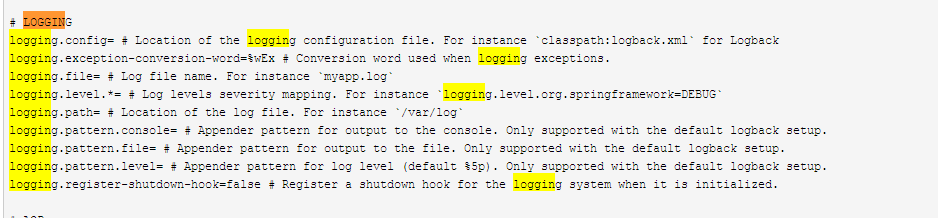
https://www.cnblogs.com/sxdcgaq8080/p/7852858.html

**[【spring boot】8.spring boot的日志框架logback使用](https://www.cnblogs.com/sxdcgaq8080/p/7852858.html)**

 在继续上一篇的Debug调试之后，把spring boot的日志框架使用情况逐步蚕食。

**参考：http://tengj.top/2017/04/05/springbo**

开篇之前，贴上完整application.properties日志相关配置



=============================================================================================

**简介：spring boot的默认日志框架Logback**

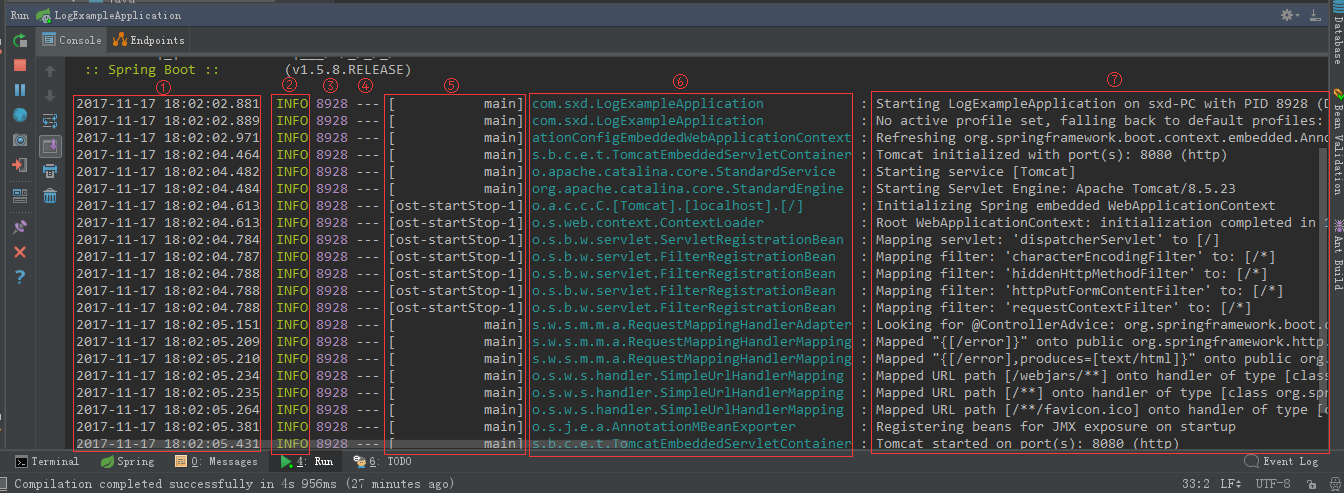
SLF4J——Simple Logging Facade For Java，它是一个针对于各类Java日志框架的统一Facade抽象。Java日志框架众多——常用的有java.util.logging, log4j, logback，commons-logging, Spring框架使用的是Jakarta Commons Logging API (JCL)。而SLF4J定义了统一的日志抽象接口，而真正的日志实现则是在运行时决定的——它提供了各类日志框架的binding。

Logback是log4j框架的作者开发的新一代日志框架，它效率更高、能够适应诸多的运行环境，同时天然支持SLF4J。

默认情况下，Spring Boot会用Logback来记录日志，并用INFO级别输出到控制台。在刚开始运行一个崭新的spring boot项目时，你就可以在控制台看到INFO级别的日志了。

