定时任务配置表：task\_job\_cron

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **类型** | **不为空** | **注释** |
| id | int(11) | True |  |
| job\_name | varchar(50) | True | 任务名称  注：任务分组，任务名称 字符串的拼接值是唯一的  例如：已经插入了 一条记录 的 jobName 和 jobGroup 为 （job1,group1）,那么下次就无法再将 （job1,group1）插入， 而（job2,group1）或（job1,group2）却是允许插入的 |
| job\_group | varchar(50) | True | 任务分组，事件任务： event；定时任务：cron； |
| job\_ch\_name | varchar(50) | True | 任务中文名称 |
| job\_file | varchar(200) | True | 任务文件路径 |
| job\_class | varchar(200) | True | 任务载体（0：串行载体；1：并行载体；） |
| entity\_class | varchar(200) | True | 任务类  值为 全类名(完整包路径.类名)  例如： com.gszh.HelloJob |
| cron\_expression | varchar(20) | True | cron 时间表达式，例如：0/5 \* \* \* \* ? 表示每5秒执行一次 |
| priority | int(11) | True | 任务优先权限，默认为5，两个任务在同一时间触发，值越大越优先执行，1<=priority<=10 |
| misfire | int(11) | True | 错失触发时间后执行的方案，默认为0；  0：withMisfireHandlingInstructionDoNothing  ——不触发立即执行  ——等待下次Cron触发频率到达时刻开始按照Cron频率依次执行  1：withMisfireHandlingInstructionIgnoreMisfires  ——以错过的第一个频率时间立刻开始执行  ——重做错过的所有频率周期后  ——当下一次触发频率发生时间大于当前时间后，再按照正常的Cron频率依次执行  2：withMisfireHandlingInstructionFireAndProceed  ——以当前时间为触发频率立刻触发一次执行  ——然后按照Cron频率依次执行 |
| if\_boot | int(11) | True | 是否开机启动，0：非开机启动；1：开机启动 |
| is\_rely\_on | int(11) | True | 是否存在任务依赖，0：不存在；1：存在 |
| rely\_wait\_time | int(11) | False | 任务依赖等待时间，超时则取消等待 |
| rely\_on | varchar(500) | True | 任务一级依赖；  书写规范：任务分组.任务名称;  样例：group1.job1;group2.job2; |
| start\_time | timestamp | False | 任务启动时间（为空时，则默认为每次程序的启动时间） |
| end\_time | timestamp | False | 任务结束时间（为null时，默认为不限制结束时间） |
| job\_type | varchar(50) | False | 任务类别 |
| description | varchar(200) | False | 任务描述 |
| create\_time | timestamp | False |  |
| update\_time | timestamp | False |  |
| username | varchar(50) | False | 帐号 |
| password | varchar(50) | False | 密码 |
| time | datetime(0) | False | 参数-时间 |
| time\_type | varchar(0) | False | 参数-时间类型：年月日时分 |

任务参数表：task\_job\_param

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **类型** | **不为空** | **注释** |
| id | int(11) | True |  |
| job\_name | varchar(50) | True | 任务名称 |
| job\_group | varchar(50) | True | 任务分组 |
| param\_name | varchar(200) | True | 参数名称 |
| param\_value | varchar(200) | True | 参数值 |
| param\_desc | varchar(200) | False | 参数含义 |
| create\_time | timestamp | False |  |
| update\_time | timestamp | False |  |

任务状态表：task\_job\_state

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **类型** | **不为空** | **注释** |
| id | int(11) | True |  |
| instance\_no | varchar(50) | True | 实例编号 |
| job\_name | varchar(50) | True | 任务名称  注：任务分组，任务名称 字符串的拼接值是唯一的 |
| job\_group | varchar(50) | True | 任务分组  注：任务分组，任务名称 字符串的拼接值是唯一的 |
| fire\_time\_long | bigint(20) | True | 计划的触发时间 |
| job\_state | varchar(50) | True | 任务状态：  执行，等待，等待超时，异常，暂停，恢复，销毁或完成 |
| record\_time | timestamp | True | 记录时间 |

