**定时任务操作说明文档**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **日期** | **版本** | **变更位置** | **说明** | **作者** |
| 2016.6.12 | 1.0 |  | 编写文档 | 曾波 |
| 2016.6.13 | 1.1 | 4.定时任务升级需求列表 | 增加第4节 | 祝军 |
| 2016.6.17 | 1.2 | 数据库设计  主要业务逻辑 | 增加第5、6节 | 曾波 |
| 2016.6.22 | 1.3 | 接口说明 | 增加第7节 | 曾波 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1. 系统架构 4](#_Toc455130707)

[2. 操作步骤 4](#_Toc455130708)

[2.1. 任务列表 4](#_Toc455130711)

[2.2. 添加任务 6](#_Toc455130712)

[2.3. 查看任务 7](#_Toc455130713)

[2.4. 修改任务 8](#_Toc455130714)

[2.5. 触发 8](#_Toc455130715)

[2.6. 暂停 9](#_Toc455130716)

[2.7. 恢复 9](#_Toc455130717)

[2.8. 历史记录 9](#_Toc455130718)

[2.9. 统计数据 9](#_Toc455130719)

[3. 任务的迁移 10](#_Toc455130720)

[3.1. 创建Bean 10](#_Toc455130722)

[3.2. 创建方法 10](#_Toc455130723)

[3.3. 创建任务 11](#_Toc455130724)

# 系统架构



Schedule框架，通过项目公用Job：EayunJob，动态调用由Spring管理的Bean中的方法，并且由ScheduleService提供任务管理服务，如创建任务，触发任务，暂停、恢复任务，以及删除任务等。ECMC作为独立系统，通过Dubbo调用ScheduleService服务。数据库作为底层支撑，存储任务相关数据。

# 操作步骤



## 任务列表



任务列表展示了系统目前存在所有任务，展示信息如下：

**任务ID：**UUID任务的唯一标识；

**Bean名称：**任务调用的spring中的Bean名称；

**方法名：**任务执行Bean中的方法名称；

**状态：**任务状态分为删除、完成、暂停、错误、阻塞、正常；

**时间表达式：**任务执行的cron表达式；

**任务描述：**任务说明信息；

**操作：**

**修改：**修改任务的Bean名称、方法名称、调度表达式等信息；

**触发：**手动触发执行任务，可通过历史记录中查看执行情况；

**暂停：**暂停任务的自动执行机制，当任务处于暂停状态，该功能不可用；

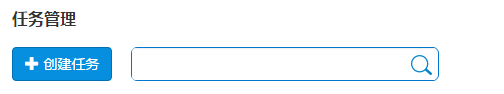
**恢复：**将暂停的任务恢复为自动执行状态，当任务状态为非暂停，该功能不可用；

