Spring日志使用：---中等搞定；

SLF4J

Log4J

错误日志，必须有；

因为有些错误，需要到错误日志中，才能找到；

一般的：

写两个输出，一个输出在控制台，一个保存到硬盘；

/////////////////////////////////////////////////////////////////////////////////////////////////

Spring文档中有一点内容；---2.32节；

<http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/>

Jar，

jcl-over-slf4j 1.7.5 org.slf4j

slf4j-api 1.7.5 org.slf4j

slf4j-log4j12 1.7.5 org.slf4j

log4j 1.2.16 log4j

类中使用，

**import** org.apache.log4j.Logger;

**public** **class** Test1 {

**private** **static** **final** Logger *logger* = Logger.*getLogger*(Test1.**class**);

*logger*.info("日志运行");

//据日志的5种类型，DEBUG，[INFO](http://baike.baidu.com/view/105555.htm" \t "_blank)，WARN，[ERROR](http://baike.baidu.com/view/722452.htm" \t "_blank)，FATAL，(由低到高)；(加上OFF、ALL,共7个参数)

*logger*.info("日志运行");

*logger*.debug("调试");

*logger*.warn("警告");

*logger*.error("错误");

*logger*.fatal("致命");

log4j.properties：

//首个参数为日志类型，显示>=此级别的日志；

log4j.rootCategory=INFO, stdout,error

# console log //ConsoleAppender，控制台输出日志；

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

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

log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p %t %c**{2}**:%L - %m%n

# error log //DailyRollingFileAppender，硬盘保存日志；

log4j.appender.error=org.apache.log4j.DailyRollingFileAppender

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

log4j.appender.error.layout.ConversionPattern=%d %p [%c] - %m%n

log4j.appender.error.file=/DATA/logs/error.log

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

log4j.appender.error.Threshold=ERROR //日志级别>=error的；

log4j.appender.error.append=true

参数详情：

log4j.rootCategory=INFO, stdout , error ---然后一般stdout 输出日志到控制台，r输出日志到硬盘文件；

此句为将等级为INFO的日志信息输出到stdout和error这两个目的地，stdout和error的定义在下面的代码，可以任意起名。等级可分为OFF、FATAL、ERROR、WARN、INFO、[DEBUG](http://baike.baidu.com/view/45108.htm" \t "_blank)、ALL，如果配置OFF则不打出任何信息，如果配置为[INFO](http://baike.baidu.com/view/105555.htm" \t "_blank)这样只显示INFO, WARN, ERROR的[log](http://baike.baidu.com/view/426996.htm" \t "_blank)信息，而DEBUG信息不会被显示，

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

此句为定义名为[stdout](http://baike.baidu.com/view/1322475.htm" \t "_blank)的输出端是哪种类型，可以是

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）

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

此句为定义名为stdout的输出端的layout是哪种类型，可以是

org.apache.log4j.HTMLLayout（以[HTML](http://baike.baidu.com/view/692.htm" \t "_blank)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](http://baike.baidu.com/view/56072.htm" \t "_blank)），

org.apache.log4j.TTCCLayout（包含日志产生的时间、[线程](http://baike.baidu.com/view/1053.htm" \t "_blank)、类别等等信息）

log4j.appender.stdout.layout.ConversionPattern= [QC] %p [%t] %C.%M(%L) | %m%n

如果使用pattern布局就要指定的打印信息的具体格式ConversionPattern，打印参数如下：

%m 输出代码中指定的消息；

%M 输出打印该条日志的方法名；

%p 输出优先级，即DEBUG，[INFO](http://baike.baidu.com/view/105555.htm" \t "_blank)，WARN，[ERROR](http://baike.baidu.com/view/722452.htm" \t "_blank)，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{yyyy-MM-dd HH:mm:ss,SSS}，输出类似：2002-10-18 22:10:28,921；

%l 输出日志事件的发生位置，及在代码中的行数；

[QC]是log信息的开头，可以为任意字符，一般为项目简称。

输出的信息

[TS] DEBUG [main] AbstractBeanFactory.getBean(189) | Returning cached instance of singleton bean 'MyAutoProxy'

**1.ConsoleAppender选项 ---输出日志到控制台时使用；**                    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 ---输出日志到硬盘文件时使用；  
                    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。

////////////////////////////////////////////////////////////////////////////////////////////////

错误记录：

项目代码，异常处理的catch中，

logger.error("异常类型名，如：login/登录，表示登录相关的异常", e);

将error的错误信息保存在硬盘log日志文件中；

或，最简单的，

***logger***.error(e); | 意思是记录错误日志；

普通提示：

在正常处理的地方，可以加

*logger*.info("日志运行");

/////////////////////////////////////////////

添加Druid的日志：---在数据源的配置之后，

参考，

<https://github.com/alibaba/druid/wiki/%E9%85%8D%E7%BD%AE_LogFilter>

先，

#log4j.logger.druid.sql=warn,stdout #log4j.logger.druid.sql.DataSource=warn,stdout #log4j.logger.druid.sql.Connection=warn,stdout log4j.logger.druid.sql.Statement=warn,stdout #log4j.logger.druid.sql.ResultSet=warn,stdout

然后，

log4j.appender.druid=org.apache.log4j.DailyRollingFileAppender

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

log4j.appender.druid.layout.ConversionPattern=%d %p [%c] - %m%n

log4j.appender.druid.file=/DATA/logs/sql.log

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

log4j.appender.druid.append=true