# 一、在线教育简介

# 1 什么是在线教育

## 1.1 基本概述

在线教育顾名思义，是以网络为介质的教学方式，通过网络，学员与教师即使相隔万里也可以开展教学活动；此外，借助网络课件，学员还可以随时随地进行学习，真正打破了时间和空间的限制，对于工作繁忙，学习时间不固定的职场人而言网络远程教育是最方便不过的学习方式。

## 1.2 发展潜力

所有人离不开教育：早期教育、课外辅导、少儿英语、职业教育、出国留学、商学院、移民服务……而在信息化爆发式发展的趋势下，在线教育越来越凸显出优势：

1）在线教育可以突破时间和空间的限制，提升了学习效率；

2）在线教育可以跨越因地域等方面造成的教育资源不平等分配，使教育资源共享化，降低了学习的门槛。

基于在线教育的特点和优势，网络学校受到越来越多人的认可，各类新兴的网校及相关网站也不断涌现。显然，这代表着网校已经逐渐走进大众的生活并成为一种学习的主流趋势。因此很多人开始选择在线教育，特别是白领一族和大学生们。仅2012年一年，中国在线教育市场份额已经达到723亿元，且在线教育用户呈规模性放大。

## 1.3 适用行业

具体来说在线培训学习系统可适合于：

1）政府：现今我们的政府也提倡学习型组织，不断变化的政策环境、不断出现的新事物对政府公务员提出了更高的要求，而且政府机构的网络资源较佳，“在线培训系统”对公务员学习新知识和提高素质有很大帮助，更关键的是政府机构是垂直管理体制，只要在一个领域中创建并维护一套知识库，就可以让整个领域共享这宝贵的知识财富。

2）学校：随着网络的兴起，各大中学校可通过建立网上学校，加强学校、老师、学生之间的相互交流沟通，提高教学质量，亦可建立公共教学资源库，建设精品课程，宣传学校的教育实力。

3）行业：许多行业知识库体系庞大，专业多且层次深，因此行业一直注重知识和经验的积累，但这些宝贵的知识财富散落在各地，并没有利用和共享，因此，充分利用现有资源就能够创建一套丰富的知识库体系，让整个行业受益。

4）企业：企业的知识库体系通常是企业的核心竞争力，使用“在线教育培训系统”，企业能够创建自己的知识库体系，并允许企业内部员工随时随地学习和分享这些知识。不断提升的员工素质和不断积累的

企业知识库是企业能够保持长久的竞争力的关键。对于大型企业，还可以为合作伙伴及客户创建远程学习平台，提升和考核合作伙伴的专业技能并降低服务和支持成本。

# 二、十个行业分类

## 1、母婴

**市场现状：**

尽管母婴在线教育市场已经发展多年，但行业总体仍处于赚吆喝不赚钱的状态，主要原因在于国内垂直母婴网站大多存在同质化竞争激烈和盈利模式单一问题，从在线教育的内容来看，大部分母婴网站功能类似大多是基础的母婴知识库问题，咨询交流社区的内容，特色区分并不明显。

**未来发展：**

中国母婴产业是朝阳产业，现在处于快速发展的初期，国内每年将近有2000万新生儿，0到36个月的婴幼儿超过6000万，加上还有2000万左右的孕妈妈，市场规模已经超过万亿，所以发展母婴及儿童产品的在线教育市场，前景非常广阔。

**代表网站：**

妈妈网 https://www.mama.cn/

类似社交网站，视频频道跳转到腾讯视频


## 2、学前教育

**市场状况：**

大多数企业缺乏有效的产品形态，没有充分的开发在线教育的巨大市场。

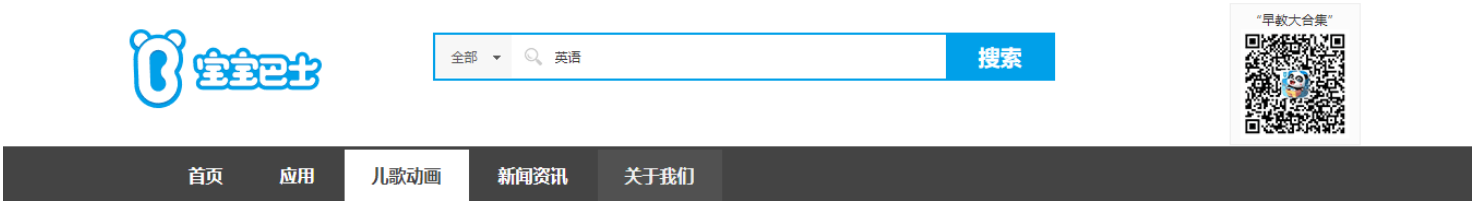
**未来发展：**

学前教育的市场，整体市场还处于起步阶段，随着针对孩子的培养和教育的重视，未来市场的发展将潜力无限。

**代表网站：**

宝宝巴士 https://www.babybus.com

网站以免费动画为主，核心产品是丰富的幼儿早教APP




# 3、少儿外语

**市场现状：**

国内在线少儿外语教育领域持续风起云涌，比如新东方VIP ABC，51talk，Englishbreak，海绵外语，爱卡微口语，魔方英语，沪江网校等为代表的英语在线教育公司，以新东方，好未来，英孚为代表的传统培训机构，以及直接瞄准这一细分市场的创业公司，例如VIPKID

**未来发展：**

少儿外语的市场虽然广阔，但是已经拥有多家培训机构，线下的成本高昂等诸多因素而倒闭，目前市场发展处于最初探索期，产品升级迫在眉睫，大部分家长对这种在线教育形式仍然存在疑虑，但有越来越多的家长开始接受，随着认可程度的提高，未来市场同样有一定的潜力。

**代表网站：**

VIPKID https://www.vipkid.com.cn/

外教一对一在线教学


# 4、中小学生

**市场现状：**

中小学在线教育呈现多样化发展，题库，英语家教等领域内均有产品获得千万美元以上投资，同时竞争压力在加剧，不断有创业者进入这个领域，或者传统教育机构开始布局，尤其在国内的一线城市，目前这方面影响力广泛的APP有很多，比如学而思网校学大教育网，一起作业网，猿题库文库，网游网学霸君等

**未来发展：**

O2O教育方式对中小学的影响极其深远，其中未来教育的发展中最重要的机会上门家教或者线下机构合作都能对原有的线上教育形成有效补充，目前几乎所有的教育机构都在尝试O2O模式

**代表网站：**

学而思 https://www.xueersi.com

录播、直播、一对一


## 5、高校学生

**市场现状：**

目前市场主要集中在学历教育方面，国家对于网校的毕业证正在逐步认可，但是这方面教学的内容需要较高的知识基础，而且需要教学机构相当大的影响力，除了学历教育以外，就是学校自己开发的在线课程平台，专业性课程主要对内部学生开放，就其他基础性课程对公开课对外开放。

**未来发展：**

对于学历教育而言，无法有效的做到O2O模式，一般以移动端的学历考证知识的学习为主，现代社会对个人水平要求越来越高，在未来的趋势中，大学相关的在线教育领域会迎来一个快速的发展期。

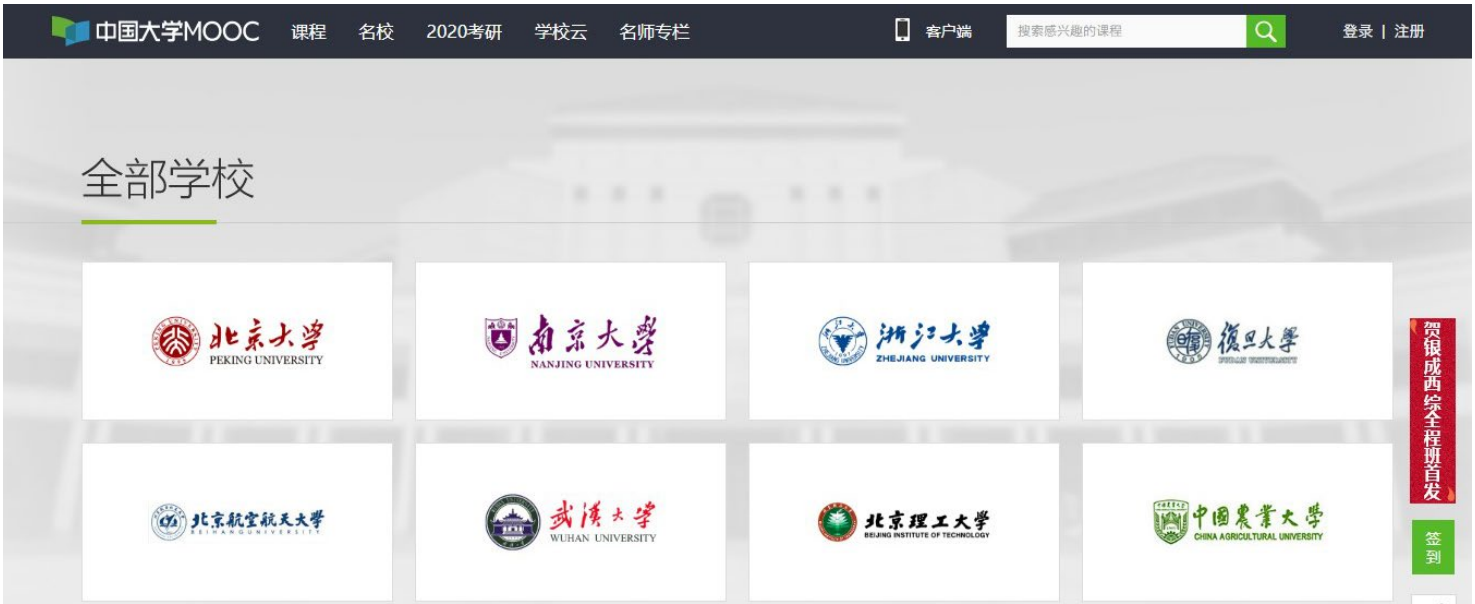
**代表网站：**

中国大学慕课 http://www.icourse163.org

由高教社联手网易推出，让每一个有提升愿望的用户能够学到中国知名高校的课程，并获得认证。

学堂在线 http://www.xuetangx.com/

由清华大学研发出的中文MOOC，面向全球提供在线课程。



## 6、留学

**市场现状：**

教育部统计显示，中国目前每年的出国留学生总数在四十万人左右，其中本科及以下层面就读的人数增长迅猛，低龄化趋势明显，在线教育市场向二三线城市蔓延，与留学相关的在线教育培训机构迅速增加。

**未来发展：**

未来的发展离不开全球化，对于个人而言留学是一个增加阅历，提高教育经历，获得更高发展方式，未来的需求量将不断增长，同样进入该行业的教育机构或企业，也会增加金融教育机构与留学机构的结合是大势所趋。

**代表网站：**

启德考培在线 http://qide.edusoho.cn/

启德教育旗下的在线视频网站


## 7、职业考试

**市场现状：**

职业教育的投融资情况表现稳定，目前仍是在线教育领域内的热门投资板块，中国在线教育市场中职业教育的占比高达30%以上，另外官方也鼓励发展职业教育。

**未来发展：**

职业考试主要是针对一些与职业相关的证书的考试，但是考试类的证书需要国家教育机构承认，因此对进入在线职业教育领域的要求较高。

**代表网站：**

中公教育 http://www.offcn.com/

公务员考试，各种职业资格认证考试，线下培训和网校


## 8、职业技能

**市场现状：**

职业技能的培训，是目前在线教育市场发展迅速的领域，其中一些企业已经形成了一定的品牌，如腾讯课堂，网易云课堂，51CTO、中国会计网校等，总体占据了市场的85%使用率，另外现有在线职业教育服务和it培训等，聚焦于垂直领域，专业性虽强，但服务过于分散，规模较小，平台化在线职业教育有望通过提升用户搜索效率，降低寻找成本而获得用户青睐。

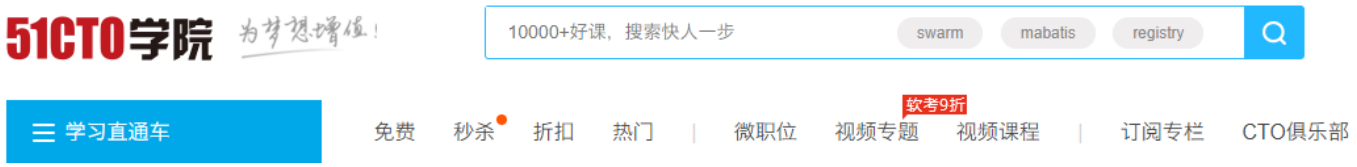
**未来发展：**

职业培训的未来市场十分广阔，并且与其他领域的盈利模式不同，容易盈利，大部分在线教育平台都是依靠这个领域的课程获得一定的利润。职业技能在未来的发展，潜力巨大，资格证书有望发展为线上服务的核心资源，但是对于新创企业而言，进入其中分享一块蛋糕并不是很容易，其他巨头占据了市场的主要份额，如果选择垂直化某个细分领域的进行运作，可获得意外的惊喜。

**代表网站：**

51cto http://edu.51cto.com/

最早一批的it职业技能网站




# 9、成人外语

**市场现状：**

与少儿外语更注重基础性不同，成人外语主要是培优业务较多，更注重高水平的外语知识，同时小语种的学习人数也在增多。

**未来发展：**

随着从业者增加，在线教育，语言学习领域竞争不断加剧，部分语言学习产品因此竞争增加，提前遭遇发展瓶颈，越来越向大型语言教育机构汇聚，功能细分的小型垂直机构，未来的生存将更加艰难。O2O模式对于成人外语同样重要语言培训课程在线下拥有更高的用户体验，与传统培训机构合作，也能缓解线上线下的竞争关系

**代表网站：**

沪江外语 https://www.hujiang.com/

## 10、个人兴趣

**市场现状：**

超过30%的用户表示在网上学习是满足个人的兴趣爱好，公开数据显示中国的兴趣爱好市场规模约为500亿元。

**未来发展：**

社会主流人群可支配收入的增长，促使兴趣培训市场，进一步提升，个人兴趣领域，在未来空间发展可以与k12教育基础教育的相媲美，随着个体对于精神满足的追求，国内市场将迎来一个长期稳定的发展期。

**代表网站：**

美食杰 https://www.meishij.net/

美食菜谱类，有图文和视频


# 二、八种商业模式

## 1、C2C模式（Consumer To Consumer 平台模式 ）

用户到用户，这种模式本质是将自己的流量或者用户转卖给视频或者直播的内容提供者，通过出售内容分成获利。

平台模式避开了非常沉重的内容和服务，扩张迅速，但实际这种模式也有缺陷，在线教育这两年的发展使内容迅速贬值，比较难带来更免费用户和流量。

**代表网站：**

51cto http://edu.51cto.com/

腾讯课堂 https://ke.qq.com/

## 2、B2C模式（Business To Customer 会员模式）

商家到用户，这种模式是自己制作大量自有版权的视频，放在自有平台上，让用户按月付费或者按年付费。 这种模式简单，快速，只要专心录制大量视频即可快速发展，其曾因为 lynda 的天价融资而 大热。但在中国由于版权保护意识不强，教育内容易于复制，有海量的免费资源的竞争对手众多等原因，难以取得像样的现金流。

**代表网站：**

lynda https://www.lynda.com/

慕课网 https://www.imooc.com/

谷粒学院 http://www.gulixueyuan.com/

## 3、B2B2C（商家到商家到用户）

平台链接第三方教育机构和用户，平台一般不直接提供课程内容，而是更多承担教育的互联网载体角色，为教学过程各个环节提供全方位支持和服务。

**代表网站：**

51cto http://edu.51cto.com/

腾讯课堂 https://ke.qq.com/

## 4、垂直领域

这种模式需要糅合录播，直播，帮助服务等多种手段，对学生学习某一项内容负责。这种模式收费高，有较强的壁垒。这种产品一旦形成口碑，会有稳定的用户群和收入，但产品非常复杂，难度大，门槛高，即使单独一个项目都会耗费大量的人力物力，因此发展速度较慢。

**代表网站：**

51cto的微职位 http://edu.51cto.com/

网易云课堂的微专业 https://study.163.com/

## 5、直播、互动

这种模式将传统课堂上的反馈，交互，答疑搬到线上。让用户容易接受，只要服务贴心，用　户就愿意买单，因此有丰富现金流。但缺陷是只能通过平台吸引用户，造成了竞争门槛过低，　　模式雷同，对手众多，收益的永远是拥有流量或者用户的大平台。

**代表网站：**

腾讯课堂： https://ke.qq.com/

学而思 https://www.xueersi.com

## 6、1 对 1

让一个讲师在一定时间内对一个学员进行辅导，学生按照时间支付费用。这种模式收费容易， 现金流好，产品难度不大，市场空间大，但是人力资源的获取消耗却是巨大的，如果师资上控制不好，比如优秀的讲师留不住，或者整体成本太大，都会导致 1 对 1 模式难以发展。

**代表网站：**

VIPKID https://www.vipkid.com.cn/

学而思 https://www.xueersi.com

## 7、O2O 模式（Online To Offline 线上到线下）

就是通过免费内容或者运营，让线上平台获取用户和流量，将用户吸引到线下开课，或 者让学员到加盟的线下机构上课。这种模式形式简单，收益高，只要把控用户需求，吸引到用户，收费不成问题，而且符合传统的消费习惯。

**代表网站：**

启德教育 https://www.eic.org.cn/

## 8、freemium（免费增值）

Freemium最早由AVC的Fred Wilson在2006年提出，指的是用免费服务吸引用户，然后通过增值服务，将部分免费用户转化为收费用户，实现变现。Freemium模式中有"二八定律"的因素，即一小部分对价格不敏感的高端用户，愿意为一些额 外的功能付费，为服务提供者带来大部分收入。

