# 目录

[目录 1](#_Toc445372661)

[1. 数据结构 2](#_Toc445372662)

[**1、** **逻辑结构** 2](#_Toc445372663)

[**2、** **物理结构** 3](#_Toc445372664)

[**3、** **测试数据展示** 6](#_Toc445372665)

[2. 初始化Job和Task数据 6](#_Toc445372666)

[3. 部署调度管理控制台 6](#_Toc445372667)

[**1、** **安装tomcat** 6](#_Toc445372668)

[**2、** **部署控制台war包** 6](#_Toc445372669)

[**3、** **配置Zookeeper相关参数** 8](#_Toc445372670)

[4. 配置调度任务和策略 8](#_Toc445372671)

[**1、** **添加调度任务** 8](#_Toc445372672)

[**2、** **添加调度策略** 9](#_Toc445372673)

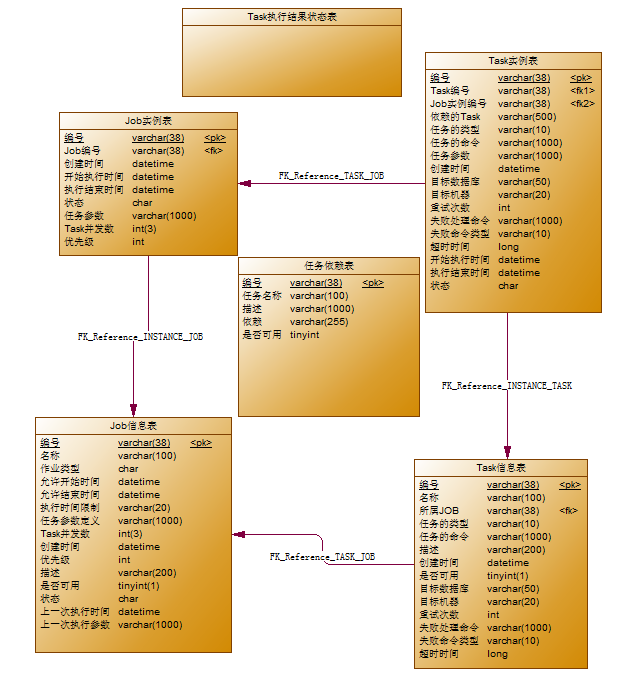
[5. 部署程序 10](#_Toc445372674)

[6. 运行与监测 12](#_Toc445372675)

# 数据结构

1. **逻辑结构**

整个Schedule框架涉及的数据模型及其关系如下图所示：



**说明：**

Job信息表：模版表，记录了各种Job的参数（一个Job包含多个要执行的Task）

Task信息表：模版表，记录了Task的各种参数

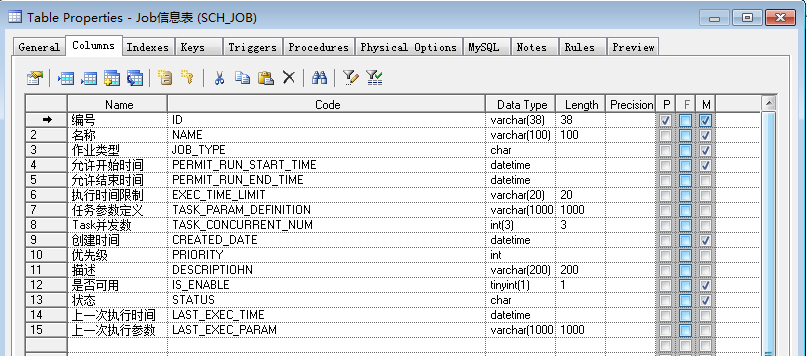
Job实例表：Job到了运行的时候会拷贝Job信息表的数据生成一个Job实例并存放在此表。

Task实例表：生成Job实例的同时会生成该Job的Task实例并存放在此表

任务依赖表：存放Task所依赖的其他Task的信息

Task执行结果状态表：Task执行完成后框架会查询此表数据判断是否业务上执行成功

1. **物理结构**
2. Job信息表



**字段说明：**

1、Job信息表的”执行时间限制”字段有着特殊的含义，它的格式如下：

yyyy-MM-dd xx::xx::xx

yyyy-MM-dd在执行时将被替换成实际日期。

2、这里重点解释一下Job信息表中”任务参数定义”字段，它是一个json数组字符串，格式如下：

'[

{

"name": "KETTLE\_PATH",

"defaultValue": "/opt/kettle/data-integration",

"value":""

