Log4j 配置信息详解：

Log4J 的配置文件(Configuration File)就是用来设置记录器的级别、存放器和布局的，它可接 key=value 格式的设置或 xml 格式的设置信息。通过配置，可以创建出 Log4J 的运行环境。

1. 配置文件

Log4J 配置文件的基本格式如下：

```
#配置根 Logger
log4j.rootLogger = [ level ] ，appenderName1，appenderName2，…

#配置日志信息输出目的地 Appender
log4j.appender.appenderName = fully.qualified.name.of.appender.class
    log4j.appender.appenderName.option1 = value1
    …
    log4j.appender.appenderName.optionN = valueN

#配置日志信息的格式（布局）
log4j.appender.appenderName.layout = fully.qualified.name.of.layout.class
    log4j.appender.appenderName.layout.option1 = value1
    …
    log4j.appender.appenderName.layout.optionN = valueN
```

其中 **[level]** 是日志输出级别，共有 5 级：

DEBUG	**7**
INFO	**6**
WARN	**4**
ERROR	**3**
FATAL	**0**

Appender 为日志输出目的地，Log4j 提供的 appender 有以下几种：

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.RollingFileAppender（文件大小到达指定尺寸的时候产生一个新的文件），
org.apache.log4j.WriterAppender（将日志信息以流格式发送到任意指定的地方）

Layout：日志输出格式，Log4j 提供的 layout 有以下几种：

org.apache.log4j.HTMLLayout（以 HTML 表格形式布局），
org.apache.log4j.PatternLayout（可以灵活地指定布局模式），
org.apache.log4j.SimpleLayout（包含日志信息的级别和信息字符串），
org.apache.log4j.TTCCLayout（包含日志产生的时间、线程、类别等等信息）

打印参数: Log4J 采用类似 C 语言中的 printf 函数的打印格式格式化日志信息，如下:

%m 输出代码中指定的消息
%p 输出优先级，即 DEBUG，INFO，WARN，ERROR，FATAL

%r 输出自应用启动到输出该 log 信息耗费的毫秒数

%c 输出所属的类目，通常就是所在类的全名

%t 输出产生该日志事件的线程名

%n 输出一个回车换行符，Windows 平台为“\r\n”，Unix 平台为“\n”

%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，比如：%d{yyy MMM dd H H:mm:ss , SSS}，输出类似：2002 年 10 月 18 日 22 ： 10 ： 28 ， 921

%l 输出日志事件的发生位置，包括类目名、发生的线程，以及在代码中的行数。举例：Testlog4.main(TestLog 4.java: 10)

2. 在代码中初始化 Logger:

1）在程序中调用 **BasicConfigurator.configure()**方法：给根记录器增加一个 ConsoleAppender，输出格式通过 PatternLayout 设为**"%-4r [%t] %-5p %c %x - %m%n"**，还有根记录器的默认级别是 **Level.DEBUG**.

2）配置放在文件里，通过命令行参数传递文件名字，通过 **PropertyConfigurator.configure(args[x])**解析并配置；

3）配置放在文件里，通过环境变量传递文件名等信息，利用 log4j 默认的初始化过程解析并配置；

4）配置放在文件里，通过应用服务器配置传递文件名等信息，利用一个特殊的 servlet 来完成配置。

3. 为不同的 Appender 设置日志输出级别：

当调试系统时，我们往往注意的只是异常级别的日志输出，但是通常所有级别的输出都是放在一个文件里的，如果日志输出的级别是 BUG！？那就慢慢去找吧。

这时我们也许会想要是能把异常信息单独输出到一个文件里该多好啊。当然可以，Log4j 已经提供了这样的功能，我们只需要在配置中修改 **Appender** 的 **Threshold** 就能实现,比如下面的例子：

[配置文件]

```
### set log levels ###
log4j.rootLogger = debug ， stdout ， D ， E

### 输出到控制台 ###
log4j.appender.stdout = org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target = System.out
log4j.appender.stdout.layout = org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern =  %d{ABSOLUTE} %5p %c{ 1 }:%L - %m%n

### 输出到日志文件 ###
log4j.appender.D = org.apache.log4j.DailyRollingFileAppender
log4j.appender.D.File = logs/log.log
log4j.appender.D.Append = true
log4j.appender.D.Threshold = DEBUG ## 输出 DEBUG 级别以上的日志
log4j.appender.D.layout = org.apache.log4j.PatternLayout
log4j.appender.D.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss}  [ %t:%r ] - [ %p ]  %m%n

### 保存异常信息到单独文件 ###
log4j.appender.D = org.apache.log4j.DailyRollingFileAppender
log4j.appender.D.File = logs/error.log ## 异常日志文件名
log4j.appender.D.Append = true
log4j.appender.D.Threshold = ERROR ## 只输出 ERROR 级别以上的日志!!!
log4j.appender.D.layout = org.apache.log4j.PatternLayout
log4j.appender.D.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss}  [ %t:%r ] - [ %p ]  %m%n
```

