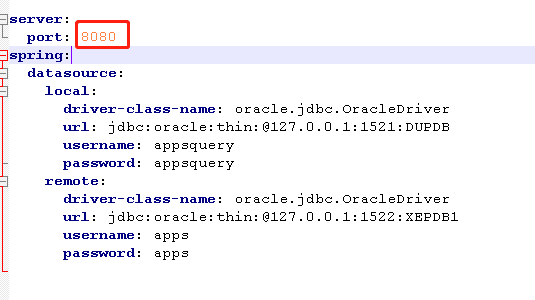
## 工具演示



这两个文件放在同一目录下，然后打开cmd,在jar包所在目录执行下面命令

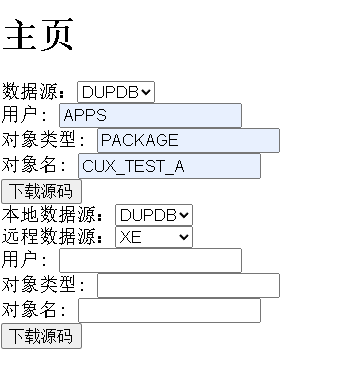
java -jar Migration-0.0.1-SNAPSHOT.jar

如果报错8080端口被占用，可以尝试修改application.yml配置文件里的端口号再次执行



然后浏览器输入http://localhost:8080/,出现下面的页面

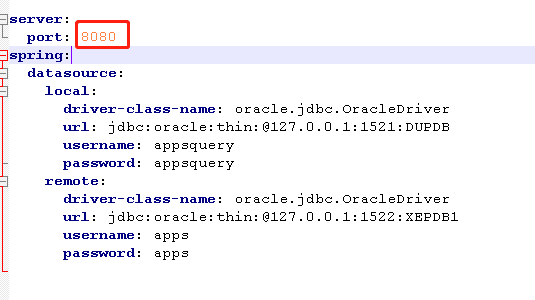
注意Release1.1更新已废弃themeleaf，下面的页面仍然存在，但重构过代码，未进行测试，请不要再次使用



选择好数据源，然后选择好对象，点击第一个按钮，就会下载此对象以及此对象所有依赖，如果需要对比两个环境不同的代码进行迁移，则填好对象后，点击第二个按钮，就会下载不同的对象。

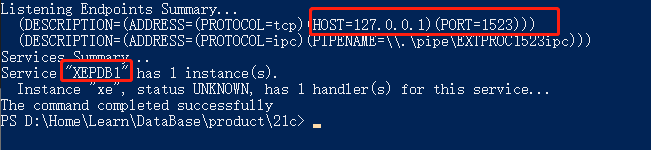
## application.yml说明

继续看这个图



driver-class-name: oracle.jdbc.OracleDriver表示只能读取oracle数据库

url是固定写法，比如我的测试环境监听如下



这样就对应了ip+端口+服务名

Local表示当前测试环境，remote可以表示正式环境或者待迁移代码的环境，目的是从local迁移到remote

## 数据库用户要求

在mapper文件中，涉及到dba\_objects,dba\_dependenciesd等权限查询依赖读写

需要DBMS\_METADATA下载对象，

需要DBMS\_CRYPTO 进行散列，从而判断对象是否发生了改变（可以使用第三方的散列工具，防止不同版本的数据库之间散列函数不同导致结果不同）

因此下面给出从零开始创建用户并授权语句

注意，realese1.1版本已经将授权语句自动化处理了，可以不提前授权



“初始化用户”需要自行修改表空间，如果用户已经存在，可以只执行“授权”

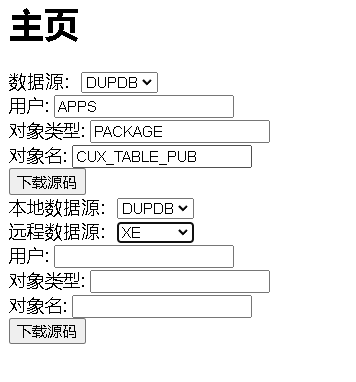
两个环境local和remote用户都要授权，最好不要用管理员用户，以防出现异常或者JAR包被修改导致数据库对象受损，只要是只读用户，数据库会保证对象不被修改，从头到尾执行的存储过程只有DBMS\_METADATA，DBMS\_CRYPTO 和dbms\_monitor

## 初始化对象并测试

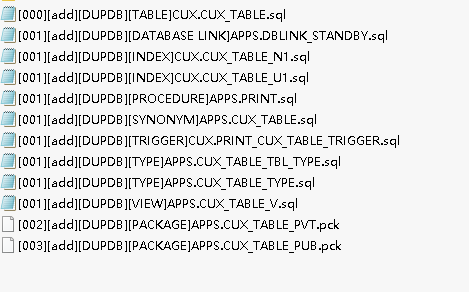
执行下面的脚本进行测试



然后回到页面，执行下载



压缩包内容如下：



按顺序编译

## 端点服务

前端页面采用thymeleaf开发，可以自行实现前端页面，这里也提供了端点服务，可以直接用postman调用

* GET请求

1. 下载所有程序包:

http://localhost:8080/downloadAllDependenciesFile?owner=APPS&objectName=CUX\_TABLE\_PUB&objectType=PACKAGE&dataSource=local

