**SpringBoot监控代理日志包**

目录

[1、说明 2](#_Toc61946408)

[2、全局错误码 2](#_Toc61946409)

[3、数据字典 2](#_Toc61946410)

[3.1、t\_job\_list 2](#_Toc61946411)

[3.2、t\_job\_logs 3](#_Toc61946412)

[3.3、t\_job\_runtime 3](#_Toc61946413)

[3.4、t\_service\_list 3](#_Toc61946414)

[3.6、t\_service\_logs 4](#_Toc61946415)

[3.5、t\_service\_runtime 4](#_Toc61946416)

[4、使用说明 5](#_Toc61946417)

[4.1、GET START 5](#_Toc61946418)

[4.2、配置项说明 6](#_Toc61946419)

[4.3、HTTP日志上报说明 7](#_Toc61946420)

[4.4、ACTION注解配置说明 8](#_Toc61946421)

[4.5、TASK注解配置说明 9](#_Toc61946422)

# 1、说明

[项目名称]：framework-logger。

包地址：<http://nexus.risenenergy.com:8081/repository/maven-releases/com/risen/framework-logger/1.0-RELEASE/framework-logger-1.01-RELEASE.jar>

[功能]：

1. Springboot日志代理。
2. 接口监控日志上报。
3. 定时任务监控日志上报

# 2、全局错误码

| **错误码** | **错误解释** |
| --- | --- |
| 4000 | 缺乏有效参数，日志消息必须提供[ReqId,ReqFrom]参数！ |

# 3、数据字典

## 3.1、t\_job\_list

* 定时任务信息表。

| **字段** | **类型** | **空** | **默认** | **注释** |
| --- | --- | --- | --- | --- |
| id | varchar(38) | 否 |  | 主键 |
| jobName | varchar(120) | 否 |  | Job名称 |
| jobInfo | varchar(200) | 是 |  | Job信息 |
| jobMethod | varchar(50) | 否 |  | Job方法 |
| jobClass | varchar(200) | 否 |  | Job类 |
| cron | varchar(20) | 是 |  | cron表达式 |
| fixedRate | bigint(20) | 是 |  | 任务间隔时间 |
| fixedDelay | bigint(20) | 是 |  | 任务轮询时间 |
| prevTime | datetime | 是 |  | 前一次执行时间 |
| nextTime | datetime | 是 |  | 下一次执行时间 |
| createTime | timestamp | 否 | CURRENT\_TIMESTAMP | 创建时间 |

* 备注：任务第一次执行的时候，自动完成注册。

定时任务注册信息，更新每个定时任务服务状态！

## 3.2、t\_job\_logs

* 定时任务日志表，记录定时任务执行详情。

| **字段** | **类型** | **空** | **默认** | **注释** |
| --- | --- | --- | --- | --- |
| id | int(32) | 否 |  | 主键 |
| taskId | varchar(38) | 否 |  | 任务ID |
| jobId | varchar(38) | 否 |  | Job ID |
| taskResult | int(11) | 是 |  | 运行结果 |
| errorMessage | text | 是 |  | 异常信息 |
| startTime | bigint(20) | 是 |  | 开始时间 |
| endTime | bigint(20) | 是 |  | 结束时间 |
| createTime | timestamp | 否 | CURRENT\_TIMESTAMP | 创建时间 |
| updateTime | timestamp | 是 |  | 更新时间 |

* 备注：根据日志LEVEL记录日志详情

记录日志每次执行的详细信息！

## 3.3、t\_job\_runtime

* JOB运行时表，任务运行情况记录。

| **字段** | **类型** | **空** | **默认** | **注释** |
| --- | --- | --- | --- | --- |
| id | int(11) | 否 |  | 主键 |
| taskId | varchar(38) | 否 |  | 任务ID |
| logDate | date | 否 |  | 日志日期 |
| scheduleTimes | int(11) | 是 |  | 调度次数 |
| errorTimes | int(11) | 是 |  | 异常次数 |
| firstTime | bigint(20) | 是 |  | 第一次执行时间 |
| lastTime | bigint(20) | 是 |  | 最后一次执行世家 |
| lastTask | varchar(38) | 是 |  | 最后一次任务ID |
| actionSize | int(11) | 是 |  | 当前任务 |
| bufferSize | int(11) | 是 |  | 缓存任务 |
| createTime | timestamp | 否 | CURRENT\_TIMESTAMP | 创建时间 |
| updateTime | timestamp | 否 |  | 更新世家 |

* 备注：以日为单位记录任务运行情况

任务运行情况记录！

## 3.4、t\_service\_list

* 服务注册表，接口服务信息注册

| **字段** | **类型** | **空** | **默认** | **注释** |
| --- | --- | --- | --- | --- |
| id | varchar(38) | 否 |  | 主键 |
| serviceName | varchar(50) | 否 |  | 服务名称 |
| serviceInfo | varchar(200) | 是 |  | 服务信息 |
| serviceMethod | varchar(50) | 否 |  | 服务方法 |
| serviceClass | varchar(200) | 否 |  | 服务类 |
| serviceUrl | varchar(200) | 否 |  | 服务地址 |
| latestTime | datetime | 是 |  | 最近访问时间 |
| createTime | timestamp | 否 | CURRENT\_TIMESTAMP | 创建时间 |