如下是一个配置较好的 log4j.properties 文件：

```
# Output pattern : date [thread] priority category - message
log4j.rootLogger=WARN, Console, RollingFile

#Console
log4j.appender.Console=org.apache.log4j.ConsoleAppender
log4j.appender.Console.Threshold = INFO
log4j.appender.Console.layout=org.apache.log4j.PatternLayout
log4j.appender.Console.layout.ConversionPattern=%d [%t] %-5p [%c] - %m%n

#RollingFile
log4j.appender.RollingFile=org.apache.log4j.DailyRollingFileAppender
log4j.appender.RollingFile.Threshold =ERROR
log4j.appender.RollingFile.File=logs/mypro.log
log4j.appender.RollingFile.layout=org.apache.log4j.PatternLayout
log4j.appender.RollingFile.layout.ConversionPattern=%d [%t] %-5p [%c] - %m%n

#Project defalult level
log4j.logger.com.oracle.lnsd=INFO
```

这里配置的日志文件输出路径不确切，有可能在 tomcat 路径下，有可能在 d 盘，有可能在项目根路径，所以最好修改为 WEB-INF/logs 路径下，那么如下方式可以实现：
将以上位置的
log4j.appender.RollingFile.File=logs/mypro.log
修改为：
log4j.appender.RollingFile.File=${webappHome}/logs/mypro.log
然后创建一个 ServletContextListener，设置一个环境变量即可：

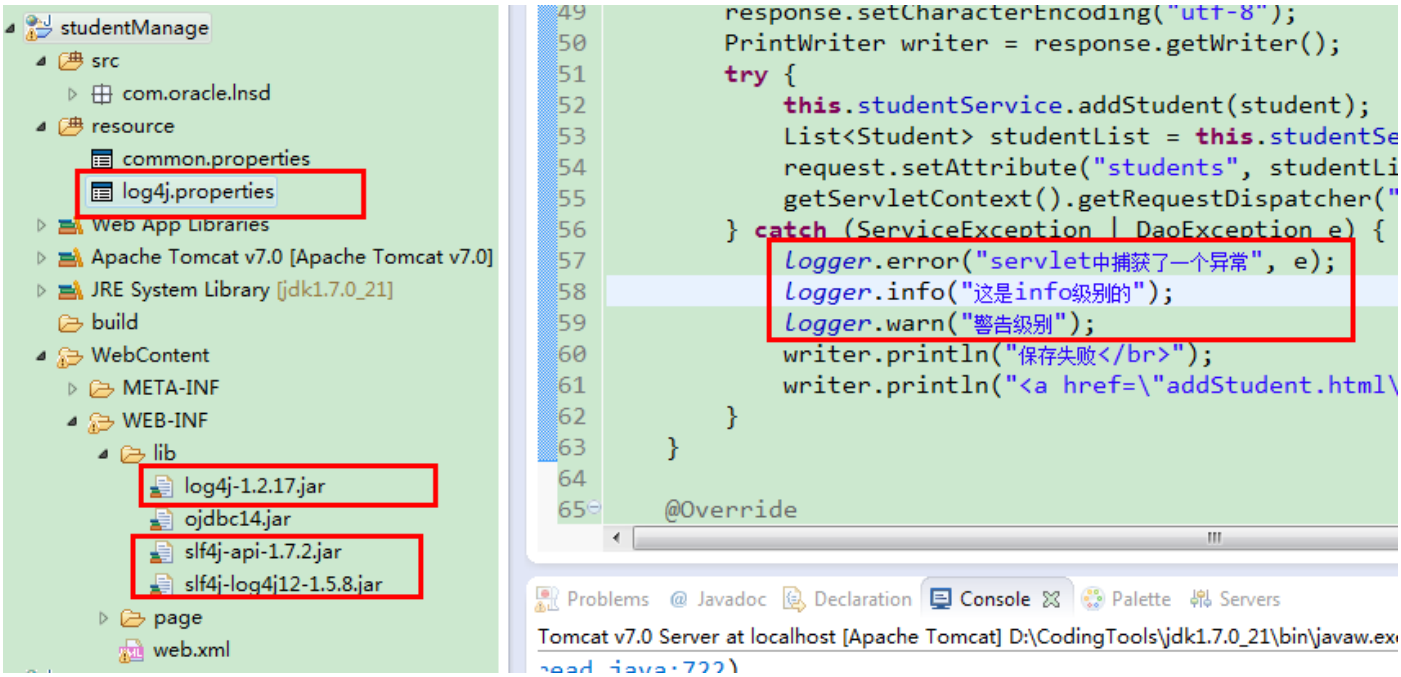
```
public class CustomListener implements ServletContextListener {

    /**
     * @see ServletContextListener#contextInitialized(ServletContextEvent)
     */
    public void contextInitialized(ServletContextEvent sce) {
        //给 log4j 指定日志输出的路径
        System.setProperty("webappHome", sce.getServletContext().getRealPath("/WEB-INF"));
    }

    /**
     * @see ServletContextListener#contextDestroyed(ServletContextEvent)
     */
    public void contextDestroyed(ServletContextEvent sce) {
    }

}
```

项目中添加的文件：



Logger 的初始化：

```
public class StudentAddServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    /**
     * 创建静态Logger
     */
    private static Logger logger =  LoggerFactory.getLogger(StudentAddServlet.class);

    private StudentService studentService = new StudentService();
```