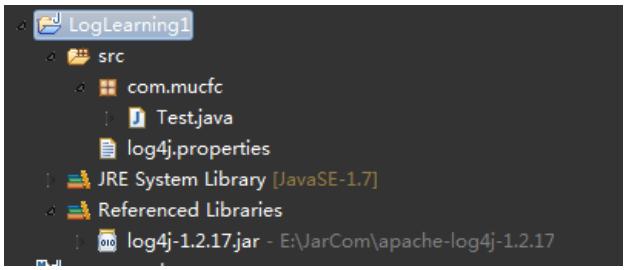
最详细的Log4j使用教程

日志是应用软件中不可缺少的部分，Apache的开源项目log4j是一个功能强大的日志组件,提供方便的日志记录。在apache网站：jakarta.apache.org/log4j 可以免费下载到Log4j最新版本的软件包。

一、入门实例

1.新建一个JAva工程，导入包log4j-1.2.17.jar，整个工程最终目录如下



2、src同级创建并设置log4j.properties

```
 ### 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=E://logs/error.log ###
log4j.appender.D = org.apache.log4j.DailyRollingFileAppender
log4j.appender.D.File = E://logs/log.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=E://logs/error.log ###
log4j.appender.E = org.apache.log4j.DailyRollingFileAppender
log4j.appender.E.File =E://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
```

3、设置日志内容

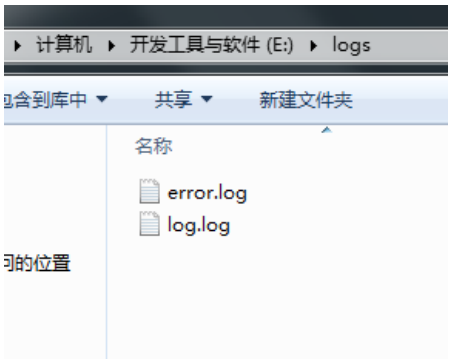
```
package com.mucfc;
import org.apache.log4j.Logger;
/**
 *@author linbingwen
 *@2015年5月18日9:14:21
 */
public class Test {
 private static Logger logger = Logger.getLogger(Test.class);

    /**
     * @param args
     */
    public static void main(String[] args) {
        // System.out.println("This is println message.");

        // 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.");
    }

}
```

4、输出结果

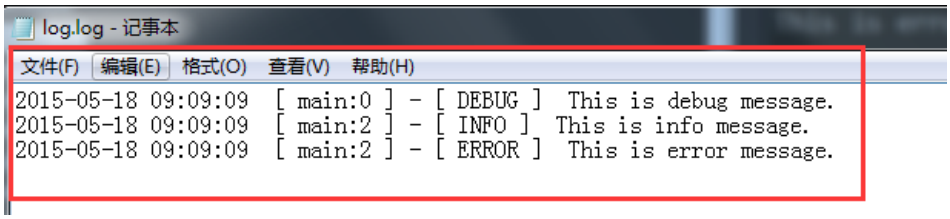
（1）首先是控制台的信息

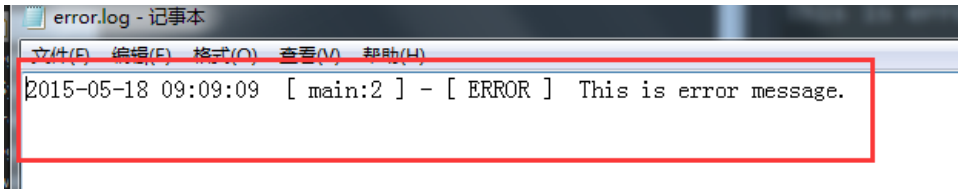
```
<terminated> Test (18) [Java Application] C:\Program Files\Java\jdk1.7.0_51\bin\javaw.exe (2015年5月18日 上午9:09:09)
[DEBUG] 2015-05-18 09:09:09,447 method:com.mucfc.Test.main(Test.java:16)
This is debug message.
[INFO ] 2015-05-18 09:09:09,449 method:com.mucfc.Test.main(Test.java:18)
This is info message.
[ERROR] 2015-05-18 09:09:09,449 method:com.mucfc.Test.main(Test.java:20)
This is error message.
```

（2）再来看输出的文件



内容如下，发现已按照要求输出到对应的文档中去了。





二、Log4j基本使用方法

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。

2.1、定义配置文件

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Java特性文件（键=值）。下面我们介绍使用Java特性文件做为配置文件的方法：
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B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）
```

3.配置日志信息的格式（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e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）
```

