# 项目结构简介



本项目下有三个文件夹，分别是gbdpcloud-provider、gbdpcloud-provider-api、jars。其中：

1. gbdpcloud-provider文件夹下的项目，包含了主要的业务实现逻辑，包括controller，service，mapper，xml等文件；
2. gbdpcloud-provider-api文件夹的项目，与gbdpcloud-provider的项目一一对应，主要是提供fegin调用接口，用于gbdpcloud-provider下项目之间的调用（fegin替代http、RestTemplate等方法），包括entity，service等；
3. jar文件夹包含了项目要手动引入的jar包。

逻辑关系上，gbdpcloud-provider-api是微服务内部互相调用所使用的模块；除需调用项目需要依赖gbdpcloud-provider-api项目外，同一个微服务的gbdpcloud-provider项目也需要依赖自己的gbdpcloud-provider-api。

# 项目配置运行

## 依赖环境

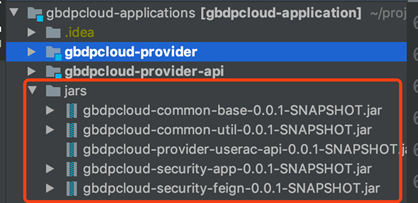
|  |  |  |
| --- | --- | --- |
| 环境 | 推荐使用版本号 | 备注 |
| Jdk | 1.8 |  |
| Maven | last |  |
| Mysql | 5.7 |  |
| Redis | 3.2+ |  |
| Zookeeper | 3.4.14 |  |
| Kafka | 2.1.1+ |  |