**代表网站：**

中国大学慕课 http://www.icourse163.org

通过免费的名校课程和高校建立合作，吸引用户。提供考研专栏和学校云增值服务

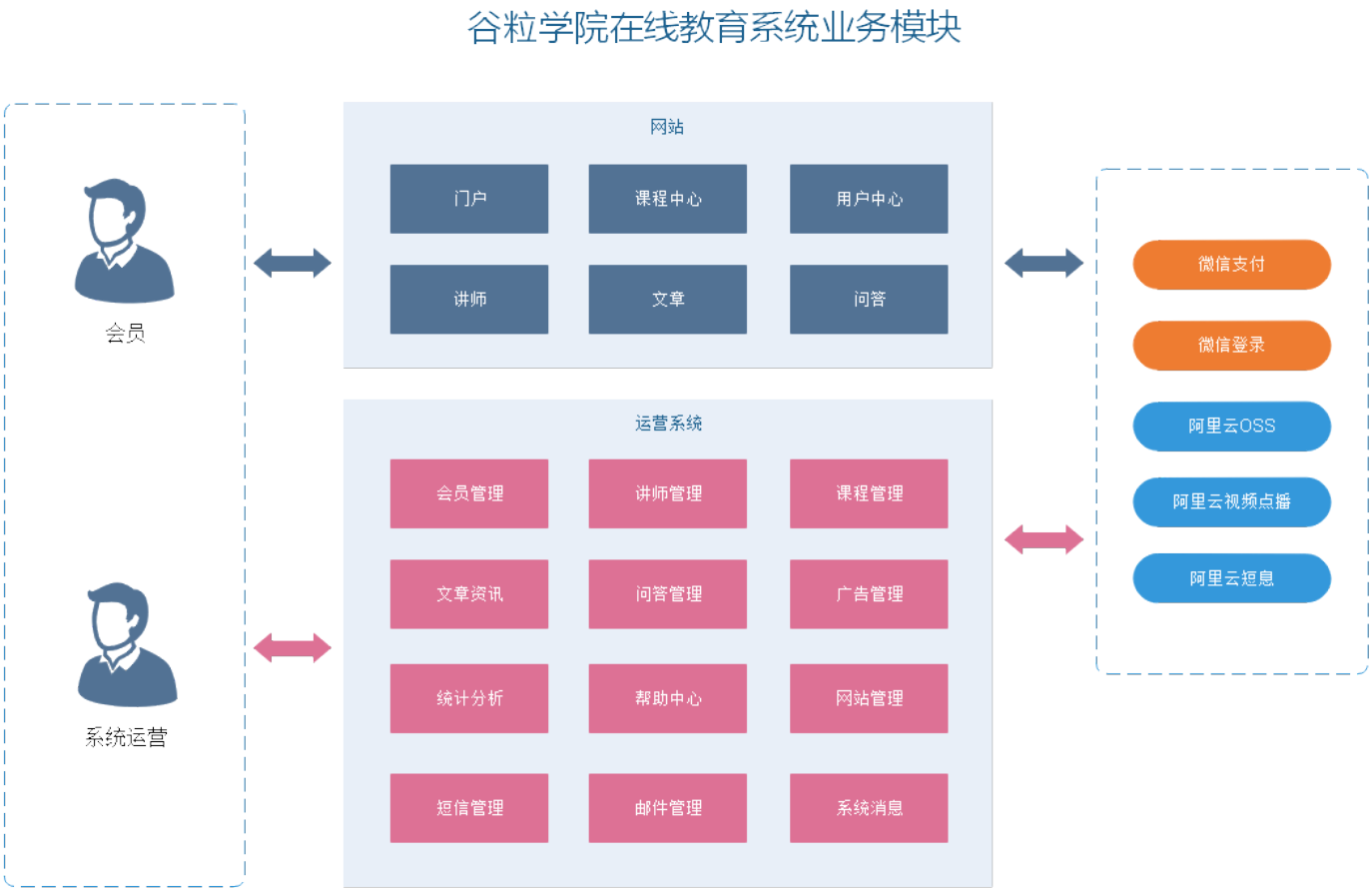
学堂在线 http://www.xuetangx.com/

课程免费，如果希望得到课程的认证证书则要缴纳相应的费用

# 一、功能简介

谷粒学院，是一个B2C模式的职业技能在线教育系统，分为前台用户系统和后台运营平台。

# 二、系统模块



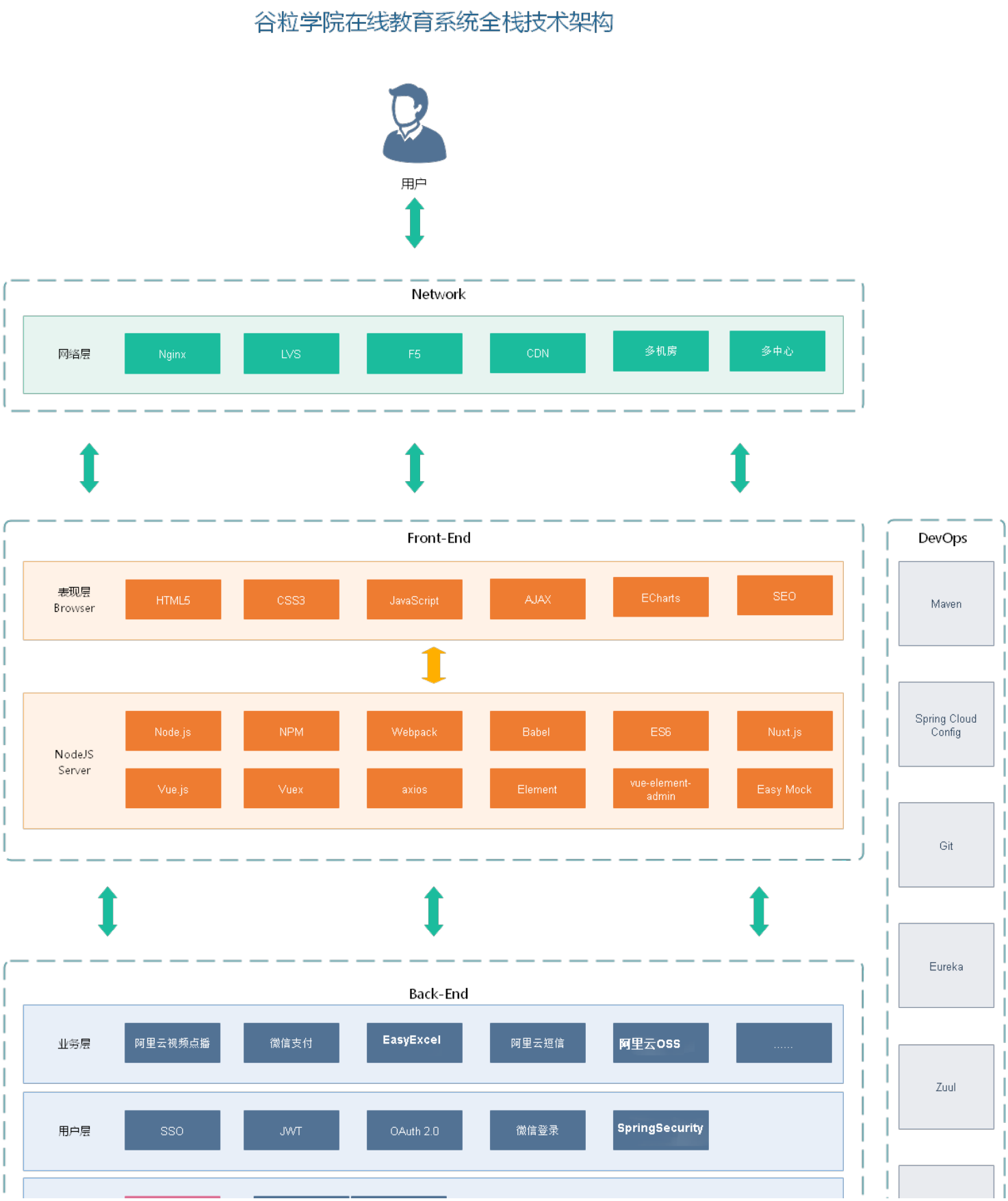

# 三、系统架构

**架构设计需要考虑的几个方面：**

- **性能：**主要考虑访问频率，每个用户每天的访问次数。项目初始阶段用户的访问量并不大，如果考虑做运营推广，可能会迎来服务器访问量骤增，因此要考虑**分布式部署，引入缓存**
- **可扩展性：**系统功能会随着用户量的增加以及多变的互联网用户需求不断地扩展，因此考虑到系统的可扩展性的要求需要**使用微服务架构，引入消息中间件**
- **高可用：**系统一旦宕机，将会带来不可挽回的损失，因此必须做负载均衡，甚至是异地多活这类复杂的方案。如果数据丢失，修复将会非常麻烦，只能靠人工逐条修复，这个很难接受，因此需要考虑存储高可靠。我们需要考虑多种异常情况：机器故障、机房故障，针对机器故障，我们需要设计

MySQL 同机房主备方案；针对机房故障，我们需要设计 MySQL 跨机房同步方案。

- **安全性：**系统的信息有一定的隐私性，例如用户的个人身份信息，不包含强隐私（例如玉照、情感）的信息，因此使用账号密码管理、数据库访问权限控制即可。
- **成本：**视频类网站的主要成本在于服务器成本、流量成本、存储成本、流媒体研发成本，中小型公司可以考虑使用云服务器和云服务。




服务层
配置中心
Nacos
git
服务中心
Nacos
Feign
GateWay
Hystrix
Spring Cloud Bus
Spring Cloud Sleuth
消息队列
RabbitMQ
开发层
开发框架
Spring Boot
Spring Cloud
MyBatis Plus
Spring MVC
Swagger
服务器
Tomcat
Nginx
容器
Docker
存储层
MySQL
Redis
阿里云OSS
系统日志
HikariCP
Hystrix
Docker
Zipkin
Jenkins

# 一、简介

官网：http://mp.baomidou.com/

参考教程：http://mp.baomidou.com/guide/

MyBatis-Plus（简称 MP）是一个 MyBatis 的增强工具，在 MyBatis 的基础上只做增强不做改变，为简化开发、提高效率而生。

# 二、特性

- **无侵入**：只做增强不做改变，引入它不会对现有工程产生影响，如丝般顺滑
- **损耗小**：启动即会自动注入基本 CURD，性能基本无损耗，直接面向对象操作
- **强大的 CRUD 操作**：内置通用 Mapper、通用 Service，仅仅通过少量配置即可实现单表大部分 CRUD 操作，更有强大的条件构造器，满足各类使用需求
- **支持 Lambda 形式调用**：通过 Lambda 表达式，方便的编写各类查询条件，无需再担心字段写错
- **支持多种数据库**：支持 MySQL、MariaDB、Oracle、DB2、H2、HSQL、SQLite、Postgre、SQLServer2005、SQLServer 等多种数据库
- **支持主键自动生成**：支持多达 4 种主键策略（内含分布式唯一 ID 生成器 - Sequence），可自由配置，完美解决主键问题
- **支持 XML 热加载**：Mapper 对应的 XML 支持热加载，对于简单的 CRUD 操作，甚至可以无 XML 启动
- **支持 ActiveRecord 模式**：支持 ActiveRecord 形式调用，实体类只需继承 Model 类即可进行强大的 CRUD 操作
- **支持自定义全局通用操作**：支持全局通用方法注入（ Write once, use anywhere ）
- **支持关键词自动转义**：支持数据库关键词（order、key……）自动转义，还可自定义关键词
- **内置代码生成器**：采用代码或者 Maven 插件可快速生成 Mapper 、 Model 、 Service 、 Controller 层代码，支持模板引擎，更有超多自定义配置等您来使用
- **内置分页插件**：基于 MyBatis 物理分页，开发者无需关心具体操作，配置好插件之后，写分页等同于普通 List 查询
- **内置性能分析插件**：可输出 Sql 语句以及其执行时间，建议开发测试时启用该功能，能快速揪出慢查询

- **内置全局拦截插件**：提供全表 delete 、 update 操作智能分析阻断，也可自定义拦截规则，预防误操作
- **内置 Sql 注入剥离器**：支持 Sql 注入剥离，有效预防 Sql 注入攻击

快速开始参考：http://mp.baomidou.com/guide/quick-start.html

测试项目： mybatis_plus

数据库：mybatis_plus

---

# 一、创建并初始化数据库

## 1、创建数据库：

mybatis_plus

## 2、创建 User 表

其表结构如下：

| id | name | age | email |
| --- | --- | --- | --- |
| 1 | Jone | 18 | test1@baomidou.com |
| 2 | Jack | 20 | test2@baomidou.com |
| 3 | Tom | 28 | test3@baomidou.com |
| 4 | Sandy | 21 | test4@baomidou.com |
| 5 | Billie | 24 | test5@baomidou.com |

其对应的数据库 Schema 脚本如下：

```
DROP TABLE IF EXISTS user;

CREATE TABLE user
(
    id BIGINT(20) NOT NULL COMMENT '主键ID',
    name VARCHAR(30) NULL DEFAULT NULL COMMENT '姓名',
    age INT(11) NULL DEFAULT NULL COMMENT '年龄',
    email VARCHAR(50) NULL DEFAULT NULL COMMENT '邮箱',
    PRIMARY KEY (id)
);
```

其对应的数据库 Data 脚本如下：

```sql
DELETE FROM user;

INSERT INTO user (id, name, age, email) VALUES
(1, 'Jone', 18, 'test1@baomidou.com'),
(2, 'Jack', 20, 'test2@baomidou.com'),
(3, 'Tom', 28, 'test3@baomidou.com'),
(4, 'Sandy', 21, 'test4@baomidou.com'),
(5, 'Billie', 24, 'test5@baomidou.com');
```

# 二、初始化工程

使用 Spring Initializr 快速初始化一个 Spring Boot 工程

Group：com.atguigu

Artifact：mybatis-plus

版本：2.2.1.RELEASE

# 三、添加依赖

## 1、引入依赖

spring-boot-starter、spring-boot-starter-test

添加：mybatis-plus-boot-starter、MySQL、lombok、

在项目中使用Lombok可以减少很多重复代码的书写。比如说getter/setter/toString等方法的编写

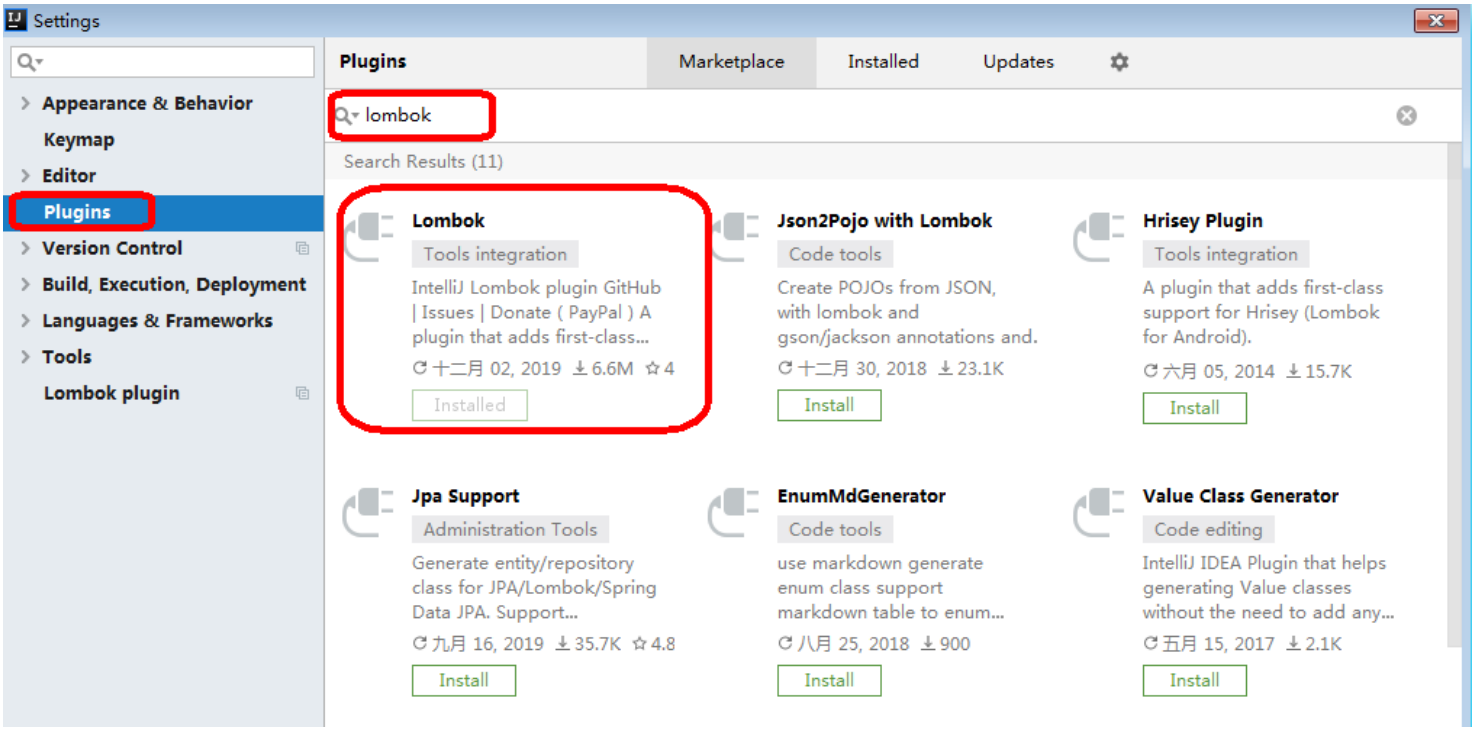
```xml
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter</artifactId>
    </dependency>

    <dependency>
```

```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
        <exclusions>
            <exclusion>
                <groupId>org.junit.vintage</groupId>
                <artifactId>junit-vintage-engine</artifactId>
            </exclusion>
        </exclusions>
    </dependency>

    <!--mybatis-plus-->
    <dependency>
        <groupId>com.baomidou</groupId>
        <artifactId>mybatis-plus-boot-starter</artifactId>
        <version>3.0.5</version>
    </dependency>

    <!--mysql-->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>

    <!--lombok用来简化实体类-->
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
    </dependency>
</dependencies>
```

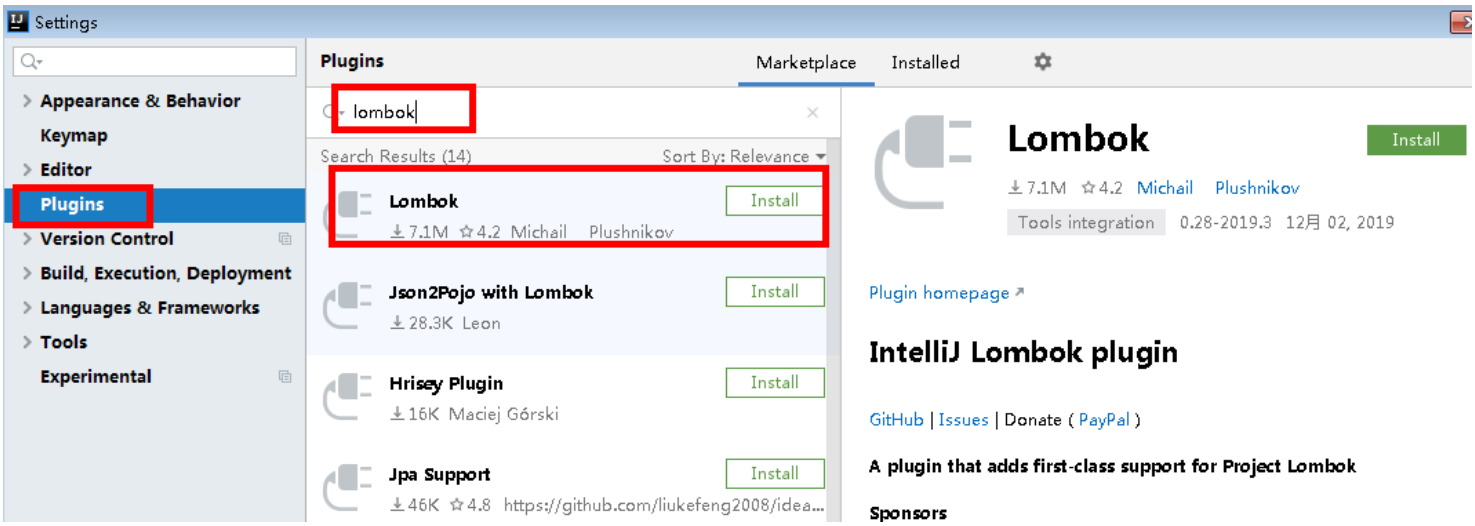
**注意：**引入 MyBatis-Plus 之后请不要再次引入 MyBatis 以及 MyBatis-Spring，以避免因版本差异导致的问题。