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，打印参数如下： %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“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)
```

2.2、在代码中使用Log4j

1.得到记录器

使用Log4j，第一步就是获取日志记录器，这个记录器将负责控制日志信息。其语法为：

public static Logger getLogger(String name)

通过指定的名字获得记录器，如果必要的话，则为这个名字创建一个新的记录器。Name一般取本类的名字，比如：

static Logger logger = Logger.getLogger (ServerWithLog4j.class.getName ())

2.读取配置文件

当获得了日志记录器之后，第二步将配置Log4j环境，其语法为：

```
BasicConfigurator.configure ()： 自动快速地使用缺省Log4j环境。
PropertyConfigurator.configure ( String configFilename) ：读取使用Java的特性文件编写的配置文件。
DOMConfigurator.configure ( String filename ) ：读取XML形式的配置文件。
```

3.插入记录信息（格式化日志信息）

当上两个必要步骤执行完毕，您就可以轻松地使用不同优先级别的日志记录语句插入到您想记录日志的任何地方，其语法如下：

```
Logger.debug ( Object message ) ;
Logger.info ( Object message ) ;
Logger.warn ( Object message ) ;
Logger.error ( Object message ) ;
```

2.3、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。

三、Web项目中使用Log4j实例

上面代码描述了Log4j的简单应用，其实使用Log4j也就是这样简单方便。当然除了上面的配置方法，还有其它，比如做一个J2EE应用，在J2EE应用使用Log4j，必须先在启动服务时加载Log4j的配置文件进行初始化，可以在web.xml中进行。

1、web应用的log4j使用基本上都采用：新建一个servlet，这个servlet在init函数中为log4j执行配置。一般就是读入配置文件。所以需要在web.xml中为这个servlet配置，同时设定load-on-startup为1。

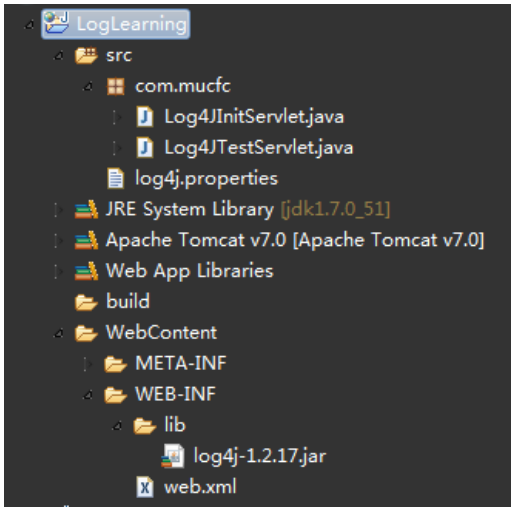
2、这个servlet配置log4j就是读出配置文件，然后调用configure函数。这里有两个问题：一、需要知道文件在哪里；二、需要正确的文件类型

3、配置文件位置在web.xml中配置一个param即可，路径一般是相对于web的root目录

4、文件类型一般有两种，一个是Java的property文件，另一种是xml文件

配置文件的大致内容：log4j可以指定输出的log级别的最低等级，以及log的输出配置格式，每个log可以指定多个输出方式

（1）创建Web工程，整个工程最后目录如下



（2）web.xml配置如下：

```
<?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"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.x
sd"
    id="WebApp_ID" version="3.0">
    <display-name>LogLearning</display-name>

    <servlet>
        <servlet-name>Log4JTestServlet</servlet-name>
        <servlet-class>com.mucfc.Log4JTestServlet</servlet-class>
    </servlet>

    <!--用来启动 log4jConfigLocation的servlet -->
    <servlet>
        <servlet-name>Log4JInitServlet</servlet-name>
        <servlet-class>com.mucfc.Log4JInitServlet</servlet-class>
        <init-param>
            <param-name>log4j-properties-location</param-name>
            <param-value>/WEB-INF/classes/log4j.properties</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>Log4JTestServlet</servlet-name>
        <url-pattern>/test</url-pattern>
    </servlet-mapping>

</web-app>
```

（3）配置文件log4j.properties

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

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
```

```
log4j.appender.D = org.apache.log4j.DailyRollingFileAppender
log4j.appender.D.File = F://logs/log.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

log4j.appender.E = org.apache.log4j.DailyRollingFileAppender
log4j.appender.E.File =F://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
```