## 配置运行

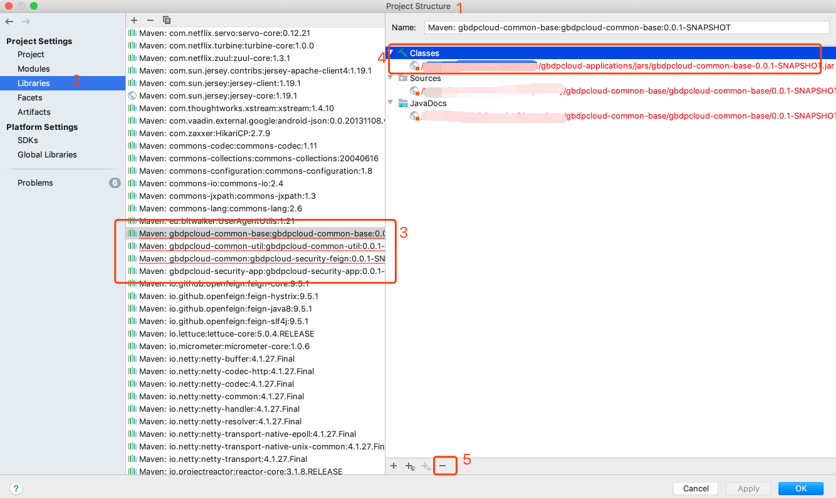
## 导入jar包

项目运行需依赖的jar文件夹下的jar文件，因此需要导入项目之中。

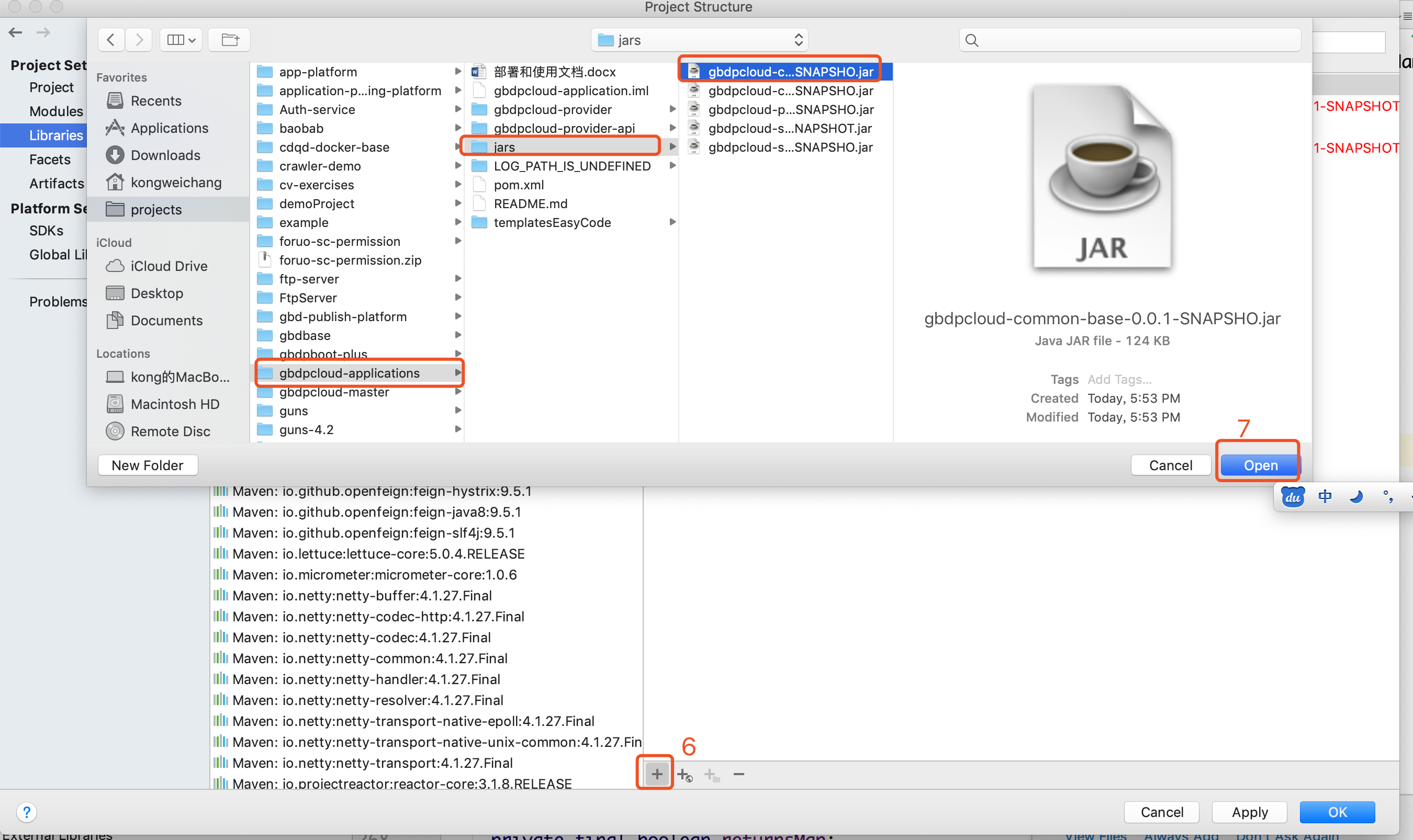
手动配置方法：



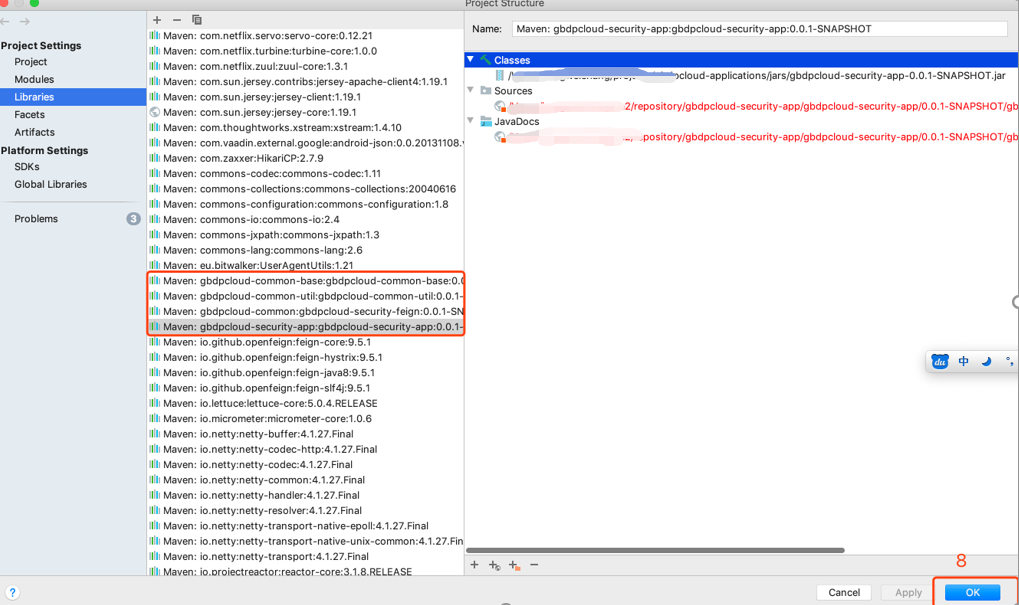
按如图所示步骤将项目中本地的jar包为导入。



1. 打开project structure，选择libraries，点击需要导入的包（下面会有红线标记），选中右边Classes选项，点击下面减号删除。



1. 点击+，把本地对应名称的jar进行关联



1. 配置完成，项目不报错后，可以运行和启动项目。

# 项目代码编写和接口访问

## 代码编写

### 自动代码生成

具体请查阅第五章代码生成部分。

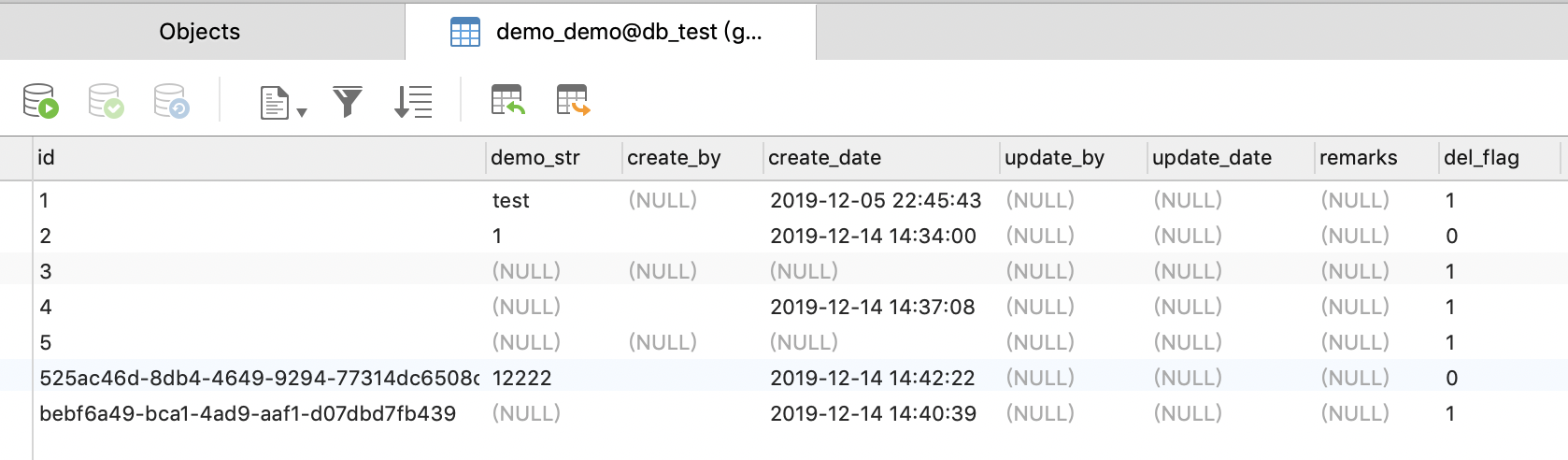
### 业务表代码解析

1. **新业务项目新建**

