XXL-JOB 1.3 用户手册

[一 简介 2](#_Toc18071)

[1.1 背景 2](#_Toc4077)

[1.2 发展 2](#_Toc991)

[1.3 概述 2](#_Toc25499)

[1.4 特点 2](#_Toc20316)

[1.5 下载 3](#_Toc29225)

[1.6 环境 3](#_Toc703)

[二 快速入门 4](#_Toc9335)

[2.1 初始化“调度数据库” 4](#_Toc18835)

[2.2 编译源码 5](#_Toc28004)

[2.3 配置部署“调度中心” 5](#_Toc29363)

[2.4 配置部署“执行器” 7](#_Toc20976)

[2.5 开发第一个任务“Hello World” 8](#_Toc25193)

[三 系统详解 12](#_Toc29538)

[3.1 源码目录介绍 12](#_Toc21114)

[3.1 “调度数据库”详解 12](#_Toc13209)

[3.2 “调度中心”详解 12](#_Toc14259)

[3.3 “执行器”详解 12](#_Toc3318)

[四 任务详解 13](#_Toc8363)

[4.1 BEAN模式任务 13](#_Toc17022)

[4.2 GLUE模式任务 13](#_Toc16350)

[五 任务管理 14](#_Toc29221)

[5.1 编辑任务信息 14](#_Toc9033)

[5.2 编辑GLUE代码 14](#_Toc8805)

[5.3 恢复/暂停 14](#_Toc28201)

[5.4 手动触发一次调度 14](#_Toc28040)

[5.5 查看日志 14](#_Toc19863)

[5.6 删除任务 14](#_Toc16467)

[六 设计原理 15](#_Toc26385)

[6.1 总体架构 15](#_Toc3826)

[6.2 调度原理 15](#_Toc14382)

[6.3 通讯原理 15](#_Toc14930)

[6.4 执行器原理 15](#_Toc22797)

[七 其他 16](#_Toc29011)

[7.1 接入登记 16](#_Toc24396)

[7.2 报告问题 16](#_Toc29810)

# 一 简介

## 1.1 背景

作业调度系统我使用过相当一部分，比如Java Timer、linux corntab和quartz，但是都遇到一些共同的问题，开发成本高，维护复杂等等。在我进入大众点评之后，我试用过内部调度系统，功能完善，但是开发和部署流程略复杂。

我钟爱KISS原则，因此我萌发了打造XXL-JOB的想法，期望是能够创造一种全新的调度体验。

## 1.2 发展

我于2015-11-28在github上创建XXL-JOB项目仓库并提交第一个commit，随之进行系统结构设计，UI选型，交互设计……

于2016-12-05日XXL-JOB终于release了第一个大版本V1.0， 随后我将之发布到OSCHINA，XXL-JOB在OSCHINA上获得了@红薯的推荐，同期分别达到了OSCHINA的“热门动弹”排行第一和git.oschina的月热度排行第一，在此特别感谢红薯，感谢大家的关注和支持。

于2015-12月中旬我将XXL-JOB发表到我司内部知识库，得到内部同事认可。于2016-01月我司展开XXL-JOB的内部接入和定制工作。

我司大众点评已接入XXL-JOB，内部别名《Ferrari》（Ferrari基于XXL-JOB的V1.1版本定制而成，新接入应用推荐升级最新版本V1.3）。自2016-01-21接入至2016-05-20为止，该系统已调度40000余次，表现优异。

