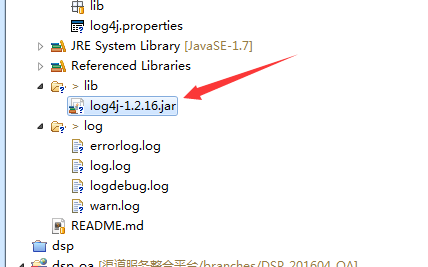
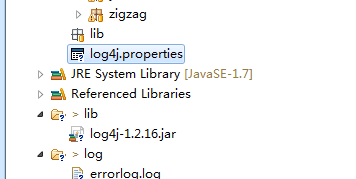
# 1、添加jar包



# 2、src根目录下建立 log4j.properties文件



# 3、log4j.properties 中写入内容

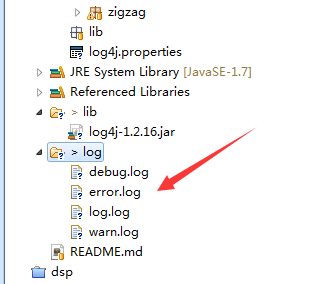
## 1、必填项

|  |
| --- |
| ## 1 必填内容，info/all/.., stdout 为必填，后面的根据log4j.appender.内容进行填写，如果下面有内容则，这里必须加上，all这里的位置为权限级别，level 是日志记录的优先级，分为OFF、FATAL、ERROR、WARN、INFO、DEBUG、ALL或者自定义的级别。RollingFileAppender按log文件最大长度限度生成新文件，DailyRollingFileAppender按日期生成新文件  log4j.rootLogger=all, stdout, log, errorlog ,ROLLING\_FILE,warnlog  ## 2 必填内容，控制台打印 不需要改动  ###Console print infomation to console ,--> must exist###  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} [ %t ] [ %p ]:%L - %m%n  log4j.appender.stdout.layout.ConversionPattern = %d{yyyy-MM-dd HH:mm:ss}[%x][%t] %p [%c] %m%n |

## 2、根据等级设置相应的log日志文件

|  |
| --- |
| ## 3  ### Log ###  log4j.appender.log = org.apache.log4j.DailyRollingFileAppender  log4j.appender.log.File = log/log.log  log4j.appender.log.Append = true  log4j.appender.log.Threshold = INFO  log4j.appender.log.DatePattern='.'yyyy-MM-dd  log4j.appender.log.layout = org.apache.log4j.PatternLayout  log4j.appender.log.layout.ConversionPattern = %d{yyyy-MM-dd HH:mm:ss}[%x][%t] %p [%c] %m%n  ## 4  ### Error ###  log4j.appender.errorlog = org.apache.log4j.DailyRollingFileAppender  log4j.appender.errorlog.File = log/errorlog.log  log4j.appender.errorlog.Append = true  log4j.appender.errorlog.File = log/error.log  log4j.appender.errorlog.DatePattern='.'yyyy-MM-dd  log4j.appender.errorlog.layout = org.apache.log4j.PatternLayout  log4j.appender.errorlog.layout.ConversionPattern =%d{yyyy-MM-dd HH:mm:ss}[%x][%t] %p [%c] %m%n    ## 5  ### debug \u63A7\u5236\u53F0\u4FE1\u606F###  log4j.appender.ROLLING\_FILE=org.apache.log4j.RollingFileAppender  log4j.appender.ROLLING\_FILE.File=log/debug.log  log4j.appender.ROLLING\_FILE.Append=true  log4j.appender.ROLLING\_FILE.Threshold=debug  log4j.appender.ROLLING\_FILE.MaxFileSize=2000KB  log4j.appender.ROLLING\_FILE.MaxBackupIndex=200  log4j.appender.ROLLING\_FILE.layout=org.apache.log4j.PatternLayout  log4j.appender.ROLLING\_FILE.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss}[%x][%t] %p [%c] %m%n  ## 6  ### warn \u63A7\u5236\u53F0\u4FE1\u606F###  warn=\u63A7\u5236\u53F0\u4FE1\u606F  log4j.appender.warnlog=org.apache.log4j.RollingFileAppender  log4j.appender.warnlog.File=log/warn.log  log4j.appender.warnlog.Append=true  log4j.appender.warnlog.Threshold=WARN  log4j.appender.warnlog.MaxFileSize=200KB  log4j.appender.warnlog.MaxBackupIndex=200  log4j.appender.warnlog.layout=org.apache.log4j.PatternLayout  log4j.appender.warnlog.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss}[%x][%t] %p [%c] %m%n |