新建业务需要新建项目时，可以复制gbdpcloud-provider和gbdpcloud-provider-api文件夹下的demo项目。

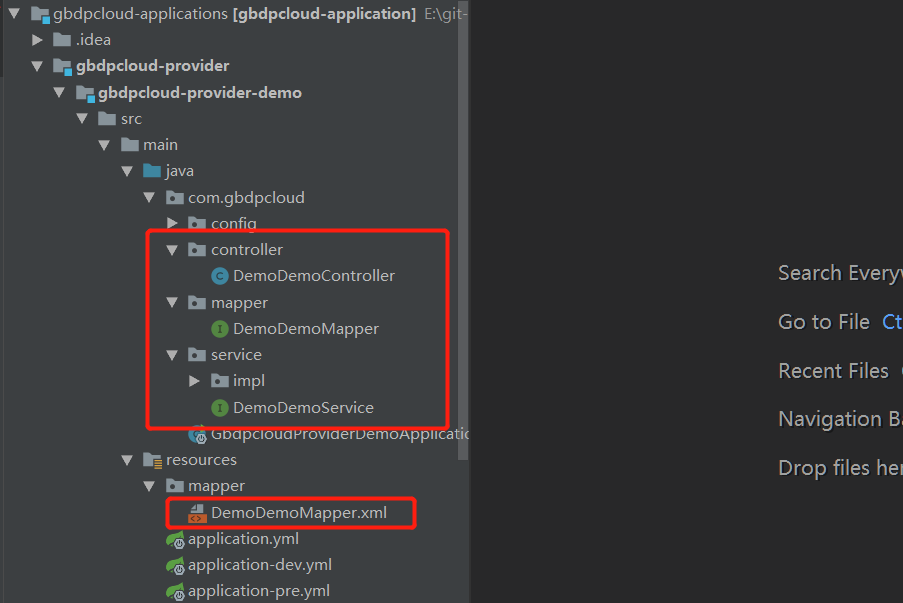
1. **单表业务代码编写**

一张业务表，gbdpcloud-provider下的项目和gbdpcloud-provider-api下的项目都需要编写内容。

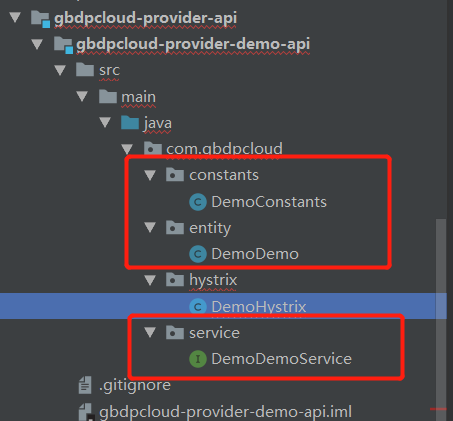
以DemoDemo表为例：

gbdpcloud-provider下的业务代码编写：

如下图所示，选择代码生成或手动编写，其中系统已经内置了：基本的增删改查等方法；系统内置方法的使用，请查看4.1基础接口规范说明。



gbdpcloud-provider-api下的业务项目，需要编写：



## 接口访问

本项目接口不能直接访问，访问的顺序为：访问远程登录接口，获取token-》进行本地服务的访问，将token作为参数。如下所述为具体步骤。

### 登录获取token

1. 获取验证码

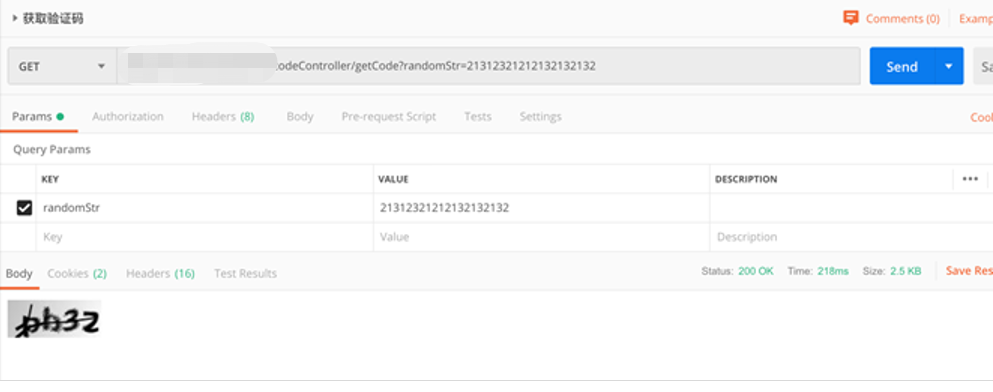
url:

* http://{远程ip}:{接口}/codeController/getCode?randomStr=123456

访问参数：

* randomStr：随机数（可随便填，但需要与登录接口保持一致）

说明：在获取验证码时需要传入，登录接口要传入同一个随机数，为了验证验证码的有效性。



1. 登陆并获取token

url:

* http://{远程连接}：{远程接口}/userac/login?username=test&password=test&imageCode=pb32&randomStr=123456

Header请求参数：

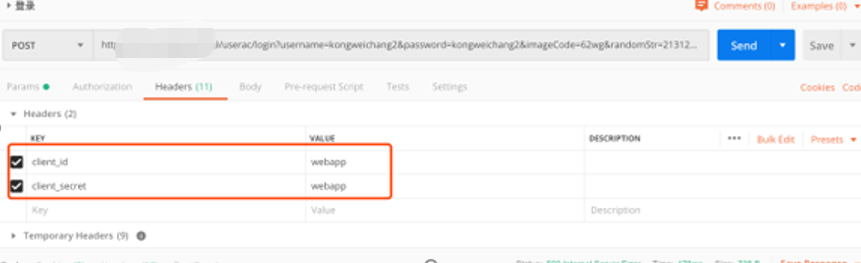
* client\_id : webapp
* client\_secret : webapp

普通参数：

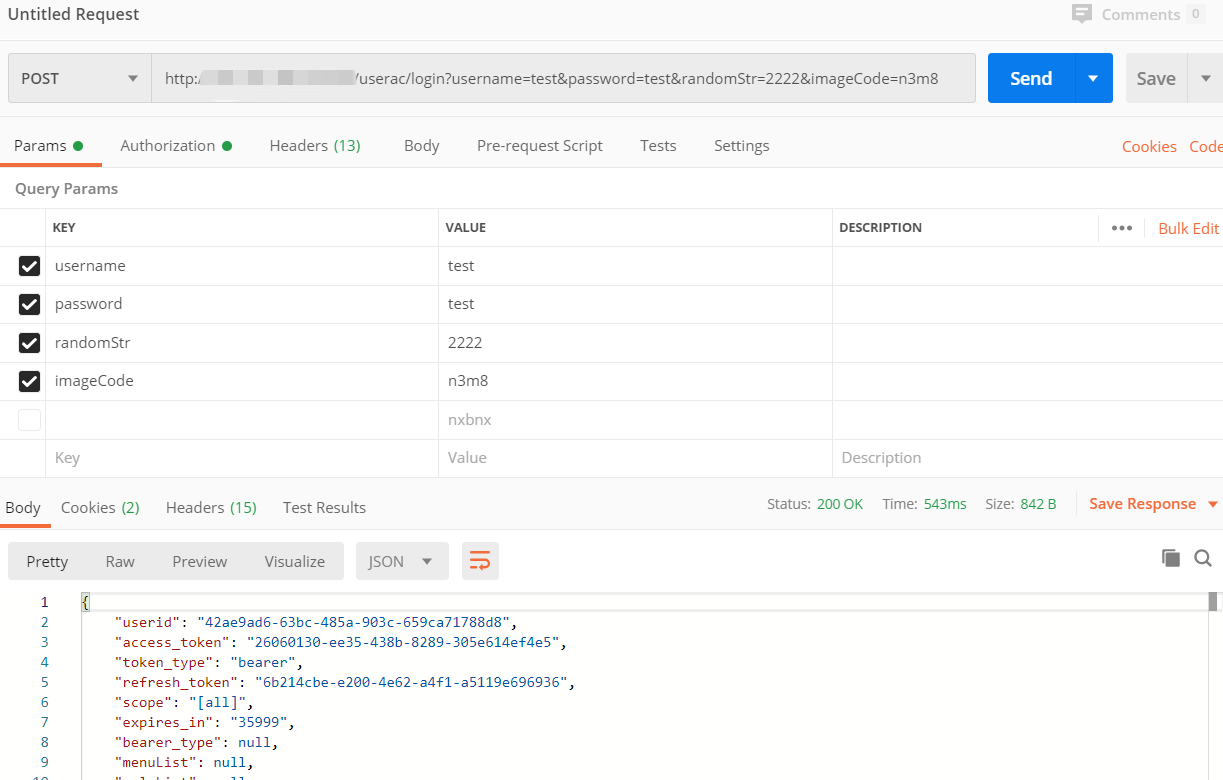
* randomStr：获取验证码时传的随机数
* imageCode：验证码
* username：用户名（默认：test）
* password：密码（默认：test）