# 2、idea中安装lombok插件

（1）**idea2018**版本




（2）**idea2019**版本




# 四、配置

在 application.properties 配置文件中添加 MySQL 数据库的相关配置：

mysql5

```
#mysql数据库连接
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/mybatis_plus
spring.datasource.username=root
spring.datasource.password=123456
```

mysql8以上（spring boot 2.1）

**注意：**driver和url的变化

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/mybatis_plus?serverTimezone=GMT%2B8
spring.datasource.username=root
spring.datasource.password=123456
```

**注意：**

1、这里的 url 使用了 ?serverTimezone=GMT%2B8 后缀，因为Spring Boot 2.1 集成了 8.0版本的jdbc驱动，这个版本的 jdbc 驱动需要添加这个后缀，否则运行测试用例报告如下错误：

java.sql.SQLException: The server time zone value 'ÖÐ¹ú±ê×¼Ê±¼ä' is unrecognized or represents more

2、这里的 driver-class-name 使用了 com.mysql.cj.jdbc.Driver ，在 jdbc 8 中 建议使用这个驱动，之前的 com.mysql.jdbc.Driver 已经被废弃，否则运行测试用例的时候会有 WARN 信息

# 五、编写代码

## 1、主类

在 Spring Boot 启动类中添加 @MapperScan 注解，扫描 Mapper 文件夹

**注意：**扫描的包名根据实际情况修改

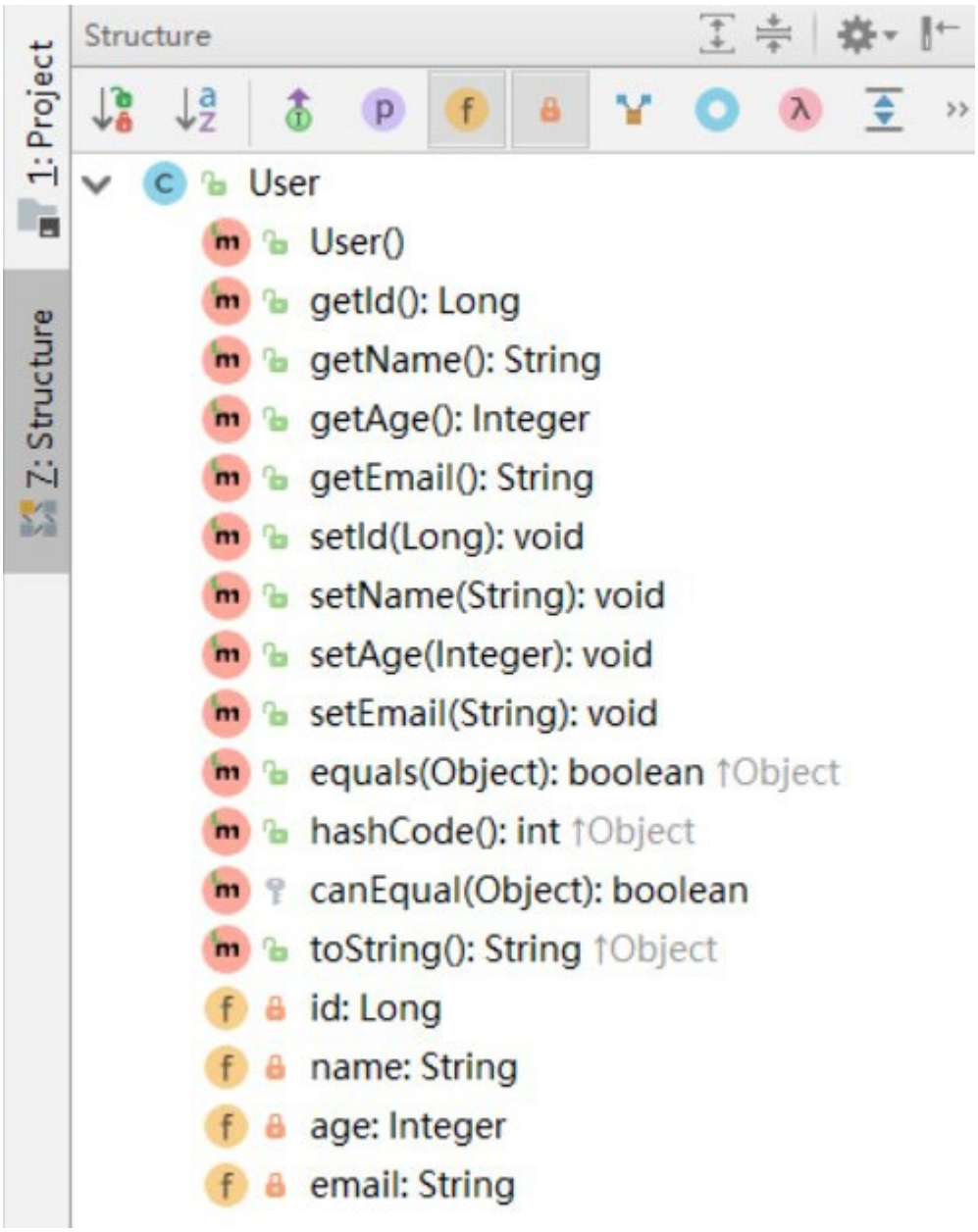
```
@SpringBootApplication
@MapperScan("com.atguigu.mybatisplus.mapper")
public class MybatisPlusApplication {
    ......
}
```

## 2、实体

创建包 entity 编写实体类 User.java（此处使用了 Lombok 简化代码）

```java
@Data
public class User {
    private Long id;
    private String name;
    private Integer age;
    private String email;
}
```

查看编译结果




Lombok使用参考：

https://blog.csdn.net/motui/article/details/79012846

## 3、mapper

创建包 mapper 编写Mapper 接口： UserMapper.java

```
public interface UserMapper extends BaseMapper<User> {

}
```

# 六、开始使用

添加测试类，进行功能测试：

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class MybatisPlusApplicationTests {

    @Autowired
    private UserMapper userMapper;

    @Test
    public void testSelectList() {
        System.out.println(("----- selectAll method test ------"));
        //UserMapper 中的 selectList() 方法的参数为 MP 内置的条件封装器 Wrapper
        //所以不填写就是无任何条件
        List<User> users = userMapper.selectList(null);
        users.forEach(System.out::println);
    }
}
```

注意：

IDEA在 userMapper 处报错，因为找不到注入的对象，因为类是动态创建的，但是程序可以正确的执行。

为了避免报错，可以在 dao 层 的接口上添加 @Repository 注

控制台输出：

```
User(id=1, name=Jone, age=18, email=test1@baomidou.com)
User(id=2, name=Jack, age=20, email=test2@baomidou.com)
User(id=3, name=Tom, age=28, email=test3@baomidou.com)
User(id=4, name=Sandy, age=21, email=test4@baomidou.com)
User(id=5, name=Billie, age=24, email=test5@baomidou.com)
```

通过以上几个简单的步骤，我们就实现了 User 表的 CRUD 功能，甚至连 XML 文件都不用编写！

# 七、配置日志

查看sql输出日志

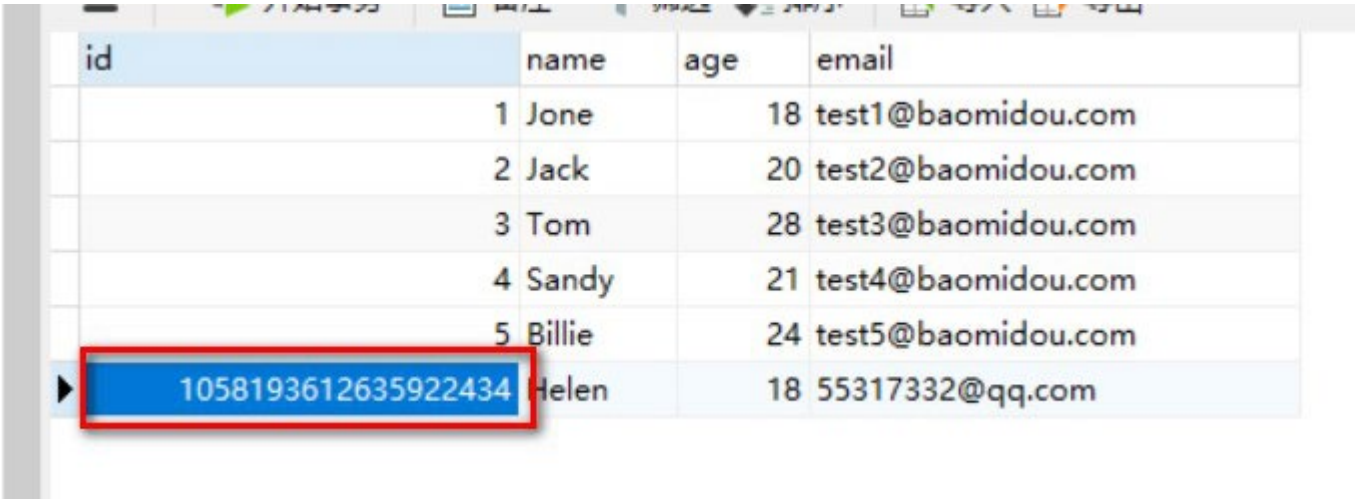
```
#mybatis日志
mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl
```

# 一、insert

## 1、插入操作

```java
@RunWith(SpringRunner.class)
@SpringBootTest
public class CRUDTests {

    @Autowired
    private UserMapper userMapper;

    @Test
    public void testInsert(){

        User user = new User();
        user.setName("Helen");
        user.setAge(18);
        user.setEmail("55317332@qq.com");

        int result = userMapper.insert(user);
        System.out.println(result); //影响的行数
        System.out.println(user); //id自动回填
    }
}
```

注意：数据库插入id值默认为：全局唯一id



| id | name | age | email |
|---|---|---|---|
| 1 | Jone | 18 | test1@baomidou.com |
| 2 | Jack | 20 | test2@baomidou.com |
| 3 | Tom | 28 | test3@baomidou.com |
| 4 | Sandy | 21 | test4@baomidou.com |
| 5 | Billie | 24 | test5@baomidou.com |
| 1058193612635922434 | Helen | 18 | 55317332@qq.com |

# 2、主键策略

## （1）ID_WORKER

MyBatis-Plus默认的主键策略是：ID_WORKER 全局唯一ID

参考资料：分布式系统唯一ID生成方案汇总：https://www.cnblogs.com/haoxinyue/p/5208136.html

## （2）自增策略

- 要想主键自增需要配置如下主键策略
  - 需要在创建数据表的时候设置主键自增
  - 实体字段中配置 @TableId(type = IdType.AUTO)

```
@TableId(type = IdType.AUTO)
private Long id;
```

要想影响所有实体的配置，可以设置全局主键配置

```
#全局设置主键生成策略
mybatis-plus.global-config.db-config.id-type=auto
```

其它主键策略：分析 IdType 源码可知

```
@Getter
public enum IdType {
    /**
     * 数据库ID自增
     */
    AUTO(0),
    /**
     * 该类型为未设置主键类型
     */
    NONE(1),
    /**
     * 用户输入ID
```

```
     * 该类型可以通过自己注册自动填充插件进行填充
     */
    INPUT(2),

    /* 以下3种类型、只有当插入对象ID 为空，才自动填充。 */
    /**
     * 全局唯一ID (idWorker)
     */
    ID_WORKER(3),
    /**
     * 全局唯一ID (UUID)
     */
    UUID(4),
    /**
     * 字符串全局唯一ID (idWorker 的字符串表示)
     */
    ID_WORKER_STR(5);

    private int key;

    IdType(int key) {
        this.key = key;
    }
}
```

# 二、update

## 1、根据Id更新操作

注意：update时生成的sql自动是动态sql：UPDATE user SET age=? WHERE id=?

```
    @Test
    public void testUpdateById(){

        User user = new User();
        user.setId(1L);
        user.setAge(28);

        int result = userMapper.updateById(user);
```

```
        System.out.println(result);

    }
```

# 2、自动填充

项目中经常会遇到一些数据，每次都使用相同的方式填充，例如记录的创建时间，更新时间等。

我们可以使用MyBatis Plus的自动填充功能，完成这些字段的赋值工作：

（1）数据库表中添加自动填充字段

在User表中添加datetime类型的新的字段 create_time、update_time

### （2）实体上添加注解

```
@Data
public class User {
    ......

    @TableField(fill = FieldFill.INSERT)
    private Date createTime;

    //@TableField(fill = FieldFill.UPDATE)
    @TableField(fill = FieldFill.INSERT_UPDATE)
    private Date updateTime;
}
```

### （3）实现元对象处理器接口

注意：不要忘记添加 **@Component** 注解

```
package com.atguigu.mybatisplus.handler;
import com.baomidou.mybatisplus.core.handlers.MetaObjectHandler;
import org.apache.ibatis.reflection.MetaObject;
import org.slf4j.Logger;
```

```java
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Component;
import java.util.Date;

@Component
public class MyMetaObjectHandler implements MetaObjectHandler {

    private static final Logger LOGGER =
LoggerFactory.getLogger(MyMetaObjectHandler.class);

    @Override
    public void insertFill(MetaObject metaObject) {
        LOGGER.info("start insert fill ....");
        this.setFieldValByName("createTime", new Date(), metaObject);
        this.setFieldValByName("updateTime", new Date(), metaObject);
    }

    @Override
    public void updateFill(MetaObject metaObject) {
        LOGGER.info("start update fill ....");
        this.setFieldValByName("updateTime", new Date(), metaObject);
    }
}
```

**（4）测试**

# 3、乐观锁

**主要适用场景：**当要更新一条记录的时候，希望这条记录没有被别人更新，也就是说实现线程安全的数据更新

**乐观锁实现方式：**

- 取出记录时，获取当前version
- 更新时，带上这个version
- 执行更新时， set version = newVersion where version = oldVersion
- 如果version不对，就更新失败

### （1）数据库中添加version字段

```
ALTER TABLE `user` ADD COLUMN `version` INT
```

| | | | | | |
|---|---|---|---|---|---|
| version | int | 11 | 0 | ☐ | |

### （2）实体类添加version字段

并添加 @Version 注解

```
@Version
@TableField(fill = FieldFill.INSERT)
private Integer version;
```

### （3）元对象处理器接口添加version的insert默认值

```
@Override
public void insertFill(MetaObject metaObject) {
    ......
    this.setFieldValByName("version", 1, metaObject);
}
```

**特别说明:**

- 支持的数据类型只有 int,Integer,long,Long,Date,Timestamp,LocalDateTime
- 整数类型下 newVersion = oldVersion + 1
- newVersion 会回写到 entity 中
- 仅支持 updateById(id) 与 update(entity, wrapper) 方法
- 在 update(entity, wrapper) 方法下, wrapper 不能复用!!!

### （4）在 MybatisPlusConfig 中注册 Bean

创建配置类

```
package com.atguigu.mybatisplus.config;
```

```
import com.baomidou.mybatisplus.extension.plugins.PaginationInterceptor;
import org.mybatis.spring.annotation.MapperScan;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.transaction.annotation.EnableTransactionManagement;

@EnableTransactionManagement
@Configuration
@MapperScan("com.atguigu.mybatis_plus.mapper")
public class MybatisPlusConfig {

    /**
     * 乐观锁插件
     */
    @Bean
    public OptimisticLockerInterceptor optimisticLockerInterceptor() {
        return new OptimisticLockerInterceptor();
    }
}
```

### （5）测试乐观锁可以修改成功

测试后分析打印的sql语句，将version的数值进行了加1操作

```
/**
 * 测试 乐观锁插件
 */
@Test
public void testOptimisticLocker() {

    //查询
    User user = userMapper.selectById(1L);
    //修改数据
    user.setName("Helen Yao");
    user.setEmail("helen@qq.com");
    //执行更新
    userMapper.updateById(user);
}
```

### （5）测试乐观锁修改失败

```
/**
 * 测试乐观锁插件 失败
 */
@Test
public void testOptimisticLockerFail() {

    //查询
    User user = userMapper.selectById(1L);
    //修改数据
    user.setName("Helen Yao1");
    user.setEmail("helen@qq.com1");

    //模拟取出数据后，数据库中version实际数据比取出的值大，即已被其它线程修改并更新了version
    user.setVersion(user.getVersion() - 1);

    //执行更新
    userMapper.updateById(user);
}
```

# 三、select

## 1、根据id查询记录

```
@Test
public void testSelectById(){

    User user = userMapper.selectById(1L);
    System.out.println(user);
}
```

## 2、通过多个id批量查询

完成了动态sql的foreach的功能

```
@Test
public void testSelectBatchIds(){

    List<User> users = userMapper.selectBatchIds(Arrays.asList(1, 2, 3));
    users.forEach(System.out::println);
}
```

# 3、简单的条件查询

通过map封装查询条件

```
@Test
public void testSelectByMap(){

    HashMap<String, Object> map = new HashMap<>();
    map.put("name", "Helen");
    map.put("age", 18);
    List<User> users = userMapper.selectByMap(map);

    users.forEach(System.out::println);
}
```

注意：map中的key对应的是数据库中的列名。例如数据库user_id，实体类是userId，这时map的key需要填写user_id

# 4、分页

MyBatis Plus自带分页插件，只要简单的配置即可实现分页功能

**（1）创建配置类**

此时可以删除主类中的 *@MapperScan* 扫描注解

```
/**
```

```
 * 分页插件
 */
@Bean
public PaginationInterceptor paginationInterceptor() {
    return new PaginationInterceptor();
}
```

### （2）测试selectPage分页

测试：最终通过page对象获取相关数据

```
@Test
public void testSelectPage() {

    Page<User> page = new Page<>(1,5);
    userMapper.selectPage(page, null);

    page.getRecords().forEach(System.out::println);
    System.out.println(page.getCurrent());
    System.out.println(page.getPages());
    System.out.println(page.getSize());
    System.out.println(page.getTotal());
    System.out.println(page.hasNext());
    System.out.println(page.hasPrevious());
}
```

控制台sql语句打印：SELECT id,name,age,email,create_time,update_time FROM user LIMIT 0,5

### （3）测试selectMapsPage分页：结果集是Map

```
@Test
public void testSelectMapsPage() {

    Page<User> page = new Page<>(1, 5);

    IPage<Map<String, Object>> mapIPage = userMapper.selectMapsPage(page, null);

    //注意：此行必须使用 mapIPage 获取记录列表，否则会有数据类型转换错误
```

```
    mapIPage.getRecords().forEach(System.out::println);
    System.out.println(page.getCurrent());
    System.out.println(page.getPages());
    System.out.println(page.getSize());
    System.out.println(page.getTotal());
    System.out.println(page.hasNext());
    System.out.println(page.hasPrevious());
}
```

# 四、delete

## 1、根据id删除记录

```
@Test
public void testDeleteById(){

    int result = userMapper.deleteById(8L);
    System.out.println(result);
}
```

## 2、批量删除

```
    @Test
    public void testDeleteBatchIds() {

        int result = userMapper.deleteBatchIds(Arrays.asList(8, 9, 10));
        System.out.println(result);
    }
```

