# 一、项目概述

基于Spring Boot、Spring Cloud & Spring Cloud Alibaba、OAuth2前后端分离微服务架构。

## 1、主要技术选型

* Spring Boot
* Spring Cloud & Spring Cloud Alibaba
* Spring Security
* Mybatis
* Vue

## 2、项目结构

**后端结构：**

com.oumasoft

├── ouma-ui // 前端框架 [80]

├── ouma-gateway // 网关模块 [8080]

├── ouma-auth // 认证中心 [9200]

├── ouma-api // 接口模块

│ └── ouma-api-system // 系统接口

├── ouma-common // 通用模块

│ └── ouma-common-core // 核心模块

│ └── ouma-common-datascope // 权限范围

│ └── ouma-common-log // 日志记录

│ └── ouma-common-redis // 缓存服务

│ └── ouma-common-security // 安全模块

│ └── ouma-common-swagger // 系统接口

├── ouma-modules // 业务模块

│ └── ouma-system // 系统模块 [9201]

│ └── ouma-gen // 代码生成 [9202]

│ └── ouma-job // 定时任务 [9203]

├── ouma-visual // 图形化管理模块

│ └── ouma-visual-monitor // 监控中心 [9100]

├──pom.xml // 公共依赖

**前端结构：**

├── build // 构建相关

├── bin // 执行脚本

├── public // 公共文件

│ ├── favicon.ico // favicon图标

│ └── index.html // html模板

├── src // 源代码

│ ├── api // 所有请求

│ ├── assets // 主题 字体等静态资源

│ ├── components // 全局公用组件

│ ├── directive // 全局指令

│ ├── layout // 布局

│ ├── router // 路由

│ ├── store // 全局 store管理

│ ├── utils // 全局公用方法

│ ├── views // 页面vue文件

│ ├── App.vue // 入口页面

│ ├── main.js // 入口 加载组件 初始化等

│ ├── permission.js // 权限管理

│ └── settings.js // 系统配置

├── .editorconfig // 编码格式

├── .env.development // 开发环境配置

├── .env.production // 生产环境配置

├── .env.staging // 测试环境配置

├── .eslintignore // 忽略语法检查

├── .eslintrc.js // eslint 配置项

├── .gitignore // git 忽略项

├── babel.config.js // babel.config.js

├── package.json // 版本依赖管理，启动scripts

└── vue.config.js // vue.config.js配置

## 整体架构图

1、网关：Spring Cloud Gateway

2、认证：OAuth2

3、服务注册与配置中心: Nacos

4、服务调用：OpenFeign

5、负载均衡：Ribbon

6、服务熔断：Hystrix

7、资源限流、熔断： Sentinel

8、监控： spring-boot-admin

9、部署：Docker、Kubernetes

## C:\Users\Administrator\Downloads\SpringCloudAlibaba架构图.png

# 二、环境准备

项目开发所需环境：

JDK >= 1.8 (推荐1.8版本)

Mysql >= 5.7.0 (推荐5.7版本)

Redis >= 3.0

Maven >= 3.0

Node >= 10

nacos >= 1.1.0

sentinel >= 1.6.0

IntelliJ IDEA (Ultimate)

## 1、安装JDK

因为开发工具IntelliJ IDEA需要JDK8以上环境，而且项目使用了Lambda等JDK8以上才有的新特性，所以安装的版本必须大于8。

JDK下载地址：

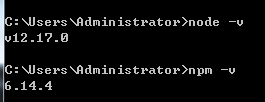
<https://www.oracle.com/technetwork/java/javase/downloads>