至今，XXL-JOB已接入多家公司的线上产品线，场景如电商业务，O2O业务和大数据作业等等，欢迎大家使用，XXL-JOB也将拥抱变化，持续发展。

## 1.3 概述

XXL-JOB是一款简单高效的分布式任务调度平台，支持可视化且实时动态的管理Trigger和Job。

## 1.4 特点

简单：支持通过Web在线开发Job并配置，流程简洁；

高效：任务动态部署和配置，节省打包和部署消耗，提高开发效率；

分布式：得益于Quartz的集群支持，XXL-JOB调度中心天生支持HA，避免单点故障；

实时Trigger：支持可视化的管理Trigger，动态管理Triggrer；

实时Job：支持可视化的管理Job，动态管理Job，V1.3支持在线开发业务代码；

## 1.5 下载

Github地址：<https://github.com/xuxueli/xxl-job>

Git@OSC地址：<http://git.oschina.net/xuxueli0323/xxl-job>

（我将会在两个git仓库同步发布最新代码）



（图1.3：github下载链接位置）

源码下载请前往github自行下载，下载位置见上图1.3；

## 1.6 环境

Maven3

JDK1.7

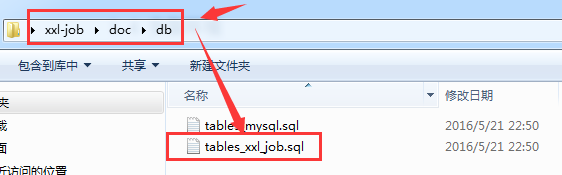
Tomcat7

Mysql5.5

# 二 快速入门

## 2.1 初始化“调度数据库”

请下载项目源码并解压，然后前往“图2.1A”所示目录，获取SQL脚本并执行，脚本文件位置：**“源码解压根目录”\xxl-job\doc\db\tables\_xxl\_job.sql**



