分布式数据采集平台

2017年6月5号

目录

[第一章 综述 1](#_Toc25435)

[1.1 系统概述 1](#_Toc11623)

[1.2 系统模块结构图 1](#_Toc3246)

[1.3 系统物理结构图 2](#_Toc12227)

[1.4 系统配置 2](#_Toc22581)

[1.5 配置一个采集任务说明 3](#_Toc27773)

[1.6 系统任务执行流程图 4](#_Toc32446)

[1.6.1 执行任务流程图1 4](#_Toc13024)

[1.6.2 执行任务流程图2 5](#_Toc3383)

[1.6.3 执行任务流程图3 6](#_Toc13234)

[1.6.4 执行任务worker流程图 8](#_Toc4785)

[第二章 模块设计 9](#_Toc26003)

[2.1 系统元数据访问层 9](#_Toc10174)

[2.1.1 BasePo实体类 9](#_Toc20859)

[2.1.2 网页站点 9](#_Toc24951)

[2.1.2.1 数据库表设计 9](#_Toc12312)

[2.1.2.2 Java PO 对象 10](#_Toc5608)

[2.1.2.3 数据访问层接口设计 10](#_Toc18192)

[2.1.3 网页站点页面元素抽取路径 11](#_Toc17596)

[2.1.3.1 数据库表设计 11](#_Toc11646)

[2.1.3.2 Java PO 对象 12](#_Toc20055)

[2.1.3.3 数据访问层接口设计 12](#_Toc6430)

[2.1.4 网页站点页面下载后的缓存 14](#_Toc13511)

[2.1.4.1 数据库表设计 14](#_Toc8703)

[2.1.4.2 Java PO 对象 14](#_Toc15665)

[2.1.4.3 数据访问层接口设计 15](#_Toc6332)

[2.1.5 http代理数据 15](#_Toc15778)

[2.1.5.1 数据库表设计 16](#_Toc11149)

[2.1.5.2 Java PO 对象 16](#_Toc619)

[2.1.5.3 数据访问层接口设计 16](#_Toc14497)

[2.1.6 任务数据 17](#_Toc20347)

[2.1.6.1 数据库表设计 17](#_Toc24749)

[2.1.6.2 Java PO 对象 18](#_Toc25928)

[2.1.6.3 数据访问层接口设计 19](#_Toc17044)

[2.1.7 任务参数数据 20](#_Toc28234)

[2.1.7.1 数据库表设计 20](#_Toc20872)

[2.1.7.2 Java PO 对象 20](#_Toc8476)

[2.1.7.3 数据访问层接口设计 21](#_Toc291)

[2.1.8 任务触发关系数据 22](#_Toc1204)

[2.1.8.1 数据库表设计 22](#_Toc7999)

[2.1.8.2 Java PO 对象 22](#_Toc28502)

[2.1.8.3 数据访问层接口设计 23](#_Toc12696)

[2.1.9 任务执行记录数据 23](#_Toc24111)

[2.1.9.1 数据库表设计 23](#_Toc7552)

[2.1.9.2 Java PO 对象 24](#_Toc29373)

[2.1.9.3 数据访问层接口设计 25](#_Toc21033)

[2.1.10 任务worker执行记录数据 26](#_Toc3176)

[2.1.10.1 数据库表设计 26](#_Toc12887)

[2.1.10.2 Java PO 对象 27](#_Toc26618)

[2.1.10.3 数据访问层接口设计 27](#_Toc8303)

[2.1.11 任务worker执行异常数据 28](#_Toc15330)

[2.1.11.1 数据库表设计 28](#_Toc26694)

[2.1.11.2 Java PO 对象 29](#_Toc16215)

[2.1.11.3 数据访问层接口设计 29](#_Toc23578)

[2.2 节点资源管理 31](#_Toc15327)

[2.2.1 初始化 31](#_Toc26263)

[2.2.2 API说明 31](#_Toc966)

[2.3 调度管理 31](#_Toc8674)

[2.3.1 任务调度管理 31](#_Toc15747)

[2.3.1.1 初始化 31](#_Toc17393)

[2.3.1.2 API说明 31](#_Toc13093)

[2.3.2 worker调度管理 31](#_Toc29377)

[2.3.2.1 初始化 31](#_Toc3639)

[2.3.2.2 API说明 31](#_Toc26410)

[2.3.3 worker类说明 39](#_Toc29377)

[2.3.3.1 API说明 39](#_Toc13093)

[2.4 Admin后台管理 42](#_Toc27584)

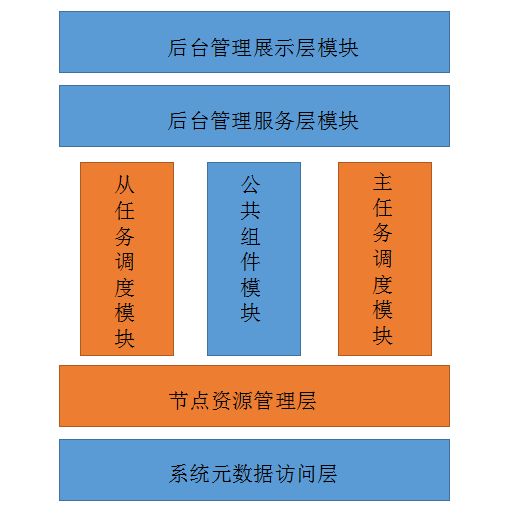
[2.4.1 Api说明 42](#_Toc15199)

# 综述

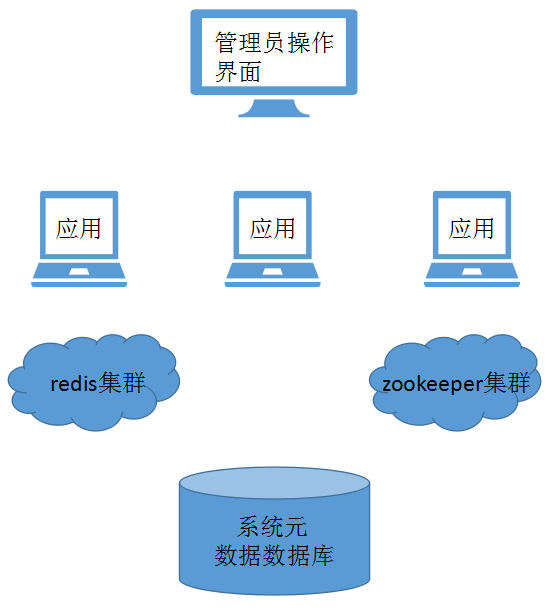
## 系统概述

本系统采用java开发,依赖mysql,redis,zookeeper等中间件服务，主要为用户提供指定网站业务数据采集并提供采集任务监控等功能。解决采集过程中遇到用户登录,ip频率限制，js内容加载,验证码识别等防爬策略。

