**Java中Log4j使用笔记**

**1.log4j的作用**

使用Log4J，程序员可以【控制日志信息输送的目的地】，包括控制台，文件，GUI组件和NT事件记录器，也可以控制每一条日志的【输出格式】，或通过定义每一条日志信息的级别，更加细致地控制日志的生成过程。

**2.Log4j的组成：**  
 Log4j由三个重要的组成构成：日志记录器(Loggers)，输出端(Appenders)和日志格式化器(Layout)。  
 Logger：控制要启用或禁用哪些日志记录语句，并对日志信息进行级别限制： Appenders指定了日志将打印到控制台还是文件中;而Layout则控制日志信息的显示格式。  
  
3.**Logger对象的获得或创建**  
 Logger被指定为实体，由一个String类的名字识别。Logger的名字是大小写敏感的，且名字之间具有继承关系，子名用父名作为前缀，用点“.”分隔，例如x.y是x.y.z的父亲。

root Logger(根Logger)是所有Logger的祖先，它有如下属性：  
 a.它总是存在的。  
 b.它不可以通过名字获得。  
root Logger可以通过以下语句获得：

**public static Logger Logger.getRootLogger();**

或

**public static Logger Logger.getLogger(Class clazz)**

其中调用Logger.getLogger(Class clazz)是目前ogger对象最理想的方法。

4.**日志级别** 每个Logger都被了一个日志级别（log level），用来控制日志信息的输出。日志级别从高到低分为：  
 A：off         最高等级，用于关闭所有日志记录。  
 B：fatal       指出每个严重的错误事件将会导致应用程序的退出。  
 C：error      指出虽然发生错误事件，但仍然不影响系统的继续运行。  
 D：warm     表明会出现潜在的错误情形。  
 E：info         一般和在粗粒度级别上，强调应用程序的运行全程。  
 F：debug     一般用于细粒度级别上，对调试应用程序非常有帮助。  
 G：all           最低等级，用于打开所有日志记录。  
  
 上面这些级别是定义在org.apache.log4j.Level类中。Log4j只建议使用4个级别，优先级从高到低分别是error,warn,info和debug。通过使用日志级别，可以控制应用程序中相应级别日志信息的输出。  
 例如，如果使用b了info级别，则应用程序中所有低于info级别的日志信息(如debug)将不会被打印出来。

具体实例:

package log4j;

import org.apache.log4j.BasicConfigurator;

import org.apache.log4j.Level;

import org.apache.log4j.Logger;

public class Log4jTest {

public static void main(String[] args) {

Logger logger = Logger.getLogger(Log4jTest.class);

//使用默认的配置信息，不需要写log4j.properties

BasicConfigurator.configure();

//设置日志输出级别为info，这将覆盖配置文件中设置的级别

logger.setLevel(Level.INFO);

//下面的消息将被输出

logger.info("this is an info");

logger.warn("this is a warn");

logger.error("this is an error");

logger.fatal("this is a fatal");

}

}

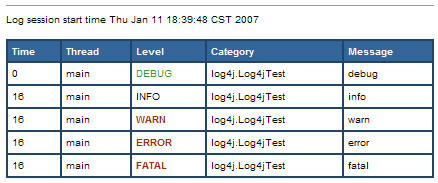
1. **输出端Appender**  
    Appender用来指定日志信息输出到哪个地方，可以同时指定多个输出目的地。Log4j允许将信息输出到许多不同的输出设备中，一个log信息输出目的地就叫做一个Appender。  
    每个Logger都可以拥有一个或多个Appender，每个Appender表示一个日志的输出目的地。

可以使用Logger.addAppender(Appender app)为Logger增加一个Appender，也可以使用Logger.removeAppender(Appender app)为Logger删除一个Appender。