（图2.1A：数据库建表SQL文件位置截图，）

正常情况下，应该生成（图2.1B）中所示14张表。

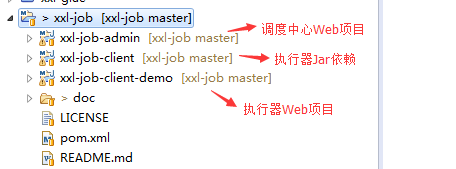


（图2.1B：调度数据库表一览）

## 2.2 编译源码

解压源码，按照maven格式将源码导入IDE（文档以Eclipse为例），更新项目pom依赖，maven编译项目。

正常情况下，项目结构应该如图2.2所示，



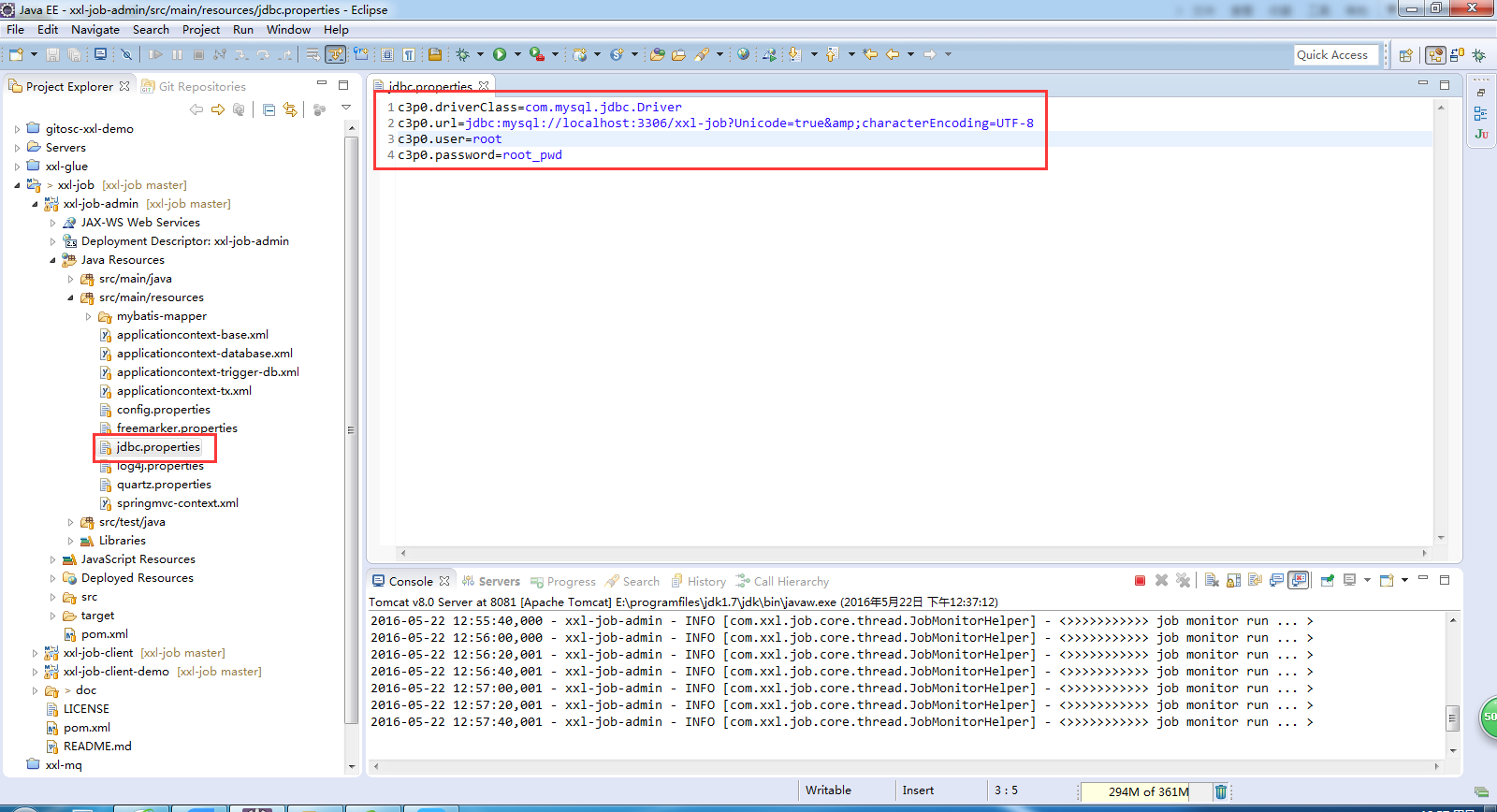
（图2.2：项目源码导入Eclipse截图）

## 2.3 配置部署“调度中心”

“调度中心”项目：xxl-job-admin

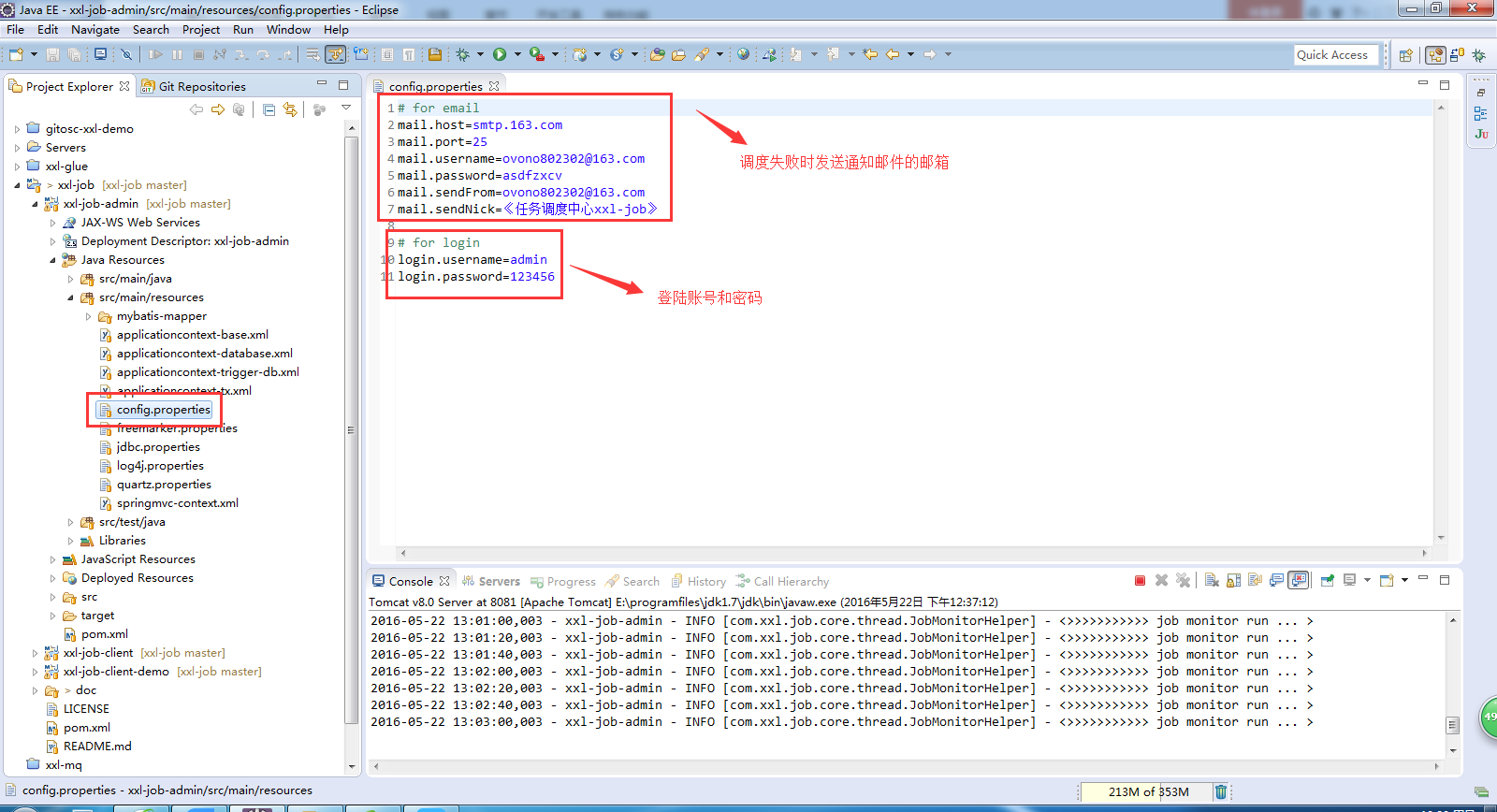
作用：统一管理任务调度平台上调度任务，负责触发调度执行。