## 系统模块结构图



## 系统物理结构图



## 系统配置

####后台管理页面登录端口

server.port=8081

http.mappers.json-pretty-print=false

http.mappers.json-sort-keys=false

####静态页面配置路径

spring.view.prefix: /static/html/

spring.view.suffix: .html

##系统mysql配置

jdbc.driver=com.mysql.jdbc.Driver

jdbc.url=jdbc:mysql://172.18.88.44:3306/test?user=root&password=123456&useUnicode=true&characterEncoding=UTF8&useSSL=false

jdbc.username=root

jdbc.password=123456

## 集群名称

spider.cluster.name=crawler\_cluster

## 开启集群yes 或者no(单机模式)

spider.cluster.enable=no

## 节点名称

spider.node.name=test

## 节点ip

spider.node.host=172.18.84.62

## 节点rpc通信端口

spider.node.trafficPort=8180

## 系统job定时cronTrigger表达式

spider.master.systemjob.cronTrigger=0 0 1 \* \* ? \*

## 清理系统7天历史记录数据

spider.master.systemjob.clearBeforeDays=7

## 节点上最大运行的worker数量

spider.worker.running.max.size=10

##reids 集群连接串

spider.redis.host=172.18.88.44:6379;172.18.88.45:6379;172.18.88.46:6379

spider.redis.timeout=1000

##系统通知邮件配置

spider.email.host=spider.email.post=

spider.email.user=

spider.email.pwd=

spider.email.admin=

##测试http代理的url

spider.httpProxy.test.url=http://www.baidu.com

##zookeeper集群连接串

spider.zookeeper.host=172.18.88.44:2181,172.18.88.45:2181,172.18.88.46:2181

## 配置一个采集任务说明

配置步骤如下:

1. 首先需要配置采集目标网站信息，对应ex\_crawler\_platform\_site 表。
2. 配置采集目标网站页面元素jsoup select(cssQuery)路径，对应ex\_crawler\_platform\_extract\_path表。
3. 配置采集任务基本信息,对应ex\_crawler\_platform\_job表。
4. 配置采集任务参数信息，对应ex\_crawler\_platform\_job\_param表。
5. 配置采集任务抽取项信息，对应ex\_crawler\_platform\_extract\_item表。

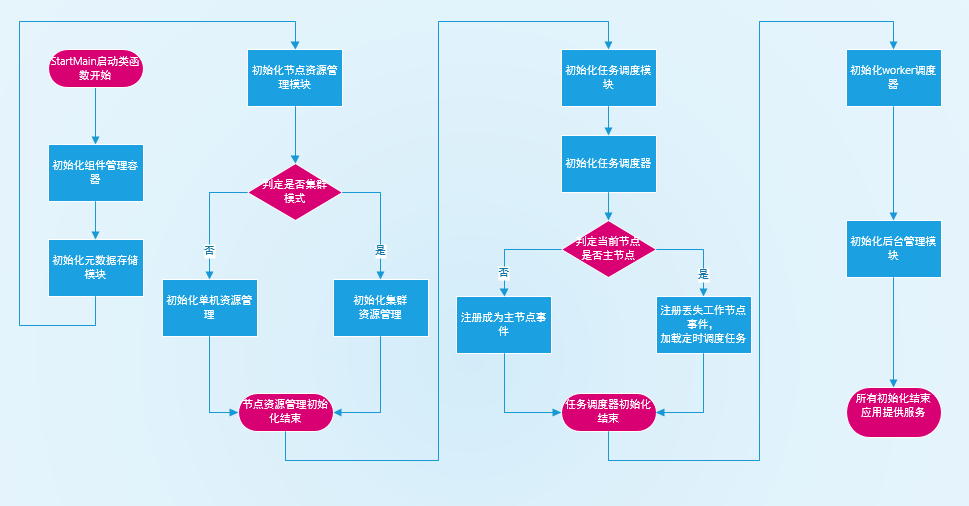
通过以上5点步骤配置后,可通过如下3种方式执行:

1. 通过管理页面点击运行任务。
2. 设置好定时时间开启定时调度后自动触发执行任务。
3. 通过任务关系触发(ex\_crawler\_platform\_job\_relationship表)触发执行。

## 系统任务执行流程图

### **系统启动流程图**

从StartMain.main函数开始:

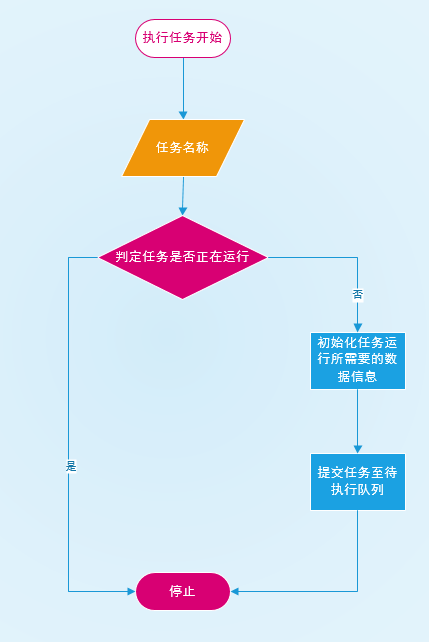


### **执行任务流程图1**

从MasterSchedulerManager.execute函数开始

主节点调度器执行任务，这里任务触发由3种方式:

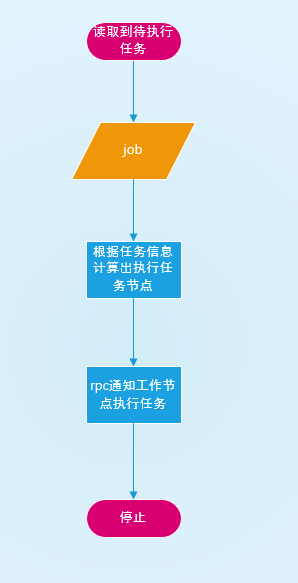
1. 管理页面手动触发
2. 定时触发器触发
3. 任务关系触发



### **执行任务流程图2**

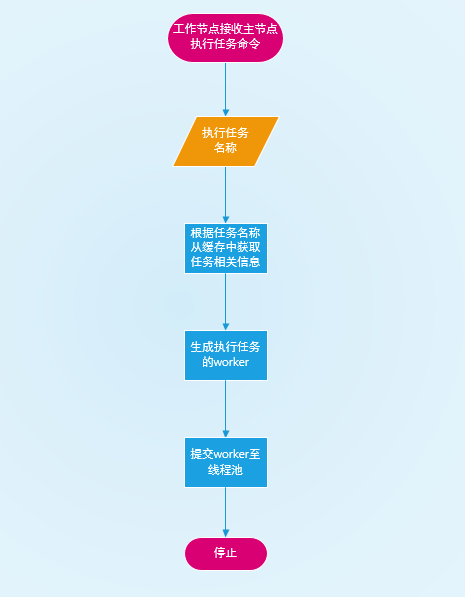
MasterSchedulerManager.doExecute函数开始

主调度器有单独的读取待执行任务队列的线程专门负责处理任务



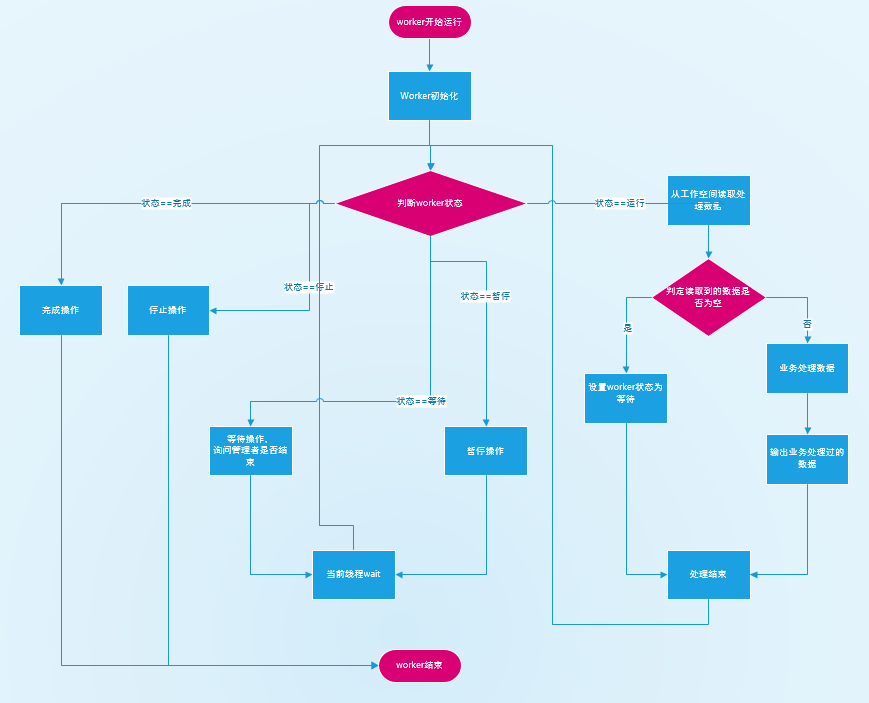
### **执行任务流程图3**

WorkerSchedulerManager.execute函数开始



### **执行任务worker流程图**

从AbstractWorker.start函数开始



# 模块设计

## **系统元数据访问层**

采用mysql进行存储,主要用保存系统所需要的元数据。对上层应用提供增删查改等功能接口。

所有po对象需要继承BasePo类

### **BasePo实体类**

对象名称: BasePo

概述: 实现java 序列化(Serializable)接口,用于作分页查询和更新数据时版本号

| **字段** | **类型** | **说明** |
| --- | --- | --- |
| totalSize | int | 分页查询时，统计查询到的数据总数量 |
| version | int | 数据版本 |

### **网页站点**

用来描述指定网络站点的相关信息。

#### **数据库表设计**

概述: 用来保存网站(site)相关信息

表名: ex\_crawler\_platform\_site

| **字段** | **类型** | **主键** | **外键** | **非空** | **说明** |
| --- | --- | --- | --- | --- | --- |
| code | VARCHAR(20) | 是 | 是 | 是 | 站点编号,字段唯一，用来表示每个网站的主键 |
| mainurl | VARCHAR(100) | 否 | 否 | 是 | 站点主url |
| visitInterval | INT(11) | 否 | 否 | 否 | 站点访问频率，用来保证任务采集时使用http代理时每个代理ip休息时间 |
| describe | VARCHAR(200) | 否 | 否 | 否 | 描述 |
| version | INT(11) | 否 | 否 | 是 | 数据版本 |

#### **Java PO 对象**

对象名称: Site

概述: 必须继承 BasePo

| **字段** | **类型** | **对应数据库表字段** | **说明** |
| --- | --- | --- | --- |
| code | String | code | 站点编号,字段唯一，用来表示每个网站的主键 |
| mainurl | String | mainurl | 站点主url |
| visitInterval | long | visitInterval | 站点访问频率，用来保证任务采集时使用http代理时每个代理ip休息时间 |
| describe | String | describe | 数据描述 |
| totalSize | int |  | 分页查询时，统计查询到的数据总数量 |
| version | int | version | 数据版本 |

#### 数据访问层接口设计

接口名称: SiteDao

接口方法:

/\*\*

\* 分页查询

\* **@param** pageIndex

\* **@param** pageSize

\* **@return** 返回分页后的数据集合

\*/

**public** List<Site> pageQuery(**int** pageIndex, **int** pageSize);

/\*\*

\* 根据指定code查询

\* **@param** sitecode

\* **@return** 返回相等code的数据

\*/

**public** Site query(String code);

/\*\*

\* 保存指定site

\* **@param** site

\* **@return** 返回受影响数据条数

\*/

**public** **int** save(Site site);

/\*\*

\* 删除指定code的站点数据

\* **@param** sitecode

\* **@return** 返回受影响数据条数

\*/

**public** **int** del(String sitecode);

### 网页站点页面元素抽取路径

用来描述指定网络站点某种类型页面上的指定元素抽取取路径

#### **数据库表设计**

概述: 用来保存网站页面元素抽取路径相关信息

表名: ex\_crawler\_platform\_site\_extract\_path

| **字段** | **类型** | **主键** | **外键** | **非空** | **说明** |
| --- | --- | --- | --- | --- | --- |
| name | VARCHAR(50) | 是 | 是 | 是 | 页面元素抽取路径名称 |
| siteCode | VARCHAR(100) | 是 | 是 | 是 | 页面元素抽取路径所属站点 |
| ranking | INT(11) | 是 | 否 | 是 | 页面元素抽取路径排名 |
| path | VARCHAR(300) | 否 | 否 | 否 | 页面元素抽取路径 |
| tableHeadPath | VARCHAR(300) | 否 | 否 | 否 | 表格多条数据类型 列名path |
| tableDataPath | VARCHAR(300) | 否 | 否 | 否 | 表格多条数据类型 数据path |
| filterPath | VARCHAR(300) | 否 | 否 | 否 | 需要过滤的 path |
| extractAttName | VARCHAR(50) | 否 | 否 | 是 | 获取值得 属性名字 |
| substringStart | VARCHAR(50) | 否 | 否 | 否 | 字符串截取开始 |
| substringEnd | VARCHAR(50) | 否 | 否 | 否 | 字符串截取结束 |
| compareAttName | VARCHAR(50) | 否 | 否 | 否 | 需要比较的元素属性名称 |
| containKeyWord | VARCHAR(50) | 否 | 否 | 否 | 需要比较的元素属性值 |
| replaceWord | VARCHAR(50) | 否 | 否 | 否 | 需要替换的字符 |
| replaceValue | VARCHAR(50) | 否 | 否 | 否 | 需要被替换的字符 |
| appendHead | VARCHAR(50) | 否 | 否 | 否 | 向头部追加字符 |
| appendEnd | VARCHAR(50) | 否 | 否 | 否 | 向尾部追加字符 |
| extractEmptyCount | INT(11) | 否 | 否 | 是 | 路径空抽取次数 |
| describe | VARCHAR(200) | 否 | 否 | 否 | 描述 |
| version | INT(11) | 否 | 否 | 是 | 数据版本 |

#### **Java PO 对象**

对象名称: ExtractPath

概述: 必须继承 BasePo

| **字段** | **类型** | **对应数据库表字段** | **说明** |
| --- | --- | --- | --- |
| name | String | name | 页面元素抽取路径名称 |
| siteCode | String | siteCode | 页面元素抽取路径所属站点 |
| ranking | String | ranking | 页面元素抽取路径排名 |
| path | String | path | 页面元素抽取路径 |
| tableHeadPath | String | tableHeadPath | 表格多条数据类型 列名path |
| tableDataPath | String | tableDataPath | 表格多条数据类型 数据path |
| filterPath | String | filterPath | 需要过滤的 path |
| extractAttName | String | extractAttName | 获取值得 属性名字 |
| substringStart | String | substringStart | 字符串截取开始 |
| substringEnd | String | substringEnd | 字符串截取结束 |
| compareAttName | String | compareAttName | 需要比较的元素属性名称 |
| containKeyWord | String | containKeyWord | 需要比较的元素属性值 |
| replaceWord | String | replaceWord | 需要替换的字符 |
| replaceValue | String | replaceValue | 需要被替换的字符 |
| appendHead | String | appendHead | 向头部追加字符 |
| appendEnd | String | appendEnd | 向尾部追加字符 |
| extractEmptyCount | int | extractEmptyCount | 路径空抽取次数 |
| describe | String | describe | 描述 |
| totalSize | int |  | 分页查询时，统计查询到的数据总数量 |
| version | int | version | 数据版本 |

