## **GODDESS打包及运行**

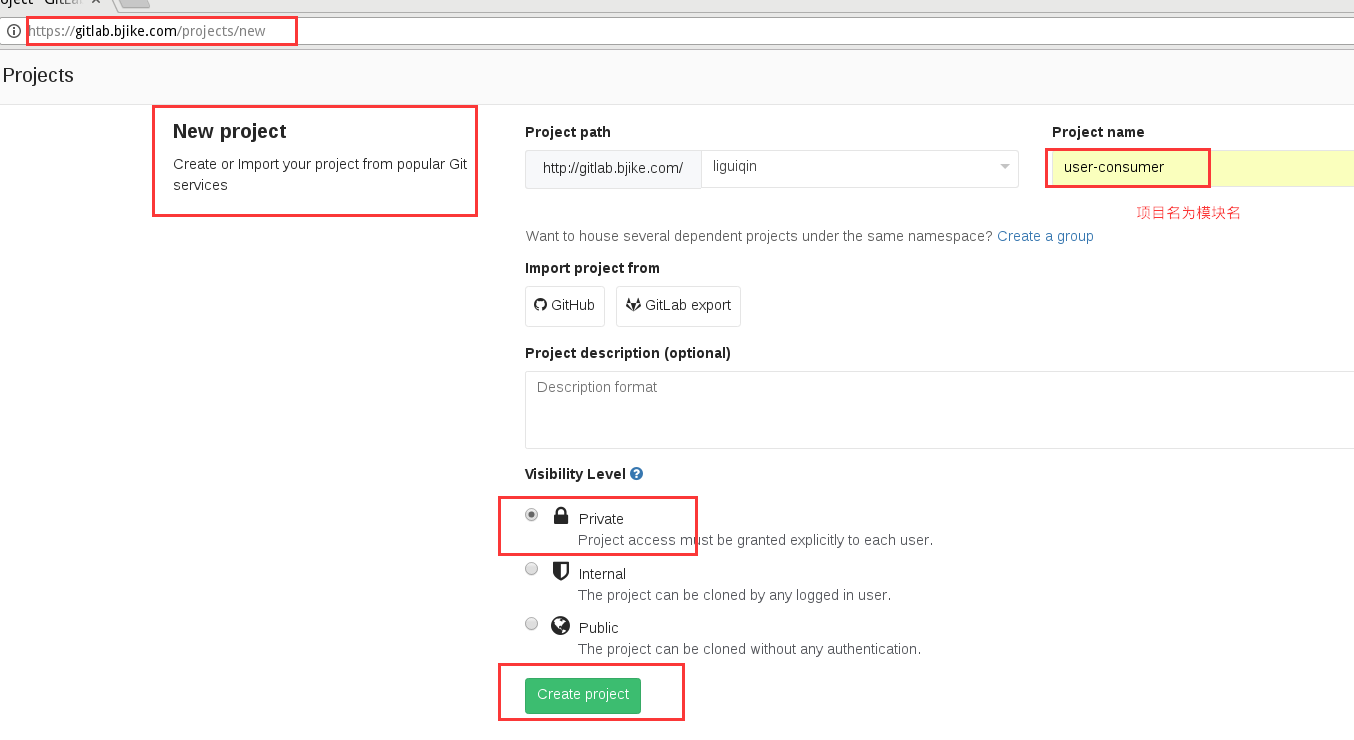
仓库地址:https://gitlab.bjike.com

1. **如何打包jar发布到仓库**

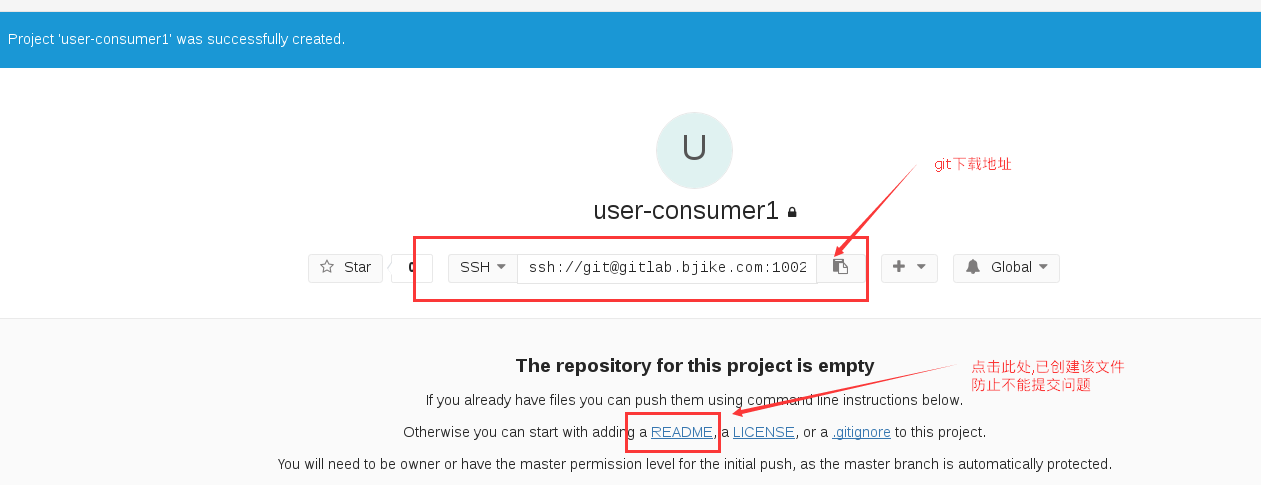
注:我们的项目jar包仓库是放在gitlab地址上,(参见绿字仓库地址没有账号问负责人申请),一但jar包提交到仓库上,然后按照Portainer(参