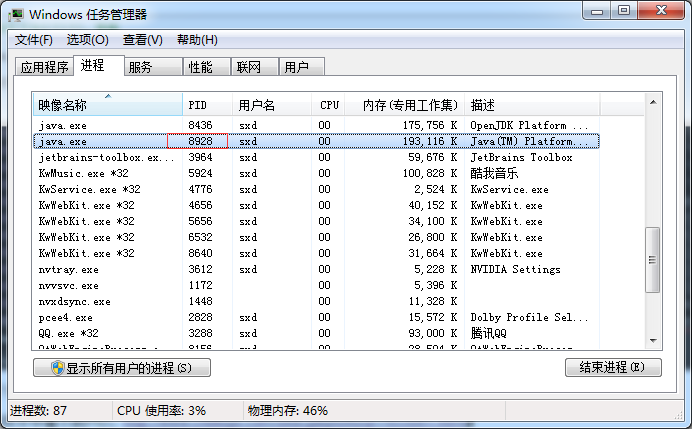
有如下的7各部分：

* ①时间日期：精确到毫秒
* ②日志级别：ERROR> WARN>INFO>DEBUG > TRACE　　优先级如前面，默认情况下是INFO，则INFO以及优先级更高的WARN和ERROR会被显示
* ③进程ID　　就是PID
* ④分隔符：--- 标识实际日志的开始
* ⑤线程名：方括号括起来（可能会截断控制台输出）
* ⑥Logger名：通常使用源代码的类名
* ⑦日志内容



可以从上面看到，这个spring boot是启动在8928这个进程上的。

在任务管理器中可以看到【如果任务管理的的进程选项卡中看不到PID，[http://www.cnblogs.com/sxdcgaq8080/p/7853081.html](http://www.cnblogs.com/sxdcgaq8080/p/7853081.html" \o "windows查看进程PID" \t "_blank)】



 ======================================================================================================

**一、在项目application.properties配置就可以使用log功能**

**1.要在spring boot中使用日志功能，在pom.xml中配置maven架包**

<!--使用spring boot日志框架-->

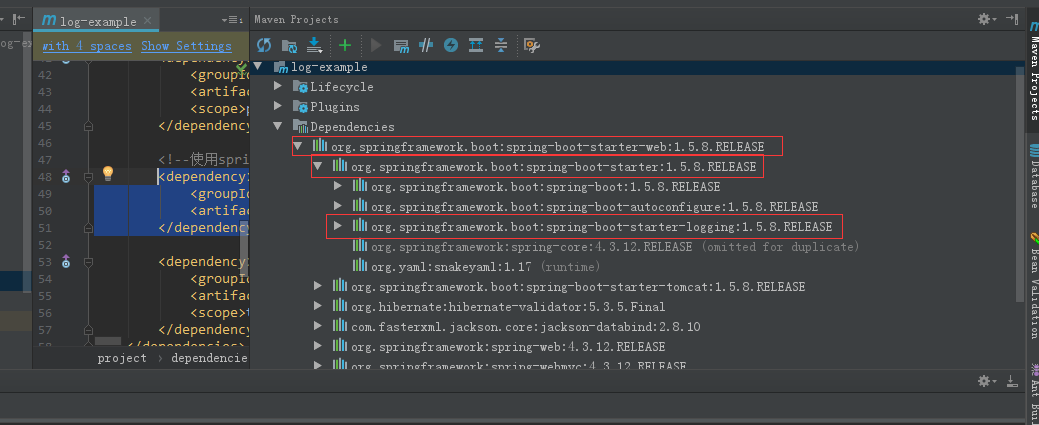
<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-logging</artifactId>

</dependency>

但是其实是不用配置的，因为在spring-boot-starter-web下已经引用了日志的架包，如下图：



**2.两种使用情况**

日志级别从低到高分为TRACE < DEBUG < INFO < WARN < ERROR < FATAL，如果设置为WARN，则低于WARN的信息都不会输出。

**2.1　控制台输出**

Spring Boot中默认配置ERROR、WARN和INFO级别的日志输出到控制台。您还可以通过启动您的应用程序--debug标志来启用“调试”模式（开发的时候推荐开启）,以下两种方式：