* 备注：服务第一次访问的时候自动完成注册

服务信息注册，记录所有服务列表！

## 3.6、t\_service\_logs

* 服务日志表，服务访问日志信息

| **字段** | **类型** | **空** | **默认** | **注释** |
| --- | --- | --- | --- | --- |
| id | int(11) | 否 |  | 主键 |
| actionId | varchar(38) | 否 |  | 执行ID |
| reqId | varchar(50) | 否 |  | 请求ID |
| reqFrom | varchar(50) | 否 |  | 来源方 |
| fromIp | varchar(50) | 否 |  | 来源IP |
| reqTime | bigint(20) | 否 |  | 请求时间 |
| serviceId | varchar(38) | 否 |  | 服务ID |
| queryString | varchar(200) | 是 |  | 查询参数 |
| actionResult | int(11) | 是 |  | 服务结果 |
| errorMessage | text | 是 |  | 异常信息 |
| startTime | bigint(20) | 是 |  | 开始时间 |
| endTime | bigint(20) | 是 |  | 结束时间 |
| createTime | timestamp | 否 | CURRENT\_TIMESTAMP | 创建时间 |
| updateTime | timestamp | 否 |  | 更新时间 |

* 备注：根据日志LEVEL记录服务请求情况详情。

记录日志请求详情！

## 3.5、t\_service\_runtime

* 服务运行时表，接口访问详情及异常记录

| **字段** | **类型** | **空** | **默认** | **注释** |
| --- | --- | --- | --- | --- |
| id | int(11) | 否 |  |  |
| reqFrom | varchar(38) | 否 |  | 来源方ID |
| fromIp | varchar(50) | 否 |  | 来源IP |
| logDate | date | 否 |  | 日志日期 |
| serviceId | varchar(38) | 否 | 0 | 服务名称 |
| requestTimes | int(11) | 是 | 0 | 请求次数 |
| errorTimes | int(11) | 是 | 0 | 失败次数 |
| firstTime | bigint(20) | 是 | 0 | 开始访问时间 |
| lastTime | bigint(20) | 是 | 0 | 最后访问时间 |
| lastAction | varchar(38) | 是 | 0 | 最后访问actionId |
| actionSize | int(11) | 是 |  | 当前执行 |
| bufferSize | int(11) | 是 |  | 缓存执行 |
| createTime | timestamp | 否 | CURRENT\_TIMESTAMP | 创建时间 |
| updateTime | timestamp | 否 |  | 更新时间 |

* 备注：一以天为单位记录接口访问情况

记录每天接口访问情况以及失败次数，具体接口访问详情以及错误信息保存在t\_action\_runtime表！

# 4、使用说明

## 4.1、GET START

##### 1. 添加依赖

包地址：<http://nexus.risenenergy.com:8081/repository/maven-releases/com/risen/framework-logger/1.0-RELEASE/framework-logger-1.01-RELEASE.jar>

|  |
| --- |
| <dependency>  <groupId>com.risen</groupId>  <artifactId>framework-logger</artifactId>  <version>1.01-RELEASE</version>  </dependency> |

##### 2. 增加包扫描（com.risen.logger）

|  |
| --- |
| @SpringBootApplication(scanBasePackages= {"com.risen.dingtalk","com.risen.logger"})  public class DingtalkServiceApplication {  public static void main(String[] args) {  SpringApplication.run(DingtalkServiceApplication.class, args);  }  } |

##### 3. 增加日志配置

|  |
| --- |
| **framework**:  **logger**:  *#OFF(0),INFO(1),ERROR(2),ALL(3);* **level**: INFO  **validate-request**: false  **only-annotation**: false  **datasource**:  **driver-class-name**: com.mysql.cj.jdbc.Driver  **jdbc-url**: jdbc:mysql://10.10.8.60:3306/risen\_base?useUnicode=true&characterEncoding=utf8&serverTimezone=GMT%2B8  **username**: finereport  **password**: Risen\*1337  **http**:  **service-url**: http://srv.risenenergy.com:8780/msg/api/action/log |

##### 4. 运行

## 4.2、配置项说明

##### 简要描述

* 用于配置实现日志的记录或着上报。日志通过SpringBoot AOP实现。 默认情况下会扫描如下定义的所有ACTION请求，并且记录日志。 execution(public \* com.risen..controller..\*Controller.\*(..)) 也可以如下注解方式灵活实现 @ActionLog(name="日志测试",info = "日志接口测试")

##### 配置项

| **配置项** | **可选值** | **说明** |
| --- | --- | --- |
| framework.logger.validate-request | TRUE,FALSE | 是否验证请求 |
| framework.logger.only-annotation | TRUE,FALSE | 是否仅注解 |
| framework.logger.level | #OFF(0),INFO(1),ERROR(2),ALL(3) | 日志等级 |
| framework.logger.datasource | 定义日志数据库 | 数据源 |
| - datasource.driver-class-name |  | 数据库驱动类 |
| - datasource.jdbc-url |  | 数据库链接URL |
| - datasource.username |  | 用户名 |
| - datasource.password |  | 密码 |
| framework.logger.http.service-url | 定义日志上报地址 | 上报接口,与database二选一。 |

