# 一、Java日志简介

**1、在Java开发中，常用的日志记录框架有log4j、log4j2、logback、JUL。**

**JCL(Jakarta Commons Logging)**：比较流行的日志框架，非常多框架都依赖JCL，比如Spring等。

**SLF4j**：提供新的API，初衷是配合Logback使用，但同一时候兼容Log4j，提供接口，不提供实现类。

**Log4j**：经典的一种日志解决方式。内部把日志系统抽象封装成Logger 、appender 等实现。我们能够通过配置文件轻松的实现日志系统的管理和多样化配置。

**Log4j2**：Log4j的2.0版本号。对Log4j进行了优化。比方支持參数API、支持异步appender、插件式架构等。

**Logback**：Log4j的替代产品。和log4j同作者，须要配合日志框架SLF4j使用。

**JUL**(java.util.logging)：JDK提供的日志系统。

# 二、slf4j、log4j、log4j2、logback关系

**Slf4j**是一个门面日志，只提供接口，不提供实现类。

**Log4j**可以单独使用，也可以用slf4j的标准来使用，只需引入log4j-over-slf4j.jar。

**logback**同样是由log4j的作者设计完成的，拥有更好的特性，用来取代log4j的一个日志框架,是slf4j的原生实现(即直接实现了slf4j的接口，而log4j并没有直接实现，所以就需要一个适配器slf4j-log4j12.jar)。

log4j与log4j2的区别：

1.获取Logger的api不一样，log4j的api为org.apache.log4j.Logger，而log4j2的api为org.apache.logging.log4j.Logger  
2.配置方式不一样，log4j2对properties的配置支持不是很好，它的格式一般为xml格式或者yaml格式，这种格式的可读性比较好，各种配置一目了然  
3.Log4j2.0基于LMAX Disruptor的异步日志在多线程环境下性能会远远优于Log4j 1.x和logback（官方数据是10倍以上）。

# 三、配置xml、properties(log4j为例)

1、Log4j有三个主要的组件：Loggers(记录器)，Appenders(输出源)和Layouts(布局)。Log4j支持两种配置文件格式，一种是XML格式的文件，一种是properties属性文件。

1.1、配置appender

<appender name=*"UNCATEGORIZED"* class=*"org.apache.log4j.RollingFileAppender"*>

<param name=*"File"* value=*"log/uncategorized.log"* />

<param name=*"MaxFileSize"* value=*"10MB"* />

<param name=*"MaxBackupIndex"* value=*"5"* />

<param name=*"Append"* value=*"true"* />

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

<param name=*"ConversionPattern"* value=*"%d %p [%c] - %m%n"* />

</layout>

<filter class=*"org.apache.log4j.varia.LevelRangeFilter"*>

<param name=*"LevelMin"* value=*"info"* />

<param name=*"LevelMax"* value=*"fatal"* />

</filter>

</appender>

(1). 输出方式appender一般有5种：

org.apache.log4j.RollingFileAppender(滚动文件，自动记录最新日志)

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.WriterAppender (将日志信息以流格式发送到任意指定的地方)

(2). 日记记录的优先级priority，优先级由高到低分为

OFF ,FATAL ,ERROR ,WARN ,INFO ,DEBUG ,ALL。

Log4j建议只使用FATAL ,ERROR ,WARN ,INFO ,DEBUG这五个级别。

(3). 格式说明layout中的参数都以%开始，后面不同的参数代表不同的格式化信息（参数按字母表顺序列出）：

%c        输出所属类的全名，可在修改为 %d{Num} ,Num类名输出的维（如："org.apache.elathen.ClassName",%C{2}将输出elathen.ClassName）

