

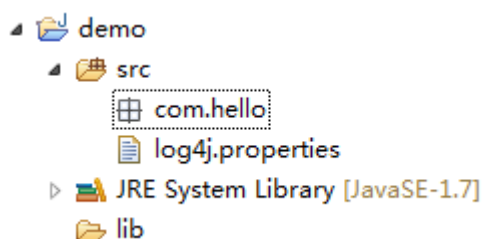
# 一、Log4j 简介

Log4j ( log for java ) 是我们常用的日志管理小助手，它是 Apache 的一个开源项目，我们可以通过这个小助手很方便的将日志信息输出到控制台、文件、GUI组件，甚至是套接口服务器、NT的事件记录器、UNIX Syslog守护进程等，而且也能控制每一条日志的输出格式，并且设置每一条日志信息的级别，这样我们能够更加细致地控制日志的生成过程。最令人兴奋的就是，这些所有的配置都可以通过一个配置文件来灵活地进行配置，而不需要修改应用的代码。下面我们就通过一个 小栗子 来让你体验这位小助手的强大之处。

## 二、小栗子

### 1、准备工作

我们先新建一个项目，在项目下新建一个文件夹，起名叫 lib 。



### 2、加入 Log4j 的 jar包

加入 log4j-1.2.15.jar ( 可以选择 Log4j 更高版本哦 ) 到 lib 下。

### 3、在src下建立 log4j.properties ，其内容如下：

```
log4j.rootLogger=INFO,console,file

log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d %5p [%t] (%F:%L) - %m%n

log4j.appender.file=org.apache.log4j.FileAppender
log4j.appender.file.File=src/log.txt
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d %5p [%t] (%F:%L) - %m%n
```

### 3、新建一个测试类，其内容如下：

```
package com.hello;

import java.io.IOException;

import org.apache.log4j.Logger;

import org.apache.log4j.PropertyConfigurator;
```

```

public class Main {

    public static void main(String[] args) throws IOException {

        // 获取日志管理器
        Logger logger = Logger.getLogger(Main.class);
        // 日志管理器配置文件位置
        PropertyConfigurator.configure("src/log4j.properties");

        // 日志记录
        logger.warn("考试开始！");
        logger.info("开始计算1+1");
        System.out.println("1 + 1 = " + (1+1));
        logger.info("开始计算1/0");
        try {
            System.out.println("1 / 1 = " + (1/0));
        } catch (Exception e) {
            logger.error("除数不能为0！！");
        }
        logger.warn("考试结束！");

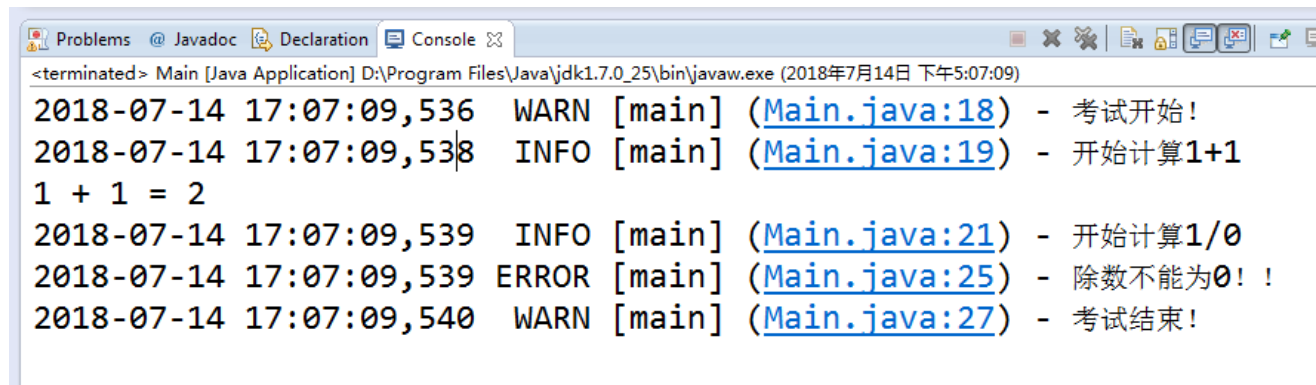
    }

}