## 2、安装Node.js

使用vue-cli构建的前端项目是以NodeJS为基础的，前端工程运行和打包编译需要使用NodeJS，中文网下载地址：

<http://nodejs.cn/download/>

windows环境下载后是msi文件，安装完成之后命令行查看node和npm版本：

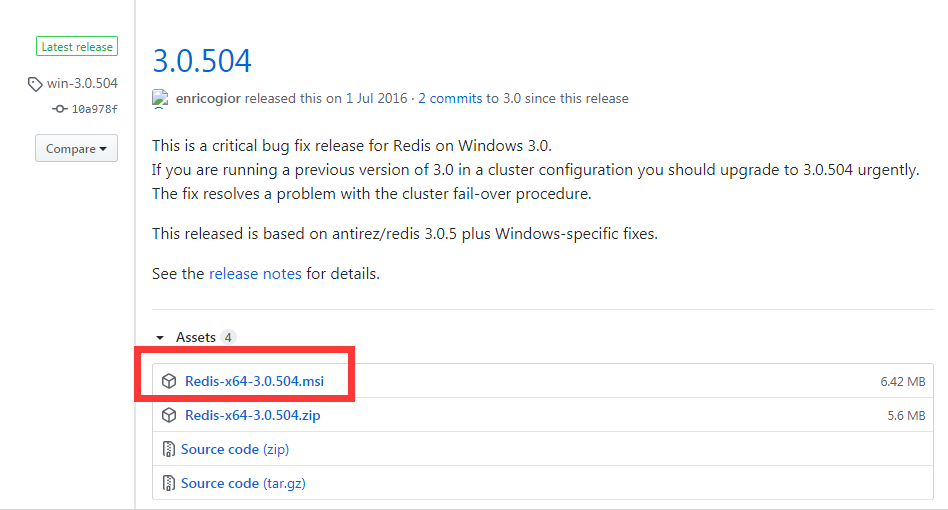


## 3、安装Redis

微服务使用Redis实现分布式缓存，以实现服务的无状态化。

Redis的github下载地址：

<https://github.com/MicrosoftArchive/redis/releases>



Windows环境下载msi文件直接双击安装运行。

## 4、安装MySQL

目前项目数据库使用的是MySQL，推荐安装的版本为5.7.x以上，下载地址：

<https://dev.mysql.com/downloads/windows/installer/5.7.html>

## **5、配置maven**

本项目使用maven作依赖管理，maven下载地址

https://maven.apache.org/download.cgi

下载完成之后解压缩，在conf下的settings.xml配置文件中添加阿里云镜像配置

<mirror>

<id>nexus-aliyun</id>

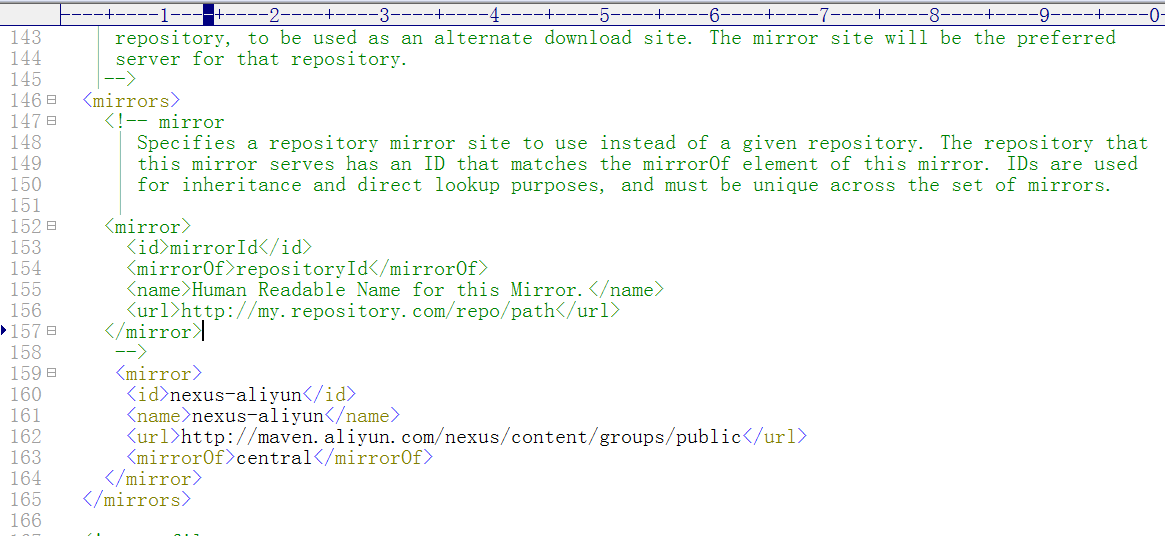
<name>nexus-aliyun</name>

<url>http://maven.aliyun.com/nexus/content/groups/public</url>

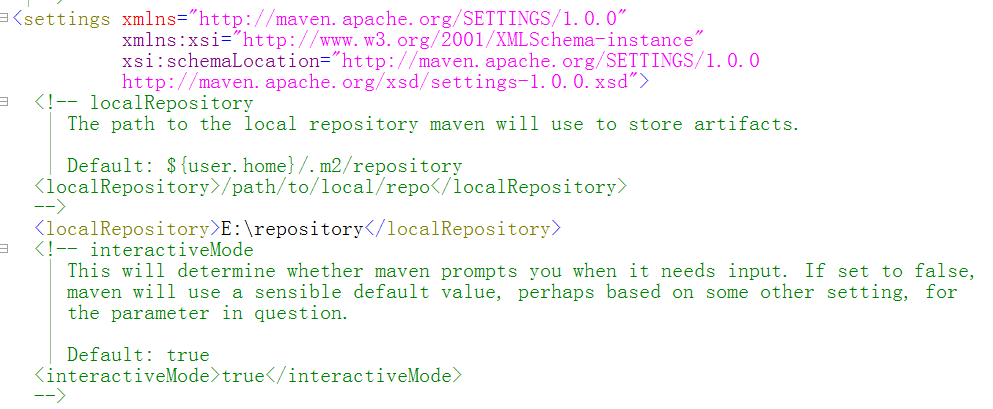
<mirrorOf>central</mirrorOf>

</mirror>

如下图

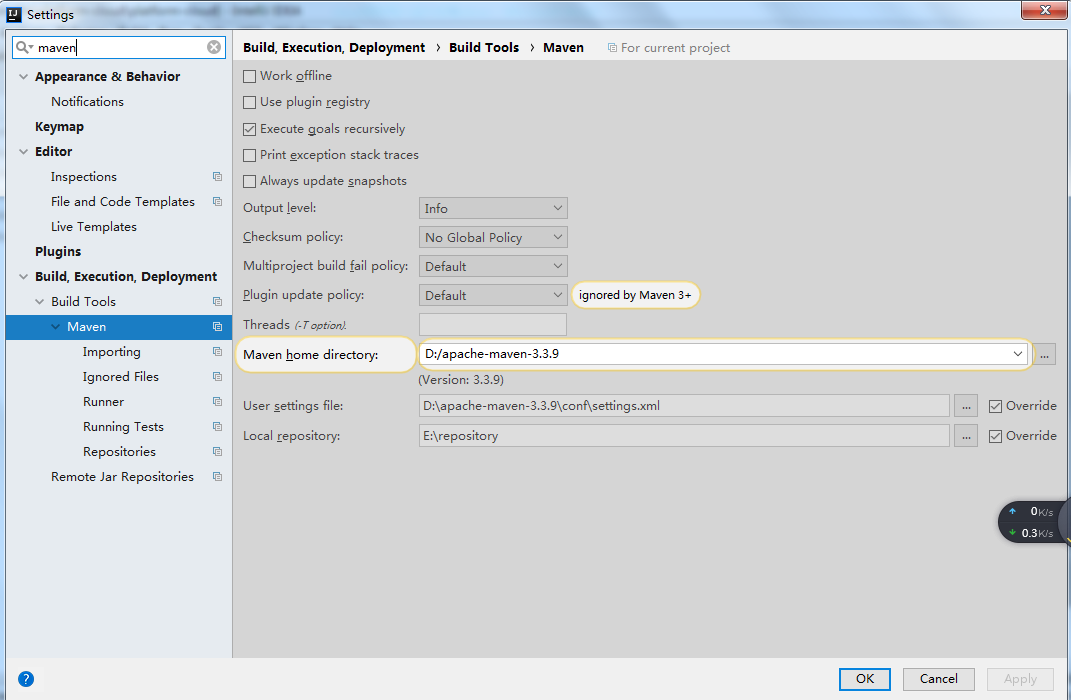


<localRepository> </localRepository>标签可修改默认仓库地址。例如放在E盘

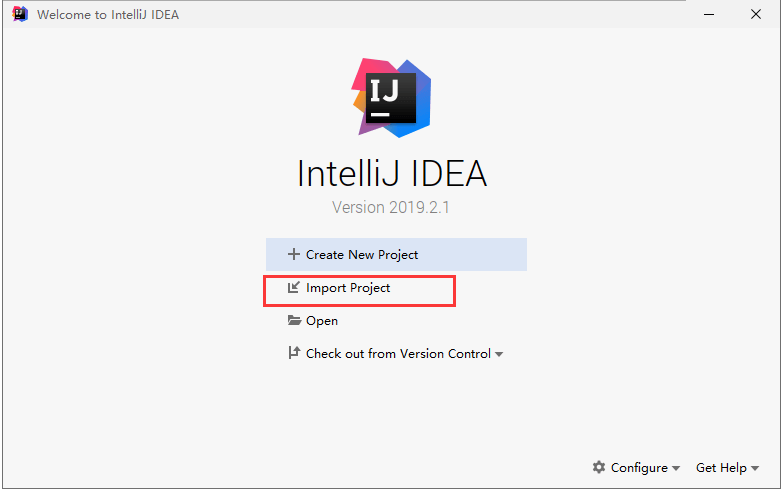


# 三、启动后端

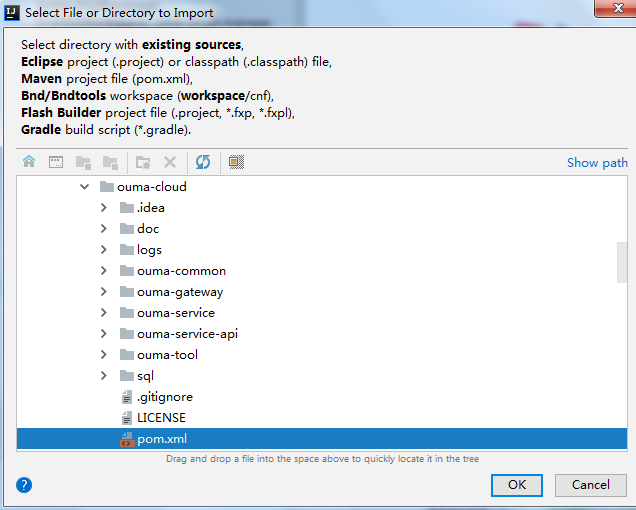
1、导入项目前配置好maven（以下示例的IDEA版本为2018.1.2，现在**Ultimate**最新版为2019.3.4，界面会稍有不同）

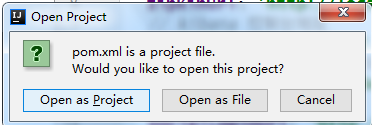


2、打开IntelliJ IDEA，选择Import Project：



选择项目ouma-cloud目录下的pom.xml：

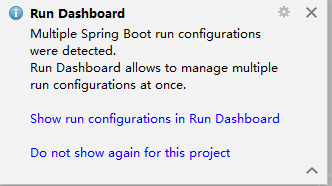




Open as Project点击OK，

导入项目后，等待Maven下载依赖。依赖下载完成之后，右下角会有个弹框，

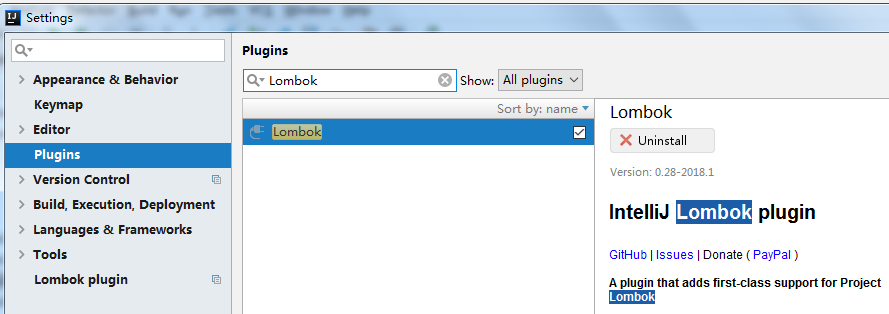
可以点击show run configurations， 这个窗口包含所有微服务启动类。



## 1、安装Lombok

[Lombok](https://www.projectlombok.org/)能够简化开发，项目编译运行需要在IDEA中先安装Lombok插件。

选择菜单 File -> Settings -> Plugins，输入Lombok查找，安装后需要重启IDEA

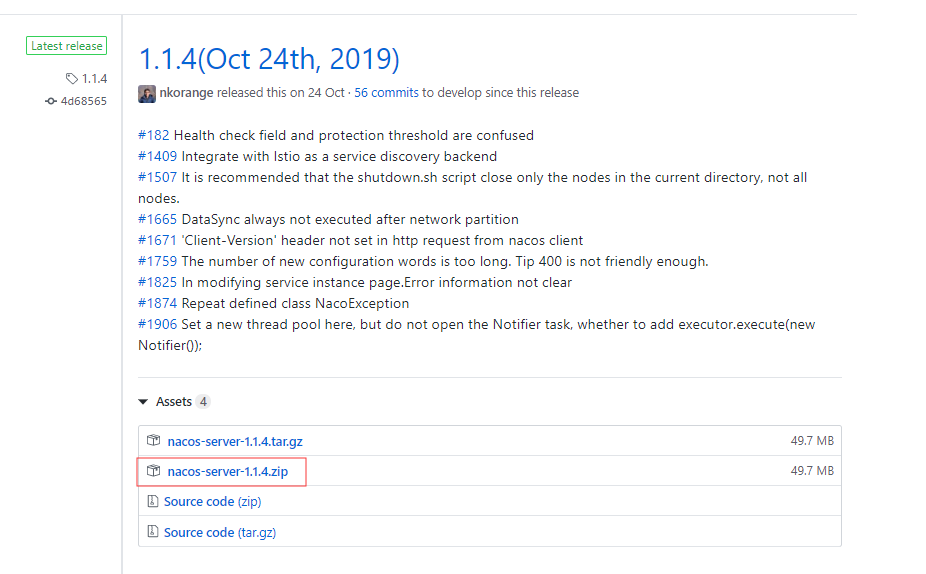


## 2、配置Nacos

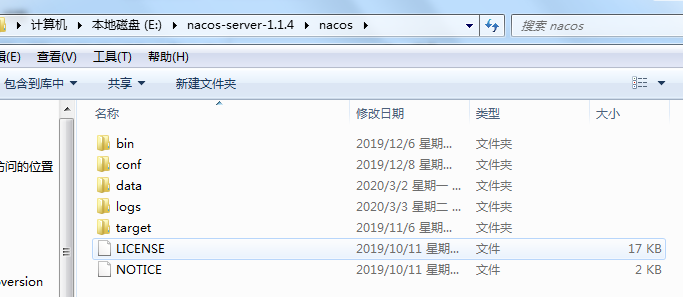
1、Nacos的release版本下载地址

<https://github.com/alibaba/nacos/releases>

选择下载zip文件，目前最新版为1.3.0（June 5, 2020）

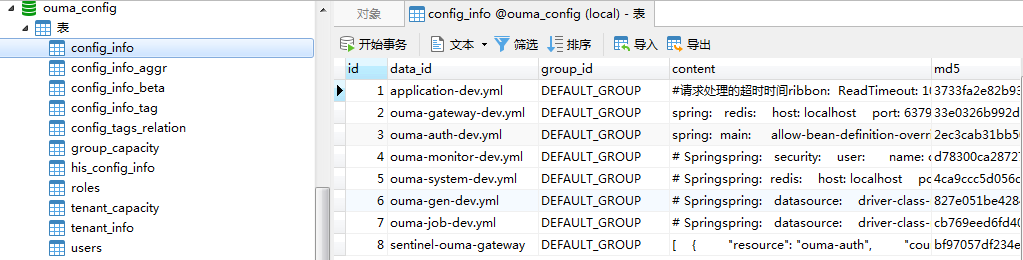


2、下载后解压目录如下：



bin目录存放windows和linux环境下运行和停止的脚本程序，conf目录主要有nacos的配置文件application.properties和用于持久化的脚本nacos-mysql.sql

3、使用Navicat或者SQLYog导入ouma-cloud/sql/ouma\_config.sql



4、配置nacos持久化

打开nacos目录下的conf/application.properties配置文件，在文件的最后增加支持mysql数据源配置：

spring.datasource.platform=mysql

db.num=1

db.url.0=jdbc:mysql://localhost:3306/ouma\_config?characterEncoding=utf8&connectTimeout=1000&socketTimeout=3000&autoReconnect=true