（4）web容器一来就初始化的servlet

Log4JInitServlet.java

```
package com.mucfc;

import java.io.File;
import java.io.IOException;

import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.log4j.BasicConfigurator;
import org.apache.log4j.PropertyConfigurator;

/**
 * Servlet implementation class Log4JInitServlet
 */
@WebServlet("/Log4JInitServlet")
public class Log4JInitServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Log4JInitServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see Servlet#init(ServletConfig)
     */
    public void init(ServletConfig config) throws ServletException {
        System.out.println("Log4JInitServlet 正在初始化 log4j日志设置信息");
        String log4jLocation = config.getInitParameter("log4j-properties-location");

        ServletContext sc = config.getServletContext();

        if (log4jLocation == null) {
            System.err.println("*** 没有 log4j-properties-location 初始化的文件, 所以使用 BasicConfigurator
初始化");
            BasicConfigurator.configure();
        } else {
            String webAppPath = sc.getRealPath("/");
            String log4jProp = webAppPath + log4jLocation;
            File yoMamaYesThisSaysYoMama = new File(log4jProp);
            if (yoMamaYesThisSaysYoMama.exists()) {
                System.out.println("使用: " + log4jProp+"初始化日志设置信息");
                PropertyConfigurator.configure(log4jProp);
            } else {
                System.err.println("*** " + log4jProp + " 文件没有找到， 所以使用 BasicConfigurator初始化");

                BasicConfigurator.configure();
            }
        }
        super.init(config);
    }

    /**
```

```
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletExceptio
n, IOException {
        // TODO Auto-generated method stub
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletExcepti
on, IOException {
        // TODO Auto-generated method stub
    }

}
```

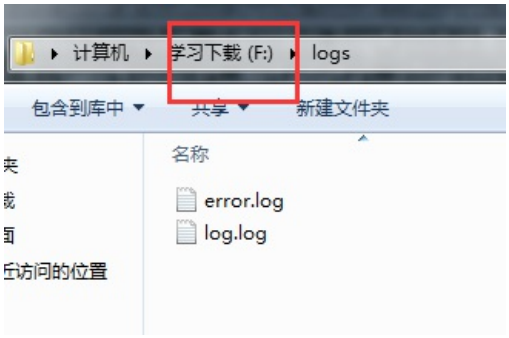
调用日志Log4JTestServlet.java

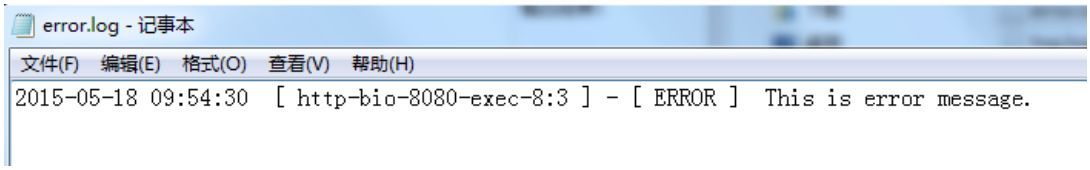
```
package com.mucfc;

import java.io.IOException;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.log4j.Logger;

/**
 * Servlet implementation class Log4JTestServlet
 */
@WebServlet("/Log4JTestServlet")
public class Log4JTestServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static Logger logger = Logger.getLogger(Log4JTestServlet.class);

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Log4JTestServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see Servlet#init(ServletConfig)
     */
    public void init(ServletConfig config) throws ServletException {
        // TODO Auto-generated method stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletExceptio
n, IOException {
        // 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.");
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletExcepti
on, IOException {
        doGet(request,response);
    }

}
```

