

一、日志

1、配置日志级别

日志记录器 (Logger) 的行为是分等级的。如下表所示：

分为：OFF、FATAL、ERROR、WARN、INFO、DEBUG、ALL

默认情况下，spring boot从控制台打印出来的日志级别只有INFO及以上级别，可以配置日志级别

```
1 # 设置日志级别
2 logging.level.root=WARN
```

这种方式只能将日志打印在控制台上

二、Logback日志

spring boot内部使用Logback作为日志实现的框架。

Logback和log4j非常相似，如果你对log4j很熟悉，那对logback很快就会得心应手。

logback相对于log4j的一些优点：https://blog.csdn.net/caisini_vc/article/details/48551287

1、配置logback日志

删除application.properties中的日志配置

安装idea彩色日志插件：grep-console

resources 中创建 logback-spring.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <configuration scan="true" scanPeriod="10 seconds">
3     <!-- 日志级别从低到高分TRACE < DEBUG < INFO < WARN < ERROR < FATAL，如果设置为WARN
4     <!-- 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 -->
5     <!-- 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，默认单位是秒
6     <!-- 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。
7
8     <contextName>logback</contextName>
9     <!-- name的值是变量的名称，value的值是变量定义的值。通过定义的值会被插入到logger上下文
```

```
10 <property name="log.path" value="D:/guli_log/edu" />
11
12 <!-- 彩色日志 -->
13 <!-- 配置格式变量: CONSOLE_LOG_PATTERN 彩色日志格式 -->
14 <!-- magenta:洋红 -->
15 <!-- boldMagenta:粗红-->
16 <!-- cyan:青色 -->
17 <!-- white:白色 -->
18 <!-- magenta:洋红 -->
19 <property name="CONSOLE_LOG_PATTERN"
20         value="%yellow(%date{yyyy-MM-dd HH:mm:ss}) |%highlight(%-5level) |%blue
21
22
23 <!--输出到控制台-->
24 <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
25     <!--此日志appender是为开发使用，只配置最底级别，控制台输出的日志级别是大于或等于此级别-->
26     <!-- 例如：如果此处配置了INFO级别，则后面其他位置即使配置了DEBUG级别的日志，也不会被记录 -->
27     <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
28         <level>INFO</level>
29     </filter>
30     <encoder>
31         <Pattern>${CONSOLE_LOG_PATTERN}</Pattern>
32         <!-- 设置字符集 -->
33         <charset>UTF-8</charset>
34     </encoder>
35 </appender>
36
37
38 <!--输出到文件-->
39
40 <!-- 时间滚动输出 level为 INFO 日志 -->
41 <appender name="INFO_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
42     <!-- 正在记录的日志文件的路径及文件名 -->
43     <file>${log.path}/log_info.log</file>
44     <!--日志文件输出格式-->
45     <encoder>
46         <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n</pattern>
47         <charset>UTF-8</charset>
48     </encoder>
49     <!-- 日志记录器的滚动策略，按日期，按大小记录 -->
50     <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
51         <!-- 每天日志归档路径以及格式 -->
52         <fileNamePattern>${log.path}/info/log-info-%d{yyyy-MM-dd}.%i.log</fileNar
```

```

53         <timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling
54             <maxFileSize>100MB</maxFileSize>
55         </timeBasedFileNamingAndTriggeringPolicy>
56         <!-- 日志文件保留天数-->
57         <maxHistory>15</maxHistory>
58     </rollingPolicy>
59     <!-- 此日志文件只记录info级别的 -->
60     <filter class="ch.qos.logback.classic.filter.LevelFilter">
61         <level>INFO</level>
62         <onMatch>ACCEPT</onMatch>
63         <onMismatch>DENY</onMismatch>
64     </filter>
65 </appender>
66
67 <!-- 时间滚动输出 level为 WARN 日志 -->
68 <appender name="WARN_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender
69     <!-- 正在记录的日志文件的路径及文件名 -->
70     <file>${log.path}/log_warn.log</file>
71     <!-- 日志文件输出格式-->
72     <encoder>
73         <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %ms
74         <charset>UTF-8</charset> <!-- 此处设置字符集 -->
75     </encoder>
76     <!-- 日志记录器的滚动策略，按日期，按大小记录 -->
77     <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
78         <fileNamePattern>${log.path}/warn/log-warn-%d{yyyy-MM-dd}.%i.log</fileNar
79         <timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling
80             <maxFileSize>100MB</maxFileSize>
81         </timeBasedFileNamingAndTriggeringPolicy>
82         <!-- 日志文件保留天数-->
83         <maxHistory>15</maxHistory>
84     </rollingPolicy>
85     <!-- 此日志文件只记录warn级别的 -->
86     <filter class="ch.qos.logback.classic.filter.LevelFilter">
87         <level>warn</level>
88         <onMatch>ACCEPT</onMatch>
89         <onMismatch>DENY</onMismatch>
90     </filter>
91 </appender>
92
93
94 <!-- 时间滚动输出 level为 ERROR 日志 -->
95 <appender name="ERROR_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender

```

