LEARNING log4j

#log4j

Table of Contents

About

Chapter 1: Getting started with log4j

Remarks

This section provides an overview of what log4j is, and why a developer might want to use it.

It should also mention any large subjects within log4j, and link out to the related topics. Since the Documentation for log4j is new, you may need to create initial versions of those related topics.

Log4j Lifecycle

Log4j 1.x is end-of-life as of August 5, 2015. [1][2]. Apache Log4j 2 is the successor to Log4j 1.x.

1 https://logging.apache.org/log4j/1.2/
[2] https://blogs.apache.org/foundation/entry/apache_logging_services_project_announces

Versions

Version	Notice	Release Date
2.8	latest version	2017-01-21
2.7		2016-10-02
2.6.2		2016-07-09
2.4	Log4j 2.4 and greater requires Java 7	2015-09-20
2.3.6	last version that support java 6	2015-05-15
2.0	first stable version of branch 2.x. Breaks api compatibility. Use bridge: log4j-1.2-api.jar	2014-07-01
1.2.17	EOF log4j branch 1.x	2015-08-05

Examples

Installation and Setup

Installation

Installation of Log4j2 is as simple as putting log4j2 jar in application classpath. Though you might want to customize logs output through additional config file

Configuration

maven

To add log4j to project in maven, add it's dependency: In pom.xml add following dependency:

```
<dependencies>
  <dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>${log4j2.version}</version>
  </dependency>
  <dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>${log4j2.version}</version>
  </dependency>
</dependencies>
```

springboot with maven

Spring-boot is commonly used framework for web application. It features support auto-configuration for many features including logging façade like log4j2. To add log4j2 to your spring-boot project make sure you exclude default logging façade: commons-logging. Log4j will be used, when it is only logging façade on classpath.

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
        <!-- exclude spring-boot java commons logging in favour of log4j2 -->
        <exclusions>
            <exclusion>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-logging</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
    <!-- add log4j2 to spring-boot: -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-log4j2</artifactId>
    </dependency>
```

Notice, that there is no version in above snippet. It is because project inherit version from parent. Make sure you also inherit from spring-boot-starter-parent, by adding:

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.3.3.RELEASE</version>
</parent>
```

ivy

In ivy.xml, add following dependency:

```
<dependencies>
  <dependency org="org.apache.logging.log4j" name="log4j-api" rev="${log4j2.version}" />
  <dependency org="org.apache.logging.log4j" name="log4j-core" rev="${log4j2.version}" />
</dependencies>
```

gradle

In your .gradle file:

```
dependencies {
  compile group: 'org.apache.logging.log4j', name: 'log4j-api', version: '2.6.2'
  compile group: 'org.apache.logging.log4j', name: 'log4j-core', version: '2.6.2'
}
```

Read Getting started with log4j online: https://riptutorial.com/log4j/topic/5845/getting-started-with-log4j

Chapter 2: configuration

Examples

Log4j property file

Below is a configuration file for log4j. Log4j2 can use the same syntax, but there are different appender classes:

```
log4j.rootLogger=INFO, FOO

## ConsoleAppender
log4j.appender.CA=org.apache.log4j.ConsoleAppender
log4j.appender.CA.layout=org.apache.log4j.PatternLayout
log4j.appender.CA.layout.ConversionPattern= %d{hh:mm:ss,SSS} [%t] %-5p %c %x - %m%n

## FileAppender
log4j.appender.FOO=org.apache.log4j.RollingFileAppender
log4j.appender.FOO.File=${catalina.home}/logs/app.log
log4j.appender.FOO.Append=true
log4j.appender.FOO.layout=org.apache.log4j.PatternLayout
log4j.appender.FOO.layout.ConversionPattern= %d{hh:mm:ss,SSS} [%t] %-5p %c %x - %m%n

## attaching appender to specific package:
log4j.logger.com.example.package=INFO, CA
```

Directive `log4j.rootLogger` defines log level and appender for any class that doesn't meet `logger` criteria. Notice that appender `name` is defined after word 'appender'.

Resolve run time issue with log4j configuration

Users may face the following issue:

```
 log4j:WARN No appenders could be found for logger (dao.hsqlmanager).
 log4j:WARN Please initialize the log4j system properly.
 log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
```

One reason this can occur is if the log4j.properties or .xml file is not located within the project itself. (This can happen when you ship a tool, where the tool/JAR is in one directory and all the configuration is another directory).

You will then need to specify the path to the log4j.properties or .xml file. In the command line utility,

```
java -Dlog4j.configuration=file:///path/To/log4j.properties YourProject.jar
```

or if you have a script to run the tool you can add

`-Dlog4j.configuration=file:///path/To/log4j.properties` to the place where you do the equivalent action of the command line version. Not that `log4j.configuration` is specified in a URL format, prefixed by `file:///`.

Read configuration online: https://riptutorial.com/log4j/topic/7510/configuration

Chapter 3: log4j based loggers

Introduction

Pros and Cons of different loggers which can be used to create a log4j-formatted log to be viewed using the Log4View viewer.

I will review 3 loggers in this article, Log4cxx, Log4cplus and Log4cpp.