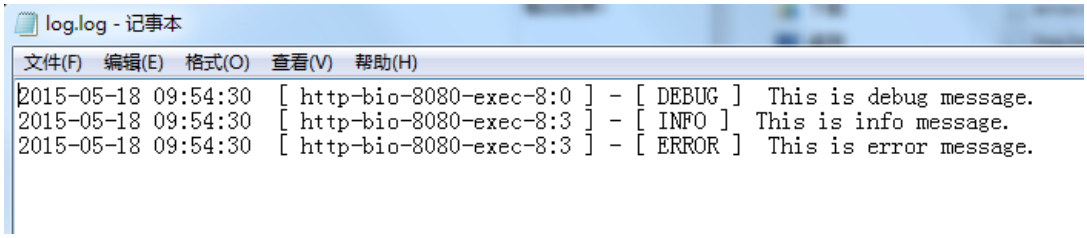
接下来就是运行了，来看看结果：

```
五月18, 2015 10:04:11 上午 org.apache.catalina.core.StandardEngine startInternal
信息: Starting Servlet Engine: Apache Tomcat/7.0.59
Log4JInitServlet 正在初始化 log4j日志设置信息
使用: E:\workspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\LogLearning\/WEB-INF/classes/log4j.prop
五月18, 2015 10:04:11 上午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["http-bio-8080"]
五月18, 2015 10:04:11 上午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["ajp-bio-8009"]
五月18, 2015 10:04:11 上午 org.apache.catalina.startup.Catalina start
信息: Server startup in 454 ms
[DEBUG] 2015-05-18 10:04:19,447 method:com.mucfc.Log4JTestServlet.doGet(Log4JTestServlet.java:42)
This is debug message.
[INFO ] 2015-05-18 10:04:19,457 method:com.mucfc.Log4JTestServlet.doGet(Log4JTestServlet.java:44)
This is info message.
[ERROR] 2015-05-18 10:04:19,457 method:com.mucfc.Log4JTestServlet.doGet(Log4JTestServlet.java:46)
This is error message.
```

输出结果：

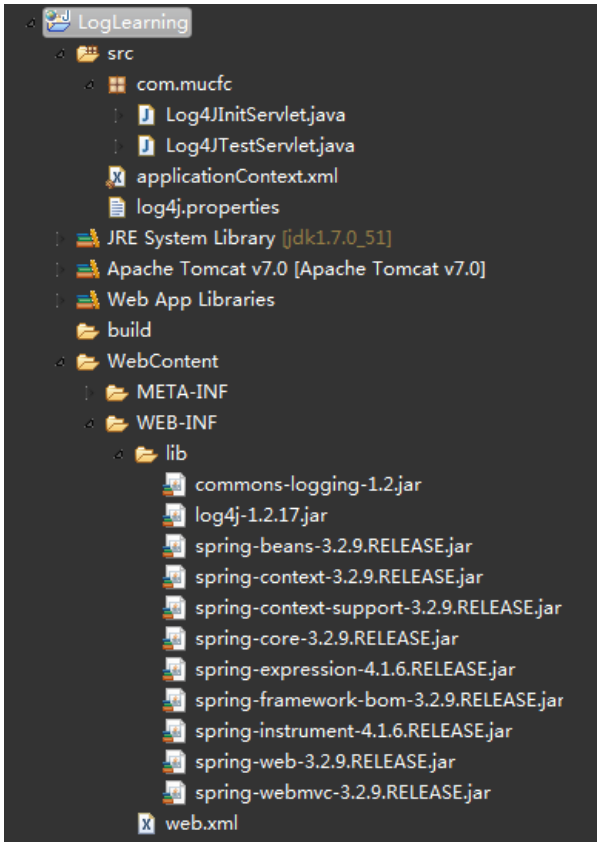






四、Spring中使用Log4j

这里要实现web项目中利用Spring来使用Log4j



（1）接上面的工程，然后再导入Spring的包

（2）web.xml增加

```
<!-- 设置根目录 -->
   <context-param>
       <param-name>webAppRootKey</param-name>
       <param-value>webapp.root</param-value>
   </context-param>

   <context-param>
    <param-name>log4jConfigLocation</param-name>
    <param-value>/WEB-INF/classes/log4j.properties</param-value>
</context-param>
<!-- 3000表示 开一条watchdog线程每60秒扫描一下配置文件的变化;这样便于日志存放位置的改变 -->
<context-param>
        <param-name>log4jRefreshInterval</param-name>
        <param-value>3000</param-value>
   </context-param>
<listener>
    <listener-class>org.springframework.web.util.Log4jConfigListener</listener-class>
</listener>
```