　　　　2.1.1在运行命令后加入--debug标志，如：$ java -jar springTest.jar --debug

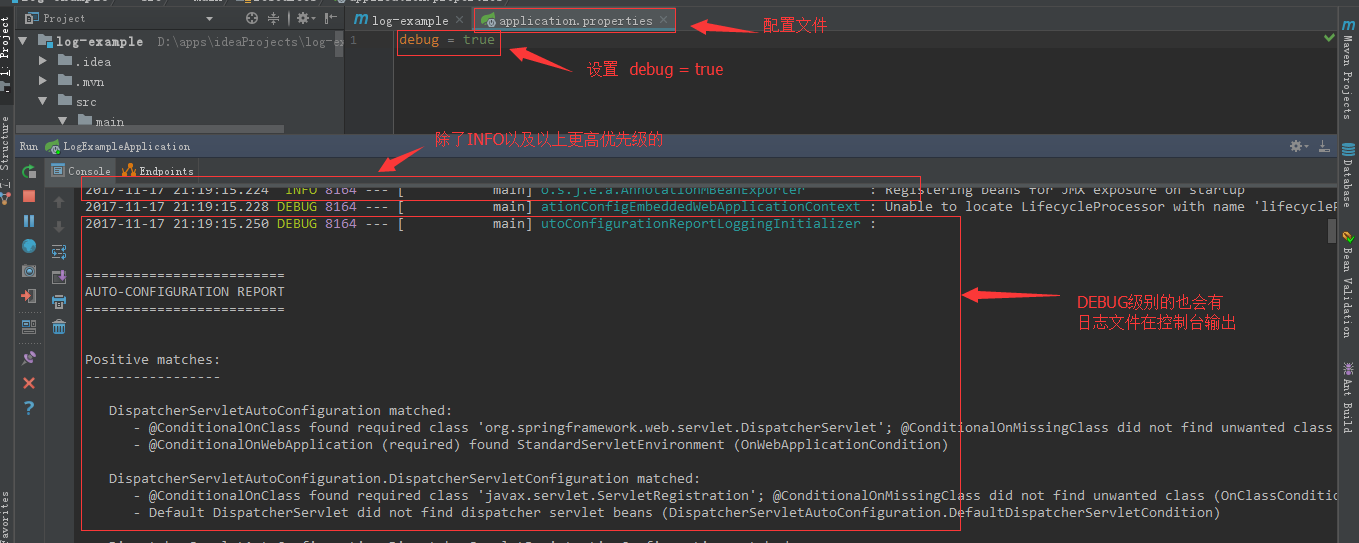
　　　　2.1.2在application.properties中配置debug=true，该属性置为true的时候，核心Logger（包含嵌入式容器、hibernate、spring）会输出更多内容，**但是你自己应用的日志并不会输出为DEBUG级别**。

如下：

配置文件application.properties中配置：

debug=true

启动，观察控制台打印出来的日志如下：



**2.2文件输出**

默认情况下，Spring Boot将日志输出到控制台，不会写到日志文件。如果要编写除控制台输出之外的日志文件，则需在application.properties中设置logging.file或logging.path属性。

* logging.file，设置文件，可以是绝对路径，也可以是相对路径。如：logging.file=my.log
* logging.path，设置目录，会在该目录下创建spring.log文件，并写入日志内容，如：logging.path=/var/log