**A：配置调度中心JDBC链接**：请在图2.2A所示位置配置jdbc链接地址，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。



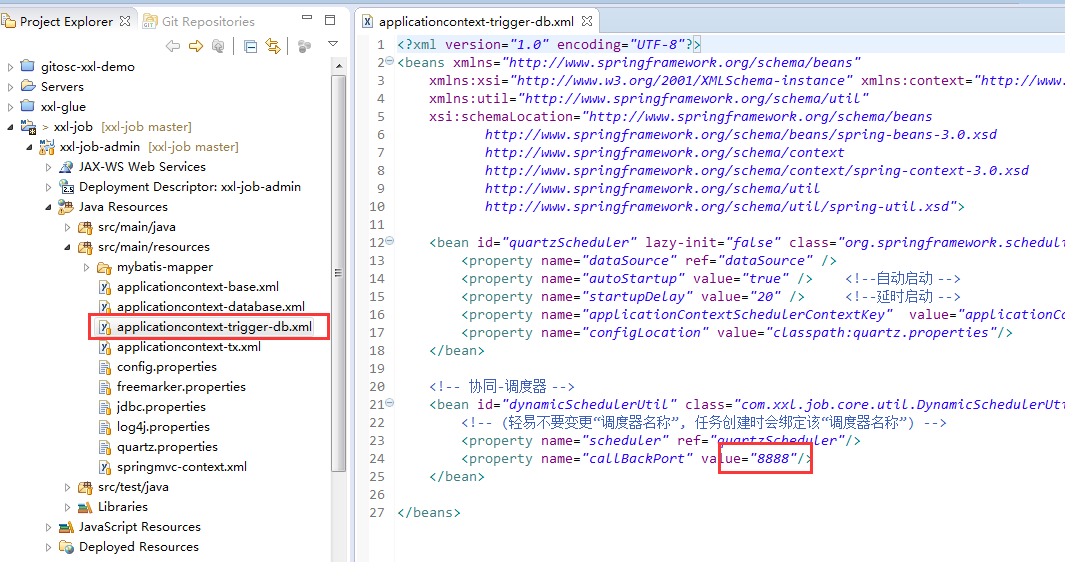
（图2.2A：数据库JDBC配置截图）

**B：配置报警邮箱和登陆账号**：请在图2.2B所示位置，设置自己的报警邮件发送邮箱和登陆账号密码。



（图2.2B：系统配置截图）

**C：配置“调度中心”日志回调端口**：由于“调度中心”和“执行器”部署在不同机器，“执行器”会请求该端口回调通知任务执行情况。如图2.2C所示，默认回调服务端口号为8888。（此端口除非与现有端口冲突，可自行修改，否则请忽视）



（图2.2C）

