

TRACE < DEBUG < INFO < WARN < ERROR < FATAL

%d{HH:mm:ss.SSS} 表示输出到毫秒的时间

%t 输出当前线程名称

%-5level 输出日志级别，-5表示左对齐并且固定输出5个字符，如果不足在右边补0

%logger 输出logger名称，因为Root Logger没有名称，所以没有输出

%msg 日志文本

%n 换行

其他常用的占位符有：

%F 输出所在的类文件名，如Client.java

%L 输出行号

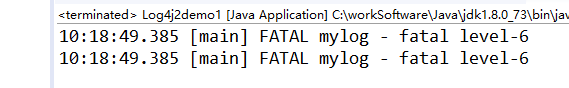
%M 输出所在方法名

%l  输出语句所在的行数, 包括类名、方法名、文件名、行数

最后是Logger的配置，这里只配置了一个Root Logger。

additivity="false"表示在该logger中输出的日志不会再延伸到父层logger。这里如果改为true，则会延伸到Root Logger，遵循Root Logger的配置也输出一次。

注意根节点增加了一个monitorInterval属性，含义是每隔300秒重新读取配置文件，可以不重启应用的情况下修改配置，还是很好用的功能。



**3 自定义Appender**

****

<properties>定义了两个常量方便后面复用

RollingRandomAccessFile的属性：

fileName  指定当前日志文件的位置和文件名称

filePattern  指定当发生Rolling时，文件的转移和重命名规则

SizeBasedTriggeringPolicy  指定当文件体积大于size指定的值时，触发Rolling

DefaultRolloverStrategy  指定最多保存的文件个数

TimeBasedTriggeringPolicy  这个配置需要和filePattern结合使用，注意filePattern中配置的文件重命名规则是${FILE\_NAME}-%d{yyyy-MM-dd HH-mm}-%i，最小的时间粒度是mm，即分钟，TimeBasedTriggeringPolicy指定的size是1，结合起来就是每1分钟生成一个新文件。如果改成%d{yyyy-MM-dd HH}，最小粒度为小时，则

过滤器也有很多种，具体可以参考手册的第十一章Filters，大家可以去具体看一下。下面介绍一种比较实用的过滤器，组合过滤器CompositeFilter。CompositeFilter可以包含多种不同的过滤器，这样，多种过滤条件可以同时过滤日志信息，符合所有条件的日志信息才会被写入日志文件。下面举一个例子：

**[html]** [view plain](http://blog.csdn.net/chenhaotong/article/details/50487557) [copy](http://blog.csdn.net/chenhaotong/article/details/50487557)

1. **<Filters>**
2. **<ThresholdFilter** level="TRACE" onMatch="NEUTRAL" onMismatch="DENY"**/>**
3. **<RegexFilter** regex=".\* test .\*" onMatch="NEUTRAL" onMismatch="DENY"**/>**
4. **<TimeFilter** start="05:00:00" end="05:30:00" onMatch=" NEUTRAL " onMismatch="DENY"**/>**
5. **</Filters>**

<Filters>是组合过滤器额标签，它包含的子标签是具体的过滤器，这三个具体过滤器分别是日志等级过滤器，正则表达式过滤器和时间过滤器。

需要说明的是onMatch和onMismatch可以选择的值及其含义。可选的值分别是，ACCEPT, DENY, NEUTRAL，ACCEP和DENY比较好理解就是接受和拒绝的意思，在使用单个过滤器的时候，一般就是使用这两个值。但是在组合过滤器中，如果用接受ACCEPT的话，日志信息就会直接写入日志文件,后续的过滤器不再进行过滤。所以，在组合过滤器中，接受使用NEUTRAL（中立），被第一个过滤器接受的日志信息，会继续用后面的过滤器进行过滤，只有符合所有过滤器条件的日志信息，才会被最终写入日志文件

日志按级别输出