"increment": -1,

"canAdditive": false,

"type": "string(date、string、number)",

"unit": ""(for data:yyyy,MM,dd,HH,mm,ss)

},

{

"name": "yesterday",

"defaultValue": "2016-03-08",

"value":""

"increment": -1,

"canAdditive": true,

"type": "date",

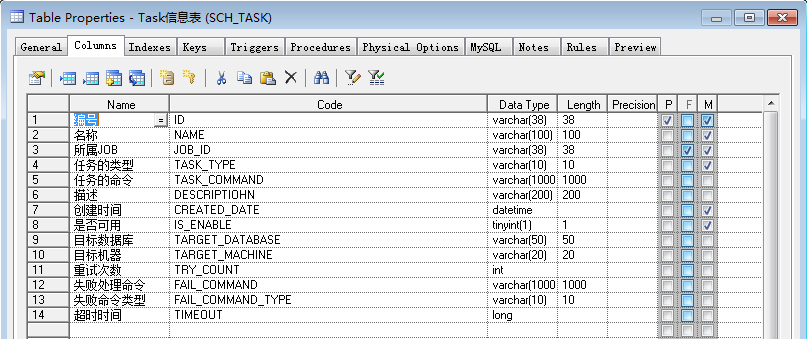
"unit": "dd"

}

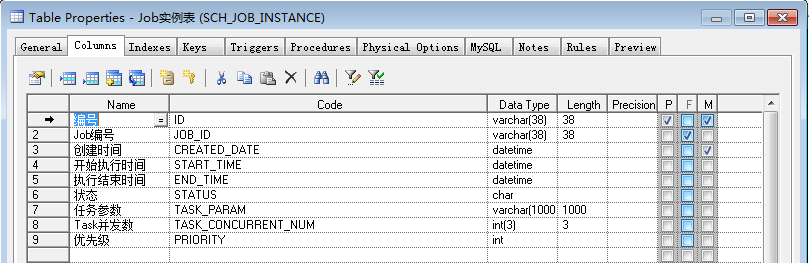
……

]'

