# [Elastic-Job——分布式定时任务框架](https://www.cnblogs.com/wyb628/p/7682580.html)

## Elastic-Job主要功能

* **定时任务：** 基于成熟的定时任务作业框架Quartz cron表达式执行定时任务。
* **作业注册中心：** 基于Zookeeper和其客户端Curator实现的全局作业注册控制中心。用于注册，控制和协调分布式作业执行。
* **作业分片：** 将一个任务分片成为多个小任务项在多服务器上同时执行。
* **弹性扩容缩容：** 运行中的作业服务器崩溃，或新增加n台作业服务器，作业框架将在下次作业执行前重新分片，不影响当前作业执行。
* **支持多种作业执行模式：** 支持OneOff，Perpetual和SequencePerpetual三种作业模式。
* **失效转移：** 运行中的作业服务器崩溃不会导致重新分片，只会在下次作业启动时分片。启用失效转移功能可以在本次作业执行过程中，监测其他作业服务器空闲，抓取未完成的孤儿分片项执行。
* **运行时状态收集：** 监控作业运行时状态，统计最近一段时间处理的数据成功和失败数量，记录作业上次运行开始时间，结束时间和下次运行时间。
* **作业停止，恢复和禁用：**用于操作作业启停，并可以禁止某作业运行（上线时常用）。
* **被错过执行的作业重触发：**自动记录错过执行的作业，并在上次作业完成后自动触发。可参考Quartz的misfire。
* **多线程快速处理数据：**使用多线程处理抓取到的数据，提升吞吐量。
* **幂等性：**重复作业任务项判定，不重复执行已运行的作业任务项。由于开启幂等性需要监听作业运行状态，对瞬时反复运行的作业对性能有较大影响。
* **容错处理：**作业服务器与Zookeeper服务器通信失败则立即停止作业运行，防止作业注册中心将失效的分片分项配给其他作业服务器，而当前作业服务器仍在执行任务，导致重复执行。
* **Spring支持：**支持spring容器，自定义命名空间，支持占位符。
* **运维平台：**提供运维界面，可以管理作业和注册中心。

项目开源地址：<https://github.com/dangdangdotcom/elastic-job>

## ****目录结构说明****

* **elastic-job-core**

elastic-job核心模块，只通过Quartz和Curator就可执行分布式作业。

* **elastic-job-spring**

elastic-job对spring支持的模块，包括命名空间，依赖注入，占位符等。

* **elastic-job-console**

elastic-job web控制台，可将编译之后的war放入tomcat等servlet容器中使用。

* **elastic-job-example**

使用例子。

* **elastic-job-test**

测试elastic-job使用的公用类，使用方无需关注。

**OneOff类型作业**

OneOff作业类型比较简单，需要继承AbstractOneOffElasticJob，该类只提供了一个方法用于覆盖，此方法将被定时执行。用于执行普通的定时任务，与Quartz原生接口相似，只是增加了弹性扩缩容和分片等功能。

public class MyElasticJob extends AbstractOneOffElasticJob {

    @Override

    protected void process(JobExecutionMultipleShardingContext context) {

        // do something by sharding items

    }

}

**Perpetual类型作业**

Perpetual作业类型略为复杂，需要继承AbstractPerpetualElasticJob并可以指定返回值泛型，该类提供两个方法可覆盖，分别用于抓取和处理数据。可以获取数据处理成功失败次数等辅助监控信息。**需要注意fetchData方法的返回值只有为null或长度为空时，作业才会停止执行，否则作业会一直运行下去。**这点是参照TbSchedule的设计。Perpetual作业类型更适用于流式不间歇的数据处理。

作业执行时会将fetchData的数据传递给processData处理，其中processData得到的数据是通过多线程（线程池大小可配）拆分的。建议processData处理数据后，更新其状态，避免fetchData再次抓取到，从而使得作业永远不会停止。processData的返回值用于表示数据是否处理成功，抛出异常或者返回false将会在统计信息中归入失败次数，返回true则归入成功次数。

public class MyElasticJob extends AbstractPerpetualElasticJob<Foo> {

@Override

protected List<Foo> fetchData(JobExecutionMultipleShardingContext context) {

List<Foo> result = // get data from database by sharding items

return result;

}

@Override

protected boolean processData(JobExecutionMultipleShardingContext context, Foo data) {

// process data

return true;

}

}

**SequencePerpetual类型作业**

SequencePerpetual作业类型和Perpetual作业类型极为相似，所不同的是Perpetual作业类型可以将获取到的数据多线程处理，但不会保证多线程处理数据的顺序。如：从2个分片共获取到100条数据，第1个分片40条，第2个分片60条，配置为两个线程处理，则第1个线程处理前50条数据，第2个线程处理后50条数据，无视分片项；SequencePerpetual类型作业则根据当前服务器所分配的分片项数量进行多线程处理，每个分片项使用同一线程处理，防止了同一分片的数据被多线程处理，从而导致的顺序问题。如：从2个分片共获取到100条数据，第1个分片40条，第2个分片60条，则系统自动分配两个线程处理，第1个线程处理第1个分片的40条数据，第2个线程处理第2个分片的60条数据。由于Perpetual作业可以使用多余分片项的任意线程数处理，所以性能调优的可能会优于SequencePerpetual作业。

public class MyElasticJob extends AbstractSequencePerpetualElasticJob<Foo> {

    @Override

    protected List<Foo> fetchData(JobExecutionSingleShardingContext context) {

        List<Foo> result = // get data from database by sharding items

        return result;

    }

    @Override

    protected boolean processData(JobExecutionSingleShardingContext context, Foo data) {

        // process data

        return true;

    }

}

<dependency>

<groupId>com.dangdang</groupId>

<artifactId>elastic-job-core</artifactId>

<version>1.0.1</version>

</dependency>

参考网址: <https://www.cnblogs.com/wyb628/p/7682580.html>

<http://ju.outofmemory.cn/entry/369298>

<dependency>

<groupId>com.dangdang</groupId>

<artifactId>elastic-job-lite-core</artifactId>

<version>2.1.5</version>

</dependency>

作业开发：Elastic-Job提供Simple、Dataflow和Script（文中没介绍）3种作业类型。

新手指南:

<https://www.cnblogs.com/yjmyzz/p/elastic-job-tutorial.html>