%d       输出日志时间其格式为 %d{yyyy-MM-dd HH:mm:ss,SSS}，可指定格式 如 %d{HH:mm:ss}

%l        输出日志事件发生位置，包括类目名、发生线程，在代码中的行数

%n       换行符

%m      输出代码指定信息，如info(“message”),输出message

%p       输出优先级，即 FATAL ,ERROR 等

%r        输出从启动到显示该log信息所耗费的毫秒数

%t        输出产生该日志事件的线程名

1.2、配置logger

<logger additivity=*"false"* name=*"com.sinosoft.application.prpall.mb"*>

<level value=*"trace"* />

<appender-ref ref=*"simple\_stdout"* />

</logger>

当additivity="false"时，root中的配置就失灵了，不遵循缺省的继承机制。

Name在此名的所有记录器都遵循同样的配置。

Level是日志输出的级别。

1.3、配置properties

#log4j.rootLogger = [ level ] , appenderName1 , appenderName2 , …

log4j.rootLogger=error, stdout,logfile

#stdout控制器（标准输出）

log4j.appender.stdout=org.apache.log4j.ConsoleAppender #设置Appender

log4j.appender.stdout.Target=System.out #设置输出目标

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout #设置布局

log4j.appender.stdout.layout.ConversionPattern=%d %p [%c]:%L - %m%n #设置输出格式

#输出日志文件

log4j.appender.logfile=org.apache.log4j.RollingFileAppender #设置Appender

log4j.appender.logfile.File=pss.log #默认输出到tomcat的根路径或者运行测试工程的根路径

log4j.appender.logfile.MaxFileSize=512KB #设置日志文件的最大内存大小

# Keep three backup files.

log4j.appender.logfile.MaxBackupIndex=3 #设置日志文件的最大数量，当日志文件达到最大指定内存时则另外再生成一个文件存放日志

log4j.appender.logfile.Append = true #ture则将日志信息增加到指定文件中，false则将日志信息覆盖指定文件内容，默认是true

log4j.appender.logfile.Threshold = ERROR #设置只输出ERROR级别以上的日志

# Pattern to output: date priority [category] - message

log4j.appender.logfile.layout=org.apache.log4j.PatternLayout #设置布局

log4j.appender.logfile.layout.ConversionPattern=%d %p [%c] - %m%n #设置输出格式

举个栗子(影像系统)：

#全局日志等级配置,输出位置

log4j.rootLogger=INFO, Console, RollingFile

#设置Appender

log4j.appender.Console=org.apache.log4j.ConsoleAppender

#设置布局

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

#设置输出格式

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

#设置Appender

log4j.appender.RollingFile=org.apache.log4j.DailyRollingFileAppender

#默认输出到tomcat的根路径或者运行测试工程的根路径

log4j.appender.RollingFile.File=log/jeesite/jeesite.log

#设置布局

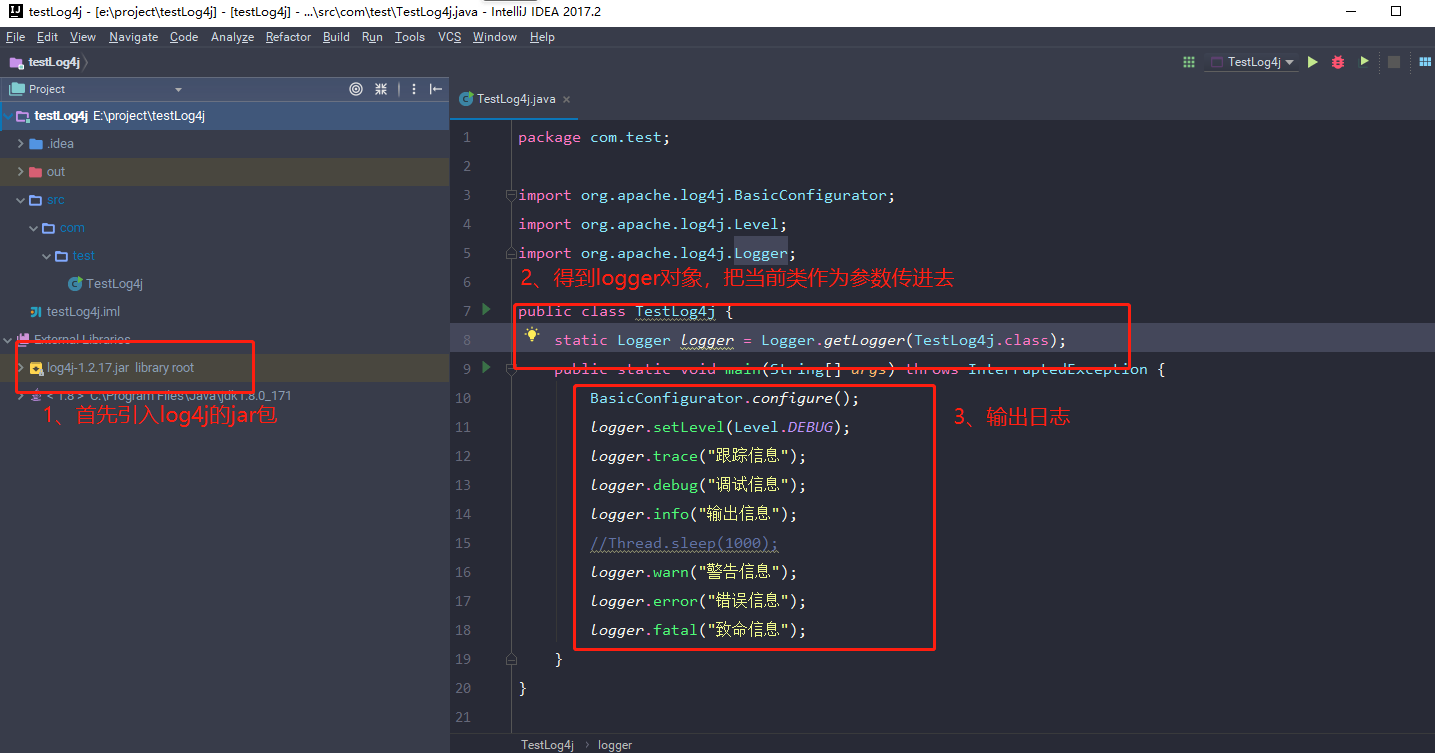
log4j.appender.RollingFile.layout=org.apache.log4j.PatternLayout

