



JAVA任务调度

Seventeen



目录

CONTENTS

01 Timer

02 Quartz

03 Xxl-Job

04 What have we done

Timer

Timer是Java中Util包提供的定时器类。简单来说，它能让程序在指定的时间开始执行某些特定功能，也能让特定功能按照指定的周期循环执行。TimerTask是一个实现了Runnable接口的抽象类，代表一个可以被Timer执行的任务。





方法摘要

void	<code>cancel()</code>	终止此计时器，丢弃所有当前已安排的任务。
int	<code>purge()</code>	从此计时器的任务队列中移除所有已取消的任务。
void	<code>schedule(TimerTask task, Date time)</code>	安排在指定的时间执行指定的任务。
void	<code>schedule(TimerTask task, Date firstTime, long period)</code>	安排指定的任务在指定的时间开始进行重复的固定延迟执行。
void	<code>schedule(TimerTask task, long delay)</code>	安排在指定延迟后执行指定的任务。
void	<code>schedule(TimerTask task, long delay, long period)</code>	安排指定的任务从指定的延迟后开始进行重复的固定延迟执行。
void	<code>scheduleAtFixedRate(TimerTask task, Date firstTime, long period)</code>	安排指定的任务在指定的时间开始进行重复的固定速率执行。
void	<code>scheduleAtFixedRate(TimerTask task, long delay, long period)</code>	安排指定的任务在指定的延迟后开始进行重复的固定速率执行。

```
@Test
public void test() throws Exception {

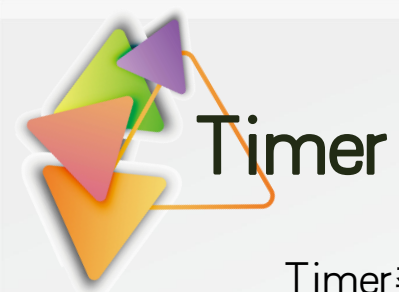
    new Timer().schedule(new TimerTask() {
        @Override
        public void run() {
            System.out.println("Hello!!");
        }
    }, delay: 3000, period: 2000);

    Thread.sleep( millis: 100000000);
}
```

