# 简介

这个文档是我边操作边写的文档，把学到的东西持久化并建立索引，索引算法和常用的一些东西已经缓存在脑袋里，这也算是三个星期的总结吧；这个文档重点按照搭建部署、开发、遇到问题的目录方式写的。

# Gaea用到的一些技术

Maven管理

Sping、MyBatis、zookeeper、dubbo 实现类似mvc架构

Redis 内存存储

Oracle或mysql 数据库

Mapreduce hadoop-分布式运算框架之一

Jetty servlet容器

整个项目采用的是分布式思想，分布式就是类似多个人同时干一件事

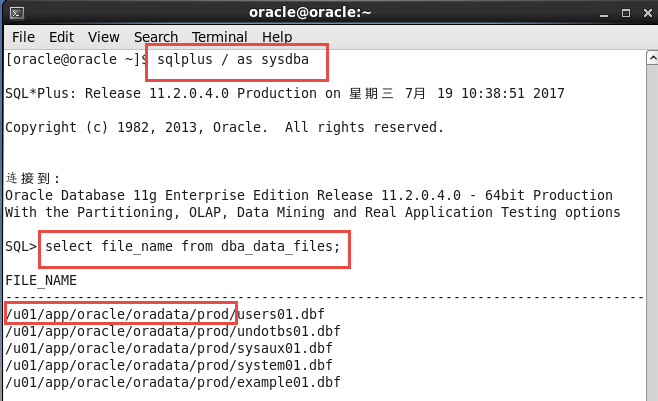
# 搭建部署

## 数据库配置

**1.首先查看表空间所在路径**

sqlplus / as sysdba

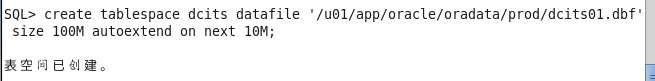
select file\_name from dba\_data\_files;



最后红框中的内容为数据文件存放路径

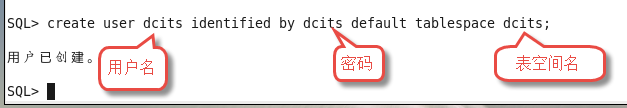
**2.然后创建表空间**

create tablespace dcits datafile '/u01/app/oracle/oradata/prod/dcits01.dbf' size 100M autoextend on next 10M;



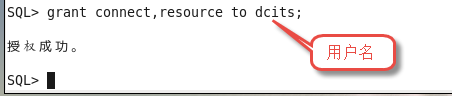
之后创建用户并指定表空间

create user dcits identified by dcits default tablespace dcits;



**3.授权**

grant connect,resource to dcits;

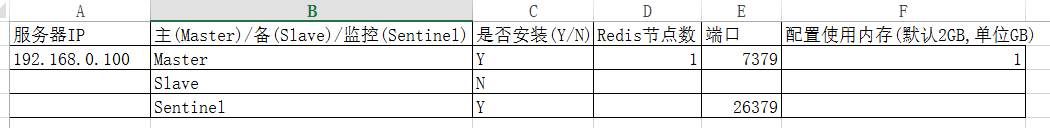


至此，创建用户及表空间完毕。

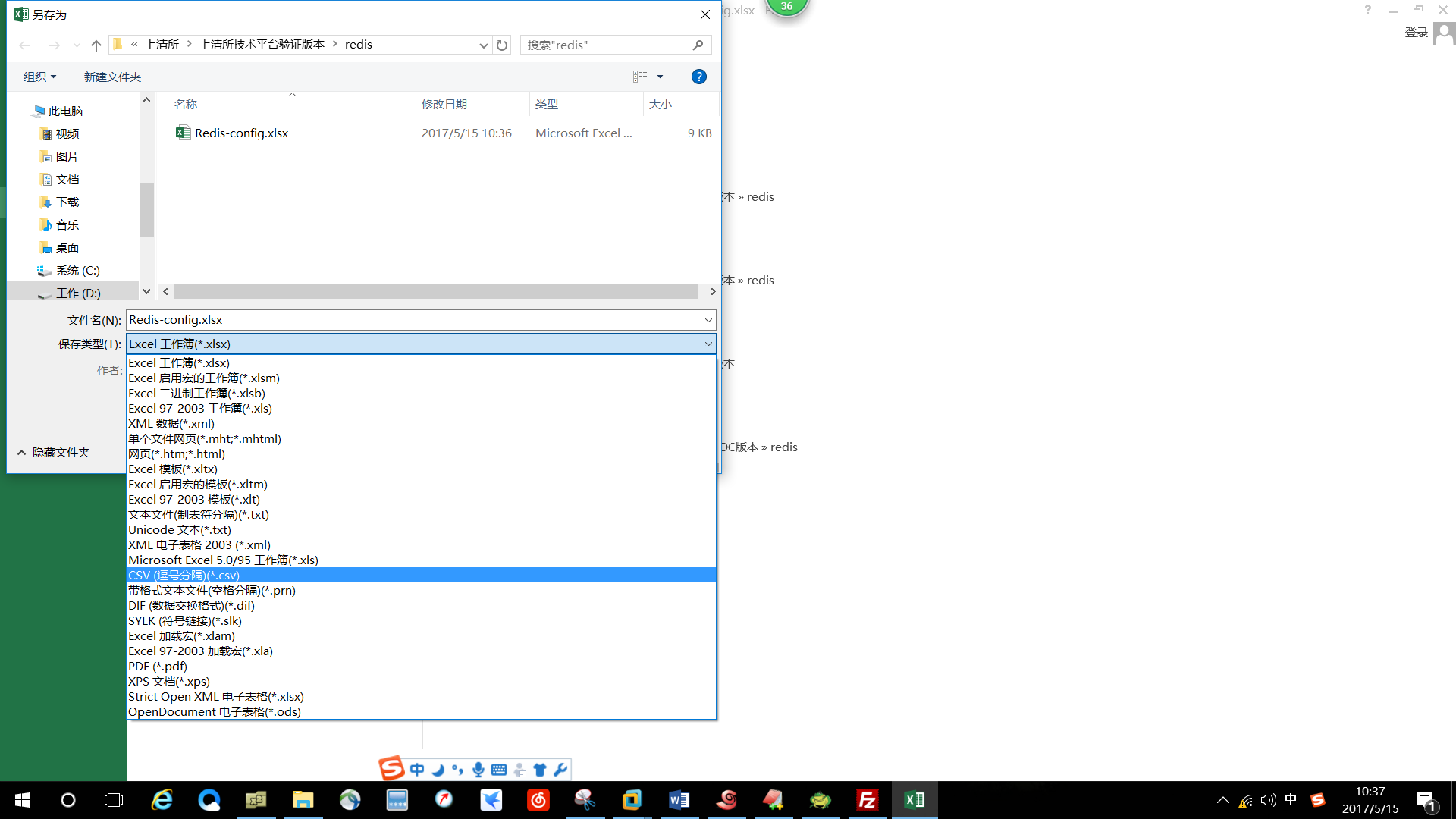
最后用开发工具连接对应的数据库即可导入sql文件操作

## Redis部署（单节点部署）

1. 打开Redis-config.xlsx，规划redis服务器安装的服务器。下图是单机节点只安装哨兵Sentinel和Master主节点，从Slave节点不安装。

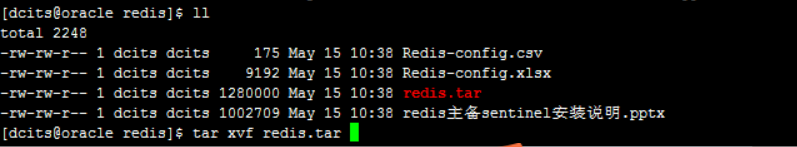


1. 将表格内容导出为csv文件Redis-config.csv



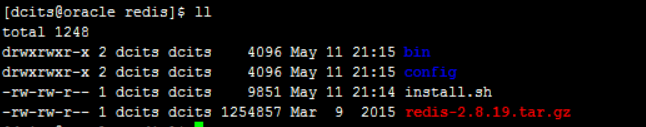
1. 将redis文件夹上传到服务器。进入redis文件夹，解压redis.tar,如下图所示：