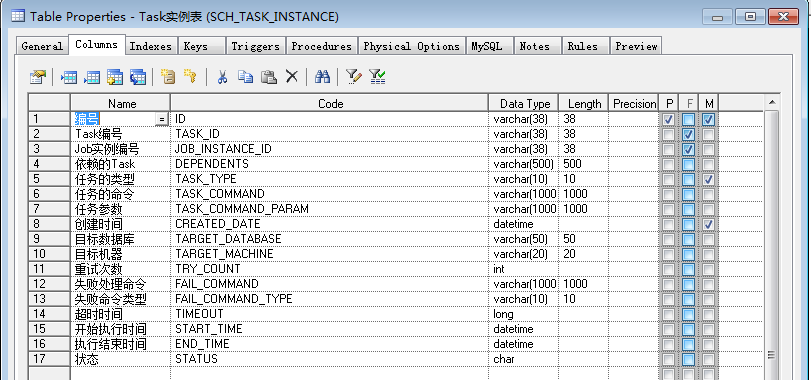
1. Task信息表



1. Job实例表



1. Task实例表

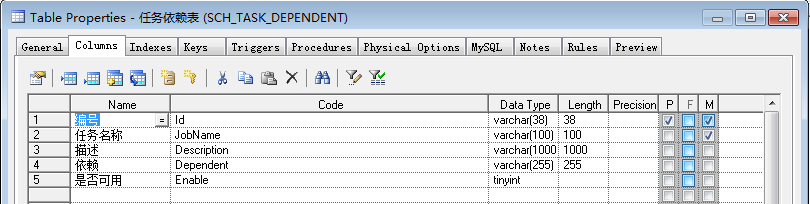


**说明：**

任务类型：SHELLl：Shell任务；SQL：SQL语句任务

任务的状态：W:等待调度;A:可执行;R:执行中;S:成功;F:失败;K:跳过;T:处理超时

1. 任务依赖表



1. **测试数据展示**

新建一个日作业（Job）：

insert into `SCH\_JOB` (`ID`, `NAME`, `JOB\_TYPE`, `PERMIT\_RUN\_START\_TIME`, `PERMIT\_RUN\_END\_TIME`, `EXEC\_TIME\_LIMIT`, `TASK\_PARAM\_DEFINITION`, `TASK\_CONCURRENT\_NUM`, `CREATED\_DATE`, `PRIORITY`, `DESCRIPTIOHN`, `IS\_ENABLE`, `STATUS`, `LAST\_EXEC\_TIME`, `LAST\_EXEC\_PARAM`) values('767959b1-e296-11e5-8723-0050569d49b2','Job\_D','D','2016-03-01 00:00:00','2016-08-01 00:00:00','yyyy-MM-dd 8:12:12','[ { \"name\": \"KETTLE\_PATH\", \"defaultValue\": \"/opt/kettle/data-integration\",\"increment\": -1,\"canAdditive\": false,\"type\": \"string\",\"unit\": \"\"},{\"name\": \"yesterday\",\"defaultValue\": \"2016-03-08\",\"increment\": -1,\"canAdditive\": true,\"type\": \"date\",\"unit\": \"dd\"}]','2','2016-03-05 13:52:36','1','test job','1','R','2016-03-01 11:11:11','[{\"canAdditive\":false,\"defaultValue\":\"/opt/kettle/data-integration\",\"increment\":-1,\"name\":\"KETTLE\_PATH\",\"type\":\"string\",\"unit\":\"\",\"value\":\"/opt/kettle/data-integration\"},{\"canAdditive\":true,\"defaultValue\":\"2016-03-08\",\"increment\":-1,\"name\":\"yesterday\",\"type\":\"date\",\"unit\":\"dd\",\"value\":\"2016-02-19 00:00:00\"}]');

新建一个日任务（Task）

insert into `SCH\_TASK` (`ID`, `NAME`, `JOB\_ID`, `TASK\_TYPE`, `TASK\_COMMAND`, `DESCRIPTIOHN`, `CREATED\_DATE`, `IS\_ENABLE`, `TARGET\_DATABASE`, `TARGET\_MACHINE`, `TRY\_COUNT`, `FAIL\_COMMAND`, `FAIL\_COMMAND\_TYPE`, `TIMEOUT`) values('10','j\_t\_f\_financing\_FIN','767959b1-e296-11e5-8723-0050569d49b2','SHELL','','test task','2016-03-05 16:39:45','1',NULL,NULL,'3',NULL,NULL,'100');

# 初始化Job和Task数据

可以通过工具把原有Job和Task数据迁移到上面的数据表中

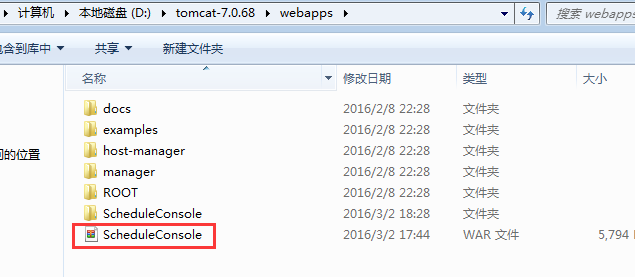
# 部署调度管理控制台

1. **安装tomcat**

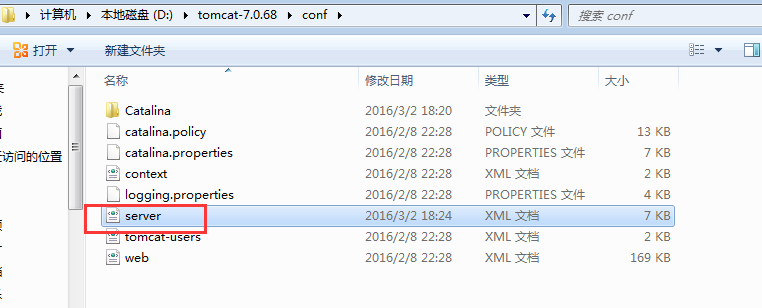
调度管理控制台是一个Web程序，因此需要Tomcat，下载Tomcat后解压到任意目录。首先确认机器已经安装java的jdk并设置好环境变量，然后进入tomcat的bin目录，双击”startup.sh”（window下为startup.bat）启动tomcat，如果没有异常则表明Tomcat安装成功。