以下为Log4j几种常用的输出目的地。  
 a：org.apache.log4j.ConsoleAppender：将日志信息输出到控制台。  
 b：org.apache.log4j.FileAppender：将日志信息输出到一个文件。  
 c：org.apache.log4j.DailyRollingFileAppender：将日志信息输出到一个日志文件，并且每天输出到一个新的日志文件。  
 d：org.apache.log4j.RollingFileAppender：将日志信息输出到一个日志文件，并且指定文件的尺寸，当文件大小达到指定尺寸时，会自动把文件改名，同时产生一个新的文件。  
 e：org.apache.log4j.WriteAppender：将日志信息以流格式发送到任意指定地方。  
 f:：org.apache.log4j.jdbc.JDBCAppender：通过JDBC把日志信息输出到数据库中。

1. **日志格式化器Layout**  
   有三种：  
   **1).HTMLLayout**

格式化日志输出为HTML表格形式：如下图所示：



**2).SimpleLayout**

以一种非常简单的方式格式化日志输出，它打印三项内容：级别-信息。如：INFO - info

**3).PatternLayout**

根据指定的转换模式格式化日志输出，或者如果没有指定任何转换模式，就使用默认的转化模式格式。  
下面的代码实现了SimpleLayout和FileAppender的程序。

public static void main(String[] args) {

Logger logger = Logger.getLogger(Log4jTest.class);

SimpleLayout layout = new SimpleLayout();

//HTMLLayout layout = new HTMLLayout();

FileAppender appender = null;

try

{

//把输出端配置到out.txt

appender = new FileAppender(layout,"out.txt",false);

}catch(Exception e)

{

}

logger.addAppender(appender);//添加输出端

logger.setLevel((Level)Level.DEBUG);//覆盖配置文件中的级别

logger.debug("debug");

logger.info("info");

logger.warn("warn");

logger.error("error");

logger.fatal("fatal");

}

1. **Log4j的配置** 配置Log4j环境就是指配置root Logger，包括把Logger为哪个级别，为它增加哪些Appender,以及为这些Appender设置Layout等等。因为所有其他的Logger都是root Logger的后代，所以它们都继承了root Logger的性质。这些可以通过设置系统属性的方法来隐式地完成，也可以在程序中调用XXXConfigurator.configure()方法来显式地完成。有以下几种方式来配置Log4j。

A：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。  
 B：配置放在文件里，通过应用服务器配置传递文件甸等信息，利用一个特定的Servlet来完成配置。  
 C：在程序中调用BasicConfigurator.configure()方法。  
 D：配置放在文件里，通过命令行PropertyConfigurator.configure(args[])解析log4j.properties文件并配置Log4j。

下面对BasicConfigurator.configure()方法和PropertyConfigurator.config()方法分别进行介绍。  
BasicConfigurator.configure()方法：  
它使用简单的方法配置Log4j环境。这个方法完成的任务是：  
1：用默认的方式创建PatternLayout对象p:

PatternLayout p = new PatternLayout("%-4r[%t]%-5p%c%x-%m%n");

2：用p创建ConsoleAppender对象a，目标是System.out，标准输出设备：

ConsoleAppender a = new ConsoleAppender(p,ConsoleAppender.SYSTEM\_OUT);

3：为root Logger增加一个ConsoleAppender p;

rootLogger.addAppender(a);

4：把rootLogger的log level设置为DUBUG级别;

rootLogger.setLevel(Level.DEBUG);

PropertyConfigurator.configure()方法：

当使用以下语句生成Logger对象时：

static Logger logger = Logger.getLogger(mycalss.class);

如果没有调用BasicConfigurator.configure()，PropertyConfigurator.configure()或DOMConfigurator.configure()方法，Log4j会自动加载CLASSPATH下名为log4j.properties的配置文件。如果把此配置文件改为其他名字，例如my.properties,程序虽然仍能运行，但会报出不能正确初始化Log4j系统的提示。这时可以在程序中加上：

PropertyConfigurator.configure("classes/my.properties");

问题即可解决。