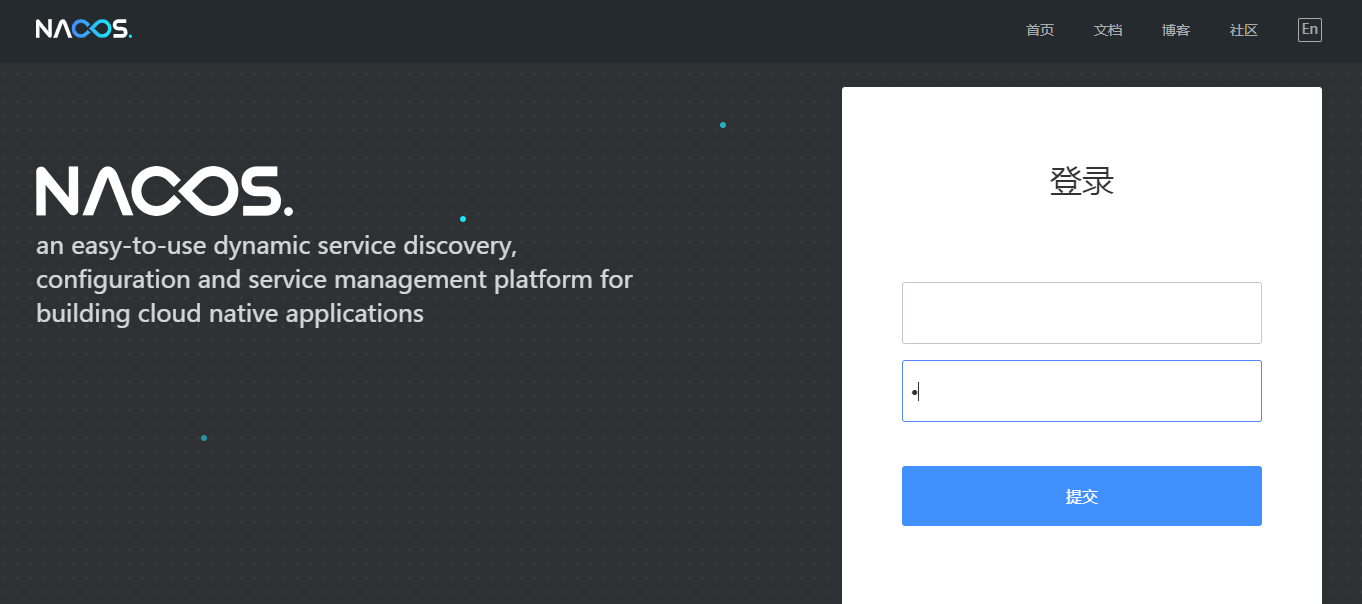
db.user=root

db.password=123456

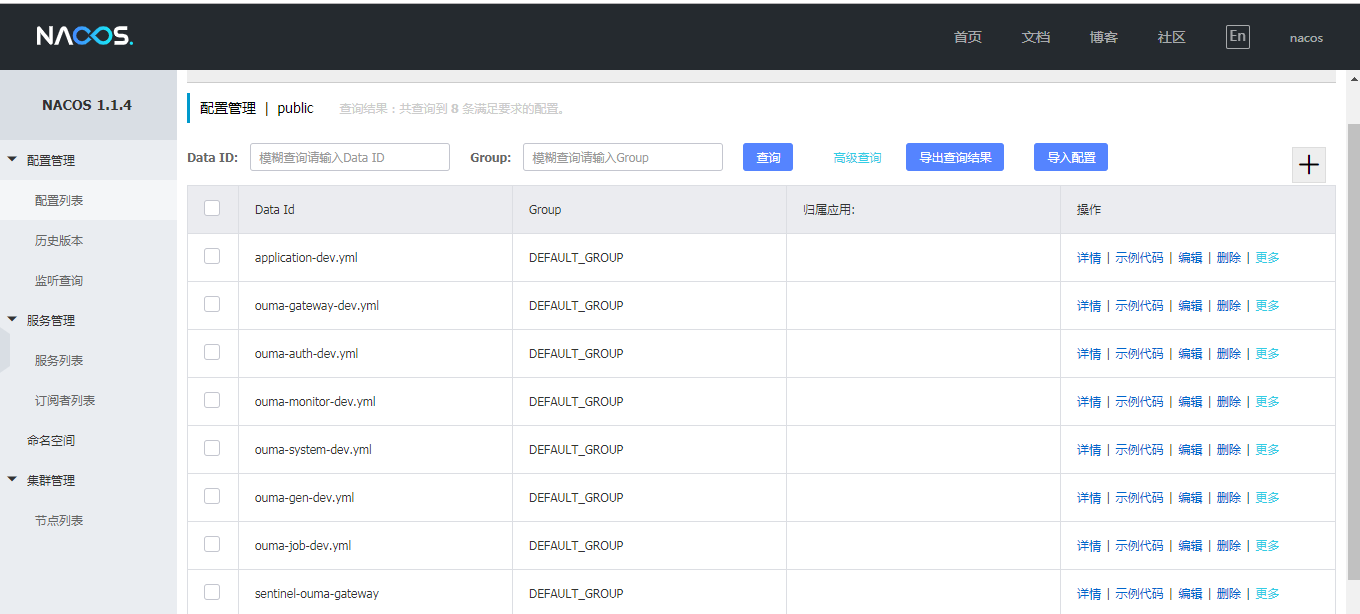
spring.datasource.platform表示使用的持久化数据库类型为mysql，db数据库配置修改成自己的连接端口、用户名、密码。

5、保存配置，双击bin/startup.cmd启动nacos 

浏览器访问nacos地址：<http://127.0.0.1:8848/nacos>



6、输入用户名nacos密码nacos登录

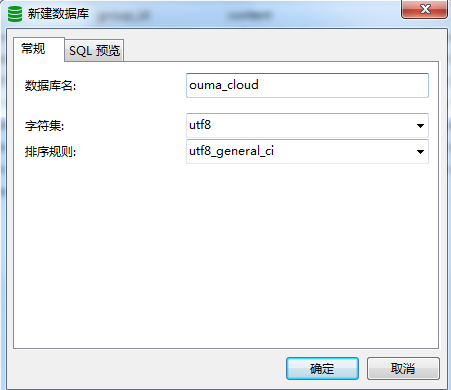


将ouma-gateway-dev.yml、ouma-auth-dev.yml、ouma-system-dev.yml、ouma-gen-dev.yml、

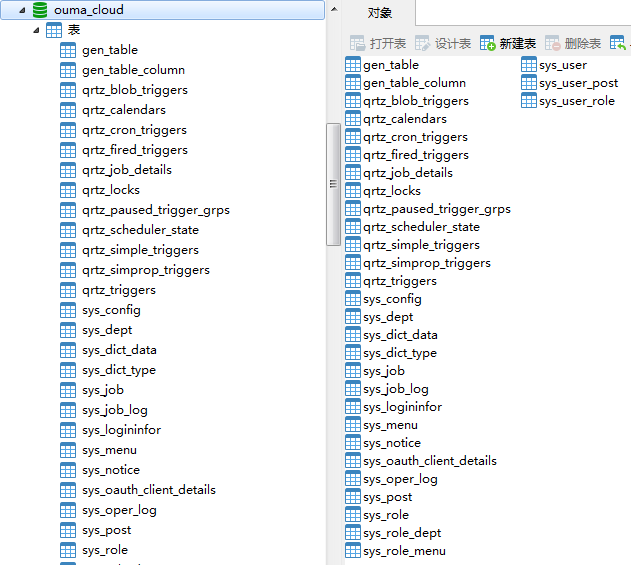
ouma-job-dev.yml文件里redis、mysql配置修改成自己的地址、端口、用户名、密码

## 3、创建数据库

使用Navicat（或SQLYog）连接MySQL数据库，新建ouma\_cloud数据库



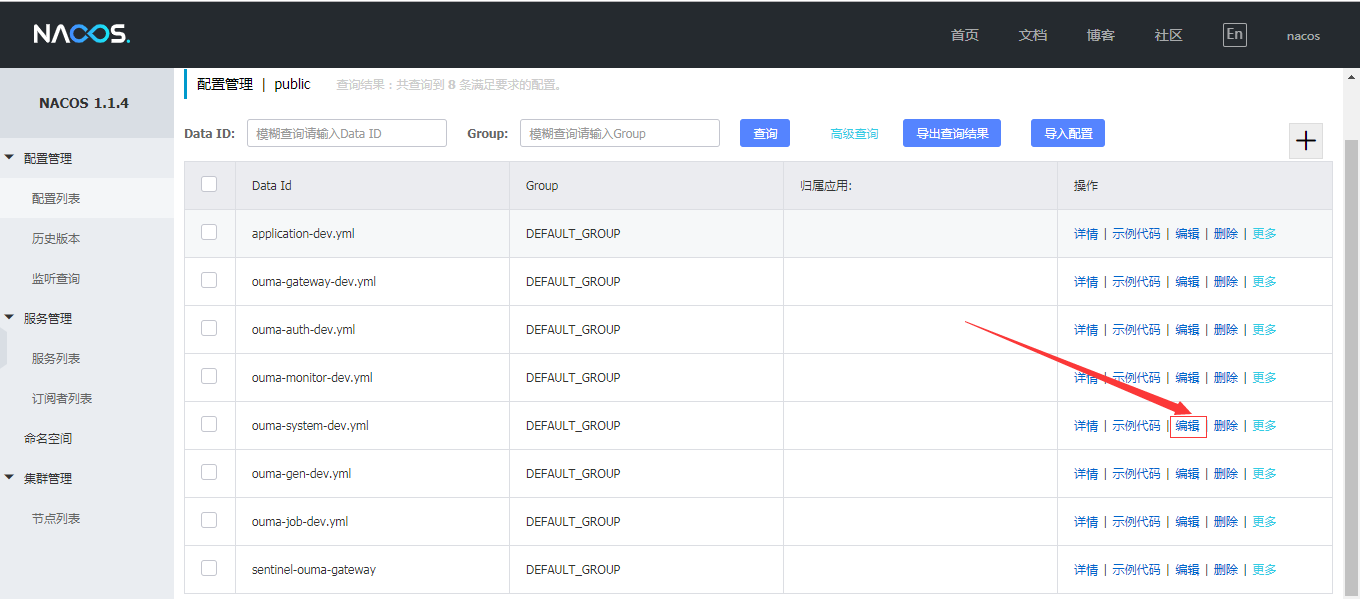
接着将ouma-cloud\sql目录下的ouma\_cloud.sql（系统基础库）和quartz.sql（定时任务）导入到该数据库中。