#### 数据访问层接口设计

接口名称: ExtractPathDao

接口方法:

/\*\*

\* 通过指定sitecode查询相关的元素抽取路径集合

\* **@param** siteCode 网站编号

\* **@return** 返回匹配到的数据集合

\*/

**public** List<ExtractPath> queryBySite(String siteCode);

/\*\*

\* 通过指定siteCode和name查询相关的元素抽取路径集合

\* **@param** siteCode 网站编号

\* **@param** name 元素抽取路径名称

\* **@return** 返回匹配到的数据集合

\*/

**public** List<ExtractPath> queryBySiteAndName(String siteCode,String name);

/\*\*

\* 通过指定siteCode和name模糊查询相关的元素抽取路径集合

\*

\* **@param** siteCode 网站编号

\* **@param** name 元素抽取路径名称

\* **@return** 返回匹配到的数据集合

\*/

**public** List<ExtractPath> fuzzyQuery(String siteCode,String pathName);

/\*\*

\* 批量保存元素抽取路径集合

\* **@param** paserResults 需要批量保存的元素抽取路径集合

\* **@return** 插入的数据条数

\*/

**public** **int** batchSave(List<ExtractPath> paserResults);

/\*\*

\* 通过指定名字删除元素抽取路径

\* **@param** name 元素抽取路径名称

\* **@return** 删除的数据条数

\*/

**public** **int** delByName(String name);

/\*\*

\* 通过指定sitecode删除元素抽取路径

\* **@param** siteCode 网站编号

\* **@return** 删除的数据条数

\*/

**public** **int** delBySiteCode(String siteCode);

### 网页站点页面下载后的缓存

概述: 用来描述采集任务下载的页面后的缓存

#### **数据库表设计**

概述: 用来保存网站页面元素抽取路径相关信息

表名: ex\_crawler\_platform\_site\_page

| **字段** | **类型** | **主键** | **外键** | **非空** | **说明** |
| --- | --- | --- | --- | --- | --- |
| jobName | VARCHAR(50) | 是 | 是 | 是 | 页面所属得运行job名称 |
| jobSnapshotId | VARCHAR(50) | 是 | 是 | 是 | 页面所属得运行job记录id |
| siteCode | VARCHAR(20) | 是 | 是 | 是 | 页面所属站点唯一标识 |
| pageKey | VARCHAR(32) | 是 | 否 | 是 | 页面key(由url+请求参数md5) |
| pageUrl | VARCHAR(500) | 否 | 否 | 是 | 页面url(由url+请求参数拼接) |
| pageSrc | MEDIUMTEXT | 否 | 否 | 否 | 页面源码 |
| data | MEDIUMBLOB | 否 | 否 | 否 | Page对象序列化byte[] |
| updateTime | TIMESTAMP | 否 | 否 | 是 | 数据更新时间 |
| version | INT(11) | 否 | 否 | 是 | 数据版本 |

#### **Java PO 对象**

对象名称: PagePo

概述: 必须继承 BasePo

| **字段** | **类型** | **对应数据库表字段** | **说明** |
| --- | --- | --- | --- |
| jobName | String | jobName | 页面所属得运行job名称 |
| jobSnapshotId | String | jobSnapshotId | 页面所属得运行job记录id |
| siteCode | String | siteCode | 网站唯一标识 |
| pageKey | String | pageKey | 网站页面唯一key |
| pageUrl | String | pageUrl | 网站页面url表示 |
| pageSrc | String | pageSrc | 网站页面源码 |
| data | byte[] | data | 网站页面实体对象序列化后byte数组 |
| totalSize | int |  | 分页查询时，统计查询到的数据总数量 |
| version | int | version | 数据版本 |

#### 数据访问层接口设计

接口名称: PageDao

接口方法:

/\*\*

\* 通过指定siteCode和pageKey查询，返回最新的一条数据

\* **@param** siteCode

\* **@param** pageKey

\* **@return**

\*/

**public** PagePo queryBySiteAndKey(String siteCode,String pageKey);

/\*\*

\* 保存PagePo

\* **@param** page

\* **@return** 返回保存的数据条数

\*/

**public** **int** save(PagePo pagePo);

/\*\*

\* 删除指定多少天以前的数据

\* **@param** beforeDays

\* 多少天以前

\* **@return** 返回删除掉的数据条数

\*/

**public** **int** delBeforeDate(**int** beforeDays);

### http代理数据

用来描述http 代理信息

#### **数据库表设计**

概述: 保存采集任务使用的http代理

表名: ex\_crawler\_platform\_http\_proxy

| **字段** | **类型** | **主键** | **外键** | **非空** | **说明** |
| --- | --- | --- | --- | --- | --- |
| host | VARCHAR(100) | 是 | 否 | 是 | http代理host |
| port | INT(11) | 是 | 否 | 是 | http代理端口 |
| type | INT(11) | 否 | 否 | 否 | http代理类型 |
| userName | VARCHAR(100) | 否 | 否 | 否 | http代理用户 |
| passWord | VARCHAR(100) | 否 | 否 | 否 | http代理密码 |
| expire | INT(11) | 否 | 否 | 否 | http代理有效期 |
| describe | VARCHAR(200) | 否 | 否 | 否 | http代理描述 |
| version | INT(11) | 否 | 否 | 是 | 数据版本 |

#### **Java PO 对象**

对象名称: HttpProxy

概述: 必须继承 BasePo

| **字段** | **类型** | **对应数据库表字段** | **说明** |
| --- | --- | --- | --- |
| host | String | host | http代理host |
| port | int | port | http代理post |
| type | int | type | http代理类型 |
| userName | String | userName | http代理用户 |
| passWord | String | passWord | http代理密码 |
| expire | date | expire | http代理有效期 |
| describe | String | describe | http代理描述 |
| totalSize | int |  | 分页查询时，统计查询到的数据总数量 |
| version | int | version | 数据版本 |

#### 数据访问层接口设计

接口名称: HttpProxyDao

接口方法:

/\*\*

\* 获取所有代理

\* **@return** 返回所有数据集合

\*/

List<HttpProxy> getAll();

/\*\*

\* 保存指定代理

\* **@param** httpProxy

\* **@return** 返回保存的数据条数

\*/

**int** save(HttpProxy httpProxy);

/\*\*

\* 通过指定host和post删除代理

\* **@param** host http代理主机

\* **@param** port http代理端口

\* **@return** 返回删除掉的数据条数

\*/

**int** del(String host, **int** port);

/\*\*

\* 删除所有代理

\* **@return** 返回删除掉的数据条数

\*/

**int** delAll();

### 任务数据

#### **数据库表设计**

概述: 保存任务相关信息

表名: ex\_crawler\_platform\_job

| **字段** | **类型** | **主键** | **外键** | **非空** | **说明** |
| --- | --- | --- | --- | --- | --- |
| name | VARCHAR(100) | 是 | 否 | 是 | 任务名称，主键，必须唯一 |
| level | INT(1) | 否 | 否 | 是 | 任务级别 |
| designatedNode | VARCHAR(100) | 否 | 否 | 否 | 任务运行指定节点 |
| neeadNodes | INT(11) | 否 | 否 | 是 | 任务运行需要节点数 |
| needThreads | INT(11) | 否 | 否 | 是 | 任务运行需要线程数 |
| isScheduled | INT(1) | 否 | 否 | 是 | 任务开启定时调度 |
| cronTrigger | VARCHAR(100) | 否 | 否 | 否 | 任务定时调度时间表达式 |
| workFrequency | INT(11) | 否 | 否 | 否 | 任务工作频率 |
| workClass | VARCHAR(100) | 否 | 否 | 是 | 任务插件java class |
| workSpaceName | VARCHAR(100) | 否 | 否 | 是 | 任务工作空间名 |
| user | VARCHAR(100) | 否 | 否 | 否 | 任务所属用户 |
| describe | VARCHAR(300) | 否 | 否 | 否 | 任务描述 |
| version | INT(1) | 否 | 否 | 是 | 数据版本 |