## 3、日志文件位置，项目根目录下



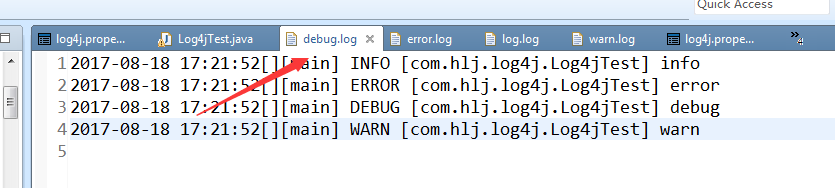
# 4、测试java文件

|  |
| --- |
| **public** **class** Log4jTest {  **public** **static** **void** main(String[] args) {  Logger logger = Logger.*getLogger*(Log4jTest.**class**);  **try** {  **int** i = 1/0;  } **catch** (Exception e) {  e.printStackTrace();  }  System.*out*.println("testlog");  logger.info("info");  logger.error("error");  logger.debug("debug");  logger.warn("warn");  }  } |
|  |

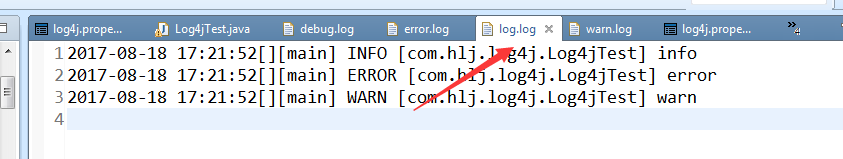
## 2、控制台

## 4、查看日志内容

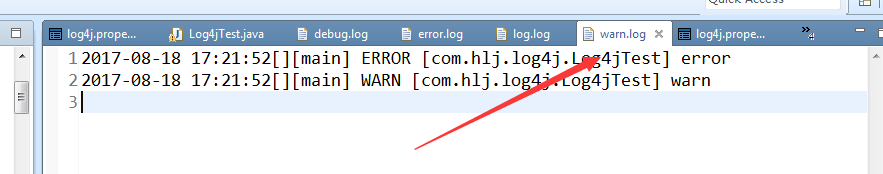
### 1、debug debug.log



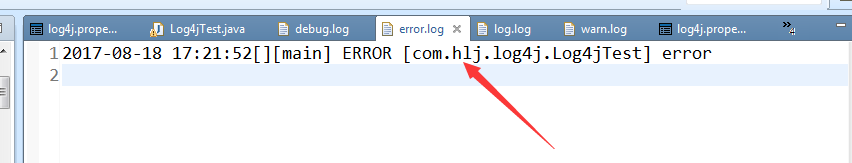
### 2、info log.log



### 3、warn warn.log



### 4、error error.log



# 5、分析上述4种

### 1、debug 级别最低，可以显示所有的

### 1、info 显示inf 和以下两个

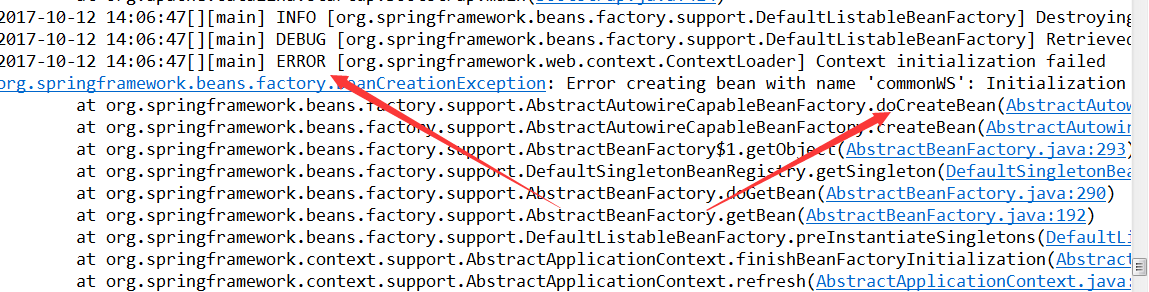
### 2、warn可以显示自己和error的

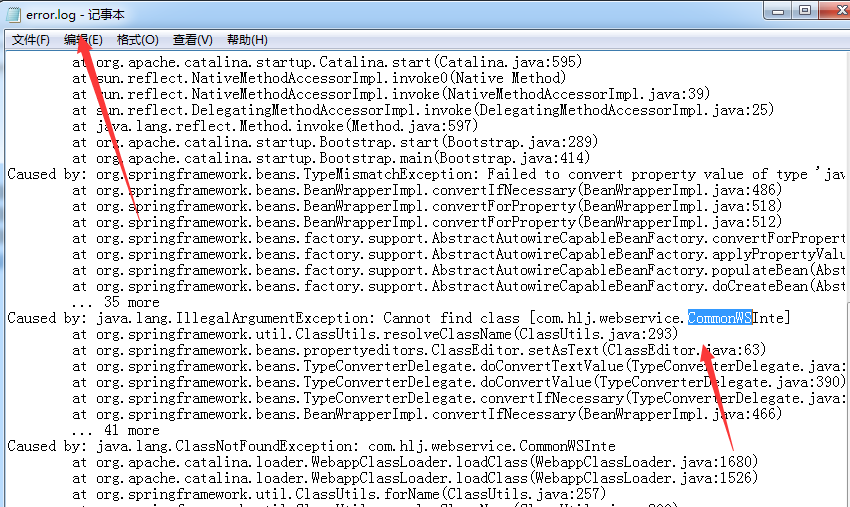
### 3、error 只能显示自己的

# 5、web项目中（ SpringWebService）

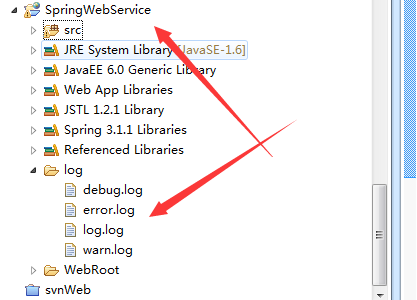
## 1、配置如上，本实例是采用xml方式启动，只要在classpath路径下（src目录或者resource目录下）添加了log4j.propetries文件。则其中的所写入的log日志文件以及，所在目录会自动帮我们创建（工程中和tomcat bin下面的日志都会帮我们创建）

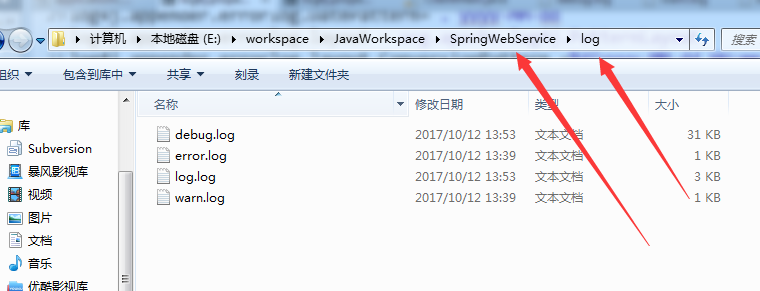
