# 1、基础

在开始改动前需要了解Apollo工程的整体结构，以及调试启动的方法。建议本地先搭建个MySQL库，然后把Apollo跑起来。  
  
从MySQL到Oracle，一个比较大的问题是Apollo中很多表名、字段名和Oracle的关键字有冲突，导致无法在Oracle建表。网上针对这个问题提供了使用双引号来解决。Oracle的SQL语句在解释时会把所有的字符都转为大写（字符串常量除外），所以我们一般认为Oracle的SQL语法不区分大小写。比如说我可以用如下SQL建表：  
  
create TaBle aPplicAtiOn(...);最终得到一张名为APPLICATION的表。在一些情况下，我们会需要区分大小写的表名，这时候我们可以在建表时用双引号将表名括起来：  
  
create TaBle "aPplicAtiOn"(...);这样就真正得到了名为aPplicAtiOn的表了。同理，字段名、同名等Oracle对象也都可以用双引号来区分大小写。  
  
由于Apollo使用了JPA，因此为了便于后续维护，针对PO层进行统一的调整，而不是将表名进行双引号括起来的方式。

# 2、Oracle适配改动

下面的改动大部分是参考已有的vanpersl/apollo基于0.8.0的Oracle版本的代码。改动链接：https://github.com/ctripcorp/apollo/compare/v0.8.0...vanpersl:db-oracle  
  
本次改动，sql将依赖jpa重新生成，不沿用现有mysql脚本，oracle sql需要以实体类为准。

## 2.1、Apollo工程加入ojdbc的依赖

Apollo默认使用的是oracle数据库，在最新的版本1.7.0还提供了针对flywaydb的支持。  
  
我们通过工具搜索mysql-connector-java关键字，同时将maven依赖里面的mysql依赖全部换成oracle

<!-- oracle -->  
 <dependency>  
 <groupId>com.oracle</groupId>  
 <artifactId>ojdbc6</artifactId>  
 <version>11.2.0.4.0</version>  
 </dependency>

主要修改如下两个文件：  
apollo/apollo-common/pom.xml  
apollo/pom.xml  
  
将里面的mysql-connector-java依赖替换成oracle依赖。

## 2.2、修改数据库连接配置

新建好Oracle的库后，将Apollo配置中原本MySQL的连接配置改为Oracle的连接配置  
注意，根据我们的配置，需要修改以下三处配置。  
  
apollo/apollo-adminservice/src/main/resources/application.yml  
apollo/apollo-configservice/src/main/resources/application.yml  
apollo/apollo-portal/src/main/resources/application.yml

spring:  
 datasource:  
 password: '{password}'  
 url: jdbc:oracle:thin:@{ip}:{port}:{db}  
 username: '{ApolloConfig|ApolloPortal}'

## 2.3、添加Oracle配置。

### 2.3.1、修改application.properties

这些配置改动在apollo/apollo-common/src/main/resources/的application.properties配置文件。  
  
1、修改spring.datasource.driver-class-name=oracle.jdbc.driver.OracleDriver指定使用Oracle驱动。  
  
2、将spring.datasource.hikari.connectionInitSql=set names utf8mb4删除  
  
3、增加spring.jpa.database-platform=org.hibernate.dialect.OracleDialect  
4、关闭Hibernate的ORM生成SQL语句自动给表名列名都加上双引号  
spring.jpa.hibernate.globally\_quoted\_identifiers=false  
spring.jpa.properties.hibernate.globally\_quoted\_identifiers=false

### 2.3.2、修改连接池配置

这些配置改动在apollo/apollo-common/src/main/resources/的datasource.xml配置文件。  
  
将

validationQuery="SELECT 1"  
initSql="set names utf8mb4"

换成

validationQuery="SELECT 1 FROM DUAL"

## 2.4、Sequence

Apollo的表使用的自增ID作为主键。Oracle没有直接支持自增ID的功能，必须通过Sequence实现。在建表的SQL里已为两个库分别建了名为APOLLO\_ID\_SEQ的Sequence。代码中所有的Entity类的id属性都要加上GeneratedValue和SequenceGenerator的配置。Apollo的代码中大部分的Entity类是继承自一个BaseEntity基类的，所以要修改的地方并不多。  
这几个类需要修改：BaseEntity、ReleaseMessage、InstanceConfig、Instance、UserPO、ConsumerAudit  
如下添加GeneratedValue和SequenceGenerator两行：

public abstract class BaseEntity {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "sequence")  
 @SequenceGenerator(name = "sequence", sequenceName = "APOLLO\_ID\_SEQ", allocationSize = 1)  
 @Column(name = "ID")  
 private long id;  
 ......  
}

## 2.5、修改columnDefinition

将所有的columnDefinition = "Bit default '0'"替换为columnDefinition = "Number(1) default 0 "  
  
 这几个类需要修改：AccessKey、AppNamespace、BaseEntity、Release  
   
同时设置默认值为false  
  
private boolean enabled = false;

## 2.6、修改Oracle关键字的问题

1、BaseEntity里面的isDeleted更改为DELETED\_FLAG  
  
2、@SQLDelete和@Where里面的isDeleted更改为DELETED\_FLAG，同时将小写id换成大写ID  
  
@Where(clause = "isDeleted = 0")  
换成  
@Where(clause = "DELETED\_FLAG = 0")  
  
set isDeleted = 1 where id =   
换成  
set DELETED\_FLAG = 1 where ID =   
  
3、修改AuthConfiguration.java里面的原生sql  
  
4、实体类里面修改表名称，字段名称以及索引等信息

## 2.7、表明冲突问题

项目中很多表明已经存在  
  
针对portal模块下的表，所有表名称增加"PORTAL\_"前缀，便于区分  
  
针对config模块下的表，所有表名称增加"CONFIG\_"前缀，便于区分

## 2.8、portal模块下App和APP\_NAMESPACE单独处理，不使用com.ctrip.framework.apollo.common.entity下的实体

源码使用mysql，使用了两个数据库，但是换成oracle后，只能使用一个数据库  
  
1、在com.ctrip.framework.apollo.portal.entity.po增加PortalApp和PortalAppNamespace，字段和com.ctrip.framework.apollo.common.entity下的App和AppNamespace一样。  
  
2、修改dao层  
 修改com.ctrip.framework.apollo.portal.repository下的AppRepository和AppNamespaceRepository，将App和AppNamespace换成PortalApp和PortalAppNamespace  
  
3、修改service层  
 修改com.ctrip.framework.apollo.portal.service下的AppService和AppNamespaceService，由于dao返回的是PortalApp和PortalAppNamespace按照转换修改即可，可以使用BeanUtils进行转换

## 2.9、检查所有表字段的长度和是否为空

## 2.10、为所有表字段增加字段描述