log4j.jar + commons-loggin.jar一起使用

<!-- https://mvnrepository.com/artifact/log4j/log4j -->

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

1. 项目的src目录下新建log4j.properties配置文件，文件内容大致如下

# 第一个参数配置的是日志输出级别（后面讲）

log4j.rootLogger=DEBUG,Console,File,DailyRollingFile,RollingFile

# **Console** 控制台输出日志

log4j.appender.Console=org.apache.log4j.ConsoleAppender  #日志要输出的地方

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

log4j.appender.Console.layout.ConversionPattern=%d [%t] %-5p [%c] - %m%n #日志格式（后面涉及）

# **File** 文件中输出日志

log4j.appender.File = org.apache.log4j.FileAppender

log4j.appender.File.Encoding=UTF-8 #保存编码格式

log4j.appender.File.File = C://log2.log #日志文件的位置

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

log4j.appender.File.layout.ConversionPattern =%d [%t] %-5p [%c] - %m%n #日志输出格式

# **DailyRollingFile** 按天输出日志

log4j.appender.DailyRollingFile = org.apache.log4j.DailyRollingFileAppender

log4j.appender.DailyRollingFile.File = C://log3.log

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

log4j.appender.DailyRollingFile.layout.ConversionPattern =%d [%t] %-5p [%c] - %m%n

# **RollingFile** 根据日志尺寸输出，日志文件达到一定大小就新建一个日志文件

log4j.appender.RollingFile = org.apache.log4j.RollingFileAppender

log4j.appender.RollingFile.File = C://log4.log

# 日志文件最大所占空间

log4j.appender.RollingFile.MaxFileSize=1KB

# 日志文件最多量，超过了就覆盖

log4j.appender.RollingFile.MaxBackupIndex=100

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

log4j.appender.RollingFile.layout.ConversionPattern =%d [%t] %-5p [%c] - %m%n

1. 使用

每个类中加入如下语句，xxx.class替换成本类名

**private** **static** Logger *logger* = Logger.*getLogger*(xxx.**class**);

*logger*.info(i);

1. log4j将日志从低到高分为OFF < FATAL < ERROR < WARN < INFO < DEBUG < ALL；指定哪一级别，比该级别低的日志都会输出，高的不会输出，比如配置了INFO，那么FATAL、ERROR、WARN、INFO都会输出，而DEBUG不会输出。
   1. logger.error("错误",e);或者logger.error("错误")
   2. logger.warn("警告",e);或者logger.warn("警告")
   3. logger.info(("消息",e);或者logger.info("消息")
2. 日志格式参照

|  |  |
| --- | --- |
| %d | 日期时间，默认格式为ISO8601，支持自定义格式，如：%d{yyyy/MM/dd HH:mm:ss,SSS}。 |
| %r | 自应用程序启动到输出该log信息耗费的毫秒数。 |
| %t | 产生该日志事件的线程名。 |
| %l | 日志事件的发生位置，相当于%c.%M(%F:%L)的组合，包括类全名、方法、文件名以及在代码中的行数。例如：test.TestLog4j.main(TestLog4j.java:10)。 |
| %c | 日志信息所属的类目，通常就是所在类的全名。%-20c："-"号表示左对齐 |
| %M | 产生日志信息的方法名。 |
| %F | 日志消息产生时所在的文件名称。 |
| %L | 代码中的行号。 |
| %m | 代码中指定的具体日志信息。 |
| %n | 一个回车换行符，Windows平台为"\r\n"，Unix平台为"\n"。 |
| %x | 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 |