##### 说明 :

###### framework.logger.validate-request：

**false(默认):** 不验证请求

**true：**所有请求必须带【ReqId,ReqFrom】参数。

###### framework.logger.only-annotation：

**false(默认):** 所有符合扫描要求或着注解的ACTION都会触发日志。

**true：**只有注解的方法才会触发日志。

###### framework.logger.level：

###### OFF(0):关闭日志。

###### INFO(1)：只记录运行情况。不记录详细日志。

###### ERROR(2)：记录错误信息。

###### ALL(3)：记录每次详细信息。

###### framework.logger.datasource：

日志记录数据库配置,与正常的jdbc数据源配置一致（建议mysql）。 **driver-class-name：**数据库驱动类 **jdbc-url：**数据库链接URL **username：**用户名 **password：**密码

###### framework.logger.http.servic-url：

日志上报接口，将会HTTP的方式向该定义接口推送日志消息。

## 4.3、HTTP日志上报说明

#### framework.logger.http.service-url：

日志上报接口，将会HTTP的方式向该定义接口推送日志消息。

HTTP消息将包含如下消息：

##### HEAD参数

| **参数名** | **必选** | **类型** | **说明** |
| --- | --- | --- | --- |
| ReqId | 是 | string | 请求ID,不重复UUID |
| ReqFrom | 是 | string | 来源系统ID |
| ReqTime | 是 | integer | 日志等级 |

##### 参数

| **参数名** | **必选** | **类型** | **说明** |
| --- | --- | --- | --- |
| actionLogType | 是 | string | 日志类型 |
| message | 是 | string | 消息内容 |
| logLevel | 是 | string | 日志等级 |

###### actionLogType： 日志消息类型

**Before：**ACTION前置消息 **After：**ACTION前置消息 **AfterRunning：**ACTION返回消息 **AfterThrowing：**ACTION异常消息

正常情况一个接口调用下会根据AOP状态发起3个消息，前置，(返回|异常)二选一，后置。

**Before：**前置消息，在方法执行之前发起 **After：**后置消息，在方法执行后发起 **AfterRunning：**返回消息，在方法返回结果之后发起 **AfterThrowing：**异常消息，在方法抛出异常发起

###### logLevel：日志等级（可通过@ActionLog注解提供设置）

INFO 只记录访问次数，以及异常情况。 **2：**ERROR 会记录访问异常详情。 **3：**ALL 记录每一次访问详情。

###### message：消息体

|  |
| --- |
| public class ActionModel {   private String actionId;  private String serviceId;  private String serviceName ;  private String reqId;  private String reqFrom;  private String fromIp;  private Long reqTime;  private String serviceMethod;  private String serviceClass;  private String serviceUrl;  private String queryString;  private Long actionTime;  private String serviceInfo;  private Integer actionResult;  private String errorMessage;  private String logLevel;  ……  } |

## 4.4、ACTION注解配置说明

##### 简要描述

* 用于配置实现日志的记录或着上报。日志通过SpringBoot AOP实现。 默认情况下会扫描如下定义的所有ACTION请求，并且记录日志。 execution(public \* com.risen..controller..\*Controller.\*(..)) 也可以如下注解方式灵活实现 @ActionLog(name="日志测试",info = "日志接口测试")

##### 配置项目说明：

| **名称** | **类型** | **说明** |
| --- | --- | --- |
| actionName | string | 接口名称 |
| validate | boolean | 校验等级 |
| value | LogLevel | 日志等级 |
| actionInfo | string | 接口描述 |

###### validate

**false(默认):**

**true：**所有请求必须带【ReqId,ReqFrom】参数。

与配置项功能一致，优先级大于配置项。

###### logLevel：日志等级

**OFF：**NOLOG 不记录日志。

**INFO：**INFO只记录访问次数，以及异常情况。

**ERROR：**ERROR 会记录访问异常详情。

**ALL：**ALL 记录每一次访问详情。

## 4.5、TASK注解配置说明

##### 简要描述

* 用于配置实现日志的记录或着上报。日志通过SpringBoot AOP实现。 默认情况下会扫描如下定义的所有@Scheduled请求，并且记录日志。可以如下注解方式灵活实现 @TaskLog(taskName = "测试定时任务" ,taskInfo = "测试定时任务")

##### 配置项目说明：

| **名称** | **类型** | **说明** |
| --- | --- | --- |
| taskName | string | 任务名称 |
| value | LogLevel | 日志等级 |
| taskInfo | string | 任务描述 |

###### logLevel：日志等级

**OFF：**NOLOG 不记录日志。

**INFO：**INFO只记录访问次数，以及异常情况。

**ERROR：**ERROR 会记录访问异常详情。

**ALL：**ALL 记录每一次访问详情。