1. **部署控制台war包**

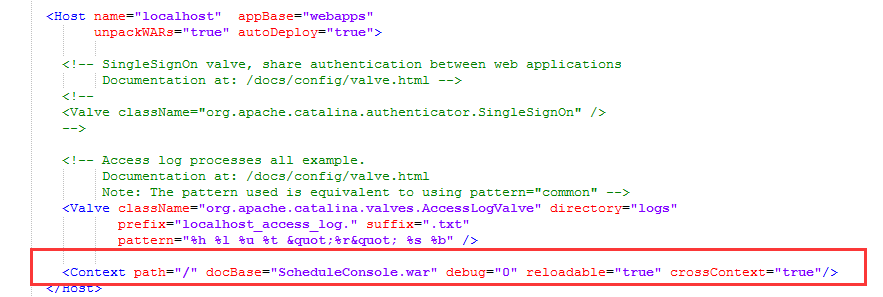
然后拷贝”ScheduleConsole.war”到Tomcat的webapps目录，如下所示：



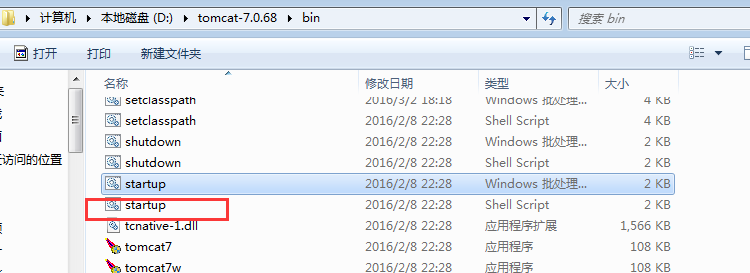
进入tomcat的conf目录，编辑server.xml文件



添加如下内容：



进入tomcat的bin目录，双击”startup.sh”（window下为startup.bat）启动tomcat：



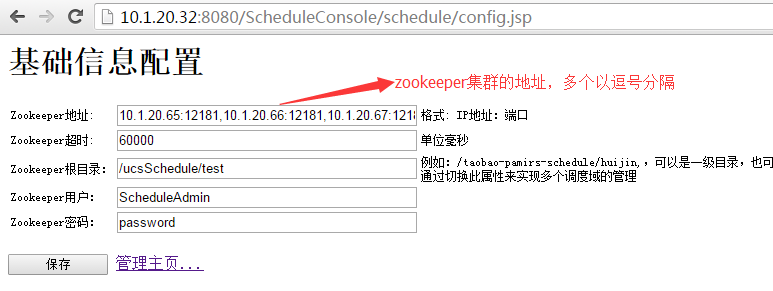
打开浏览器，输入地址：

http://您的机器IP:8080/ScheduleConsole/schedule/config.jsp

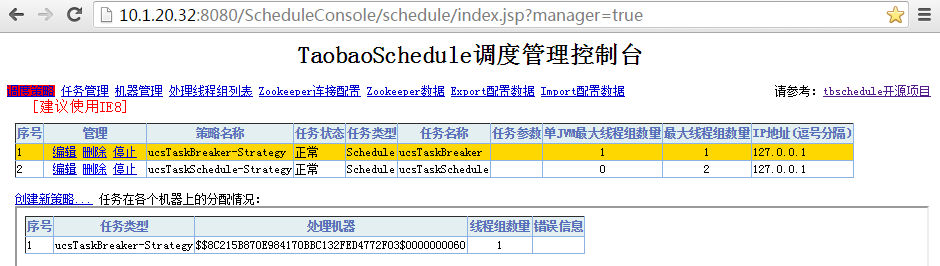
如果能够打开网页说明已经部署成功。

1. **配置Zookeeper相关参数**

Schedule框架需要Zookeeper来管理任务的元数据，因此需要正确的配置Zookeeper的地址和参数，如下图所示，配置成您实际环境的参数即可：



