<http://blog.csdn.net/u013142781/article/details/50405684>

版权声明：本文为博主原创文章，转载注明出处http://blog.csdn.net/u013142781

目录[(?)](http://blog.csdn.net/u013142781/article/details/50405684" \o "系统根据文章中H1到H6标签自动生成文章目录)[[+]](http://blog.csdn.net/u013142781/article/details/50405684)

## 一、关于Log4j日志

Log4j是Apache的一个开源项目，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。最令人感兴趣的就是，这些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。

在强调可重用组件开发的今天，除了自己从头到尾开发一个可重用的日志操作类外，Apache为我们提供了一个强有力的日志操作包-Log4j。

此外，通过Log4j其他语言接口，您可以在C、C++、.Net、PL/SQL程序中使用Log4j，其语法和用法与在Java程序中一样，使得多语言分布式系统得到一个统一一致的日志组件模块。而且，通过使用各种第三方扩展，您可以很方便地将Log4j集成到J2EE、JINI甚至是SNMP应用中。

## 二、Log4j使用前必须要了解的一些概念

**2.1、日志信息的优先级**

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。而且在企业中一般也只使用着四种。下面是各个级别的极介绍：

OFF 最高等级，用于关闭所有日志记录。   
FATAL 指出每个严重的错误事件将会导致应用程序的退出。   
ERROR 指出虽然发生错误事件，但仍然不影响系统的继续运行。   
WARN 表明会出现潜在的错误情形。   
INFO 一般和在粗粒度级别上，强调应用程序的运行全程。   
DEBUG 一般用于细粒度级别上，对调试应用程序非常有帮助。   
ALL 最低等级，用于打开所有日志记录。

**2.2、日志信息的输出目的地**

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。参考如下配置文件log4j.properties，其既配置了输出到控制台，也配置了输出到日志文件中：

### 设置###

log4j.rootLogger = debug,stdout,D,E

### 输出信息到控制抬 ###

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 = [%-5p] %d{yyyy-MM-dd HH:mm:ss,SSS} method:%l%n%m%n

### 输出DEBUG 级别以上的日志到=D://logs/debug.log ###

log4j.appender.D = org.apache.log4j.DailyRollingFileAppender

log4j.appender.D.File = D://logs/debug.log

log4j.appender.D.Append = true

log4j.appender.D.Threshold = DEBUG

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

### 输出ERROR 级别以上的日志到=D://logs/error.log ###

log4j.appender.E = org.apache.log4j.DailyRollingFileAppender

log4j.appender.E.File =D://logs/error.log

log4j.appender.E.Append = true

log4j.appender.E.Threshold = ERROR

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

log4j.appender.E.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss} [ %t:%r ] - [ %p ] %m%n

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24

其中log4j.appender.D.Threshold是一个过滤器，配置了表示输出该级别及其以上的日志，   
另外log4j.appender.D.Append = true一般输出到文件的都会配置，而且配置成true，true：表示添加，false：表示覆盖。

**2.3、配置日志信息的格式**

2.2中的log4j.properties配置文件中，应该会看log4j.appender.\*.layout，这表示输出的日志格式配置。

log4j.appender.\*.layout值的选择有以下几种：

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）

* 1
* 2
* 3
* 4

一般我们都采用第二种org.apache.log4j.PatternLayout，自定义格式，如2.2中的log4j.properties配置文件中，这种格式需要我们自定义，这里小宝鸽如下定义：

log4j.appender.\*.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss} [ %t:%r ] - [ %p ] %m%n

* 1

其中含义介绍：

%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“rn”，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 HH: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(TestLog4.java:10)

* 1
* 2
* 3
* 4
* 5
* 6
* 7

## 三、Log4j在java中的使用

本项目使用的环境：eclipse + maven 。

**3.1、新建maven项目**

（Packagding 选war，因为后面会讲到其在web项目中的使用，同样用这个新建的项目），如果不知道如何新建maven项目的，可以参考博主另外一篇文章：   
<http://blog.csdn.net/u013142781/article/details/50316383>

**3.2、在pom.xml添加Log4j的jar包依赖：**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.luo</groupId>

<artifactId>log4j</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<properties>

<!-- log4j日志文件管理包版本 -->

<slf4j.version>1.6.6</slf4j.version>

<log4j.version>1.2.12</log4j.version>

</properties>

<dependencies>

<!-- 日志文件管理包 -->

<!-- log start -->

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>${log4j.version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>${slf4j.version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>${slf4j.version}</version>