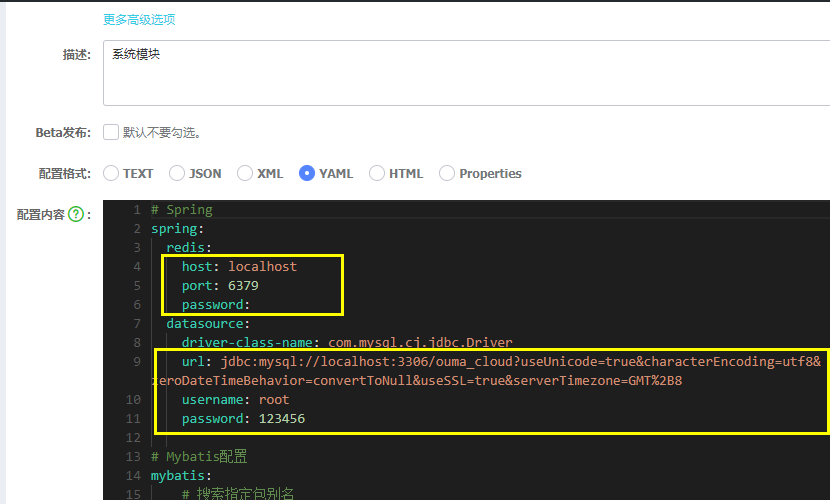


## 4、快速启动

（1）启动MySQL，Redis和Nacos服务

（2）Nacos控制台修改mysql和redis的连接配置。以system模块为例：



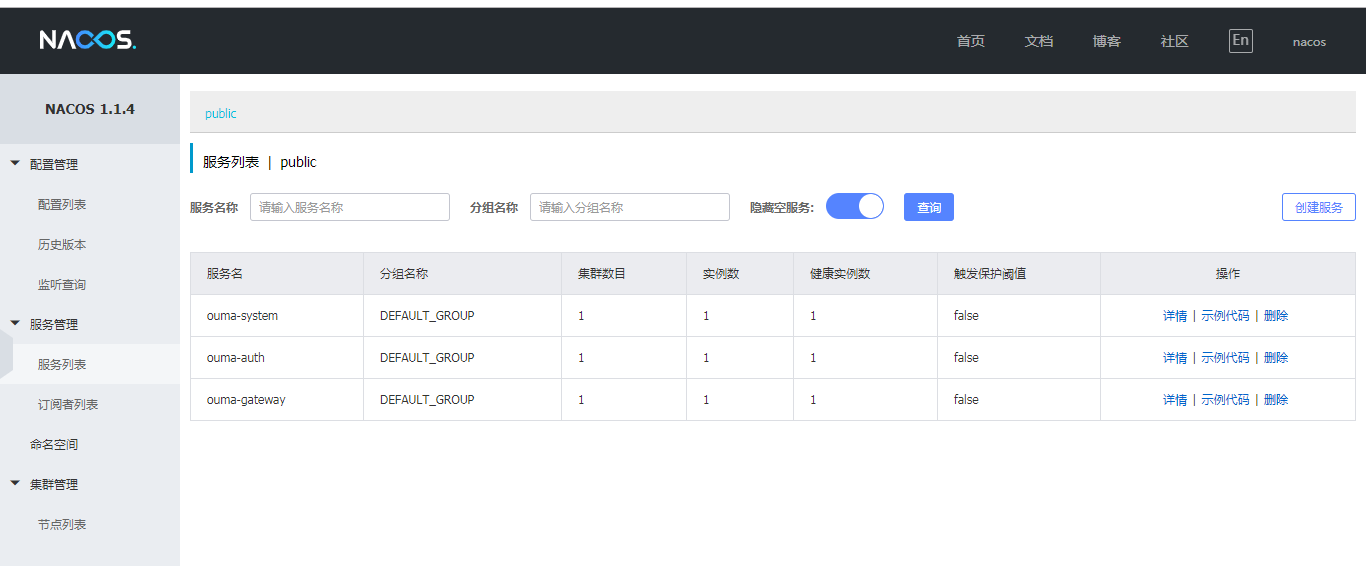


ouma-gateway-dev.yml、ouma-auth-dev.yml、ouma-system-dev.yml、ouma-gen-dev.yml、ouma-job-dev.yml都需要修改（目前redis和mysql还没有配置连接池）。修改完成之后点击“发布”按钮。

（3）本地启动(启动没有先后顺序)

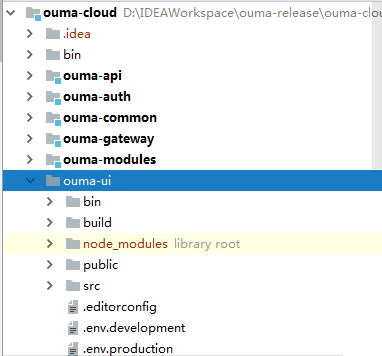
* OumaGatewayApplication （网关模块 必须）
* OumaAuthApplication （认证模块 必须）
* OumaSystemApplication （系统模块 必须）
* OumaMonitorApplication （监控中心 可选）
* OumaGenApplication （代码生成 可选）
* OumaJobApplication （定时任务 可选）

（4）微服务启动后，可以到Nacos控制台<http://127.0.0.1:8848/nacos>的服务管理->服务列表来查看是否注册成功。



# 四、启动前端

前端工程可用WebStorm（或者使用IDEA、 VSCode）直接打开单独作为一个前端项目启动，这里可直接作为一个模块运行。



# 进入项目目录

cd ouma-ui

# 安装依赖

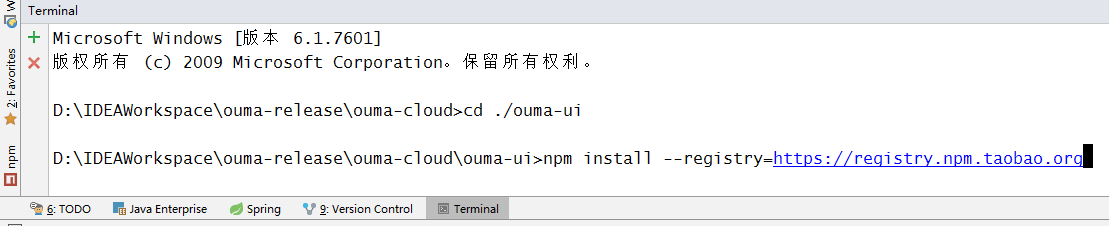
npm install

# 官方建议不要用直接使用 cnpm 安装，会有各种诡异的bug，可以通过重新指定 registry 来解决 npm 安装速度慢的问题。

npm install --registry=https://registry.npm.taobao.org

# 本地开发 启动项目

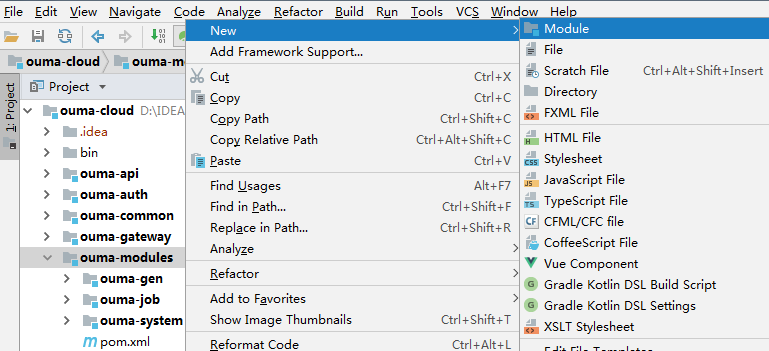
npm run dev



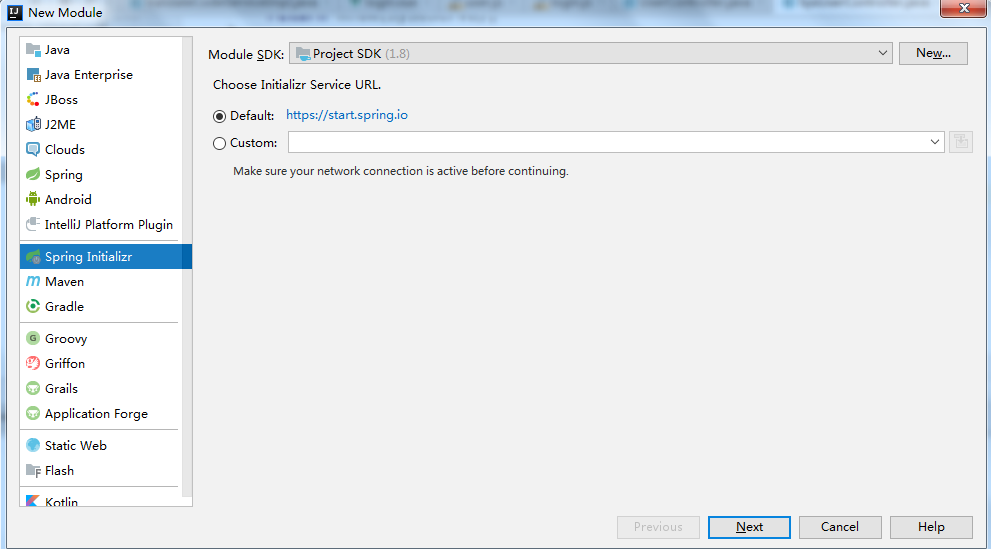
# 五、新增服务

## 1、创建模块

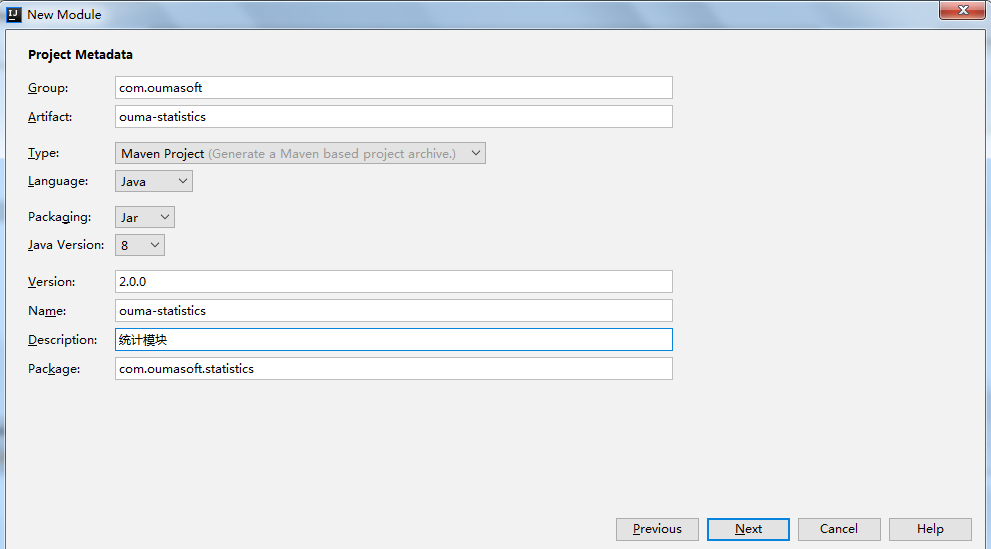
1. 业务模块统一在ouma-modules下创建。在ouma-modules目录上右键菜单选择 New -> Module