Timer

定时调用

TimerTask



Timer类负责管理延时任务以及周期任务，然而它存在一些缺陷

01 Thread

Timer在执行所有定时任务时只会创建一个线程。如果某个任务的执行时间过长，那么将破坏其他TimerTask的定时精确性。

02 Exception

如果TimerTask抛出一个未检查的异常，将导致其终止定时线程，全挂了

03 Corn

不支持corn 表达式，应对复杂调度需求能力很弱

04 Distributed system

不支持分布式



Quartz

Quartz 是一个完全由 Java 编写的开源作业调度框架，为在 Java 应用程序中进行作业调度提供了简单却强大的机制。

Quartz 可以与 J2EE 与 J2SE 应用程序相结合也可以单独使用。

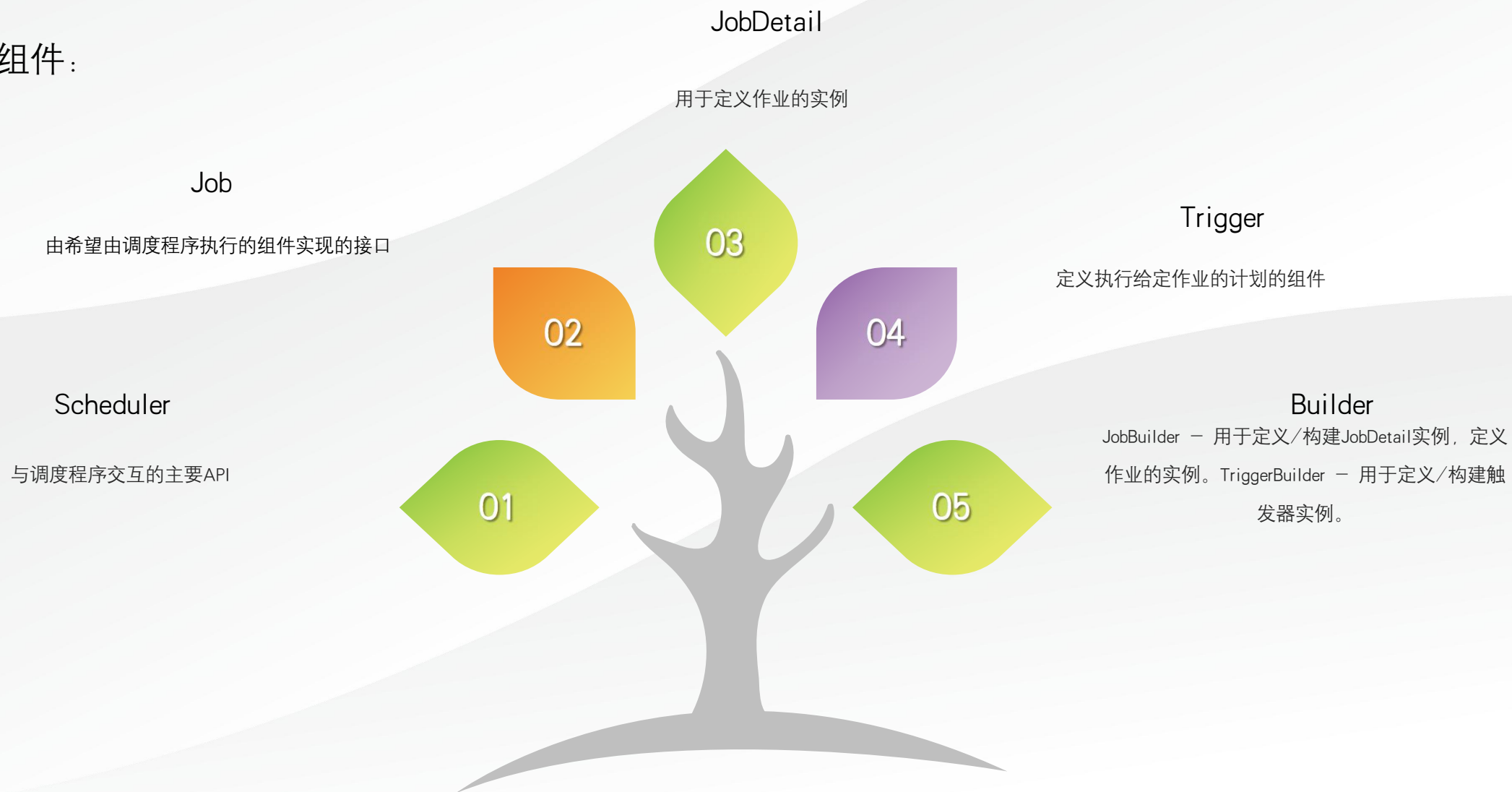
Quartz 允许程序开发人员根据时间的间隔来调度作业。

Quartz 实现了作业和触发器的多对多的关系，还能把多个作业与不同的触发器关联。





基本组件：





编程模
型:

```
SchedulerFactory schedFact = new org.quartz.impl.StdSchedulerFactory();

Scheduler sched = schedFact.getScheduler();

sched.start();

// define the job and tie it to our HelloJob class
JobDetail job = newJob(HelloJob.class)
    .withIdentity("myJob", "group1")
    .build();

// Trigger the job to run now, and then every 40 seconds
Trigger trigger = newTrigger()
    .withIdentity("myTrigger", "group1")
    .startNow()
    .withSchedule(simpleSchedule()
        .withIntervalInSeconds(40)
        .repeatForever())
    .build();

// Tell quartz to schedule the job using our trigger
sched.scheduleJob(job, trigger);
```




