**-**[**DEBUG**]** debug一個異常

7 java.lang.NullPointerException

8 at com.log4j.test2.Log4jTest.main(Log4jTest.java:17)

9 2013-06-06 17:40:36,051 **[**com.log4j.test2.Log4jTest**]**-**[**INFO**]** info一個異常

10 java.lang.NullPointerException

11 at com.log4j.test2.Log4jTest.main(Log4jTest.java:17)

12 2013-06-06 17:40:36,051 **[**com.log4j.test2.Log4jTest**]**-**[**WARN**]** warn一個異常

13 java.lang.NullPointerException

14 at com.log4j.test2.Log4jTest.main(Log4jTest.java:17)

15 2013-06-06 17:40:36,051 **[**com.log4j.test2.Log4jTest**]**-**[**ERROR**]** error一個異常

16 java.lang.NullPointerException

17 at com.log4j.test2.Log4jTest.main(Log4jTest.java:17)

18 2013-06-06 17:40:36,052 **[**com.log4j.test2.Log4jTest**]**-**[**FATAL**]** fatal一個異常

19 java.lang.NullPointerException

20 at com.log4j.test2.Log4jTest.main(Log4jTest.java:17)

[复制代码](javascript:void(0);)

Log4j的使用非常靈活,功能也很強大。可以設定檔中配置輸出樣式,可以吧日誌輸出到螢幕,控制台,各種樣式的檔,資料庫,FTP 伺服器,HTTP 伺服器,遠端即時監控程序,系統日誌,發送郵件到指定的郵箱,甚至發送資訊。這些會在後面撿到,好了以上就是Log4j的快速入門,很簡單吧.

[**Log4j詳細介紹(二)----Commons-logging控制項**](http://www.cnblogs.com/ArtsCrafts/archive/2013/06/06/Log4j2.html)

　　除了Log4j還有一種選擇是使用commons-logging。commons-logging是Apache commons類庫中的一員。Apache commons類庫是一個通用的類庫,提供了基礎的功能,例如commons-fileupload,commons-httpclient,commons-io,commons-codec等

　　commons-logging能夠選擇使用Log4j還是JDK Logging，但不依賴與Log4j,JDK logging的API。如果專案的classpath中包含log4j的類庫，則使用Log4j,否則使用JDK Logging，使用commons-logging能夠靈活地選擇使用哪種日誌方式,而不需要修改原始程式碼。

　　commons-logging的使用類似於Log4j,它們的級別以及使用規則完全一樣的,例如:

[复制代码](javascript:void(0);)

1 import org.apache.commons.logging.Log;

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

3

4 public class CommonsLoggingTest {

5

6 public static Log log = LogFactory.getLog(CommonsLoggingTest.class);

7

8 public static void main(String[] args) {

9 log.trace("trace信息");

10 log.debug("debug信息");

11 log.info("info信息");

12 log.warn("warn信息");

13 log.error("errof信息");

14 log.fatal("fatal信息");

15 try {

16 String s = null;

17 s.length();

18 } catch (Exception e) {

19 log.trace("trace一個異常", e);

20 log.debug("debug一個異常", e);

21 log.info("info一個異常", e);

22 log.warn("warn一個異常", e);

23 log.error("error一個異常", e);

24 log.fatal("fatal一個異常", e);

25 }

26 }

27

28 }

[复制代码](javascript:void(0);)

如果有log4j,commons-logging會把輸出原封不動交給log4j。如果沒有log4j ,commons-logging會將相應的輸出轉換為JDK Logging的輸出。

　　默認地,commons-logging會自動檢查是否使用log4j。也可以使用設定檔顯示的啟動log4j.設定檔為commons-logging.properties,放到程式的classpath下即可,例如:

[复制代码](javascript:void(0);)

1 org.apache.commons.logging.Log=org.apache.commons.logging.impl.Log4JLogger

2 org.apache.commons.logging.LogFactory=org.apache.commons.logging.impl.LogFactoryImpl

[复制代码](javascript:void(0);)

   嚴格來說,commons-logging不是弱智控制項,沒有日誌功能。它只是統一了JDK Logging和Log4j的API，並把日誌功能JDK Logging或者Log4j。對於不能確定的日誌系統commons-logging是個不錯的選擇,Spring，Hibernate,Struts等都使用的是commons-logging

標籤: [Java](http://www.cnblogs.com/ArtsCrafts/tag/Java/)

[**Log4j詳細介紹(三)----Log4j的優化**](http://www.cnblogs.com/ArtsCrafts/archive/2013/06/06/log4j3.html)

　　Log4j內部使用了大量的優化,緩存工作,使輸出時對伺服器的壓力,消耗的時間,資源等都達到最小。例如:

1 log.debug("debug信息");

2 log.warn("warn信息");

     這兩句日誌可能被記錄到日誌檔中,但是寫的過程不同於常規的打開檔,寫檔,關閉檔等流程,頻繁地打開檔,關閉檔需要消耗大量資源。Log4j只有初始化時候打開檔,並保持對檔的寫控制，知道系統結束的時候才關閉檔。這樣就使I/O次數達到最小，提高了效率。  
　　而當輸出級別為ERROR時,log.debug(),log.waring()方法會因為輸出級別小雨ERROR而直接返回,因此不會消耗太多的資源。debug()內部實現如下:

[复制代码](javascript:void(0);)

1 public void debug(Object message){

2 if (this.repository.isDisabled(10000))

3 return;

4 if (Level.DEBUG.isGreaterOrEqual(getEffectiveLevel()))

5 forcedLog(FQCN, Level.DEBUG, message, null);

6 }

7 }

[复制代码](javascript:void(0);)

     雖然設置為ERROR時debug()會直接返回,但是下面的代碼仍然會消耗額外的時間,原因是在執行debug()前,需要先將字串進行連接。

1 log.debug("開始計算 "+i+" \* "+j+" 次 運算");

如果想要避免這種消耗,最好使用isXxxEnable()或者isEnableFor()判斷一下，例如:

[复制代码](javascript:void(0);)

1 if(log.isDebugEnabled())//如果debug級別可用

2 log.debug("開始計算 "+i+" \* "+j+" 次 運算");//連接字串,輸出

3 if(log.isEnabledFor(Priority.WARN))//或者,如果warn可用

4 log.debug("開始計算 "+i+" \* "+j+" 次 運算");// 連接字串輸出

[复制代码](javascript:void(0);)

[**Log4j詳細介紹(四)----日誌記錄器Logger**](http://www.cnblogs.com/ArtsCrafts/archive/2013/06/06/log4j4.html)

　　Log4j配置中有3個重要的概念:日誌記錄器(Logger),輸出地(Appender)以及日誌格式化器(Layout)。其中Logger負責記錄日誌,Appender負責輸出到什麼地方,Layout負責以什麼格式輸出,輸出那些附加資訊(例如時間,類名,方法名,所在行數等)

　　Logger就是Java代碼中的Logger,例如:

1 public static Logger log = Logger.getLogger(Log4jTest.class);

     Logger是有名字的,它的名字便是Logger.getLogger()方法參數。如果參數為所在的類,Log4j會去類名為Logger的名稱,例如com.log4j.test.Log4jTes

　　Logger為單例模式:相同名字的Logger只會有一個實例。如果在構建一個同名的Logger，Log4j會返回先前的Logger實例。

　　命名規則:一般都以類名作為Logger的名稱。Logger的名字類似於Java中的Package名字,大小寫敏感,用點分開具有繼承關係例如com.log4j是com.log4j.test的父親。log4j.properties中通過名稱來配置Logger的屬性。

　　Log4j中有一個跟記錄器rootLogger,它是所有Logger的父親

　　Logger的配置

　　在log.properties配置中,log4j.logger後面配置的是Logger，log4j.appender後面配置的是 Appender,例如:

[复制代码](javascript:void(0);)

1 # 配置Logger為ERROR級別,輸出到A1

2 log4j.logger.com.log4j.test.Log4jTest=DEBUG, A1

3

4 # 配置Appender A1,輸出到控制台,使用運算式佈局

5 log4j.appender.A1=org.apache.log4j.ConsoleAppender

6 log4j.appender.A1.layout=org.apache.log4j.PatternLayout

7 log4j.appender.A1.layout.ConversionPattern=%-d{yyyy-MM-dd HH\:mm\:ss,SSS} **[**%C**]**-**[**%p**]** %m%n

[复制代码](javascript:void(0);)

　　如某個Logger沒有配置,則使用它的父親配置,知道找到為止。一般情況下,只需配置根記錄器rootLogger即可,所有Logger都會沿用rootLogger的配置

　　rootLogger配置

　　根記錄器rootLogger直接使用log4j.rootLogger配置。rootLogger是所有記錄器的父親。任何記錄器都可以繼承rootLogger的配置代碼如下

[复制代码](javascript:void(0);)

1 #配置rootLogger為ERROR級別,輸出地位A1

2 log4j.rootLogger= ERROR, A1

3

4 #配置該logger為DEBUG 級別，輸出地則繼承rootLogger配置

5 log4j.logger.com.log4j.test2.Log4jConfigTest=DEBUG 6

7 # 配置Appender A1,輸出到控制台,使用規則運算式佈局

8 log4j.appender.A1=org.apache.log4j.ConsoleAppender

9 log4j.appender.A1.layout=org.apache.log4j.PatternLayout

10 #定義輸出格式

11 log4j.appender.A1.layout.ConversionPattern=%-d{yyyy-MM-dd HH\:mm\:ss,SSS} **[**%C**]**-**[**%p**]** %m%n

[复制代码](javascript:void(0);)

　　如果要對某個Logger進行特殊輸出,只需要在配置一下該Logger，覆蓋父親的配置即可。覆蓋時，可以配置級別，輸出地，也可兩者都配置

      類別category配置

　　Logger還有個類別(Category)的概念,通過設置類別來設置該類別下所有的Logger:

[复制代码](javascript:void(0);)

