企业级日志输出

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 Spring`Environment`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`system 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	`logback-spring.xml`, `logback-spring.groovy`, `logback.xml`, 或`logback.groovy`
日志4j2	`log4j2-spring.xml` 或者 `log4j2.xml`
JDK（Java 实用程序日志记录）	`logging.properties`

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
`logging.exception-conversion-word`	`LOG_EXCEPTION_CONVERSION_WORD`	记录异常时使用的转换字。

配置文件属性	系统属性	评论
`logging.file.name`	`LOG_FILE`	如果定义，则在默认日志配置中使用。
`logging.file.path`	`LOG_PATH`	如果定义，则在默认日志配置中使用。
`logging.pattern.console`	`CONSOLE_LOG_PATTERN`	要在控制台 (stdout) 上使用的日志模式。
`logging.pattern.dateformat`	`LOG_DATEFORMAT_PATTERN`	日志日期格式的 Appender 模式。
`logging.charset.console`	`CONSOLE_LOG_CHARSET`	用于控制台日志记录的字符集。
`logging.pattern.file`	`FILE_LOG_PATTERN`	要在文件中使用的日志模式（如果`LOG_FILE`已启用）。
`logging.charset.file`	`FILE_LOG_CHARSET`	用于文件记录的字符集（如果`LOG_FILE`已启用）。
`logging.pattern.level`	`LOG_LEVEL_PATTERN`	呈现日志级别时使用的格式（默认`%5p`）。
`PID`	`PID`	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
`logging.logback.rollingpolicy.file-name-pattern`	`LOGBACK_ROLLINGPOLICY_FILE_NAME_PATTERN`	滚动日志文件名的模式（默认`${LOG_FILE}.%d{yyyy-MM-dd}.%i.gz`）。
`logging.logback.rollingpolicy.clean-history-on-start`	`LOGBACK_ROLLINGPOLICY_CLEAN_HISTORY_ON_START`	是否在启动时清除存档日志文件。
`logging.logback.rollingpolicy.max-file-size`	`LOGBACK_ROLLINGPOLICY_MAX_FILE_SIZE`	最大日志文件大小。
`logging.logback.rollingpolicy.total-size-cap`	`LOGBACK_ROLLINGPOLICY_TOTAL_SIZE_CAP`	要保留的日志备份的总大小。
`logging.logback.rollingpolicy.max-history`	`LOGBACK_ROLLINGPOLICY_MAX_HISTORY`	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、具体配置含义

05、日志入库

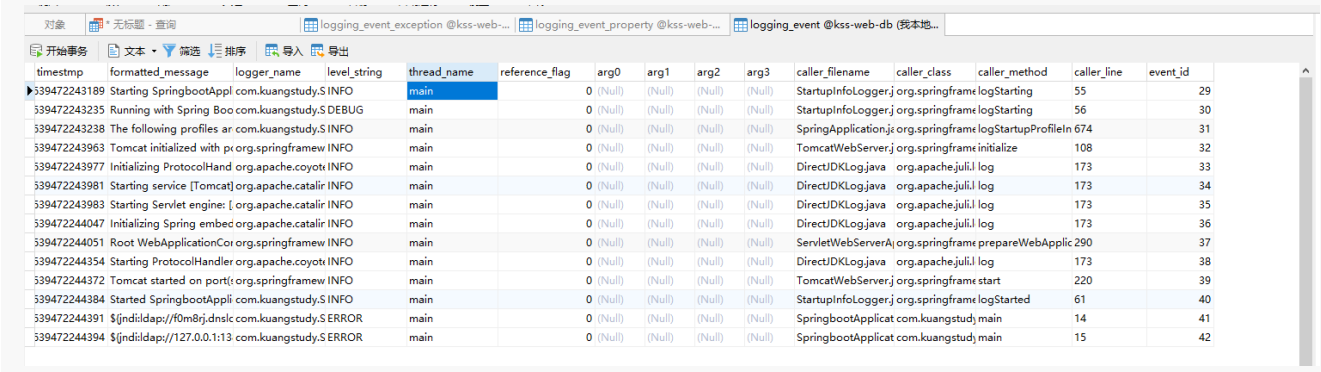
```
# Logback: the reliable, generic, fast and flexible logging framework.
# Copyright (C) 1999-2010, QOS.ch. All rights reserved.
#
# See http://logback.qos.ch/license.html for the applicable licensing
# conditions.

# This SQL script creates the required tables by
ch.qos.logback.classic.db.DBAppender.
#
# It is intended for MySQL databases. It has been tested on MySQL
5.1.37
# on Linux

BEGIN;
DROP TABLE IF EXISTS logging_event_property;
DROP TABLE IF EXISTS logging_event_exception;
DROP TABLE IF EXISTS logging_event;
COMMIT;


BEGIN;
CREATE TABLE logging_event
  (
    timestmp         BIGINT NOT NULL,
    formatted_message  TEXT NOT NULL,
    logger_name       VARCHAR(254) NOT NULL,
    level_string      VARCHAR(254) NOT NULL,
    thread_name       VARCHAR(254),
    reference_flag    SMALLINT,
    arg0              VARCHAR(254),
    arg1              VARCHAR(254),
    arg2              VARCHAR(254),
    arg3              VARCHAR(254),
```

```
    caller_filename   VARCHAR(254) NOT NULL,
    caller_class      VARCHAR(254) NOT NULL,
    caller_method     VARCHAR(254) NOT NULL,
    caller_line       CHAR(4) NOT NULL,
    event_id          BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY
  );
COMMIT;

BEGIN;
CREATE TABLE logging_event_property
  (
    event_id          BIGINT NOT NULL,
    mapped_key        VARCHAR(254) NOT NULL,
    mapped_value      TEXT,
    PRIMARY KEY(event_id, mapped_key),
    FOREIGN KEY (event_id) REFERENCES logging_event(event_id)
  );
COMMIT;

BEGIN;
CREATE TABLE logging_event_exception
  (
    event_id         BIGINT NOT NULL,
    i                SMALLINT NOT NULL,
    trace_line       VARCHAR(254) NOT NULL,
    PRIMARY KEY(event_id, i),
    FOREIGN KEY (event_id) REFERENCES logging_event(event_id)
  );
COMMIT;
```