整个内容如下：

```
<?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"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.x
sd"
    id="WebApp_ID" version="3.0">
    <display-name>LogLearning</display-name>

    <servlet>
        <servlet-name>Log4JTestServlet</servlet-name>
        <servlet-class>com.mucfc.Log4JTestServlet</servlet-class>
    </servlet>

    <!--用来启动 log4jConfigLocation的servlet -->
<!--     <servlet>
        <servlet-name>Log4JInitServlet</servlet-name>
        <servlet-class>com.mucfc.Log4JInitServlet</servlet-class>
        <init-param>
            <param-name>log4j-properties-location</param-name>
            <param-value>/WEB-INF/classes/log4j.properties</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
```

```
    </servlet>-->

    <servlet-mapping>
        <servlet-name>Log4JTestServlet</servlet-name>
        <url-pattern>/test</url-pattern>
    </servlet-mapping>

        <!-- Spring 容器加载 -->
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext.xml</param-value>
    </context-param>

    <!-- 设置根目录 -->
    <context-param>
        <param-name>webAppRootKey</param-name>
        <param-value>webapp.root</param-value>
    </context-param>

    <context-param>
        <param-name>log4jConfigLocation</param-name>
        <param-value>/WEB-INF/classes/log4j.properties</param-value>
    </context-param>
    <!-- 3000表示 开一条watchdog线程每60秒扫描一下配置文件的变化;这样便于日志存放位置的改变 -->
    <context-param>
         <param-name>log4jRefreshInterval</param-name>
         <param-value>3000</param-value>
    </context-param>
    <listener>
        <listener-class>org.springframework.web.util.Log4jConfigListener</listener-class>
    </listener>

</web-app>
```

这里Log4JInitServlet.java就相当于没用到了。

（2）applicationContext.xml

没有内容：

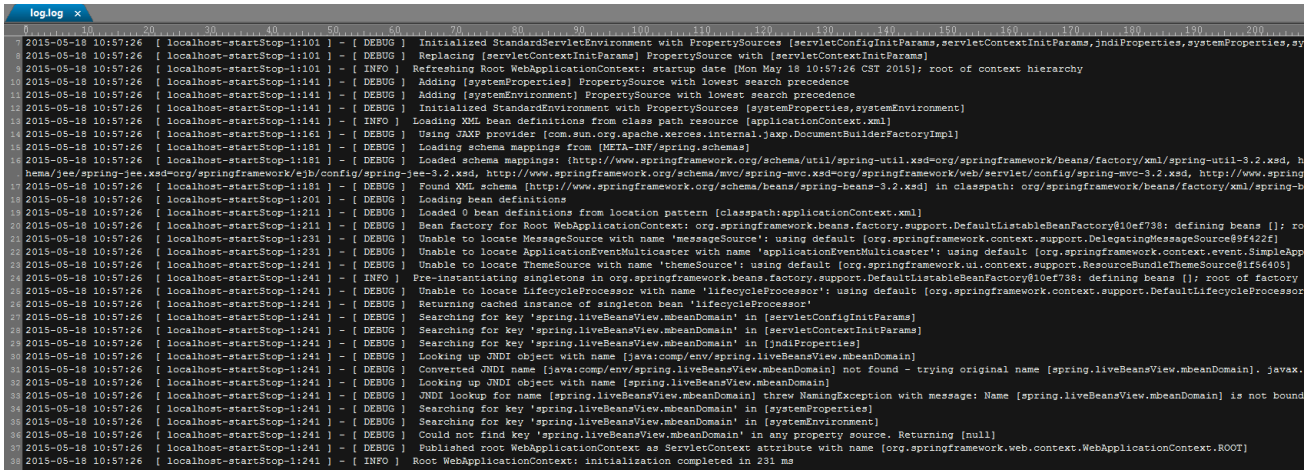
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/s
chema/context"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="

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


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


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


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


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

           http://www.springframework.org/schema/context/spring-context-3.2.xsd">
</beans>
```

（3）这样日志就跟随Spring窗口启动而启动了

程序一运行，就会自动把日志打印

