任务调度(定时任务)

# 什么是任务调度？

简单来说就是一个定时任务，在什么时间执行什么样的代码任务。

示例:

同步任务，调用失败之后，补偿机制 日志+定时job分布式事务解决方案

# 实现定时任务的几种方式？

Timer TimerTask

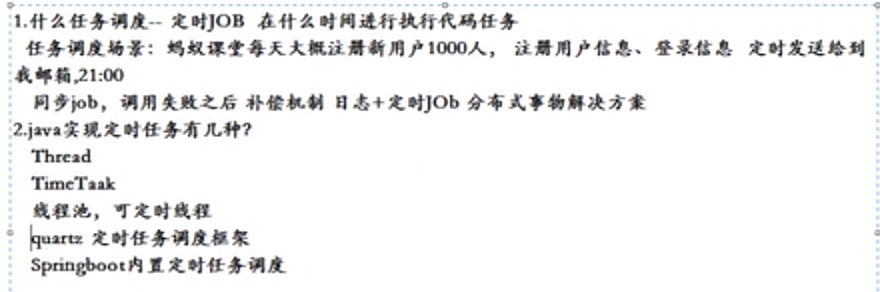
QuartZ

线程池-🡪可定时线程

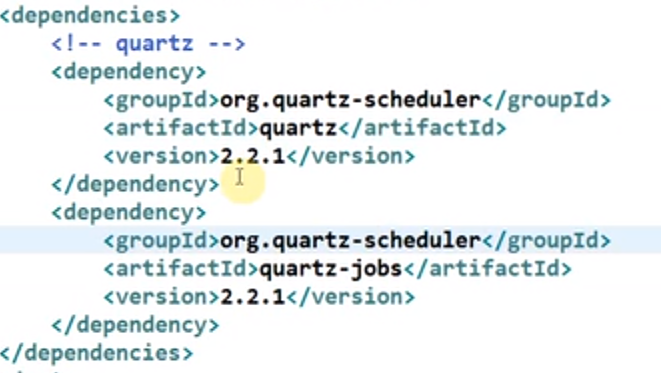
springboot内置的任务调度

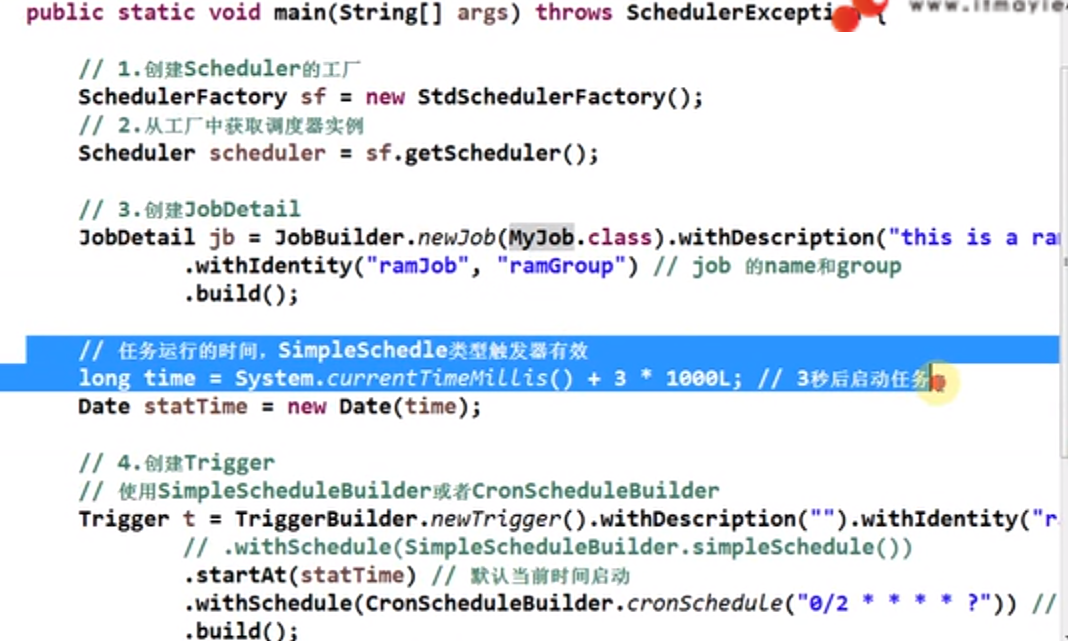
XXL-JOB

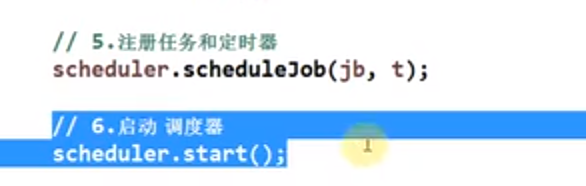
# 部分代码示例：

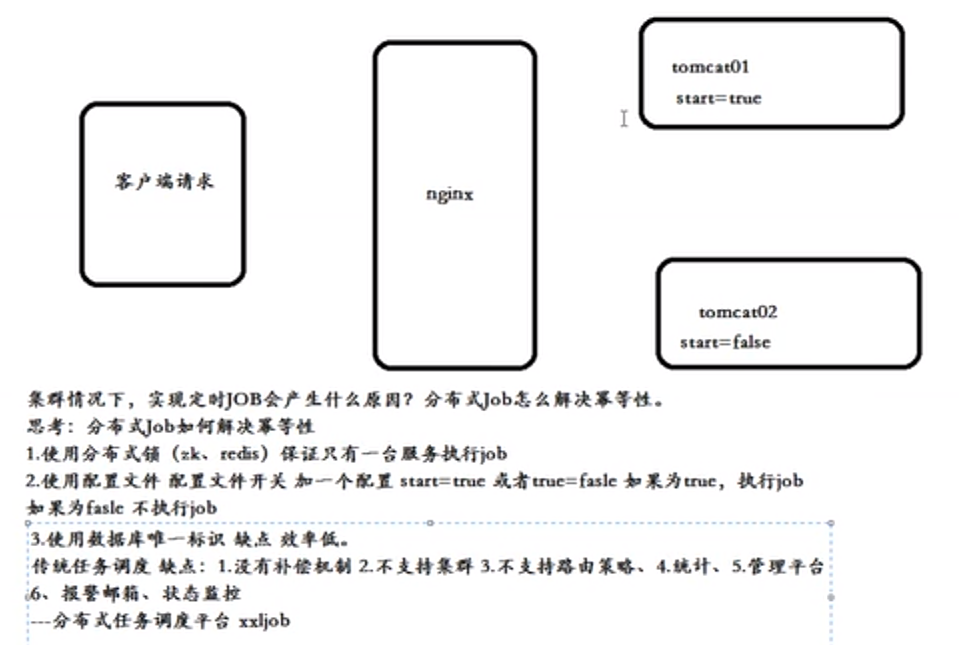












# 问题：分布式job怎么解决幂等性问题（数据一致性）。

1 使用分布式锁（zk redis）保证只有一台服务器执行job

2 使用配置文件，配置文件开关 加一个配置start=true 或者fasle，为true执行，false不执行

3 使用数据库唯一标识---比如两个服务器同时插入相同的唯一id，由于id不能重复，必定有一个会报错，则插入成功的执行任务，插入失败的不执行。（效率低）

# 传统任务调度的缺点：

1 没有补偿机制（就是失败后的备用处理方法）

2 不支持集群（传统调度做集群，会重复执行）

3 不支持路由策略（不能选择性的指派谁去执行任务）

4 没有统计（统计任务执行状况）

5 没有管理平台（可视化管理界面）

6 没有报警邮箱，状态监控

# XXL-JOB使用

解决方案：使用分布式任务调度平台：XXL-JOB

## [快速入门](http://www.xuxueli.com/xxl-job/#/?id=%e4%ba%8c%e3%80%81%e5%bf%ab%e9%80%9f%e5%85%a5%e9%97%a8)

官网：

http://www.xuxueli.com/xxl-job/#/?id=%E3%80%8A%E5%88%86%E5%B8%83%E5%BC%8F%E4%BB%BB%E5%8A%A1%E8%B0%83%E5%BA%A6%E5%B9%B3%E5%8F%B0xxl-job%E3%80%8B