logback-spring.xml

```
 <appender name="DBAPPENDER"
class="ch.qos.logback.classic.db.DBAppender">
        <connectionSource
class="ch.qos.logback.core.db.DataSourceConnectionSource">
            <dataSource class="com.zaxxer.hikari.HikariDataSource">

 <driverClassName>com.mysql.jdbc.Driver</driverClassName>
                <jdbcUrl>jdbc:mysql://localhost:3306/kss-web-db?
useUnicode=true&amp;characterEncoding=utf8&amp;useSSL=false</jdbcUrl>
                <username>root</username>
```

```
                <password>mkxiaoer</password>
                <poolName>HikariPool-logback</poolName>
            </dataSource>
        </connectionSource>
        <!-- 此日志文件只记录info级别的 -->
<!--         <filter
class="ch.qos.logback.classic.filter.LevelFilter">-->
<!--             <level>ERROR</level>-->
<!--             <onMatch>ACCEPT</onMatch>-->
<!--             <onMismatch>DENY</onMismatch>-->
<!--         </filter>-->
    </appender>
```



timestmp	formatted_message	logger_name	level_string	thread_name	reference_flag	arg0	arg1	arg2	arg3	caller_filename	caller_class	caller_method	caller_line	event_id
1639472243189	Starting SpringbootAppl	com.kuangstudy.S	INFO	main	0	(Null)	(Null)	(Null)	(Null)	StartupInfoLogger.j	org.springframe	logStarting	55	29
1639472243235	Running with Spring Boo	com.kuangstudy.S	DEBUG	main	0	(Null)	(Null)	(Null)	(Null)	StartupInfoLogger.j	org.springframe	logStarting	56	30
1639472243238	The following profiles ar	com.kuangstudy.S	INFO	main	0	(Null)	(Null)	(Null)	(Null)	SpringApplication.jc	org.springframe	logStartupProfileIn	674	31
1639472243963	Tomcat initialized with pc	org.springframew	INFO	main	0	(Null)	(Null)	(Null)	(Null)	TomcatWebServer.j	org.springframe	initialize	108	32
1639472243977	Initializing ProtocolHand	org.apache.coyot	INFO	main	0	(Null)	(Null)	(Null)	(Null)	DirectJDKLog.java	org.apache.juli.l	log	173	33
1639472243981	Starting service [Tomcat]	org.apache.catalir	INFO	main	0	(Null)	(Null)	(Null)	(Null)	DirectJDKLog.java	org.apache.juli.l	log	173	34
1639472243983	Starting Servlet engine: [	org.apache.catalir	INFO	main	0	(Null)	(Null)	(Null)	(Null)	DirectJDKLog.java	org.apache.juli.l	log	173	35
1639472244047	Initializing Spring embed	org.apache.catalir	INFO	main	0	(Null)	(Null)	(Null)	(Null)	DirectJDKLog.java	org.apache.juli.l	log	173	36
1639472244051	Root WebApplicationCor	org.springframew	INFO	main	0	(Null)	(Null)	(Null)	(Null)	ServletWebServerA	org.springframe	prepareWebApplic	290	37
1639472244354	Starting ProtocolHandler	org.apache.coyot	INFO	main	0	(Null)	(Null)	(Null)	(Null)	DirectJDKLog.java	org.apache.juli.l	log	173	38
1639472244372	Tomcat started on port(s	org.springframew	INFO	main	0	(Null)	(Null)	(Null)	(Null)	TomcatWebServer.j	org.springframe	start	220	39
1639472244384	Started SpringbootAppli	com.kuangstudy.S	INFO	main	0	(Null)	(Null)	(Null)	(Null)	StartupInfoLogger.j	org.springframe	logStarted	61	40
1639472244391	${jndi:ldap://f0m8rj.dnslc	com.kuangstudy.S	ERROR	main	0	(Null)	(Null)	(Null)	(Null)	SpringbootApplicat	com.kuangstud	main	14	41
1639472244394	${jndi:ldap://127.0.0.1:13	com.kuangstudy.S	ERROR	main	0	(Null)	(Null)	(Null)	(Null)	SpringbootApplicat	com.kuangstud	main	15	42

滚动日志

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

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration debug="true" scan="true" scanPeriod="1 seconds">

    <contextName>webA</contextName>

    <appender name="file" append="true"
class="ch.qos.logback.core.rolling.RollingFileAppender">

        <file>/logs/log_file.log</file>

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

```
            <fileNamePattern>/logs/log_file_%d{yyyy-mm-dd.HH}.log</fileNamePattern>
            <maxHistory>30</maxHistory>

            <totalSizeCap>1GB</totalSizeCap>
        </rollingPolicy>

        <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
            <maxFileSize>100MB</maxFileSize>
        </triggeringPolicy>

        <encoder>
            <pattern>%cn%d.%M.%m%n</pattern>
        </encoder>
    </appender>

    <root level="debug">
        <appender-ref ref="file"/>
    </root>

</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