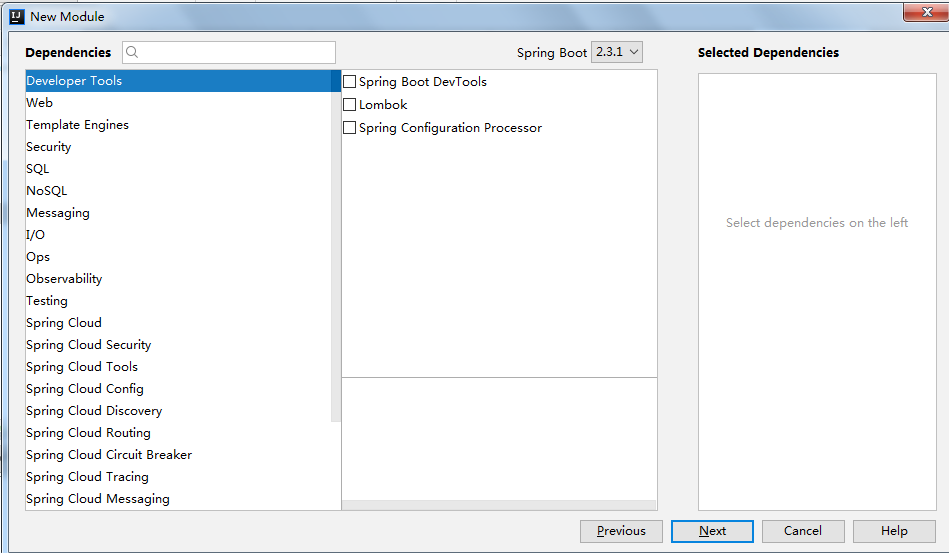
1. 左侧菜单选择Spring Initializr后点击Next



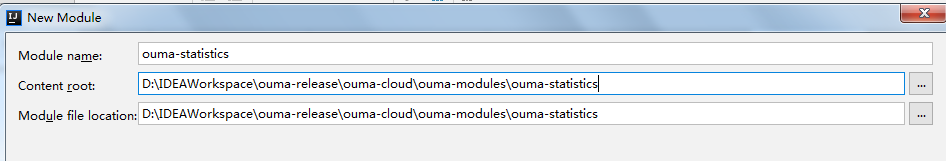
1. 以新增一个统计服务为例，填写如下内容



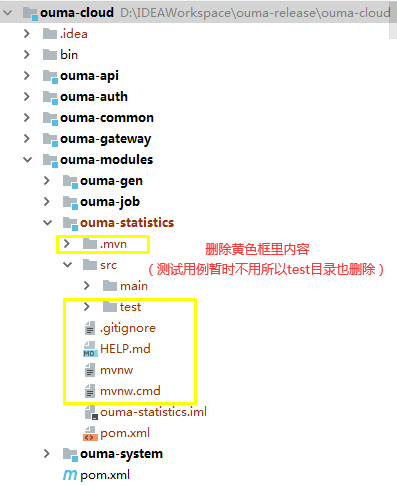
1. 点击Next，在Selected Dependencies界面什么也不选，接着点击Next



1. 注意路径，一定要在ouma-modules目录下创建ouma-statistics，然后点击Finish等待下载



1. 下载完成之后删除不必要文件。



1. 修改pom.xml文件，内容如下所示：

*<?***xml version="1.0" encoding="UTF-8"***?>*<**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"**>  
 <**modelVersion**>4.0.0</**modelVersion**>  
 <**parent**>  
 <**groupId**>com.oumasoft</**groupId**>  
 <**artifactId**>ouma-modules</**artifactId**>  
 <**version**>2.0.0</**version**>  
 </**parent**>  
 <**artifactId**>ouma-statistics</**artifactId**>  
 <**name**>ouma-statistics</**name**>  
 <**description**>统计模块</**description**>  
  
 <**dependencies**>  
  
 *<!-- SpringCloud Ailibaba Nacos -->* <**dependency**>  
 <**groupId**>com.alibaba.cloud</**groupId**>  
 <**artifactId**>spring-cloud-starter-alibaba-nacos-discovery</**artifactId**>  
 </**dependency**>  
  
 *<!-- SpringCloud Ailibaba Nacos Config -->* <**dependency**>  
 <**groupId**>com.alibaba.cloud</**groupId**>  
 <**artifactId**>spring-cloud-starter-alibaba-nacos-config</**artifactId**>  
 </**dependency**>  
  
 *<!-- SpringCloud Ailibaba Sentinel -->* <**dependency**>  
 <**groupId**>com.alibaba.cloud</**groupId**>  
 <**artifactId**>spring-cloud-starter-alibaba-sentinel</**artifactId**>  
 </**dependency**>  
  
 *<!-- SpringBoot Web -->* <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-web</**artifactId**>  
 </**dependency**>  
  
 *<!-- Swagger -->* <**dependency**>  
 <**groupId**>io.springfox</**groupId**>  
 <**artifactId**>springfox-swagger-ui</**artifactId**>  
 <**version**>${swagger.fox.version}</**version**>  
 </**dependency**>  
  
 *<!-- Mysql Connector -->* <**dependency**>  
 <**groupId**>mysql</**groupId**>  
 <**artifactId**>mysql-connector-java</**artifactId**>  
 </**dependency**>  
  
 *<!-- Ouma Common Security -->* <**dependency**>  
 <**groupId**>com.oumasoft</**groupId**>  
 <**artifactId**>ouma-common-security</**artifactId**>  
 </**dependency**>  
  
 *<!-- Ouma Common Log -->* <**dependency**>  
 <**groupId**>com.oumasoft</**groupId**>  
 <**artifactId**>ouma-common-log</**artifactId**>  
 </**dependency**>  
  
 *<!-- Ouma Common Swagger -->* <**dependency**>  
 <**groupId**>com.oumasoft</**groupId**>  
 <**artifactId**>ouma-common-swagger</**artifactId**>  
 </**dependency**>  
  
 </**dependencies**>  
  
 <**build**>  
 <**plugins**>  
 <**plugin**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-maven-plugin</**artifactId**>  
 <**executions**>  
 <**execution**>  
 <**goals**>  
 <**goal**>repackage</**goal**>  
 </**goals**>  
 </**execution**>  
 </**executions**>  
 </**plugin**>  
 </**plugins**>  
 </**build**>  
  
</**project**>

1. 修改ouma-modules的pom文件，添加Module

<**modules**>  
 <**module**>ouma-system</**module**>  
 <**module**>ouma-gen</**module**>  
 <**module**>ouma-job</**module**>  
 <**module**>ouma-statistics</**module**>  
</**modules**>

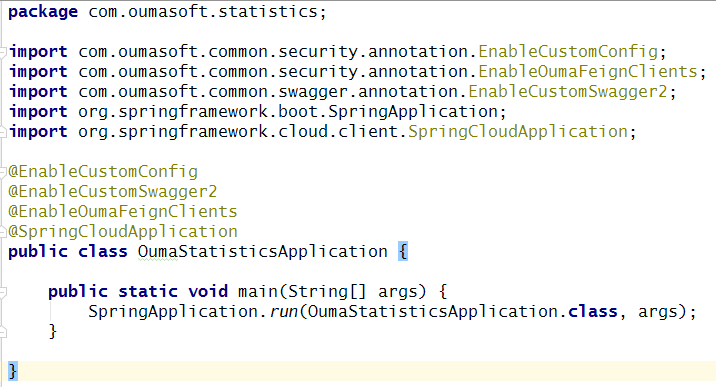
1. 删除resources目录下的application.properties文件，创建bootstrap.yml文件，添加如下内容：

**server:  
 port:** 9204  
  
**spring:   
 application:  
 name:** ouma-statistics  
 **profiles:** *# 环境配置* **active:** dev  
 **cloud:  
 nacos:  
 discovery:** *# 服务注册地址* **server-addr:** 127.0.0.1:8848  
 **config:** *# 配置中心地址* **server-addr:** 127.0.0.1:8848  
 *# 配置文件格式* **file-extension:** yml  
 *# 共享配置* **shared-dataids:** application-${**spring.profiles.active**}.${**spring.cloud.nacos.config.file-extension**}

1. 在resources目录下新建banner.txt（服务启动会读取此文件内容）和logback.xml日志文件。复制system模块的logback.xml内容，修改日志存放路径。

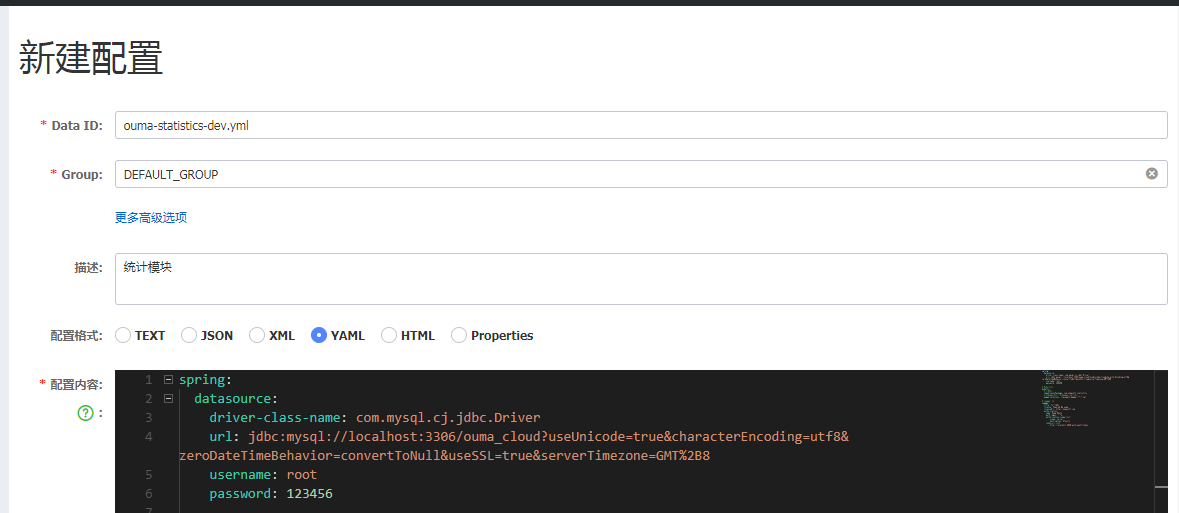
<**property name="log.path" value="/home/ouma/logs/ouma-statistics"** />