1 #配置rootLogger為ERROR級別,輸出地位A1

2 log4j.rootLogger= ERROR, A1

3

4 #作用於類別com.log4j.test下所有的Logger, 輸出則繼承rootLogger配置

5 log4j.category.com.log4j.test=DEBUG

6

7 # 配置Appender A1,輸出到控制台,使用規則運算式佈局

8 log4j.appender.A1=org.apache.log4j.ConsoleAppender

9 log4j.appender.A1.layout=org.apache.log4j.PatternLayout

10 #定義輸出格式

11 log4j.appender.A1.layout.ConversionPattern=%-d{yyyy-MM-dd HH\:mm\:ss,SSS} **[**%C**]**-**[**%p**]** %m%n

[复制代码](javascript:void(0);)

　　catergory類似於Java中的Package,效果跟Logger的名字等價

標籤: [Java](http://www.cnblogs.com/ArtsCrafts/tag/Java/)

[**Log4j詳細介紹(五)----輸出地Appender**](http://www.cnblogs.com/ArtsCrafts/archive/2013/06/07/log4j5.html)

　　Appender表示日誌輸出到什麼地方,常用的輸出地有控制台，檔，資料庫，遠端伺服器等。Log4j中內置了常用的輸出地,一般情況下配置一下即可使用。所有的Appender都實現自org.apache.log4j.Appender藉口。在log4j.properties中，Apender都使用log4j.appender.\*配置

　　輸出到控制台

　　控制台是最常用的輸出地，前面的例子用的就是控制台輸出(ConsoleAppender)。控制台輸出實現類為org.apache.log4j.ConsoleAppender，例如:

[复制代码](javascript:void(0);)

#跟記錄器,ERROR,輸出到A1

log4j.rootLogger=ERROR, A1

#本記錄器為DEBUG級別

log4j.category.com.log4j.test=DEBUG

#控制台輸出

log4j.appender.A1=org.apache.log4j.ConsoleAppender

#DEBUG以上級別是輸出

log4j.appender.A1.Threshold=DEBUG

#編碼方式

log4j.appender.A1.Encoding=UTF-8

#是否立即輸出

log4j.appender.A1.ImmediateFlush=true

#使用System.err輸出

log4j.appender.A1.Target=System.err

#輸出格式,運算式配置

log4j.appender.A1.layout=org.apache.log4j.PatternLayout

log4j.appender.A1.ConversionPattern=%-d{yyyy-MM-d HH:mm:ss, SSS}**[**%C**]**-**[**%p**]** %m%n

[复制代码](javascript:void(0);)

 　　控制台輸出需要配置layout屬性,最常用的是規則運算式。控制台輸出的資訊一般是TRACE,DEBUG或者INFO級的，只有在開發，調試時才啟用。

　　Threshold用來設置該Appender的級別，只對該Appender生效。所有的Appender都可通過設置Threshold來設置本Appender的啟用級別

　輸出到檔:

　　檔輸出(FileAppender)把日誌輸出到指定的檔。檔輸出的實現類為org.apache.log4j.FileAppender，配置時需要File指定檔案名稱。可以使用相對路徑，也可以使用絕對路徑例如：

[复制代码](javascript:void(0);)

1 log4j.logger.com.log4j.test.Log4jTest=DEBUG, F

2

3 #輸出到檔

4 log4j.appender.F=org.apache.log4j.FileAppender

5 #檔位置

6 log4j.appender.F.File=C:\\log4j.log

7 #追加檔內容

8 log4j.appender.F.Append=true

9 #輸出格式運算式

10 log4j.appender.F.layout=org.apache.log4j.PatternLayout

11 log4j.appender.F.layout.ConversionPattern=%-d{yyyy-MM-dd HH\:mm\:ss,SSS} **[**%C**]**-**[**%p**]** %m%n

[复制代码](javascript:void(0);)

　　可選參數Append配置是否在原文件內容的基礎上追加日誌。如果為false，Logger 初始化時會先清掉檔內容，也就是說每次重啟程式，原來的日誌會丟失。如果為true，日誌檔會越來越大。默認為true

　　輸出到按大小滾動檔

　　按大小滾動檔輸出(RollingFileAppender)吧日誌輸出到指定的檔，檔達到指定大小時，會自動更名。按尺寸滾動檔輸出類為org.apache.log4j.RollingFileAppender,需要設定檔名稱，檔的最大尺寸，例如：

[复制代码](javascript:void(0);)

1 #DEBUG 級別，兩個輸出：文件與滾動文件

2 log4j.rootLooger=DEBUG, F, Rolling\_File

3

4 #輸出到檔

5 log4j.appender.F=org.apache.log4j.FileAppender

6 log4j.appender.F.File=C:\\F.log

7 log4j.appender.F.Appender=true

8 log4j.appender.F.layout=org.apache.log4j.PatternLayout

9 log4j.appender.F.layout.ConversionPattern=%-d{yyyy-MM-dd HH\:mm\:ss,SSS} **[**%C**]**-**[**%p**]** %m%n