### XXL-JOB整合SpringBoot：

只需要三大步：

1 加入依赖maven:

<dependency>

<groupId>com.xuxueli</groupId>

<artifactId>xxl-job-core</artifactId>

<version>${最新稳定版本}</version>

</dependency>

2 加入配置类：

直接把XxlJobConfig这个类添加到项目中即可

3 创建job，如不创建也可以使用平台GLUE JAVA的方法发布任务。

|  |
| --- |
| package com.xxl.job.executor.service.jobhandler;  import com.xxl.job.core.biz.model.ReturnT;  import com.xxl.job.core.handler.IJobHandler;  import com.xxl.job.core.handler.annotation.JobHandler;  import com.xxl.job.core.log.XxlJobLogger;  import org.springframework.stereotype.Component;  import java.util.concurrent.TimeUnit;  /\*\*  \* 任务Handler示例（Bean模式）  \*  \* 开发步骤：  \* 1、继承"IJobHandler"：“com.xxl.job.core.handler.IJobHandler”；  \* 2、注册到Spring容器：添加“@Component”注解，被Spring容器扫描为Bean实例；  \* 3、注册到执行器工厂：添加“@JobHandler(value="自定义jobhandler名称")”注解，注解value值对应的是调度中心新建任务的JobHandler属性的值。  \* 4、执行日志：需要通过 "XxlJobLogger.log" 打印执行日志；  \*  \* @author xuxueli 2015-12-19 19:43:36  \*/  @JobHandler(value="demoJobHandler")  @Component  public class DemoJobHandler extends IJobHandler {  @Override  public ReturnT<String> execute(String param) throws Exception {  XxlJobLogger.log("XXL-JOB, Hello World.");  for (int i = 0; i < 5; i++) {  XxlJobLogger.log("beat at:" + i);  TimeUnit.SECONDS.sleep(2);  }  System.out.println("====hello world");  return SUCCESS;  }  } |

### [初始化“调度数据库”](http://www.xuxueli.com/xxl-job/#/?id=_21-%e5%88%9d%e5%a7%8b%e5%8c%96%e8%b0%83%e5%ba%a6%e6%95%b0%e6%8d%ae%e5%ba%93)

请下载项目源码并解压，获取 "调度数据库初始化SQL脚本" 并执行即可，正常情况下应该生成16张表。

"调度数据库初始化SQL脚本" 位置为:

/xxl-job/doc/db/tables\_xxl\_job.sql

调度中心支持集群部署，集群情况下各节点务必连接同一个mysql实例;

如果mysql做主从,调度中心集群节点务必强制走主库;

### [编译源码](http://www.xuxueli.com/xxl-job/#/?id=_22-%e7%bc%96%e8%af%91%e6%ba%90%e7%a0%81)

解压源码,按照maven格式将源码导入IDE, 使用maven进行编译即可，源码结构如下：

xxl-job-admin：调度中心

xxl-job-core：公共依赖

xxl-job-executor-samples：执行器Sample示例（选择合适的版本执行器，可直接使用，也可以参考其并将现有项目改造成执行器）

：xxl-job-executor-sample-springboot：Springboot版本，通过Springboot管理执行器，推荐这种方式；

：xxl-job-executor-sample-spring：Spring版本，通过Spring容器管理执行器，比较通用；

：xxl-job-executor-sample-frameless：无框架版本；

：xxl-job-executor-sample-jfinal：JFinal版本，通过JFinal管理执行器；

：xxl-job-executor-sample-nutz：Nutz版本，通过Nutz管理执行器；

### [配置部署“调度中心”](http://www.xuxueli.com/xxl-job/#/?id=_23-%e9%85%8d%e7%bd%ae%e9%83%a8%e7%bd%b2%e8%b0%83%e5%ba%a6%e4%b8%ad%e5%bf%83)