</dependency>

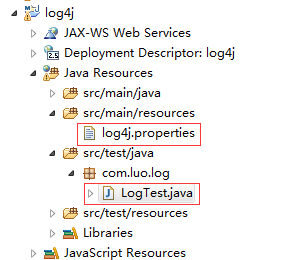
<!-- log end -->

</dependencies>

</project>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33

**3.3、添加配置文件和测试类**



log4j.properties内容如下：

### 设置###

log4j.rootLogger = debug,stdout,D,E

### 输出信息到控制抬 ###

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 = [%-5p] %d{yyyy-MM-dd HH:mm:ss,SSS} method:%l%n%m%n

### 输出DEBUG 级别以上的日志到=D://logs/debug.log ###

log4j.appender.D = org.apache.log4j.DailyRollingFileAppender

log4j.appender.D.File = D://logs/debug.log

log4j.appender.D.Append = true

log4j.appender.D.Threshold = DEBUG

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

### 输出ERROR 级别以上的日志到=D://logs/error.log ###

log4j.appender.E = org.apache.log4j.DailyRollingFileAppender

log4j.appender.E.File =D://logs/error.log

log4j.appender.E.Append = true

log4j.appender.E.Threshold = ERROR

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

log4j.appender.E.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss} [ %t:%r ] - [ %p ] %m%n

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24

LogTest.java内容如下：

package com.luo.log;

import org.apache.log4j.Logger;

public class LogTest {

private static Logger logger = Logger.getLogger(LogTest.class);

public static void main(String[] args) {

// 记录debug级别的信息

logger.debug("This is debug message.");

// 记录info级别的信息

logger.info("This is info message.");

// 记录error级别的信息

logger.error("This is error message.");

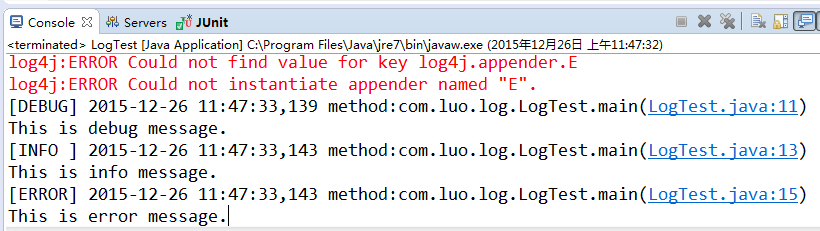
}

}

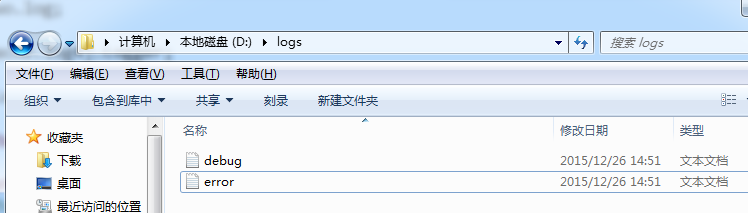
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18

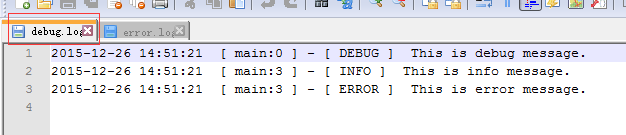
**3.4、运行结果**

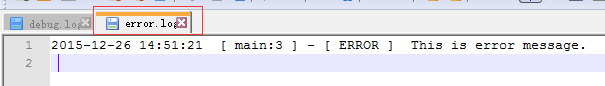
运行LogTest.java，会看到控制台如下结果：



另外，D:\logs下也会多了两个log文件，且内容如下图：







## 四、Log4j在web项目中的使用

继续是刚刚的那个工程，这里博主增加了spring + springmvc的配置，虽然spring配置这里是必须的，一般一个springmvc项目里面都会有spring。

**4.1、将它转成web项目**

转换方法，请阅读博主的另外一篇文章：   
<http://blog.csdn.net/u013142781/article/details/50380920>



**4.2、添加spring和springmvc以及javaee-api的依赖**

修改pom.xml如下：

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.luo</groupId>

<artifactId>log4j</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<properties>

<!-- log4j日志文件管理包版本 -->

<slf4j.version>1.6.6</slf4j.version>

<log4j.version>1.2.12</log4j.version>

<!-- spring版本号 -->

<spring.version>3.2.8.RELEASE</spring.version>

</properties>

<dependencies>

<!-- 日志文件管理包 -->

<!-- log start -->

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>${log4j.version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>${slf4j.version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>${slf4j.version}</version>

</dependency>

<!-- log end -->

<!-- 添加Spring依赖 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- javaee-web-api包 注意和项目使用的JDK版本对应 -->

<dependency>

<groupId>javax</groupId>

<artifactId>javaee-web-api</artifactId>

<version>6.0</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.activiti</groupId>