**历史记录：**查看任务历史执行记录，是否成功和错误信息等；

**数据统计：**按天统计任务的执行总次数，成功次数，失败次数；

**删除：**删除任务

**搜索功能：**



输入的关键字，系统会根据Bean名称、方法名称以及描述信息来进行模糊匹配

## 添加任务



**Bean名称：**在spring中通过配置文件或者注解方式声明的Bean；

**方法名：**在声明的Bean中存在的方法名，方法可以没有参数，也可以有参数，但参数类型必须为org.quartz.JobDataMap类型；

**时间表达式：**cron表达式，如：0/30 \* \* \* \* ? \*；

**参数：**指定时任务执行Bean中方法的时候，所需要的参数；页面上以Key-Value的形式填写，可增加多个，但参数不能为空且多个参数之间的Key不能相同；

**描述：**对任务的说明信息；

## 查看任务



查看任务详情，字段说明参考以上，对新增字段说明如下：

**创建人：**任务创建人，系统管理账号名称；

**创建时间：**任务的创建时间：

**上次运行时间：**任务上次执行时间；

**下次运行时间：**任务下次执行时间；

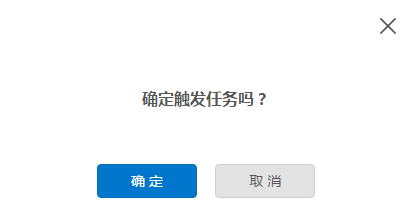
## 修改任务



注意：对任务信息进行修改，修改后立即查看任务信息，上次执行时间会被清空；

## 触发







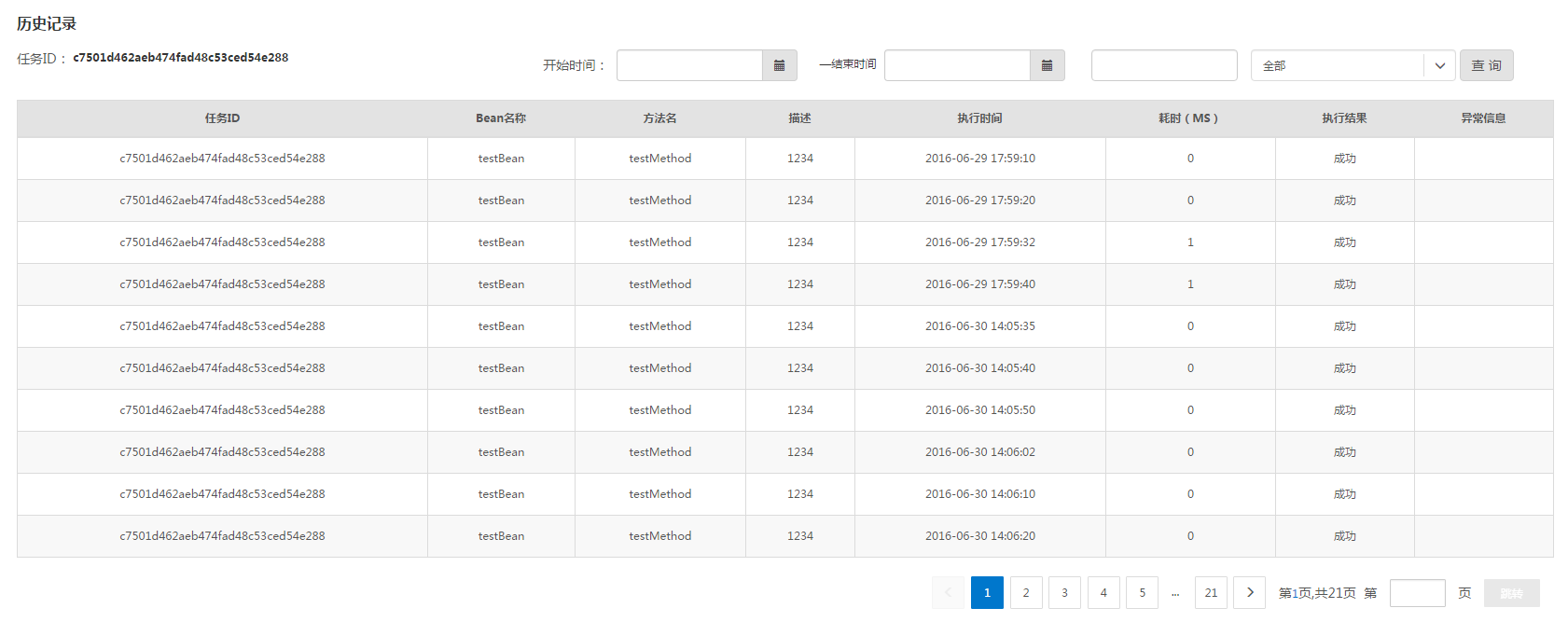
## 暂停

同上，点击操作—暂停按钮；

## 恢复

同上，点击操作—恢复按钮；

## 历史记录

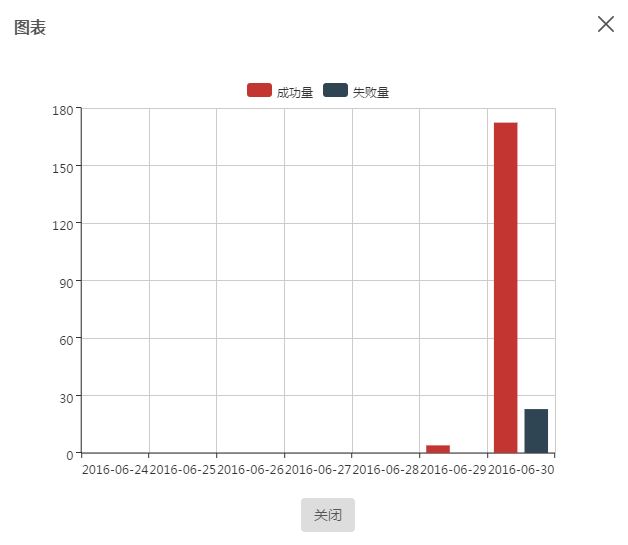


历史记录，可查询任务的往次执行情况，用户可根据开始时间和结束时间，任务执行结果（成功或者失败）、关键字来查询，注意：关键字搜索会根据Bean名称、方法名和描述信息三个字段进行模糊匹配。

