<http://blog.csdn.net/yaohuanjames/article/details/52848790?locationNum=5&fps=1>

Log4j是Apache的一个开放源代码项目，通过使用Log4j，我们可以控制日志信息输送的目的地是控制台、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；我们也可以控制每一条日志的输出格式；通过定义每一条日志信息的级别，我们能够更加细致地控制日志的生成过程。

Log4j由三个重要的组件构成：日志信息的优先级，日志信息的输出目的地，日志信息的输出格式。日志信息的优先级从高到低有ERROR、WARN、 INFO、DEBUG，分别用来指定这条日志信息的重要程度；日志信息的输出目的地指定了日志将打印到控制台还是文件中；而输出格式则控制了日志信息的显示内容。

其实您也可以完全不使用配置文件，而是在代码中配置Log4j环境。但是，使用配置文件将使您的应用程序更加灵活。

Log4j支持两种配置文件格式，一种是XML格式的文件，一种是properties格式的文件。下面我们介绍使用properties格式做为配置文件的方法：

示例：

log4j.rootLogger=INFO, A1

log4j.appender.A1=org.apache.log4j.ConsoleAppender

log4j.appender.A1.layout=org.apache.log4j.PatternLayout

log4j.appender.A1.layout.ConversionPattern=%-4r %-5p [%t] %37c %3x - %m%n

1. 配置根Logger，其语法为：

log4j.rootLogger = [ level ] , appenderName, appenderName, …

其中，level 是日志记录的优先级，分为OFF、FATAL、ERROR、WARN、INFO、DEBUG、ALL或者您定义的级别。Log4j建议只使用四个级别，优先级从高到低分别是ERROR、WARN、INFO、DEBUG。通过在这里定义的级别，您可以控制到应用程序中相应级别的日志信息的开关。比如在这里定义了INFO级别，则应用程序中所有DEBUG级别的日志信息将不被打印出来。

appenderName就是指定日志信息输出到哪个地方。您可以同时指定多个输出目的地。

2. 配置日志信息输出目的地Appender，其语法为：

log4j.appender.appenderName = fully.qualified.name.of.appender.class

log4j.appender.appenderName.option1 = value1

…

log4j.appender.appenderName.option = valueN

其中，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）

(1).ConsoleAppender选项

Threshold=WARN:指定日志消息的输出最低层次。

ImmediateFlush=true:默认值是true,意谓着所有的消息都会被立即输出。

Target=System.err：默认情况下是：System.out,指定输出控制台

(2).FileAppender 选项

Threshold=WARN:指定日志消息的输出最低层次。

ImmediateFlush=true:默认值是true,意谓着所有的消息都会被立即输出。

File=mylog.txt:指定消息输出到mylog.txt文件。

Append=false:默认值是true,即将消息增加到指定文件中，false指将消息覆盖指定的文件内容。

(3).DailyRollingFileAppender 选项

Threshold=WARN:指定日志消息的输出最低层次。

ImmediateFlush=true:默认值是true,意谓着所有的消息都会被立即输出。

File=mylog.txt:指定消息输出到mylog.txt文件。

Append=false:默认值是true,即将消息增加到指定文件中，false指将消息覆盖指定的文件内容。

DatePattern=’.'yyyy-ww:每周滚动一次文件，即每周产生一个新的文件。当然也可以指定按月、周、天、时和分。即对应的格式如下：

  1)’.'yyyy-MM: 每月

  2)’.'yyyy-ww: 每周

  3)’.'yyyy-MM-dd: 每天

  4)’.'yyyy-MM-dd-a: 每天两次

  5)’.'yyyy-MM-dd-HH: 每小时

  6)’.'yyyy-MM-dd-HH-mm: 每分钟

(4).RollingFileAppender 选项

Threshold=WARN:指定日志消息的输出最低层次。

ImmediateFlush=true:默认值是true,意谓着所有的消息都会被立即输出。

File=mylog.txt:指定消息输出到mylog.txt文件。

Append=false:默认值是true,即将消息增加到指定文件中，false指将消息覆盖指定的文件内容。

MaxFileSize=100KB: 后缀可以是KB, MB 或者是 GB. 在日志文件到达该大小时，将会自动滚动，即将原来的内容移到mylog.log.1文件。

MaxBackupIndex=2:指定可以产生的滚动文件的最大数。

3. 配置日志信息的布局，其语法为：

log4j.appender.appenderName.layout = fully.qualified.name.of.layout.class

log4j.appender.appenderName.layout.option1 = value1

…

log4j.appender.appenderName.layout.option = valueN

其中，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）

4、输出格式设置

在配置文件中可以通过log4j.appender.A1.layout.ConversionPattern设置日志输出格式。

参数：

%p: 输出日志信息优先级，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，比如：%d{yyy MMM dd HH:mm:ss,SSS}，输出类似：2002年10月18日 22：10：28，921

%r: 输出自应用启动到输出该log信息耗费的毫秒数

%c: 输出日志信息所属的类目，通常就是所在类的全名

%t: 输出产生该日志事件的线程名

%l: 输出日志事件的发生位置，相当于%C.%M(%F:%L)的组合,包括类目名、发生的线程，以及在代码中的行数。举例：Testlog4.main(TestLog4.java:10)