<artifactId>activiti-engine</artifactId>

<version>5.17.0</version>

</dependency>

<dependency>

<groupId>com.googlecode.xmemcached</groupId>

<artifactId>xmemcached</artifactId>

<version>1.3.2</version>

</dependency>

</dependencies>

</project>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57
* 58
* 59
* 60
* 61
* 62
* 63
* 64
* 65
* 66
* 67
* 68
* 69
* 70
* 71
* 72
* 73
* 74
* 75
* 76
* 77
* 78
* 79
* 80
* 81
* 82
* 83
* 84
* 85
* 86
* 87
* 88
* 89
* 90
* 91
* 92
* 93
* 94
* 95
* 96
* 97
* 98

**4.3、配置web.xml**

其实web项目中Log4j的使用与上面的区别就在于，web项目进行了log4j.properties的加载，web项目加上了如下配置：

<!-- 日志记录 -->

<context-param>

<!-- 日志配置文件路径 -->

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

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

</context-param>

<context-param>

<!-- 日志页面的刷新间隔 -->

<param-name>log4jRefreshInterval</param-name>

<param-value>6000</param-value>

</context-param>

<listener>

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

</listener>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15

配置文件web.xml的全部内容如下：

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"

id="WebApp\_ID" version="2.5">

<!-- 读取spring配置文件 -->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:application.xml</param-value>

</context-param>

<!-- 设计路径变量值 -->

<context-param>

<param-name>webAppRootKey</param-name>

<param-value>springmvc.root</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!-- 日志记录 -->

<context-param>

<!-- 日志配置文件路径 -->

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

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

</context-param>

<context-param>

<!-- 日志页面的刷新间隔 -->

<param-name>log4jRefreshInterval</param-name>

<param-value>6000</param-value>

</context-param>

<listener>

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

</listener>

<!-- springMVC核心配置 -->

<servlet>

<servlet-name>dispatcherServlet</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<!--spingMVC的配置路径 -->

<param-value>classpath:springmvc/spring-mvc.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<!-- 拦截设置 -->

<servlet-mapping>

<servlet-name>dispatcherSe

rvlet</servlet-name>

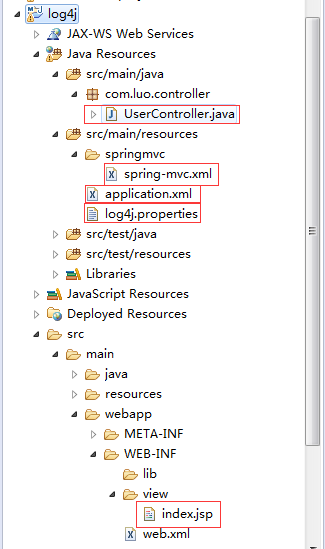
<url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57

**4.4、配置其他文件如下目录**



UserController.java内容如下：

package com.luo.controller;

import org.apache.log4j.Logger;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.servlet.ModelAndView;

@Controller

public class UserController {

private static Logger logger = Logger.getLogger(UserController.class);

@RequestMapping("/")

public ModelAndView getIndex(){

// 记录debug级别的信息

logger.debug("This is debug message.");

// 记录info级别的信息

logger.info("This is info message.");

// 记录error级别的信息

logger.error("This is error message.");

ModelAndView mav = new ModelAndView("index");

return mav;

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26

spring-mvc.xml内容如下：

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:p="http://www.springframework.org/schema/p"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:mvc="http://www.springframework.org/schema/mvc"

xsi:schemaLocation="

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.2.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.2.xsd

http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">

<mvc:annotation-driven />

<!-- 扫描controller（controller层注入） -->

<context:component-scan base-package="com.luo.controller"> </context:component-scan>

<!-- 对模型视图添加前后缀 -->

<bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver"

p:prefix="/WEB-INF/view/" p:suffix=".jsp"/>

</beans>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23

application.xml内容如下：

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:p="http://www.springframework.org/schema/p"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:mvc="http://www.springframework.org/schema/mvc"

xsi:schemaLocation="

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.2.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.2.xsd

http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">

<mvc:annotation-driven />

<!-- 扫描controller（controller层注入） -->

<context:component-scan base-package="com.luo.controller"> </context:component-scan>

<!-- 对模型视图添加前后缀 -->

<bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver"

p:prefix="/WEB-INF/view/" p:suffix=".jsp"/>

</beans>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23

log4j.properties内容如下：

### 设置###

log4j.rootLogger = debug,stdout,D,E

### 输出信息到控制抬 ###

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 = [%-5p] %d{yyyy-MM-dd HH:mm:ss,SSS} method:%l%n%m%n