Listeners是您创建的对象，用于根据调度程序中发生的事件执行操作。



JobListener

job相关事件包括：job即将执行的通知，以及job完成执行时的通知。



TriggerListener

与触发相关的事件包括：触发器触发，触发失灵，触发完成（触发器关闭的作业完成）。



SchedulerListener

与计划程序相关的事件包括：添加job/触发器，删除job/触发器，调度程序中的严重错误，关闭调度程序的通知等。



JobStore负责跟踪您提供给调度程序的所有“工作数据”：jobs, triggers, 日历等。

RAMJobStore是使用最简单的JobStore，它将其所有数据保存在RAM中。当您的应用程序结束（或崩溃）时，所有调度信息都将丢失。

JDBCJobStore也被恰当地命名 – 它通过JDBC将其所有数据保存在数据库中。JDBCJobStore几乎与任何数据库一起使用，已被广泛应用于Oracle, PostgreSQL, MySQL, MS SQL Server, HSQLDB和DB2。要使用JDBCJobStore，必须首先创建一组数据库表以供Quartz使用。

Clustering目前与JDBC-Jobstore（JobStoreTX或JobStoreCMT）和TerracottaJobStore一起使用。功能包括负载平衡和job故障转移（如果JobDetail的“请求恢复”标志设置为true）。使用JobStoreTX或JobStoreCMT进行聚类通过将“org.quartz.jobStore.isClustered”属性设置为“true”来启用Clustering。Clustering中的每个实例都应该使用相同的quartz.properties文件。这样做的例外是使用相同的属性文件，具有以下允许的异常：不同的线程池大小，以及“org.quartz.scheduler.instanceId”属性的不同值。Clustering中的每个节点必须具有唯一的instanceId，通过将“AUTO”作为此属性的值，可以轻松完成（不需要不同的属性文件）。

Xxl-Job

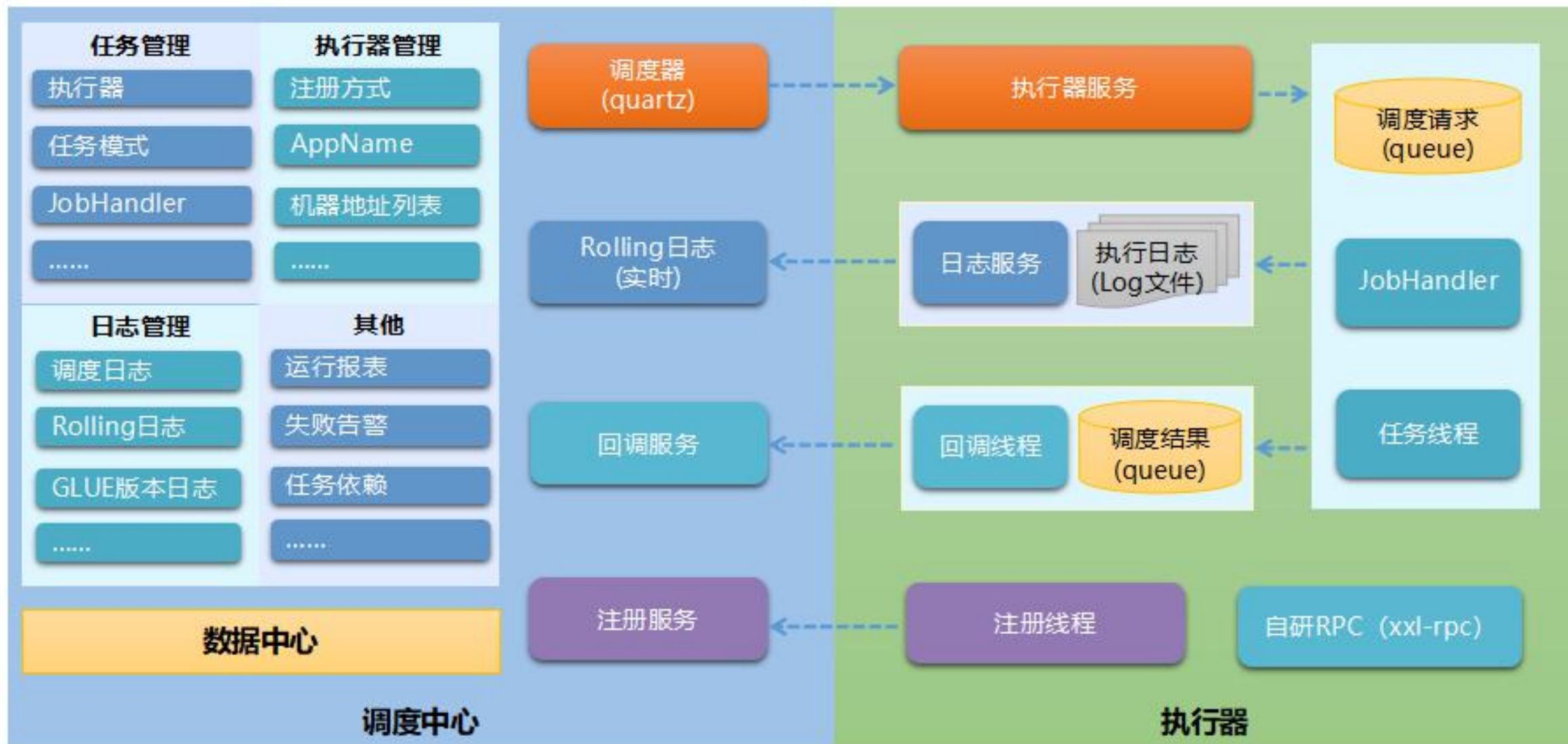
XXL-JOB是一个轻量级分布式任务调度平台，其核心设计目标是开发迅速、学习简单、轻量级、易扩展。现已开放源代码并接入多家公司线上产品线，开箱即用。