照Portainer的使用)的流程,就可以发布并且部署在服务器集群上.

下面以打包User模块的provider及consumer为例:

1.1在gitlab上创建user-provider,user-consumer项目



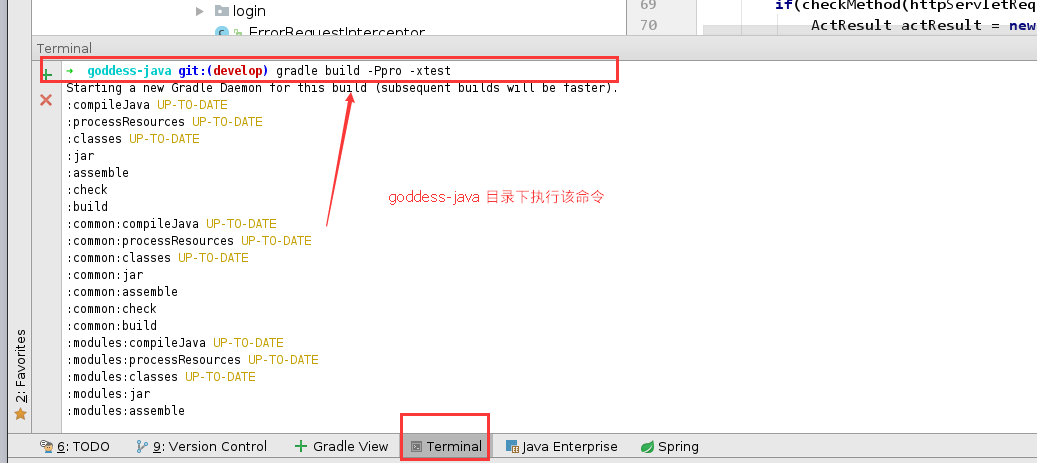
项目创建后见下图操作:



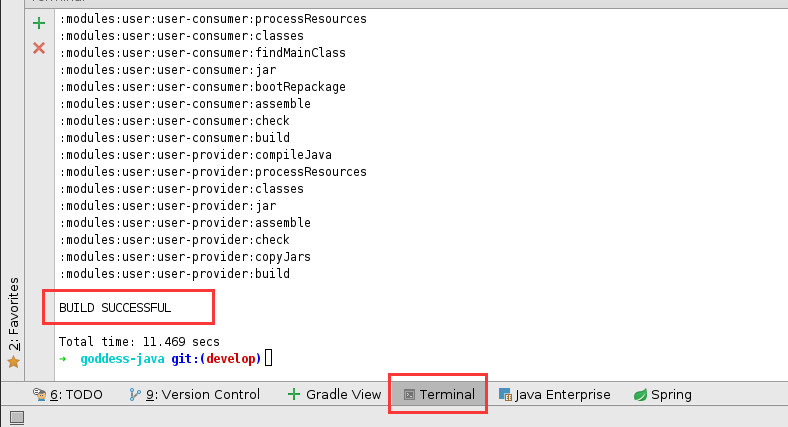
1.2克隆项目到本地:



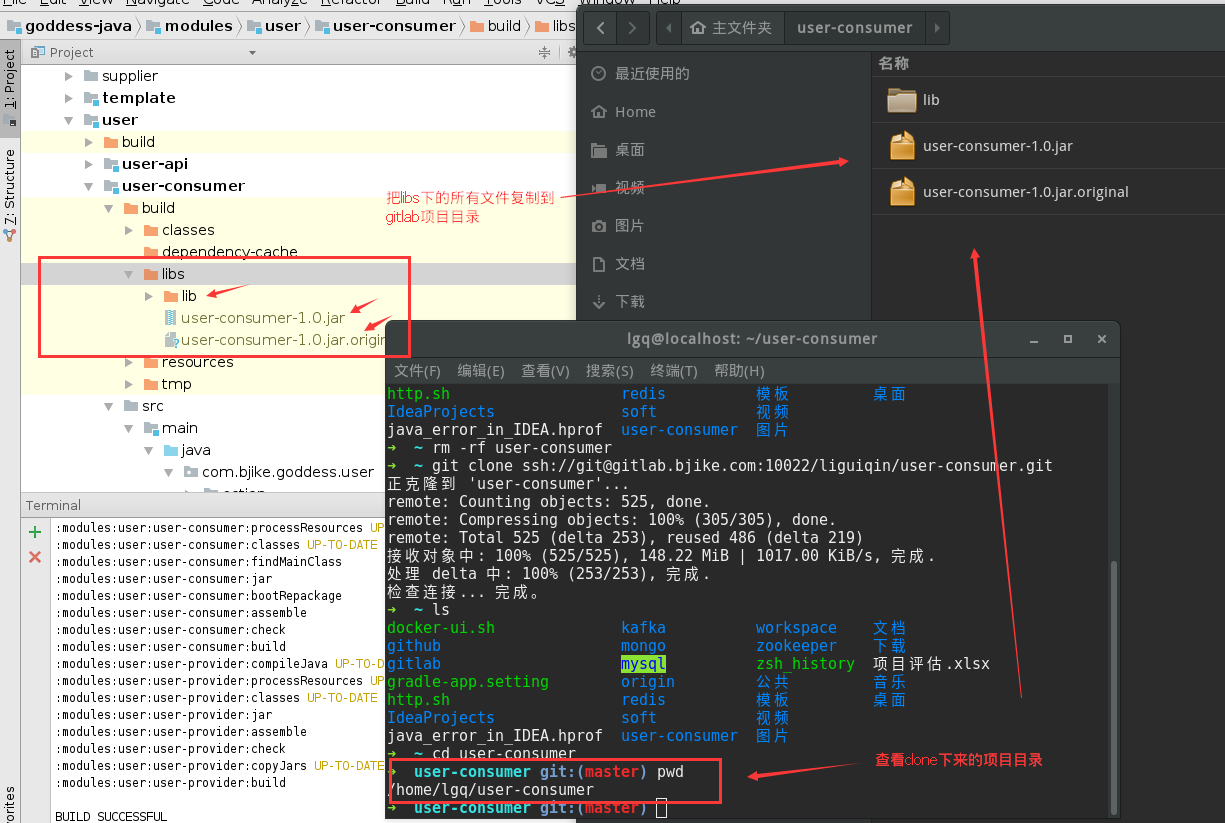
1.3打包!进到goddess目录执行以下命令 :gradle build -Ppro -xtest,执行两次