## 3、简单的条件查询删除

```java
@Test
public void testDeleteByMap() {

    HashMap<String, Object> map = new HashMap<>();
    map.put("name", "Helen");
    map.put("age", 18);

    int result = userMapper.deleteByMap(map);
    System.out.println(result);
}
```

# 4、逻辑删除

- 物理删除：真实删除，将对应数据从数据库中删除，之后查询不到此条被删除数据
- 逻辑删除：假删除，将对应数据中代表是否被删除字段状态修改为“被删除状态”，之后在数据库中仍旧能看到此条数据记录

### （1）数据库中添加 deleted字段

```sql
ALTER TABLE `user` ADD COLUMN `deleted` boolean
```

| | | | | |
|---|---|---|---|---|
| deleted | tinyint | 1 | 0 | ☐ |

### （2）实体类添加deleted 字段

并加上 @TableLogic 注解 和 @TableField(fill = FieldFill.INSERT) 注解

```java
@TableLogic
@TableField(fill = FieldFill.INSERT)
private Integer deleted;
```

### （3）元对象处理器接口添加deleted的insert默认值

```
@Override
public void insertFill(MetaObject metaObject) {
    ......
    this.setFieldValByName("deleted", 0, metaObject);
}
```

（4）**application.properties** 加入配置

此为默认值，如果你的默认值和mp默认的一样,该配置可无

```
mybatis-plus.global-config.db-config.logic-delete-value=1
mybatis-plus.global-config.db-config.logic-not-delete-value=0
```

**（5）在 MybatisPlusConfig 中注册 Bean**

```
@Bean
public ISqlInjector sqlInjector() {
    return new LogicSqlInjector();
}
```

（6）测试逻辑删除

- 测试后发现，数据并没有被删除，deleted字段的值由0变成了1
- 测试后分析打印的sql语句，是一条update
- 注意：被删除数据的deleted 字段的值必须是 0，才能被选取出来执行逻辑删除的操作

```
/**
 * 测试 逻辑删除
 */
@Test
public void testLogicDelete() {

    int result = userMapper.deleteById(1L);
    System.out.println(result);
}
```

（7）测试逻辑删除后的查询

MyBatis Plus中查询操作也会自动添加逻辑删除字段的判断

```
/**
 * 测试 逻辑删除后的查询：
 * 不包括被逻辑删除的记录
 */
@Test
public void testLogicDeleteSelect() {
    User user = new User();
    List<User> users = userMapper.selectList(null);
    users.forEach(System.out::println);
}
```

测试后分析打印的sql语句，包含 WHERE deleted=0

SELECT id,name,age,email,create_time,update_time,deleted FROM user WHERE deleted=0

# 五、性能分析

性能分析拦截器，用于输出每条 SQL 语句及其执行时间

SQL 性能执行分析,开发环境使用，超过指定时间，停止运行。有助于发现问题

## 1、配置插件

### （1）参数说明

参数：maxTime： SQL 执行最大时长，超过自动停止运行，有助于发现问题。

参数：format： SQL是否格式化，默认false。

### （2）在 MybatisPlusConfig 中配置

```
/**
 * SQL 执行性能分析插件
 * 开发环境使用，线上不推荐。 maxTime 指的是 sql 最大执行时长
 */
@Bean
@Profile({"dev","test"})// 设置 dev test 环境开启
public PerformanceInterceptor performanceInterceptor() {
```

```
    PerformanceInterceptor performanceInterceptor = new PerformanceInterceptor();
    performanceInterceptor.setMaxTime(100);//ms，超过此处设置的ms则sql不执行
    performanceInterceptor.setFormat(true);
    return performanceInterceptor;
}
```

### （3）Spring Boot 中设置dev环境

```
#环境设置：dev、test、prod
spring.profiles.active=dev
```

可以针对各环境新建不同的配置文件application-dev.properties、application-test.properties、application-prod.properties

也可以自定义环境名称：如test1、test2

## 2、测试

### （1）常规测试

```
/**
 * 测试 性能分析插件
 */
@Test
public void testPerformance() {
    User user = new User();
    user.setName("我是Helen");
    user.setEmail("helen@sina.com");
    user.setAge(18);
    userMapper.insert(user);
}
```

输出：

```
 Time: 9 ms - ID: com.atguigu.mybatis_plus.mapper.UserMapper.insert
Execute SQL:
    INSERT
    INTO
        user
        ( id, name, age, email, create_time, update_time, deleted, version )
    VALUES
        ( 1072851634398244865, '我是Helen', 18, 'helen@sina.com', '2018-12-12 21:52:16.918', '2018-12-12 21:52:16.918', 0, 1 )
```

### （2）将maxTime 改小之后再次进行测试

```
performanceInterceptor.setMaxTime(5);//ms，超过此处设置的ms不执行
```

如果执行时间过长，则抛出异常：The SQL execution time is too large,

输出：

```
org.mybatis.spring.MyBatisSystemException: nested exception is org.apache.ibatis.exceptions.PersistenceException:
### Error updating database.  Cause: com.baomidou.mybatisplus.core.exceptions.MybatisPlusException:  The SQL execution time is too
 large, please optimize !
```

# 六、其它

如果想进行复杂条件查询，那么需要使用条件构造器 Wapper，涉及到如下方法

**1、delete**

**2、selectOne**

**3、selectCount**

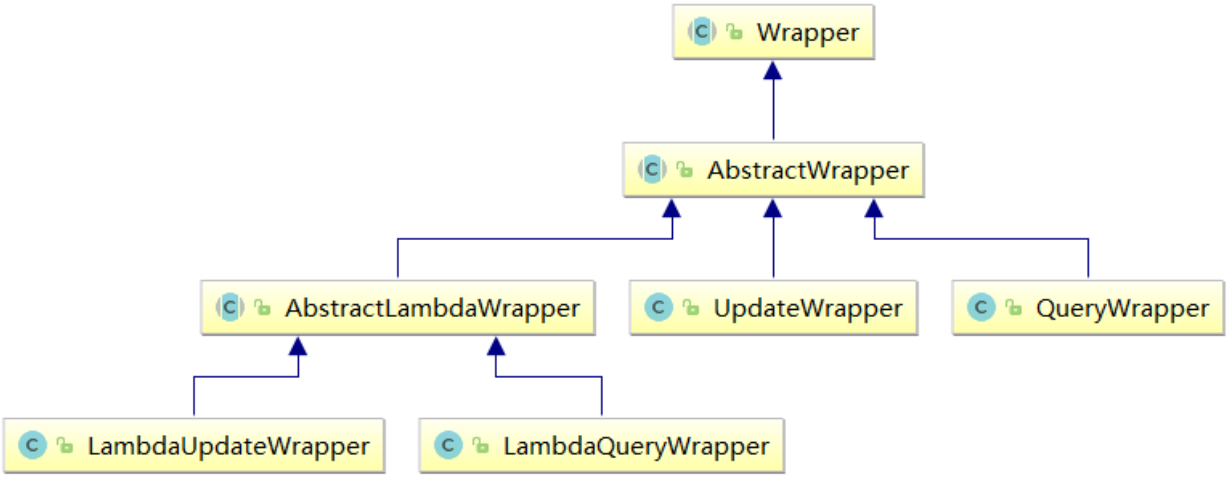
**4、selectList**

**5、selectMaps**

**6、selectObjs**

**7、update**

# 一、wapper介绍




Wrapper： 条件构造抽象类，最顶端父类

AbstractWrapper： 用于查询条件封装，生成 sql 的 where 条件

QueryWrapper： Entity 对象封装操作类，不是用lambda语法

UpdateWrapper： Update 条件封装，用于Entity对象更新操作

AbstractLambdaWrapper： Lambda 语法使用 Wrapper统一处理解析 lambda 获取 column。

LambdaQueryWrapper：看名称也能明白就是用于Lambda语法使用的查询Wrapper

LambdaUpdateWrapper： Lambda 更新封装Wrapper

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class QueryWrapperTests {

    @Autowired
    private UserMapper userMapper;
}
```

# 二、AbstractWrapper

**注意：**以下条件构造器的方法入参中的 column 均表示数据库字段

# 1、ge、gt、le、lt、isNull、isNotNull

```
@Test
public void testDelete() {
    QueryWrapper<User> queryWrapper = new QueryWrapper<>();
    queryWrapper
        .isNull("name")
        .ge("age", 12)
        .isNotNull("email");
    int result = userMapper.delete(queryWrapper);
    System.out.println("delete return count = " + result);
}
```

SQL：UPDATE user SET deleted=1 WHERE deleted=0 AND name IS NULL AND age >= ? AND email IS NOT NULL

# 2、eq、ne

**注意：**seletOne返回的是一条实体记录，当出现多条时会报错

```
@Test
public void testSelectOne() {
    QueryWrapper<User> queryWrapper = new QueryWrapper<>();
    queryWrapper.eq("name", "Tom");
    User user = userMapper.selectOne(queryWrapper);
    System.out.println(user);
}
```

SELECT id,name,age,email,create_time,update_time,deleted,version FROM user WHERE deleted=0 AND name = ?

# 3、between、notBetween

包含大小边界

```
@Test
public void testSelectCount() {
    QueryWrapper<User> queryWrapper = new QueryWrapper<>();
    queryWrapper.between("age", 20, 30);
    Integer count = userMapper.selectCount(queryWrapper);
    System.out.println(count);
}
```

SELECT COUNT(1) FROM user WHERE deleted=0 AND age BETWEEN ? AND ?

# 4、allEq

```
@Test
public void testSelectList() {
    QueryWrapper<User> queryWrapper = new QueryWrapper<>();
    Map<String, Object> map = new HashMap<>();
    map.put("id", 2);
    map.put("name", "Jack");
    map.put("age", 20);
    queryWrapper.allEq(map);
    List<User> users = userMapper.selectList(queryWrapper);
    users.forEach(System.out::println);
}
```

SELECT id,name,age,email,create_time,update_time,deleted,version

FROM user WHERE deleted=0 AND name = ? AND id = ? AND age = ?

# 5、like、notLike、likeLeft、likeRight

selectMaps返回Map集合列表

```
@Test
public void testSelectMaps() {
    QueryWrapper<User> queryWrapper = new QueryWrapper<>();
    queryWrapper
```

```
        .notLike("name", "e")
        .likeRight("email", "t");
    List<Map<String, Object>> maps = userMapper.selectMaps(queryWrapper);//返回值是Map列表
    maps.forEach(System.out::println);
}
```

SELECT id,name,age,email,create_time,update_time,deleted,version

FROM user WHERE deleted=0 AND name NOT LIKE ? AND email LIKE ?

# 6、in、notIn、inSql、notinSql、exists、notExists

in、notIn：

- notIn("age",{1,2,3})--->age not in (1,2,3)
- notIn("age", 1, 2, 3)--->age not in (1,2,3)

inSql、notinSql：可以实现子查询

- 例: inSql("age", "1,2,3,4,5,6")--->age in (1,2,3,4,5,6)
- 例: inSql("id", "select id from table where id < 3")--->id in (select id from table where id < 3)

```
@Test
public void testSelectObjs() {
    QueryWrapper<User> queryWrapper = new QueryWrapper<>();
    //queryWrapper.in("id", 1, 2, 3);
    queryWrapper.inSql("id", "select id from user where id < 3");
    List<Object> objects = userMapper.selectObjs(queryWrapper);//返回值是Object列表
    objects.forEach(System.out::println);
}
```

SELECT id,name,age,email,create_time,update_time,deleted,version

FROM user WHERE deleted=0 AND id IN (select id from user where id < 3)

# 7、or、and

**注意：**这里使用的是 UpdateWrapper

不调用or则默认为使用 and 连

```
@Test
public void testUpdate1() {
    //修改值
    User user = new User();
    user.setAge(99);
    user.setName("Andy");
    //修改条件
    UpdateWrapper<User> userUpdateWrapper = new UpdateWrapper<>();
    userUpdateWrapper
        .like("name", "h")
        .or()
        .between("age", 20, 30);
    int result = userMapper.update(user, userUpdateWrapper);
    System.out.println(result);
}
```

UPDATE user SET name=?, age=?, update_time=? WHERE deleted=0 AND name LIKE ? OR age BETWEEN ? AND ?

# 8、嵌套or、嵌套and

这里使用了lambda表达式，or中的表达式最后翻译成sql时会被加上圆括号

```
@Test
public void testUpdate2() {
    //修改值
    User user = new User();
    user.setAge(99);
    user.setName("Andy");
    //修改条件
    UpdateWrapper<User> userUpdateWrapper = new UpdateWrapper<>();
    userUpdateWrapper
        .like("name", "h")
        .or(i -> i.eq("name", "李白").ne("age", 20));
```

```
    int result = userMapper.update(user, userUpdateWrapper);
    System.out.println(result);
}
```

UPDATE user SET name=?, age=?, update_time=?

WHERE deleted=0 AND name LIKE ?

OR ( name = ? AND age <> ? )

# 9、orderBy、orderByDesc、orderByAsc

```
@Test
public void testSelectListOrderBy() {
    QueryWrapper<User> queryWrapper = new QueryWrapper<>();
    queryWrapper.orderByDesc("id");
    List<User> users = userMapper.selectList(queryWrapper);
    users.forEach(System.out::println);
}
```

SELECT id,name,age,email,create_time,update_time,deleted,version

FROM user WHERE deleted=0 ORDER BY id DESC

# 10、last

直接拼接到 sql 的最后

**注意：**只能调用一次,多次调用以最后一次为准 有sql注入的风险,请谨慎使用

```
@Test
public void testSelectListLast() {
    QueryWrapper<User> queryWrapper = new QueryWrapper<>();
    queryWrapper.last("limit 1");
    List<User> users = userMapper.selectList(queryWrapper);
    users.forEach(System.out::println);
}
```

SELECT id,name,age,email,create_time,update_time,deleted,version

FROM user WHERE deleted=0 limit 1

## 11、指定要查询的列

```
@Test
public void testSelectListColumn() {
    QueryWrapper<User> queryWrapper = new QueryWrapper<>();
    queryWrapper.select("id", "name", "age");
    List<User> users = userMapper.selectList(queryWrapper);
    users.forEach(System.out::println);
}
```

SELECT id,name,age FROM user WHERE deleted=0

## 12、set、setSql

最终的sql会合并 user.setAge()，以及 userUpdateWrapper.set() 和 setSql() 中 的字段

```
@Test
public void testUpdateSet() {
    //修改值
    User user = new User();
    user.setAge(99);
    //修改条件
    UpdateWrapper<User> userUpdateWrapper = new UpdateWrapper<>();
    userUpdateWrapper
        .like("name", "h")
        .set("name", "老李头")//除了可以查询还可以使用set设置修改的字段
        .setSql(" email = '123@qq.com'");//可以有子查询
    int result = userMapper.update(user, userUpdateWrapper);
}
```

UPDATE user SET age=?, update_time=?, name=?, email = '123@qq.com' WHERE deleted=0 AND name LIKE ?

# 一、数据库设计

## 1、数据库

guli_edu

## 2、数据表