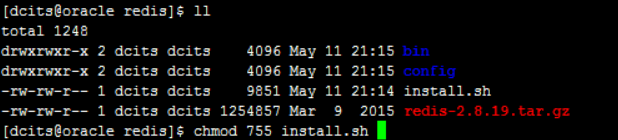
./tar xvf redis.tar



1. 进入解压后的redis文件夹，并给install.sh赋权限：

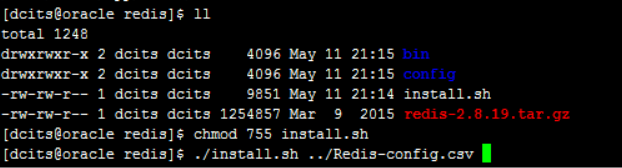
chmod 755 install.sh



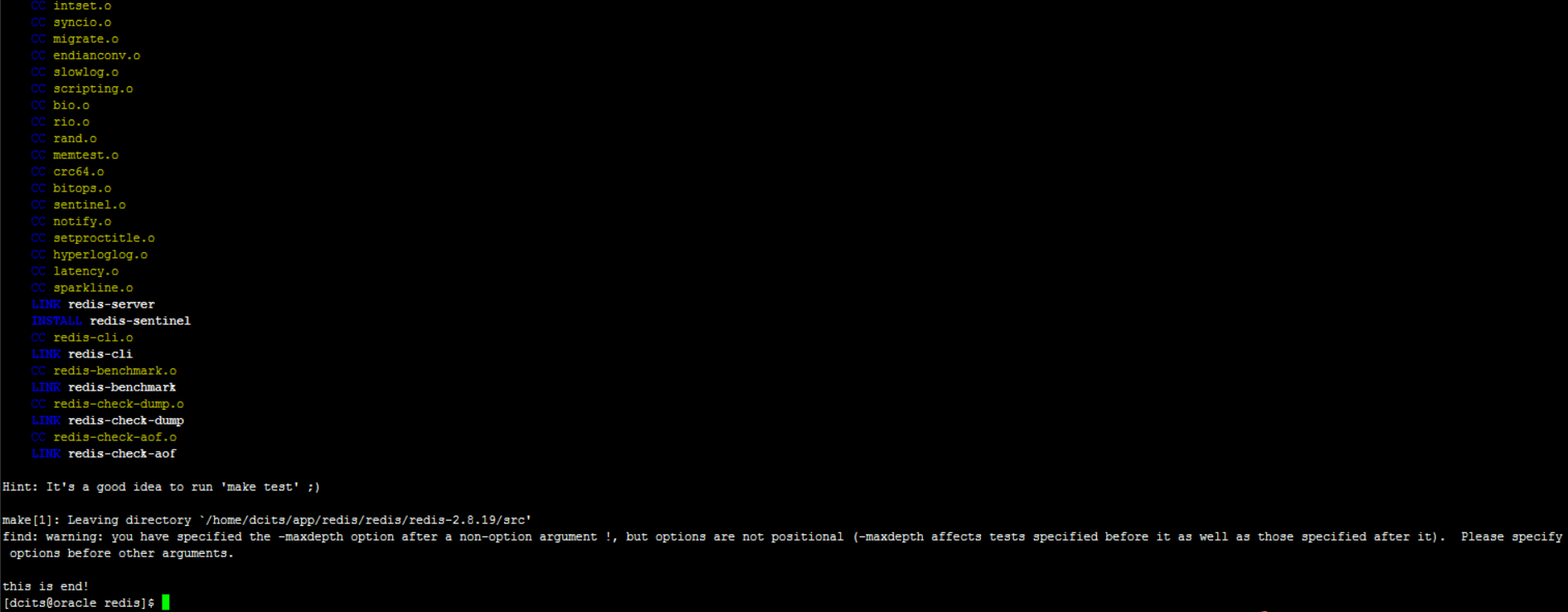


1. 执行安装redis服务：

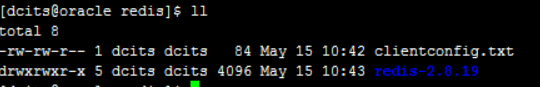
./install.sh ../Redis-config.csv



1. 界面执行this is end!安装结束，如下图所示：



1. 安装后的目录，其中clientconfig.txt对应galaxy.redis.properties中redis服务器配置的地址。

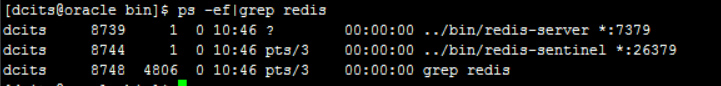


1. 执行启停Redis和哨兵，进入redis-2.8.19/bin目录。

执行启动命令：./startAll.sh

执行停止命令：./stopAll.sh

查看redis服务的进程，若如下图所示，则表明启动成功



## Zookeeper部署（单节点部署）

1. 解压ZOOKEEPER安装包zookeeper-3.4.3.zip到服务器目录：

unzip zookeeper-3.4.3.zip

1. 配置ZOOKEEPER监听端口：

Zookeeper默认监听端口为2181，如需修改，可进入zoo.cfg修改对应监听端口即可。

1. 配置脚本执行权限：

进入bin目录，增加zookeeper脚本执行权限，如下所示

chmod 755\*.sh

1. 执行启停ZOOKEEPER，进入bin目录：

启动命令：./zkServer.sh start

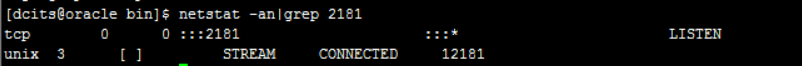
停止命令：./zkServer.sh stop

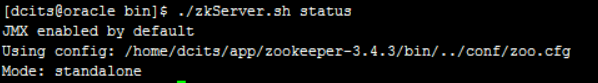
1. 查看ZOOKEEPER是否启动成功：

执行netstat –an|grep 2181 或者

执行./zkServer.sh status

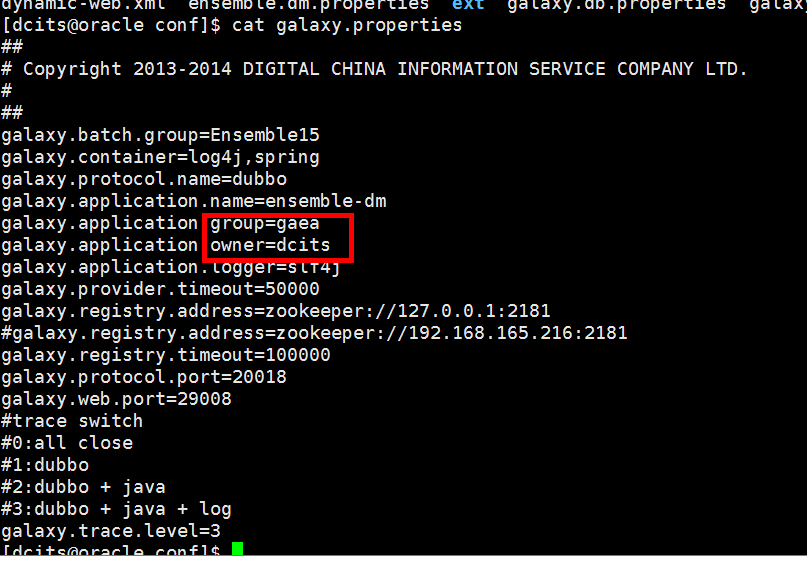
若显示如下图结果，说明启动成功：





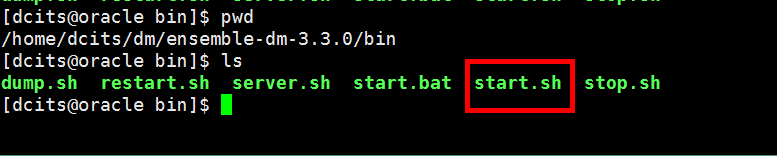
## Dm部署

1. 上传DM项目到虚拟机中
2. 修改galaxy.db.properties中db额数据库配置
3. 修改galaxy.properties中zk地址配置
4. 修改如下跟gaea中的一致