如果只配置 logging.file，会在项目的当前路径下生成一个 xxx.log 日志文件。

如果只配置 logging.path，在 /var/log文件夹生成一个日志文件为 spring.log

**注：二者不能同时使用，如若同时使用，则只有logging.file生效**

**①默认情况下，日志文件的大小达到10MB时会切分一次，产生新的日志文件**

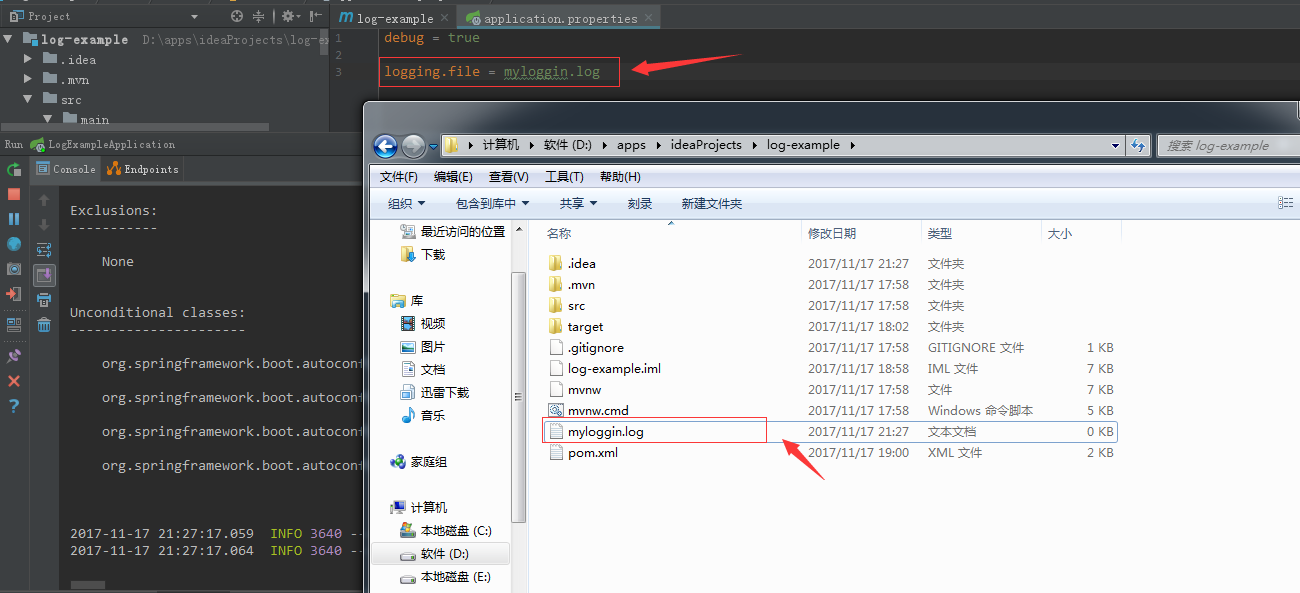
**②默认级别为：ERROR、WARN、INFO**

**2.2.1　　logging.file，设置文件**

**2.2.1.1 配置文件application.properties中配置【logging.file  相对路径】**

logging.file = myloggin.log

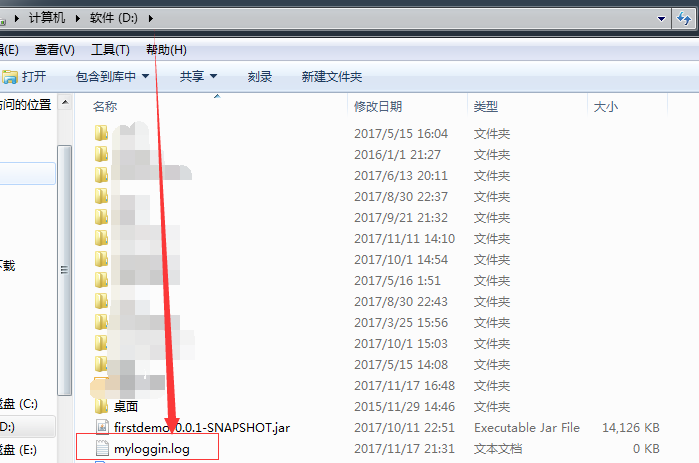
生成日志文件在项目【相当于项目根目录下的相对路径】



**2.2.1.2　在配置文件applicarion.properties中配置【logging.file  绝对路径】**

logging.file = d:/myloggin.log

生成日志文件，位置如下：

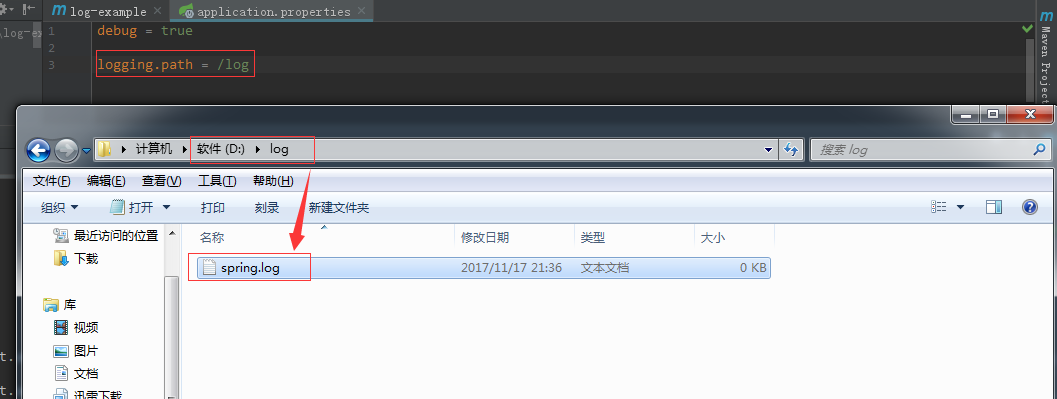


**2.2.2　logging.path，设置目录**

**2.2.2.1　在配置文件appilication.properties中设置【logging.path  相对路径】**

logging.path = /log

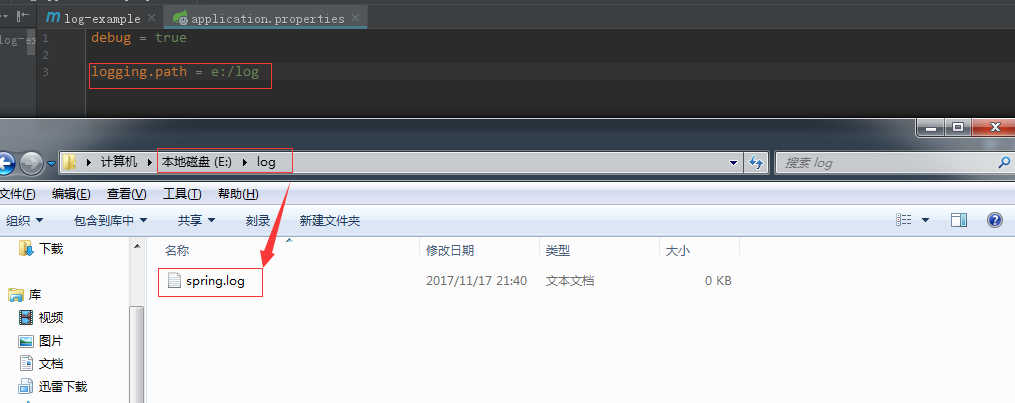
生成日志文件在当前项目所在的磁盘的根目录下生成一个log文件夹，在文件夹中生成spring.log日志文件【特定名字就是spring.log】



**2.2.2.2　在配置文件application.properties中配置【logging.path 绝对路径】**

logging.path = e:/log

会在指定目录的文件夹中生成spring.log日志文件【特定文件名spring.log】



 =========================================================================================================

**二、日志等级在application.properties的控制**

 根据开篇贴上的配置文件中的配置属性，可以看到在application.properties中配置日志等级就是一个配置，

格式为：logging.level.\* = LEVEL

* logging.level：日志级别控制前缀，\*为包名或Logger名