```
[DEBUG] 2015-05-18 10:57:26,881 method:org.springframework.jndi.JndiTemplate.lookup(JndiTemplate.java:150)
Looking up JNDI object with name [java:comp/env/spring.liveBeansView.mbeanDomain]
[DEBUG] 2015-05-18 10:57:26,881 method:org.springframework.jndi.JndiLocatorSupport.lookup(JndiLocatorSupport.java:101)
Converted JNDI name [java:comp/env/spring.liveBeansView.mbeanDomain] not found - trying original name [spring.liveBeansV
[DEBUG] 2015-05-18 10:57:26,881 method:org.springframework.jndi.JndiTemplate.lookup(JndiTemplate.java:150)
Looking up JNDI object with name [spring.liveBeansView.mbeanDomain]
[DEBUG] 2015-05-18 10:57:26,881 method:org.springframework.jndi.JndiPropertySource.getProperty(JndiPropertySource.java:8
JNDI lookup for name [spring.liveBeansView.mbeanDomain] threw NamingException with message: Name [spring.liveBeansView.m
[DEBUG] 2015-05-18 10:57:26,881 method:org.springframework.core.env.PropertySourcesPropertyResolver.getProperty(Property
Searching for key 'spring.liveBeansView.mbeanDomain' in [systemProperties]
[DEBUG] 2015-05-18 10:57:26,881 method:org.springframework.core.env.PropertySourcesPropertyResolver.getProperty(Property
Searching for key 'spring.liveBeansView.mbeanDomain' in [systemEnvironment]
[DEBUG] 2015-05-18 10:57:26,881 method:org.springframework.core.env.PropertySourcesPropertyResolver.getProperty(Property
Could not find key 'spring.liveBeansView.mbeanDomain' in any property source. Returning [null]
[DEBUG] 2015-05-18 10:57:26,881 method:org.springframework.web.context.ContextLoader.initWebApplicationContext(ContextLo
Published root WebApplicationContext as ServletContext attribute with name [org.springframework.web.context.WebApplicati
[INFO ] 2015-05-18 10:57:26,881 method:org.springframework.web.context.ContextLoader.initWebApplicationContext(ContextLo
Root WebApplicationContext: initialization completed in 231 ms
五月18, 2015 10:57:26 上午 org.apache.catalina.core.ApplicationContext log
信息: Set web app root system property: 'webapp.root' = [E:\workspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp0
五月18, 2015 10:57:26 上午 org.apache.catalina.core.ApplicationContext log
信息: Initializing log4j from [E:\workspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\LogLearning\WE
五月18, 2015 10:57:26 上午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["http-bio-8080"]
五月18, 2015 10:57:26 上午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["ajp-bio-8009"]
五月18, 2015 10:57:26 上午 org.apache.catalina.startup.Catalina start
信息: Server startup in 1118 ms
```

log.log

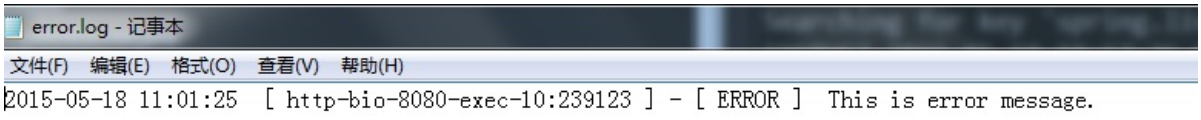


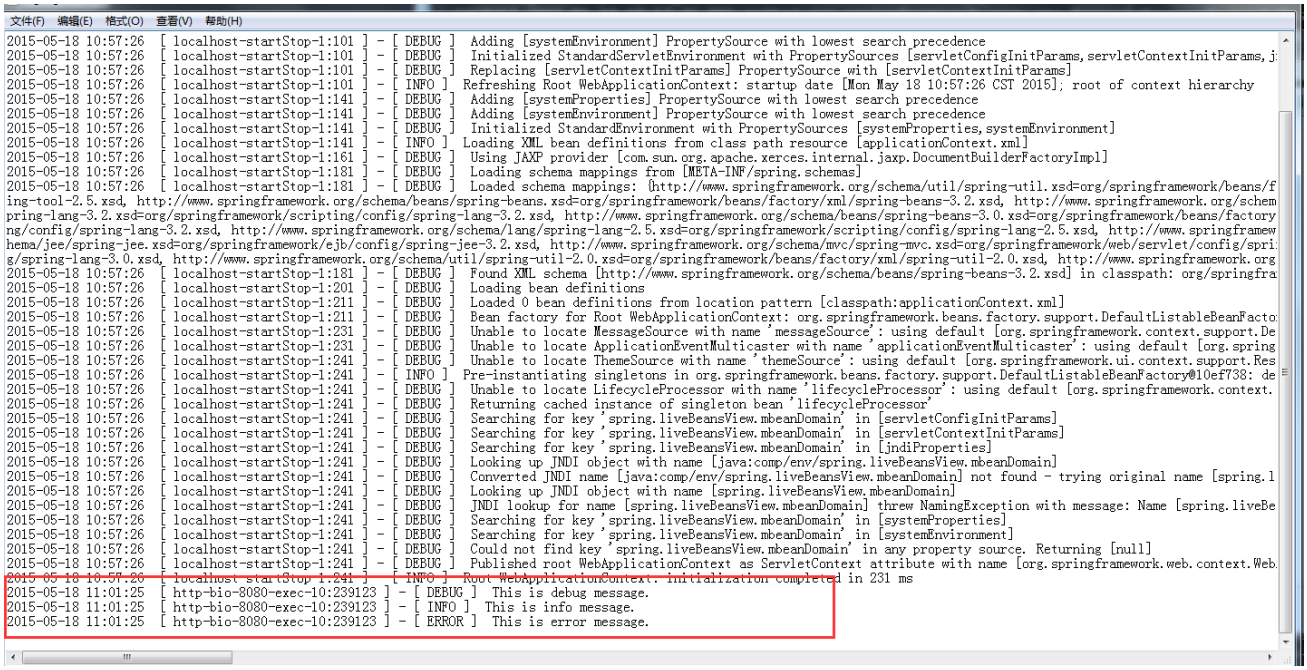
error.log为空，因为它只打印error级别以上的信息

浏览器输入http://localhost:8080/LogLearning2/test