1. 下载依赖图：

<http://localhost:8080/downAllDependenciesGraph?owner=APPS&objectName=CUX_TABLE_PUB&objectType=PACKAGE&dataSource=local>

1. 对比并下载源码

[http://localhost:8080/downloadCompareDependenciesFile?owner1=APPS&objectName1=CUX\_TABLE\_PUB&objectType1=PACKAGE&dataSource1=local&owner2=APPS&objectName2=CUX\_TABLE\_PUB&objectType2=PACKAGE&dataSource2=r](http://localhost:8080/downloadCompareDependenciesFile?owner1=APPS&objectName1=CUX_TABLE_PUB&objectType1=PACKAGE&dataSource1=local&owner2=APPS&objectName2=CUX_TABLE_PUB&objectType2=PACKAGE&dataSource2=local)emote

* POST请求

1. 获取所有对象节点

<http://localhost:8080/getAllDependencies>

{

"owner":"APPS",

"objectName":"CUX\_TEST\_A",

"objectType":"PACKAGE",

"dataSource":"local"

}

1. 下载图

http://localhost:8080/allDependenciesGraph

{

"owner":"APPS",

"objectName":"CUX\_TEST\_A",

"objectType":"PACKAGE",

"dataSource":"local"

}

1. 下载图数据结构

http://localhost:8080/allDependenciesGraphCode

{

"owner":"APPS",

"objectName":"CUX\_TEST\_A",

"objectType":"PACKAGE",

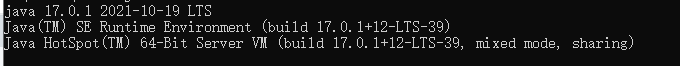
"dataSource":"local"

}

## 其它说明

1.项目JDK版本是JDK17，如果不能正常运行，请使用java --version查看并升级当前版本

下图是我的开发环境版本



2.依赖图基于jgrapht-core实现，效果着实有点丑，可以使用http://localhost:8080/allDependenciesGraphCode返回边和顶点信息，然后使用第三方框架自己实现图

3虽然项目存储依赖信息的对象全是原型Bean,但是未充分测试并发情况下，其对象是否被其它请求篡改，因此最好只在本地部署并多次测试

4为监控本工具对数据库负载影响，已为工具申请的会话设置了客户端标识

可以用下面sql进行查询

select \* from v$session s where s.client\_identifier='mybatis';

当开启资源消耗统计后就可以进行查询

begin

dbms\_monitor.client\_id\_trace\_enable(client\_id => 'mybatis');

end;

select \* from v$client\_stats;*--需要开启client\_id\_trace\_enable*

当开启跟踪后就能为这些会话生成跟踪文件

begin

dbms\_monitor.client\_id\_trace\_enable(client\_id => 'mybatis');

end;

生成跟踪文件语法

trcsess output=/tmp/mybatis\_trace.trc clientid='mybatis' /usr/local/oracle19c/diag/rdbms/prod\_primary/DUPDB/trace*/\*.trc /usr/local/oracle19c/diag/rdbms/prod\_primary/DUPDB/trace替换成自己的trace文件所在路径*

7.报错信息暂时未完善，比如你输入不存在的对象，可能界面没有反应，控制台会输出报错

8.两个数据库的对象必须要求owner（schema）一致，如果测试环境的APPS用户和正式环境的CUX用户，即使源码一致，仍然视为不同的对象

9.输入的字符串必须大写

## 附件



## 8.更新内容

**Release1.0版本更新内容**

1. 使用线程池优化读取DDL速度,广度搜索改成并行执行
2. 修改hash函数为java Hash.getSha256

**Release1.1版本更新内容**

1. 添加redis自动缓存DDL对象
2. DBLINK的DDL有问题，已经删除
3. 由于性能问题，已屏蔽拦截器标识客户端,执行时间优化从18秒到2秒
4. 开发前后端分离版本
5. 引入swagger3（openAPI）依赖，地址http://localhost:8080/swagger-ui/index.html
6. 将下载节点对象的方法抽出，重新封装SourceCodeService
7. tomcat容器切换到undertow
8. controller由单例改成请求作用域
9. Node对象在postman调用序列化失败问题
10. 添加紧凑树（紧凑树显示效果不好，已删除）
11. 添加树选择
12. 修复8操作后，redieTemplate注入失败
13. 添加springboot热部署依赖，修改方法无需重启应用
14. Redis缓存修复了hash key验证导致错误判断sourceCode存在，需要强制判断sourceCode是否存在
15. 重构所有的线程池和thread方法为steam实现
16. 新增BFS工厂方法
17. 新增服务端状态检查，授权语句集成到页面中提示