

📖 企业级日志输出 - 自定义日志配置

🐼 官网:

<http://logback.qos.ch/>

🐼 日志

- 优点: 可以协助和帮助我们调试和分析业务
- 缺点: 日志级别越高, 对程序方法执行性能有所损耗。一定要想清楚, 所以为什么说在生产环境一般配置成`error`。除了特殊的业务, 你单独开启针对日志级别。

🐼 **springboot**自身支持**logback**的日志问题

- 输出的方式只有两种: 文件 和 控制台
 - 输出数据库
 - 输出rabbitmq
- 文件的分隔, 文件大小分隔不方便控制

🐼 如果使用自定义的日志配置我们需要做什么事情?

- 把`application.yml`中配置的`logging.xxx`全部删除注释掉。
- 因为现在所有日志的管理和处理都是由`logback-spring.xml`来管理了。

🐼 01、官网

<https://docs.spring.io/spring-boot/docs/2.6.1/reference/htmlsingle/#features.logging>

🐼 02、自定义日志配置

可以通过在类路径中包含适当的库来激活各种日志系统，并且可以通过在类路径的根目录或以下 `SpringEnvironment` 属性指定的位置提供合适的配置文件来进一步定制：`logging.config`。

您可以使用 `org.springframework.boot.logging.LoggingSystem` 属性强制 Spring Boot 使用特定的日志记录系统。该值应该是实现的完全限定类名 `LoggingSystem`。您还可以使用值完全禁用 Spring Boot 的日志记录配置 `none`。

根据您的日志系统，加载以下文件：

日志系统	定制
登录	<code>logback-spring.xml</code> , <code>logback-spring.groovy</code> , <code>logback.xml</code> , 或 <code>logback.groovy</code>
日志4j2	<code>log4j2-spring.xml</code> 或者 <code>log4j2.xml</code>
JDK（Java 实用程序日志记录）	<code>logging.properties</code>

springboot建议使用`logback-spring.xml`的方式配置Logback。日志一定早于`application.properties`的初始化，因此会在`ApplicaitonContext`创建之前进行初始化。如下：

🐼 03、配置表

为了帮助定制，一些其他属性从 Spring 传输 `Environment` 到系统属性，如下表所述：

配置文件属性	系统属性	评论
<code>logging.exception-conversion-word</code>	<code>LOG_EXCEPTION_CONVERSION_WORD</code>	记录异常时使用的转换字。
<code>logging.file.name</code>	<code>LOG_FILE</code>	如果定义，则在默认日志配置中使用。
<code>logging.file.path</code>	<code>LOG_PATH</code>	如果定义，则在默认日志配置中使用。
<code>logging.pattern.console</code>	<code>CONSOLE_LOG_PATTERN</code>	要在控制台 (stdout) 上使用的日志模式。

配置文件属性	系统属性	评论
<code>logging.pattern.dateformat</code>	<code>LOG_DATEFORMAT_PATTERN</code>	日志日期格式的 Appender 模式。
<code>logging.charset.console</code>	<code>CONSOLE_LOG_CHARSET</code>	用于控制台日志记录的字符集。
<code>logging.pattern.file</code>	<code>FILE_LOG_PATTERN</code>	要在文件中使用的日志模式（如果 <code>LOG_FILE</code> 已启用）。
<code>logging.charset.file</code>	<code>FILE_LOG_CHARSET</code>	用于文件记录的字符集（如果 <code>LOG_FILE</code> 已启用）。
<code>logging.pattern.level</code>	<code>LOG_LEVEL_PATTERN</code>	呈现日志级别时使用的格式（默认 <code>%5p</code> ）。
<code>PID</code>	<code>PID</code>	当前进程 ID（如果可能并且尚未定义为操作系统环境变量时发现）。

如果使用 Logback，还会传输以下属性：

配置文件属性	系统属性	评论
<code>logging.logback.rollingpolicy.file-name-pattern</code>	<code>LOGBACK_ROLLINGPOLICY_FILE_NAME_PATTERN</code>	滚动日志文件名的模式（默认 <code>\${LOG_FILE}_%d{yyyy-MM-dd}_%i.gz</code> ）。
<code>logging.logback.rollingpolicy.clean-history-on-start</code>	<code>LOGBACK_ROLLINGPOLICY_CLEAN_HISTORY_ON_START</code>	是否在启动时清除存档日志文件。
<code>logging.logback.rollingpolicy.max-file-size</code>	<code>LOGBACK_ROLLINGPOLICY_MAX_FILE_SIZE</code>	最大日志文件大小。
<code>logging.logback.rollingpolicy.total-size-cap</code>	<code>LOGBACK_ROLLINGPOLICY_TOTAL_SIZE_CAP</code>	要保留的日志备份的总大小。
<code>logging.logback.rollingpolicy.max-history</code>	<code>LOGBACK_ROLLINGPOLICY_MAX_HISTORY</code>	要保留的最大归档日志文件数。

所有支持的日志系统在解析其配置文件时都可以查询系统属性。有关 `spring-boot.jar` 示例，请参阅中的默认配置：

- [登录](#)
- [Log4j 2](#)
- [Java Util 日志记录](#)

🔗 04、日志入库

在pom.xml导入依赖

```
1 <dependency>
2     <groupId>mysql</groupId>
3     <artifactId>mysql-connector-java</artifactId>
4 </dependency>
5
6 <dependency>
7     <groupId>org.springframework.boot</groupId>
8     <artifactId>spring-boot-starter-jdbc</artifactId>
9 </dependency>
```