```
guli_edu.sql
```

# 二、数据库设计规约

以下规约只针对本模块，更全面的文档参考《阿里巴巴Java开发手册》：五、MySQL数据库

1、库名与应用名称尽量一致

2、表名、字段名必须使用小写字母或数字，禁止出现数字开头，

3、表名不使用复数名词

4、表的命名最好是加上“业务名称_表的作用”。如，edu_teacher

5、表必备三字段：id, gmt_create, gmt_modified

说明：

其中 id 必为主键，类型为 bigint unsigned、单表时自增、步长为 1。

（如果使用分库分表集群部署，则id类型为verchar，非自增，业务中使用分布式id生成器）

gmt_create, gmt_modified 的类型均为 datetime 类型，前者现在时表示主动创建，后者过去分词表示被动更新。

6、单表行数超过 500 万行或者单表容量超过 2GB，才推荐进行分库分表。 说明：如果预计三年后的数据量根本达不到这个级别，请不要在创建表时就分库分表。

7、表达是与否概念的字段，必须使用 is_xxx 的方式命名，数据类型是 unsigned tinyint （1 表示是，0 表示否）。

说明：任何字段如果为非负数，必须是 unsigned。

注意：POJO 类中的任何布尔类型的变量，都不要加 is 前缀。数据库表示是与否的值，使用 tinyint 类型，坚持 is_xxx 的 命名方式是为了明确其取值含义与取值范围。

正例：表达逻辑删除的字段名 is_deleted，1 表示删除，0 表示未删除。

8、小数类型为 decimal，禁止使用 float 和 double。 说明：float 和 double 在存储的时候，存在精度损失的问题，很可能在值的比较时，得到不 正确的结果。如果存储的数据范围超过 decimal 的范围，建议将数据拆成整数和小数分开存储。

9、如果存储的字符串长度几乎相等，使用 char 定长字符串类型。

10、varchar 是可变长字符串，不预先分配存储空间，长度不要超过 5000，如果存储长度大于此值，定义字段类型为 text，独立出来一张表，用主键来对应，避免影响其它字段索 引效率。

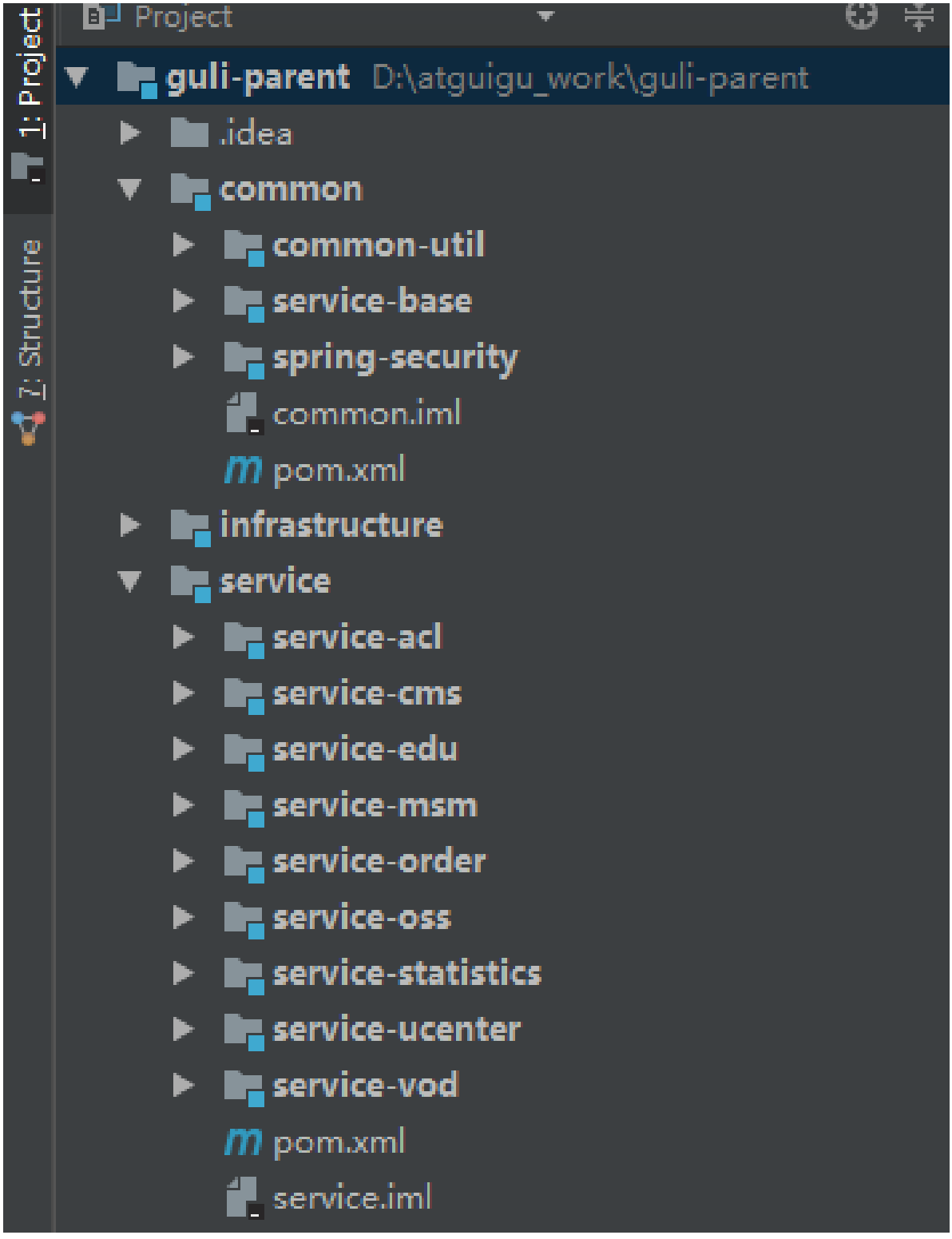
11、唯一索引名为 uk_字段名；普通索引名则为 idx_字段名。

说明：uk_ 即 unique key；idx_ 即 index 的简称

12、不得使用外键与级联，一切外键概念必须在应用层解决。外键与级联更新适用于单机低并发，不适合分布式、高并发集群；级联更新是强阻塞，存在数据库更新风暴的风险；外键影响数据库的插入速度。

# 一、工程结构介绍

## 1、工程结构




## 2、模块说明

**guli-parent**：在线教学根目录（父工程），管理四个子模块：

**canal-client**：**canal**数据库表同步模块（统计同步数据）

**common**：公共模块父节点

common-util：工具类模块，所有模块都可以依赖于它

service-base：service服务的base包，包含service服务的公共配置类，所有service模块依赖于它

spring-security：认证与授权模块，需要认证授权的service服务依赖于它

**infrastructure**：基础服务模块父节点

api-gateway：api网关服务

**service**：**api**接口服务父节点

service-acl：用户权限管理api接口服务（用户管理、角色管理和权限管理等）

service-cms：cms api接口服务

service-edu：教学相关api接口服务

service-msm：短信api接口服务

service-order：订单相关api接口服务

service-oss：阿里云oss api接口服务

service-statistics：统计报表api接口服务

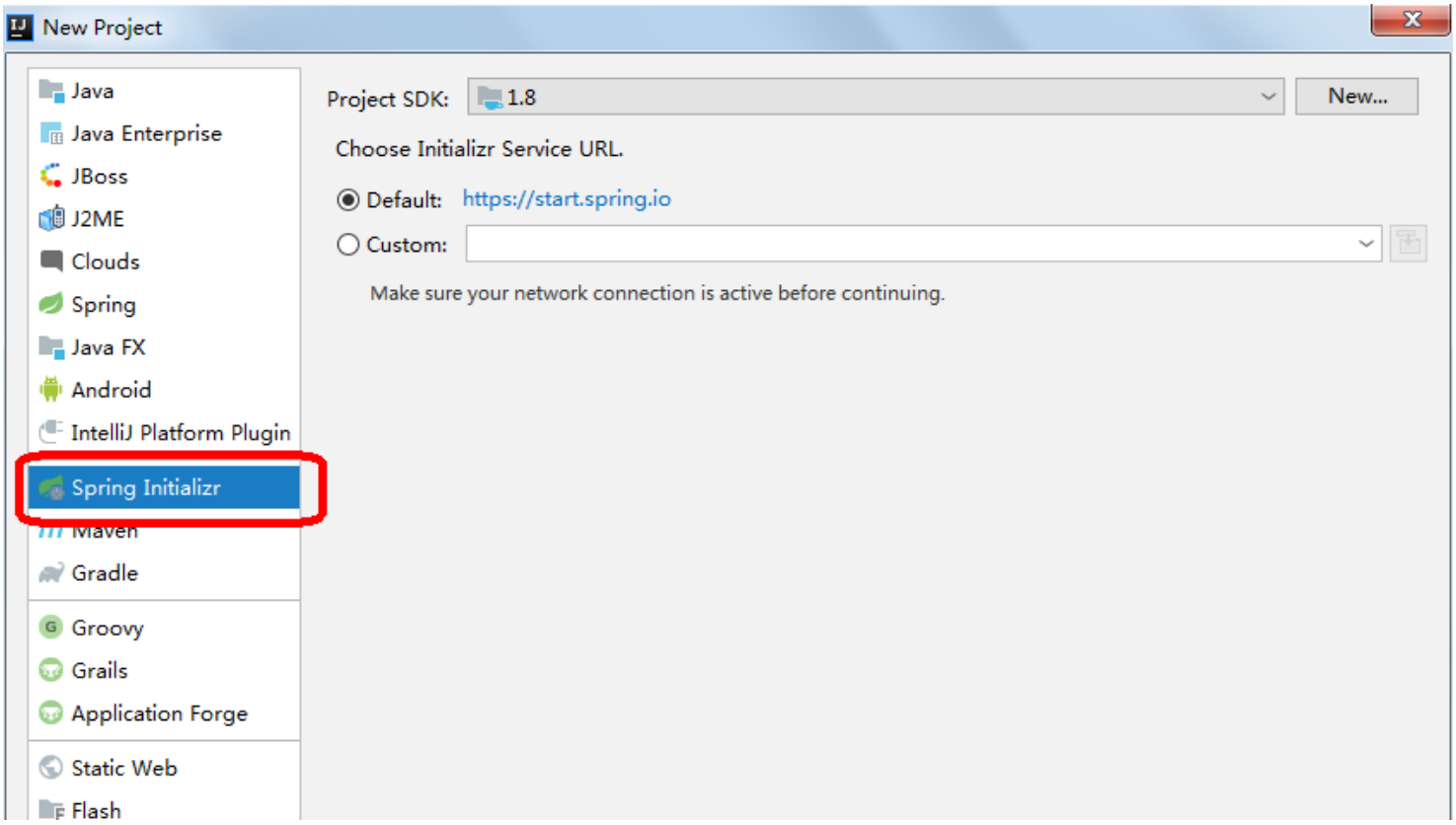
service-ucenter：会员api接口服务

service-vod：视频点播api接口服务

# 二、创建父工程

## 1、创建**sprigboot**工程**guli-parent**

在idea开发工具中，使用 Spring Initializr 快速初始化一个 Spring Boot 模块，版本使用：2.2.1.RELEASE

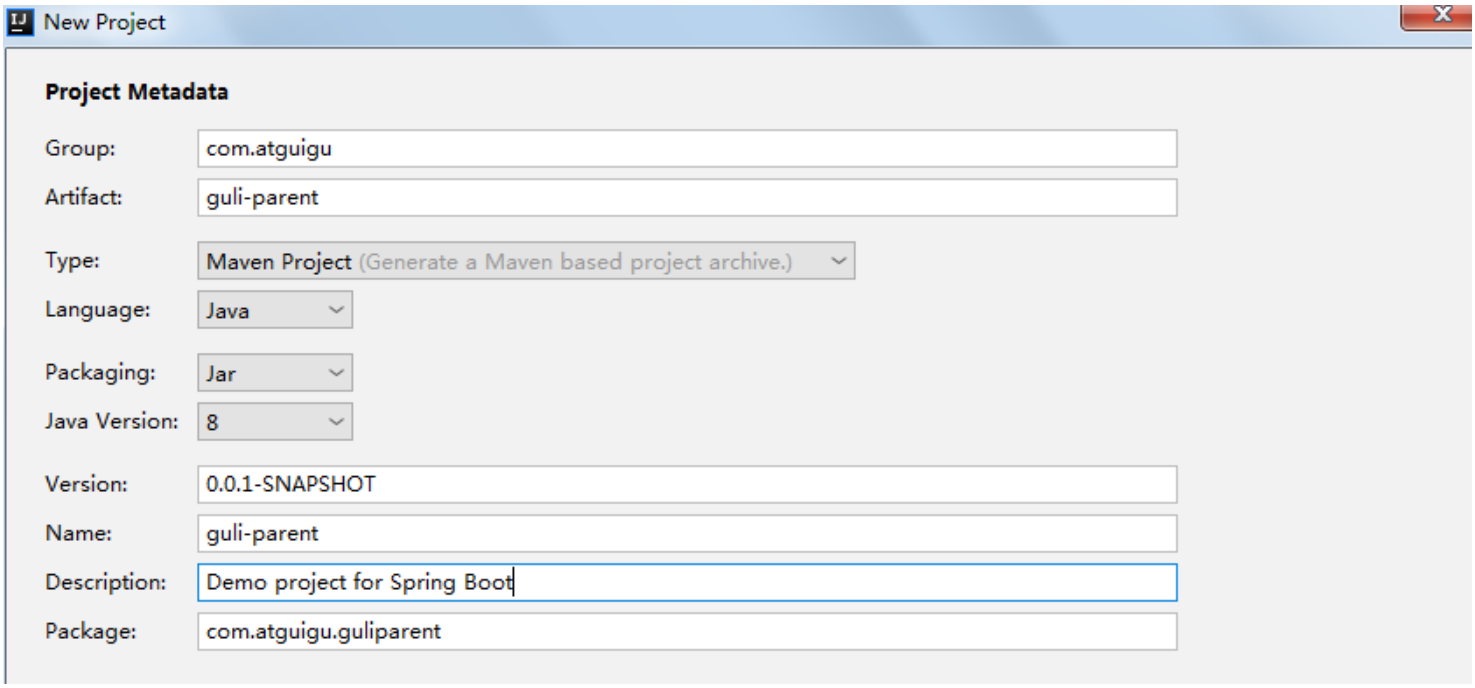



配置：

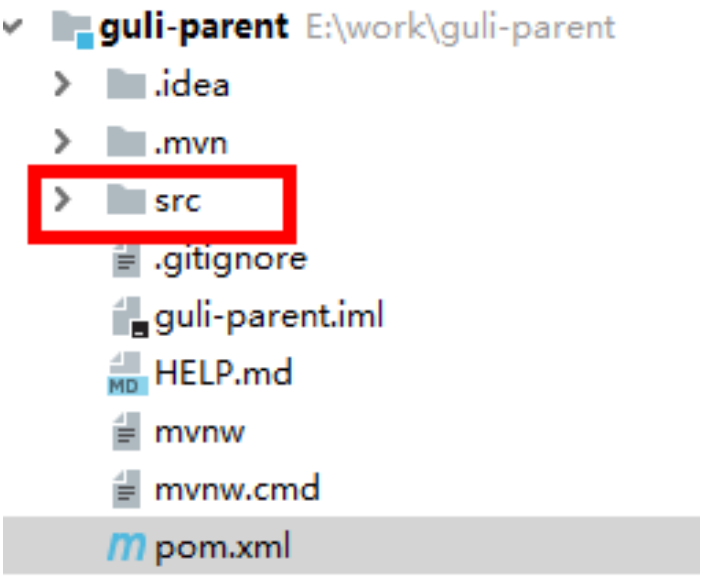
groupId：com.atguigu

artifactId：guli-parent

一直下一步到完成



## 2、删除 src 目录




## 3、配置 pom.xml

修改版本为： 2.2.1.RELEASE

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.2.1.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>
```

<artifactId> 节点后面添加 pom类型

```
<artifactId>guli-parent</artifactId>
<packaging>pom</packaging>
```

## 4、在pom.xml中添加依赖的版本

**删除pom.xml中的**<dependencies>内容

```xml
<!-- 以下内容删除 -->
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

**添加 <properties>确定依赖的版本**

```xml
<properties>
    <java.version>1.8</java.version>
    <guli.version>0.0.1-SNAPSHOT</guli.version>
    <mybatis-plus.version>3.0.5</mybatis-plus.version>
    <velocity.version>2.0</velocity.version>
    <swagger.version>2.7.0</swagger.version>
    <aliyun.oss.version>2.8.3</aliyun.oss.version>
    <jodatime.version>2.10.1</jodatime.version>
    <poi.version>3.17</poi.version>
    <commons-fileupload.version>1.3.1</commons-fileupload.version>
    <commons-io.version>2.6</commons-io.version>
    <httpclient.version>4.5.1</httpclient.version>
    <jwt.version>0.7.0</jwt.version>
    <aliyun-java-sdk-core.version>4.3.3</aliyun-java-sdk-core.version>
    <aliyun-sdk-oss.version>3.1.0</aliyun-sdk-oss.version>
    <aliyun-java-sdk-vod.version>2.15.2</aliyun-java-sdk-vod.version>
    <aliyun-java-vod-upload.version>1.4.11</aliyun-java-vod-upload.version>
    <aliyun-sdk-vod-upload.version>1.4.11</aliyun-sdk-vod-upload.version>
    <fastjson.version>1.2.28</fastjson.version>
    <gson.version>2.8.2</gson.version>
    <json.version>20170516</json.version>
```

```
    <commons-dbutils.version>1.7</commons-dbutils.version>
    <canal.client.version>1.1.0</canal.client.version>
    <docker.image.prefix>zx</docker.image.prefix>
    <cloud-alibaba.version>0.2.2.RELEASE</cloud-alibaba.version>
</properties>
```

### 配置 `<dependencyManagement>` 锁定依赖的版本

```
<dependencyManagement>
    <dependencies>
        <!--Spring Cloud-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>Hoxton.RELEASE</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>

        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-alibaba-dependencies</artifactId>
            <version>${cloud-alibaba.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
        <!--mybatis-plus 持久层-->
        <dependency>
            <groupId>com.baomidou</groupId>
            <artifactId>mybatis-plus-boot-starter</artifactId>
            <version>${mybatis-plus.version}</version>
        </dependency>

        <!-- velocity 模板引擎, Mybatis Plus 代码生成器需要 -->
        <dependency>
            <groupId>org.apache.velocity</groupId>
            <artifactId>velocity-engine-core</artifactId>
            <version>${velocity.version}</version>
        </dependency>

        <!--swagger-->
```

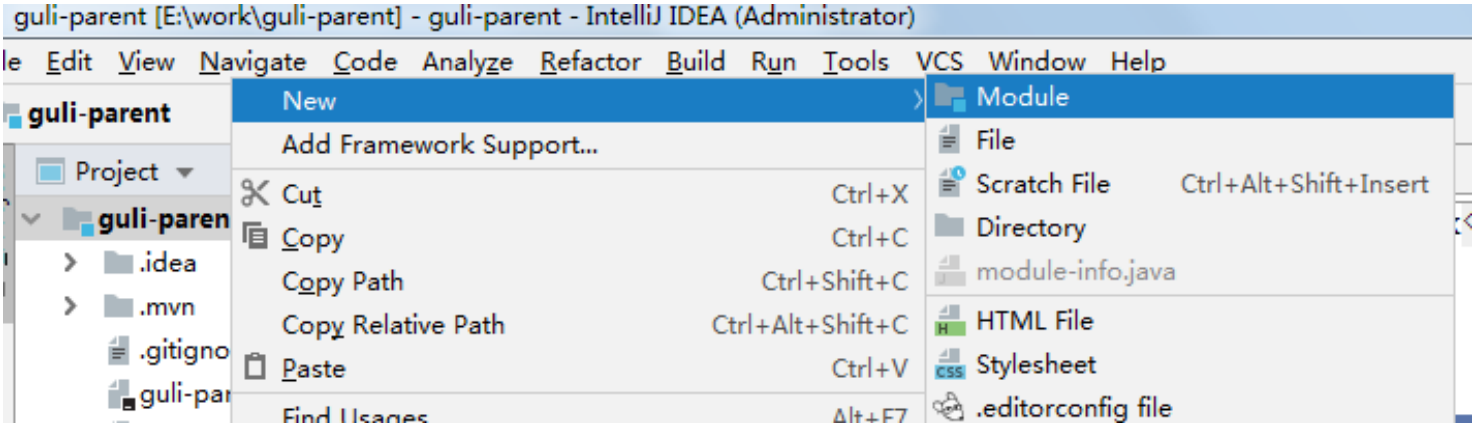
```
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>${swagger.version}</version>
        </dependency>
        <!--swagger ui-->
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>${swagger.version}</version>
        </dependency>

        <!--aliyunOSS-->
        <dependency>
            <groupId>com.aliyun.oss</groupId>
            <artifactId>aliyun-sdk-oss</artifactId>
            <version>${aliyun.oss.version}</version>
        </dependency>

        <!--日期时间工具-->
        <dependency>
            <groupId>joda-time</groupId>
            <artifactId>joda-time</artifactId>
            <version>${jodatime.version}</version>
        </dependency>

        <!--xls-->
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi</artifactId>
            <version>${poi.version}</version>
        </dependency>
        <!--xlsx-->
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi-ooxml</artifactId>
            <version>${poi.version}</version>
        </dependency>

        <!--文件上传-->
        <dependency>
            <groupId>commons-fileupload</groupId>
            <artifactId>commons-fileupload</artifactId>
```

```
            <version>${commons-fileupload.version}</version>
        </dependency>

        <!--commons-io-->
        <dependency>
            <groupId>commons-io</groupId>
            <artifactId>commons-io</artifactId>
            <version>${commons-io.version}</version>
        </dependency>

        <!--httpclient-->
        <dependency>
            <groupId>org.apache.httpcomponents</groupId>
            <artifactId>httpclient</artifactId>
            <version>${httpclient.version}</version>
        </dependency>

        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>${gson.version}</version>
        </dependency>

        <!-- JWT -->
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt</artifactId>
            <version>${jwt.version}</version>
        </dependency>

