**[springboot2.0整合logback日志(详细)](https://www.cnblogs.com/zhangjianbing/p/8992897.html)**

一。

近期自己的项目想要一个记录日志的功能，而springboot本身就内置了日志功能，然而想要输入想要的日志，并且输出到磁盘，然后按天归档，或者日志的切分什么的，自带的日志仅仅具有简单的功能，百度了一番，总结如下，适合大多数的应用场景

二。

springboot的pom文件都会引一个parent

<parent>

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

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

<version>2.0.0.RELEASE</version>

</parent>

点进去这个parent，会有一个这个dependency

<parent>

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

<artifactId>spring-boot-dependencies</artifactId>

<version>2.0.0.RELEASE</version>

<relativePath>../../spring-boot-dependencies</relativePath>

</parent>

再点进去就是2.0版本，所谓的它给你集成的各种包依赖，而且规定了版本号，其中有一个包如下

<dependency>

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

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

<version>2.0.0.RELEASE</version>

</dependency>

再点进去

<dependency>

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

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

<version>2.0.0.RELEASE</version>

<scope>compile</scope>

</dependency>

再点，这些都是原有的日志包，所以，不必再引依赖了，直接用就ok了，用法如下：

<dependencies>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-to-slf4j</artifactId>

<version>2.10.0</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>jul-to-slf4j</artifactId>

<version>1.7.25</version>

<scope>compile</scope>

</dependency>

============================分割线====================================

首先，官方推荐使用的xml名字的格式为：logback-spring.xml而不是logback.xml，至于为什么，因为带spring后缀的可以使用<springProfile>这个标签。

在resource下创建logback-spring.xml文件

内容如下，每行都有注释了

<?xml version="1.0" encoding="UTF-8"?>

<!-- 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 -->

<!-- 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。 -->

<configuration scan="true" scanPeriod="10 seconds">

<!--<include resource="org/springframework/boot/logging/logback/base.xml" />-->

<contextName>logback</contextName>

<!-- name的值是变量的名称，value的值时变量定义的值。通过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="D:/nmyslog/nmys" />

<!-- 彩色日志 -->

<!-- 彩色日志依赖的渲染类 -->

<conversionRule conversionWord="clr" converterClass="org.springframework.boot.logging.logback.ColorConverter" />

<conversionRule conversionWord="wex" converterClass="org.springframework.boot.logging.logback.WhitespaceThrowableProxyConverter" />

<conversionRule conversionWord="wEx" converterClass="org.springframework.boot.logging.logback.ExtendedWhitespaceThrowableProxyConverter" />

<!-- 彩色日志格式 -->

<property name="CONSOLE\_LOG\_PATTERN" value="${CONSOLE\_LOG\_PATTERN:-%clr(%d{yyyy-MM-dd HH:mm:ss.SSS}){faint} %clr(${LOG\_LEVEL\_PATTERN:-%5p}) %clr(${PID:- }){magenta} %clr(---){faint} %clr([%15.15t]){faint} %clr(%-40.40logger{39}){cyan} %clr(:){faint} %m%n${LOG\_EXCEPTION\_CONVERSION\_WORD:-%wEx}}"/>

<!--输出到控制台-->

<appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">

<!--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的日志信息-->

<filter class="ch.qos.logback.classic.filter.ThresholdFilter">

<level>info</level>

</filter>

<encoder>

<Pattern>${CONSOLE\_LOG\_PATTERN}</Pattern>

<!-- 设置字符集 -->

<charset>UTF-8</charset>

</encoder>

</appender>

<!--输出到文件-->

<!-- 时间滚动输出 level为 DEBUG 日志 -->

<appender name="DEBUG\_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">

<!-- 正在记录的日志文件的路径及文件名 -->

<file>${log.path}/log\_debug.log</file>

<!--日志文件输出格式-->

<encoder>

<pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n</pattern>

<charset>UTF-8</charset> <!-- 设置字符集 -->

</encoder>

<!-- 日志记录器的滚动策略，按日期，按大小记录 -->

<rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<!-- 日志归档 -->

<fileNamePattern>${log.path}/debug/log-debug-%d{yyyy-MM-dd}.%i.log</fileNamePattern>

<timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">

<maxFileSize>100MB</maxFileSize>

</timeBasedFileNamingAndTriggeringPolicy>

<!--日志文件保留天数-->

<maxHistory>15</maxHistory>

</rollingPolicy>

<!-- 此日志文件只记录debug级别的 -->

<filter class="ch.qos.logback.classic.filter.LevelFilter">

<level>debug</level>

<onMatch>ACCEPT</onMatch>

<onMismatch>DENY</onMismatch>

</filter>

</appender>

<!-- 时间滚动输出 level为 INFO 日志 -->

<appender name="INFO\_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">

<!-- 正在记录的日志文件的路径及文件名 -->

<file>${log.path}/log\_info.log</file>

<!--日志文件输出格式-->

<encoder>

<pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n</pattern>

<charset>UTF-8</charset>

</encoder>

<!-- 日志记录器的滚动策略，按日期，按大小记录 -->

<rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<!-- 每天日志归档路径以及格式 -->

<fileNamePattern>${log.path}/info/log-info-%d{yyyy-MM-dd}.%i.log</fileNamePattern>

<timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">

<maxFileSize>100MB</maxFileSize>

</timeBasedFileNamingAndTriggeringPolicy>

<!--日志文件保留天数-->

<maxHistory>15</maxHistory>

</rollingPolicy>

<!-- 此日志文件只记录info级别的 -->

<filter class="ch.qos.logback.classic.filter.LevelFilter">

<level>info</level>

<onMatch>ACCEPT</onMatch>

<onMismatch>DENY</onMismatch>

</filter>

</appender>

<!-- 时间滚动输出 level为 WARN 日志 -->

<appender name="WARN\_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">

<!-- 正在记录的日志文件的路径及文件名 -->

<file>${log.path}/log\_warn.log</file>

<!--日志文件输出格式-->

<encoder>

<pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n</pattern>

<charset>UTF-8</charset> <!-- 此处设置字符集 -->

</encoder>

<!-- 日志记录器的滚动策略，按日期，按大小记录 -->

<rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<fileNamePattern>${log.path}/warn/log-warn-%d{yyyy-MM-dd}.%i.log</fileNamePattern>

<timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">

<maxFileSize>100MB</maxFileSize>

</timeBasedFileNamingAndTriggeringPolicy>

<!--日志文件保留天数-->

<maxHistory>15</maxHistory>

</rollingPolicy>

<!-- 此日志文件只记录warn级别的 -->

<filter class="ch.qos.logback.classic.filter.LevelFilter">

<level>warn</level>

<onMatch>ACCEPT</onMatch>

<onMismatch>DENY</onMismatch>

</filter>

</appender>

<!-- 时间滚动输出 level为 ERROR 日志 -->

<appender name="ERROR\_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">

<!-- 正在记录的日志文件的路径及文件名 -->

<file>${log.path}/log\_error.log</file>

<!--日志文件输出格式-->

<encoder>

<pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n</pattern>

<charset>UTF-8</charset> <!-- 此处设置字符集 -->

</encoder>

<!-- 日志记录器的滚动策略，按日期，按大小记录 -->

<rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<fileNamePattern>${log.path}/error/log-error-%d{yyyy-MM-dd}.%i.log</fileNamePattern>

<timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">

<maxFileSize>100MB</maxFileSize>

</timeBasedFileNamingAndTriggeringPolicy>

<!--日志文件保留天数-->

<maxHistory>15</maxHistory>

</rollingPolicy>

<!-- 此日志文件只记录ERROR级别的 -->

<filter class="ch.qos.logback.classic.filter.LevelFilter">

<level>ERROR</level>

<onMatch>ACCEPT</onMatch>

<onMismatch>DENY</onMismatch>

</filter>

</appender>

<!--

<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>。<logger>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logger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logger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。

-->

<!--<logger name="org.springframework.web" level="info"/>-->

<!--<logger name="org.springframework.scheduling.annotation.ScheduledAnnotationBeanPostProcessor" level="INFO"/>-->

<!--

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sql语句的话，有以下两种操作：

第一种把<root level="info">改成<root level="DEBUG">这样就会打印sql，不过这样日志那边会出现很多其他消息

第二种就是单独给dao下目录配置debug模式，代码如下，这样配置sql语句会打印，其他还是正常info级别：

-->

<!--

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

可以包含零个或多个元素，标识这个appender将会添加到这个logger。

-->

<!--开发环境:打印控制台-->

<springProfile name="dev">

<logger name="com.nmys.view" level="debug"/>

</springProfile>

<root level="info">

<appender-ref ref="CONSOLE" />

<appender-ref ref="DEBUG\_FILE" />

<appender-ref ref="INFO\_FILE" />

<appender-ref ref="WARN\_FILE" />

<appender-ref ref="ERROR\_FILE" />

</root>

<!--生产环境:输出到文件-->

<!--<springProfile name="pro">-->

<!--<root level="info">-->

<!--<appender-ref ref="CONSOLE" />-->

<!--<appender-ref ref="DEBUG\_FILE" />-->

<!--<appender-ref ref="INFO\_FILE" />-->

<!--<appender-ref ref="ERROR\_FILE" />-->

<!--<appender-ref ref="WARN\_FILE" />-->

<!--</root>-->

<!--</springProfile>-->

</configuration>

其中说一下，彩色日志这个功能，需要下载插件，然后在yml或者properties中配置。

logging.config=classpath:log/logback-spring.xml

logging.path=D:/nmyslog/nmys