1. 启动类OumaStatisticsApplication上添加如下注解



## 2、添加配置

1. 登录nacos控制台，创建ouma-statistics-dev.yml配置



配置内容如下：

spring:

datasource:

driver-class-name: com.mysql.cj.jdbc.Driver

url: jdbc:mysql://localhost:3306/ouma\_cloud?useUnicode=true&

characterEncoding=utf8&zeroDateTimeBehavior=convertToNull&

useSSL=true&serverTimezone=GMT%2B8

username: root

password: 123456

# Mybatis配置

mybatis:

# 搜索指定包别名

typeAliasesPackage: com.oumasoft.statistics

# 配置mapper的扫描，找到所有的mapper.xml映射文件

mapperLocations: classpath:mapper/\*\*/\*.xml

# swagger 配置

swagger:

title: 统计模块接口文档

license: Powered By ouma

licenseUrl: http://oumasoft.com

authorization:

name: Ouma OAuth

auth-regex: ^.\*$

authorization-scope-list:

- scope: server

description: 客户端授权范围

token-url-list:

- http://localhost:8080/auth/oauth/token

1. 修改sentinel-ouma-gateway配置，添加如下限流规则配置

{

"resource": "ouma-statistics",

"count": 1000,

"grade": 1,

"limitApp": "default",

"strategy": 0,

"controlBehavior": 0

}

resource：资源名，即限流规则的作用对象

count: 限流阈值

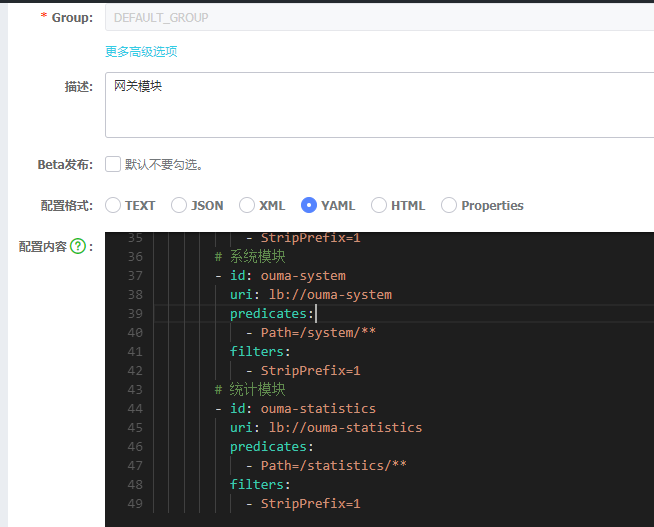
grade: 限流阈值类型（QPS或并发线程数）

limitApp: 流控针对的调用来源，若为default则不区分调用来源

strategy: 调用关系限流策略

controlBehavior: 流量控制效果（直接拒绝、Warm Up、匀速排队）

1. 修改ouma-gateway-dev.yml网关配置，添加ouma-statistics路由转发规则



新增的内容如下：

spring:

cloud:

gateway:

routes:

# 统计模块

- id: ouma-statistics

uri: lb://ouma-statistics

predicates:

- Path=/statistics/\*\*

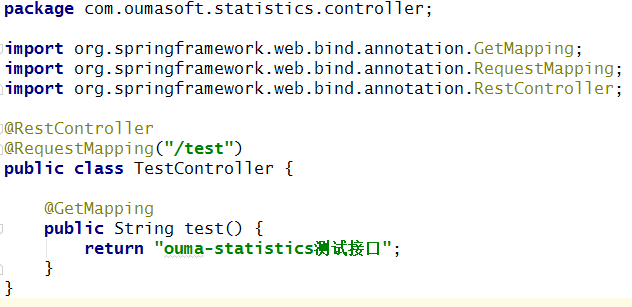
filters:

- StripPrefix=1

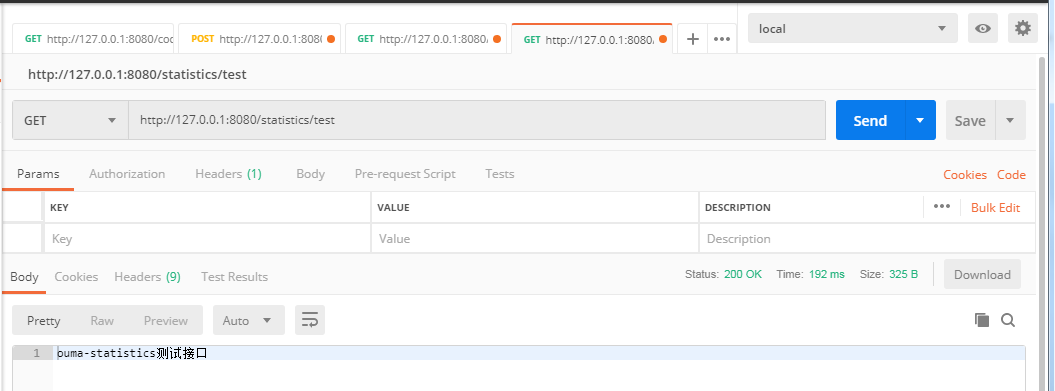
## 3、测试接口

1. 在com.oumasoft.statistics包下新建controller包，包下新建

TestController类



1. 启动服务，使用postman测试接口（先获取token令牌）

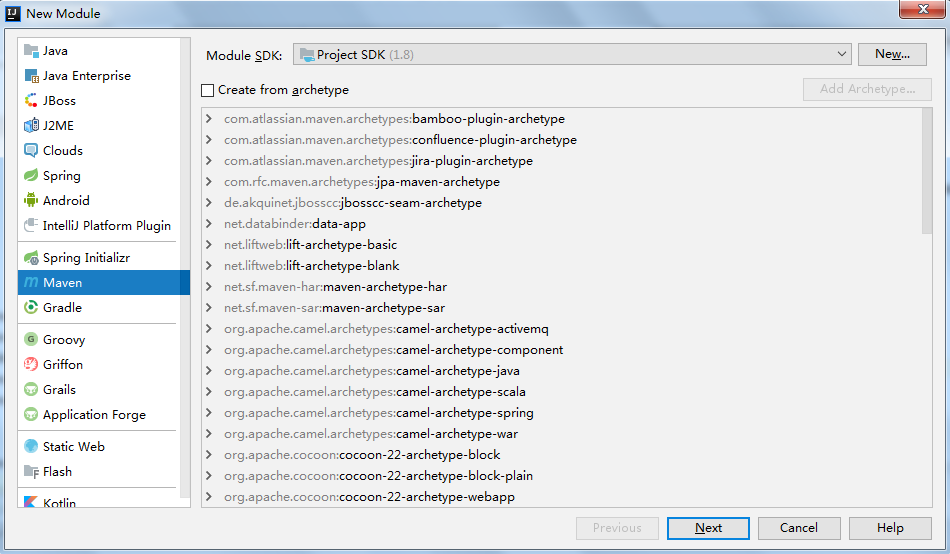


状态码200，成功返回。

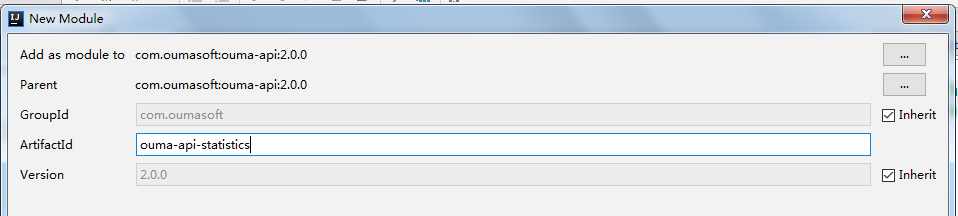
## 4、服务调用

微服务之间使用feign进行通信。参照ouma-api-system模块，我们来创建ouma-api-statistics模块，然后测试system服务调用statistics服务。

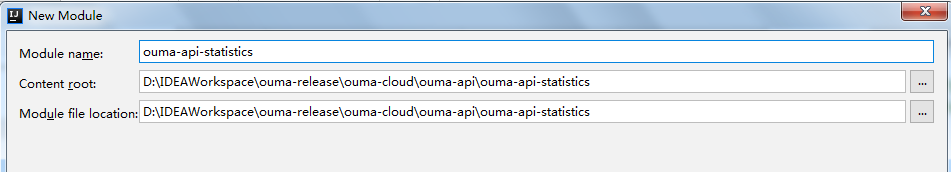
1. 右键ouma-api目录，选择New -> Module，左侧菜单栏选择Maven，点击Next



1. ArtifactId填写ouma-api-statistics，点击Next



1. 修改模块路径和名称，Finish



1. 待下载完成之后，删除test目录，修改pom文件，添加如下内容：

*<?***xml version="1.0" encoding="UTF-8"***?>*<**project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  
 <**parent**>  
 <**artifactId**>ouma-api</**artifactId**>  
 <**groupId**>com.oumasoft</**groupId**>  
 <**version**>2.0.0</**version**>  
 </**parent**>  
 <**modelVersion**>4.0.0</**modelVersion**>  
  
 <**artifactId**>ouma-api-statistics</**artifactId**>  
  
 <**description**>  
 ouma-api-statistics统计接口模块  
 </**description**>  
  
 <**dependencies**><**dependency**>  
 <**groupId**>com.oumasoft</**groupId**>  
 <**artifactId**>ouma-common-core</**artifactId**>  
 </**dependency**>  
 </**dependencies**>  
  
</**project**>

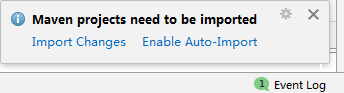
1. 修改ouma-cloud下的项目父级pom.xml，增加版本依赖管理。

*<!-- 统计接口 -->*<**dependency**>  
 <**groupId**>com.oumasoft</**groupId**>  
 <**artifactId**>ouma-api-statistics</**artifactId**>  
 <**version**>${ouma-cloud.version}</**version**>  
</**dependency**>

1. 修改ouma-system模块的pom.xml，增加statistics服务的api依赖。

<**dependency**>  
 <**groupId**>com.oumasoft</**groupId**>  
 <**artifactId**>ouma-api-statistics</**artifactId**>  
</**dependency**>

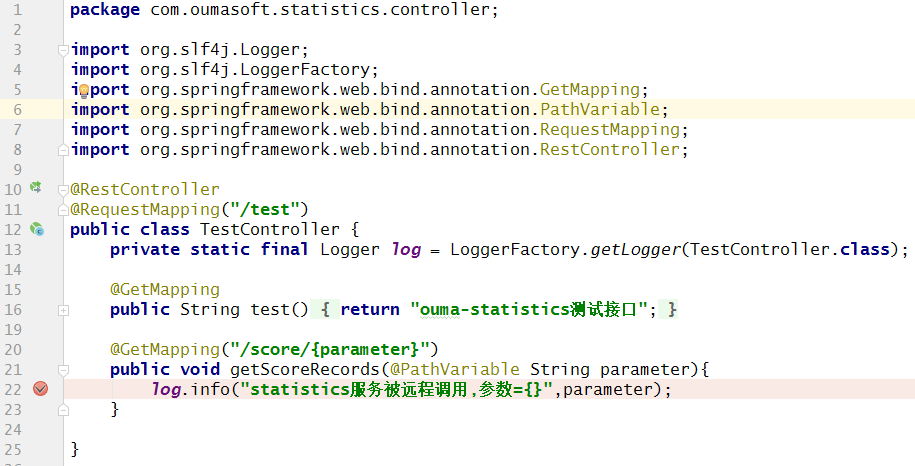
1. 右下角选择Import Changes，或者右键父级pom.xml，选择Maven ->Reimport 来重新导入模块。