#设置输出格式

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

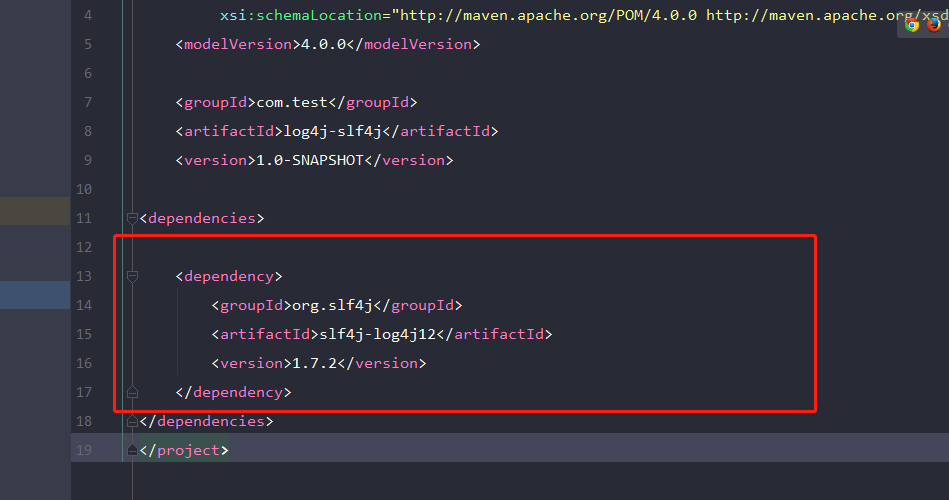
# log4j简单使用

1. 直接使用log4j

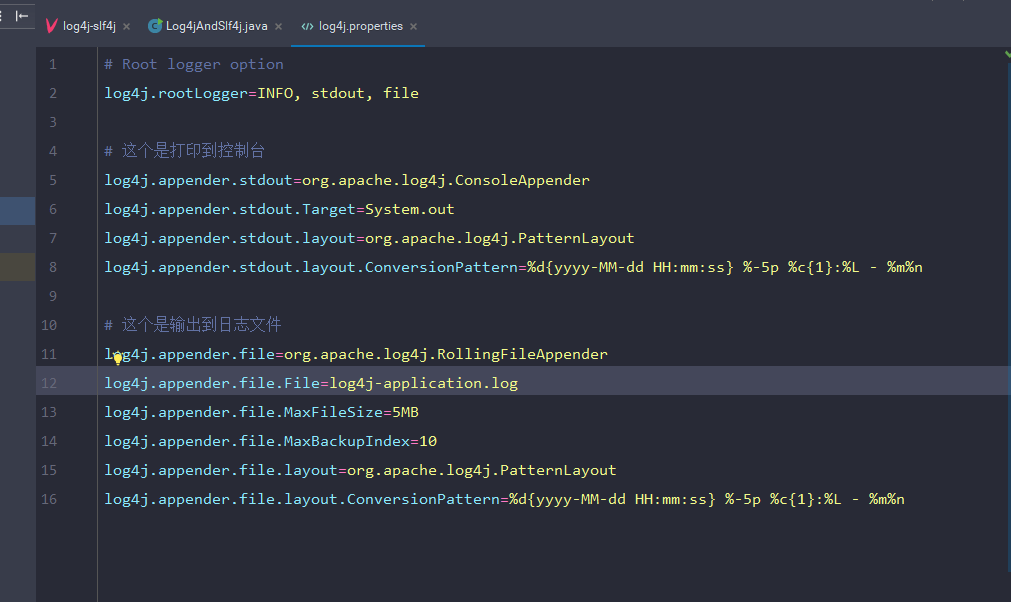


1. log4j升级为slf4j标准

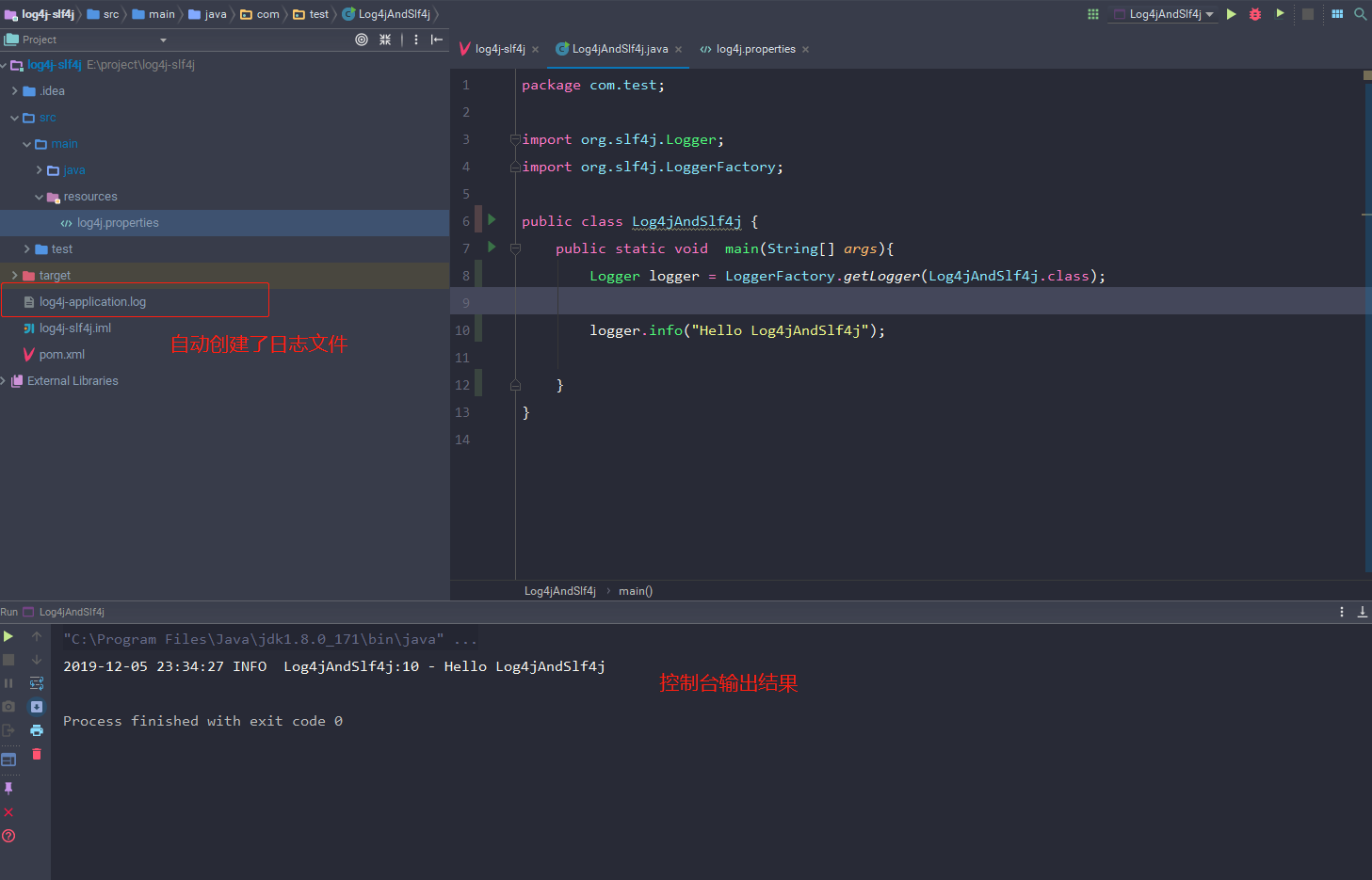
以maven为例，pom文件，导入依赖

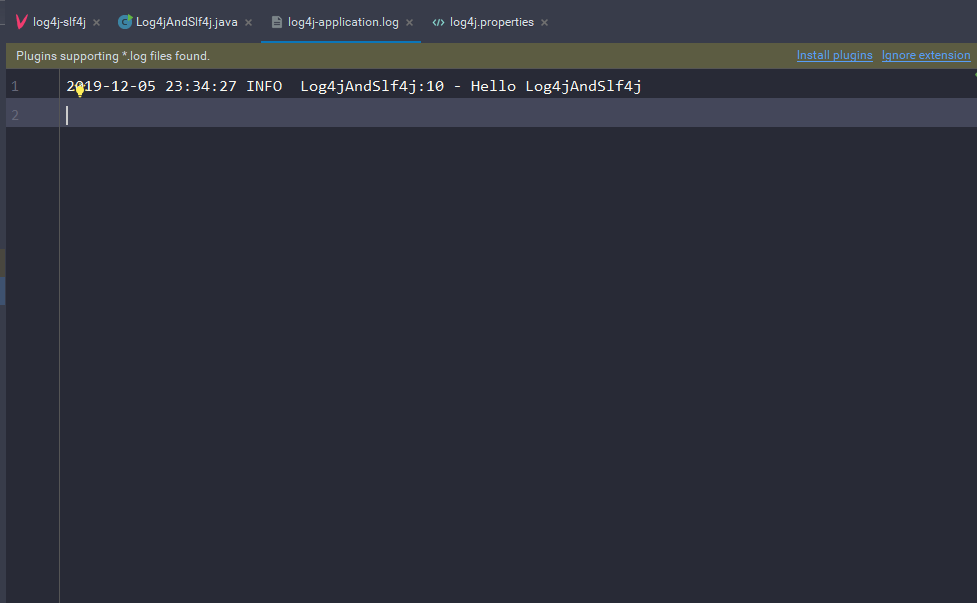


在resource下加入log4j.properties文件



输出日志





# 五、tomcat日志详解

1、tomcat 对应日志的配置文件：tomcat目录下的/conf/logging.properties。