6.启动项目，若是windows则是双击start.bat



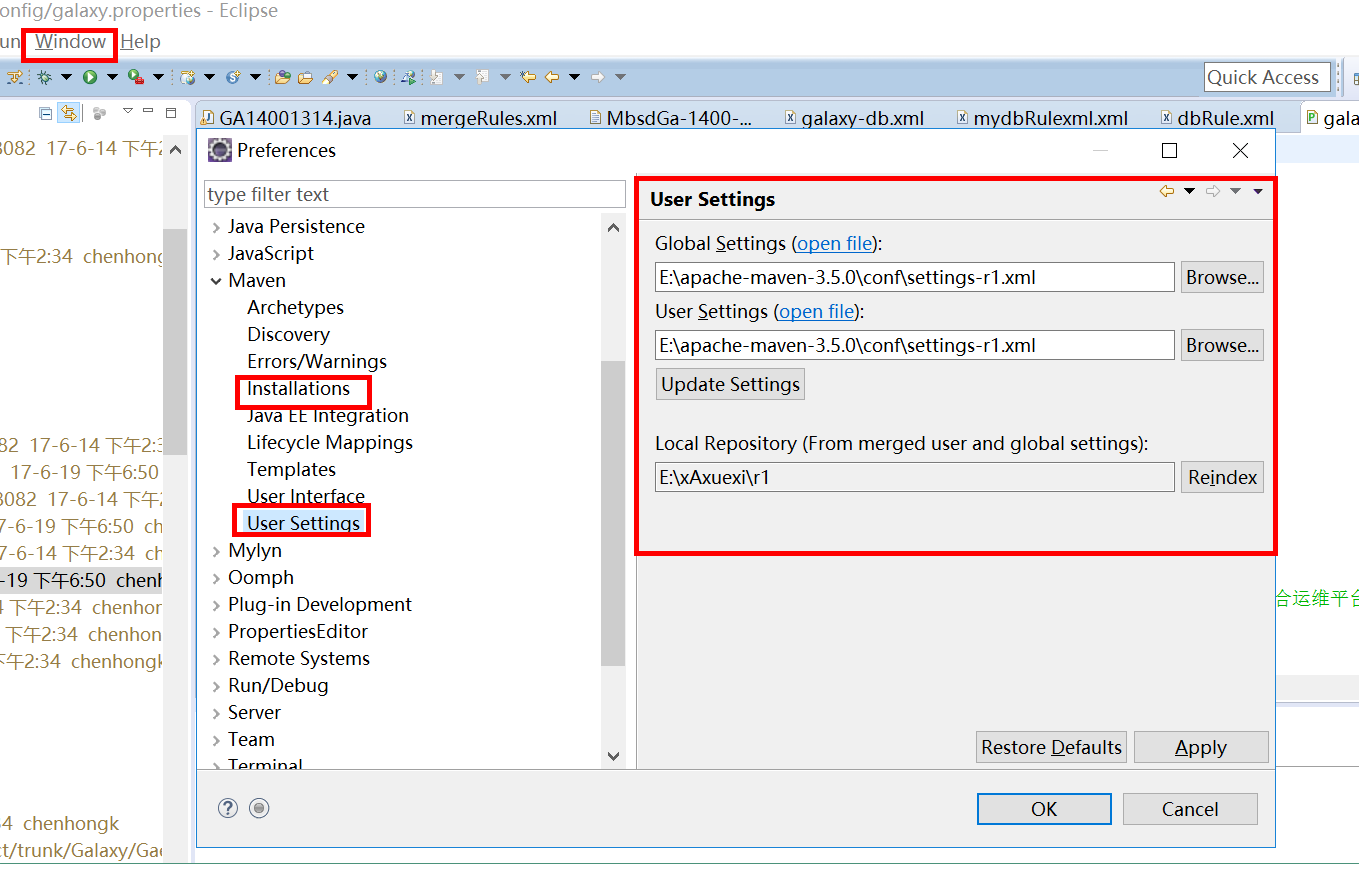
## Om部署

与dm的搭建相同，参照dm搭建

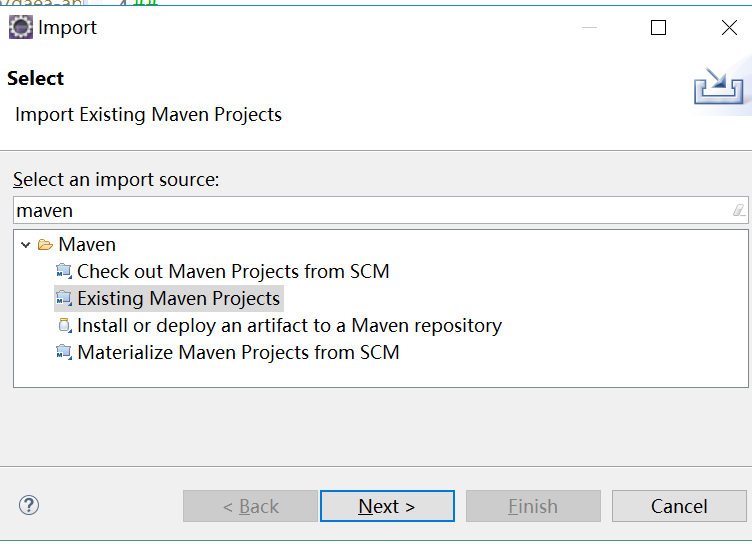
## gaea搭建

### eclipse项目导入

1. 进入eclipse首先配置maven如下

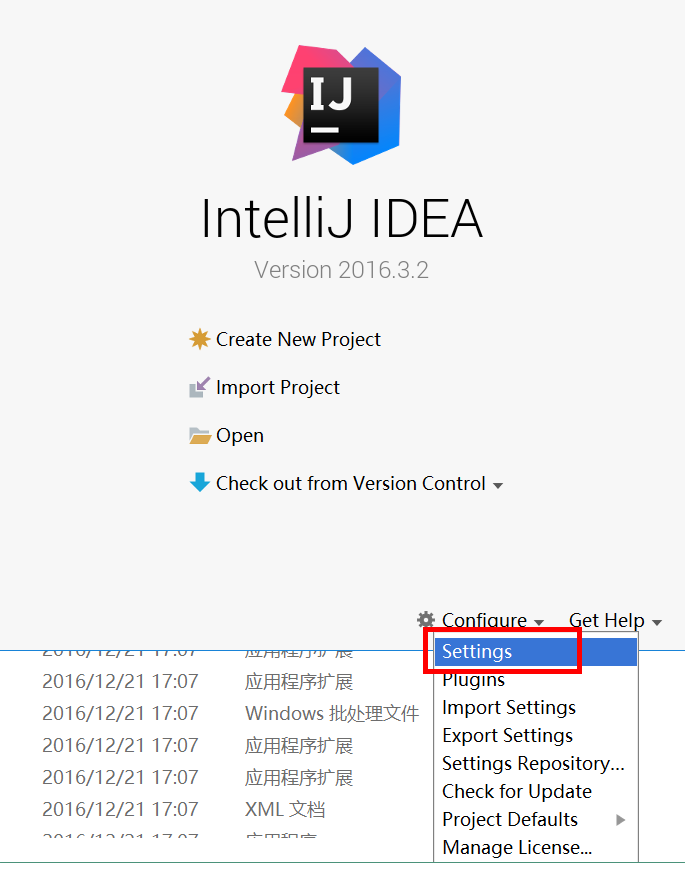


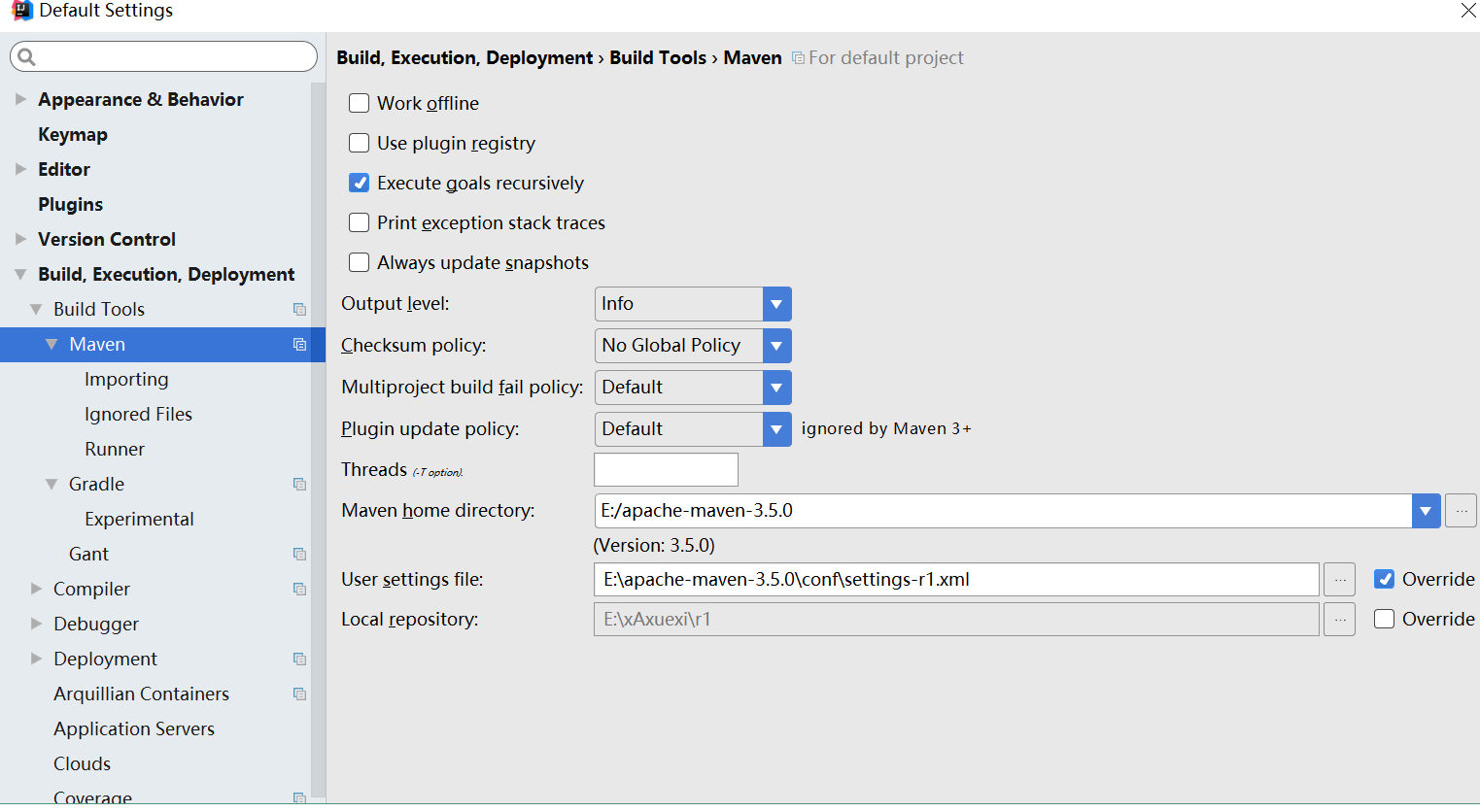
1. 导入maven工程即可



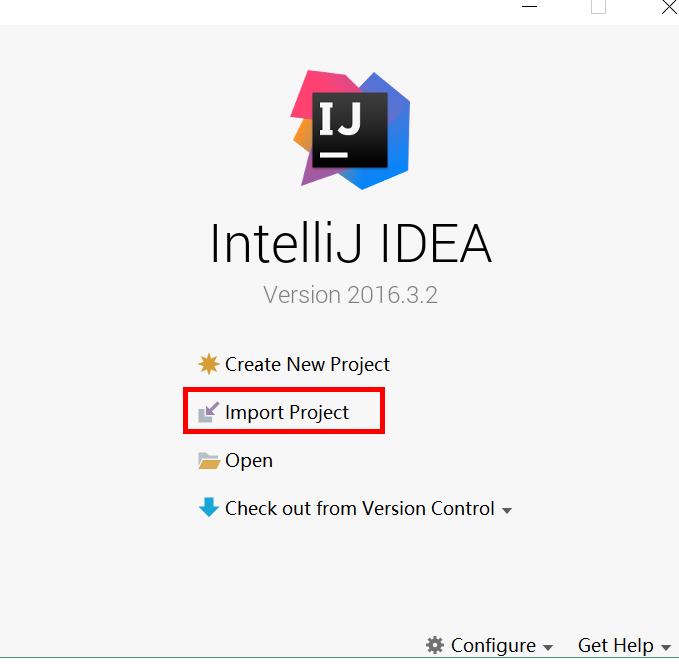
### IDEA项目导入

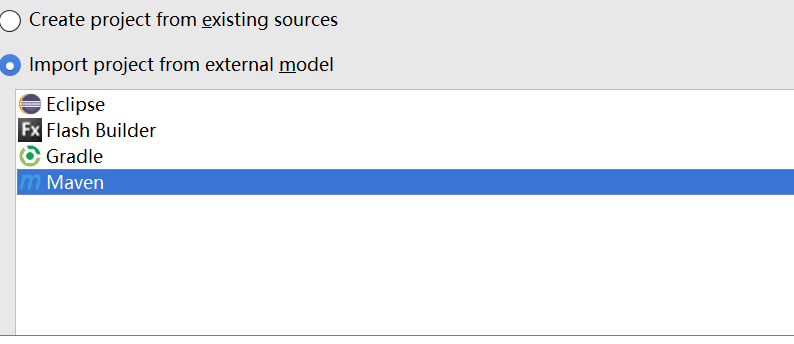
1. 配置maven如下





1. 导入maven项目一路next即可





## Gaea部署

去到gaea项目下面使用maven命令

mvn clean compile 清除再编译

mvn package 打包命令，首先pom得引入maven-assembly-plugin插件即可打包

-DskipTests 不执行测试用例，但是编译测试用例到target/test-classes下

-Dmaven.test.skip=true 不执行测试用例，也不编译测试用例

打好的jar包conn和app在对应的target中丢到服务器解压启动

# 模块开发

## Stria开发

就是使得外部请求能够访问接口的一个操作，其整个开发的思想还是mvc的思想