        <!--aliyun-->
        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-core</artifactId>
            <version>${aliyun-java-sdk-core.version}</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun.oss</groupId>
            <artifactId>aliyun-sdk-oss</artifactId>
            <version>${aliyun-sdk-oss.version}</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun</groupId>
```

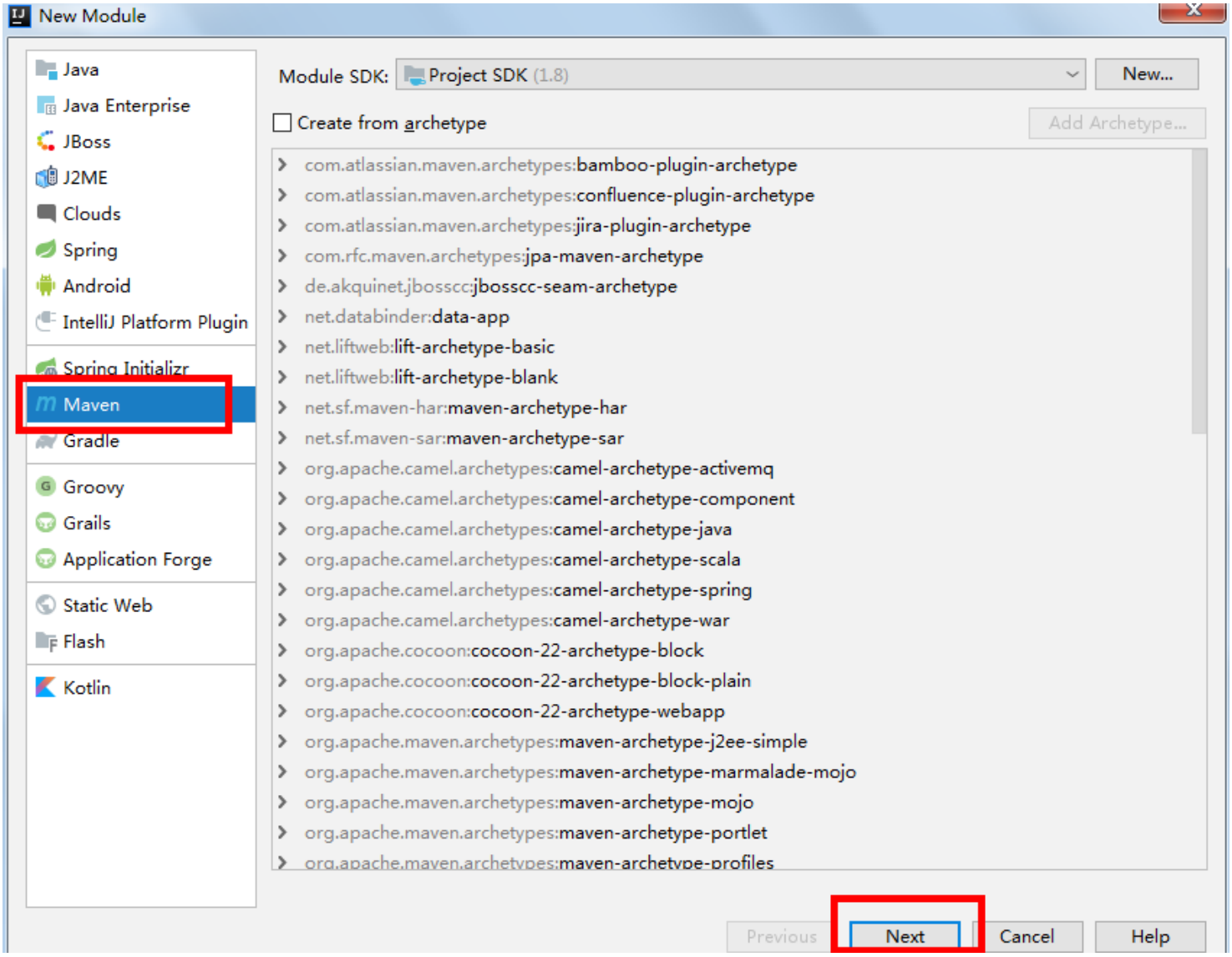
```
            <artifactId>aliyun-java-sdk-vod</artifactId>
            <version>${aliyun-java-sdk-vod.version}</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-vod-upload</artifactId>
            <version>${aliyun-java-vod-upload.version}</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-sdk-vod-upload</artifactId>
            <version>${aliyun-sdk-vod-upload.version}</version>
        </dependency>
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>${fastjson.version}</version>
        </dependency>
        <dependency>
            <groupId>org.json</groupId>
            <artifactId>json</artifactId>
            <version>${json.version}</version>
        </dependency>

        <dependency>
            <groupId>commons-dbutils</groupId>
            <artifactId>commons-dbutils</artifactId>
            <version>${commons-dbutils.version}</version>
        </dependency>

        <dependency>
            <groupId>com.alibaba.otter</groupId>
            <artifactId>canal.client</artifactId>
            <version>${canal.client.version}</version>
        </dependency>
    </dependencies>
</dependencyManagement>
```

# 一、搭建service模块

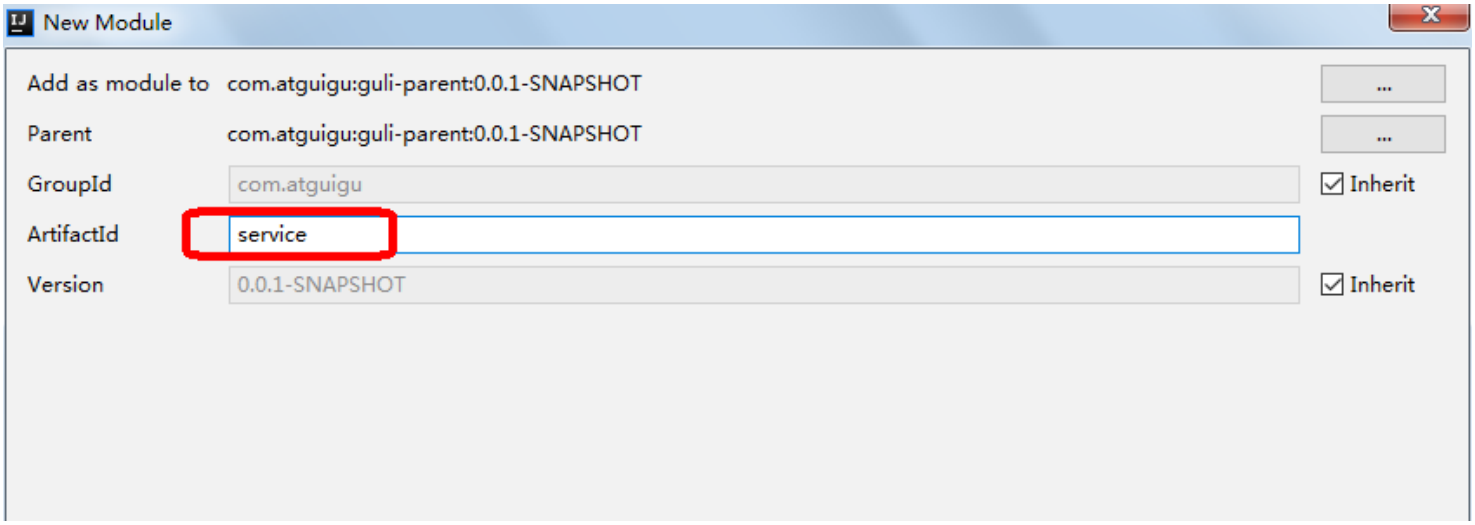
## 1、在父工程guli-parent下面创建模块service




选择 **maven**类型，点击下一步




输入模块名称 **service**，下一步完成创建




## 2、添加模块类型是**pom**

`<artifactId>` 节点后面添加 pom类型

```
<artifactId>service</artifactId>
<packaging>pom</packaging>
```

## 3、添加项目需要的依赖

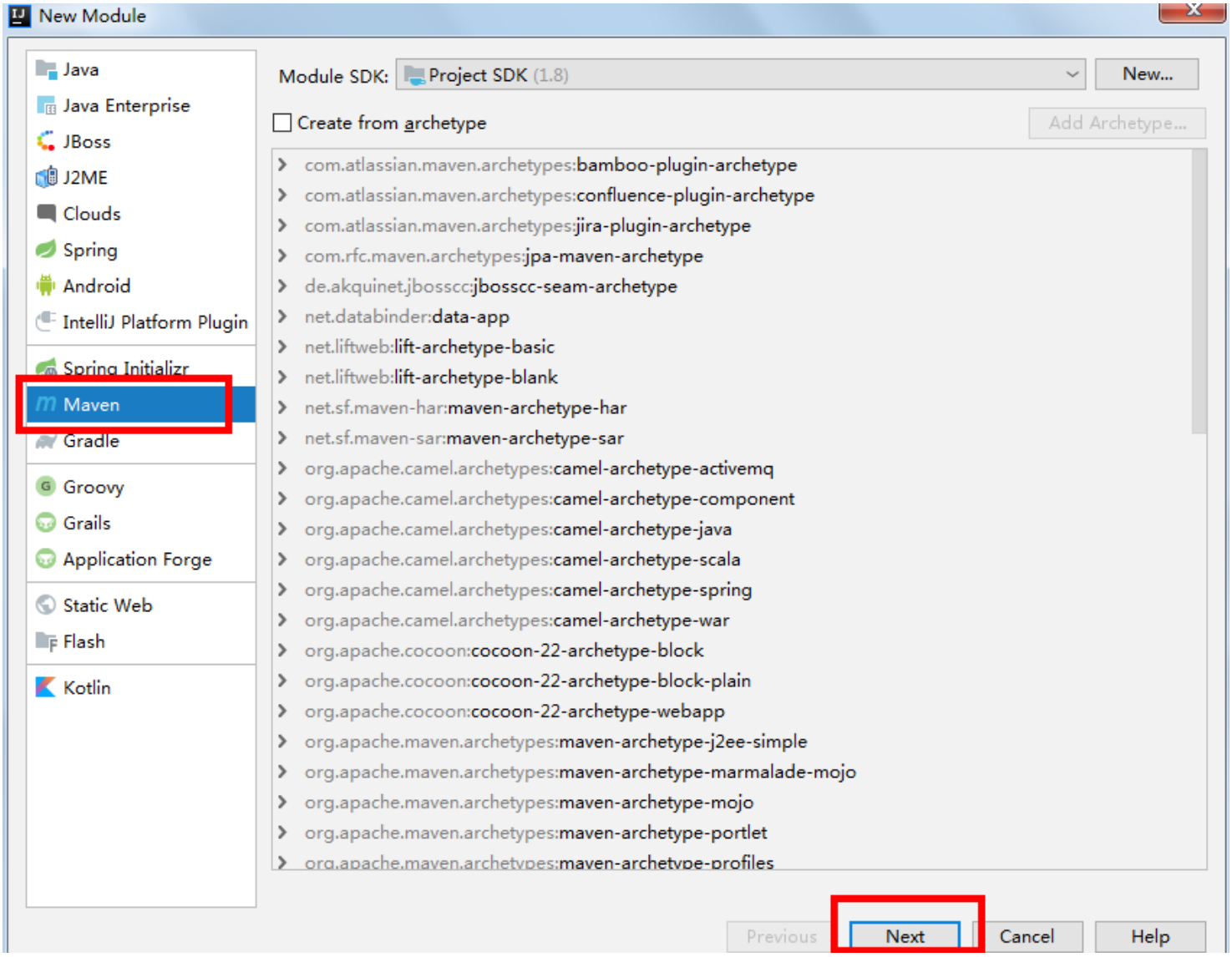
```
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-ribbon</artifactId>
    </dependency>

    <!--hystrix依赖，主要是用  @HystrixCommand -->
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
    </dependency>

    <!--服务注册-->
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
    </dependency>
    <!--服务调用-->
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-openfeign</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!--mybatis-plus-->
    <dependency>
```

```
        <groupId>com.baomidou</groupId>
        <artifactId>mybatis-plus-boot-starter</artifactId>
    </dependency>

    <!--mysql-->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>

    <!-- velocity 模板引擎, Mybatis Plus 代码生成器需要 -->
    <dependency>
        <groupId>org.apache.velocity</groupId>
        <artifactId>velocity-engine-core</artifactId>
    </dependency>

    <!--swagger-->
    <dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-swagger2</artifactId>
    </dependency>
    <dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-swagger-ui</artifactId>
    </dependency>

    <!--lombok用来简化实体类：需要安装lombok插件-->
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
    </dependency>

    <!--xls-->
    <dependency>
        <groupId>org.apache.poi</groupId>
        <artifactId>poi</artifactId>
    </dependency>

    <dependency>
        <groupId>org.apache.poi</groupId>
        <artifactId>poi-ooxml</artifactId>
    </dependency>

```

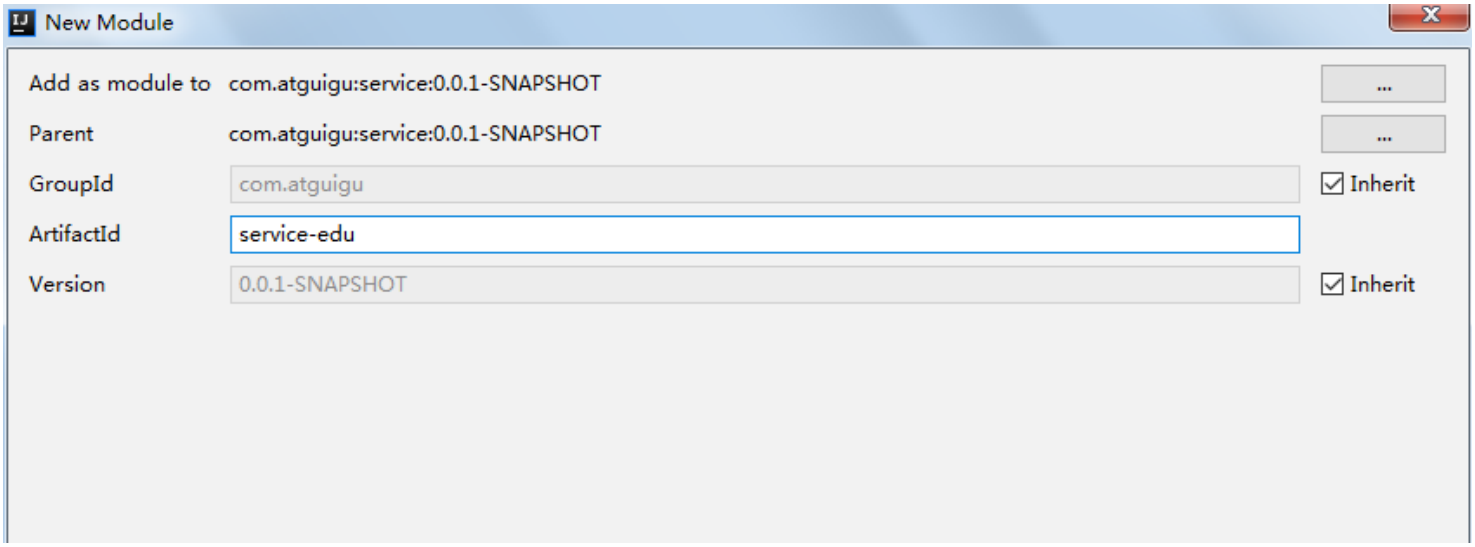
```
    <dependency>
        <groupId>commons-fileupload</groupId>
        <artifactId>commons-fileupload</artifactId>
    </dependency>

    <!--httpclient-->
    <dependency>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpclient</artifactId>
    </dependency>
    <!--commons-io-->
    <dependency>
        <groupId>commons-io</groupId>
        <artifactId>commons-io</artifactId>
    </dependency>
    <!--gson-->
    <dependency>
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
    </dependency>

    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
    </dependency>
</dependencies>
```

# 二、搭建**service-edu**模块

## 1、在父工程**service**模块下面创建子模块**service-edu**

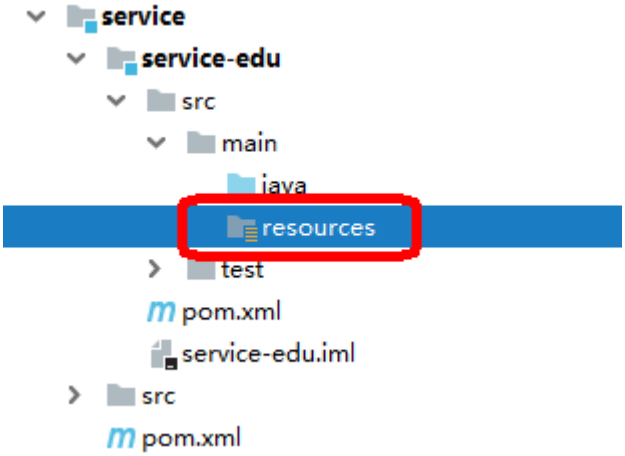



输入模块名称 **service-edu**，下一步完成创建



# 一、讲师管理模块配置

## 1、在service下面service-edu模块中创建配置文件




### resources目录下创建文件 application.properties

```
# 服务端口
server.port=8001
# 服务名
spring.application.name=service-edu

# 环境设置：dev、test、prod
spring.profiles.active=dev

# mysql数据库连接
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/guli?serverTimezone=GMT%2B8
spring.datasource.username=root
spring.datasource.password=root

#mybatis日志
mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl
```

### 或者在resources目录下创建文件 application.yml

```
#### application.yml
```

```
spring:
  application:
    name: service-edu
profiles:
    active: dev


####  application-dev.yml
server:
  port: 8001
mybatis-plus:
  configuration:
    log-impl: org.apache.ibatis.logging.stdout.StdOutImpl
mapper-locations: classpath:com/atguigu/service/*/mapper/*.xml
global-config:
    db-config:
      logic-delete-value: 1
logic-not-delete-value: 0
spring:
datasource:
    type: com.zaxxer.hikari.HikariDataSource
driver-class-name: com.mysql.cj.jdbc.Driver
url: jdbc:mysql://localhost:3306/guli?serverTimezone=GMT%2B8
username: root
password: root
hikari:
      connection-test-query: SELECT 1
connection-timeout: 60000
idle-timeout: 500000
max-lifetime: 540000
maximum-pool-size: 12
minimum-idle: 10
pool-name: GuliHikariPool
jackson:
    date-format: yyyy-MM-dd HH:mm:ss
time-zone: GMT+8

```

## 2、创建MP代码生成器

在test/java目录下创建包com.atguigu.eduservice，创建代码生成器：CodeGenerator.java

```java
public class getCode {

    @Test
    public void main1() {

        // 1、创建代码生成器
        AutoGenerator mpg = new AutoGenerator();

        // 2、全局配置
        GlobalConfig gc = new GlobalConfig();
        String projectPath = System.getProperty("user.dir");
        System.out.println(projectPath);
        gc.setOutputDir(projectPath + "/src/main/java");
        gc.setAuthor("atguigu");
        gc.setOpen(false); //生成后是否打开资源管理器
        gc.setFileOverride(false); //重新生成时文件是否覆盖
        /*
         * mp生成service层代码，默认接口名称第一个字母有 I
         * UcenterService
         * */
        gc.setServiceName("%sService"); //去掉Service接口的首字母I
        gc.setIdType(IdType.ID_WORKER); //主键策略
        gc.setDateType(DateType.ONLY_DATE);//定义生成的实体类中日期类型
        gc.setSwagger2(true);//开启Swagger2模式

        mpg.setGlobalConfig(gc);

        // 3、数据源配置
        DataSourceConfig dsc = new DataSourceConfig();
        dsc.setUrl("jdbc:mysql://localhost:3306/guli?serverTimezone=GMT%2B8");
        dsc.setDriverName("com.mysql.cj.jdbc.Driver");
        dsc.setUsername("root");
        dsc.setPassword("root");
        dsc.setDbType(DbType.MYSQL);
        mpg.setDataSource(dsc);

        // 4、包配置
        PackageConfig pc = new PackageConfig();
        pc.setModuleName("serviceedu"); //模块名
        pc.setParent("com.atguigu");
        pc.setController("controller");
        pc.setEntity("entity");
```

```
        pc.setService("service");
        pc.setMapper("mapper");
        mpg.setPackageInfo(pc);