任务状态变更历史表：task\_job\_state\_history

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **类型** | **不为空** | **注释** |
| id | int(11) | True |  |
| instance\_no | varchar(50) | True | 实例编号 |
| job\_name | varchar(50) | True | 任务名称  注：任务分组，任务名称 字符串的拼接值是唯一的 |
| job\_group | varchar(50) | True | 任务分组  注：任务分组，任务名称 字符串的拼接值是唯一的 |
| fire\_time\_long | bigint(20) | True | 计划的触发时间 |
| job\_state | varchar(50) | True | 任务状态：  执行，等待，等待超时，异常，暂停，恢复，销毁或完成 |
| record\_time | timestamp | True | 记录时间 |

数据库为下划线命名规则，项目为驼峰命名规则

如：job\_name ->jobName

1.定时任务配置：

在数据库直接对任务配置进行操作，只会在重新请求后才能生效

操作方法： 添加、启动、暂停、恢复、停止

1. 添加

请求：<http://localhost:8080/addOne>

请求类型：GET

请求参数：TaskJobCron po

1. 启动

请求：<http://localhost:8080/startOne>

请求类型：GET

请求参数：String jobName, String jobGroup（任务名称，任务分组）

1. 暂停

请求：

<http://localhost:8080/pauseOne>

请求类型：GET

请求参数：String jobName, String jobGroup（任务名称，任务分组）

1. 恢复

请求：

<http://localhost:8080/resumeOne>

[请求类型：GET](http://localhost:8080/resumeOne)

[请求参数：String jobName, String jobGroup](http://localhost:8080/resumeOne)（任务名称，任务分组）

1. 删除

请求：<http://localhost:8080/deleteOne>

请求类型：GET

请求参数：String jobName, String jobGroup（任务名称，任务分组）

1. 修改

请求：[http://localhost:8080/updateOne](http://localhost:8080/addOne)

请求类型：GET

请求参数：TaskJobCron po

1. 重跑

请求：<http://localhost:8080/restartInstance>

请求类型：GET

请求参数：String instanceNo,Date fireTime（实例名称，触发时间）

2. 依赖任务配置：

限制：仅定时任务存在任务依赖

任务依赖，需要设置的字段为

|  |  |  |  |
| --- | --- | --- | --- |
| is\_rely\_on | int(11) | True | 是否存在任务依赖，0：不存在；1：存在 |
| rely\_wait\_time | int(11) | False | 任务依赖等待时间，超时则取消等待 |
| rely\_on | varchar(500) | True | 任务一级依赖；  书写规范：任务分组.任务名称;  多个依赖：group1.job1;group2.job2; |

3.事件任务配置：

事件任务为http 请求触发，仅执行一次

常驻任务，可以直接通过事件任务设置，其停止方法通过下文中的实例配置实现

暂停、恢复、停止 可以通过对实例的操作实现

操作：启动、添加、修改、删除

1. 启动

请求：

<http://localhost:8080/startEventJob>

请求类型：GET

请求参数：String jobName（任务名称）

1. 添加

请求：

<http://localhost:8080/addEventJob>

请求类型：GET

请求参数：TaskJobEvent po

1. 修改

请求：

<http://localhost:8080/updateEventJob>

请求类型：GET

请求参数：String jobName（任务名称）

1. 删除

请求：

<http://localhost:8080/deleteEventJob>

请求类型：GET

请求参数：String jobName（任务名称）

任务实例配置：

操作：暂停、恢复、停止

1. 暂停

请求：

<http://localhost:8080/deleteEventJob>

请求类型：GET

请求参数：String instanceNo（实例编号）

1. 恢复

请求：

<http://localhost:8080/deleteEventJob>

请求类型：GET

请求参数：String instanceNo（实例编号）

1. 停止

请求：

<http://localhost:8080/deleteEventJob>

请求类型：GET

请求参数：String instanceNo（实例编号）

实例状态记录

记录实例的 执行，等待，等待超时，异常，暂停，恢复，销毁或完成