```

#### 4、好了，大功告成！我们来看看效果

在控制台上输出了以下内容：



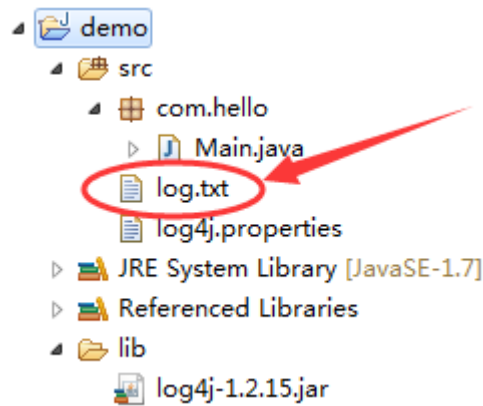
The screenshot shows an IDE console window with the following content:

```

<terminated> Main [Java Application] D:\Program Files\Java\jdk1.7.0_25\bin\javaw.exe (2018年7月14日 下午5:07:09)
2018-07-14 17:07:09,536 WARN [main] (Main.java:18) - 考试开始!
2018-07-14 17:07:09,538 INFO [main] (Main.java:19) - 开始计算1+1
1 + 1 = 2
2018-07-14 17:07:09,539 INFO [main] (Main.java:21) - 开始计算1/0
2018-07-14 17:07:09,539 ERROR [main] (Main.java:25) - 除数不能为0!!
2018-07-14 17:07:09,540 WARN [main] (Main.java:27) - 考试结束!

```

细心的小伙伴们会在 `src` 下发现多了一个 `log.txt` 的文件，里面记录了一些信息。



```
Main.java  log4j.properties  log.txt x
1 2018-07-14 17:07:09,536 WARN [main] (Main.java:18) - 考试开始!
2 2018-07-14 17:07:09,538 INFO [main] (Main.java:19) - 开始计算1+1
3 2018-07-14 17:07:09,539 INFO [main] (Main.java:21) - 开始计算1/0
4 2018-07-14 17:07:09,539 ERROR [main] (Main.java:25) - 除数不能为0!!
5 2018-07-14 17:07:09,540 WARN [main] (Main.java:27) - 考试结束!
6 |
```

### 三、配置文件中的常用属性说明

#### 1、日志管理级别

```
log4j.rootLogger=INFO,console,file
```

这句话表示日志管理的级别为 INFO，它要将日志信息输出到 console 和 file 这两个地方，console 和 file 的定义在下面的代码中，它们可以任意起名。

级别 ( Level )	描述	级别
ALL	各级包括自定义级别	1
DEBUG	指定细粒度信息事件是最有用的应用程序调试	2
INFO	指定能够突出在粗粒度级别的应用程序运行情况的信息的消息	3
WARN	指定具有潜在危害的情况	4
ERROR	错误事件可能仍然允许应用程序继续运行	5
FATAL	指定非常严重的错误事件，这可能导致应用程序中止	6
OFF	这是最高等级，为了关闭日志记录	7

日志管理的级别如上表，级别从高到低（1最高）。如果我们配置的是 OFF 则不打出任何信息；如果配置为 INFO 这样只显示比 INFO 级别低的 log 信息。

## 2、输出端

```
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.file=org.apache.log4j.FileAppender
```

这两句话表示输出端是控制台和日志文件。当然也可以指定其他地方，如下表：

输出端 ( Appender )	说明	值
控制台	只会在控制台上显示	org.apache.log4j.ConsoleAppender
日志文件	只会产生一个日志文件	org.apache.log4j.FileAppender
日志文件	每天产生一个日志文件	org.apache.log4j.DailyRollingFileAppender
日志文件	文件大小到达指定尺寸的时候产生一个新的文件	org.apache.log4j.RollingFileAppender
任意地方	将日志信息以流格式发送到任意指定的地方	org.apache.log4j.WriterAppender

如果输出端是日志文件的话，则需要指定存放文件的位置，如下面这句话，它指定了生成日志文件的位置：

```
log4j.appender.file.File=src/log.txt
```