配置好相关参数后保存，然后点击”管理主页”便可进入管理控制台（页面上的数据是我添加的，你的页面可能没有）。



# 配置调度任务和策略

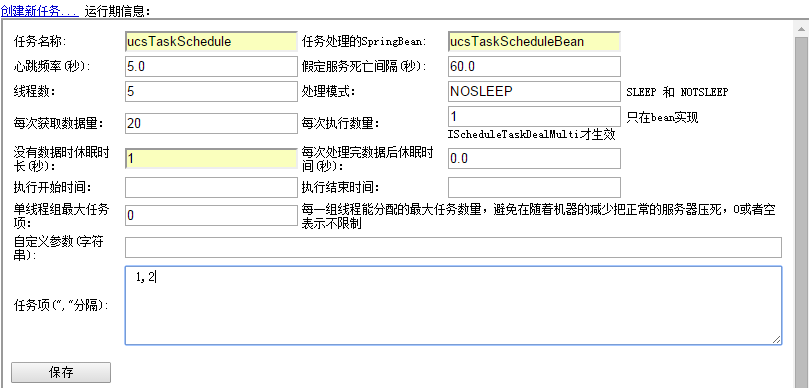
1. **添加调度任务**

点击”任务管理”菜单进行添加任务。我们主要是配置两个任务：

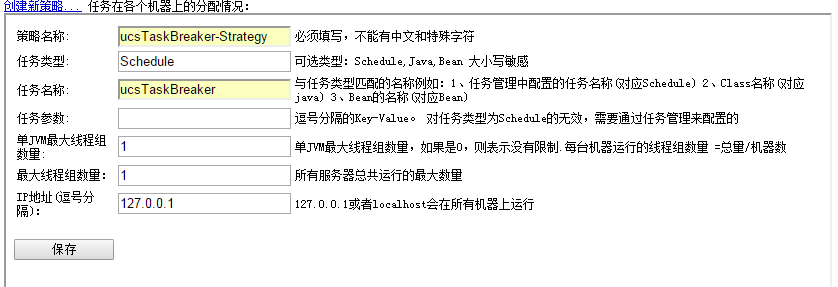
1. ucsTaskBreaker:负责解析Task依赖并修改为可执行状态。因为只是简单的扫描状态和修改状态，为了避免多线程并发造成数据库死锁，我们需要把它配置成单线程。

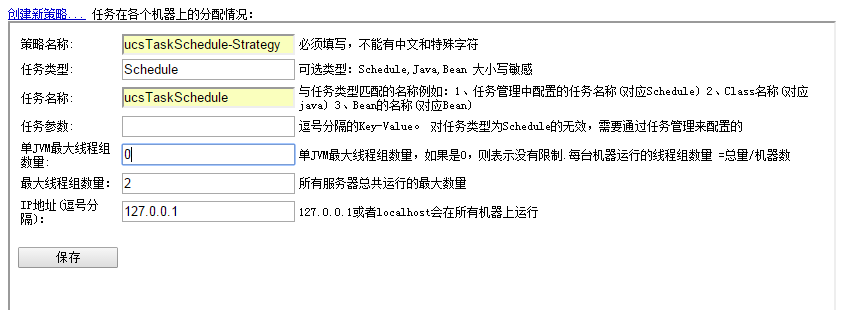


1. ucsTaskSchedule:负责执行已经可以执行的Task，我们可以配置成多线程并发执行。

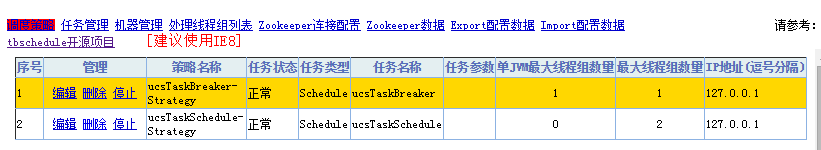
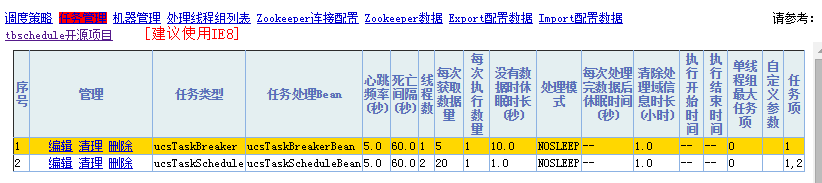