```
信息: Starting ProtocolHandler ["http-bio-8080"]
五月18, 2015 10:57:26 上午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["ajp-bio-8009"]
五月18, 2015 10:57:26 上午 org.apache.catalina.startup.Catalina start
信息: Server startup in 1118 ms
[DEBUG] 2015-05-18 11:01:25,763 method:com.mucfc.Log4JTestServlet.doGet(Log4JTestServlet.java:42)
This is debug message.
[INFO ] 2015-05-18 11:01:25,763 method:com.mucfc.Log4JTestServlet.doGet(Log4JTestServlet.java:44)
This is info message.
[ERROR] 2015-05-18 11:01:25,763 method:com.mucfc.Log4JTestServlet.doGet(Log4JTestServlet.java:46)
This is error message.
```

然后打开文件




文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
2015-05-18 10:57:26 [localhost-startStop-1:101] - [DEBUG] Adding [systemEnvironment] PropertySource with lowest search precedence
2015-05-18 10:57:26 [localhost-startStop-1:101] - [DEBUG] Initialized StandardServletEnvironment with PropertySources [servletConfigInitParams,servletContextInitParams,j
2015-05-18 10:57:26 [localhost-startStop-1:101] - [DEBUG] Replacing [servletContextInitParams] PropertySource with [servletContextInitParams]
2015-05-18 10:57:26 [localhost-startStop-1:101] - [INFO] Refreshing Root WebApplicationContext: startup date [Mon May 18 10:57:26 CST 2015]; root of context hierarchy
2015-05-18 10:57:26 [localhost-startStop-1:141] - [DEBUG] Adding [systemProperties] PropertySource with lowest search precedence
2015-05-18 10:57:26 [localhost-startStop-1:141] - [DEBUG] Adding [systemEnvironment] PropertySource with lowest search precedence
2015-05-18 10:57:26 [localhost-startStop-1:141] - [DEBUG] Initialized StandardEnvironment with PropertySources [systemProperties,systemEnvironment]
2015-05-18 10:57:26 [localhost-startStop-1:141] - [INFO] Loading XML bean definitions from class path resource [applicationContext.xml]
2015-05-18 10:57:26 [localhost-startStop-1:161] - [DEBUG] Using JAXP provider [com.sun.org.apache.xerces.internal.jaxp.DocumentBuilderFactoryImpl]
2015-05-18 10:57:26 [localhost-startStop-1:181] - [DEBUG] Loading schema mappings from [META-INF/spring.schemas]
2015-05-18 10:57:26 [localhost-startStop-1:181] - [DEBUG] Loaded schema mappings: {http://www.springframework.org/schema/util/spring-util.xsd=org/springframework/beans/f
ing-tool-2.5.xsd, http://www.springframework.org/schema/beans/spring-beans.xsd=org/springframework/beans/factory/xml/spring-beans-3.2.xsd, http://www.springframework.org/schem
pring-lang-3.2.xsd=org/springframework/scripting/config/spring-lang-3.2.xsd, http://www.springframework.org/schema/beans/spring-beans-3.0.xsd=org/springframework/beans/factory
ng/config/spring-lang-3.2.xsd, http://www.springframework.org/schema/lang/spring-lang-2.5.xsd=org/springframework/scripting/config/spring-lang-2.5.xsd, http://www.springframew
hema/jee/spring-jee.xsd=org/springframework/ejb/config/spring-jee-3.2.xsd, http://www.springframework.org/schema/mvc/spring-mvc.xsd=org/springframework/web/servlet/config/spri
g/spring-lang-3.0.xsd, http://www.springframework.org/schema/util/spring-util-2.0.xsd=org/springframework/beans/factory/xml/spring-util-2.0.xsd, http://www.springframework.org
2015-05-18 10:57:26 [localhost-startStop-1:181] - [DEBUG] Found XML schema [http://www.springframework.org/schema/beans/spring-beans-3.2.xsd] in classpath: org/springfra