```

96      <!-- 正在记录的日志文件的路径及文件名 -->
97      <file>${log.path}/log_error.log</file>
98      <!--日志文件输出格式-->
99      <encoder>
100          <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %ms
101          <charset>UTF-8</charset> <!-- 此处设置字符集 -->
102      </encoder>
103      <!-- 日志记录器的滚动策略，按日期，按大小记录 -->
104      <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
105          <fileNamePattern>${log.path}/error/log-error-%d{yyyy-MM-dd}.%i.log</file>
106          <timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling
107              <maxFileSize>100MB</maxFileSize>
108          </timeBasedFileNamingAndTriggeringPolicy>
109          <!--日志文件保留天数-->
110          <maxHistory>15</maxHistory>
111      </rollingPolicy>
112      <!-- 此日志文件只记录ERROR级别的 -->
113      <filter class="ch.qos.logback.classic.filter.LevelFilter">
114          <level>ERROR</level>
115          <onMatch>ACCEPT</onMatch>
116          <onMismatch>DENY</onMismatch>
117      </filter>
118  </appender>
119
120  <!--
121      <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>。
122      <logger>仅有一个name属性，
123      一个可选的level和一个可选的additivity属性。
124      name:用来指定受此logger约束的某一个包或者具体的某一个类。
125      level:用来设置打印级别，大小写无关：TRACE，DEBUG，INFO，WARN，ERROR，ALL 和 OFF
126          如果未设置此属性，那么当前logger将会继承上级的级别。
127  -->
128  <!--
129      使用mybatis的时候，sql语句是debug下才会打印，而这里我们只配置了info，所以想要查看s
130      第一种把<root level="INFO">改成<root level="DEBUG">这样就会打印sql，不过这样日志
131      第二种就是单独给mapper下目录配置DEBUG模式，代码如下，这样配置sql语句会打印，其他还
132  -->
133  <!--开发环境:打印控制台-->
134  <springProfile name="dev">
135      <!--可以输出项目中的debug日志，包括mybatis的sql日志-->
136      <logger name="com.guli" level="INFO" />
137
138  <!--

```

```
139      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
140      level:用来设置打印级别，大小写无关：TRACE，DEBUG，INFO，WARN，ERROR，ALL 和
141      可以包含零个或多个appender元素。
142      -->
143      <root level="INFO">
144          <appender-ref ref="CONSOLE" />
145          <appender-ref ref="INFO_FILE" />
146          <appender-ref ref="WARN_FILE" />
147          <appender-ref ref="ERROR_FILE" />
148      </root>
149  </springProfile>
150
151
152  <!--生产环境:输出到文件-->
153  <springProfile name="pro">
154
155      <root level="INFO">
156          <appender-ref ref="CONSOLE" />
157          <appender-ref ref="DEBUG_FILE" />
158          <appender-ref ref="INFO_FILE" />
159          <appender-ref ref="ERROR_FILE" />
160          <appender-ref ref="WARN_FILE" />
161      </root>
162  </springProfile>
163
164  </configuration>
```

2、将错误日志输出到文件

GlobalExceptionHandler.java 中

类上添加注解

```
1  @Slf4j
```

异常输出语句

```
1  log.error(e.getMessage());
```