* LEVEL：选项TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF

举例：

* logging.level.com.sxd=DEBUG　　com.sxd包下所有class以DEBUG级别输出，包括其优先级以上
* logging.level.root=WARN　　　　root日志以WARN级别输出，包括其优先级以上

=====================================================================================================

**三、完全控制和解析日志配置**

**1.正经的提出问题：**

上面两步，可以看到日志可以在控制台或者文件中显示出来，但是都是统一的样式，统一的等级，或者只能配置一个路径。

如果想完全控制日志，

　　　　①例如：我想在控制台输出的日志格式是这样的，但是在文件中能不能以另外一种格式输出？

 　　　　②例如：我的日志文件输出，能不能自己控制文件的大小，文件名，日志文件总共多少就得清理？

　　　　③例如：能不能在不同的环境下，有不同的日志操作，比如在生产环境下，我的Service层的日志不会被打印，但是在开发环境下就要被打印出来

等等等等的，如果想完全控制日志的使用，往下看----->

**2.慢慢的分析问题：**

由于日志服务一般都在ApplicationContext创建前就初始化了，它并不是必须通过Spring的配置文件控制。那现在在application.properties中配置些许的属性已经不能完全满足了。

所以通过系统属性和传统的Spring Boot外部配置文件 才 可以很好的支持日志控制和管理。

**问题1：外部的日志配置文件应该叫什么名字呢？**

根据不同的日志系统，你可以按如下规则组织配置文件名，就能被正确加载：

* Logback：logback-spring.xml, logback-spring.groovy, logback.xml, logback.groovy
* Log4j：log4j-spring.properties, log4j-spring.xml, log4j.properties, log4j.xml
* Log4j2：log4j2-spring.xml, log4j2.xml
* JDK (Java Util Logging)：logging.properties

**注意：Spring Boot官方推荐优先使用带有-spring的文件名作为你的日志配置（如使用logback-spring.xml，而不是logback.xml），命名为logback-spring.xml的日志配置文件，spring boot可以为它添加一些spring boot特有的配置项【特有的就是spring boot的profile多环境配置】**