**部署项目**：如果已经正确进行上述配置，可将项目部署到eclipse下的tomcat服务器中，如图2.2D所示。或者，将“调度中心”项目导出war包单独部署。



（图2.2D：调度中心部署截图）

**访问链接**：http://localhost:8080/xxl-job-admin/ ，登陆界面如图2.2E所示。

至此“调度中心”项目已经部署结束。



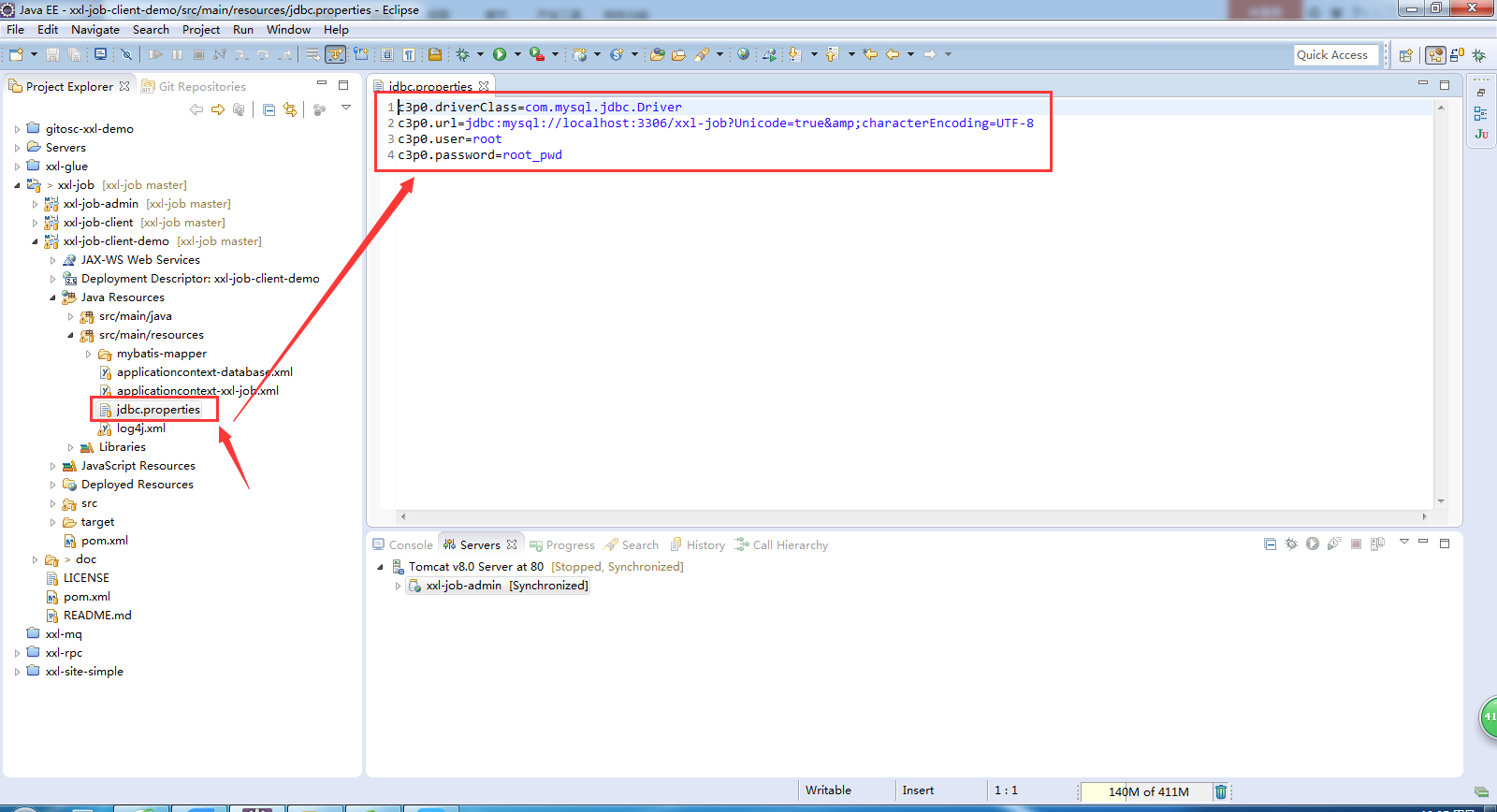
（图2.2E：“调度中心”登陆界面）

## 2.4 配置部署“执行器项目”

“执行器”项目：xxl-job-client-demo

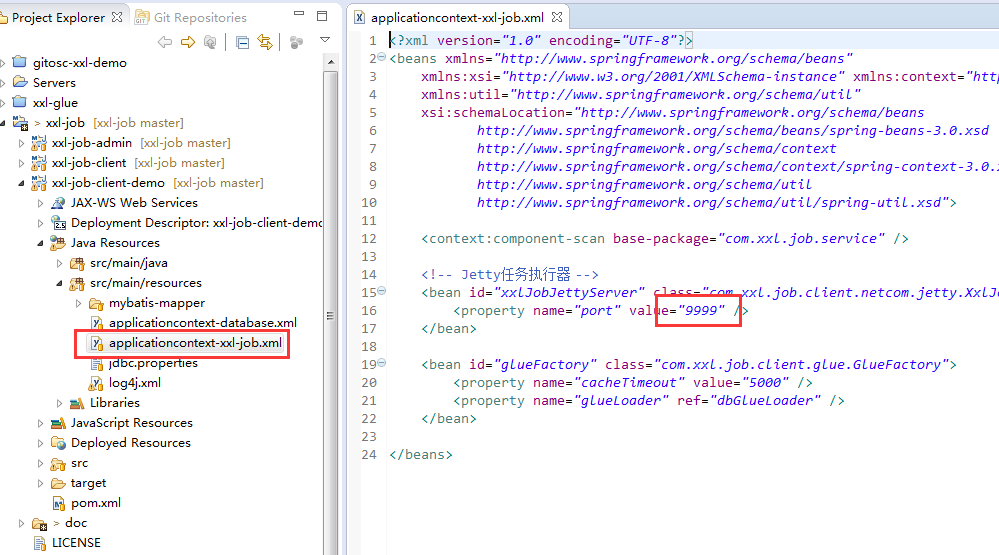
作用：负责接收“调度中心”的调度并执行；

**A：配置Jdbc链接**：请在图2.4A所示位置配置jdbc链接地址，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。



（图2.4A：“执行器”项目Jdbc链接配置截图）

**B：配置“执行器”端口**：由于“调度中心”和“执行器”部署在不同机器上，“调度中心”会请求该端口触发任务执行。如图2.4B所示，默认的“执行器”端口是9999，如果与系统现有端口冲突可自行修改，如若不冲突，可忽略。



（图2.4B：“执行器”端口截图）

**部署项目**：如果已经正确进行上述配置，可将项目部署到eclipse下的tomcat服务器中。或者，将“执行器”项目导出war包单独部署。

至此“执行器”项目已经部署结束。

## 2.5 开发第一个任务“Hello World”

本示例为新建一个“GLUE模式任务”（GLUE模式任务的执行代码支持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）。更多有关任务的详细配置，请查看“章节三：任务详解”。