### 首先配置好dm用户以及系统信息





### 导入接口文档生成相应的代码

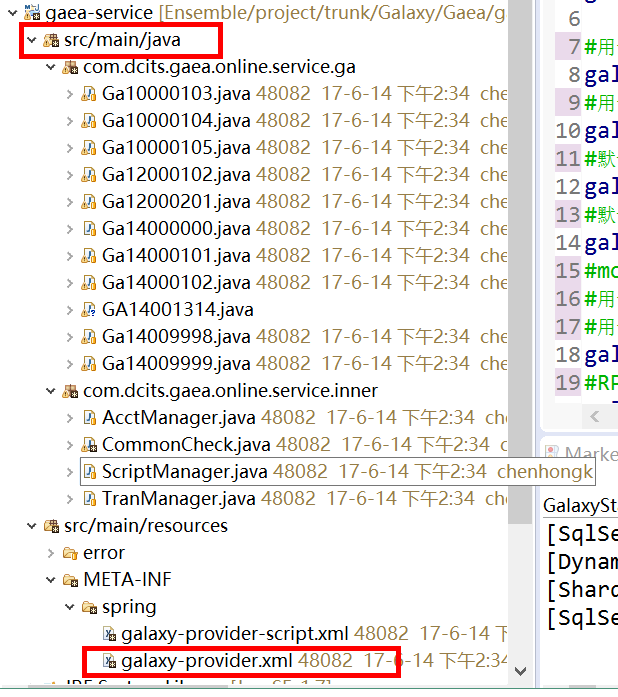


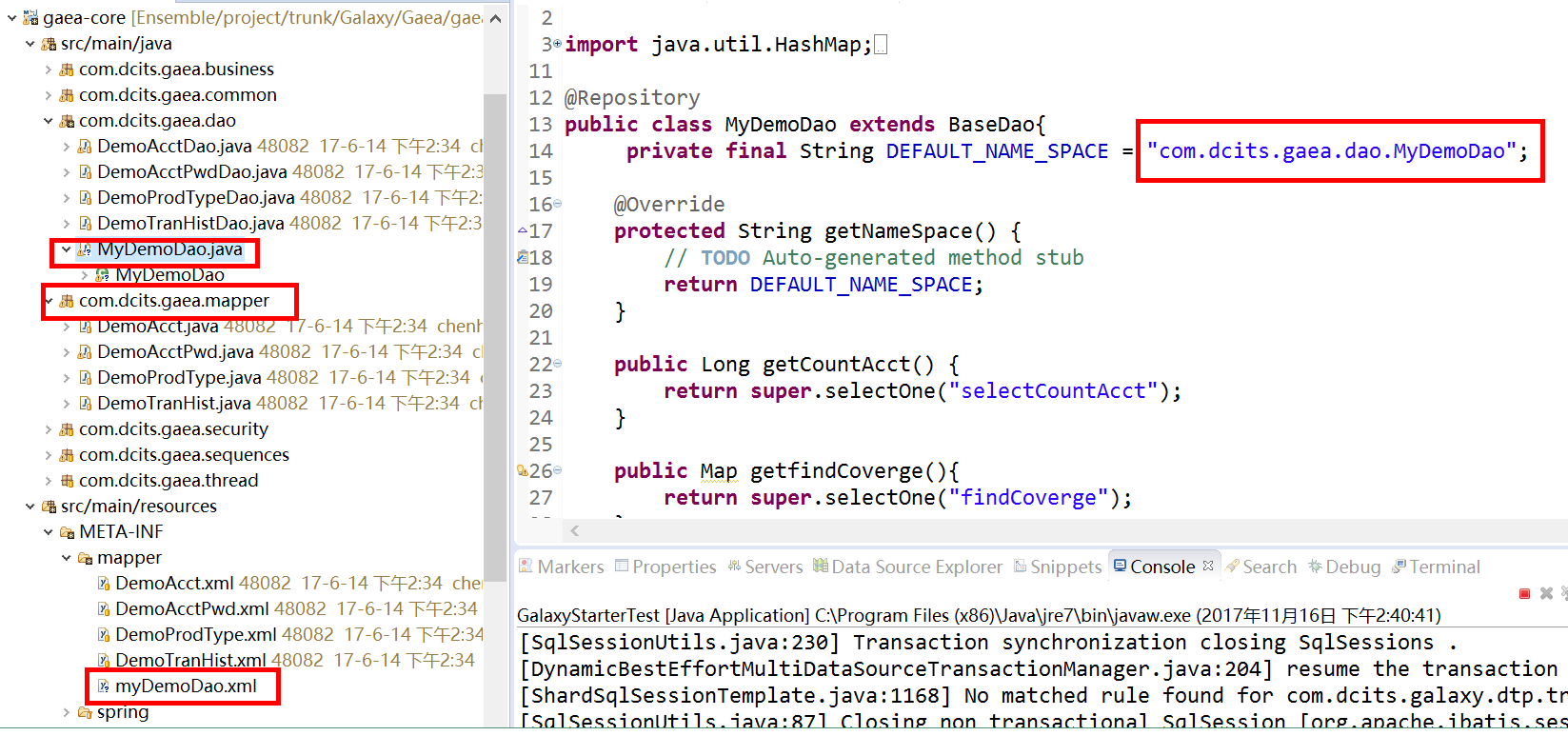


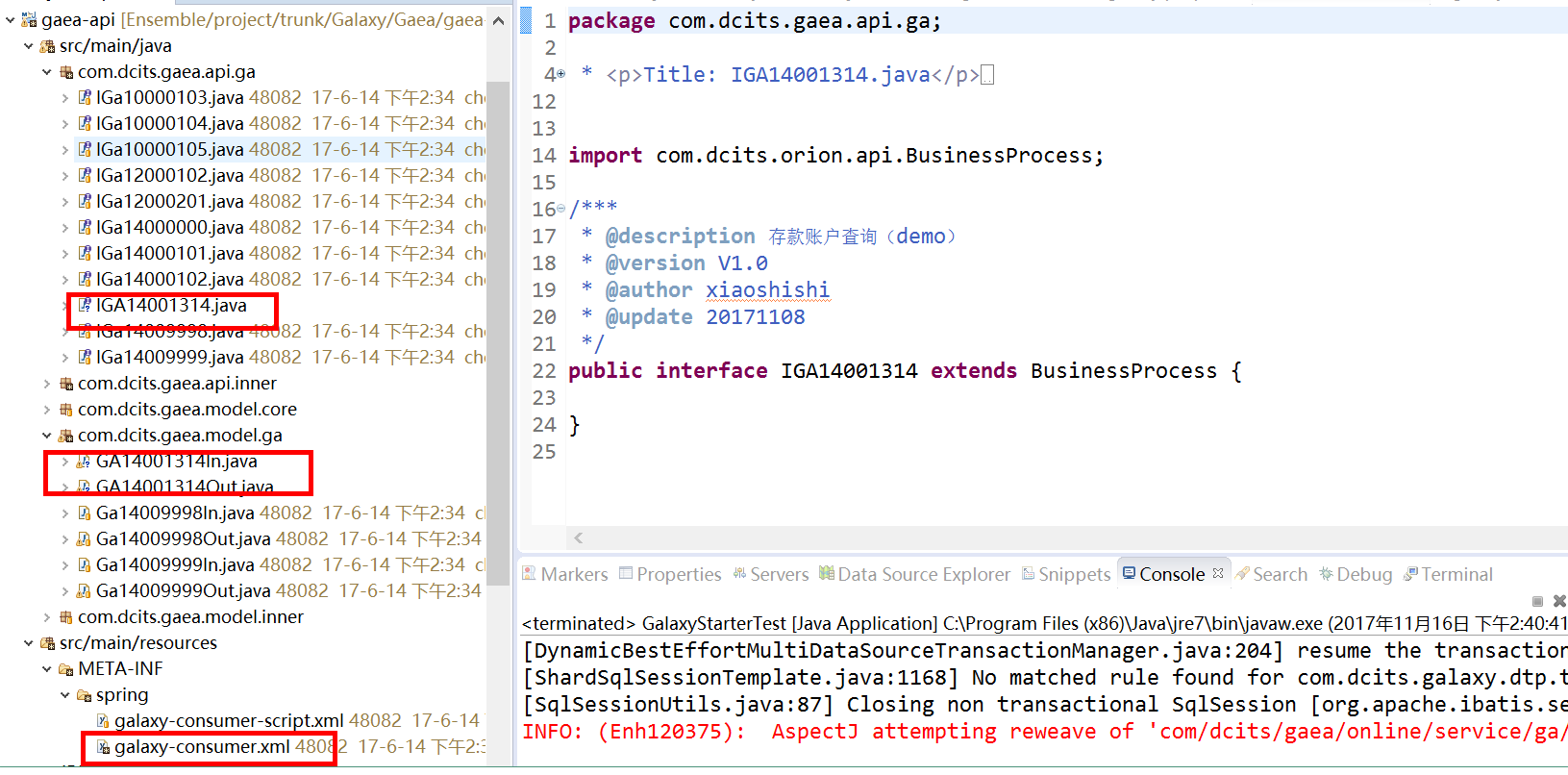
### 生成数据库表对应的db代码



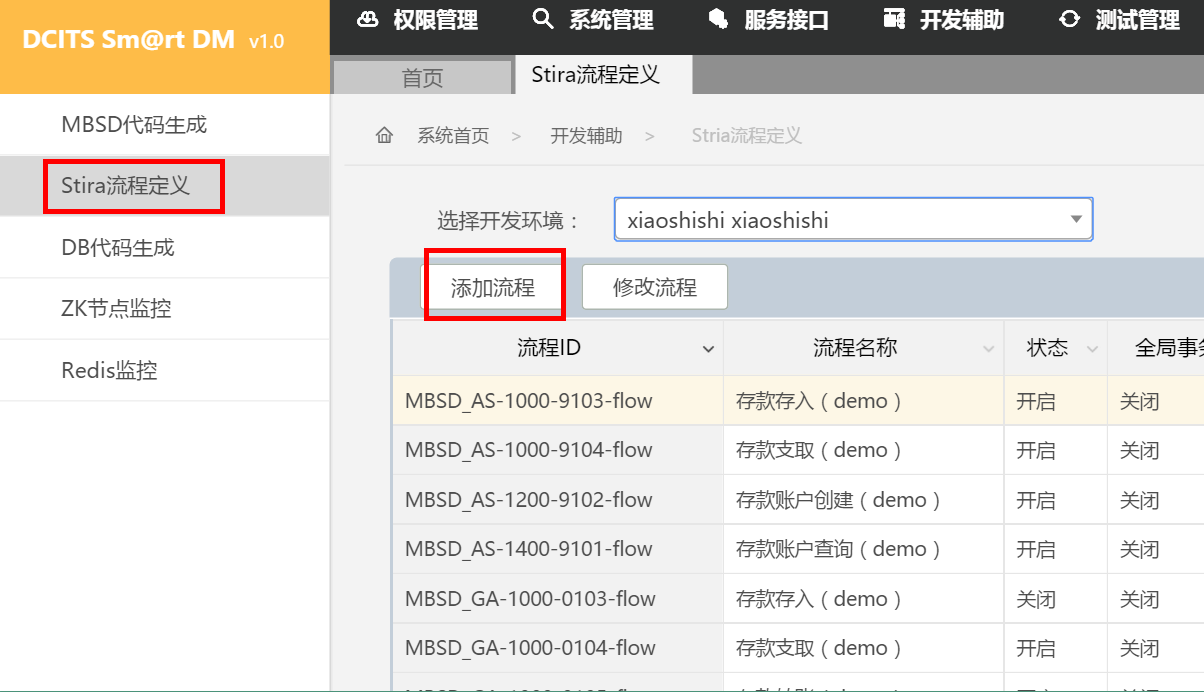
### 将代码复制到如下对应目录中







### 配置Stria流程图



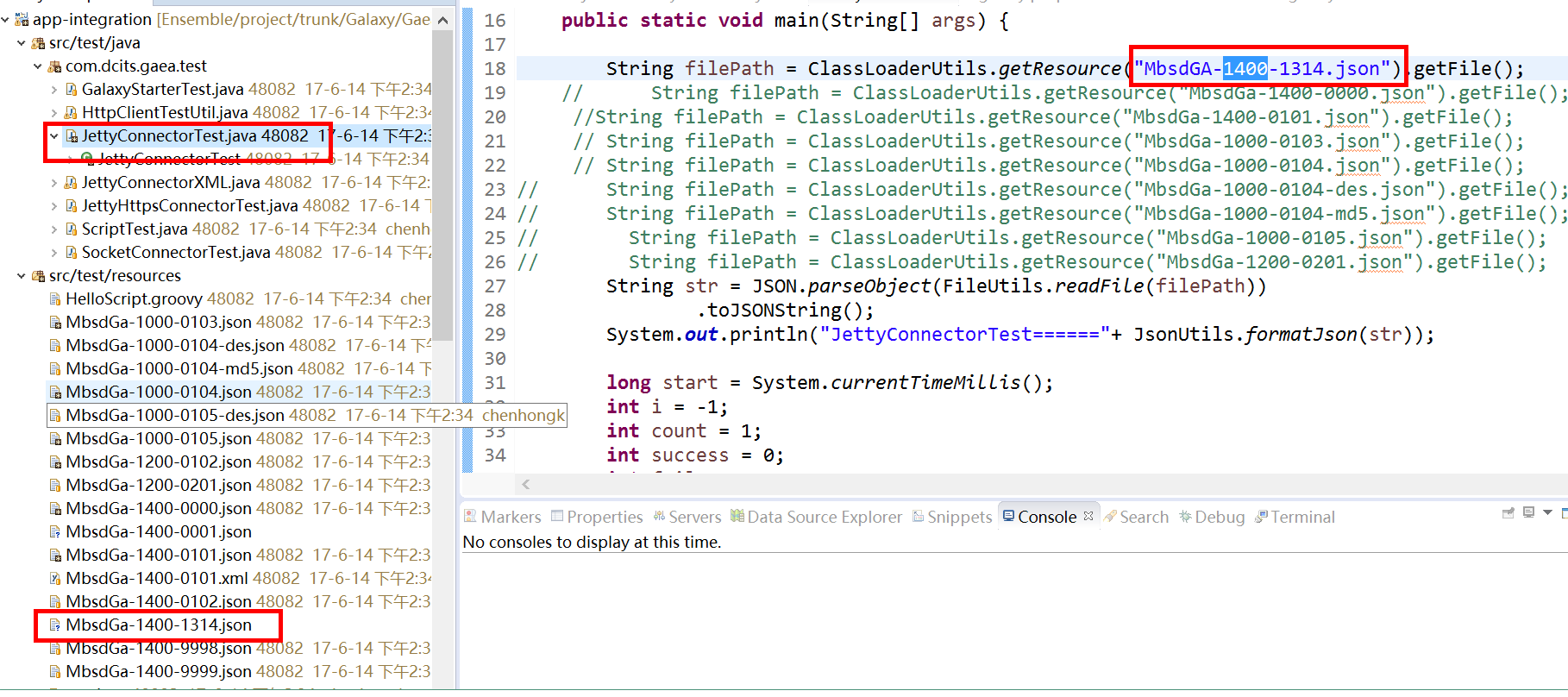




其中传入参数\*代表入参是整个入参bean对象里面对应的所有值

### 启动项目后测试

运行这个mian函数如果通了说明成功了



## 批处理开发

我理解的批处理，从开发角度，某个时间段拿一批数据进行某种业务操作，例如每天银行要进行一天的一个对账处理。

### 操作对应批处理表

BATCH\_DEF 批处理交易表

BATCH\_RUN\_TASK JOB生成的执行Task状态

BATCH\_RUN\_JOB

BATCH\_STATUS JOB状态

BATCH\_TIMER\_DEF 定时任务定义表

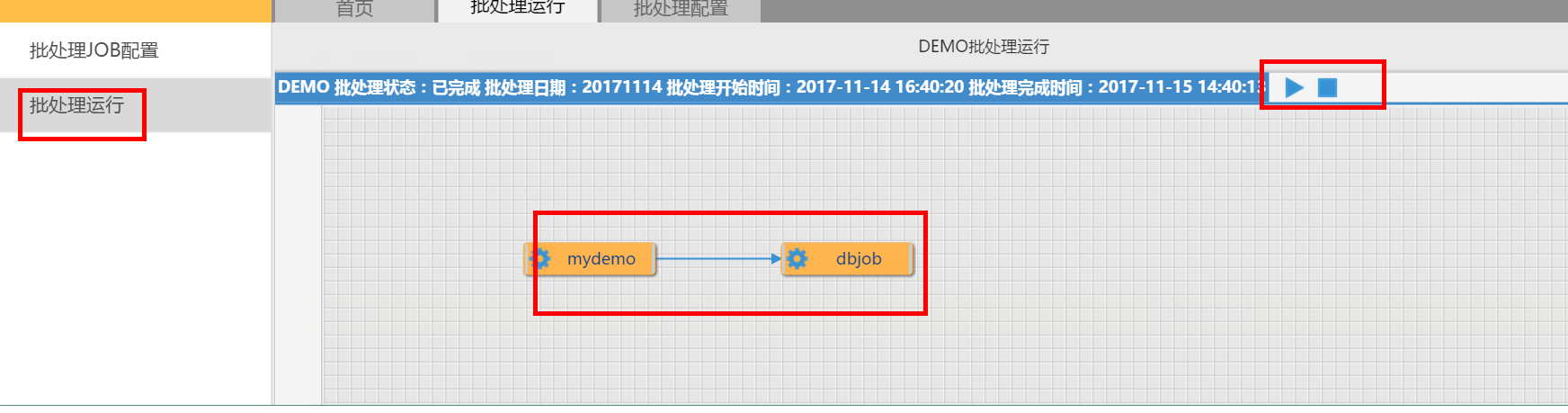
### 配置批处，理流程图并且配置好对应的类和方法







### 启动批处理查看控制台输出即可



### 定时任务运行批处理开发

配置Spring

|  |
| --- |