        // 5、策略配置
        StrategyConfig strategy = new StrategyConfig();
        strategy.setInclude("edu_teacher");
        strategy.setNaming(NamingStrategy.underline_to_camel);//数据库表映射到实体的命名策略
        strategy.setTablePrefix(pc.getModuleName() + "_"); //生成实体时去掉表前缀

        strategy.setColumnNaming(NamingStrategy.underline_to_camel);//数据库表字段映射到实体的命名策略
        strategy.setEntityLombokModel(true); // lombok 模型 @Accessors(chain = true) setter链式操作

        strategy.setRestControllerStyle(true); //restful api风格控制器
        strategy.setControllerMappingHyphenStyle(true); //url中驼峰转连字符

        mpg.setStrategy(strategy);

        // 6、执行
        mpg.execute();
    }
}
```

# 二、编写后台管理api接口

## 1、编写controller代码

```
@Autowired
private TeacherService teacherService;

@GetMapping
public List<Teacher> list(){
    return teacherService.list(null);
}
```

## 2、创建SpringBoot配置类

在edu包下创建config包，创建MyBatisPlusConfig.java

```
package com.guli.edu.config;

@Configuration
@EnableTransactionManagement
@MapperScan("com.atguigu.eduservice.mapper")
public class MyBatisPlusConfig {

}
```

## 3、配置SQL执行性能分析插件

```
/**
     * SQL 执行性能分析插件
     * 开发环境使用，线上不推荐。 maxTime 指的是 sql 最大执行时长
     */
@Bean
@Profile({"dev","test"})// 设置 dev test 环境开启
public PerformanceInterceptor performanceInterceptor() {
    PerformanceInterceptor performanceInterceptor = new PerformanceInterceptor();
    performanceInterceptor.setMaxTime(1000);//ms，超过此处设置的ms则sql不执行
    performanceInterceptor.setFormat(true);
    return performanceInterceptor;
}
```

## 4、创建SpringBoot启动类

创建启动类 EduApplication.java，注意启动类的创建位置

```
@SpringBootApplication
public class EduApplication {

    public static void main(String[] args) {
        SpringApplication.run(EduApplication.class, args);
    }
```

```
}
```

## 5、运行启动类

访问http://localhost:8001/eduservice/teacher

得到json数据

## 6、统一返回的json时间格式

默认情况下json时间格式带有时区，并且是世界标准时间，和我们的时间差了八个小时

在application.properties中设置

```
#返回json的全局时间格式
spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
spring.jackson.time-zone=GMT+8
```

# 三、讲师逻辑删除功能

## 1、EduTeacherController添加删除方法

```
@DeleteMapping("{id}")
public boolean removeById(@PathVariable String id){
    return teacherService.removeById(id);
}
```

## 2、配置逻辑删除插件

MyBatisPlusConfig中配置

```
/**
      * 逻辑删除插件
      */
```

```
@Bean
public ISqlInjector sqlInjector() {
    return new LogicSqlInjector();
}
```

## 3、使用postman测试删除


测试结果：数据库中的is_deleted字段被修改为1

# 四、跨域配置

## 1、什么是跨域

浏览器从一个域名的网页去请求另一个域名的资源时，域名、端口、协议任一不同，都是跨域 。前后端分离开发中，需要考虑ajax跨域的问题。

这里我们可以从服务端解决这个问题

## 2、配置

在Controller类上添加注解

```
@CrossOrigin //跨域
```

# 一、Swagger2介绍

前后端分离开发模式中，api文档是最好的沟通方式。

Swagger 是一个规范和完整的框架，用于生成、描述、调用和可视化 RESTful 风格的 Web 服务。

1. 及时性 (接口变更后，能够及时准确地通知相关前后端开发人员)
2. 规范性 (并且保证接口的规范性，如接口的地址，请求方式，参数及响应格式和错误信息)
3. 一致性 (接口信息一致，不会出现因开发人员拿到的文档版本不一致，而出现分歧)
4. 可测性 (直接在接口文档上进行测试，以方便理解业务)

# 二、配置Swagger2

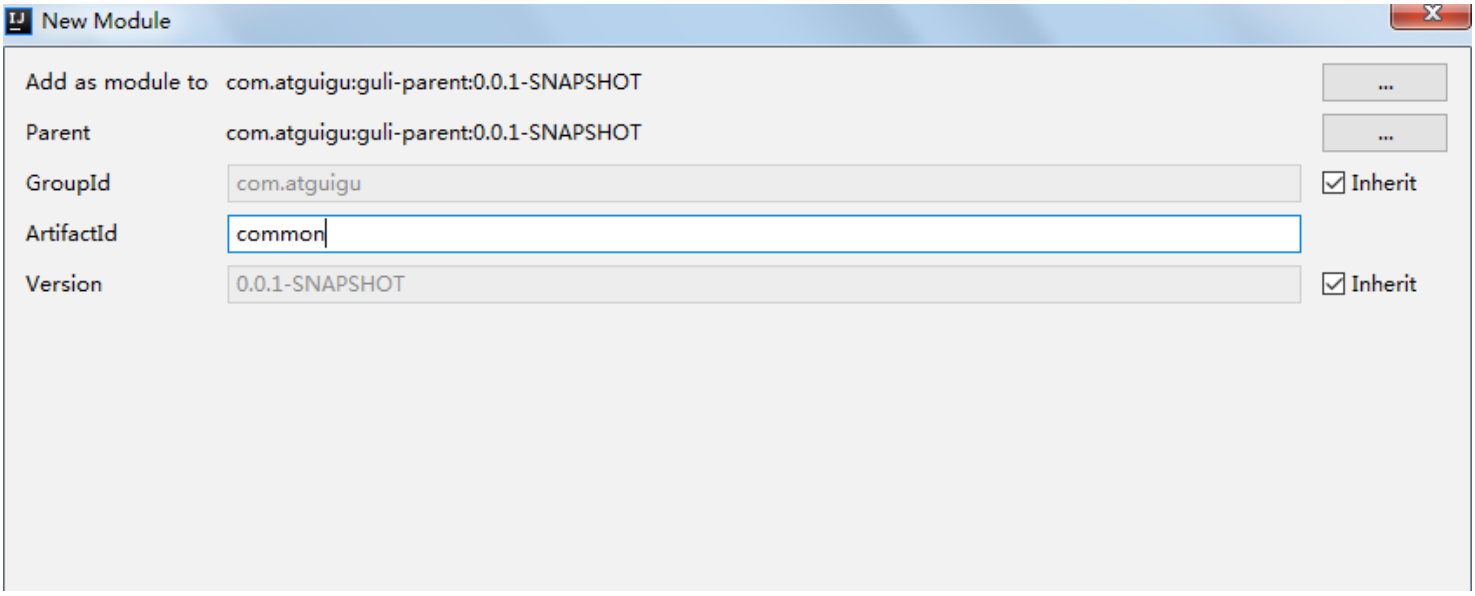
## 1、创建common模块

在**guli-parent**下创建模块**common**

配置：

groupId：com.atguigu

artifactId：common

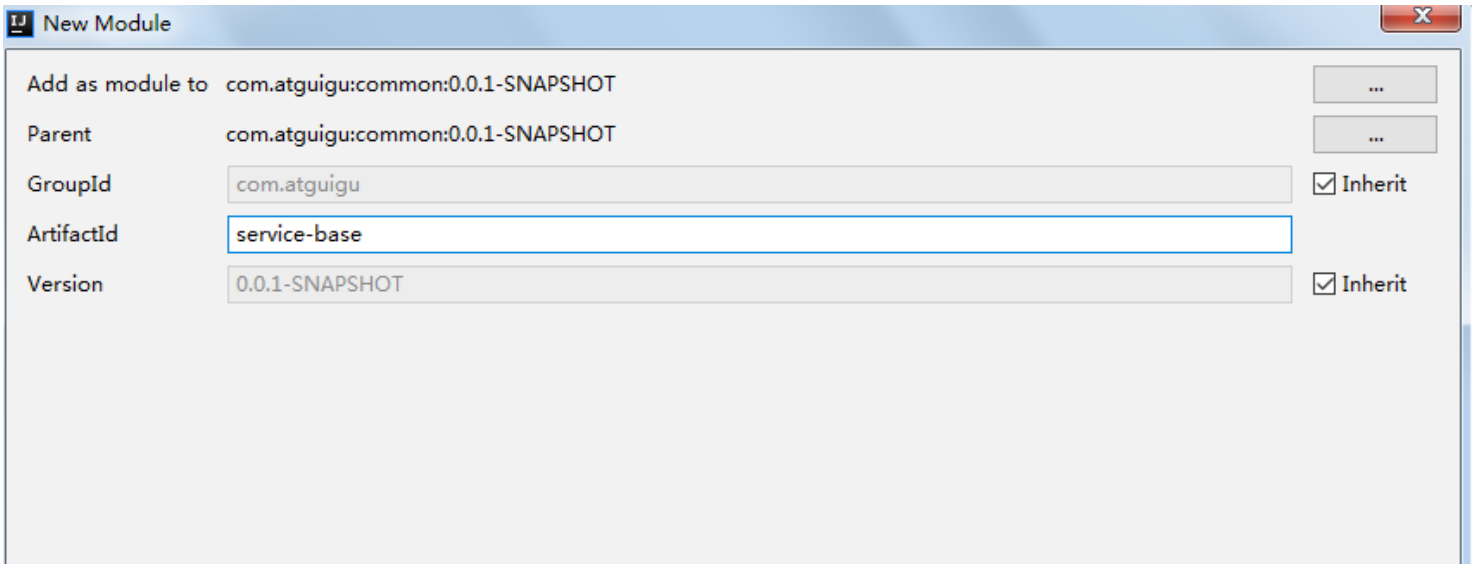



## 2、在common中引入**相关依赖**

```xml
<dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
            <scope>provided </scope>
        </dependency>

        <!--mybatis-plus-->
        <dependency>
            <groupId>com.baomidou</groupId>
            <artifactId>mybatis-plus-boot-starter</artifactId>
            <scope>provided </scope>
        </dependency>

        <!--lombok用来简化实体类：需要安装lombok插件-->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <scope>provided </scope>
        </dependency>

        <!--swagger-->
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <scope>provided </scope>
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <scope>provided </scope>
        </dependency>

        <!-- redis -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-redis</artifactId>
        </dependency>

        <!-- spring2.X集成redis所需common-pool2
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-pool2</artifactId>
```

```
            <version>2.6.0</version>
        </dependency>-->
    </dependencies>
```

## 3、在common下面创建子模块service-base




## 3、在模块service-base中，创建swagger的配置类

创建包com.atguigu.servicebase.config，创建类SwaggerConfig

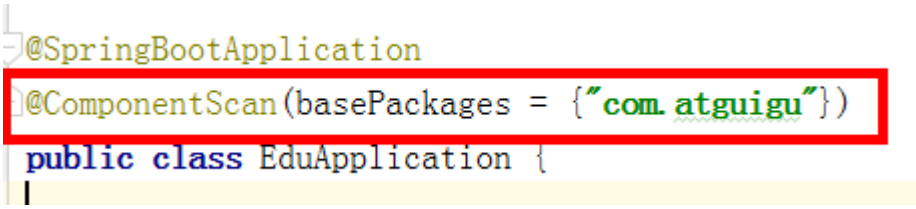
```
@Configuration
@EnableSwagger2
public class SwaggerConfig {

    @Bean
    public Docket webApiConfig(){

        return new Docket(DocumentationType.SWAGGER_2)
                .groupName("webApi")
                .apiInfo(webApiInfo())
                .select()
                .paths(Predicates.not(PathSelectors.regex("/admin/.*")))
                .paths(Predicates.not(PathSelectors.regex("/error.*")))
                .build();

    }

```

```
    private ApiInfo webApiInfo(){

        return new ApiInfoBuilder()
                .title("网站-课程中心API文档")
                .description("本文档描述了课程中心微服务接口定义")
                .version("1.0")
                .contact(new Contact("Helen", "http://atguigu.com",
"55317332@qq.com"))
                .build();
    }
}
```

## 4、在模块service模块中引入service-base

```
<dependency>
    <groupId>com.atguigu</groupId>
    <artifactId>service-base</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</dependency>
```

## 5、在service-edu启动类上添加注解，进行测试




## 6、API模型

可以添加一些自定义设置，例如：

定义样例数据

```
@ApiModelProperty(value = "创建时间", example = "2019-01-01 8:00:00")
```

```
@TableField(fill = FieldFill.INSERT)
private Date gmtCreate;

@ApiModelProperty(value = "更新时间", example = "2019-01-01 8:00:00")
@TableField(fill = FieldFill.INSERT_UPDATE)
private Date gmtModified;
```


## 5、定义接口说明和参数说明

定义在类上：@Api

定义在方法上：@ApiOperation

定义在参数上：@ApiParam

```
@Api(description="讲师管理")
@RestController
@RequestMapping("/admin/edu/teacher")
public class TeacherAdminController {

    @Autowired
    private TeacherService teacherService;

```

```
    @ApiOperation(value = "所有讲师列表")
    @GetMapping
    public List<Teacher> list(){
        return teacherService.list(null);
    }

    @ApiOperation(value = "根据ID删除讲师")
    @DeleteMapping("{id}")
    public boolean removeById(
            @ApiParam(name = "id", value = "讲师ID", required = true)
            @PathVariable String id){
        return teacherService.removeById(id);
    }
}
```

# 一、统一返回数据格式

项目中我们会将响应封装成json返回，一般我们会将所有接口的数据格式统一， 使前端(iOS Android, Web)对数据的操作更一致、轻松。

一般情况下，统一返回数据格式没有固定的格式，只要能描述清楚返回的数据状态以及要返回的具体数据就可以。但是一般会包含状态码、返回消息、数据这几部分内容

例如，我们的系统要求返回的基本数据格式如下：

**列表：**

```
{
  "success": true,
  "code": 20000,
  "message": "成功",
  "data": {
    "items": [
      {
        "id": "1",
        "name": "刘德华",
        "intro": "毕业于师范大学数学系，热爱教育事业，执教数学思维6年有余"
      }
    ]
  }
}
```

**分页：**

```
{
  "success": true,
  "code": 20000,
  "message": "成功",
  "data": {
    "total": 17,
    "rows": [
      {
        "id": "1",
        "name": "刘德华",
        "intro": "毕业于师范大学数学系，热爱教育事业，执教数学思维6年有余"
```

```
      }
    ]
  }
}
```

**没有返回数据：**

```
{
  "success": true,
  "code": 20000,
  "message": "成功",
  "data": {}
}
```

**失败：**

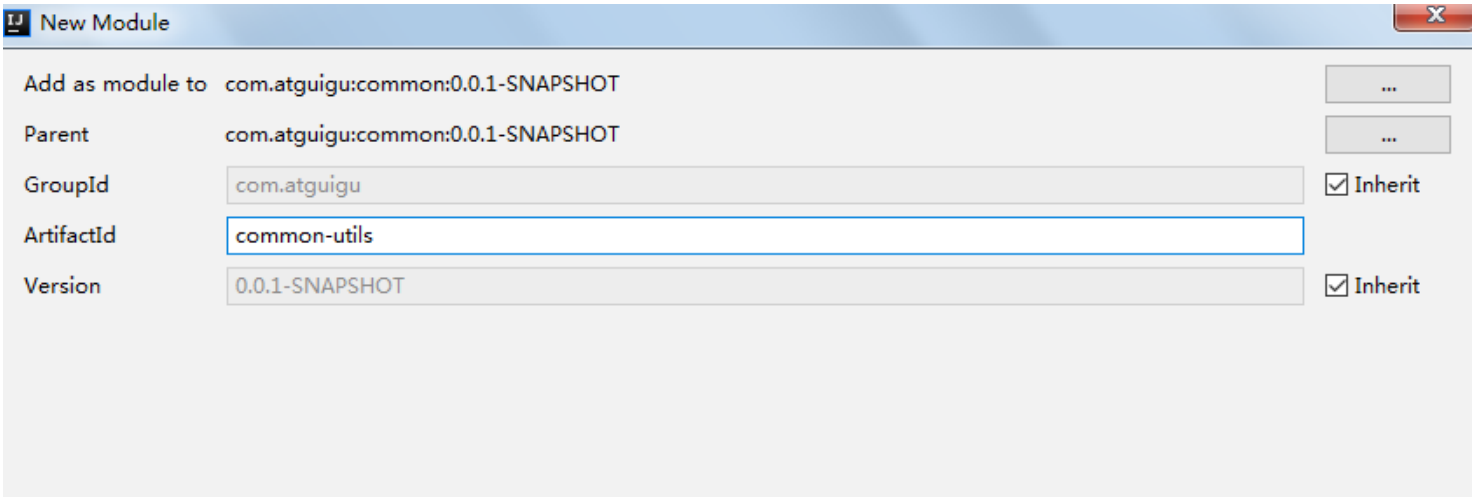
```
{
  "success": false,
  "code": 20001,
  "message": "失败",
  "data": {}
}
```

因此，我们定义统一结果

```
{
  "success": 布尔, //响应是否成功
  "code": 数字, //响应码
  "message": 字符串, //返回消息
  "data": HashMap //返回数据，放在键值对中
}
```

# 二、创建统一结果返回类

## 1、在common模块下创建子模块common-utils




# 2、创建接口定义返回码

**创建包com.atguigu.commonutils，创建接口 ResultCode.java**

```
package com.atguigu.commonutils;

public interface ResultCode {

    public static Integer SUCCESS = 20000;

    public static Integer ERROR = 20001;
}
```

# 4、创建结果类

**创建类 R.java**

```
@Data
public class R {
    @ApiModelProperty(value = "是否成功")
    private Boolean success;

    @ApiModelProperty(value = "返回码")
    private Integer code;

    @ApiModelProperty(value = "返回消息")
    private String message;

```

```
    @ApiModelProperty(value = "返回数据")
    private Map<String, Object> data = new HashMap<String, Object>();

    private R(){}

    public static R ok(){
        R r = new R();
        r.setSuccess(true);
        r.setCode(ResultCode.SUCCESS);
        r.setMessage("成功");
        return r;
    }

    public static R error(){
        R r = new R();
        r.setSuccess(false);
        r.setCode(ResultCode.ERROR);
        r.setMessage("失败");
        return r;
    }

    public R success(Boolean success){
        this.setSuccess(success);
        return this;
    }

    public R message(String message){
        this.setMessage(message);
        return this;
    }

    public R code(Integer code){
        this.setCode(code);
        return this;
    }

    public R data(String key, Object value){
        this.data.put(key, value);
        return this;
    }

    public R data(Map<String, Object> map){
        this.setData(map);
```

```
        return this;
    }
}
```

# 二、统一返回结果使用

## 1、在service模块中添加依赖

```
<dependency>
    <groupId>com.atguigu</groupId>
    <artifactId>common_utils</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</dependency>
```

## 2、修改Controller中的返回结果

列表

```
@ApiOperation(value = "所有讲师列表")
@GetMapping
public R list(){
    List<Teacher> list = teacherService.list(null);
    return R.ok().data("items", list);
}
```

删除

```
@ApiOperation(value = "根据ID删除讲师")
@DeleteMapping("{id}")
public R removeById(
    @ApiParam(name = "id", value = "讲师ID", required = true)
    @PathVariable String id){
```

```
    teacherService.removeById(id);
    return R.ok();
}
```

# 一、分页

## 1、MyBatisPlusConfig中配置分页插件