**问题2：日志配置文件应该放在哪里呢？**

放在src/main/resources下面即可。

**问题3：那不使用logback-spring.xml名字，使用自己定义的日志配置文件名行不行？**

如果你即想完全掌控日志配置，但又不想用logback.xml作为Logback配置的名字，可以通过logging.config属性指定自定义的名字：

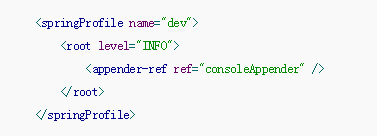
logging.config=classpath:logging-自定义.xml

**问题4：那我想使用spring boot的profile多环境配置，日志配置文件应该用什么名字？使用流程是怎么样？**

如果想使用profile多环境配置，

　　第一：日志配置文件名应该定义成logback-spring.xml。

　　第二：然后在日志配置文件中，使用<springProfile name="dev">  </springProfile>标签来包住想要在这种环境下使用的配置。使用如下截图：

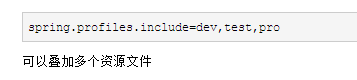






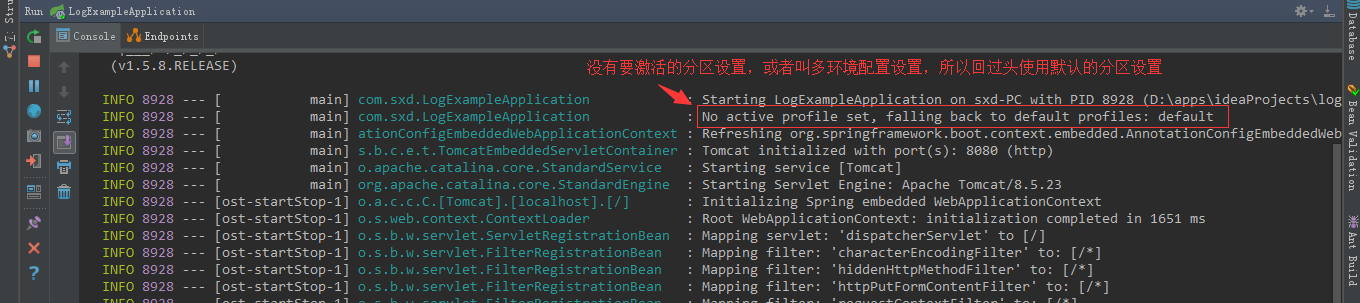
　　第三：最后application.properties配置文件中，选择激活哪一种或哪几种环境即可

https://images2017.cnblogs.com/blog/978388/201711/978388-20171119192227640-700454134.png



附录：

 在没有设置profile的情况下，一般启动情况下，控制台打印出的日志都能看到这么一句话：



**3.拆析一个基本的logback-spring.xml文件**

下面粘上一个基本的文件代码：

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif View Code

**根节点<configuration>包含的属性**

* scan:当此属性设置为true时，配置文件如果发生改变，将会被重新加载，默认值为true。
* scanPeriod:设置监测配置文件是否有修改的时间间隔，如果没有给出时间单位，默认单位是毫秒。当scan为true时，此属性生效。默认的时间间隔为1分钟。
* debug:当此属性设置为true时，将打印出logback内部日志信息，实时查看logback运行状态。默认值为false。

根节点<configuration>的子节点：

<configuration>下面一共有2个属性，3个子节点，分别是：

**属性一：设置上下文名称<contextName>**

每个logger都关联到logger上下文，默认上下文名称为“default”。但可以使用<contextName>设置成其他名字，用于区分不同应用程序的记录。一旦设置，不能修改,可以通过%contextName来打印日志上下文名称。

<contextName>logback</contextName>

**属性二：设置变量<property>**

用来定义变量值的标签，<property> 有两个属性，name和value；其中name的值是变量的名称，value的值时变量定义的值。通过<property>定义的值会被插入到logger上下文中。定义变量后，可以使“${}”来使用变量。