2015-05-18 10:57:26 [localhost-startStop-1:201] - [DEBUG] Loading bean definitions
2015-05-18 10:57:26 [localhost-startStop-1:211] - [DEBUG] Loaded 0 bean definitions from location pattern [classpath:applicationContext.xml]
2015-05-18 10:57:26 [localhost-startStop-1:211] - [DEBUG] Bean factory for Root WebApplicationContext: org.springframework.beans.factory.support.DefaultListableBeanFacto
2015-05-18 10:57:26 [localhost-startStop-1:231] - [DEBUG] Unable to locate MessageSource with name 'messageSource': using default [org.springframework.context.support.De
2015-05-18 10:57:26 [localhost-startStop-1:231] - [DEBUG] Unable to locate ApplicationEventMulticaster with name 'applicationEventMulticaster': using default [org.spring
2015-05-18 10:57:26 [localhost-startStop-1:241] - [DEBUG] Unable to locate ThemeSource with name 'themeSource': using default [org.springframework.ui.context.support.Res
2015-05-18 10:57:26 [localhost-startStop-1:241] - [INFO] Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultListableBeanFactory@10ef738: de
2015-05-18 10:57:26 [localhost-startStop-1:241] - [DEBUG] Unable to locate LifecycleProcessor with name 'lifecycleProcessor': using default [org.springframework.context.
2015-05-18 10:57:26 [localhost-startStop-1:241] - [DEBUG] Returning cached instance of singleton bean 'lifecycleProcessor'
2015-05-18 10:57:26 [localhost-startStop-1:241] - [DEBUG] Searching for key 'spring.liveBeansView.mbeanDomain' in [servletConfigInitParams]
2015-05-18 10:57:26 [localhost-startStop-1:241] - [DEBUG] Searching for key 'spring.liveBeansView.mbeanDomain' in [servletContextInitParams]
2015-05-18 10:57:26 [localhost-startStop-1:241] - [DEBUG] Searching for key 'spring.liveBeansView.mbeanDomain' in [jndiProperties]
2015-05-18 10:57:26 [localhost-startStop-1:241] - [DEBUG] Looking up JNDI object with name [java:comp/env/spring.liveBeansView.mbeanDomain]
2015-05-18 10:57:26 [localhost-startStop-1:241] - [DEBUG] Converted JNDI name [java:comp/env/spring.liveBeansView.mbeanDomain] not found - trying original name [spring.l
2015-05-18 10:57:26 [localhost-startStop-1:241] - [DEBUG] Looking up JNDI object with name [spring.liveBeansView.mbeanDomain]
2015-05-18 10:57:26 [localhost-startStop-1:241] - [DEBUG] JNDI lookup for name [spring.liveBeansView.mbeanDomain] threw NamingException with message: Name [spring.liveBe
2015-05-18 10:57:26 [localhost-startStop-1:241] - [DEBUG] Searching for key 'spring.liveBeansView.mbeanDomain' in [systemProperties]
2015-05-18 10:57:26 [localhost-startStop-1:241] - [DEBUG] Searching for key 'spring.liveBeansView.mbeanDomain' in [systemEnvironment]
2015-05-18 10:57:26 [localhost-startStop-1:241] - [DEBUG] Could not find key 'spring.liveBeansView.mbeanDomain' in any property source. Returning [null]
2015-05-18 10:57:26 [localhost-startStop-1:241] - [DEBUG] Published root WebApplicationContext as ServletContext attribute with name [org.springframework.web.context.Web
2015-05-18 10:57:26 [localhost-startStop-1:241] - [INFO] Root WebApplicationContext: initialization completed in 231 ms
2015-05-18 11:01:25 [http-bio-8080-exec-10:239123] - [DEBUG] This is debug message.
2015-05-18 11:01:25 [http-bio-8080-exec-10:239123] - [INFO] This is info message.
2015-05-18 11:01:25 [http-bio-8080-exec-10:239123] - [ERROR] This is error message.