### 输出DEBUG 级别以上的日志到=D://logs/debug.log ###

log4j.appender.D = org.apache.log4j.DailyRollingFileAppender

log4j.appender.D.File = D://logs/debug.log

log4j.appender.D.Append = true

log4j.appender.D.Threshold = DEBUG

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

### 输出ERROR 级别以上的日志到=D://logs/error.log ###

log4j.appender.E = org.apache.log4j.DailyRollingFileAppender

log4j.appender.E.File =D://logs/error.log

log4j.appender.E.Append = true

log4j.appender.E.Threshold = ERROR

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

log4j.appender.E.layout.ConversionPattern = %-d{yyyy-MM-dd HH:mm:ss} [ %t:%r ] - [ %p ] %m%n

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24

index.jsp内容乳如下：

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<html>

<body>

<h2>Hello World!</h2>

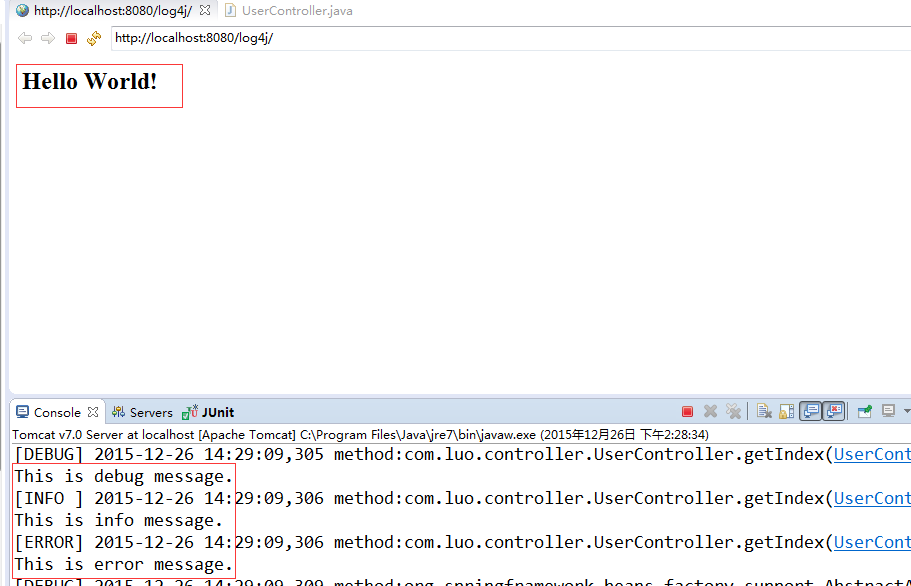
</body>

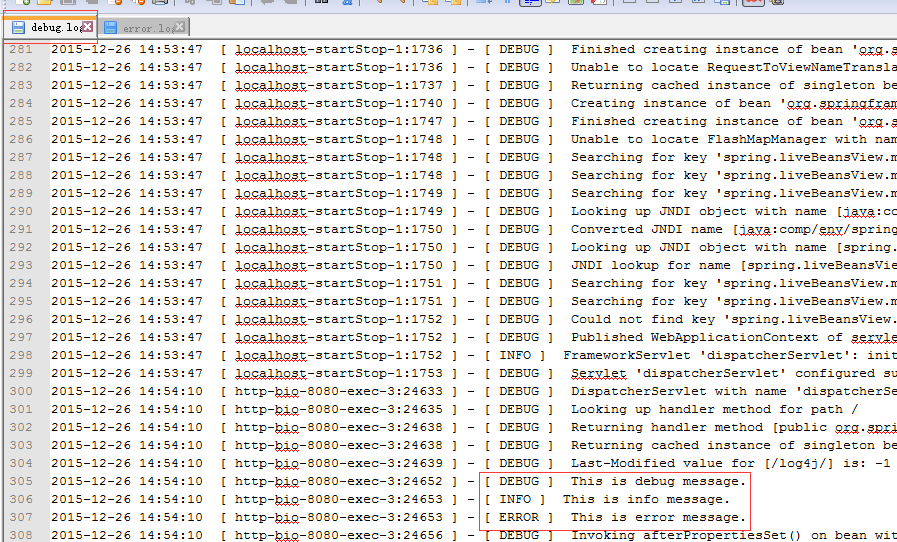
</html>

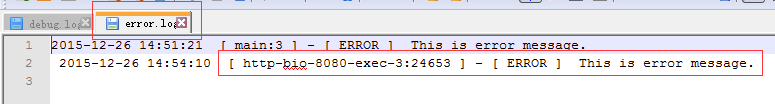
* 1
* 2
* 3
* 4
* 5
* 6

**4.5、运行结果**

用运行web项目log4j，访问：<http://localhost:8080/log4j/>，会看到eclipse的控制台和log文件的日志：







## 五、源码下载

<http://download.csdn.net/detail/u013142781/9378146>