请求示例如下图所示：

* Header参数设置：



* 普通参数设置：

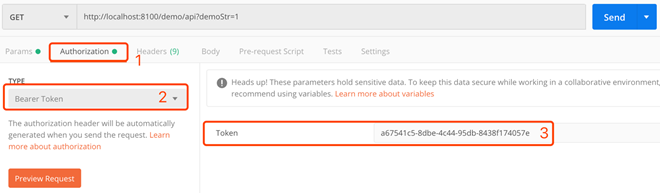


### 业务微服务接口

本地访问接口的地址一般为：

* <http://localhost>:{业务端口}/{访问链接}

访问本地服务接口，需要携带token，也就是在请求中加入token参数，Postman加入方法如下。然后再加入普通参数进行接口的正常访问。



# 编写代码规范

## 基础接口规范

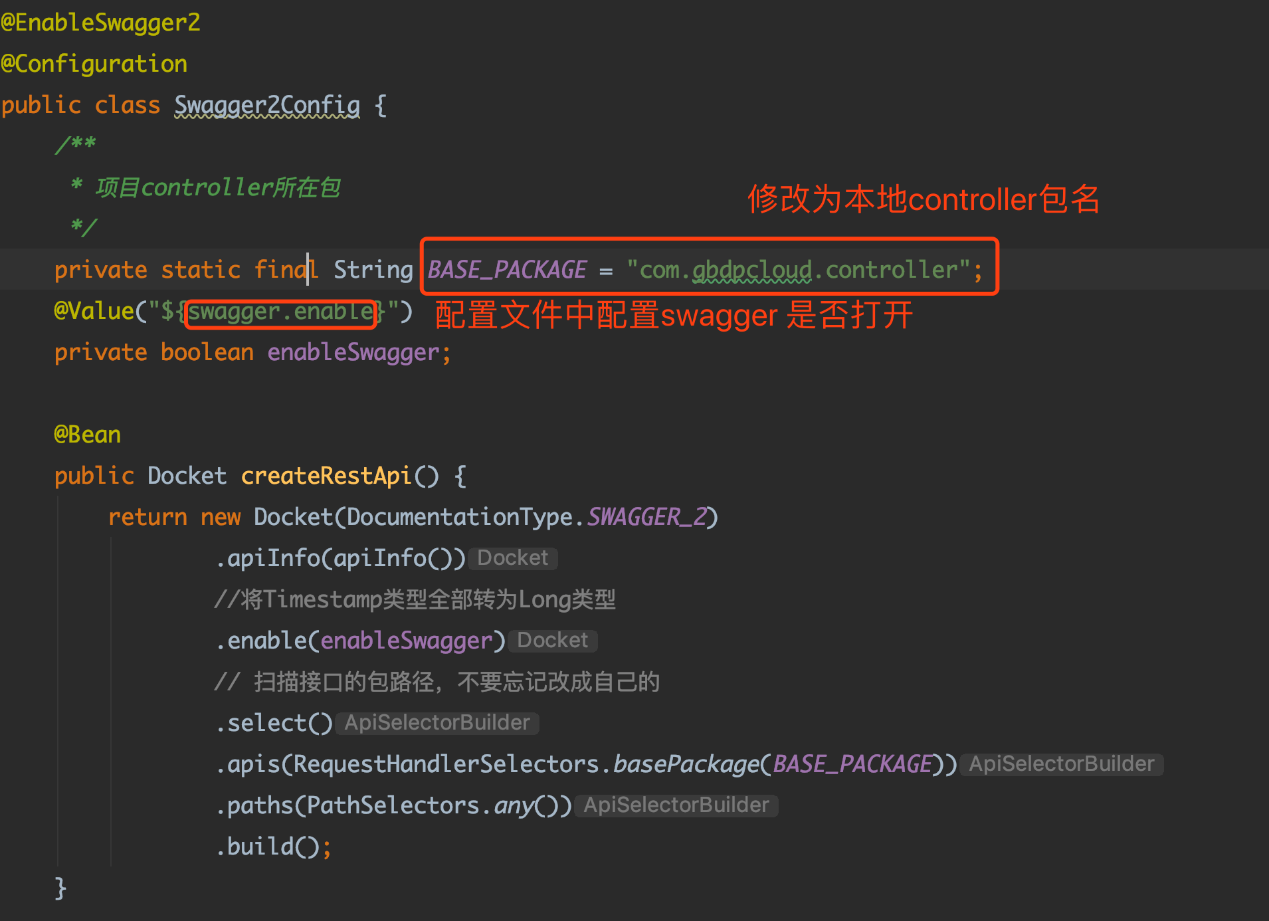
基础接口的方法已将包含：

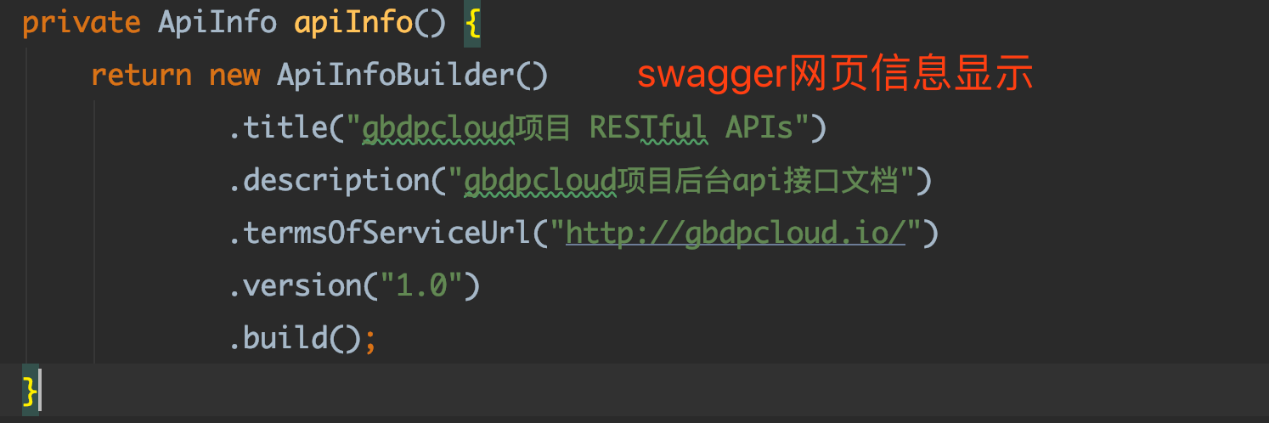
* save(T model); 保存数据
* saveBatch(List<T> models); 批量保存数据
* update(T model); 更新数据
* updateBatch(List<T> models); 批量更新数据
* saveOrUpdate(T model); 根据数据是否携带主键判断save或者update
* count(T query); 根据对象条件，查询总记录条数
* delete(String id); 根据id删除一条数据；
* deleteByIds (String ids); 根据id，一次删除多条数据；
* getById(String id); 根据id，获取数据详情；
* getOne(T query); 根据对象条件获取单条
* page(PageRequest page, T query); 数据分页获取；
* list(T query); 根据条件查询对象列表

## Swagger使用方法和规范

**Swagger配置类：**

SwaggerConfig.Class





**用法说明：**

**参考：**<https://www.cnblogs.com/zhuhui-site/p/10092322.html>

接口类说明注解：

@Api(value = "DemoController", description = "Demo接口")

接口方法说明注解：

@ApiOperation(value = "xx操作接口", notes = "xxModel保存接口")

其他注解参考博客。

## 字段验证使用方法和规范

在接口参数前添加@Validated进行参数验证

下面有几种常用情况：

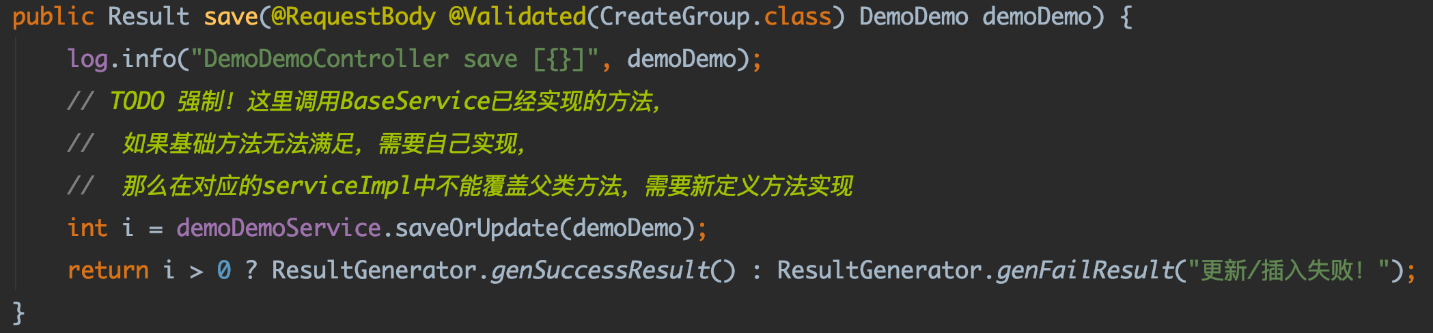
1. entity定义：

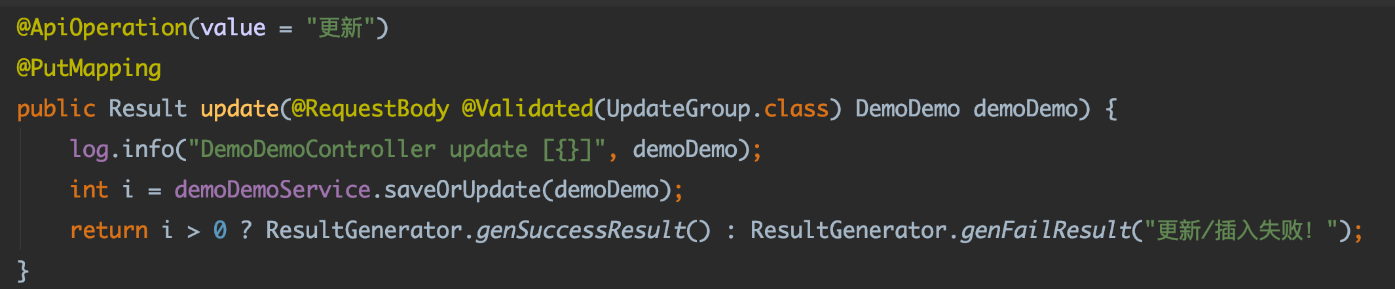
在entity需要校验的字段上添加对应注解，groups指定什么情况下校验，message为校验失败的返回信息。

@NotBlank( message = "Id不能为空", groups = {UpdateGroup.class})  
protected String id;