| <bean id=*"zkThreads"* class=*"com.dcits.orion.core.zookeeper.ZkThreads"*>  <property name=*"threadNames"*>  <list> <value>com.dcits.orion.batch.common.TimerThread</value>  </list>  </property>  </bean> |
| 相关的表BATCH\_TIMER\_DEF |
| quartz定时任务时间设置  这些星号由左到右按顺序代表 ： \* \* \* \* \* \* \*  格式： [秒] [分] [小时] [日] [月] [周] [年]  序号 说明  是否必填 允许填写的值 允许的通配符  1 秒 是 0-59 , - \* /  2 分 是 0-59  , - \* /  3 小时 是 0-23 , - \* /  4 日 是 1-31 , - \* ? / L W  5 月 是 1-12 or JAN-DEC , - \* /  6 周 是 1-7 or SUN-SAT , - \* ? / L #  7 年 否 empty 或 1970-2099 , - \* /  通配符说明:  \* 表示所有值. 例如:在分的字段上设置 "\*",表示每一分钟都会触发。  ? 表示不指定值。使用的场景为不需要关心当前设置这个字段的值。例如:要在每月的10号触发一个操作，但不关心是周几，所以需要周位置的那个字段设置为"?" 具体设置为 0 0 0 10 \* ?  - 表示区间。例如 在小时上设置 "10-12",表示 10,11,12点都会触发。  , 表示指定多个值，例如在周字段上设置 "MON,WED,FRI" 表示周一，周三和周五触发  / 用于递增触发。如在秒上面设置"5/15" 表示从5秒开始，每增15秒触发(5,20,35,50)。在月字段上设置'1/3'所示每月1号开始，每隔三天触发一次。  L 表示最后的意思。在日字段设置上，表示当月的最后一天(依据当前月份，如果是二月还会依据是否是润年[leap]), 在周字段上表示星期六，相当于"7"或"SAT"。如果在"L"前加上数字，则表示该数据的最后一个。例如在周字段上设置"6L"这样的格式,则表示“本月最后一个星期五"  W 表示离指定日期的最近那个工作日(周一至周五). 例如在日字段上设置"15W"，表示离每月15号最近的那个工作日触发。如果15号正好是周六，则找最近的周五(14号)触发, 如果15号是周未，则找最近的下周一(16号)触发.如果15号正好在工作日(周一至周五)，则就在该天触发。如果指定格式为 "1W",它则表示每月1号往后最近的工作日触发。如果1号正是周六，则将在3号下周一触发。(注，"W"前只能设置具体的数字,不允许区间"-").  小提示  'L'和 'W'可以一组合使用。如果在日字段上设置"LW",则表示在本月的最后一个工作日触发(一般指发工资 )  # 序号(表示每月的第几个周几)，例如在周字段上设置"6#3"表示在每月的第三个周六.注意如果指定"#5",正好第五周没有周六，则不会触发该配置(用在母亲节和父亲节再合适不过了)  小提示  周字段的设置，若使用英文字母是不区分大小写的 MON 与mon相同.  常用示例:  0 0 12 \* \* ? 