feature	quartz	elastic-job-cloud	xxl-job
依赖	mysql	jdk1.7+, zookeeper 3.4.6+ ,maven3.0.4+ ,mesos	mysql ,jdk1.7+ , maven3.0+
HA	多节点部署，通过竞争数据库锁来保证只有一个节点执行任务	通过zookeeper的注册与发现，可以动态的添加服务器。支持水平扩容	集群部署
任务分片	—	支持	支持
文档完善	完善	完善	完善
管理界面	无	支持	支持
难易程度	简单	较复杂	简单
公司	OpenSymphony	当当网	个人
社区活跃度	国际化产品	停止更新	版本迭代中
高级功能	—	弹性扩容，多种作业模式，失效转移，运行状态收集，多线程处理数据，幂等性，容错处理，spring命名空间支持	弹性扩容，分片广播，故障转移，Rolling实时日志，GLUE（支持在线编辑代码，免发布），任务进度监控，任务依赖，数据加密，邮件报警，运行报表，国际化
缺点	没有管理界面，以及不支持任务分片等。不适用于分布式场景	需要引入zookeeper，mesos，增加系统复杂度，学习成本较高	调度中心通过获取DB锁来保证集群中执行任务的唯一性，如果短任务很多，随着调度中心集群数量增加，那么数据库的锁竞争会比较厉害，性能不好。
使用企业	大众化产品，对分布式调度要求不高的公司大面积使用	36氪，当当网，国美，金柚网，联想，唯品会，亚信，平安，猪八戒等公司	已登记200余家公司

Xxl-Job





快速入门doc:

<http://wiki.hunliji.com/pages/viewpage.action?pageId=30934338>

XXL-JOB的系统架构图:



server端jar包: xxl-job-admin-2.0.1.jar

一.各环境配置信息

目前使用基于2.0.1稳定版内部分支二次开发改造, 涉及前端, 后端, 数据库表, 请不要随便变更版本。

二.整合XXL-JOB执行器

1. Maven依赖

打开pom.xml文件, 添加XXL-JOB执行器的依赖关系, 如下所示:

```
<!-- https://mvnrepository.com/artifact/com.xuxueli/xxl-job-core -->
<dependency>
  <groupId>com.xuxueli</groupId>
  <artifactId>xxl-job-core</artifactId>
  <version>2.0.1</version>
</dependency>
```