1. **添加调度策略**





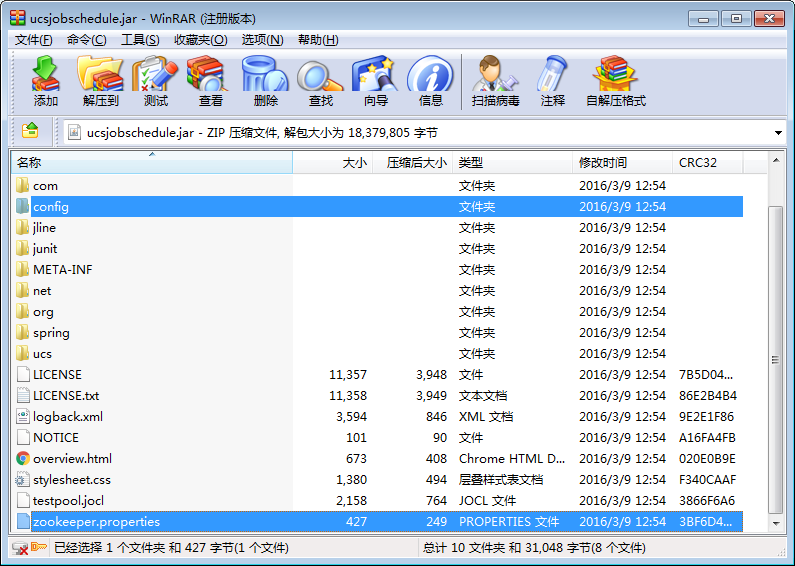
创建后的任务和策略情况如下所示：



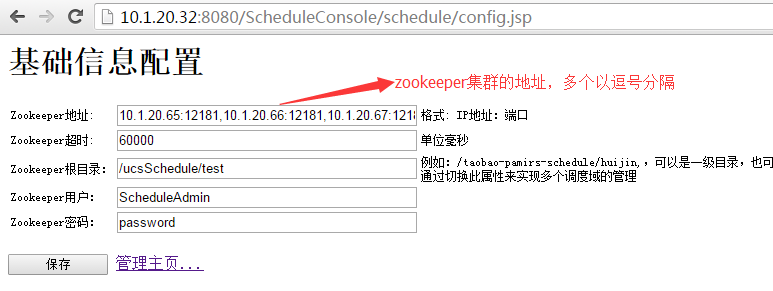
# 部署程序

拷贝” ucsjobschedule.jar”到机器的任意目录，然后需要您手动修改Zookeeper以及数据库连接配置。

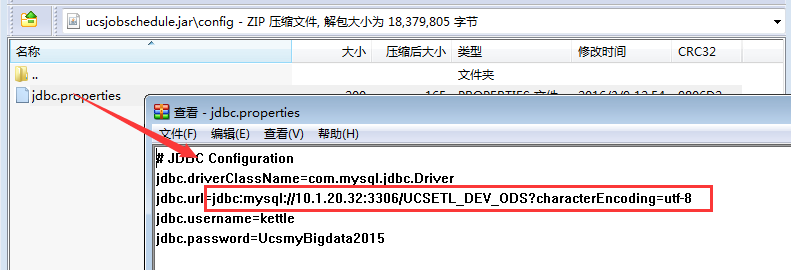
首先用解压工具解压” ucsjobschedule.jar”，找到下图高亮的两个东东：



1、打开”zookeeper.properties”,编辑Zookeeper的相关参数，与下图中的配置（前面已经配置过的）保持一致：



2、进入config目录，修改”jdbc.properties”文件中的数据库地址为实际的数据库连接地址：



最后重新压缩成.jar即可（也可以找我直接打包成目标环境的jar包，就不需要您手动修改参数了）。

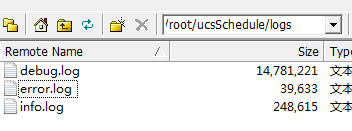
# 运行与监测

在” ucsjobschedule.jar”所在的目录开启终端控制台，输入以下命令启动程序：

**Java –jar ucsjobschedule.jar**

程序便会注册到Zookeeper，然后框架会根据您先前配置的任务和调度策略进行任务调度了。

开始任务调度后，在” ucsjobschedule.jar”所在的目录会看到一个logs文件夹，里面记录了程序的相关日志，通过查看日志便可知道程序的运行情况，如我的机器：



当数据表中有可以执行的Job时，程序便会调度执行Job所有Task。

您可以通过管理控制台查看任务的情况：