每天12点触发  0 15 10 ? \* \* 每天10点15分触发  0 15 10 \* \* ? 每天10点15分触发  0 15 10 \* \* ? \* 每天10点15分触发  0 15 10 \* \* ? 2005 2005年每天10点15分触发  0 \* 14 \* \* ? 每天下午的 2点到2点59分每分触发  0 0/5 14 \* \* ? 每天下午的 2点到2点59分(整点开始，每隔5分触发)  0 0/5 14,18 \* \* ? 每天下午的 2点到2点59分(整点开始，每隔5分触发)  每天下午的 18点到18点59分(整点开始，每隔5分触发)  0 0-5 14 \* \* ? 每天下午的 2点到2点05分每分触发  0 10,44 14 ? 3 WED 3月分每周三下午的 2点10分和2点44分触发 （特殊情况，在一个时间设置里，执行两次或 两次以上的情况）  0 59 2 ? \* FRI 每周5凌晨2点59分触发；  0 15 10 ? \* MON-FRI 从周一到周五每天上午的10点15分触发  0 15 10 15 \* ? 每月15号上午10点15分触发  0 15 10 L \* ? 每月最后一天的10点15分触发  0 15 10 ? \* 6L 每月最后一周的星期五的10点15分触发  0 15 10 ? \* 6L 2002-2005 从2002年到2005年每月最后一周的星期五的10点15分触发  0 15 10 ? \* 6#3 每月的第三周的星期五开始触发  0 0 12 1/5 \* ? 每月的第一个中午开始每隔5天触发一次  0 11 11 11 11 ? 每年的11月11号 11点11分触发(光棍节) |