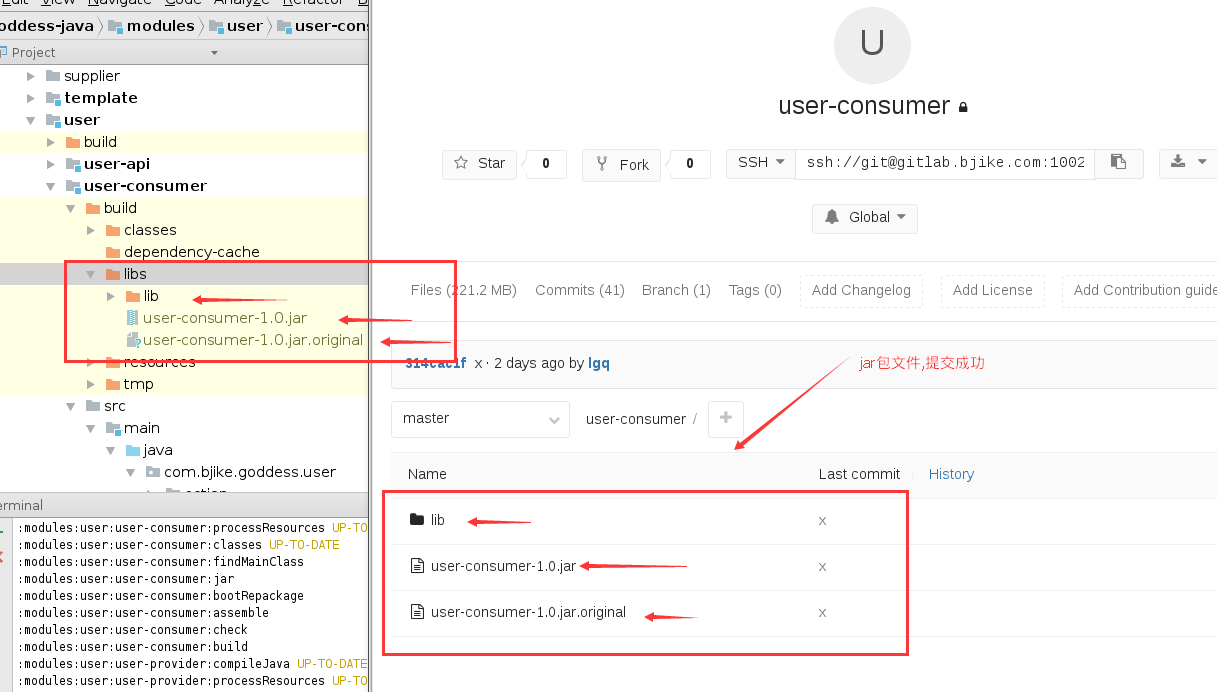
看到下图,打包成功



把jar包复制到gitlab项目目录下,以便提交



最后进入user-consumer用git命令进行提交即可,见下图,操作步骤完成:

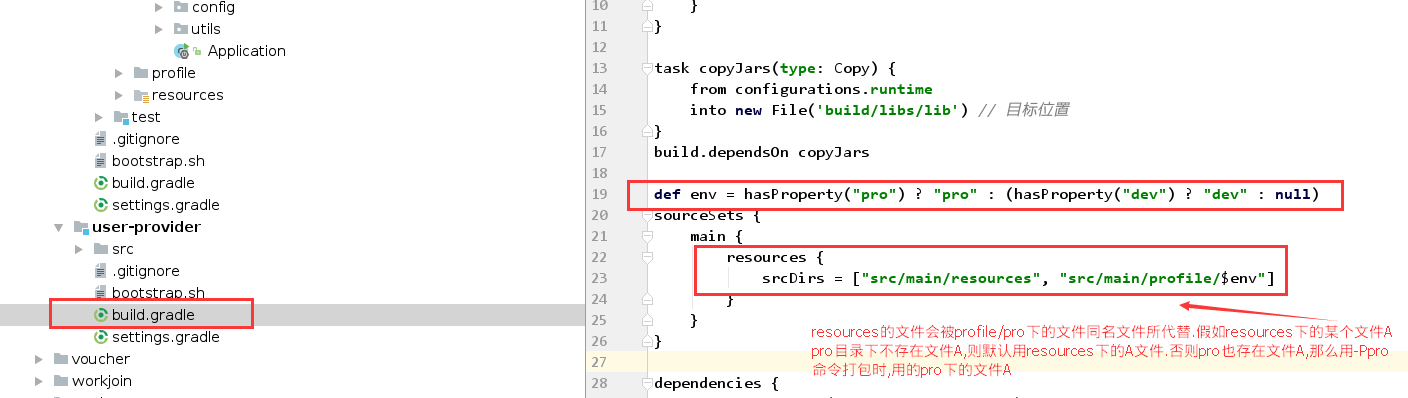


最后一步很重要:以上操作完后一定要执行gradle clean清除掉jar包,原因请看关于gradle build -Ppro -xtest 打包命令

1. **关于gradle build -Ppro -xtest 打包命令**

-xtest作用:过滤掉测试包

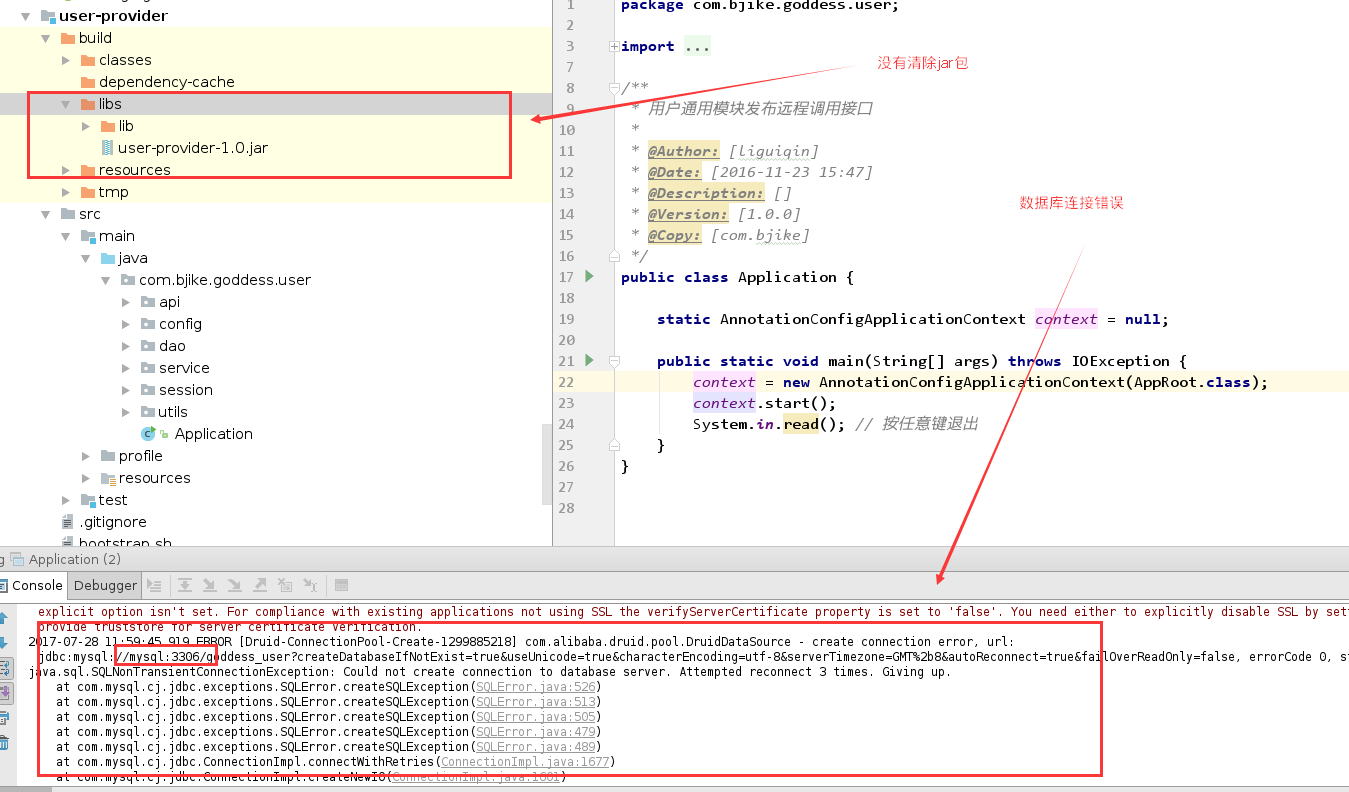
-Ppro作用: 打包替换配置文件,见下图



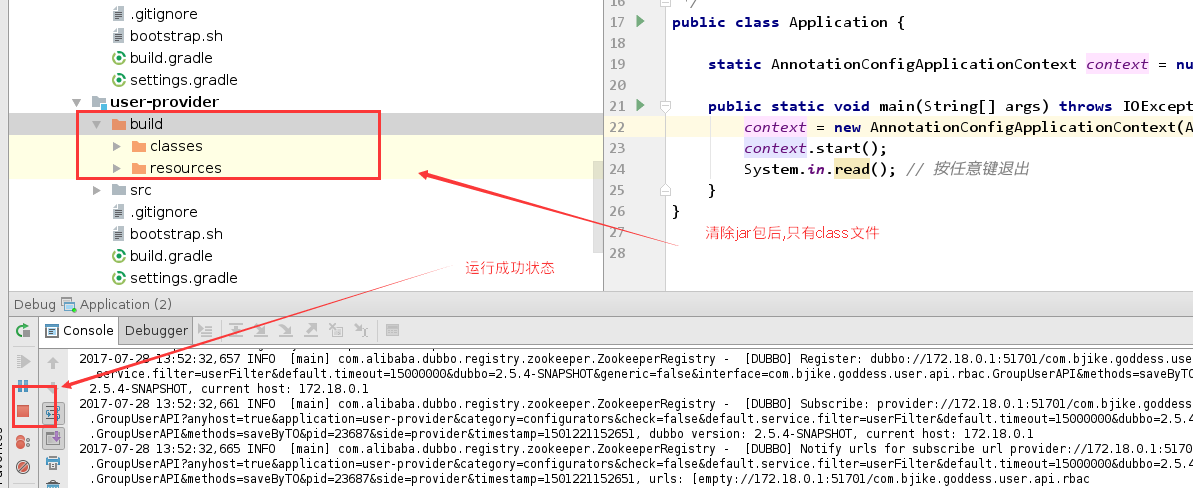
注:gradle build -Ppro -xtest打包完成一定要执行gradle clean清除掉jar包,