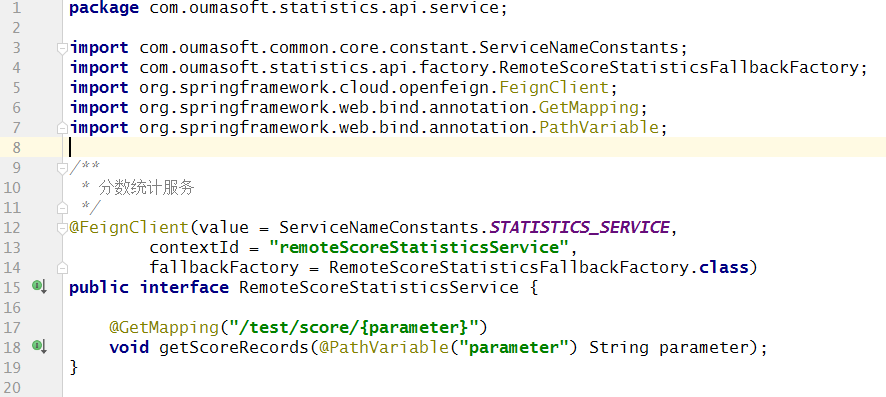
检查ouma-api的pom.xml是否添加了ouma-api-statistics

<**modules**>  
 <**module**>ouma-api-system</**module**>  
 <**module**>ouma-api-statistics</**module**>  
</**modules**>

1. 在ouma-statictics服务的TestController中继续添加一个测试接口以供system服务调用：



1. 新建com.oumasoft.statistics.api.service包，包下新建RemoteScoreStatisticsService接口

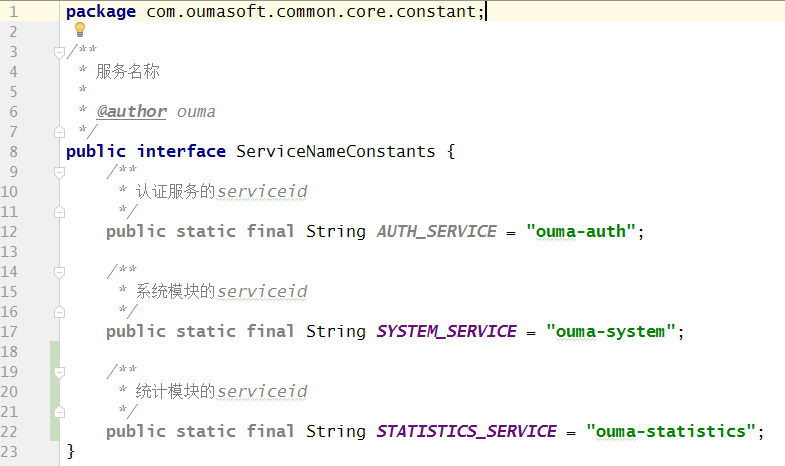


value：微服务的名称

contextId：指定别名

fallbackFactory：服务降级回退

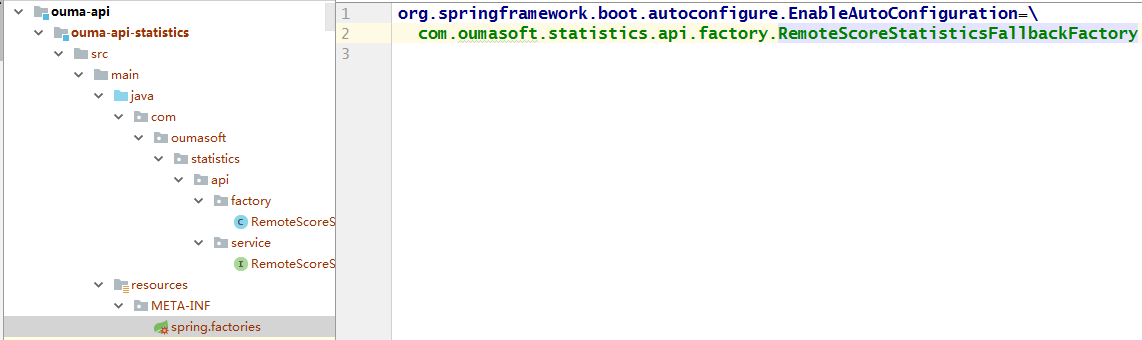
在com.oumasoft.common.core.constant. ServiceNameConstants增加服务名称常量：



新建com.oumasoft.statistics.api.factory包，包下新建用于处理方法回退类：

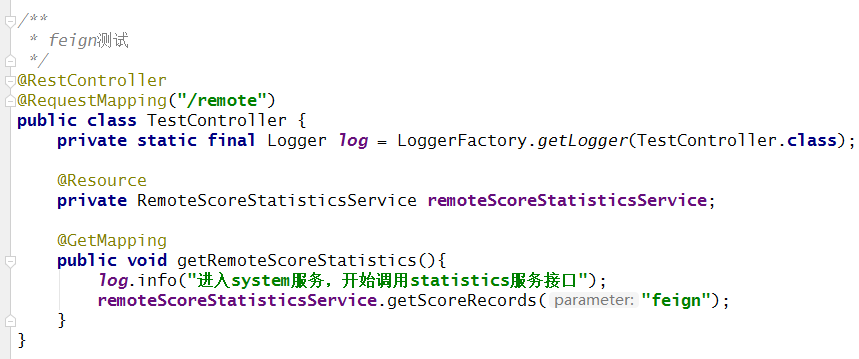


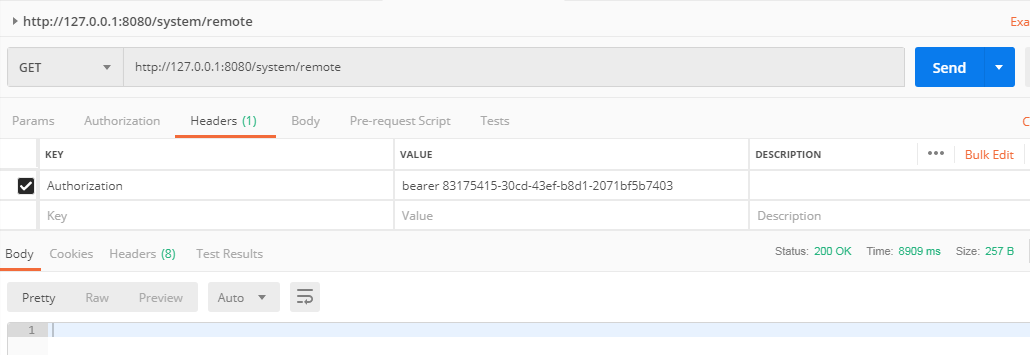
Resources目录下新建META-INF目录，META-INF目录下新增spring.factoties文件，添加如下配置（自动装配）：



1. 在ouma-system服务下新建TestController，增加一个接口作为调用

ouma-statistics的入口：



1. 重启服务，使用postman发送测试请求（先获取令牌）。

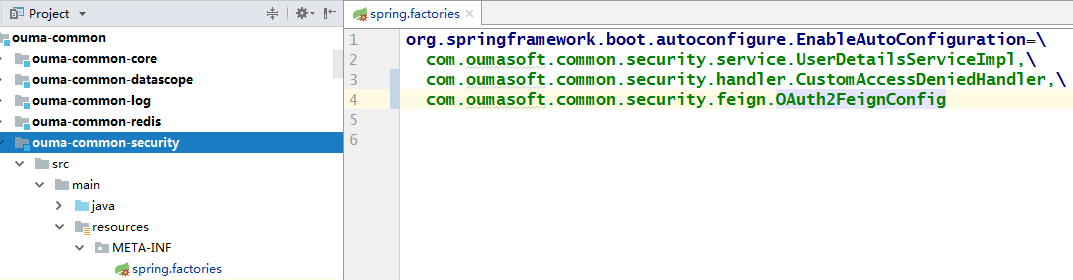
观察控制台输出内容：



可以看到服务调用失败，错误码401，这是由于

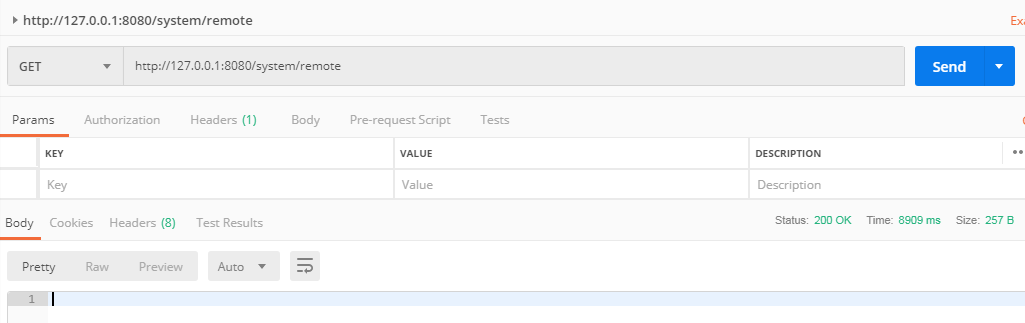
com.oumasoft.common.security.feign. OAuth2FeignConfig配置没有生效。我们暂时修改ouma-common-security模块的spring.factories文件，添加