<property name="log.path" value="E:\\logback.log" />

**子节点一<appender>**

appender用来格式化日志输出节点，有俩个属性name和class，class用来指定哪种输出策略，常用就是控制台输出策略和文件输出策略。

**控制台输出ConsoleAppender：**



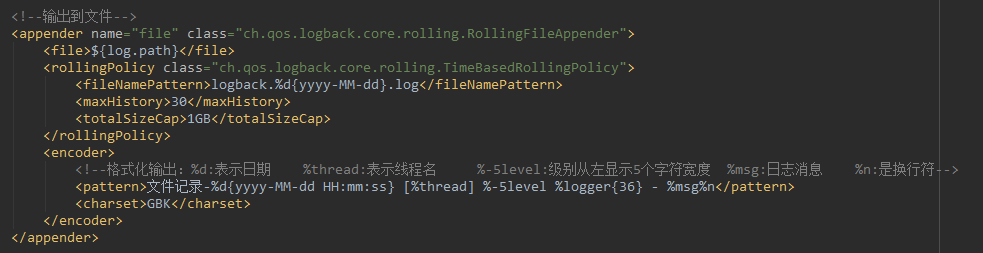
<encoder>表示对日志进行编码：

* %d{HH: mm:ss.SSS}——日志输出时间
* %thread——输出日志的进程名字，这在Web应用以及异步任务处理中很有用
* %-5level——日志级别，并且使用5个字符靠左对齐
* %logger{36}——日志输出者的名字
* %msg——日志消息
* %n——平台的换行符

ThresholdFilter为系统定义的拦截器，例如我们用ThresholdFilter来过滤掉ERROR级别以下的日志不输出到文件中。如果不用记得注释掉，不然你控制台会发现没日志~

**输出到文件RollingFileAppender**

另一种常见的日志输出到文件，随着应用的运行时间越来越长，日志也会增长的越来越多，将他们输出到同一个文件并非一个好办法。RollingFileAppender用于切分文件日志：



其中重要的是rollingPolicy的定义，上例中

<fileNamePattern>logback.%d{yyyy-MM-dd}.log</fileNamePattern>定义了日志的切分方式——把每一天的日志归档到一个文件中，

<maxHistory>30</maxHistory>表示只保留最近30天的日志，以防止日志填满整个磁盘空间。

同理，可以使用%d{yyyy-MM-dd\_HH-mm}来定义精确到分的日志切分方式。

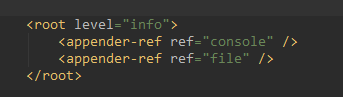
<totalSizeCap>1GB</totalSizeCap>用来指定日志文件的上限大小，例如设置为1GB的话，那么到了这个值，就会删除旧的日志。

**子节点二<root>**

root节点是必选节点，用来指定最基础的日志输出级别，只有一个level属性。

* level:用来设置打印级别，大小写无关：TRACE, DEBUG, INFO, WARN, ERROR, ALL 和 OFF，不能设置为INHERITED或者同义词NULL，默认是DEBUG。

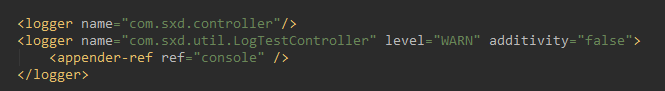
<root>可以包含零个或多个<appender-ref>元素，标识这个appender将会添加到这个logger。



**子节点三<logger>**

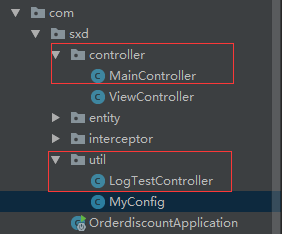
<logger>用来设置某一个包或者具体的某一个类的日志打印级别、以及指定<appender>。<loger>仅有一个name属性，一个可选的level和一个可选的addtivity属性。

* name:用来指定受此loger约束的某一个包或者具体的某一个类。
* level:用来设置打印级别，大小写无关：TRACE, DEBUG, INFO, WARN, ERROR, ALL 和 OFF，还有一个特俗值INHERITED或者同义词NULL，代表强制执行上级的级别。如果未设置此属性，那么当前loger将会继承上级的级别。
* addtivity:是否向上级logger传递打印信息。默认是true。



从上面可以看到，logger在实际使用的时候有两种情况。

然后分别在com.sxd.controller中和com.sxd.util.LogTestController中添加相同的日志打印，分别看看对应上面两种<logger>的使用情况：



com.sxd.controller.MainController

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif View Code

com.sxd.util.LogTestController

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif View Code

我们引入日志，并且打印不同级别的日志，然后根据logback-spring.xml中的配置来看看打印了哪几种级别日志。

**第一种：带有loger的配置，不指定级别，不指定appender[对应com.sxd.controller.MainController]**

https://images2017.cnblogs.com/blog/978388/201711/978388-20171123153348290-1530338561.png