因为应用运行时是寻找项目build下的jar包(没有jar包时执行编译的class文件文件)

运行的,而我们打的jar包到服务器,pro的mysql,zookeeper等连接地址都是服务器真实环境的地址,一但IDE(idea)项目运行起来,优先执行jar包而不是编译的class文件,那么就会导致数据库连接不上等错误,见下图



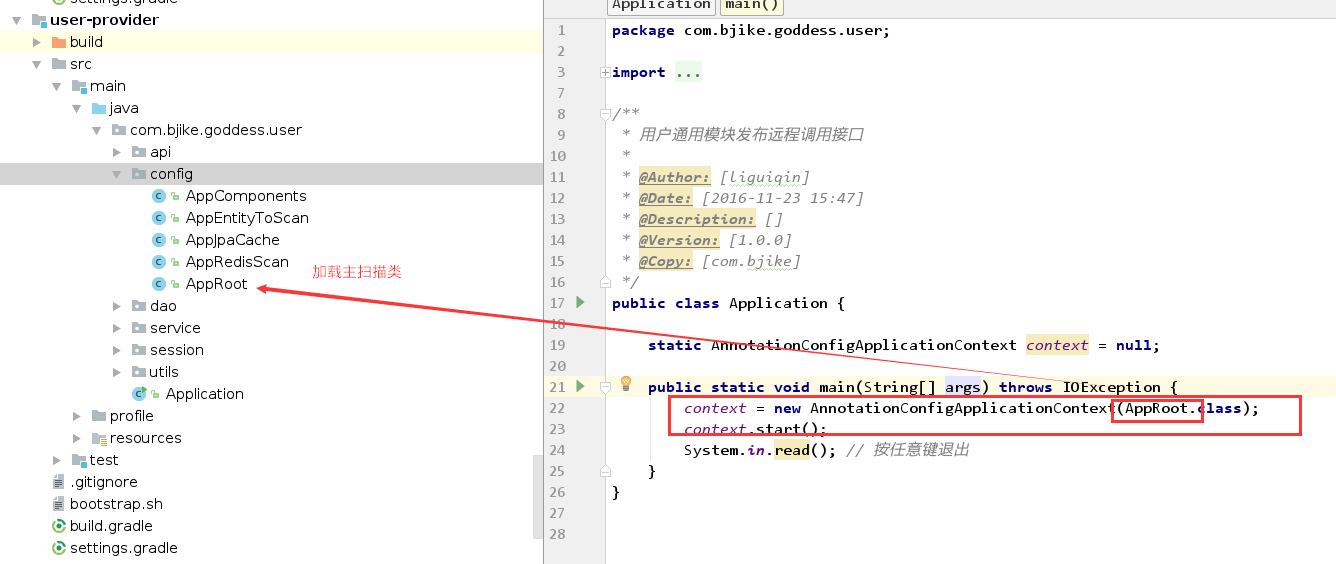
成功状态见图:



1. **项目运行Bean的注入及扫描**

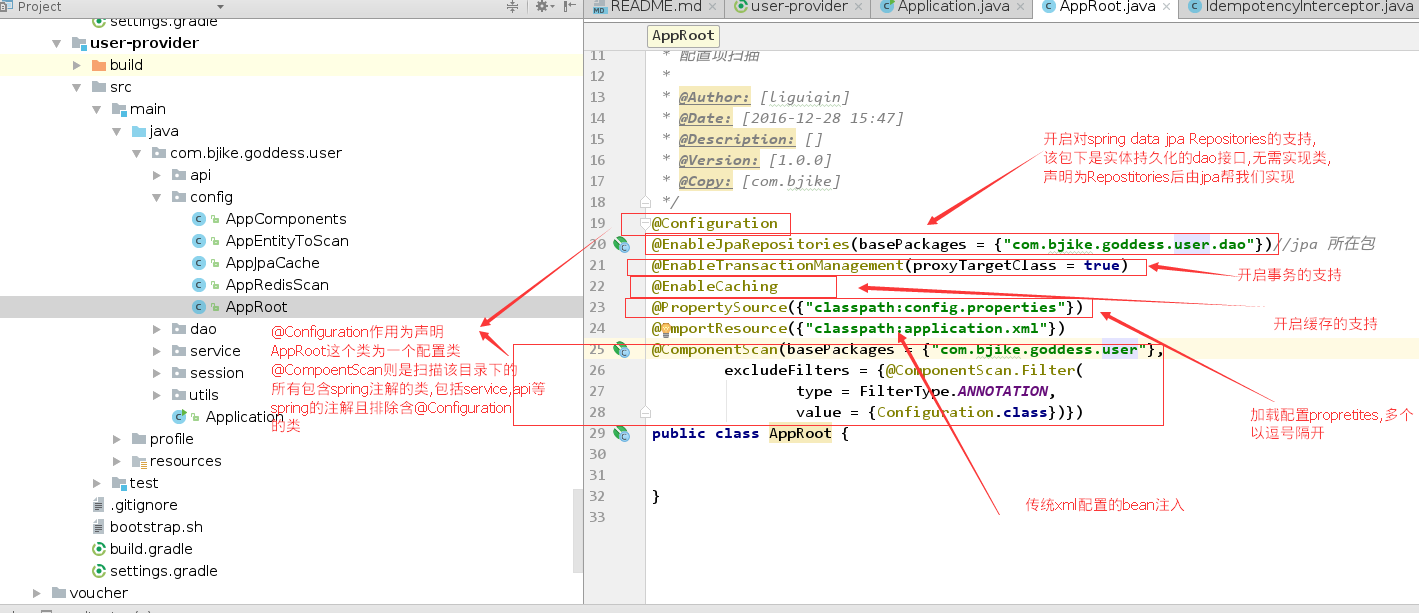
3.1.provider运行如何扫描对象

*注:AnnotationConfigApplicationContext是spring多种上下文应用之一,AnnotationConfigApplicationContext可以实现基于Java的配置类加载Spring的应用上下文.避免使用application.xml进行配置。在使用spring框架进行服务端开发时，注解配置在便捷性，和操作上都优于使用XML进行配置*

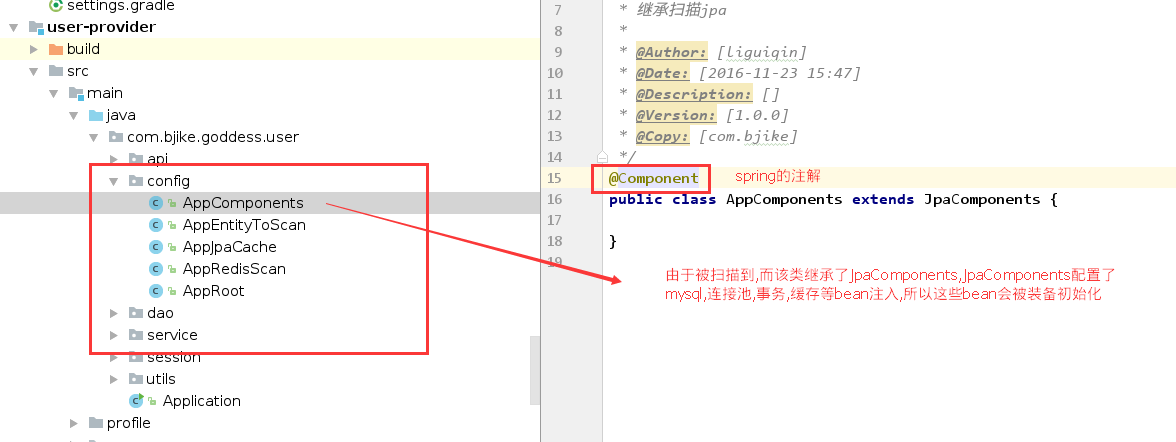
**

*下面我们来看看AppRoot.java这个主程序启动类的各注解作用:*

*简单解释下config下的所有配置扫描类,特别注意的是spring只会扫描属于spring的注解,不要误认为其他注解类也会被扫描,如entity包下的@Entity注解*