10

11 #輸出到滾動檔

12 log4j.appender.Rolling\_File=org.apache.log4j.RollingFileAppender

13 #DEBUG以上才使用

14 log4j.appender.Rolling\_File.Threshold=DEBUG

15 #滾動檔案名

16 log4j.appender.Rolling\_File.File=C:\\rolling.log

17 #追加方式

18 log4j.appender.Rolling\_File.Append=true

19 #檔達到10KB就自動命名

20 log4j.appender.Rolling\_File.MaxFileSize=10KB

21 #最多備份100個檔

22 log4j.appender.Rolling\_File.MaxBackupIndex=100

23 log4j.appender.Rolling\_File.layout=org.apache.log4j.PatternLayout

24 log4j.appender.Rolling\_File.layout.ConversionPattern=%-d{yyyy-MM-dd HH\:mm\:ss,SSS} **[**%C**]**-**[**%p**]** %m%n

[复制代码](javascript:void(0);)

　　配置的滾動檔案名為rolling.log，文件最大為10KB。當rolling.log達到10KB時，會自動更名為rolling.log1,rolling.log2,rolling.log3………………知道rolling.log100。

　　Logger支持多個Appender，用逗號講多個Appender名字隔開即可。

　　輸出到按日期滾動檔

　　按日期滾動檔輸出(DailyRollingFileAppender)將日誌輸出到指定的檔，當日期發生變化時，會自動按指定的日期格式自動改名。日期滾動檔輸出類為org.apache.log4j.DailyRollingFileAppender，例如:

[复制代码](javascript:void(0);)

1 #DEBUG 級別，兩個輸出：文件與滾動文件

2 log4j.rootLooger=DEBUG, Daily\_Rolling

3

4 #輸出到滾動檔

5 log4j.appender.Daily\_Rolling=org.apache.log4j.Daily\_RollingFileAppender

6 #DEBUG以上才使用

7 log4j.appender.Daily\_Rolling.Threshold=DEBUG

8 #滾動檔案名

9 log4j.appender.Daily\_Rolling.File=C:\\daily\_rolling.log

10 #追加方式

11 log4j.appender.Daily\_Rolling.Append=true

12 #滾動日期格式

13 log4j.appender.Daily\_Rolling.DatePattern=.yyyy-MM-dd

14 # 輸出日期格式

15 log4j.appender.Daily\_Rolling.layout=org.apache.log4j.PatternLayout

16 log4j.appender.Daily\_Rolling.layout.ConversionPattern=%-d{yyyy-MM-dd HH\:mm\:ss,SSS} **[**%C**]**-**[**%p**]** %m%n

[复制代码](javascript:void(0);)

　　日誌檔名稱為daily\_rolling.log，日期格式為 yyyy-MM-dd。進入新的一天后，檔會被自動更名，格式為daily\_rolling.log.2013-06-06。

　　輸出到JDBC資料庫

　　資料庫輸出(JDBCAppender)，通過JDBC連結把日誌輸出到資料庫中，配置時需要配置JDBC驅動，連接字串，用戶名，密碼以及SQL語句。注意JDBCAppender需要相應的資料庫驅動。先創建資料庫，並建立資料庫日誌表。本例使用MySQL資料庫，腳本如下：

[复制代码](javascript:void(0);)

1 create database log4j;

2 use log4j;

3 create table tb\_log(id int primary key auto\_increment,

4 date varchar(**255**),

5 priority varchar(**255**),

6 message text,

7 classname varchar(**255**));

[复制代码](javascript:void(0);)

    配置JDBCAppender的代碼如下:log4j.properties

[复制代码](javascript:void(0);)

1 #配置rootLogger為ALL級別,輸出地位DATABASE

2 log4j.rootLogger= ALL, DATABASE

3 #資料庫輸出

4 log4j.appender.DATABASE=org.apache.log4j.jdbc.JDBCAppender

5 log4j.appender.DATABASE.Threshold=DEBUG

6 #資料庫連接URL

7 log4j.appender.DATABASE.URL=jdbc:mysql://localhost:3306/log4j

8 #資料庫驅動

9 log4j.appender.DATABASE.driver=com.mysql.jdbc.Driver

10 #用戶名

11 log4j.appender.DATABASE.user=root

12 #使用者密碼

13 log4j.appender.DATABASE.password=password

14 #執行SQL語句,列內容為運算式

15 log4j.appender.DATABASE.sql=insert into tb\_log(date, priority, message, classname) values('%d', '%p', '%m', '%c')

16 #輸出格式

17 log4j.appender.DATABASE.layout=org.apache.log4j.PatternLayout

18 log4j.appender.DATABASE.layout.ConversionPattern=%-d{yyyy-MM-dd HH\:mm\:ss,SSS} **[**%C**]**-**[**%p**]** %m%n

[复制代码](javascript:void(0);)

  配置的關鍵是SQL語句。Log4j能夠取到各種環境資料，包括時間，級別，類名，檔案名，方法名，所在行數等。具體詳見後面的格式化器Layout。SQL八個參數對應到日誌表的相關列中。運行代碼如下：

[复制代码](javascript:void(0);)

1 import org.apache.log4j.Logger;