#### **Java PO 对象**

对象名称: Job

概述: 必须继承 BasePo

| **字段** | **类型** | **对应数据库表字段** | **说明** |
| --- | --- | --- | --- |
| name | String | name | 任务名称,唯一标识 |
| level | int | level | 任务级别 |
| designatedNode | String | designatedNode | 任务运行指定节点名 |
| neeadNodes | int | neeadNodes | 任务运行所需节点数 |
| needThreads | int | needThreads | 任务运行所需线程数 |
| isScheduled | int | isScheduled | 任务开启定时调度 |
| cronTrigger | String | cronTrigger | 任务定时调度时间表达式 |
| workFrequency | long | workFrequency | 任务每次工作频率 |
| workClass | String | workClass | 任务插件java class |
| workSpaceName | String | workSpaceName | 任务工作空间名 |
| user | String | user | 任务所属用户 |
| describe | String | describe | 任务描述 |
| totalSize | int |  | 分页查询时，统计查询到的数据总数量 |
| version | int | version | 数据版本 |

#### 数据访问层接口设计

接口名称: JobDao

接口方法:

/\*\*

\* 通过指定任务名称查询

\*

\* **@param** name 任务名称

\*

\* **@return** 查询到的数据实体

\*/

Job query(String name);

/\*\*

\* 通过指定任务name进行模糊分页查询

\*

\* **@param** name任务名称前缀

\* **@param** pageIndex分页起始位置

\* **@param** pageSize分页偏移位置

\* **@return** 返回查询到的数据集合

\*/

List<Job> pageQuery(String name, **int** pageIndex,**int** pageSize);

/\*\*

\* 查询开启定时调度的任务

\* **@return** 返回查询到开启定时调度的任务集合

\*/

List<Job> queryIsScheduled();

/\*\*

\* 保存指定任务数据

\* **@param** job任务实体对象

\* **@return** 保存的数据条数

\*/

**int** save(Job job);

/\*\*

\* 通过指定任务名称删除数据

\* **@param** name 任务名称

\* **@return** 返回删除掉的数据条数

\*/

**int** del(String name);

/\*\*

\* 根据指定任务名称版本数据更新 任务是否开启定时调度

\* **@param** version 当前数据版本

\* **@param** newVersion数据更新后的版本号

\* **@param** name任务名称

\* **@param** isScheduled是否开启定时调度

\* **@return** 返回更新到数据条数

\*/

**int** updateIsScheduled(**int** version,**int** newVersion, String name,**int** isScheduled);

/\*\*

\* 根据指定任务名称版本数据更新任务时间调度表达式

\* **@param** version当前数据版本

\* **@param** newVersion数据更新后的版本号

\* **@param** name任务名称

\* **@param** cronTrigger时间调度表达式

\* **@return** 返回更新到数据条数

\*/

**int** updateCronTrigger(**int** version, **int** newVersion,String name, String cronTrigger);

### 任务参数数据

#### **数据库表设计**

概述: 保存任务相关信息

表名: ex\_crawler\_platform\_job\_param

| **字段** | **类型** | **主键** | **外键** | **非空** | **说明** |
| --- | --- | --- | --- | --- | --- |
| jobName | VARCHAR(100) | 是 | 是 | 是 | 任务名称 |
| name | VARCHAR(50) | 是 | 否 | 是 | 任务参数名称 |
| value | TEXT | 否 | 否 | 是 | 任务参数值 |
| version | INT(11) | 否 | 否 | 是 | 数据版本 |

#### **Java PO 对象**

对象名称: JobParam

概述: 必须继承 BasePo

| **字段** | **类型** | **对应数据库表字段** | **说明** |
| --- | --- | --- | --- |
| jobName | String | jobName | 任务名称 |
| name | String | name | 任务参数名称 |
| value | String | value | 任务参数值 |
| totalSize | int |  | 分页查询时，统计查询到的数据总数量 |
| version | int | version | 数据版本 |

#### 数据访问层接口设计

接口名称: JobParamDao

接口方法:

/\*\*

\* 通过指定任务名称查询任务参数

\* **@param** jobName 任务名称

\* **@return** 返回查询到的任务参数集合

\*/

List<JobParam> queryByJob(String jobName);

/\*\*

\* 批量保存任务参数集合

\* **@param** jobParams 任务参数集合

\* **@return** 返回保存的数据条数

\*/

**int** batchSave(List<JobParam> jobParams);

/\*\*

\* 根据任务参数id和版本更新 任务参数值

\* **@param** version 当前数据版本

\* **@param** newVersion 更新后的数据版本

\* **@param** id 任务参数id

\* **@param** name 任务参数名称

\* **@param** value 任务参数值

\* **@return** 返回更新到的数据条数

\*/

**int** update(**nt** version,**int** newVersion,String id,String name,String value);

/\*\*

\* 通过指定任务参数id删除数据

\* **@param** id 任务参数id

\* **@return** 返回删除掉的数据条数

\*/

**int** del(String id);

/\*\*

\* 通过指定任务名称删除其任务参数

\* **@param** jobName 任务名称

\* **@return** 返回删除掉的数据条数

\*/

**int** delByJob(String jobName);

### 任务触发关系数据

#### **数据库表设计**

概述: 保存任务相关信息

表名: ex\_crawler\_platform\_job\_relationship

| **字段** | **类型** | **主键** | **外键** | **非空** | **说明** |
| --- | --- | --- | --- | --- | --- |
| currentJobName | VARCHAR(100) | 是 | 是 | 是 | 当前任务名称 |
| nextJobName | VARCHAR(100) | 是 | 是 | 是 | 被触发任务名称 |
| executeType | INT(1) | 否 | 否 | 是 | 执行类型 |
| version | INT(1) | 否 | 否 | 是 | 数据版本 |

#### **Java PO 对象**

对象名称: JobRelationship

概述: 必须继承 BasePo

| **字段** | **类型** | **对应数据库表字段** | **说明** |
| --- | --- | --- | --- |
| currentJobName | String | currentJobName | 当前任务 |
| nextJobName | String | nextJobName | 下个触发任务 |
| executeType | int | executeType | 执行类型 |
| totalSize | int |  | 分页查询时，统计查询到的数据总数量 |
| version | int | version | 数据版本 |

#### 数据访问层接口设计

接口名称: JobRelationshipDao

接口方法:

/\*\*

\* 通过jobName 查询

\* **@param** id nextJobName

\* **@return**

\*/

**public** List<JobRelationship> query(String jobName);

### 任务执行记录数据

#### **数据库表设计**

概述: 统计任务每次执行信息相关信息

表名: ex\_crawler\_platform\_job\_snapshot

| **字段** | **类型** | **主键** | **外键** | **非空** | **说明** |
| --- | --- | --- | --- | --- | --- |
| jobName | VARCHAR(100) | 是 | 是 | 是 | 任务名称 |
| executeTimestamp | VARCHAR(30) | 是 | 是 | 是 | 任务执行时间戳 |
| startTime | DATETIME | 否 | 否 | 是 | 任务执行开始时间 |
| endTime | DATETIME | 否 | 否 | 是 | 任务执行结束时间 |
| Status | String | 否 | 否 | 是 | 任务工作空间名称 |
| triggerType | String | 否 | 否 | 是 | 任务被触发类型 |
| totalProcessCount | INT(11) | 否 | 否 | 是 | 统计处理次数 |
| totalResultCount | INT(11) | 否 | 否 | 是 | 统计结果数量 |
| totalProcessTime | INT(11) | 否 | 否 | 是 | 统计处理时间 |
| avgProcessTime | INT(11) | 否 | 否 | 是 | 统计平均处理时间 |
| maxProcessTime | INT(11) | 否 | 否 | 是 | 统计最大处理时间 |
| minProcessTime | INT(11) | 否 | 否 | 是 | 统计最小处理时间 |
| errCount | INT(11) | 否 | 否 | 是 | 异常次数 |
| runtimeParams | String | 否 | 否 | 否 | 运行时参数 |
| version | INT(11) | 否 | 否 | 是 | 数据版本 |

#### **Java PO 对象**

对象名称: JobSnapshot

概述: 必须继承 BasePo

| **字段** | **类型** | **对应数据库表字段** | **说明** |
| --- | --- | --- | --- |
| jobName | String | jobName | 任务名称 |
| executeTimestamp | String | executeTimestamp | 任务执行时间戳 |
| startTime | String | startTime | 任务执行开始时间 |
| endTime | String | endTime | 任务执行结束时间 |
| status | int | status | 任务工作空间名称 |
| triggerType | String | triggerType | 任务被触发类型 |
| totalProcessCount | int | totalProcessCount | 统计处理次数 |
| totalResultCount | int | totalResultCount | 统计结果数量 |
| totalProcessTime | int | totalProcessTime | 统计处理时间 |
| avgProcessTime | int | avgProcessTime | 统计平均处理时间 |
| maxProcessTime | int | maxProcessTime | 统计最大处理时间 |
| minProcessTime | int | minProcessTime | 统计最小处理时间 |
| errCount | int | errCount | 异常次数 |
| runtimeParams | String | runtimeParams | 运行时参数 |
| version | int | version | 数据版本 |

#### 数据访问层接口设计

接口名称: JobSnapshotDao

接口方法:

/\*\*

\* 通过 jobSnapshotId 和 jobName 查询 任务运行记录

\* **@param** jobSnapshotId 任务运行批次

\* **@param** jobName 任务名称

\* **@return** 查询到的任务运行记录

\*/

JobSnapshot query(String jobSnapshotId,String jobName);

/\*\*

\* 通过任务名称查询 任务运行记录 的集合

\* **@param** jobName 任务名称

\* **@return** 查询到的任务的运行记录 集合

\*/

List<JobSnapshot> queryByJob(String jobName);

/\*\*

\* 通过任务名称分页查询任务运行记录集合

\* **@param** jobName 任务名称

\* **@param** pageIndex 分页查询起始位置

\* **@param** pageSize 分页查询偏移位置

\* **@return** 查询到的任务的运行记录 集合

\*/

List<JobSnapshot> pageQuery(String jobName, **int** pageIndex, **int** pageSize);

/\*\*

\* 通过任务名称查询 任务的最后一条运行记录

\* **@param** jobName 任务名称

\* **@return** 最后一条运行记录

\*/

JobSnapshot queryLastEnd(String jobName,String excludeId);

/\*\*

\* 保存任务运行记录

\* **@param** jobSnapshot

\* **@return** 受影响数据的条数

\*/

**int** save(JobSnapshot jobSnapshot);

/\*\*

\* 更新任务运行记录

\* **@param** jobSnapshot

\* **@return** 受影响数据的条数

\*/

**int** update(JobSnapshot jobSnapshot);

/\*\*

\* 更新任务运行记录的状态

\* **@param** id

\* **@param** status

\* **@return** 受影响数据的条数

\*/

**int** updateStatus(**int** version,**int** newVersion,String id,**int** status);

/\*\*

\* 删除多少天以前的数据

\* **@param** beforeDays

\* **@return** 受影响数据的条数

\*/

**int** delBeforeDate(**int** beforeDays);

### 任务worker执行记录数据

#### **数据库表设计**

概述: 统计任务worker运行相关信息

表名: ex\_crawler\_platform\_job\_worker\_snapshot

| **字段** | **类型** | **主键** | **外键** | **非空** | **说明** |
| --- | --- | --- | --- | --- | --- |
| jobName | VARCHAR(100) | 是 | 是 | 是 | 任务名称 |
| executeTimestamp | VARCHAR(30) | 是 | 是 | 是 | 任务执行时间戳 |
| name | VARCHAR(200) | 是 | 否 | 是 | Worker名称 |
| localNode | VARCHAR(50) | 否 | 否 | 是 | Worker运行所在的节点名 |
| startTime | DATETIME | 否 | 否 | 是 | Worker执行开始时间 |
| endTime | DATETIME | 否 | 否 | 是 | Worker执行结束时间 |
| totalProcessCount | INT(11) | 否 | 否 | 是 | 统计处理次数 |
| totalResultCount | INT(11) | 否 | 否 | 是 | 统计结果数量 |
| totalProcessTime | INT(11) | 否 | 否 | 是 | 统计处理时间 |
| avgProcessTime | INT(11) | 否 | 否 | 是 | 统计平均处理时间 |
| maxProcessTime | INT(11) | 否 | 否 | 是 | 统计最大处理时间 |
| minProcessTime | INT(11) | 否 | 否 | 是 | 统计最小处理时间 |
| errCount | INT(11) | 否 | 否 | 是 | 异常次数 |
| version | INT(11) | 否 | 否 | 是 | 数据版本 |

#### **Java PO 对象**

对象名称: WorkerSnapshot

概述: 必须继承 BasePo

| **字段** | **类型** | **对应数据库表字段** | **说明** |
| --- | --- | --- | --- |
| executeTimestamp | String | executeTimestamp | 任务执行时间戳 |
| jobName | String | jobName | 任务名称 |
| name | String | name | Worker名称 |
| localNode | String | localNode | Worker运行所在的节点名 |
| startTime | String | startTime | Worker执行开始时间 |
| endTime | String | endTime | Worker执行结束时间 |
| totalProcessCount | int | totalProcessCount | 统计处理次数 |
| totalResultCount | int | totalResultCount | 统计结果数量 |
| totalProcessTime | int | totalProcessTime | 统计处理时间 |
| avgProcessTime | int | avgProcessTime | 统计平均处理时间 |
| maxProcessTime | int | maxProcessTime | 统计最大处理时间 |
| minProcessTime | int | minProcessTime | 统计最小处理时间 |
| errCount | int | errCount | 异常次数 |
| version | int | version | 数据版本 |

#### 数据访问层接口设计

接口名称: WorkerSnapshotDao

接口方法:

/\*\*

\* 通过任务名称和任务运行批次(jobSnapshotId)查询

\* **@param** jobName 任务名称

\* **@param** jobSnapshotId 任务运行批次

\* **@return** 匹配到的数据集合

\*/

**public** List<WorkerSnapshot> query(String jobName,String jobSnapshotId);

/\*\*

\* 保存 worker 运行记录

\* **@param** workerSnapshot worker 运行记录

\* **@return** 受影响行数

\*/

**public** **int** save(WorkerSnapshot workerSnapshot);

/\*\*

\* 保存 worker 运行记录集合

\* **@param** workerSnapshots

\*/

**public** **void** batchSave(List<WorkerSnapshot> workerSnapshots);

/\*\*

\* 更新 worker 运行记录

\* **@param** workerSnapshot worker 运行记录

\* **@return** 受影响行数

\*/

**public** **int** update(WorkerSnapshot workerSnapshot);

/\*\*

\* 删除多少天以前的数据

\* **@param** beforeDays

\* **@return** 受影响的行数

\*/

**public** **int** delBeforeDate(**int** beforeDays);

### 任务worker执行异常数据

#### **数据库表设计**

概述: 统计任务worker运行相关信息

表名: ex\_crawler\_platform\_job\_worker\_err

| **字段** | **类型** | **主键** | **外键** | **非空** | **说明** |
| --- | --- | --- | --- | --- | --- |
| id | INT(11) | 是 | 是 | 是 | 自增长id |
| jobName | VARCHAR(200) | 否 | 是 | 是 | 任务名称 |
| jobSnapshotId | VARCHAR(200) | 否 | 是 | 是 | 任务执行时批次 |
| workerName | VARCHAR(200) | 否 | 是 | 是 | Worker名称 |
| startTime | DATETIME | 否 | 否 | 是 | 异常时间 |
| type | VARCHAR(100) | 否 | 否 | 是 | 异常类型 |
| msg | TEXT | 否 | 否 | 是 | 异常信息 |
| version | INT(11) | 否 | 否 | 是 | 数据版本 |

#### **Java PO 对象**

对象名称: WorkeErrMsg

概述: 必须继承 BasePo

| **字段** | **类型** | **对应数据库表字段** | **说明** |
| --- | --- | --- | --- |
| id | String | id | 自增长id |
| jobName | String | jobName | 任务名称 |
| executeTimestamp | String | executeTimestamp | 任务执行时批次 |
| workerName | String | workerName | Worker名称 |
| startTime | String | startTime | 异常时间 |
| type | String | type | 异常类型 |
| msg | int | msg | 异常信息 |
| version | int | version | 数据版本 |

#### 数据访问层接口设计

接口名称: WorkeErrMsgDao

接口方法:

/\*\*

\* 通过任务名称，任务执行批次分页查询异常信息

\* **@param** jobName 任务名称

\* **@param** jobSnapshotId 任务执行批次

\* **@param** start 分页起始位置

\* **@param** end 分页偏移位置

\* **@return** 匹配到的数据集合

\*/

List<WorkerErrMsg> pageQuery(String jobName,String jobSnapshotId,

**int** start,**int** end);

/\*\*

\* 通过任务名称，任务执行批次查询异常信息

\* **@param** jobName 任务名称

\* **@param** jobSnapshotId 任务执行批次

\* **@return** 匹配到的数据集合

\*/

List<WorkerErrMsg> queryByJob(String jobName,String jobSnapshotId);

/\*\*

\* 保存worker运行异常信息

\* **@param** workerErrMsg worker运行异常信息

\* **@return** 受影响行数

\*/

**public** **int** save(WorkerErrMsg workerErrMsg);

/\*\*

\* 批量保存worker运行异常信息

\* **@param** list worker运行异常信息集合

\* **@return** 受影响行数

\*/

**public** **int** batchSave(List<WorkerErrMsg> list);

/\*\*

\* 删除多少天以前的数据

\* **@param** beforeDays

\* **@return**

\*/

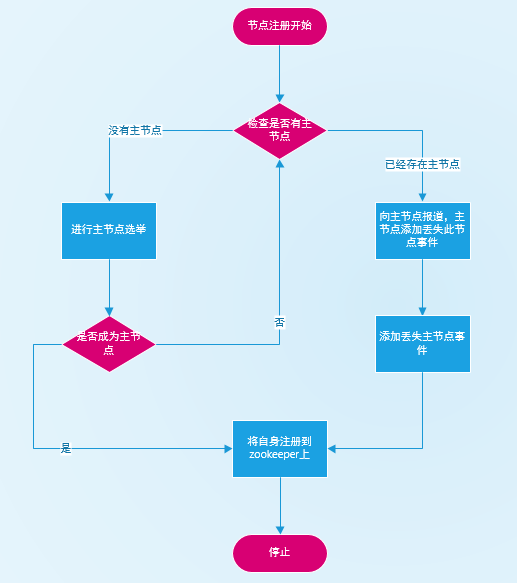
**public** **int** delBeforeDate(**int** beforeDays);

## **节点资源管理**

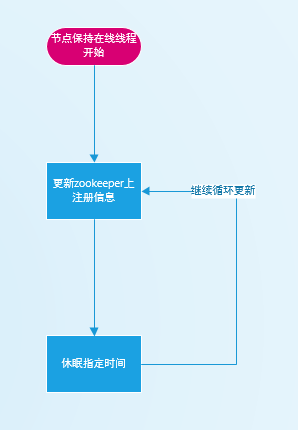
主要用来维护节点注册，节点心跳，节点master选举，节点通信，节点变换事件，节点资源管理等功能。

### 初始化

初始化流程图：



节点keeplive(程序会有单独的一个守护线程执行此流程)流程图:



### API说明

/\*\*

\* 获取当前集群名

\* **@return** 当前集群名称

\*/

String getClusterName();

/\*\*

\* 获取当前节点名

\* **@return** 当前节点名称

\*/

String getNodeName();

/\*\*

\* 获取当前节点

\* **@return** 当前节点

\*/

Node getCurrentNode();

/\*\*

\* 获取当前集群主节点

\*

\* **@return**

\*/

Node getMaster();

/\*\*

\* 获取当前集群所有工作节点

\* **@return**

\*/

List<Node> getWorkerNodes();

/\*\*

\* 通过节点名获取指定工作节点

\* **@param** nodeName

\* **@return**

\*/

Node getWorkerNode(String nodeName);

/\*\*

\* 通过needFresNodes获取空闲节点集合

\* **@param** needFresNodes

\* **@return**

\*/

List<Node> getFreeWorkerNodes(**int** needFresNodes);

/\*\*

\* 获取指定节点的最新节点信息

\* **@param** targetNode

\* **@return**

\*/

Node getNewestNode(Node targetNode);

/\*\*

\* 向master节点添加工作节点

\* **@param** workerNode

\*/

**void** addWorkerNode(Node workerNode);

/\*\*

\* master节点移除工作节点

\* **@param** workerNodeName

\*/

**void** removeWorkerNode(String workerNodeName);

/\*\*

\* newMaster成为新master节点

\* **@param** master

\*/

**void** masterChange(Node newMaster);

/\*\*

\* 注册节点变化watcher

\* **@param** watcher

\*/

**void** registerToMasterNodeWatcher(NodeChangeWatcher watcher);

/\*\*

\* 注册节点变化watcher

\* **@param** watcher

\*/

**void** registerMissWorkerNodeWatcher(NodeChangeWatcher watcher);

/\*\*

\* 获取指定node上的指定异步服务(clz),支持成功调用回调

\* **@param** node 指定节点

\* **@param** clz 服务class

\* **@param** asyCallback 回调操作

\* **@return** 返回一个服务实例

\*/

<T> T loolup(Node node, Class<T> clz, AsyCallback asyCallback);

/\*\*

\* 获取指定node上的指定同步服务(clz)