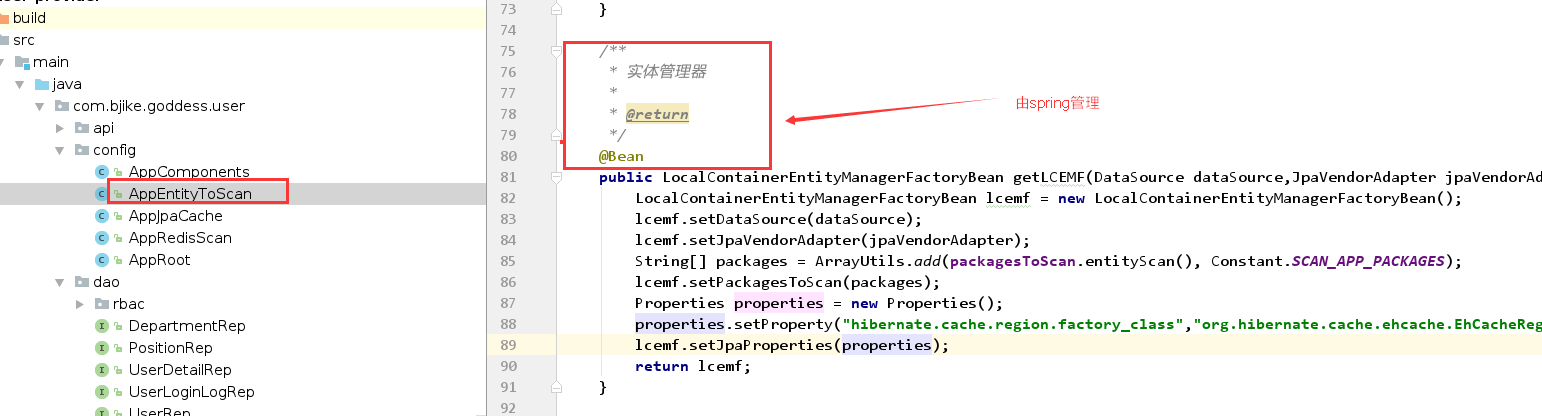
**

*AppComponents:*



AppEntityToScan(如上,同样是继承父类子字类扫描,让子类原因是解藕,控制权在子类):

Entity的注解由LocalContainerEntityManagerFactoryBean管理类注入,并由spring管理,见JpaComponents配置



其他的扫描配置类都是类似同样的方法,由子类重写注入,配置项父类包在

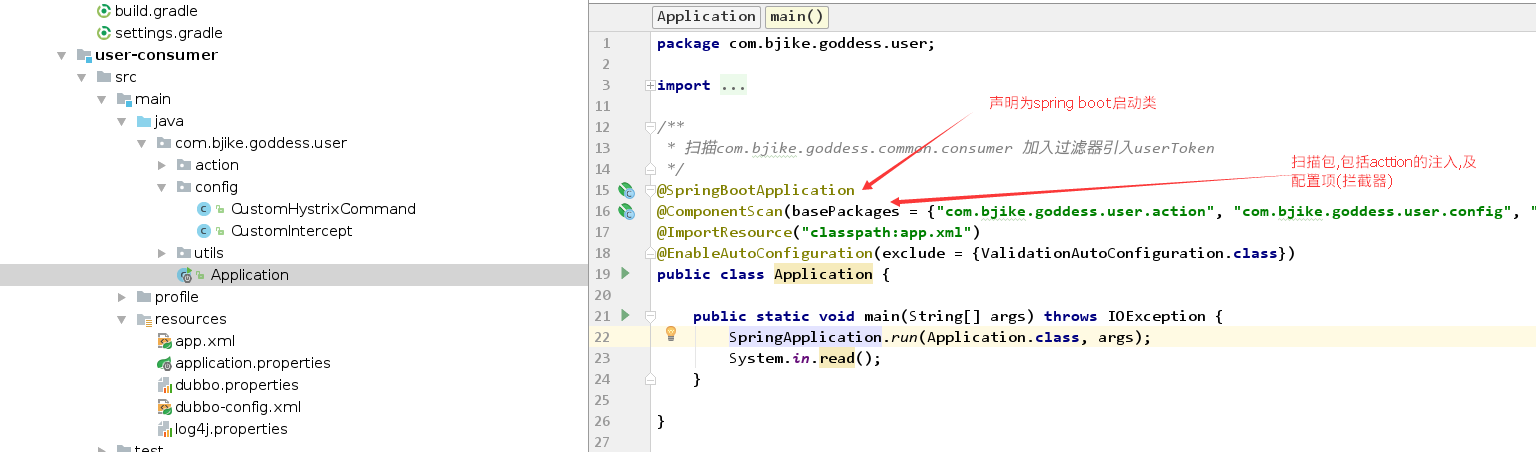
com.bjike.goddess.common.jpa包下,请自行参考

3.2 consumer的扫描

Spingboot是一个微服务的开源框架，它可用于创建可执行的spirng应用程序，采用了习惯优于配置的方法此框架的神奇之处在于@EnableAutoConfiguration

此注解自动载入应用程序所需的所有Bean,这依赖于Spring Boot在类路径中的查找,可以说搭建服务非常方便

如下图user-consumer:



扫描过程跟provider也是非常类似,见下图