1. 新建/更新接口参数校验：

CreateGroup-新建操作 UpdateGroup-更新操作

新建：

更新：

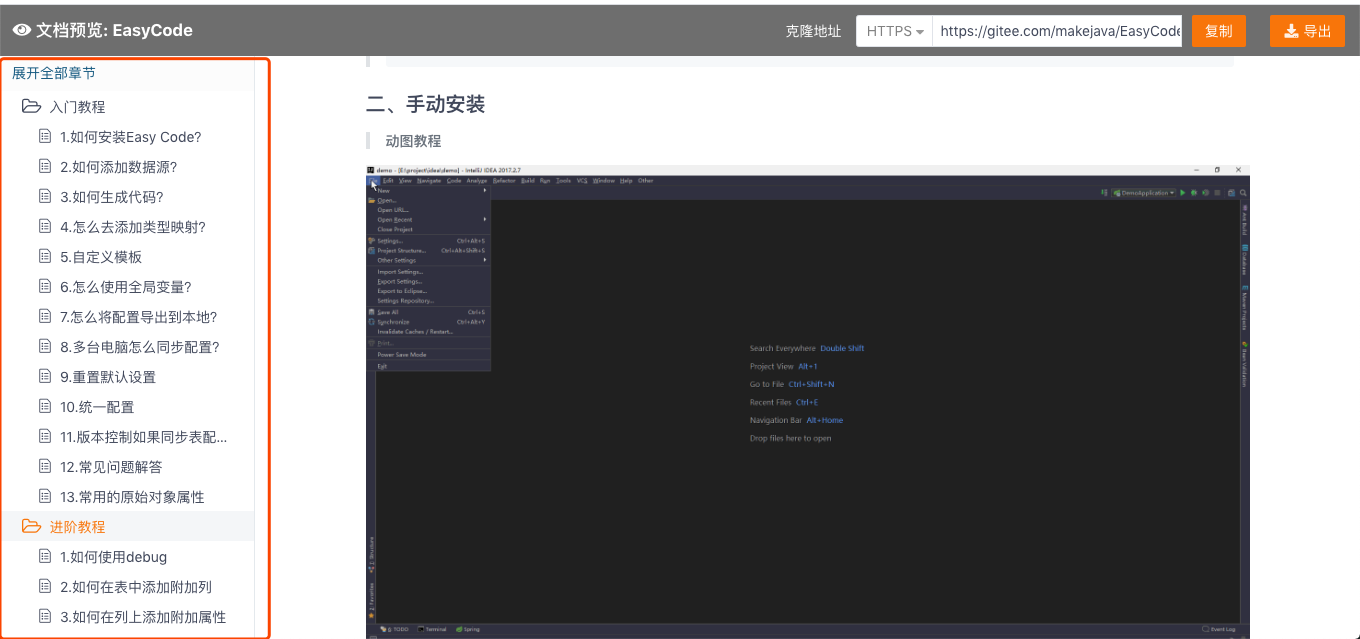
1. 仅传Id/其他的接口参数校验：

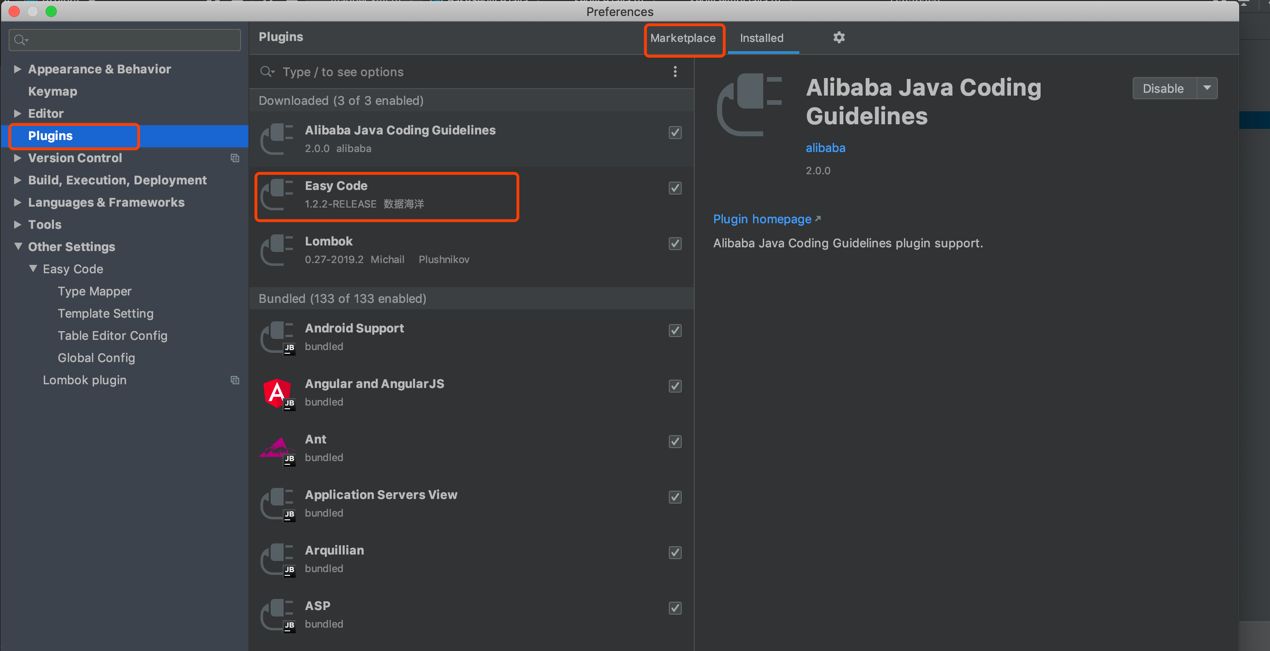
其他操作参考：<https://www.jianshu.com/p/89a675b7c900>