也可以配置自定义启动的类

## Dal分库开发

就是去多个数据库中执行sql的问题

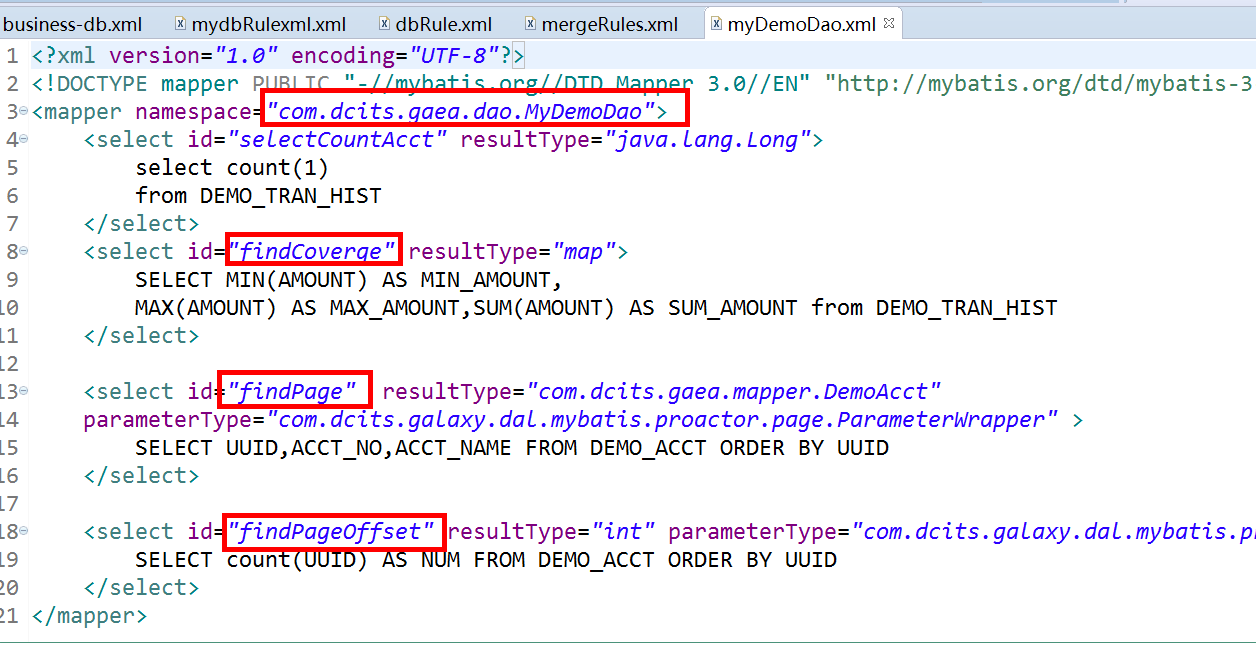
### 路由配置和多库查询合并结果集

|  |
| --- |
| <!-- 配置路由规则开始 -->  <bean id=*"internalRouter"*  class=*"com.dcits.galaxy.dal.mybatis.router.adapter.config.AdapterRouterXmlFactoryBean"*>  <!-- functionsMap是在使用自定义路由规则函数的时候使用 -->  <property name=*"functionsMap"*>  <map>  <entry key=*"router"* value-ref=*"hashFunction"*></entry>  </map>  </property>  <property name=*"configLocations"* value=*"classpath\*****:dbRule/\*.xml****"*/>  <!-- 不想使用jar的情况，可以修改这个地方配置到集中部署路径 -->  <property name=*"shardManager"* ref=*"shardManager"*/>  </bean>  <!-- 配置路由规则结束 -->  <!-- ============================== 查询处理 start ============================== -->  <bean id=*"mergeRulesFactory"*  class=*"com.dcits.galaxy.dal.mybatis.merger.config.MergeRulesFactoryBean"*>  <property name=*"locations"* value=*"classpath\*****:mergeRules/\*.xml****"*/>  </bean>  <!-- ============================== 查询处理 end ============================== --> |
|  |
| <rules-setting>  <rules type=*"INDEX\_ROUTER"* expression=*"[router.mod(router.cutLast2(#),3)]"* shards=*"partition0,partition1,partition2"*>  <rule>  <namespace>com.dcits.gaea.dao.DemoAcctDao</namespace>  <field>acctNo</field>  <field>uuid</field>  </rule>  <！-- 这个配置是根据函数mod返回的结果来确定需要选择执行的库 -->  </rules>  <rules type=*"INDEX\_ROUTER"* shards=*"partition0,partition1,partition2"*>  <rule>  <sqlmaps>  <sqlmap>com.dcits.gaea.dao.MyDemoDao.findCoverge</sqlmap>  </sqlmaps>  </rule>  </rules>  <rules type=*"INDEX\_ROUTER"* shards=*"partition0,partition1,partition2"*>这个为db配置的3个库  <rule>  <sqlmaps>  <sqlmap>com.dcits.gaea.dao.MyDemoDao.findPage</sqlmap>  </sqlmaps>  </rule>  </rules>  </rules-setting> |

### 查出结果合并

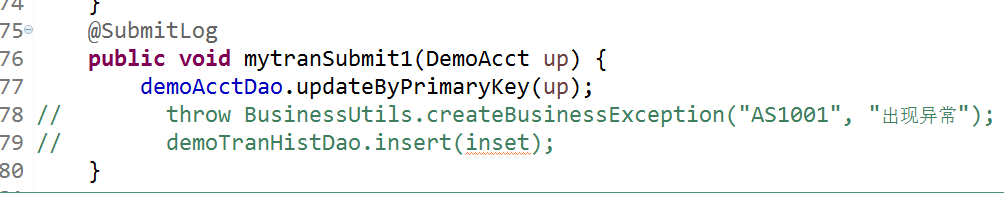
|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <merge-rules>  <merge-rule>  <sqlMap>com.dcits.gaea.dao.MyDemoDao.findCoverge</sqlMap>  <merger bean=*"aggregateMerger"*>  <columns>  <column alias=*"MIN\_AMOUNT"* aggregate=*"min"* />  <column alias=*"MAX\_AMOUNT"* aggregate=*"max"* />  <column alias=*"SUM\_AMOUNT"* aggregate=*"sum"* />  </columns>  <properties />  </merger>  </merge-rule>  <merge-rule>  <sqlMap>com.dcits.gaea.dao.MyDemoDao.findPage</sqlMap>  <proactor bean=*"pageProactor"*>  <columns>  <column alias=*"uuid"* desc=*"false"* />  </columns>    </proactor>  </merge-rule>  </merge-rules> |

### 配置mapper和dao



## 数据库事物开发

分为跨数据库(数据源)事物回滚和跨应用事物回滚(JVM),两个事物只需要如下在增删改的方法上面增加**@SubmitLog**



## Redis缓存开发

Spring的Xml配置

|  |
| --- |
| <bean id=*"ss"* class=*"com.dcits.galaxy.cache.global.RedisCache"*  p:name=*"data"* p:period=*"3"*/> |

方法上面增加注解如下，配置最好写在dao里面，增删改跟事物一起配套使用

|  |
| --- |
| @Service  **public** **class** CacheManagerDemo {  **private** **static** **final** Logger ***log*** = LoggerFactory  .*getLogger*(CacheManagerDemo.**class**);  @Resource  **private** MyDemoDao myDemoDao;    //value的data跟Spring的xml文件里面的一个配置要相同  //key为入参值，作为redis缓存中的key  //该注解是先取缓存，取不到则执行方法，最后将查到的结果存入缓存中  @Cacheable(value = "data",key="#params[acctNo]")  **public** DemoAcct selectOne(Map params){    System.***out***.println("select...");  DemoAcct aa= myDemoDao.selectOne("selectAcctInfoByAcctNo",params);  **return** aa;  }    //该注解是修改和删除缓存，null值就是删除缓存  @CacheEvict(value="data",key="#params[acctNo]")  **public** TwoTuple<Integer, DemoAcct> update2(Map params){  System.***out***.println("update...");  **int** i = myDemoDao.update("updateByPrimaryKey",params);  **return** **new** TwoTuple<Integer, DemoAcct>(i, **null**);  }  //该注解是将数据存入缓存中  @CachePut(value="data",key="#params[acctNo]")  **public** TwoTuple<Map, DemoAcct> update3(Map params){  System.***out***.println("update...");  DemoAcct aa= myDemoDao.selectOne("selectAcctInfoByAcctNo",params);  **return** **new** TwoTuple<Map, DemoAcct>(**null**, aa);  }  } |

# 遇到的问题以及解决方式

## 本地启动DM报错NoJdk异常

问题：如题

解决方法: 这是jdk安装过多导致冲突，

1.删掉jdk中bin目录下的几个java架包；2.卸载jdk然后使用解压版的重1新配置3环境变量，或者重新安装jdk

## 页面输入dm或om地址打开报错404，500，503等问题

问题：如题

解决方法：

1. dm中zk(zookeeper)、数据库db中要配置正确，其中dm中zk地址要和gaea中zk地址一样
2. 下面的配置要对上

|  |  |
| --- | --- |
| Gaea项目的  /app-integration/src/main/config/galaxy.properties | Dm或om项目  config/galaxy.properties |
| galaxy.application.name=gaea  galaxy.application.group=…  galaxy.application.owner=… | galaxy.batch.group=gaea  ….  …. |

## Linux中Zookeeper启动时候报没有权限错误

问题:如题

解决方法：由于自用户的权限问题不能进行写操作导致的错误

修改zookeeper-3.4.6/conf/zoo.cfg 文件中

dataDir=/home/dcits/zookeeper

红色部分路径为当前启动用户可写操作的路径

启动之前用root给这个自用户赋予写的权限，可以用**chown –R 子用户 路径** 给子用户赋予权限

## Gaea和dm之间通讯不通的问题

问题:由于虚拟机和本机的通讯方式是通过虚拟机代理配置的固定地址，不是桥接模式也不是net模式，所以虚拟机不能ping本机地址，而每次交互的时候需要连接本机地址

解决方式:请检查一下三个配置

1. 关闭windows防火墙，关闭虚拟机中linux防火墙或者修改防火墙相关配置文件开放zk的端口
2. Gaea中/app-integration/src/main/config/galaxy.properties 增加或修改

galaxy.protocol.host=**192.168.56.1** 其中红色部分地址是虚拟机能ping的本机地址

## 部署gaea中的conn和app时启动报错提示缺少jar包的情况

问题：如题

解决方法：

1. 由于编译时跳过test，pom.xml若含有<scope>test</scope>的注释掉
2. pom未引入对应jar包，引入即可
3. 修改启动的shell脚本，该情况是党项目不需要使用jar包，但是shell代码中却引用到了这个jar包
4. 把jar丢入lib目录下

# 数据库表说明

# 其他说明

## Dal分布式数据

|  |
| --- |
| 目前DAL支持三种结果集处理方式：聚合处理、排序处理、分页处理 |

# 一些快捷键

|  |
| --- |
| Fn+Esc 关闭windows快捷键功能，解决与软件快捷键冲突的问题 |