tomcat 的日志等级有：日志输出级别：SEVERE (最高级别) > WARNING > INFO > CONFIG > FINE > FINER > FINEST (所有内容,最低级别)。

2、tomcat  有五类日志 ：catalina、localhost、manager、admin、host-manager

2.1、**catalina.out**即标准输出和标准出错，所有输出到这两个位置的都会进入catalina.out，这里包含tomcat运行自己输出的日志以及应用里向console输出的日志。默认这个日志文件是不会进行自动切割的。

2.2、**catalina.{yyyy-MM-dd}.log**是tomcat自己运行的一些日志，这些日志还会输出到catalina.out，但是应用向console输出的日志不会输出到catalina.{yyyy-MM-dd}.log,它是tomcat的启动和暂停时的运行日志，****注意，它和catalina.out是里面的内容是不一样的。****

2.3、**localhost.{yyyy-MM-dd}.log**主要是应用初始化(listener, filter, servlet)未处理的异常最后被tomcat捕获而输出的日志,它也是包含tomcat的启动和暂停时的运行日志,但它没有catalina.2018-09-19.log 日志全。它只是记录了部分日志。

2.4、**localhost\_access\_log.2018-09-19.txt** 这个是访问tomcat的日志，请求时间和资源，状态码都有记录。

2.5、**manager.2018-09-19.log**这个是tomcat manager项目专有的日志文件。

2.6、**host-manager.2018-09-19.log**这个估计是放tomcat的自带的manager项目的日志信息的，未看到有什么重要的日志信息。