调度中心项目：xxl-job-admin

作用：统一管理任务调度平台上调度任务，负责触发调度执行，并且提供任务管理平台。

#### [步骤一：调度中心配置：](http://www.xuxueli.com/xxl-job/" \l "/?id=%e6%ad%a5%e9%aa%a4%e4%b8%80%ef%bc%9a%e8%b0%83%e5%ba%a6%e4%b8%ad%e5%bf%83%e9%85%8d%e7%bd%ae%ef%bc%9a)

调度中心配置文件地址：

/xxl-job/xxl-job-admin/src/main/resources/xxl-job-admin.properties

调度中心配置内容说明：

### 调度中心JDBC链接：链接地址请保持和 2.1章节 所创建的调度数据库的地址一致

spring.datasource.url=jdbc:mysql://127.0.0.1:3306/xxl-job?Unicode=true&characterEncoding=UTF-8

spring.datasource.username=root

spring.datasource.password=root\_pwd

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

### 报警邮箱

spring.mail.host=smtp.qq.com

spring.mail.port=25

spring.mail.username=xxx@qq.com

spring.mail.password=xxx

spring.mail.properties.mail.smtp.auth=true

spring.mail.properties.mail.smtp.starttls.enable=true

spring.mail.properties.mail.smtp.starttls.required=true

### 登录账号

xxl.job.login.username=admin

xxl.job.login.password=123456

### 调度中心通讯TOKEN，非空时启用

xxl.job.accessToken=

### 调度中心国际化设置，默认为中文版本，值设置为“en”时切换为英文版本

xxl.job.i18n=

#### [步骤二：部署项目：](http://www.xuxueli.com/xxl-job/#/?id=%e6%ad%a5%e9%aa%a4%e4%ba%8c%ef%bc%9a%e9%83%a8%e7%bd%b2%e9%a1%b9%e7%9b%ae%ef%bc%9a)

如果已经正确进行上述配置，可将项目编译打包部署。 调度中心访问地址：[**http://localhost:8080/xxl-job-admin**](http://localhost:8080/xxl-job-admin) (该地址执行器将会使用到，作为回调地址)，登录后运行界面如下图所示



至此“调度中心”项目已经部署成功。

#### [步骤三：调度中心集群（可选）：](http://www.xuxueli.com/xxl-job/#/?id=%e6%ad%a5%e9%aa%a4%e4%b8%89%ef%bc%9a%e8%b0%83%e5%ba%a6%e4%b8%ad%e5%bf%83%e9%9b%86%e7%be%a4%ef%bc%88%e5%8f%af%e9%80%89%ef%bc%89%ef%bc%9a)

调度中心支持集群部署，提升调度系统容灾和可用性。

调度中心集群部署时，几点要求和建议：

* DB配置保持一致；
* 登陆账号配置保持一致；
* 集群机器时钟保持一致（单机集群忽视）；
* 建议：推荐通过nginx为调度中心集群做负载均衡，分配域名。调度中心访问、执行器回调配置、调用API服务等操作均通过该域名进行。

#### [其他：Docker 镜像方式搭建调度中心：](http://www.xuxueli.com/xxl-job/#/?id=%e5%85%b6%e4%bb%96%ef%bc%9adocker-%e9%95%9c%e5%83%8f%e6%96%b9%e5%bc%8f%e6%90%ad%e5%bb%ba%e8%b0%83%e5%ba%a6%e4%b8%ad%e5%bf%83%ef%bc%9a)

* 下载镜像

// Docker地址：https://hub.docker.com/r/xuxueli/xxl-job-admin/ (建议指定版本号)

docker pull xuxueli/xxl-job-admin

* 创建容器并运行
* docker run -p 8080:8080 -v /tmp:/data/applogs --name xxl-job-admin -d xuxueli/xxl-job-admin
* /\*\*
* \* 如需自定义 mysql 等配置，可通过 "PARAMS" 指定，参数格式 RAMS="--key=value --key2=value2" ；
* \* 配置项参考文件：/xxl-job/xxl-job-admin/src/main/resources/application.properties
* \*/
* docker run -e PARAMS="--spring.datasource.url=jdbc:mysql://127.0.0.1:3306/xxl-job?Unicode=true&characterEncoding=UTF-8" -p 8080:8080 -v /tmp:/data/applogs --name xxl-job-admin -d xuxueli/xxl-job-admin

