1. **Tổng quan về log4j**

Logging : là quá trình của viết những trạng thái của chương trình ở những giai đoạn khác nhau khi thực hiện nó tới file hoặc repo nào đó. Với logging, những câu lệnh quan trọng liên quan đến việc thực hiện chương trình của bạn có thể được gửi tới text file, console hoặc repo bất kì nào đó. Với việc bổ sung logging trong bất kì project nào đó, 1 giải pháp kiểm tra và gỡ rối đáng tin cậy có thể được thực hiện.

* Log4j : là 1 logging framework đơn giản, nhanh và linh hoạt.

Log4j điều khiển việc thêm các lệnh log trong code của ứng dụng và quản lý chúng ở bên ngoài mà không cần chạm vào code của app, bởi sử dụng các file cấu hình bên ngoài. Nó sẽ phân chia các lệnh log theo các tiêu chuẩn của người dùng và ấn định mức độ ưu tiên cho những lệnh log đó.

Log4j để người dùng chọn từ nhiều đích cho các lệnh log như console, file, database, …

* Lợi ích của log4j :

Nó sẽ tổ chức các log trong những nhóm tách biệt, làm cho dễ dàng để tìm nguồn của lỗi.

Log4j ấn định mức độ ưu tiên cho logger và log request, nó sẽ xóa bỏ những log đầu ra không cần thiết và chỉ đồng ý những lệnh log quan trọng để được log.

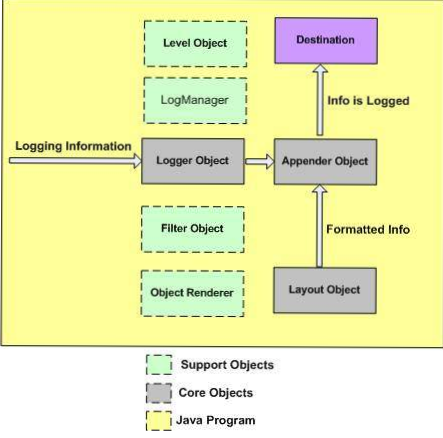
1. **Kiến trúc của log4j** :

* Có 2 loại obj với log4j framework:
* Core objects : là những obj ủy thác của framework, được yêu cầu để sử dụng framework.
* Support objects : là những obj tùy chọn của framework. Nó hỗ trợ core objs để đóng vai trò chức năng thêm nhưng có vai trò quan trọng.
* Core objects : bao gồm những loại obj :
* Logger obj : là thành phần quan trọng nhất của quá trình logging, chịu trách nhiệm cho việc bắt những thông tin logging và lưu trữ trong namespace hierarchy.
* Layout obj : tầng layout cung cấp những objs đã từng định dạng thông tin logging trong những kiểu khác nhau. Nó cung cấp sự hỗ trợ cho appender objs trước khi đăng thông tin logging.

Layout objs đóng vai trò quan trọng trong việc đăng thông tin logging theo cách mà con người có thể đọc được và tái sử dụng được.

* Appender obj : chịu trách nhiệm cho việc đăng thông tin logging tới nhiều đích đến được ưu tiên như database, file, console, …

Mô hình dưới thể hiện thành phần của log4j framework :



* Support objs : đây là những obj quan trọng khác trong log4j framework
* Level obj : định nghĩa độ ưu tiên của bất kì thông tin logging nào. Có 7 mức độ của logging :

+ ALL: tất cả các mức bao gồm cả các mức tự định nghĩa

+ DEBUG: chỉ định các thông tin sử dụng để gỡ lỗi ứng dụng

+ ERROR: Chỉ định các sự kiện lỗi mà vẫn có thể cho phép các ứng dụng để tiếp tục chạy

+ FATEL: chỉ định các sự kiện lỗi rất nghiêm trọng gây chết ứng dụng

+ INFO: chỉ định các thông tin liên quan tới xử lý ứng dụng

+ OFF: mức không ghi log

+ TRACE: chỉ định các thông tin trên mức DEBUG

+ WARN: Chỉ định các tình huống tiềm tàng có thể gây nên lỗi của ứng dụng

*Nguyên tắc ghi log như sau: một yêu cầu ghi log ở mức p trong một logger với mức q sẽ được kích hoạt neesy p>=q. Đây là quy tắc chính trong Log4j. Thứ tự các mức mặc định như sau: ALL < DEBUG < INFO < WARN < ERROR < FATAL < OFF. Như vậy muốn ghi các log ở mức DEBUG và INFO thì nên thiết lập mức log là WARN*