第二步：新建三个表

```
1 # Logback: the reliable, generic, fast and flexible logging framework.
2 # Copyright (C) 1999-2010, QOS.ch. All rights reserved.
3 #
4 # See http://logback.qos.ch/license.html for the applicable licensing
5 # conditions.
6
7 # This SQL script creates the required tables by
8 # ch.qos.logback.classic.db.DBAppender.
9 #
10 # It is intended for MySQL databases. It has been tested on MySQL
11 # 5.1.37
12 # on Linux
13
14 BEGIN;
15 DROP TABLE IF EXISTS logging_event_property;
16 DROP TABLE IF EXISTS logging_event_exception;
17 DROP TABLE IF EXISTS logging_event;
18 COMMIT;
19
20 BEGIN;
21 CREATE TABLE logging_event
22 (
23     timestamp          BIGINT NOT NULL,
24     formatted_message  TEXT NOT NULL,
25     logger_name        VARCHAR(254) NOT NULL,
```

```

25     level_string      VARCHAR(254) NOT NULL,
26     thread_name       VARCHAR(254),
27     reference_flag    SMALLINT,
28     arg0              VARCHAR(254),
29     arg1              VARCHAR(254),
30     arg2              VARCHAR(254),
31     arg3              VARCHAR(254),
32     caller_filename   VARCHAR(254) NOT NULL,
33     caller_class      VARCHAR(254) NOT NULL,
34     caller_method     VARCHAR(254) NOT NULL,
35     caller_line       CHAR(4) NOT NULL,
36     event_id          BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY
37 );
38 COMMIT;
39
40 BEGIN;
41 CREATE TABLE logging_event_property
42 (
43     event_id          BIGINT NOT NULL,
44     mapped_key        VARCHAR(254) NOT NULL,
45     mapped_value      TEXT,
46     PRIMARY KEY(event_id, mapped_key),
47     FOREIGN KEY (event_id) REFERENCES logging_event(event_id)
48 );
49 COMMIT;
50
51 BEGIN;
52 CREATE TABLE logging_event_exception
53 (
54     event_id          BIGINT NOT NULL,
55     i                 SMALLINT NOT NULL,
56     trace_line        VARCHAR(254) NOT NULL,
57     PRIMARY KEY(event_id, i),
58     FOREIGN KEY (event_id) REFERENCES logging_event(event_id)
59 );
60 COMMIT;

```

第三步：在logback-spring.xml配置数据写入appender即可如下：

```

1  <appender name="DBAPPENDER"
    class="ch.qos.logback.classic.db.DBAppender">

```

```

2      <connectionSource
class="ch.qos.logback.core.db.DataSourceConnectionSource">
3      <dataSource class="com.zaxxer.hikari.HikariDataSource">
4
5      <driverClassName>com.mysql.jdbc.Driver</driverClassName>
6      <jdbcUrl>jdbc:mysql://localhost:3306/kss-web-db?
useUnicode=true&characterEncoding=utf8&useSSL=false</jdbcUrl>
7      <username>root</username>
8      <password>mkxiaoer</password>
9      <poolName>HikariPool-logback</poolName>
10     </dataSource>
11 </connectionSource>
12 <!-- 此日志文件只记录info级别的 -->
13 <!--      <filter
class="ch.qos.logback.classic.filter.LevelFilter">-->
14 <!--          <level>ERROR</level>-->
15 <!--          <onMatch>ACCEPT</onMatch>-->
16 <!--          <onMismatch>DENY</onMismatch>-->
17 <!--      </filter>-->
18 </appender>

```

05、滚动日志

- 1 **RollingFileAppender**：滚动记录文件，先将日志记录到指定文件，当符合某个条件时，将日志记录到其他文件

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <configuration debug="true" scan="true" scanPeriod="1 seconds">
3
4     <contextName>webA</contextName>
5
6     <appender name="file" append="true"
class="ch.qos.logback.core.rolling.RollingFileAppender">
7
8         <file>/logs/log_file.log</file>
9
10        <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

```

```

11         <fileNamePattern>/logs/log_file_%d{yyyy-mm-
dd.HH}.log</fileNamePattern>
12         <maxHistory>30</maxHistory>
13
14         <totalSizeCap>1GB</totalSizeCap>
15     </rollingPolicy>
16
17     <triggeringPolicy
class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
18         <maxFileSize>100MB</maxFileSize>
19     </triggeringPolicy>
20
21     <encoder>
22         <pattern>%cn%d.%M.%m%n</pattern>
23     </encoder>
24 </appender>
25
26 <root level="debug">
27     <appender-ref ref="file"/>
28 </root>
29
30 </configuration>

```

主要节点：

(1) **file** 文件名，可选。若没有设定参考下面 **fileNamePattern** 名称生成文件

(2) **rollingPolicy** 当日志发生滚动时，决定日志文件的行为。决定文件的重命名及路径的变更。必选

其中 **class** 属性决定出发哪套行为控制

fileNamePattern 滚动后产生的文件名规则

maxHistory 日志保留时长，30天。超过天数后删除日志文件，同时配套目录一起删除。该参数不一定是指天数，也可以是月份数。具体参考滚动规则 **fileNamePattern**，看是依赖什么进行滚动的

totalSizeCap 最大日志量

(3) **triggeringPolicy** 滚动触发规则

maxFileSize 单个文件最大量，如果达到这个最大量。日志有可能会出现报错，新日志无法存入。

--注意：%cn：获取上下文名称 **contextName**。该用法是 **<encoder>** 标签的属性，只能在该标签使用。其他标签不能使用

rollingPolicy、**triggeringPolicy** 有不同的 **class** 实现类，具体每个实现类的用法及参数属性会有所不同。使用时注意区分

