**elasticjob使用说明**

-技术支持：陶杰

## 1.功能介绍

框架提供的分布式任务功能，集成elasticjob所有功能 和 elasticjob控制管理台，以及 额外的闲忙任务管理

## 2.配置说明

### 2.1依赖job模块

<dependency>

<groupId>com.yinhai</groupId>

<artifactId>ta3-modules-elasticjob-ta3-app</artifactId>

<version>4.1.0-RELEASE</version>

</dependency>

### 2.2 添加配置文件

1. 新增 spring-elasticjob.xml ,zookeeper-config.properties配置文件，放在resources下



### 2.3 app-context中添加引用

<import resource="spring-elasticjob.xml"/>

### 2.4 添加job

#### 2.4.1 job类型

**a）普通job，继承 SimpleJob**

public class SimpleJobTest implements SimpleJob {  
 @Override  
 public void execute(ShardingContext shardingContext) {  
 System.*out*.println("gg");  
 }  
}

**b）流式job，继承 DataflowJob**

public class DataflowJobTest implements DataflowJob<String>{  
 @Override  
 public List<String> fetchData(ShardingContext shardingContext) {  
 System.*out*.println("fetchData");  
 List<String> list = null;  
 //list = service.getList();  
 return list;  
 }  
  
 @Override  
 public void processData(ShardingContext shardingContext, List<String> list) {  
 System.*out*.println("processData");  
 }  
}

#### 2.4.2 job的初始化属性

|  |  |  |
| --- | --- | --- |
| 属性值 | 必输 | 作用 |
| jobName | 是 | job名称标识，唯一 |
| piecesCount | 是 | 分片数 |
| cron | 是 | 执行时间周期 |
| streamingProcess | 否，默认false  （DataflowJob 使用属性，SimpleJob不管） | 是否为流式执行作业  true：fetchData 返回的list 不为null，或者size大于0，则任务不结束，job会一直循环执行fetchData，processData，直到fetchData返回的list为null  false：不管fetchData 返回如何，job执行一次就结束，等待下次触发 |
| overwrite | 否，默认false | 是否覆盖ZK已存在的配置信息 |
| jobParameter | 否 | Job自定义参数，通过shardingContext.getJobParameter()获取 |
| description | 否 | Job描述信息 |
| failover | 否，默认false | 是否开启任务执行失效转移 |
| misfire | 否，默认false | 是否开启错过任务重新执行 |
| job\_exception\_handler | 否 | 作业异常处理器 |
| executor\_service\_handler | 否 | 线程池服务处理器 |

**failover** 增加zk，服务器压力，不建议使用

**misfire**在间隔时间短的任务中比较鸡肋。按照这个例子说，20s之后，新一次的作业就执行了，没必要再进行misfire。

misfire的正确用法是用于处理间隔时间长的作业，或者业务有局限的作业。举个例子：

一天跑一次的报表作业，作业的业务逻辑中写死本次作业只抓取昨天的数据。那么一旦上次作业跑动超过一天，那么第二天本该运行的作业就失去了运行的机会，第二天的数据也会缺失。因此需要misfire。

如果业务写的足够好，比如，无论哪天跑都能把未处理的数据分好组生成报表，misfire的存在意义就仅仅是触发时间缩短而已了。

**job\_exception\_handler**

<property name="job\_exception\_handler"><value>com.yinhai.test.elasticjob.MyJobExceptionHander</value></property>

public class MyJobExceptionHander implements JobExceptionHandler {  
 @Override  
 public void handleException(String jobName, Throwable throwable) {  
 throwable.printStackTrace();  
 }  
}

**executor\_service\_handler**

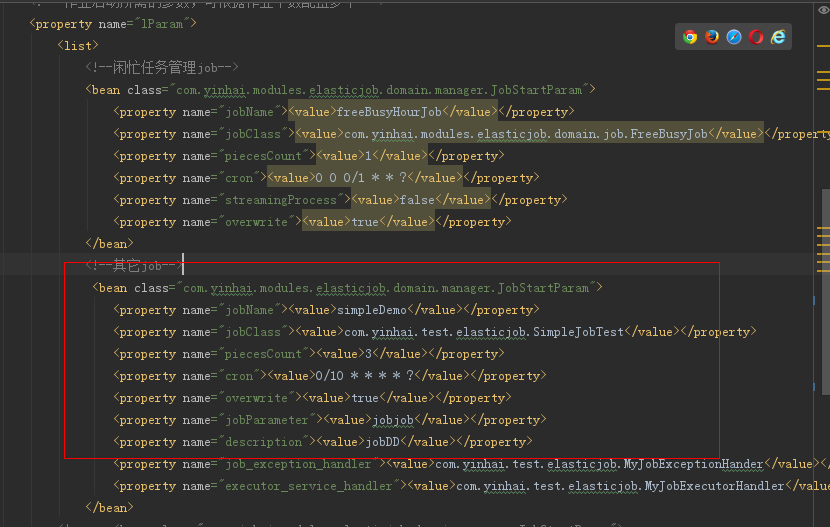
public class MyJobExecutorHandler implements ExecutorServiceHandler {  
 @Override  
 public ExecutorService createExecutorService(String jobName) {  
 return (new ExecutorServiceObject("inner-job-" + jobName, Runtime.*getRuntime*().availableProcessors() \* 2)).createExecutorService();  
 }  
}

### 2.5 例子



### 2.6 配置job

在spring-elasticjob.xml 中配置上创建的job，如下



### 2.7 修改zookeeper配置

在zookeeper-config.properties 修改使用的zk地址 ，自定该系统的命名空间

## 3.elasticjob管理功能说明

#### 3.1 zookeeper管理

（elasticJob/zookeeperCenterManagerController.do） 通过功能添加elasticjob 使用的zk信息，用于 elasticjob管理功能中 查询和管理job

#### 3.2 elasticjob管理

（elasticJob/elasticJobManagerConsoleController.do） 通过功能管理当前注册的job，可查看job运行信息，动态修改job属性等

#### 3.3 闲忙任务管理

（elasticJob/freeBusyHourManagerController.do） 通过功能管理当前注册的job的运行时间范围，闲时运行，忙时暂停