**<logger name="com.sxd.controller"/>**将控制controller包下的所有类的日志的打印，但是并没用设置打印级别，所以继承他的上级<root>的日志级别“info”；

没有设置addtivity，默认为true，将此loger的打印信息向上级传递；

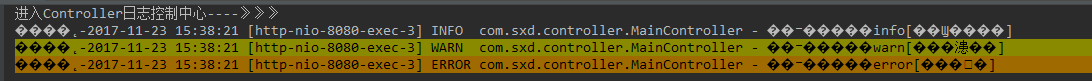
没有设置appender，此loger本身不打印任何信息。

<root level="info">将root的打印级别设置为“info”，指定了名字为“console”的appender。

当执行com.sxd.controller.MainController类的log方法时，MainController在包com.sxd.controllerr中，所以首先执行**<logger name="com.sxd.controller"/>**，将级别为“info”及大于“info”的日志信息传递给root，本身并不打印；

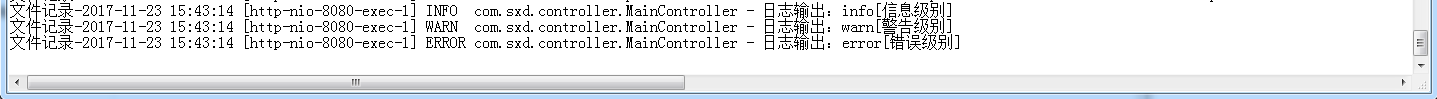
root接到下级传递的信息，交给已经配置好的名为“console”的appender处理，“console”appender将信息打印到控制台；

打印结果如下：【[spring boot日志乱码的问题](http://www.cnblogs.com/sxdcgaq8080/p/7881562.html)】



root接到下级传递的信息，交给已经配置好的名为“file”的appender处理，“file”appender将信息写入到日志文件中，如下；

https://images2017.cnblogs.com/blog/978388/201711/978388-20171123154014258-2075014708.png



**第二种：带有多个loger的配置，指定级别，指定appender【对应com.sxd.util.LogTestController】**

https://images2017.cnblogs.com/blog/978388/201711/978388-20171123154606274-1432368138.png

控制com.sxd.util.LogTestController类的日志打印，打印级别为“WARN”

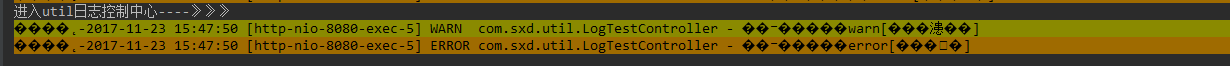
additivity属性为false，表示此loger的打印信息不再向上级传递

指定了名字为“console”的appender

这时候执行com.sxd.util.LogTestController类的log方法时，先执行**<logger name="com.sxd.util.LogTestController" level="WARN" additivity="false">**,

将级别为“WARN”及大于“WARN”的日志信息交给此loger指定的名为“console”的appender处理，在控制台中打出日志，不再向上级root传递打印信息。

打印结果如下：



注意1：

当然，此时日志文件中是没有本次日志的，因为本<logger>中指定了<appender-ref ref="console" />,特意的指出来，在console打印。所以file中并没有。

注意2：

如果你把additivity="false"改成additivity="true"的话，就会打印两次，因为打印信息向上级传递，logger本身打印一次，root接到后又打印一次。此时，file中也会有日志。

==================================================================完结=================================================================================、

所以，最后的最后。

如果你要在代码中使用日志的话，仅需要添加

Logger logger = LoggerFactory.getLogger(this.getClass());

然后就可以使用了。

附录1：[spring boot中使用logback日志乱码处理](http://www.cnblogs.com/sxdcgaq8080/p/7881562.html)

附录2：[spring boot中使用logback日志颜色渲染](http://www.cnblogs.com/sxdcgaq8080/p/7885340.html)

附录3：[spring boot使用logback-spring.xml文件后，application.properties文件中关于logging的配置还会起作用么？](http://www.cnblogs.com/sxdcgaq8080/p/7885365.html)

分类: [spring boot](https://www.cnblogs.com/sxdcgaq8080/category/1095618.html),[【logback】](https://www.cnblogs.com/sxdcgaq8080/category/1119875.html)