*Thường thì đang phát triển bạn thiết lập mức ghi log là DEBUG để hiển thị tất cả các log từ mức DEBUG trở lên, nhưng khi ứng dụng chạy thật thi nên thiết lập ghi log là INFO hoặc ERROR để tránh ghi log quá nhiều gây chậm ứng dụng*

* Filter obj : dùng để phân tích thông tin logging và tạo quyết định rằng liệu thông tin có nên được log hay không.

1 appender obj có thể có vài filter obj liên kết với nó. Nếu thông tin logging được chuyển tới 1 appender obj cụ thể, tất cả filter obj liên kết với appender đó cần được xác nhận thông tin logging đó trước khi nó có thể được đăng tới đích đến.

* ObjectRenderer : cung cấp sự thể hiện String của các obj khác nhau được chuyển tới logging framework. Obj này được dùng bởi Layout obj để chuẩn bị cho thông tin logging cuối cùng.
* LogManager obj : quản lý logging framework. Nó chịu trách nhiệm cho việc đọc thông số cấu hình đầu vào từ file cấu hình system-wide hoặc 1 class cấu hình.

1. **Sử dụng log4j**

Có ba bước

B1: thêm thư viện Log4j và project

|  |
| --- |
| <**dependency**>  <**groupId**>log4j</**groupId**>  <**artifactId**>log4j</**artifactId**>  <**version**>1.2.17</**version**> </**dependency**> |

B2. Cấu hình Log4j

Cách 1: Gọi hàm cấu hình mặc định trong org.apache.log4j.BasicConfigurator.

|  |
| --- |
| BasicConfigurator.*configure*(); |

Khi sử dụng cấu hình bằng cách sử dụng methor BasicConfigurator.configure() mặc định nó sẽ sử dụng ConsoleAppender và Patternlayout cho các logger

Cách 2: cấu hình trực tiếp trong code:

|  |
| --- |
| Properties properties = new Properties();  properties.setProperty("log4j.rootLogger", "TRACE,stdout,MyFile");  properties.setProperty("log4j.rootCategory", "TRACE");  properties.setProperty("log4j.appender.stdout", "org.apache.log4j.ConsoleAppender");  properties.setProperty("log4j.appender.stdout.layout","org.apache.log4j.PatternLayout");  properties.setProperty("log4j.appender.stdout.layout.ConversionPattern", "%d{yyyy/MM/dd HH:mm:ss.SSS} [%5p] %t (%F) - %m%n");  properties.setProperty("log4j.appender.MyFile","org.apache.log4j.RollingFileAppender");  properties.setProperty("log4j.appender.MyFile.File", "my\_example.log");  properties.setProperty("log4j.appender.MyFile.MaxFileSize", "100KB");  properties.setProperty("log4j.appender.MyFile.MaxBackupIndex", "1");  properties.setProperty("log4j.appender.MyFile.layout","org.apache.log4j.PatternLayout");  properties.setProperty("log4j.appender.MyFile.layout.ConversionPattern", "%d{yyyy/MM/dd HH:mm:ss.SSS} [%5p] %t (%F) - %m%n");  PropertyConfigurator.configure(properties); |

Cách 3: Dùng tệp cấu hình properties hoặc XML. Mặc định Log4j sẽ tự động load tệp cấu hình, nếu có hai file thì sẽ ưu tiên dùng file xml, có ba vị trí có thể in log ra là stdout, DB, file

Cách 1: in ra stdout

+ dùng file log4j.properties

|  |
| --- |
| **log4j.rootLogger**=**ERROR, stdout** *# Redirect log messages to console* **log4j.appender.stdout**=**org.apache.log4j.ConsoleAppender log4j.appender.stdout.Target**=**System.out log4j.appender.stdout.layout**=**org.apache.log4j log4j.appender.stdout.layout.ConversionPattern**=**%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n** |