Examples

Log4cxx

https://logging.apache.org/log4cxx/

- currently undergoing Incubation - there is no official release
- update/bug fixes once in the past 12 years, last release was 2008
- user can select different LogLevels – TRACE, DEBUG, INFO, WARN, ERROR, and FATAL
- hierarchical Loggers
- it is possible to log asynchronously
- supports multiple appenders
- user can select to enabled or disabled the logger
- log can be sent to different and multiple output targets
- user selected output formats
- well documented
- is licensed under the Apache License, an open source license certified by the Open Source Initiative

Log4cplus

https://sourceforge.net/projects/log4cplus/

- updates/bug fixes - last release was Jan. 2016
- user can select select different LogLevels – TRACE, DEBUG, INFO, WARN, ERROR, and FATAL hierarchical Loggers
- supports multi–threaded applications but is not safe to be used from asynchronous signals' handlers
- user selected output format: SimpleLayout, PatternLayout, TTCCLayout
- supports multiple loggers
- not well documented

- is licensed under the Apache License V2.0

Log4cpp

https://sourceforge.net/projects/log4cpp/

- bug fixes are about once a year, last release was April 2015
- supports multi-threaded applications •no clear documentation exist
- is licensed under the GNU Lesser General Public License (LGPL) as of version 0.2.1, before that have been released under the GPL.

Read log4j based loggers online: https://riptutorial.com/log4j/topic/10570/log4j-based-loggers

Chapter 4: log4j logs customization

Examples

Configuration file

Configuration

Log4j configuration file can be in any of these formats:

- JSON
- YAML
- properties (text file)
- XML

Configuration discovery

1. Log4j will inspect the `log4j.configurationFile` system property and, if set, will attempt to load the configuration.
2. If no system property is set log4j will look for `log4j2-test.properties` in the classpath.
3. If no such file is found the log4j will look for `log4j2-test.yaml` or `log4j2-test.yml` in the classpath.
4. If no such file is found the log4j will look for `log4j2-test.json` or `log4j2-test.jsn` in the classpath.
5. If no such file is found the logj4 will look for `log4j2-test.xml` in the classpath.
6. If a test file cannot be located the log4j will look for `log4j2.properties` on the classpath.
7. If a properties file cannot be located the log4j will look for `log4j2.yaml` or `log4j2.yml` on the classpath.
8. If a YAML file cannot be located the log4j will look for `log4j2.json` or `log4j2.jsn` on the classpath.
9. If a JSON file cannot be located the log4j will try to locate `log4j2.xml` on the classpath.
10. If no configuration file could be located the DefaultConfiguration will be used. This will cause logging output to go to the console.

XML Configuration

XML Example

The below configuration configures two appenders (log output). The first logs to standard system output (console) and the other logs to file. In this example, the location of the file can be set statically in configuration (appender `file`) or dynamically via maven filtering feature (`<Property name="APPENDER">`). The logs in file will be packed by day. The log line format `Conversion Pattern` is set as a variable.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- 'status' sets log level for parsing configuration file itself -->
<Configuration status="INFO">
<Properties>
    <!-- Sets variable PID, if it's not present in log4j context will take value as below: -->
    <Property name="PID">????</Property>
    <!-- Sets variable 'LOG_PATTERN', defining how log line will look like -->
    <Property name="LOG_PATTERN">%clr{%d{yyyy-MM-dd HH:mm:ss.SSS}}{faint} %clr{%5p}
%clr{${sys:PID}}{magenta} %clr{%X{usr}}{green} %clr{---}{faint}%clr{[%15.15t]}{faint} %clr{%-
40.40c{1.}:%l}{cyan} %clr{:}{faint} %m%n%wEx</Property>
    <!-- LOG_DIR may be set by maven filtering feature: -->
    <Property name="LOG_DIR">@logging.path@</Property>
    <!-- APPENDER may be set by maven filtering feature, to set 'console' or 'file' appender:
-->
    <Property name="APPENDER">@logging.default.appender@</Property>
</Properties>
<Appenders>
    <!-- Sets console output: -->
    <Console name="console" target="SYSTEM_OUT">
        <PatternLayout pattern="${LOG_PATTERN}"/>
    </Console>
    <!-- Sets output to file, and names it 'file' -->
    <RollingRandomAccessFile append="true" fileName="${LOG_DIR}/log4j2.log"
        filePattern = "${LOG_DIR}/log4j2.%d{yyyy-MM-dd}.nr%i.log.gz" name="file">
        <PatternLayout>
            <Pattern>${LOG_PATTERN}</Pattern>
        </PatternLayout>
        <Policies>
            <TimeBasedTriggeringPolicy />
        </Policies>
    </RollingRandomAccessFile>
</Appenders>
<Loggers>
    <!-- Sets debug for Class 'com.example.package.Clazz', and attaches to file 'file' -->
    <Logger name="com.example.package.Clazz" level="debug">
        <AppenderRef ref="file"/>
    </Logger>
    <!-- Sets 'info' for package 'org.springframework' -->
    <Logger name="org.springframework" level="info" />
    <Root level="WARN">
        <AppenderRef ref="${APPENDER}"/>
    </Root>
</Loggers>
</Configuration>
```

Read log4j logs customization online: https://riptutorial.com/log4j/topic/6848/log4j-logs-customization

Credits

S. No	Chapters	Contributors
1	Getting started with log4j	Augustin Ghauratto, bn., Community, Michaël K.
2	configuration	Augustin Ghauratto, Ishnark
3	log4j based loggers	Merav Kochavi
4	log4j logs customization	Augustin Ghauratto, whrrgarbl