2

3 public class Log4jJDBCTest {

4 public static Logger log = Logger.getLogger(Log4jJDBCTest.class);

5

6 public static void main(String[] args) {

7 log.trace("trace信息");

8 log.debug("debug信息");

9 log.info("info信息");

10 log.warn("warn信息");

11 log.error("errof信息");

12 log.fatal("fatal信息");

13 try {

14 String s = null;

15 s.length();

16 } catch (Exception e) {

17 log.trace("trace一個異常", e);

18 log.debug("debug一個異常", e);

19 log.info("info一個異常", e);

20 log.warn("warn一個異常", e);

21 log.error("error一個異常", e);

22 log.fatal("fatal一個異常", e);

23 }

24 }

25

26 }

[复制代码](javascript:void(0);)

運行效果如下:

 注意：JDBC輸出可能會因為短時間內大量的連接資料庫而導致連接失敗，因此要謹慎使用JDBC輸出

[**Log4j詳細介紹(六)----自訂輸出**](http://www.cnblogs.com/ArtsCrafts/archive/2013/06/07/log4j6.html)

　　Log4j支援自訂輸出。所有輸出都是實現自Appender介面。一般來說，自訂輸出只需要繼承AppenderSkeleton類，並實現幾個方法就可以了。

　　看一個自訂輸出的例子。CountingConsoleAppender跟控制台輸出類似，不同的是會統計日誌輸出的次數。當輸出次數超過預定的值時，會做相應的業務處理(這裡為列印一行提示資訊)，並停止輸出，代碼如下。

[复制代码](javascript:void(0);)

1 import org.apache.log4j.AppenderSkeleton;

2 import org.apache.log4j.spi.ErrorCode;

3 import org.apache.log4j.spi.LoggingEvent;

4

5 public class CountingConsoleAppender extends AppenderSkeleton {

6 int count = 0;// 記錄日誌的次數

7 int limit = 16;// 許可權次數

8

9 public void setLimit(int limit) {// setter方法

10 this.limit = limit;

11 }

12

13 public int getLimit() {// getter方法

14 return limit;

15 }

16

17 @Override

18 public void close() {

19 if (this.closed)// 如果被關掉了

20 return;

21 this.closed = true;

22 }

23

24 @Override

25 public boolean requiresLayout() {// 需要使用佈局

26 return true;

27 }

28

29 @Override

30 protected void append(LoggingEvent event) {

31 if (this.layout == null) {

32 // 沒有設置出佈局

33 errorHandler.error("沒有設置[" + name + "]輸出佈局.", null,

34 ErrorCode.MISSING\_LAYOUT);

35 return;

36 }

37 if (count >= limit) {

38 errorHandler.error("輸出次數[" + limit + "]達到了[" + getName() + "]的極限",

39 null, ErrorCode.WRITE\_FAILURE);

40 return;

41 }

42 System.out.println(this.layout.format(event));

43 if (layout.ignoresThrowable()) {

44 String[] t = event.getThrowableStrRep();

45 if (t != null) {

46 int len = t.length;

47 for (int i = 0; i < len; i++) {

48 System.out.println(t[i]);

49 }

50 }

51 }

52 count++;// 繼續下次日誌

53 }

54

55 }

[复制代码](javascript:void(0);)

 其中最大輸出次數可以在log4j.properties中配置。配置代碼如下:

[复制代码](javascript:void(0);)

1 log4j.rootLogger= DEBUG, COUNTING

2

3 log4j.appender.COUNTING=com.log4j.test.CountingConsoleAppender

4

5 #輸出10次停止

6 log4j.appender.COUNTING.limit=10

7 log4j.appender.COUNTING.layout=org.apache.log4j.PatternLayout

8 log4j.appender.COUTIING.layout.ConversionPattern=%-d{yyyy-MM-dd HH\:mm\:ss,SSS} **[**%C**]**-**[**%p**]** %m%n

[复制代码](javascript:void(0);)

設定輸出次數為10.當輸出超過10次後，該自訂Appender會輸出下面的資訊,並停止輸出:

1 log4j:ERROR 輸出次數**[**10**]**達到了**[**COUNTING**]**的極限

java日誌框架log4j詳細配置及與slf4j聯合使用教程

## 一、log4j基本用法

　　首先，配置log4j的jar，maven工程配置以下依賴，非maven工程從[maven倉庫](http://maven.aliyun.com/nexus/content/groups/public/log4j/)下載jar添加到“build path”

|  |  |
| --- | --- |
| 1  2  3  4  5 | <dependency>      <groupId>log4j</groupId>      <artifactId>log4j</artifactId>      <version>1.2.17</version>  </dependency> |

　　然後，在src/main/java目錄（包的根目錄即classpath）新建log4j.properties文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | log4j.rootLogger=INFO,console  log4j.additivity.org.apache=true  #console  log4j.appender.console=org.apache.log4j.ConsoleAppender  log4j.appender.console.Threshold=INFO  log4j.appender.console.ImmediateFlush=true  log4j.appender.console.Target=System.out  log4j.appender.console.layout=org.apache.log4j.PatternLayout  log4j.appender.console.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} [%p] %m%n |