# 代码生成器使用步骤

**本项目使用EasyCode代码生成插件，其官方地址为：**

**源码：<https://gitee.com/makejava/EasyCode>**

**文档：<https://gitee.com/makejava/EasyCode/wikis/pages>**

1. easyCode插件安装：

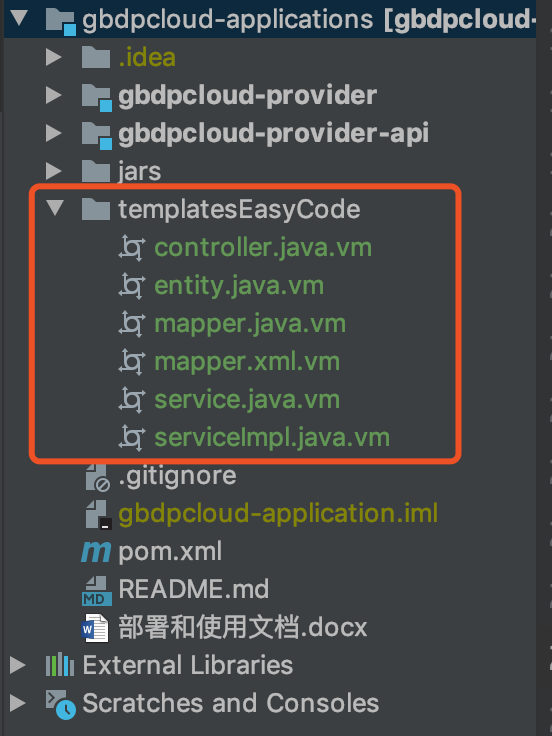
idea插件安装如图所示。

如果由于网络原因打不开插件查询页面，那么可以选择手动安装：

<https://gitee.com/makejava/EasyCode/releases>

下载安装包后在idea中导入插件。

1. easyCode自定义模板导入

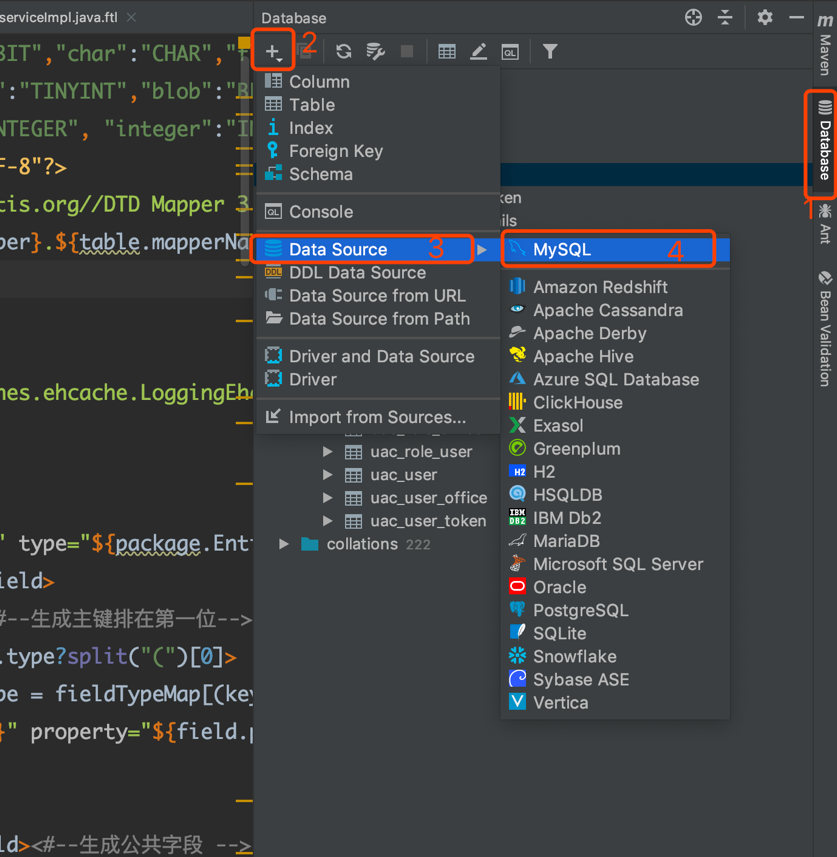
gbdpcloud项目自定义模板已写好，在如图所示路径下：

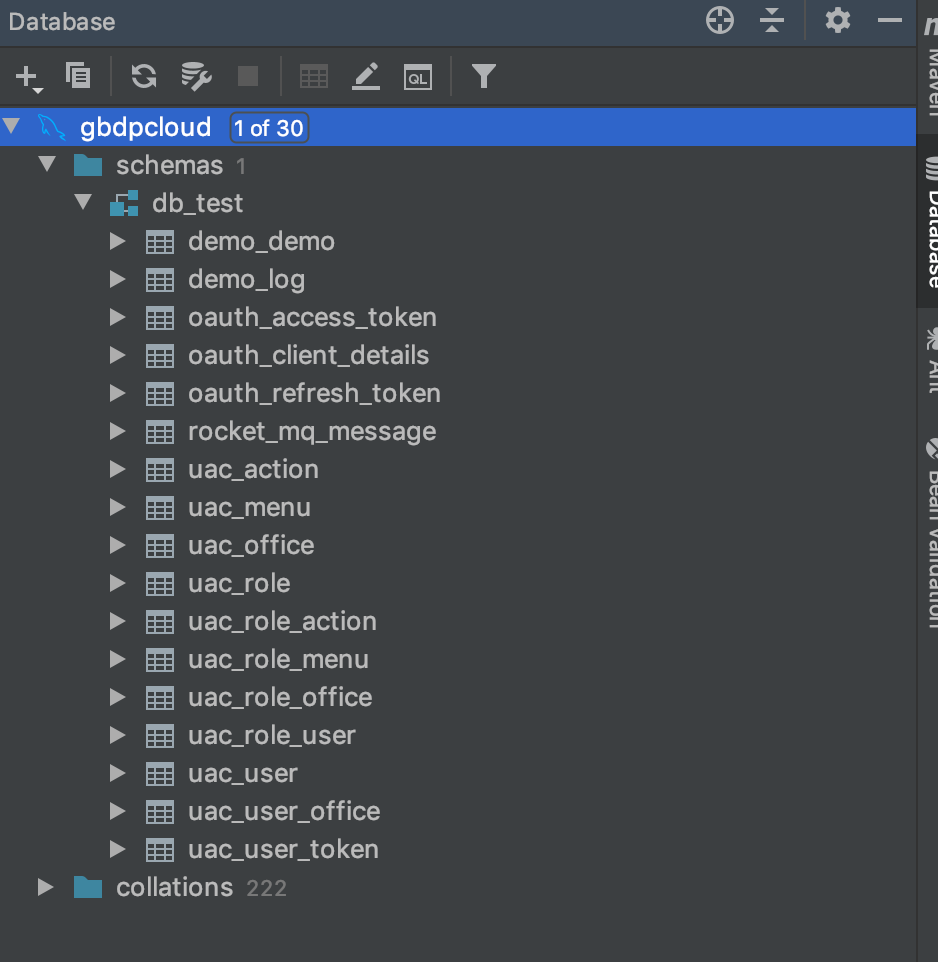
打开idea导入模板，步骤如下图所示：

导入后点击ok保存。



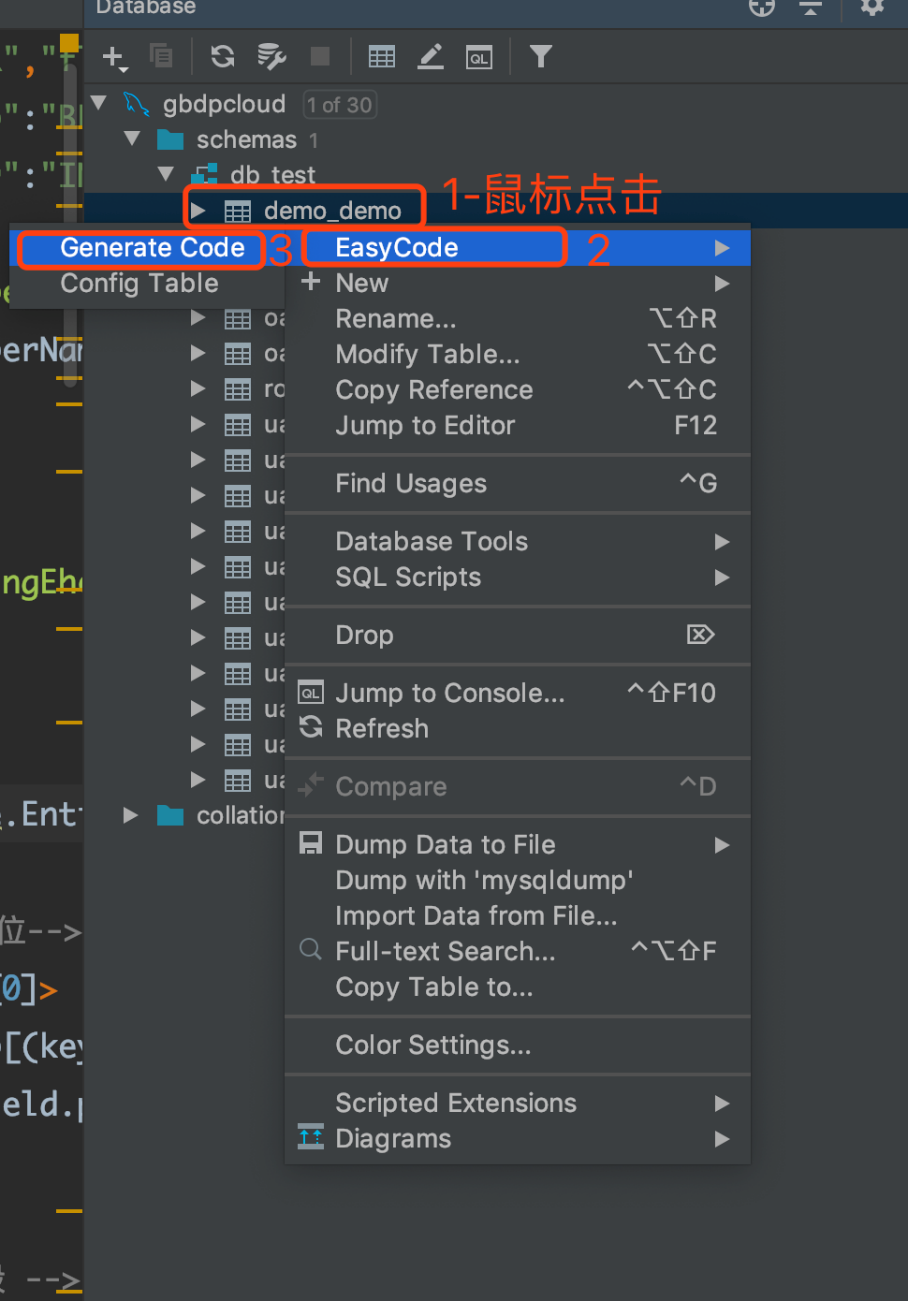
1. easyCode数据库连接与代码生成

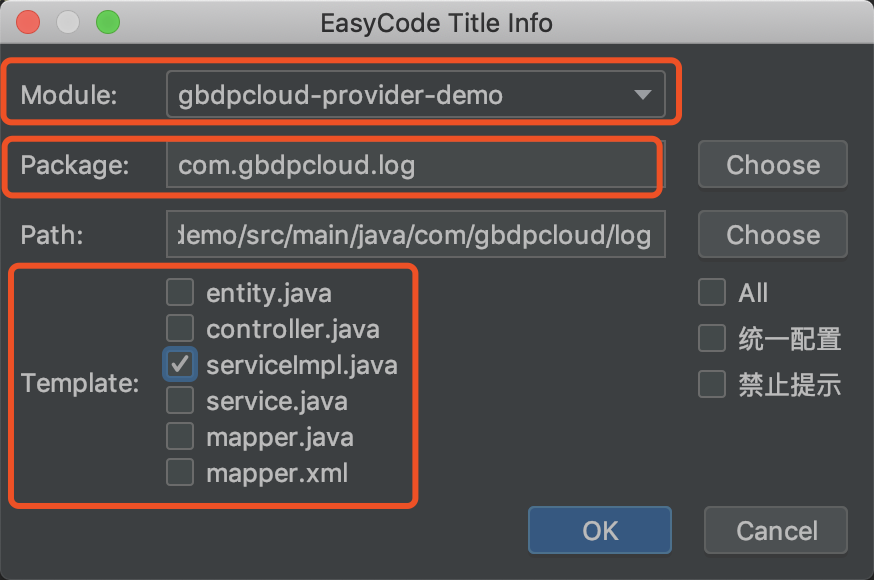