### [配置部署“执行器项目”](http://www.xuxueli.com/xxl-job/#/?id=_24-%e9%85%8d%e7%bd%ae%e9%83%a8%e7%bd%b2%e6%89%a7%e8%a1%8c%e5%99%a8%e9%a1%b9%e7%9b%ae)

“执行器”项目：xxl-job-executor-sample-springboot (提供多种版本执行器供选择，现以 springboot 版本为例，可直接使用，也可以参考其并将现有项目改造成执行器)

作用：负责接收“调度中心”的调度并执行；可直接部署执行器，也可以将执行器集成到现有业务项目中。

#### [步骤一：maven依赖](http://www.xuxueli.com/xxl-job/#/?id=%e6%ad%a5%e9%aa%a4%e4%b8%80%ef%bc%9amaven%e4%be%9d%e8%b5%96)

确认pom文件中引入了 "xxl-job-core" 的maven依赖；

#### [步骤二：执行器配置](http://www.xuxueli.com/xxl-job/#/?id=%e6%ad%a5%e9%aa%a4%e4%ba%8c%ef%bc%9a%e6%89%a7%e8%a1%8c%e5%99%a8%e9%85%8d%e7%bd%ae)

执行器配置，配置文件地址：

/xxl-job/xxl-job-executor-samples/xxl-job-executor-sample-springboot/src/main/resources/application.properties

执行器配置，配置内容说明：

### 调度中心部署跟地址 [选填]：如调度中心集群部署存在多个地址则用逗号分隔。执行器将会使用该地址进行"执行器心跳注册"和"任务结果回调"；为空则关闭自动注册；

xxl.job.admin.addresses=http://127.0.0.1:8080/xxl-job-admin

### 执行器AppName [选填]：执行器心跳注册分组依据；为空则关闭自动注册

xxl.job.executor.appname=xxl-job-executor-sample

### 执行器IP [选填]：默认为空表示自动获取IP，多网卡时可手动设置指定IP，该IP不会绑定Host仅作为通讯实用；地址信息用于 "执行器注册" 和 "调度中心请求并触发任务"；

xxl.job.executor.ip=

### 执行器端口号 [选填]：小于等于0则自动获取；默认端口为9999，单机部署多个执行器时，注意要配置不同执行器端口；

xxl.job.executor.port=9999

### 执行器通讯TOKEN [选填]：非空时启用；

xxl.job.accessToken=

### 执行器运行日志文件存储磁盘路径 [选填] ：需要对该路径拥有读写权限；为空则使用默认路径；

xxl.job.executor.logpath=/data/applogs/xxl-job/jobhandler

### 执行器日志保存天数 [选填] ：值大于3时生效，启用执行器Log文件定期清理功能，否则不生效；

xxl.job.executor.logretentiondays=-1

#### [步骤三：执行器组件配置](http://www.xuxueli.com/xxl-job/#/?id=%e6%ad%a5%e9%aa%a4%e4%b8%89%ef%bc%9a%e6%89%a7%e8%a1%8c%e5%99%a8%e7%bb%84%e4%bb%b6%e9%85%8d%e7%bd%ae)

执行器组件，配置文件地址：

/xxl-job/xxl-job-executor-samples/xxl-job-executor-sample-springboot/src/main/java/com/xxl/job/executor/core/config/XxlJobConfig.java

执行器组件，配置内容说明：（这个必须要，直接把XxlJobConfig这个类添加到项目中即可）

@Bean(initMethod = "start", destroyMethod = "destroy")

public XxlJobSpringExecutor xxlJobExecutor() {

logger.info(">>>>>>>>>>> xxl-job config init.");