　　最後，新建Main.java文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | package com.xmyself.log4j;  import org.apache.log4j.Logger;  public class Main {      public static void main(String[] args) {          new Test().test();      }  }  class Test {      final Logger log = Logger.getLogger(Test.class);      public void test() {          log.info("hello this is log4j info log");      }  } |

　　運行main方法，日誌資訊就出來了

|  |  |
| --- | --- |
| 1 | 2016-12-01 21:23:29 [INFO] hello this is log4j info log |

## 二、log4j.properties路徑

　　log4j.properties要放在哪以及怎樣配置才能被解析呢？不同工程類型配置方式不同

### 1、普通java或spring工程

　　這是最常見的java工程類型，寫demo用的多，把log4j.properties放在src/main/java目錄（包的根目錄）就行了

### 2、spring mvc工程

　　web工程裡用spring mvc構建的比較多了，把log4j.properties放在src/main/resources的conf目錄（web工程設定檔通常在resources或WEB-INF目錄），編輯web.xml，添加

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <context-param>      <param-name>log4jConfigLocation</param-name>      <param-value>classpath:/conf/log4j.properties</param-value>  </context-param>  <listener>      <listener-class>org.springframework.web.util.Log4jConfigListener</listener-class>  </listener> |

### 3、普通web工程

　　沒有了spring提供的listener載入log4j.properties，我們要怎麼載入這個檔呢？同樣，把log4j.properties放在src/main/resources的conf目錄，用servlet載入

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | public class Log4jServlet extends HttpServlet {      private static final long serialVersionUID = 1L;        public void init(ServletConfig config) throws ServletException {          String prefix = this.getClass().getClassLoader().getResource("/").getPath();          String path = config.getInitParameter("log4j-path");          PropertyConfigurator.configure(prefix + path);      }      public void doGet(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException {}      public void doPost(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException {}      public void destroy() {}  } |

　　編輯web.xml，添加

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <servlet>      <servlet-name>log4j</servlet-name>      <servlet-class>com.xmyself.log4j.Log4jServlet</servlet-class>      <init-param>          <param-name>log4j-path</param-name>          <param-value>conf/log4j.properties</param-value>      </init-param>      <load-on-startup>1</load-on-startup>  </servlet> |

　　看著是不是和spring mvc的很像，甚至你也想到了，普通java工程沒有指定log4j.properties的路徑，那說明log4j的jar包一定有一個預設的路徑。另外，建議，log4j的配置放在第一個，因為後續載入其他元件就要開始使用日誌記錄了

　　現在，你可以在多種類型的java工程中打出日誌了，但都是控制台的日誌，輸出內容也很有限，下面我們就來詳細介紹log4j.properties內容怎麼配置

## 三、log4j.properties內容

　　接下來介紹的內容看起來獨立，其實相互關聯，並且很有規律，我們要輸出日誌，首先得有日誌物件（logger），那這些日誌物件把日誌輸出到哪裡呢，控制台還是檔，這就要設置輸出位置（appender），輸出的格式與內容又是什麼樣的呢，這就要設置輸出樣式（layout），這些設置完，log4j的配置也就完了

　　在此之前，先介紹下log4j日誌等級的概念，日誌等級就是日誌的重要程度，log4j日誌分為7個等級：ALL、DEBUG、INFO、WARN、ERROR、FATAL、OFF，從左到右等級由低到高，分等級是為了設置日誌輸出的門檻，只有等級等於或高於這個門檻的日誌才有機會輸出

### 1、logger

　　日誌實例，就是代碼裡產生實體的Logger物件

|  |  |
| --- | --- |
| 1  2 | log4j.rootLogger=LEVEL,appenderName1,appenderName2,...  log4j.additivity.org.apache=false：表示不會在父logger的appender裡輸出，默認true |

　　這是全域logger的配置，LEVEL用來設定日誌等級，appenderName定義日誌輸出器，示例中的“console”就是一個日誌輸出器

　　下面給出一個更清晰的例子，配置“com.demo.test”包下所有類中產生實體的Logger物件

|  |  |
| --- | --- |
| 1  2 | log4j.logger.com.demo.test=DEBUG,test  log4j.additivity.com.demo.test=false |

### 2、appender

　　日誌輸出器，指定logger的輸出位置

|  |  |
| --- | --- |
| 1 | log4j.appender.appenderName=className |

　　appender有5種選擇

|  |  |
| --- | --- |
| 1  2  3  4  5 | org.apache.log4j.ConsoleAppender（控制台）  org.apache.log4j.FileAppender（文件）  org.apache.log4j.DailyRollingFileAppender（每天產生一個日誌檔）  org.apache.log4j.RollingFileAppender（檔大小到達指定尺寸的時候產生一個新的檔）  org.apache.log4j.WriterAppender（將日誌資訊以流格式發送到任意指定的地方） |