连接数据库：

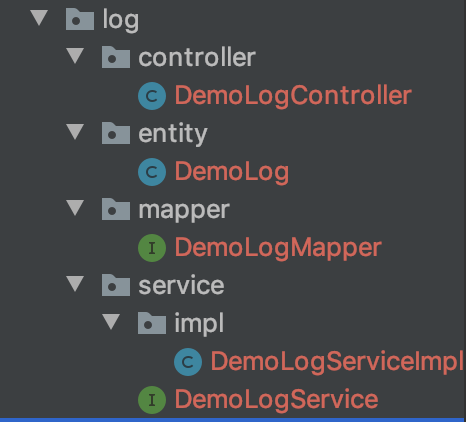
导入后可以看到表信息：



代码生成流程：



选择生成的路径以及类点击OK：



# bug修复

1. deleteManyByLogic方法无效

解决：

a.更新最新版本的jar包（common模块jar包）

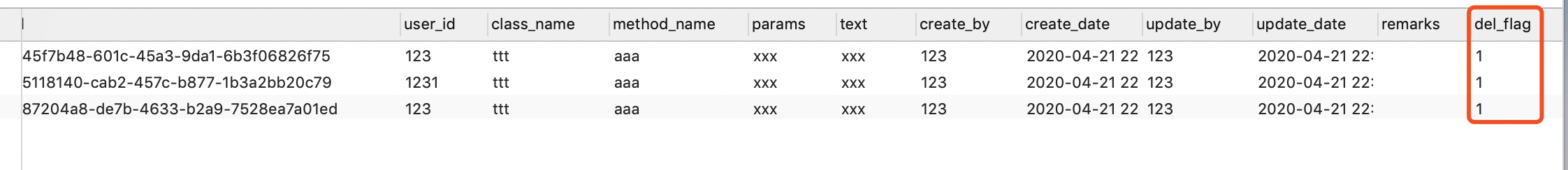
b.修改现在mapper.xml(

<**update id="deleteManyByLogic"**>  
 UPDATE table\_name SET  
 del\_flag = '1'  
 WHERE id IN (#{ids})  
</**update**>

修改为：

<**update id="deleteManyByLogic"**>  
 UPDATE table\_name SET  
 del\_flag = '1'  
 WHERE id IN  
 <**foreach item="ids" collection="array" open="(" separator="," close=")"**>  
 #{ids}  
 </**foreach**>  
</**update**>

**可以选择直接全部替换**)

c.测试接口可用性