XxlJobSpringExecutor xxlJobSpringExecutor = new XxlJobSpringExecutor();

xxlJobSpringExecutor.setAdminAddresses(adminAddresses);

xxlJobSpringExecutor.setAppName(appName);

xxlJobSpringExecutor.setIp(ip);

xxlJobSpringExecutor.setPort(port);

xxlJobSpringExecutor.setAccessToken(accessToken);

xxlJobSpringExecutor.setLogPath(logPath);

xxlJobSpringExecutor.setLogRetentionDays(logRetentionDays);

return xxlJobSpringExecutor;

}

#### [步骤四：部署执行器项目：](http://www.xuxueli.com/xxl-job/#/?id=%e6%ad%a5%e9%aa%a4%e5%9b%9b%ef%bc%9a%e9%83%a8%e7%bd%b2%e6%89%a7%e8%a1%8c%e5%99%a8%e9%a1%b9%e7%9b%ae%ef%bc%9a)

如果已经正确进行上述配置，可将执行器项目编译打部署，系统提供多种执行器Sample示例项目，选择其中一个即可，各自的部署方式如下。

xxl-job-executor-sample-springboot：项目编译打包成springboot类型的可执行JAR包，命令启动即可；

xxl-job-executor-sample-spring：项目编译打包成WAR包，并部署到tomcat中。

xxl-job-executor-sample-jfinal：同上

xxl-job-executor-sample-nutz：同上

至此“执行器”项目已经部署结束。

#### [步骤五：执行器集群（可选）：](http://www.xuxueli.com/xxl-job/#/?id=%e6%ad%a5%e9%aa%a4%e4%ba%94%ef%bc%9a%e6%89%a7%e8%a1%8c%e5%99%a8%e9%9b%86%e7%be%a4%ef%bc%88%e5%8f%af%e9%80%89%ef%bc%89%ef%bc%9a)

执行器支持集群部署，提升调度系统可用性，同时提升任务处理能力。

执行器集群部署时，几点要求和建议：

* 执行器回调地址（xxl.job.admin.addresses）需要保持一致；执行器根据该配置进行执行器自动注册等操作。
* 同一个执行器集群内AppName（xxl.job.executor.appname）需要保持一致；调度中心根据该配置动态发现不同集群的在线执行器列表。

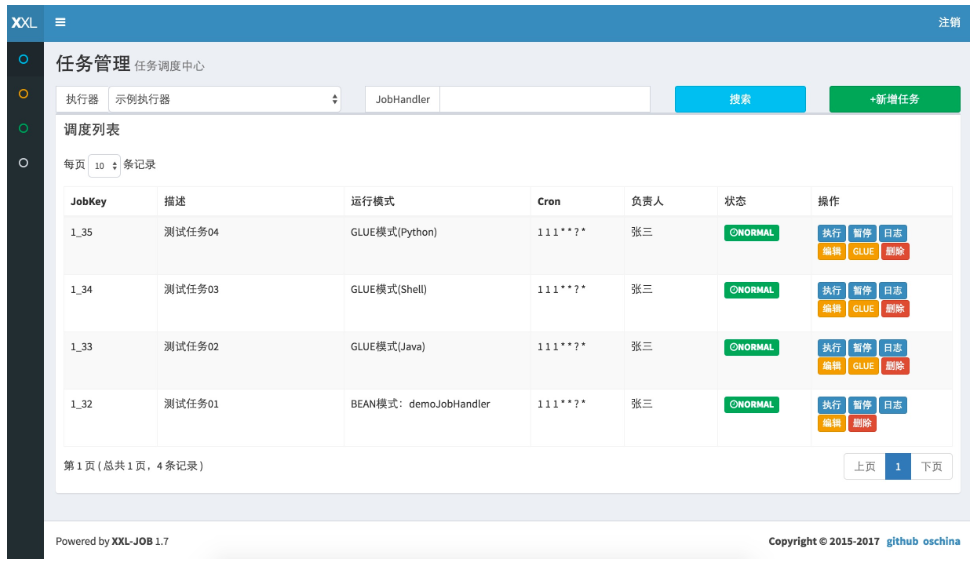
### [开发第一个任务“Hello World”](http://www.xuxueli.com/xxl-job/#/?id=_25-%e5%bc%80%e5%8f%91%e7%ac%ac%e4%b8%80%e4%b8%aa%e4%bb%bb%e5%8a%a1hello-world)