**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）：**登陆调度中心，点击图2.5A所示新建任务按钮，新建示例任务。然后，参考图2.5B进行示例任务参数配置，点击保存。



（图2.5A：“调度中心”新建任务按钮）



（图2.5B：“调度中心”任务配置截图）

**步骤二（GLUE开发）：**请点击图2.5C中所示GLUE入口按钮，进入GLUE开发界面，如图2.5D。GLUE任务默认已经初始化了示例任务代码，即打印Hello World。

（GLUE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注入执行器里中的服务）



（图2.5C：“调度中心”管理管理界面，GLUE入口按钮）



（图2.5D：“调度中心”GLUE编辑界面）

**步骤三（触发执行）：**点击图2.5E所示“执行”按钮，可手动触发一次任务执行。



（图2.5E：“调度中心”管理管理界面，任务手动执行按钮接口）

**步骤四（查看日志）：**点击图2.5F所示“日志”按钮，可前往任务日志界面查看任务日志。在如图2.5G的任务日志界面中，可查看任务调度状态，执行器接收到调度请求后的执行状态，同时，点击如图2.5G中的“执行日志”按钮，可以查看本此调度在执行器端的完整执行日志，完整日志如图2.5H。



（图2.5F：“调度中心”管理管理界面，任务日志入口）



（图2.5G：“调度中心”管理管理界面，任务日志入口）



（图2.5H：“调度中心”管理管理界面，任务日志入口）

# 三 任务详解

## 4.1 任务参数介绍



（图4.1A：新增任务表单接口）

|  |  |
| --- | --- |
| 参数名 | 功能 |
| 任务组 | 用于分组管理任务 |
| 任务名 | 任务组内，任务的唯一标示 |
| Corn | 触发执行执行的Corn表达式 |
| 描述 | 描述信息 |
| 执行器地址 | 任务所在的执行器的地址，格式为“IP:Port”。其中IP为执行器机器IP，Port为“执行器Server”端口，如图2.4B所示。 |
| JobHandler | 非GLUE模式任务是才需要配置，该模式下任务代码是一个继承IJobHandler的Java类对象，需要提供JobHandlee的名称，执行器才可以匹配任务Handler并执行。 |
| 执行参数 | JOBHandler的入参 |
| 负责人 | 任务负责人 |
| 报警邮件 | 任务失败报警邮件发送地址 |
| 报警阀值 | 失败超过该阀值才会发送邮件 |
| 开启GLUE模式 | 如开启GLUE模式，“执行器Server”会读取GLUE代码并执行，否则，“执行器Server”将会匹配项目中的JobHander并执行； |

## 4.2 BEAN模式任务

Bean模式任务：任务逻辑以JobHandler的形式存在于“执行器”所在项目中，有以下特点；

## 4.3 GLUE模式任务

# 四 任务管理

## 5.1 编辑任务信息

## 5.2 编辑GLUE代码

## 5.3 恢复/暂停

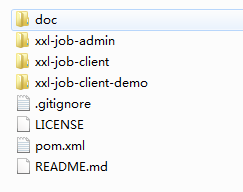
## 5.4 手动触发一次调度

## 5.5 查看日志

## 5.6 删除任务

# 三 系统详解

## 3.1 源码目录介绍



（图3.1：源码目录截图）

|  |  |
| --- | --- |
| 目录 | 介绍 |
| /doc | 用户手册 |
| /doc/db | “调度数据库”建表脚本 |
| /xxl-job-admin | “调度中心”Wed项目源码 |
| /xxl-job-client | “执行器”Jar依赖 |
| /xxl-job-client-demo | “执行器”示例Wed项目源码。 |