## 3、输出端布局类型

```
log4j.appender.console.layout=org.apache.log4j.PatternLayout
```

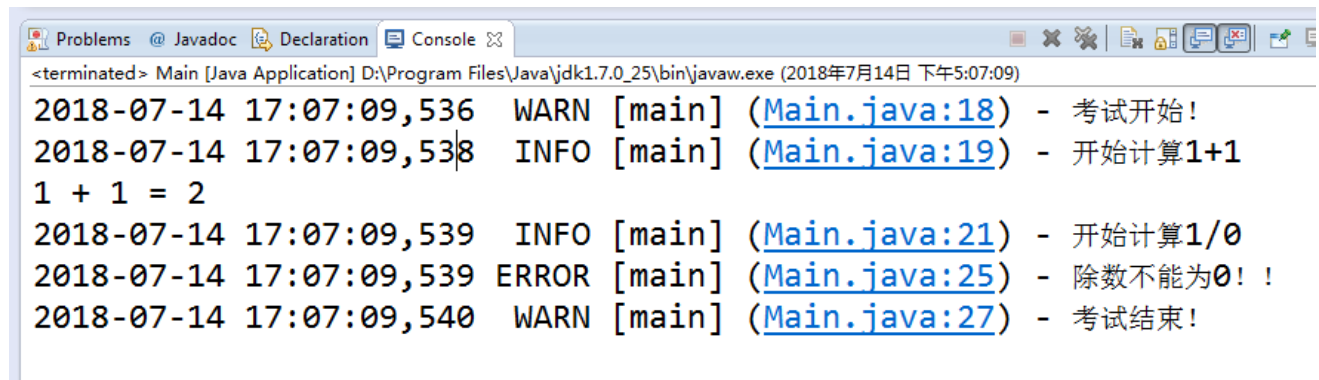
这句话表示输出端布局类型是灵活布局类型，当然也可以是其他类型，如下图：

输出端布局类型 ( layout )	说明	值
HTML表格布局类型	以HTML表格形式布局	org.apache.log4j.HTMLLayout
灵活的布局类型	可以灵活地指定布局模式	org.apache.log4j.PatternLayout
简单的布局类型	包含日志信息的级别和信息字符串	org.apache.log4j.SimpleLayout
复杂的布局类型	包含日志产生的时间、线程、类别等等信息	org.apache.log4j.TTCCLayout

## 4、日志信息格式

```
log4j.appender.console.layout.ConversionPattern=%d %5p [%t] (%F:%L) - %m%n
```

这句话表示灵活布局模式中的日志信息格式，此显示效果的格式是：时间 优先级(格式长度为5个字符)（文件名称:代码行号） - 日志信息。显示效果如下图：



The screenshot shows an IDE console window with the following log output:

```
<terminated> Main [Java Application] D:\Program Files\Java\jdk1.7.0_25\bin\javaw.exe (2018年7月14日 下午5:07:09)
2018-07-14 17:07:09,536 WARN [main] (Main.java:18) - 考试开始!
2018-07-14 17:07:09,538 INFO [main] (Main.java:19) - 开始计算1+1
1 + 1 = 2
2018-07-14 17:07:09,539 INFO [main] (Main.java:21) - 开始计算1/0
2018-07-14 17:07:09,539 ERROR [main] (Main.java:25) - 除数不能为0!!
2018-07-14 17:07:09,540 WARN [main] (Main.java:27) - 考试结束!
```

灵活布局模式中还提供了其他的日志信息格式，如下表：

格式 化符 号	说明
%p	输出日志信息的优先级，即DEBUG，INFO，WARN，ERROR，FATAL。
%d	输出日志时间点的日期或时间，默认格式为ISO8601，也可以在其后指定格式，如： %d{yyyy/MM/dd HH:mm:ss,SSS}。
%r	输出自应用程序启动到输出该log信息耗费的毫秒数。
%t	输出产生该日志事件的线程名。
%l	输出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	输出日志信息所属的类目，通常就是所在类的全名。
%M	输出产生日志信息的方法名。

另外，还可以在%与格式字符之间加上修饰符来控制其最小长度、最大长度、和文本的对齐方式。如：1)c：指定输出category的名称，最小的长度是20，如果category的名称长度小于20的话，默认的情况下右对齐。

2)%-20c：“-”号表示左对齐。3)%.30c：指定输出category的名称，最大的长度是30，如果category的名称长度大于30的话，就会将左边多出的字符截掉，但小于30的话也不会补空格。