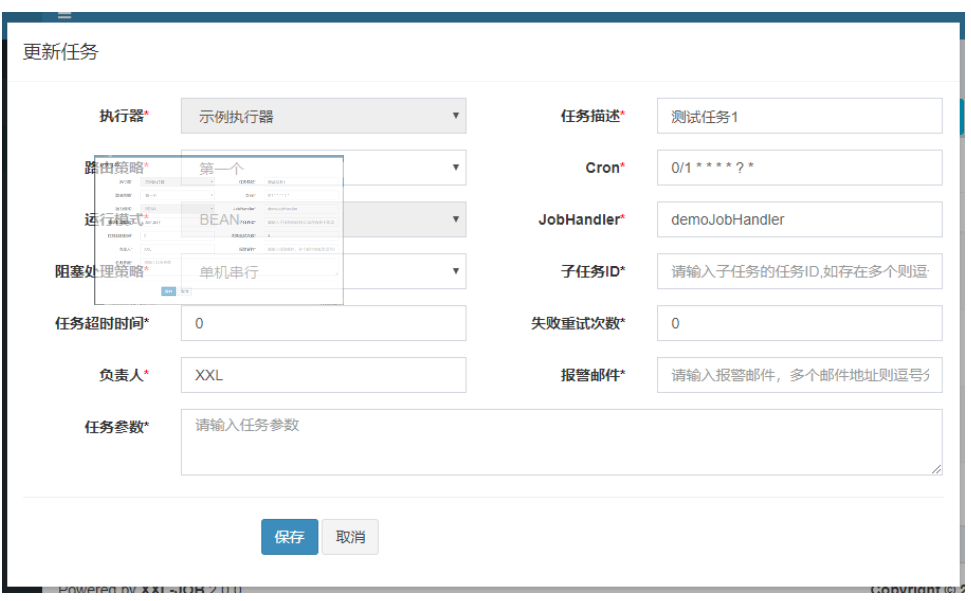
本示例以新建一个 “GLUE模式(Java)” 运行模式的任务为例。更多有关任务的详细配置，请查看“章节三：任务详解”。 （ “GLUE模式(Java)”的执行代码托管到调度中心在线维护，相比“Bean模式任务”需要在执行器项目开发部署上线，更加简便轻量）

**前提：请确认“调度中心”和“执行器”项目已经成功部署并启动；**

#### [步骤一：新建任务：](http://www.xuxueli.com/xxl-job/#/?id=%e6%ad%a5%e9%aa%a4%e4%b8%80%ef%bc%9a%e6%96%b0%e5%bb%ba%e4%bb%bb%e5%8a%a1%ef%bc%9a)

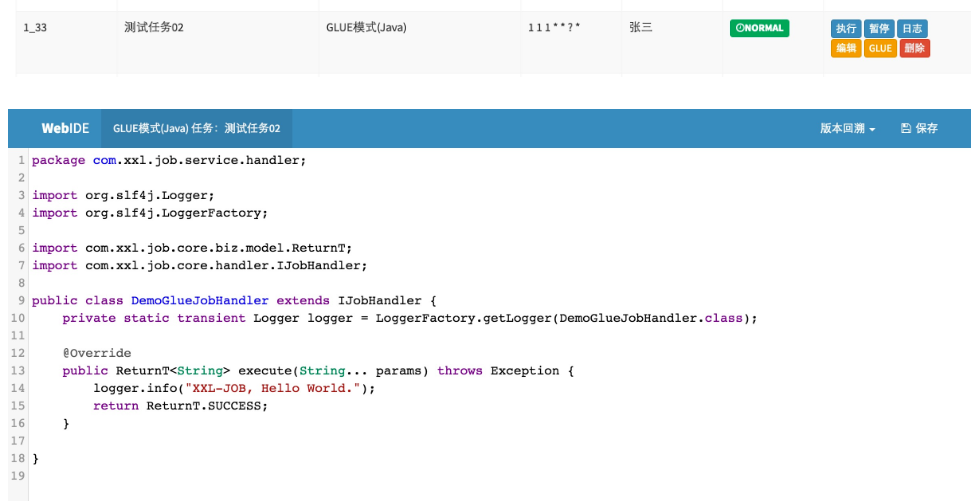
登录调度中心，点击下图所示“新建任务”按钮，新建示例任务。然后，参考下面截图中任务的参数配置，点击保存。