2. 执行器配置文件

在Apollo配置中心配置, 如下所示:

```
application

52: eureka.client.registry-fetch-interval-seconds = 2
53: eureka.instance.appname = ${spring.application.name}
54: eureka.instance.prefer-ip-address = true
55: eureka.instance.lease-renewal-interval-in-seconds = 10
56: eureka.instance.lease-expiration-duration-in-seconds = 30
57:
58:
59:
60: xxl.job.admin.addresses = http://127.0.0.1:8080/xxl-job-admin
61: xxl.job.executor.address =
62: xxl.job.executor.appname = springboot-job-executor
63: xxl.job.executor.ip =
64: xxl.job.executor.port = 9999
65: xxl.job.executor.token =
66: xxl.job.accessToken =
67: xxl.job.log.path =
68: xxl.job.executor.logpath = /data/applogs/xxl-job/jobhandler
69: xxl.job.log.retention-days =
70: xxl.job.executor.logretentiondays = 1
71:
```



目前缺少的部分：



用户管理

Xxl-Job目前没有提供用户管理的功能，只有简易登陆，用户名和密码配置在调度中心配置文件中

权限管理

目前没有权限管理，所有登陆人员都可随意操作调度中心的配置数据，无法保障平台任务配置安全性



多语言客户端

目前仅支持Java客户端，其他语言环境需要额外部署Java执行器，Http调用接口方式执行JOB



What have we done

提供更详细的Xxl-Job的使用文档，业务快速上手

Xxl-Job内部分支二次开发，完善功能





What have we done

任务调度中心

导航

运行报表

用户管理

任务管理

调度日志

执行器管理

使用教程

≡

用户管理

权限 全部

用户名

搜索

新增用户

每页 10 条记录

用户名	权限	操作
admin	管理员	<div>编辑</div> <div>删除</div>
admin1	普通用户	<div>编辑</div> <div>分配项目权限</div> <div>删除</div>
dddd	普通用户	<div>编辑</div> <div>分配项目权限</div> <div>删除</div>

第 1 页 (总共 1 页)

上页

1

下页

开发了官方未提供的用户管理模块，只有管理员角色可见与操作。在此界面进行用户和执行器的授权绑定，该方案是咨询了Xxl-Job作者之后得出的权限模型。



What have we done

任务调度中心

≡

注销

导航

运行报表

用户管理

任务管理

调度日志

执行器管理

使用教程

执行器管理

执行器列表

新增执行器

排序	AppName	名称	注册方式	OnLine 机器地址	操作
1	xxl-job-executor-sample	示例执行器	自动注册		<div>编辑</div> <div>删除</div>
2	dddddd	dddddd	自动注册		<div>编辑</div> <div>删除</div>
3	fffff	fffff	自动注册		<div>编辑</div> <div>删除</div>

执行器管理权限控制，只有管理员才能够对执行器做操作，普通用户仅能查看。



What have we done

任务调度中心

≡

注销

导航

运行报表

用户管理

任务管理

调度日志

执行器管理

使用教程

任务管理

执行器

dddddd

任务描述

JobHandler

搜索

新增任务

每页

10

条记录

任务ID	任务描述	运行模式	Cron	负责人	状态	操作
表中数据为空						
无记录						

上页

下页

任务管理权限控制，普通用户只能看到自己有授权的执行器下的Job。



What have we done

告警优化


- 按照小时+任务维度限制发送告警邮件的次数，超过`xxl.job.maxErrorCountAlarm`配置次数后不发送。

阻塞策略

- 增加阻塞策略：丢弃后续调度返回成功。该策略与丢弃后续调度的区别是，当任务阻塞时会直接返回成功。在调度日志中点击调度备注-查看，msg为`block strategy effect: Discard Return Success`的是触发该策略的调度记录。

跨平台支持

- 使用`CurlHandler`支持异构系统任务调用



感谢您的欣赏

THANKS