## 统计数据



进入数据统计，默认查看最近一周的数据记录，用户可自行选择时间段进行查看，并可点击图表显示柱状图



# 任务的迁移

对于目前已经存在的任务计划，建议进行重建，步骤如下（以VmMonitorSummaryGatherJob为例）：



## 创建Bean

创建VmMonitorSummaryGatherBean.java文件，并使用注解@Component声明，Bean的名称为类名首字母小写，即：vmMonitorSummaryGatherBean；

## 创建方法

在VmMonitorSummaryGatherBean.java中创建方法，方法名自定义，如为gather，由于该任务执行时需自定义参数，则方法需带固定参数JobDataMap，所需参数通过JobDataMap中获取，方法声明如下；

不带参数：

private void gather(){}

带参数：

private void gather(JobDataMap jobDataMap){}

在方法体中将/eayun-virtualization/src/main/java/com/eayun/virtualization/job/VmMonitorSummaryGatherJob.java的executeInternal方法体迁移过来；

## 创建任务

通过任务管理创建任务



在gather方法中，可通过jobDataMap获取参数interval，如：

String interval = jobDataMap.getString("interval");

创建成功后



# 集群节点关闭

## 需求

当关闭集群中的某一个节点时，将该节点下的非正在执行任务，划到其他节点下运行，而当前节点下正在执行的任务，则需所有任务执行完成后，再划到其他节点下执行

## 原理

Quartz定时任务框架自带安全关闭接口org.quartz.impl.StdScheduler.shutdown(boolean)，满足该需求。

## 测试步骤

1. 新建一个工程项目，加入Quartz框架，创建25个任务（MyJob1……MyJob25），并在Quartz配置文件中配置，每个任务每隔5秒执行一次，任务的内容为：让当前线程沉睡6秒。这样，就能让任务一直处于运行中。
2. 自定义一个插件，用于Ctrl-C的时候，触发该插件，并配置到quartz.properties

org.quartz.plugin.shutdownhook.class = com.xxx.schedule.plugin.QuartzPlugin

org.quartz.plugin.shutdownhook.cleanShutdown = true

插件内容为：

getLog().info("Shutting down Quartz...");

try {

getLog().info("当前{}个任务正在执行", scheduler.getCurrentlyExecutingJobs().size());

scheduler.shutdown(isCleanShutdown());

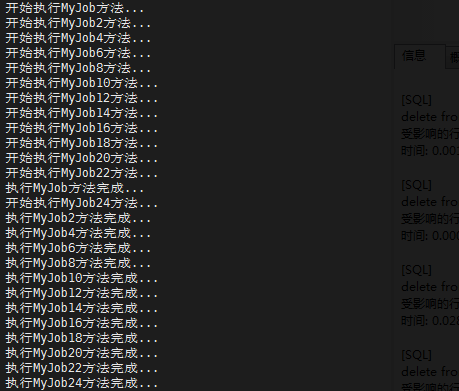
} catch (SchedulerException e) {

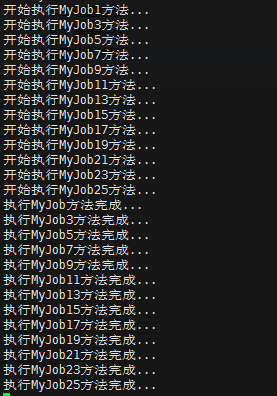
getLog().info(

"Error shutting down Quartz: " + e.getMessage(), e);

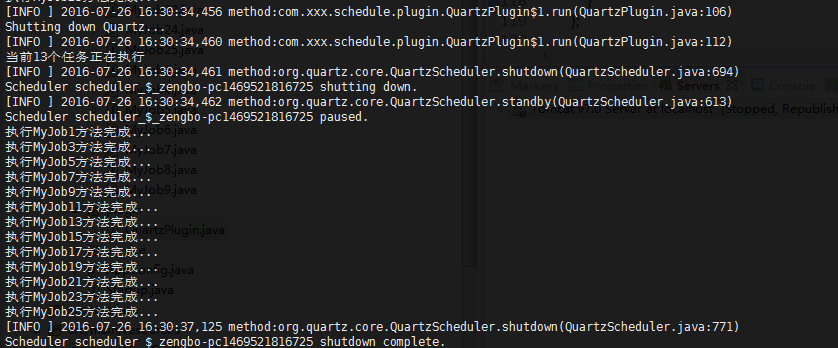
}

1. 生成jar，在命令行下分别启动两个节点，node1、node2
2. Quartz会将25个任务，轮训分配到两个节点下





1. 当使用Ctrl-C停止1节点时，打印日志如下：



1. 由上图可看出，节点停止是在所有任务执行完成之后。

## 结论

关闭节点，可使用org.quartz.impl.StdScheduler.shutdown(boolean)进行安全关闭。