　　每種appender都有若干配置項，下面逐一介紹

　　ConsoleAppender（常用）

|  |  |
| --- | --- |
| 1  2  3 | Threshold=WARN：指定日誌資訊的最低輸出級別，默認DEBUG  ImmediateFlush=true：表示所有消息都會被立即輸出，設為false則不輸出，預設值是true  Target=System.err：預設值是System.out |

　　FileAppender

|  |  |
| --- | --- |
| 1  2  3  4 | Threshold=WARN：指定日誌資訊的最低輸出級別，默認DEBUG  ImmediateFlush=true：表示所有消息都會被立即輸出，設為false則不輸出，默認true  Append=false：true表示消息增加到指定檔中，false則將消息覆蓋指定的檔內容，預設true  File=D:/logs/logging.log4j：指定消息輸出到logging.log4j文件 |

　　DailyRollingFileAppender（常用）

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | Threshold=WARN：指定日誌資訊的最低輸出級別，默認DEBUG  ImmediateFlush=true：表示所有消息都會被立即輸出，設為false則不輸出，默認true  Append=false：true表示消息增加到指定檔中，false則將消息覆蓋指定的檔內容，預設true  File=D:/logs/logging.log4j：指定當前消息輸出到logging.log4j文件  DatePattern='.'yyyy-MM：每月滾動一次日誌檔，即每月產生一個新的日誌檔。當前月的日誌檔案名為logging.log4j，前一個月的日誌檔案名為logging.log4j.yyyy-MM  另外，也可以指定按周、天、時、分等來滾動日誌檔，對應的格式如下：  1)'.'yyyy-MM：每月  2)'.'yyyy-ww：每週  3)'.'yyyy-MM-dd：每天  4)'.'yyyy-MM-dd-a：每天兩次  5)'.'yyyy-MM-dd-HH：每小時  6)'.'yyyy-MM-dd-HH-mm：每分鐘 |

　　RollingFileAppender

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | Threshold=WARN：指定日誌資訊的最低輸出級別，默認DEBUG  ImmediateFlush=true：表示所有消息都會被立即輸出，設為false則不輸出，默認true  Append=false：true表示消息增加到指定檔中，false則將消息覆蓋指定的檔內容，預設true  File=D:/logs/logging.log4j：指定消息輸出到logging.log4j文件  MaxFileSize=100KB：尾碼可以是KB,MB或者GB。在日誌檔到達該大小時，將會自動滾動，即將原來的內容移到logging.log4j.1文件  MaxBackupIndex=2：指定可以產生的滾動檔的最大數，例如，設為2則可以產生logging.log4j.1，logging.log4j.2兩個滾動檔和一個logging.log4j文件 |

### 3、layout

　　指定logger輸出內容及格式

|  |  |
| --- | --- |
| 1 | log4j.appender.appenderName.layout=className |

　　layout有4種選擇

|  |  |
| --- | --- |
| 1  2  3  4 | org.apache.log4j.HTMLLayout（以HTML表格形式佈局）  org.apache.log4j.PatternLayout（可以靈活地指定佈局模式）  org.apache.log4j.SimpleLayout（包含日誌資訊的級別和資訊字串）  org.apache.log4j.TTCCLayout（包含日誌產生的時間、執行緒、類別等資訊） |

　　layout也有配置項，下面具體介紹

　　HTMLLayout

|  |  |
| --- | --- |
| 1  2 | LocationInfo=true：輸出java檔案名稱和行號，默認false  Title=My Logging： 預設值是Log4J Log Messages |

　　PatternLayout（最常用的配置）

|  |  |
| --- | --- |
| 1 | ConversionPattern=%m%n：設定以怎樣的格式顯示消息 |

　　設置格式的參數說明如下

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | %p：輸出日誌資訊的優先順序，即DEBUG，INFO，WARN，ERROR，FATAL  %d：輸出日誌時間點的日期或時間，默認格式為ISO8601，可以指定格式如：%d{yyyy/MM/dd HH:mm:ss,SSS}  %r：輸出自應用程式啟動到輸出該log資訊耗費的毫秒數  %t：輸出產生該日誌事件的執行緒名  %l：輸出日誌事件的發生位置，相當於%c.%M(%F:%L)的組合，包括類全名、方法、檔案名以及在代碼中的行數  %c：輸出日誌資訊所屬的類目，通常就是類全名  %M：輸出產生日誌資訊的方法名  %F：輸出日誌消息產生時所在的檔案名  %L：輸出代碼中的行號  %m：輸出代碼中指定的具體日誌資訊  %n：輸出一個回車分行符號，Windows平臺為"rn"，Unix平臺為"n"  %x：輸出和當前執行緒相關聯的NDC(嵌套診斷環境)  %%：輸出一個"%"字元 |

## 四、log4j完整配置示例

　　介紹完了log4j.properties內容，我們來配置一些常用的日誌輸出吧

|  |  |
| --- | --- |
| 1  2 | log4j.rootLogger=DEBUG,console,dailyFile,rollingFile,logFile  log4j.additivity.org.apache=true |

　　控制台console日誌輸出器

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # 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=%d{yyyy-MM-dd HH:mm:ss} [%p] %m%n |