#### [步骤二：“GLUE模式(Java)” 任务开发：](http://www.xuxueli.com/xxl-job/#/?id=%e6%ad%a5%e9%aa%a4%e4%ba%8c%ef%bc%9aglue%e6%a8%a1%e5%bc%8fjava-%e4%bb%bb%e5%8a%a1%e5%bc%80%e5%8f%91%ef%bc%9a)

请点击任务右侧 “GLUE” 按钮，进入 “GLUE编辑器开发界面” ，见下图。“GLUE模式(Java)” 运行模式的任务默认已经初始化了示例任务代码，即打印Hello World。 （ “GLUE模式(Java)” 运行模式的任务实际上是一段继承自IJobHandler的Java类代码，它在执行器项目中运行，可使用@Resource/@Autowire注入执行器里中的其他服务，详细介绍请查看第三章节）

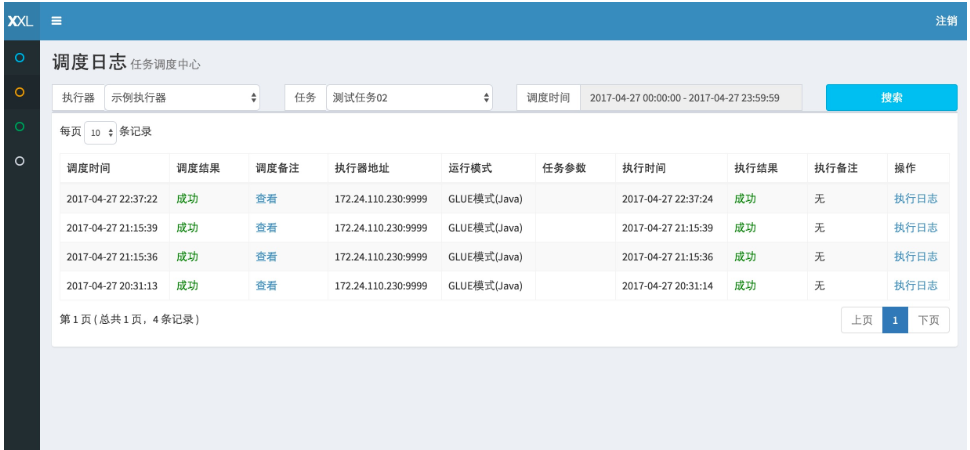


#### [步骤三：触发执行：](http://www.xuxueli.com/xxl-job/#/?id=%e6%ad%a5%e9%aa%a4%e4%b8%89%ef%bc%9a%e8%a7%a6%e5%8f%91%e6%89%a7%e8%a1%8c%ef%bc%9a)

请点击任务右侧 “执行” 按钮，可手动触发一次任务执行（通常情况下，通过配置Cron表达式进行任务调度出发）。

#### [步骤四：查看日志：](http://www.xuxueli.com/xxl-job/#/?id=%e6%ad%a5%e9%aa%a4%e5%9b%9b%ef%bc%9a%e6%9f%a5%e7%9c%8b%e6%97%a5%e5%bf%97%ef%bc%9a)

请点击任务右侧 “日志” 按钮，可前往任务日志界面查看任务日志。 在任务日志界面中，可查看该任务的历史调度记录以及每一次调度的任务调度信息、执行参数和执行信息。运行中的任务点击右侧的“执行日志”按钮，可进入日志控制台查看实时执行日志。



在日志控制台，可以Rolling方式实时查看任务在执行器一侧运行输出的日志信息，实时监控任务进度；