### 启动测试控制台如下



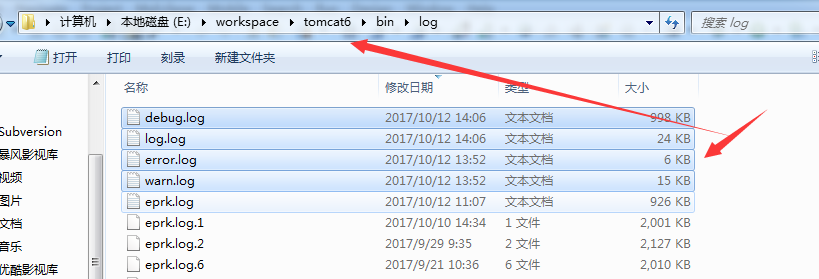


### 1、项目所在目录（惊奇的发现本地main控制台打印的都在src这里面，比如上面的testlog类中的打印）





### 2、tomcat bin目录（服务端的日志都在这里面）

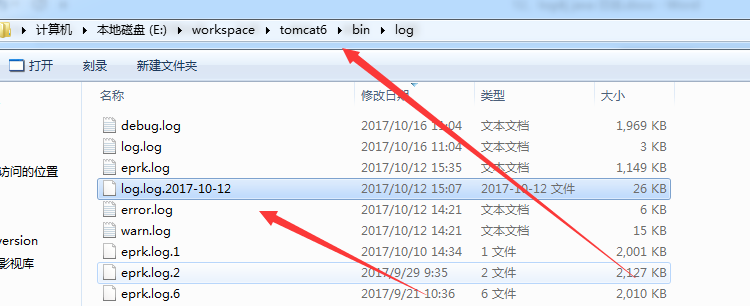


### 3、log4j系统会自动根据我们的需要根据日期进行重新创建日志文件

#### 1、src目录



#### 2、tomcat目录



# 6、根据级别显示

## 1、logger 是 category子类，现在category已经不提倡使用，儿子用法基本一样，所以以后我们尽量用logger。注意大小写，非常关键

## 2、二者其实说白了都是父亲rootLogger的儿子。有继承这个关系，也就是说如果父亲中是warn，子类中定位error，则对于它设置的包，比如com.hlj.log4j，只会打印error的日志。

## 3、logger和category。第二个参数为打印到那里去，比如打印到控制台，设置为下面的，stdout，如果和父类rootLogger也有，则表示重复了，则会在控制台中打印两变，但是其他的日志比如log，error，warn，debug则只会打印一遍，

## 4、当启用additivity（可添加性质）为false，false则表示，不可添加，则只能够给logger或者category设置的目的地对应的日志文件进行输出。其他的日志文件不再进行打印日志

|  |
| --- |
| log4j.rootLogger=warn, stdout, log, errorlog ,ROLLING\_FILE,warnlog  #log4j.Logger=search,Test  #log4j.category.com.hlj.log4j=error, stdout  log4j.logger.com.hlj.log4j=error, stdout  #log4j.additivity.com.hlj.log4j= false  ## 2  ###Console print infomation to console ,--> must exist###  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} [ %t ] [ %p ]:%L - %m%n  log4j.appender.stdout.layout.ConversionPattern = %d{yyyy-MM-dd HH:mm:ss}[%x][%t] %p [%c] %m%n |