com.oumasoft.common.security.feign.OAuth2FeignConfig



PS：这个小问题估计会在后续版本升级中修复，到时再做修改

1. 再次重启服务，发送请求：



查看控制台输出日志





statistics服务可以正常被调用

1. 测试feign回退。停掉ouma-statistics服务，再次发送测试请求，查看控制台输出内容：



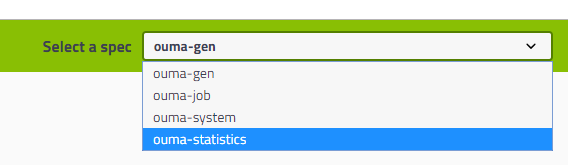
服务降级回退生效

# 六、使用Swagger

1. 启动服务，打开菜单，系统工具 -> 系统接口或者直接打开swagger地址：

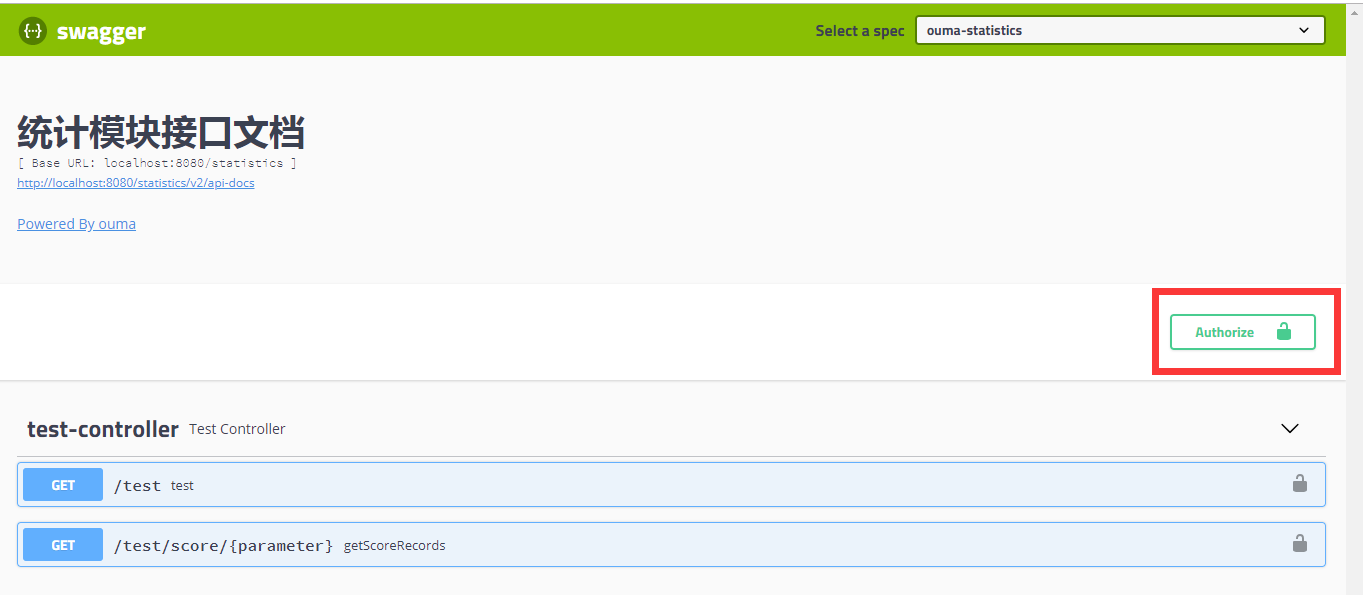
<http://localhost:8080/swagger-ui.html>

选择我们刚刚创建的ouma-statistics

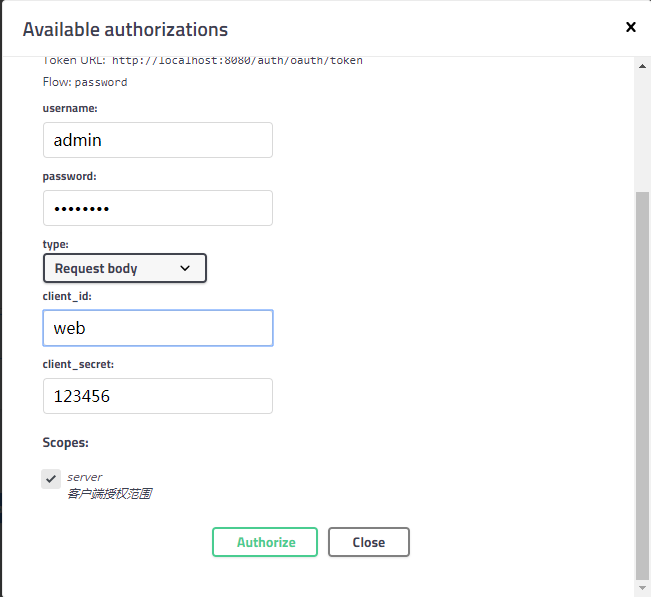


1. 默认swagger是受security保护的资源（有一把锁的标识），需要先获取授权。

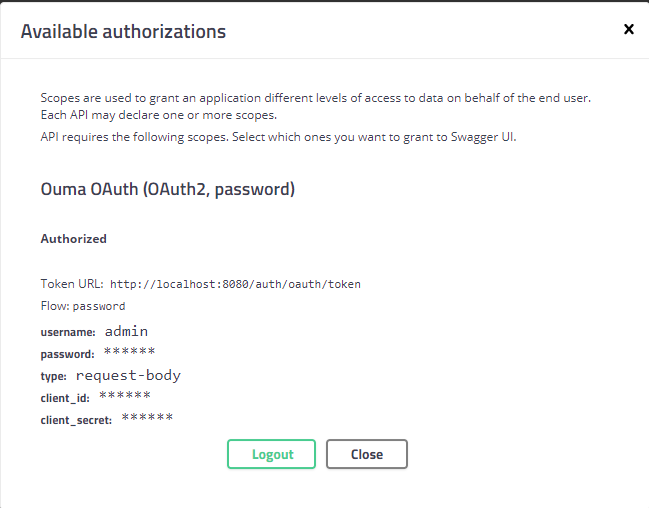
点击Authorize

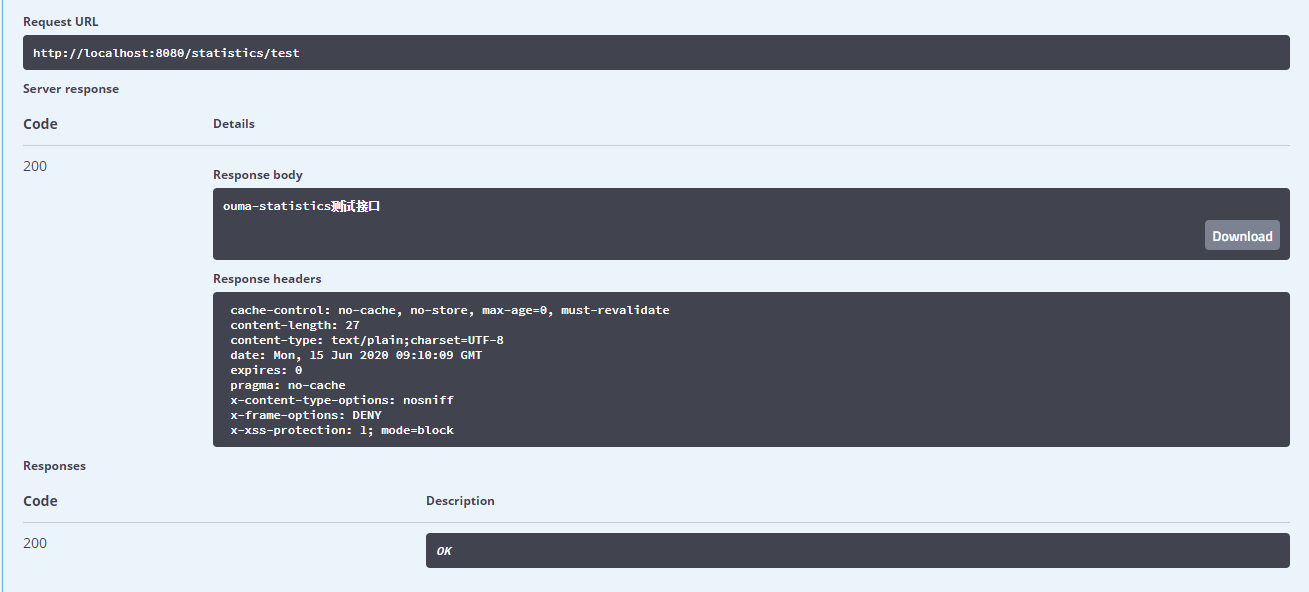


1. 弹出框填写如下信息：



Username和password 填写系统用户名密码，client\_id和client\_secret填写表sys\_oauth\_client\_details中的数据(client\_secret已加密)，这里使用的client\_id为web，Scopes勾选server选项。

1. 点击Authorize 
2. 发送测试请求，正常通过。



# 七、登录sba

1、启动OumaMonitorApplication监控，访问<http://localhost:9100/login>



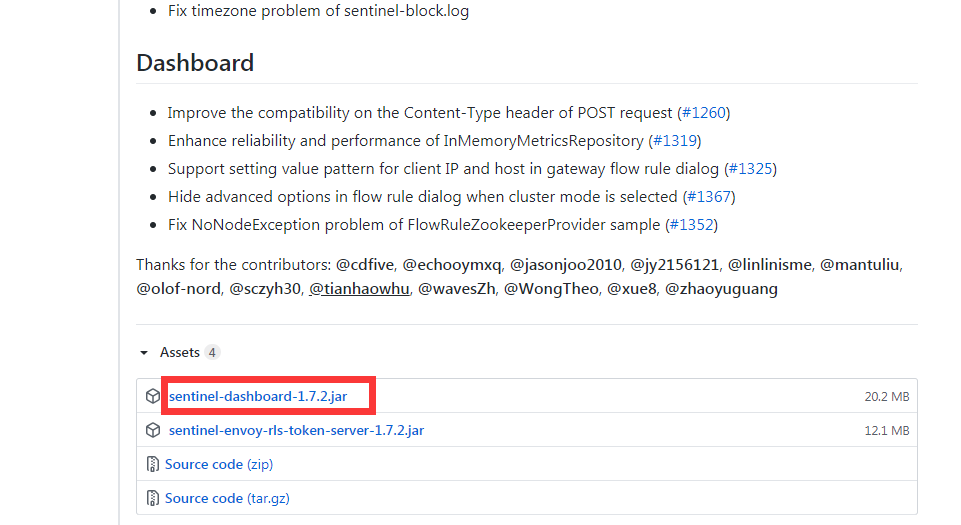
用户名oumasoft，密码123456，在ouma-monitor-dev.yml里配置

应用墙可查看启动示例，点击查看服务状态。

# 八、Sentinel控制台

1、sentinel Dashboard的release下载地址

<https://github.com/alibaba/Sentinel/releases>



1. 下载完成之后，执行启动命令

java -Dserver.port=8718 -Dcsp.sentinel.dashboard.server=localhost:8718 -Dproject.name=sentinel-dashboard -Dcsp.sentinel.api.port=8719 -jar F:\software\sentinel\sentinel-dashboard-1.7.0.jar

1. 访问[http://localhost:8718/#/login](http://localhost:8718/%23/login) 用户名密码都为sentinel