其中，Wed项目“xxl-job-client-demo”是Demo执行器项目，大家可以在该项目上进行开发，也可以将现有项目改造生成执行器项目；

## 3.2 “调度数据库”配置

XXL-JOB调度模块基于Quartz集群实现，其“调度数据库”是在Quartz的11张集群mysql表基础上扩展而成。

XXL-JOB首先定制了Quartz原生表结构前缀（XXL\_JOB\_QRTZ\_），如图3.2A所示。

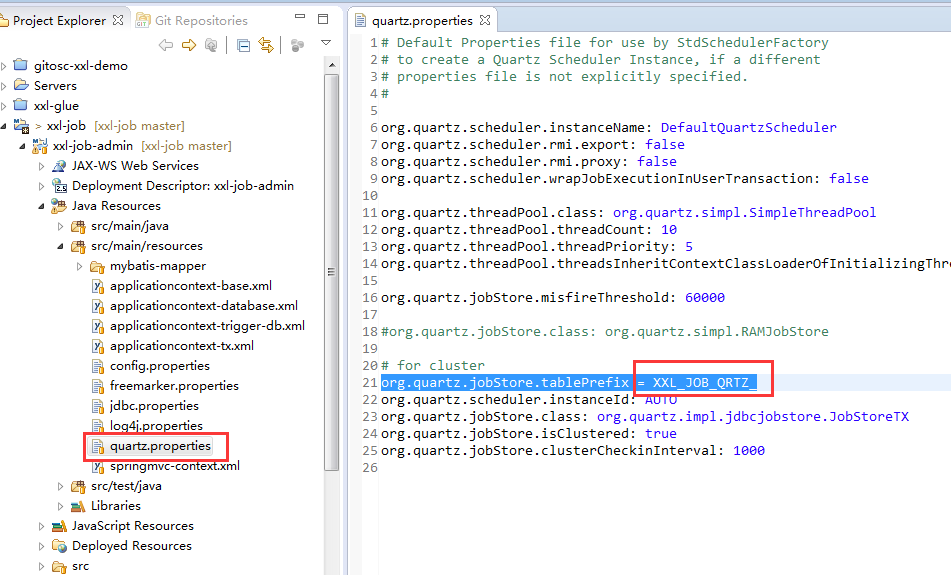
然后，在此基础上新增了三张扩展表，如下：

xxl\_job\_qrtz\_trigger\_info：

xxl\_job\_qrtz\_trigger\_log：

xxl\_job\_qrtz\_trigger\_logglue：

因此，XXL-JOB调度数据库共计用于14张数据库表，详细介绍如图3.2B所示。



（图3.2A：XXL-JOB数据库，公共前缀配置，截图）

**表介绍：**XXL-JOB拥有14张表，



（图3.2B：调度数据库，表列表截图）

|  |  |
| --- | --- |
| 表明 | 介绍 |
| xxl\_job\_qrtz\_blob\_triggers | Quartz表 |
| xxl\_job\_qrtz\_calendars | Quartz表 |
| xxl\_job\_qrtz\_cron\_triggers | Quartz表 |
| xxl\_job\_qrtz\_fired\_triggers | Quartz表 |
| xxl\_job\_qrtz\_job\_details | Quartz表 |
| xxl\_job\_qrtz\_locks | Quartz表 |
| xxl\_job\_qrtz\_paused\_trigger\_grps | Quartz表 |
| xxl\_job\_qrtz\_scheduler\_state | Quartz表 |
| xxl\_job\_qrtz\_simple\_triggers | Quartz表 |
| xxl\_job\_qrtz\_simprop\_triggers | Quartz表 |
| xxl\_job\_qrtz\_trigger\_info | 调度扩展信息表： 用于保存XXL-JOB调度任务的扩展信息，如任务分组、任务名、机器地址、执行器、执行入参和报警邮件等等； |
| xxl\_job\_qrtz\_trigger\_log | 调度日志表： 用于保存XXL-JOB任务调度的历史信息，如调度结果、执行结果、调度入参、调度机器和执行器等等； |
| xxl\_job\_qrtz\_trigger\_logglue | 任务GLUE日志：用于保存GLUE更新历史，用于支持GLUE的版本回溯功能； |
| xxl\_job\_qrtz\_triggers | Quartz表 |

## 3.2 “调度中心”详解

## 3.3 “执行器”详解

# 六 设计原理

## 6.1 总体架构

## 6.2 调度原理

## 6.3 通讯原理

## 6.4 执行器原理

# 七 其他

## 7.1 接入登记

更多接入公司，欢迎在<https://github.com/xuxueli/xxl-job/issues/1> 登记。

## 7.2 报告问题

XXL-JOB托管在Github上，如有问题可在[ISSUES](https://github.com/xuxueli/xxl-job/issues)上提问，也可以加入技术交流群(仅作技术交流)：367260654 。