%x: 输出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。

%%: 输出一个”%”字符

%F: 输出日志消息产生时所在的文件名称

%L: 输出代码中的行号

%m: 输出代码中指定的消息,产生的日志具体信息

%n: 输出一个回车换行符，Windows平台为”\r\n”，Unix平台为”\n”输出日志信息换行

可以在%与模式字符之间加上修饰符来控制其最小宽度、最大宽度、和文本的对齐方式。如：

1)%20c：指定输出category的名称，最小的宽度是20，如果category的名称小于20的话，默认的情况下右对齐。

2)%-20c:指定输出category的名称，最小的宽度是20，如果category的名称小于20的话，”-”号指定左对齐。

3)%.30c:指定输出category的名称，最大的宽度是30，如果category的名称大于30的话，就会将左边多出的字符截掉，但小于30的话也不会有空格。

4)%20.30c:如果category的名称小于20就补空格，并且右对齐，如果其名称长于30字符，就从左边交远销出的字符截掉。

**log4j应用实例**

测试log4j的项目结构



Log4j.properties的路径为    src/config/log4j

Log4j.properties文件的内容

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

复制代码

下面定义日志输出级别是 INFO，并且配置了2个输出目的地，一个是A3，一个是console

log4j.rootLogger = INFO,A3,CONSOLE

//日志最低的输出级别

log4j.appender.A3.Threshold=INFO

log4j.appender.A3.encoding=UTF-8

//每天产生一个文件DailyRollingFileAppender

log4j.appender.A3 = org.apache.log4j.DailyRollingFileAppender

//file 属性 指定产生日志文件的保存位置及文件名，这里是windows下的配置

// c:/logtest/logtest.log，

//公司项目在linux下的配置是/app/weblogic/applications/logs/sxvip\_logs

log4j.appender.A3.File=c:/logtest/logtest.log

//当有日志时立即输出，默认是true

log4j.appender.A3.ImmediateFlush=true

log4j.appender.A3.DatePattern='\_'yyyy-MM-dd

//日志布局方式

log4j.appender.A3.layout=org.apache.log4j.PatternLayout

//日志文件中日志的格式

log4j.appender.A3.layout.ConversionPattern=%-d{yyyy/MM/dd HH:mm:ss} OSS %-5p [%c] - %m%n

//这里使用org.apache.log4j.ConsoleAppender指定要把日志输出到控制台

log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender

log4j.appender.Threshold=INFO

//输出目标是 控制台

log4j.appender.CONSOLE.Target=System.out

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

log4j.appender.CONSOLE.layout.ConversionPattern=%-d{yyyy/MM/dd HH:mm:ss} OSS %-5p [%c] - %m%n

复制代码

在Web应用中web.xml利用Spring配置log4j

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

复制代码

在web.xml中添加配置

<!-- 配置log4j配置文件的路径，可以是xml或 properties（此参数必须配）-->

下面使用了classpath 参数指定log4j.properties文件的位置，这样log4j的配置文件就不用非要放到src的下面

<context-param>

<param-name>log4jConfigLocation</param-name>

<param-value>classpath:config/log4j/log4j.properties</param-value>

</context-param>

使用spring的监听器,当应用启动时来读取log4j的配置文件

<listener>

<listener-class>org.springframework.web.util.Log4jConfigListener</listener-class>

</listener>

复制代码

在java中使用log4j

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

复制代码

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

public class MyServiceParamAction extends BaseAdmAction{

private static final Log log =

LogFactory.getLog(MyServiceParamAction.class);

public ActionForward NgCallServiceInfo(ActionMapping mapping, ActionForm form,

HttpServletRequest request, HttpServletResponse response) {

// 获得登录用户信息,没有登录提示用户需要重新登录

log.info("我的服务查询开始.....................................");

UserBean user = (UserBean) request.getSession().getAttribute("USER\_INFO");

log.info("从session中获取登录用户user "+user);

if (user != null) {

log.info("user.getMisisdn() "+user.getMsisdn());

. . .

}

复制代码

当应用启动时，这里就会根据log4j的配置(log4j.appender.A3.File=c:/logtest/logtestxxx.log)在c盘下产生日志文件



当访问项目的资源时就会在该文件中添加日志信息

2013/12/05 11:46:33 OSS INFO  [com.sinovatech.myservice.action.MyServiceParamAction] - 我的服务查询开始...........

2013/12/05 11:46:33 OSS INFO  [com.sinovatech.myservice.action.MyServiceParamAction] - 从session中获取登录用户  user  null

2013/12/05 11:46:33 OSS INFO  [com.sinovatech.myservice.action.MyServiceParamAction] - VIP 我的服务  查询  结束........

注意当在java类的main方法中测试时，不会在日志文件中添加日志信息

在javaweb项目中应用

import org.apache.log4j.Logger;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.hsinghsu.testSSH.dao.UserDao;

import com.hsinghsu.testSSH.model.User;

import com.hsinghsu.testSSH.service.UserService;

@Service(value = "userService")

public class UserServiceImpl implements UserService{

 @Autowired

 private UserDao userDao;

// private final static Logger logger = Logger.getLogger(UserServiceImpl.class);

 private Logger logger = Logger.getLogger(this.getClass().getName());

 public boolean login(String name, String password)

 {

  logger.info("--UserServiceImpl login method name:"+name+" password:"+password);

  User user = userDao.getUserByName(name);

  if(user!=null)

  {

  if(password.equals(user.getPwd()))

  {

   return true;

  }

  }

  return false;

 }

}