　　檔logFile日誌輸出器

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | # 日誌檔(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=%d{yyyy-MM-dd HH:mm:ss} [%p] %m%n |

　　滾動檔rollingFile日誌輸出器

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | # 滾動文件(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=%d{yyyy-MM-dd HH:mm:ss} [%p] %m%n |

　　定期滾動檔dailyFile日誌輸出器

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | # 定期滾動日誌檔(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=%d{yyyy-MM-dd HH:mm:ss} [%p] %m%n |

## 五、log4j局部日誌配置

　　以上介紹的配置都是全域的，整個工程的代碼使用同一套配置，意味著所有的日誌都輸出在了相同的地方，你無法直接了當的去看資料庫訪問日誌、使用者登錄日誌、操作日誌，它們都混在一起，因此，需要為包甚至是類配置單獨的日誌輸出，下面給出一個例子，為“com.demo.test”包指定日誌輸出器“test”，“com.demo.test”包下所有類的日誌都將輸出到/log/test.log文件

|  |  |
| --- | --- |
| 1  2  3  4  5 | log4j.logger.com.demo.test=DEBUG,test  log4j.appender.test=org.apache.log4j.FileAppender  log4j.appender.test.File=/log/test.log  log4j.appender.test.layout=org.apache.log4j.PatternLayout  log4j.appender.test.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} [%p] %m%n |

　　也可以讓同一個類輸出不同的日誌，為達到這個目的，需要在這個類中產生實體兩個logger

|  |  |
| --- | --- |
| 1  2 | private static Log logger1 = LogFactory.getLog("myTest1");  private static Log logger2 = LogFactory.getLog("myTest2"); |

　　然後分別配置

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | log4j.logger.myTest1= DEBUG,test1  log4j.additivity.myTest1=false  log4j.appender.test1=org.apache.log4j.FileAppender  log4j.appender.test1.File=/log/test1.log  log4j.appender.test1.layout=org.apache.log4j.PatternLayout  log4j.appender.test1.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} [%p] %m%n    log4j.logger.myTest2=DEBUG,test2  log4j.appender.test2=org.apache.log4j.FileAppender  log4j.appender.test2.File=/log/test2.log  log4j.appender.test2.layout=org.apache.log4j.PatternLayout  log4j.appender.test2.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} [%p] %m%n |

## 六、slf4j與log4j聯合使用

　　slf4j是什麼？slf4j只是定義了一組日誌介面，但並未提供任何實現，既然這樣，為什麼要用slf4j呢？log4j不是已經滿足要求了嗎？

　　是的，log4j滿足了要求，但是，日誌框架並不只有log4j一個，你喜歡用log4j，有的人可能更喜歡logback，有的人甚至用jdk自帶的日誌框架，這種情況下，如果你要依賴別人的jar，整個系統就用了兩個日誌框架，如果你依賴10個jar，每個jar用的日誌框架都不同，豈不是一個工程用了10個日誌框架，那就亂了！

　　如果你的代碼使用slf4j的介面，具體日誌實現框架你喜歡用log4j，其他人的代碼也用slf4j的介面，具體實現未知，那你依賴其他人jar包時，整個工程就只會用到log4j日誌框架，這是一種典型的門面模式應用，與jvm思想相同，我們面向slf4j寫日誌代碼，slf4j處理具體日誌實現框架之間的差異，正如我們面向jvm寫java代碼，jvm處理作業系統之間的差異，結果就是，一處編寫，到處運行。況且，現在越來越多的開源工具都在用slf4j了

　　那麼，怎麼用slf4j呢？

　　首先，得弄到slf4j的jar包，maven依賴如下，log4j配置過程完全不變

|  |  |
| --- | --- |
| 1  2  3  4  5 | <dependency>      <groupId>org.slf4j</groupId>      <artifactId>slf4j-api</artifactId>      <version>1.7.21</version>  </dependency> |

　　然後，弄到slf4j與log4j的關聯jar包，通過這個東西，將對slf4j介面的調用轉換為對log4j的調用，不同的日誌實現框架，這個轉換工具不同

|  |  |
| --- | --- |
| 1  2  3  4  5 | <dependency>      <groupId>org.slf4j</groupId>      <artifactId>slf4j-log4j12</artifactId>      <version>1.7.21</version>  </dependency> |

　　當然了，slf4j-log4j12這個包肯定依賴了slf4j和log4j，所以使用slf4j+log4j的組合只要配置上面這一個依賴就夠了

　　最後，代碼裡聲明logger要改一下，原來使用log4j是這樣的

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | import org.apache.log4j.Logger;  class Test {      final Logger log = Logger.getLogger(Test.class);      public void test() {          log.info("hello this is log4j info log");      }  } |

　　現在要改成這樣

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | import org.slf4j.Logger;  import org.slf4j.LoggerFactory;  class Test {      Logger log = LoggerFactory.getLogger(Test.class);      public void test() {          log.info("hello, my name is {}", "chengyi");      }  } |

　　依賴的Logger變了，而且，slf4j的api還能使用預留位置，很方便

（完）