+ Dùng file log4j.xml

|  |
| --- |
| <**appender name="MyConsole" class="org.apache.log4j.ConsoleAppender"**>  <**param name="Target" value="System.out"** />  <**layout class="org.apache.log4j.PatternLayout"**>  <**param name="ConversionPattern" value="%-4r [%t] %-5p %c %x - %m%n"** />  </**layout**> </**appender**>  <**root**>  <**priority value="ERROR"** />  <**appender-ref ref="MyConsole"** /> </**root**> |

Cách 2: in ra file

File xml

|  |
| --- |
| <**appender name="file" class="org.apache.log4j.RollingFileAppender"**>  <**param name="File" value="logg.log"** />  <**layout class="org.apache.log4j.PatternLayout"**>  <**param name="ConversionPattern"  value="%d{dd MMM yyyy HH:mm:ss,SSS} {%t} %-5p %c %x - %m%n"** />  </**layout**> </**appender**>  <**root**>  <**priority value="ERROR"**></**priority**>  <**appender-ref ref="file"** /> </**root**> |

File properties

|  |
| --- |
| **log4j.rootLogger**=**ERROR, stdout, file**  **log4j.appender.file**=**org.apache.log4j.RollingFileAppender log4j.appender.myAppender.DatePattern**=**'.'yyyy-MM-dd log4j.appender.file.File**=**testlog.log log4j.appender.file.MaxFileSize**=**1Kb log4j.appender.file.MaxBackupIndex**=**1 log4j.appender.file.layout**=**org.apache.log4j.SimpleLayout log4j.appender.file.layout.ConversionPattern**=**%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n** |

Cách 3: in vào db

Tạo bảng trong DB

|  |
| --- |
| **CREATE TABLE** logs (  **user\_id VARCHAR**(20) **NOT NULL**,  **dated** *DATE* **NOT NULL**,  **logger VARCHAR**(50) **NOT NULL**,  **level VARCHAR**(10) **NOT NULL**,  **message VARCHAR**(1000) **NOT NULL** ); |

Cấu hình file xml

|  |
| --- |
| <**appender name="DB" class="org.apache.log4j.jdbc.JDBCAppender"**>  <**param name="url" value="jdbc:postgresql://localhost:5432/postgres"**/>  <**param name="driver" value="org.postgresql.Driver"**/>  <**param name="user" value="postgres"**/>  <**param name="password" value="PhanHanh"**/>  <**param name="sql" value="INSERT INTO logs(user\_id, dated, logger, level, message) values ('%x','%d','%C','%p','%m')"**/>  <**layout class="org.apache.log4j.PatternLayout"**>  </**layout**> </**appender**>  <**logger name="log4j.rootLogger" additivity="false"**>  <**level value="DEBUG"**/>  <**appender-ref ref="DB"**/> </**logger**> |

Cấu hình file properties

|  |
| --- |
| **log4j.appender.jdbc**=**org.apache.log4j.jdbc.JDBCAppender log4j.appender.jdbc.Driver**=**org.postgresql.Driver log4j.appender.jdbc.URL**=**jdbc:postgresql://localhost:5432/postgres log4j.appender.jdbc.user**=**postgres log4j.appender.jdbc.password**=**PhanHanh log4j.appender.jdbc.layout**=**org.apache.log4j.PatternLayout log4j.appender.jdbc.sql**=**INSERT INTO logs(user\_id, dated, logger, level, message) values ('%x','%d','%C','%p','%m')** |

B3: Khai báo trong class

|  |
| --- |
| Log ***log*** = LogFactory.*getLog*(Main.**class**); |

Có các kiểu layout là

+ HTMLlayout

+PattenLayout

+ SimpleLayout

Các thẻ trong file properties

|  |
| --- |
| <!-- render -->  <renderer></renderer>  <renderer></renderer>  <renderer></renderer>    <!-- appender -->  <appender></appender>  <appender></appender>  <appender></appender>    <!-- logger -->  <logger></logger>  <logger></logger>  <logger></logger>    <!-- root -->  <root></root>  <root></root>  <root></root> |