\* **@param** node 指定节点

\* **@param** clz 服务class

\* **@return** 返回一个服务实例

\*/

<T> T loolup(Node node, Class<T> clz);

/\*\*

\* 注册指定protocol rpc 服务

\* **@param** protocol

\* **@param** tagetOb

\*/

**void** registerNodeService(Class<?> protocol, Object tagetOb);

/\*\*

\* 根据path 获取一个 分布式读锁

\* **@param** path

\* **@return**

\*/

DistributedLock getDistributedLock(String path);

## **调度管理**

任务概念:用户配置定义的一个业务任务。

Worker概念:一个业务任务将会由n个worker并发执行。

由任务调度管理和worker调度管理2块组成,每个节点上都会有一个job调度管理和worker调度管理。

注意：

1. 当当前节点为master节点时job调度管理处于工作状态,worker调度管理处于休眠状态。
2. 当当前节点为worker节点时job调度管理处于休眠状态,worker调度管理处于工作状态。
3. Job调度管理与worker调度管理间，两者提供相应相互远程调用接口。

### 任务调度管理

负责任务级别的调度管理，协调各工作节点上的job调度，控制任务的执行流程

#### 初始化

1. 注册自身提供给worker调度的rpc服务
2. 检查当前节点是否为单机节点或者主节点
3. 如果是修复缓存信息，初始化定时调度器，加载系统任务和需要定时的用户任务
4. 如果不是添加当前节点成为主节点事件

#### API说明

/\*\*

\* triggerType 触发类型 执行指定任务

\* **@param** triggerType 触发类型1:手动触发。2.定时触发。3.任务关系触发

\* **@param** jobName指定任务

\*/

**void** execute(TriggerType triggerType, String jobName);

/\*\*

\* 暂停指定任务

\* **@param** jobName指定任务

\* **@return**

\*/

**void** suspend(String jobName);

/\*\*

\* 继续执行指定任务

\* **@param** jobName指定任务

\* **@return**

\*/

**void** goOn(String jobName);

/\*\*

\* 接收到指定任务下的woker结束 此方法调用由worker调度管理发起

\* **@param** jobName指定任务

\* **@param** wrokerNameworker名称

\*/

**void** receiveWorkerEnd(String jobName, String wrokerName);

/\*\*

\* 接收到指定任务下的woker询问是否结束 此方法调用由worker调度管理发起

\* **@param** jobName指定任务

\* **@param** wrokerName worker名称

\*/

**void** receiveWorkerAskIsEnd(String jobName, String wrokerName);

/\*\*

\* 停止指定任务

\* **@param** jobName指定任务

\* **@return**

\*/

**void** stop(String jobName);

/\*\*

\* 停止所有任务

\* **@return**

\*/

**void** stopAll();

/\*\*

\* 通过指定节点获取相应的worker调度远程服务实例

\* **@param** workerNode

\* **@return**

\*/

AbstractWorkerSchedulerManager getWorkerSchedulerManager(Node workerNode);

/\*\*

\* 向定时调度器 调度指定job

\* **@param** job

\*/

**boolean** scheduled(Job job);

/\*\*

\* 取消调度指定任务

\* **@param** jobName

\*/

**boolean** unScheduled(String jobName);

### worker调度管理

负责worker级别的调度管理，协调当前工作节点上的worker调度，控制worker的执行流程

#### 初始化

1. 注册自身提供给任务调度的rpc服务
2. 初始化worker线程池

#### API说明

/\*\*

\* 当前工作节点执行指定任务,由主节点或者单机节点上的JobSchedulerManager调用

\* **@param** jobName指定任务

\*/

**void** execute(String jobName);

/\*\*

\* 通知当前工作节点指定任务的worker暂停

\* **@param** jobName 指定任务

\* **@return**

\*/

**void** suspend(String jobName);

/\*\*

\* 通知当前工作节点指定任务的worker休息下

\* **@param** jobName 指定任务

\*/

**void** rest(String jobName);

/\*\*

\* 通知当前工作节点指定任务的worker继续

\* **@param** jobName 指定任务

\* **@return**

\*/

**void** goOn(String jobName);

/\*\*

\* 接收到指定任务下的woker结束 此方法调用由worker调度管理发起

\* **@param** jobName 指定任务

\* **@param** wrokerName worker名称

\*/

**void** workerEnd(String jobName, String wrokerName);

/\*\*

\* 接收到指定任务下的woker询问是否结束 此方法调用由worker调度管理发起

\* **@param** jobName 指定任务

\* **@param** wrokerName worker名称

\*/

**void** workerAskIsEnd(String jobName, String wrokerName);

/\*\*

\* 通知当前工作节点指定任务的worker停止

\* **@param** jobName 指定任务

\* **@return**

\*/

**void** stop(String jobName);

/\*\*

\* 通知当前工作节点指定任务的worker完成

\* **@param** jobName

\*/

**void** finish(String jobName);

/\*\*

\* 停止所有任务

\* **@return**

\*/

**void** stopAll();

/\*\*

\* 通过指定节点获取相应的worker调度远程服务实例

\* **@param** workerNode

\* **@return**

\*/

JobSchedulerManager getMasterJobSchedulerManager();

### Worker 类说明

一个任务由n个worker同时执行此任务。

#### API说明

/\*\*

\* 绑定环境配置

\* **@param** configure

\*/

**void** bindConfigure(SpiderConfigure configure);

/\*\*

\* 绑定worker管理者

\* **@param** manager

\*/

**void** bindManager(WorkerSchedulerManagerImpl manager);

/\*\*

\* 绑定worker job

\* **@param** job

\*/

**void** bindJob(Job job);

/\*\*

\* 绑定worker 运行快照

\* **@param** workerSnapshot

\*/

**void** bindWorkerSnapshot(WorkerSnapshot workerSnapshot);

/\*\*

\* 获取环境配置

\* **@return**

\*/

SpiderConfigure getConfigure();

/\*\*

\* 获取 manager

\* **@return**

\*/

WorkerSchedulerManagerImpl getManager();

/\*\*

\* 获取job运行快照

\*

\* **@return**

\*/

JobSnapshot getJobSnapshot();

/\*\*

\* 获取work Job

\* **@return**

\*/

Job getJob();

/\*\*

\* 获取工作空间

\* **@return**

\*/

WorkSpace<T> getWorkSpace();

/\*\*

\* 获取 worker name

\* **@return**

\*/

String getName();

/\*\*

\* 获取worker 快照

\* **@return**

\*/

WorkerSnapshot getWorkerSnapshot();

/\*\*

\* 获取工作频率

\* **@return**

\*/

**long** getWorkFrequency();

/\*\*

\* 获取最后一次活动时间

\* **@return**

\*/

**long** getLastActivityTime();

/\*\*

\* 开始方法 只有当 state== ready 时调用

\*/

**void** start();

/\*\*

\* 在运行状态下，没有处理数据事等待

\*/

**void** rest();

/\*\*

\* 暂停方法 只有在state==stared时候调用

\*/

**void** suspend();

/\*\*

\* 继续运行方法只有在state==suspend时调用

\*/

**void** goOn();

/\*\*

\* 停止方法在任何状态时候都可以调用

\*/

**void** stop();

/\*\*

\* 完成方法由管理者调用

\*/

**void** finish();

/\*\*

\* 销毁 方法 最后结束时候调用

\*/

**void** destroy();

/\*\*

\* 获取状态

\*

\* **@return**

\*/

WorkerLifecycleState getState();

## **Admin后台管理**

主要提供一些系统后台管理的服务接口,具体参考功能，看业务需求说明