```
/**
 * 分页插件
 */
@Bean
public PaginationInterceptor paginationInterceptor() {
    return new PaginationInterceptor();
}
```

## 2、分页Controller方法

TeacherAdminController中添加分页方法

```
@ApiOperation(value = "分页讲师列表")
@GetMapping("{page}/{limit}")
public R pageList(
    @ApiParam(name = "page", value = "当前页码", required = true)
    @PathVariable Long page,
    @ApiParam(name = "limit", value = "每页记录数", required = true)
    @PathVariable Long limit){
    Page<Teacher> pageParam = new Page<>(page, limit);
    teacherService.page(pageParam, null);
    List<Teacher> records = pageParam.getRecords();
    long total = pageParam.getTotal();
    return  R.ok().data("total", total).data("rows", records);
}
```

## 3、Swagger中测试

# 二、条件查询

根据讲师名称name，讲师头衔level、讲师入驻时间gmt_create（时间段）查询

## 1、创建查询对象

创建com.guli.edu.query包，创建TeacherQuery.java查询对象

```
package com.guli.edu.query;
@ApiModel(value = "Teacher查询对象", description = "讲师查询对象封装")
@Data
public class TeacherQuery implements Serializable {
    private static final long serialVersionUID = 1L;
    @ApiModelProperty(value = "教师名称,模糊查询")
    private String name;
    @ApiModelProperty(value = "头衔 1高级讲师 2首席讲师")
    private Integer level;

    @ApiModelProperty(value = "查询开始时间", example = "2019-01-01 10:10:10")
    private String begin;//注意，这里使用的是String类型，前端传过来的数据无需进行类型转换
    @ApiModelProperty(value = "查询结束时间", example = "2019-12-01 10:10:10")
    private String end;
}
```

## 2、service

接口

```
package com.guli.edu.service;
public interface TeacherService extends IService<Teacher> {
    void pageQuery(Page<Teacher> pageParam, TeacherQuery teacherQuery);
}
```

实现

```
package com.guli.edu.service.impl;
@Service
```

```
public class TeacherServiceImpl extends ServiceImpl<TeacherMapper, Teacher>
implements TeacherService {
    @Override
    public void pageQuery(Page<Teacher> pageParam, TeacherQuery teacherQuery)
{
        QueryWrapper<Teacher> queryWrapper = new QueryWrapper<>();
        queryWrapper.orderByAsc("sort");
        if (teacherQuery == null){
            baseMapper.selectPage(pageParam, queryWrapper);
            return;
        }
        String name = teacherQuery.getName();
        Integer level = teacherQuery.getLevel();
        String begin = teacherQuery.getBegin();
        String end = teacherQuery.getEnd();
        if (!StringUtils.isEmpty(name)) {
            queryWrapper.like("name", name);
        }
        if (!StringUtils.isEmpty(level) ) {
            queryWrapper.eq("level", level);
        }
        if (!StringUtils.isEmpty(begin)) {
            queryWrapper.ge("gmt_create", begin);
        }
        if (!StringUtils.isEmpty(end)) {
            queryWrapper.le("gmt_create", end);
        }
        baseMapper.selectPage(pageParam, queryWrapper);
    }
}
```

# 3、controller

TeacherAdminController中修改 pageList方法：

增加参数TeacherQuery teacherQuery，非必选

```
@ApiOperation(value = "分页讲师列表")
@GetMapping("{page}/{limit}")
public R pageQuery(
    @ApiParam(name = "page", value = "当前页码", required = true)
    @PathVariable Long page,
    @ApiParam(name = "limit", value = "每页记录数", required = true)
```

```
    @PathVariable Long limit,
    @ApiParam(name = "teacherQuery", value = "查询对象", required = false)
    TeacherQuery teacherQuery){
    Page<Teacher> pageParam = new Page<>(page, limit);
    teacherService.pageQuery(pageParam, teacherQuery);
    List<Teacher> records = pageParam.getRecords();
    long total = pageParam.getTotal();
    return  R.ok().data("total", total).data("rows", records);
}
```

## 4、Swagger中测试

# 一、自动填充封装

## 1、在service-base模块中添加

创建包**handler**，创建自动填充类 **MyMetaObjectHandler**

```
@Component
public class MyMetaObjectHandler implements MetaObjectHandler {
    @Override
    public void insertFill(MetaObject metaObject) {
        this.setFieldValByName("gmtCreate", new Date(), metaObject);
        this.setFieldValByName("gmtModified", new Date(), metaObject);
    }

    @Override
    public void updateFill(MetaObject metaObject) {
        this.setFieldValByName("gmtModified", new Date(), metaObject);

    }
}
```

## 2、在实体类添加自动填充注解

```
@ApiModelProperty(value = "创建时间")
@TableField(fill = FieldFill.INSERT)
private Date gmtCreate;

@ApiModelProperty(value = "更新时间")
@TableField(fill = FieldFill.INSERT_UPDATE)
private Date gmtModified;
```

# 二、controller方法定义

## 1、新增

```
    @ApiOperation(value = "新增讲师")
    @PostMapping
    public R save(
            @ApiParam(name = "teacher", value = "讲师对象", required = true)
            @RequestBody Teacher teacher){

        teacherService.save(teacher);
        return R.ok();
    }


```

## 2、根据id查询

```
@ApiOperation(value = "根据ID查询讲师")
    @GetMapping("{id}")
    public R getById(
            @ApiParam(name = "id", value = "讲师ID", required = true)
            @PathVariable String id){

        Teacher teacher = teacherService.getById(id);
        return R.ok().data("item", teacher);
    }

```

## 3、根据id修改

```
@ApiOperation(value = "根据ID修改讲师")
@PutMapping("{id}")
public R updateById(
    @ApiParam(name = "id", value = "讲师ID", required = true)
    @PathVariable String id,

    @ApiParam(name = "teacher", value = "讲师对象", required = true)
    @RequestBody Teacher teacher){

```

```
    teacher.setId(id);
    teacherService.updateById(teacher);
    return R.ok();
}
```

# 一、什么是统一异常处理

## 1、制造异常

除以0

```
int a = 10/0;
```

Response Body


```
{
  "timestamp": "2019-12-17T06:08:35.649+0000",
  "status": 500,
  "error": "Internal Server Error",
  "message": "/ by zero",
  "path": "/serviceedu/teacher"
}
```


## 2、什么是统一异常处理

我们想让异常结果也显示为统一的返回结果对象，并且统一处理系统的异常信息，那么需要统一异常处理

# 二、统一异常处理

## 1、创建统一异常处理器

在service-base中创建统一异常处理类GlobalExceptionHandler.java：

```
/**
 * 统一异常处理类
 */
@ControllerAdvice
public class GlobalExceptionHandler {

```

```
    @ExceptionHandler(Exception.class)
    @ResponseBody
    public R error(Exception e){
        e.printStackTrace();
        return R.error();
    }
}
```

## 2、测试

返回统一错误结果

Response Body


```
{
  "success": false,
  "code": 20001,
  "message": "失败",
  "data": {}
}
```


# 三、处理特定异常

## 1、添加异常处理方法

GlobalExceptionHandler.java中添加

```
@ExceptionHandler(ArithmeticException.class)
@ResponseBody
public R error(ArithmeticException e){
    e.printStackTrace();
    return R.error().message("执行了自定义异常");
}
```

## 2、测试

Response Body


```
{
  "success": false,
  "code": 20001,
  "message": "执行了自定义异常",
  "data": {}
}
```


# 四、自定义异常

## 1、创建自定义异常类

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class GuliException extends RuntimeException {

    @ApiModelProperty(value = "状态码")
    private Integer code;

    private String msg;

}
```

## 2、业务中需要的位置抛出GuliException

```
try {
    int a = 10/0;
}catch(Exception e) {
    throw new GuliException(20001,"出现自定义异常");
}
```

## 3、添加异常处理方法

GlobalExceptionHandler.java中添加

```
@ExceptionHandler(GuliException.class)
@ResponseBody
public R error(GuliException e){
    e.printStackTrace();
    return R.error().message(e.getMsg()).code(e.getCode());
}
```

## 4、测试

Response Body


```
{
  "success": false,
  "code": 20001,
  "message": "出现自定义异常",
  "data": {}
}
```

# 一、日志

## 1、配置日志级别

日志记录器（Logger）的行为是分等级的。如下表所示：

分为：OFF、FATAL、ERROR、WARN、INFO、DEBUG、ALL

默认情况下，spring boot从控制台打印出来的日志级别只有INFO及以上级别，可以配置日志级别

```
# 设置日志级别
logging.level.root=WARN
```

这种方式只能将日志打印在控制台上

# 二、Logback日志

spring boot内部使用Logback作为日志实现的框架。

Logback和log4j非常相似，如果你对log4j很熟悉，那对logback很快就会得心应手。

logback相对于log4j的一些优点：https://blog.csdn.net/caisini_vc/article/details/48551287

## 1、配置logback日志

删除application.properties中的日志配置

安装idea彩色日志插件：grep-console

resources 中创建 logback-spring.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration  scan="true" scanPeriod="10 seconds">
    <!-- 日志级别从低到高分为TRACE < DEBUG < INFO < WARN < ERROR < FATAL，如果设置为WARN，则低于WARN的信息都不会输出 -->
    <!-- scan:当此属性设置为true时，配置文件如果发生改变，将会被重新加载，默认值为true -->
    <!-- scanPeriod:设置监测配置文件是否有修改的时间间隔，如果没有给出时间单位，默认单位是毫秒。当scan为true时，此属性生效。默认的时间间隔为1分钟。 -->
    <!-- debug:当此属性设置为true时，将打印出logback内部日志信息，实时查看logback运行状态。默认值为false。 -->
```

```
    <contextName>logback</contextName>
    <!-- name的值是变量的名称，value的值时变量定义的值。通过定义的值会被插入到logger上下文中。定义变量后，可以使“${}”来使用变量。 -->
    <property name="log.path" value="D:/guli_log/edu" />
    <!-- 彩色日志 -->
    <!-- 配置格式变量：CONSOLE_LOG_PATTERN 彩色日志格式 -->
    <!-- magenta:洋红 -->
    <!-- boldMagenta:粗红-->
    <!-- cyan:青色 -->
    <!-- white:白色 -->
    <!-- magenta:洋红 -->
    <property name="CONSOLE_LOG_PATTERN"
              value="%yellow(%date{yyyy-MM-dd HH:mm:ss}) |%highlight(%-5level) |%blue(%thread) |%blue(%file:%line) |%green(%logger) |%cyan(%msg%n)"/>
    <!--输出到控制台-->
    <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
        <!--此日志appender是为开发使用，只配置最底级别，控制台输出的日志级别是大于或等于此级别的日志信息-->
        <!-- 例如：如果此处配置了INFO级别，则后面其他位置即使配置了DEBUG级别的日志，也不会被输出 -->
        <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
            <level>INFO</level>
        </filter>
        <encoder>
            <Pattern>${CONSOLE_LOG_PATTERN}</Pattern>
            <!-- 设置字符集 -->
            <charset>UTF-8</charset>
        </encoder>
    </appender>
    <!--输出到文件-->
    <!-- 时间滚动输出 level为 INFO 日志 -->
    <appender name="INFO_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <!-- 正在记录的日志文件的路径及文件名 -->
        <file>${log.path}/log_info.log</file>
        <!--日志文件输出格式-->
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n</pattern>
            <charset>UTF-8</charset>
        </encoder>
        <!-- 日志记录器的滚动策略，按日期，按大小记录 -->
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <!-- 每天日志归档路径以及格式 -->
```

```
            <fileNamePattern>${log.path}/info/log-info-%d{yyyy-MM-dd}.%i.log</fileNamePattern>
            <timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
                <maxFileSize>100MB</maxFileSize>
            </timeBasedFileNamingAndTriggeringPolicy>
            <!--日志文件保留天数-->
            <maxHistory>15</maxHistory>
        </rollingPolicy>
        <!-- 此日志文件只记录info级别的 -->
        <filter class="ch.qos.logback.classic.filter.LevelFilter">
            <level>INFO</level>
            <onMatch>ACCEPT</onMatch>
            <onMismatch>DENY</onMismatch>
        </filter>
    </appender>
    <!-- 时间滚动输出 level为 WARN 日志 -->
    <appender name="WARN_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <!-- 正在记录的日志文件的路径及文件名 -->
        <file>${log.path}/log_warn.log</file>
        <!--日志文件输出格式-->
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n</pattern>
            <charset>UTF-8</charset> <!-- 此处设置字符集 -->
        </encoder>
        <!-- 日志记录器的滚动策略，按日期，按大小记录 -->
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>${log.path}/warn/log-warn-%d{yyyy-MM-dd}.%i.log</fileNamePattern>
            <timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
                <maxFileSize>100MB</maxFileSize>
            </timeBasedFileNamingAndTriggeringPolicy>
            <!--日志文件保留天数-->
            <maxHistory>15</maxHistory>
        </rollingPolicy>
        <!-- 此日志文件只记录warn级别的 -->
        <filter class="ch.qos.logback.classic.filter.LevelFilter">
            <level>warn</level>
            <onMatch>ACCEPT</onMatch>
            <onMismatch>DENY</onMismatch>
        </filter>
```

```
    </appender>
    <!-- 时间滚动输出 level为 ERROR 日志 -->
    <appender name="ERROR_FILE"
class="ch.qos.logback.core.rolling.RollingFileAppender">
        <!-- 正在记录的日志文件的路径及文件名 -->
        <file>${log.path}/log_error.log</file>
        <!--日志文件输出格式-->
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level
%logger{50} - %msg%n</pattern>
            <charset>UTF-8</charset> <!-- 此处设置字符集 -->
        </encoder>
        <!-- 日志记录器的滚动策略，按日期，按大小记录 -->
        <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>${log.path}/error/log-error-%d{yyyy-MM-
dd}.%i.log</fileNamePattern>
            <timeBasedFileNamingAndTriggeringPolicy
class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
                <maxFileSize>100MB</maxFileSize>
            </timeBasedFileNamingAndTriggeringPolicy>
            <!--日志文件保留天数-->
            <maxHistory>15</maxHistory>
        </rollingPolicy>
        <!-- 此日志文件只记录ERROR级别的 -->
        <filter class="ch.qos.logback.classic.filter.LevelFilter">
            <level>ERROR</level>
            <onMatch>ACCEPT</onMatch>
            <onMismatch>DENY</onMismatch>
        </filter>
    </appender>
    <!--
        <logger>用来设置某一个包或者具体的某一个类的日志打印级别、以及指
定<appender>。
        <logger>仅有一个name属性，
        一个可选的level和一个可选的addtivity属性。
        name:用来指定受此logger约束的某一个包或者具体的某一个类。
        level:用来设置打印级别，大小写无关：TRACE, DEBUG, INFO, WARN, ERROR, ALL
和 OFF，
              如果未设置此属性，那么当前logger将会继承上级的级别。
    -->
    <!--
        使用mybatis的时候，sql语句是debug下才会打印，而这里我们只配置了info，所以想
要查看sql语句的话，有以下两种操作：
        第一种把<root level="INFO">改成<root level="DEBUG">这样就会打印sql，不过
```

```
这样日志那边会出现很多其他消息
        第二种就是单独给mapper下目录配置DEBUG模式，代码如下，这样配置sql语句会打印，其他还是正常DEBUG级别：
     -->
    <!--开发环境:打印控制台-->
    <springProfile name="dev">
        <!--可以输出项目中的debug日志，包括mybatis的sql日志-->
        <logger name="com.guli" level="INFO" />
        <!--
            root节点是必选节点，用来指定最基础的日志输出级别，只有一个level属性
            level:用来设置打印级别，大小写无关：TRACE, DEBUG, INFO, WARN, ERROR, ALL 和 OFF，默认是DEBUG
            可以包含零个或多个appender元素。
        -->
        <root level="INFO">
            <appender-ref ref="CONSOLE" />
            <appender-ref ref="INFO_FILE" />
            <appender-ref ref="WARN_FILE" />
            <appender-ref ref="ERROR_FILE" />
        </root>
    </springProfile>
    <!--生产环境:输出到文件-->
    <springProfile name="pro">
        <root level="INFO">
            <appender-ref ref="CONSOLE" />
            <appender-ref ref="DEBUG_FILE" />
            <appender-ref ref="INFO_FILE" />
            <appender-ref ref="ERROR_FILE" />
            <appender-ref ref="WARN_FILE" />
        </root>
    </springProfile>
</configuration>
```

## 2、将错误日志输出到文件

GlobalExceptionHandler.java 中

类上添加注解

```
@Slf4j
```

异常输出语句

```
log.error(e.getMessage());
```

## 3、将日志堆栈信息输出到文件

定义工具类

guli-framework-common下创建util包，创建ExceptionUtil.java工具类

```
package com.guli.common.util;
public class ExceptionUtil {
    public static String getMessage(Exception e) {
        StringWriter sw = null;
        PrintWriter pw = null;
        try {
            sw = new StringWriter();
            pw = new PrintWriter(sw);
            // 将出错的栈信息输出到printWriter中
            e.printStackTrace(pw);
            pw.flush();
            sw.flush();
        } finally {
            if (sw != null) {
                try {
                    sw.close();
                } catch (IOException e1) {
                    e1.printStackTrace();
                }
            }
            if (pw != null) {
                pw.close();
            }
        }
        return sw.toString();
    }
}
```

调用

```
log.error(ExceptionUtil.getMessage(e));
```

GuliException中创建toString方法

```
@Override
public String toString() {
    return "GuliException{" +
        "message=" + this.getMessage() +
        ", code=" + code +
        '}';
}
```

# 一、前端开发

前端工程师“Front-End-Developer”源自于美国。大约从2005年开始正式的前端工程师角色被行业所认可，到了2010年，互联网开始全面进入移动时代，前端开发的工作越来越重要。

最初所有的开发工作都是由后端工程师完成的，随着业务越来越繁杂，工作量变大，于是我们将项目中的可视化部分和一部分交互功能的开发工作剥离出来，形成了前端开发。

由于互联网行业的急速发展，导致了在不同的国家，有着截然不同的分工体制。

在日本和一些人口比较稀疏的国家，例如加拿大、澳洲等，流行“Full-Stack Engineer”，也就是我们通常所说的全栈工程师。通俗点说就是一个人除了完成前端开发和后端开发工作以外，有的公司从产品设计到项目开发再到后期运维可能都是同一个人，甚至可能还要负责UI、配动画，也可以是扫地、擦窗、写文档、维修桌椅等等。

而在美国等互联网环境比较发达的国家项目开发的分工协作更为明确，整个项目开发分为前端、中间层和后端三个开发阶段，这三个阶段分别由三个或者更多的人来协同完成。

国内的大部分互联网公司只有前端工程师和后端工程师，中间层的工作有的由前端来完成，有的由后端来完成。

PRD（产品原型-产品经理） - PSD（视觉设计-UI工程师） - HTML/CSS/JavaScript（PC/移动端网页，实现网页端的视觉展示和交互-前端工程师）

# 二、下载和安装VS Code

## 1、下载地址

https://code.visualstudio.com/

## 2、安装

# 三、初始设置

# 1、中文界面配置

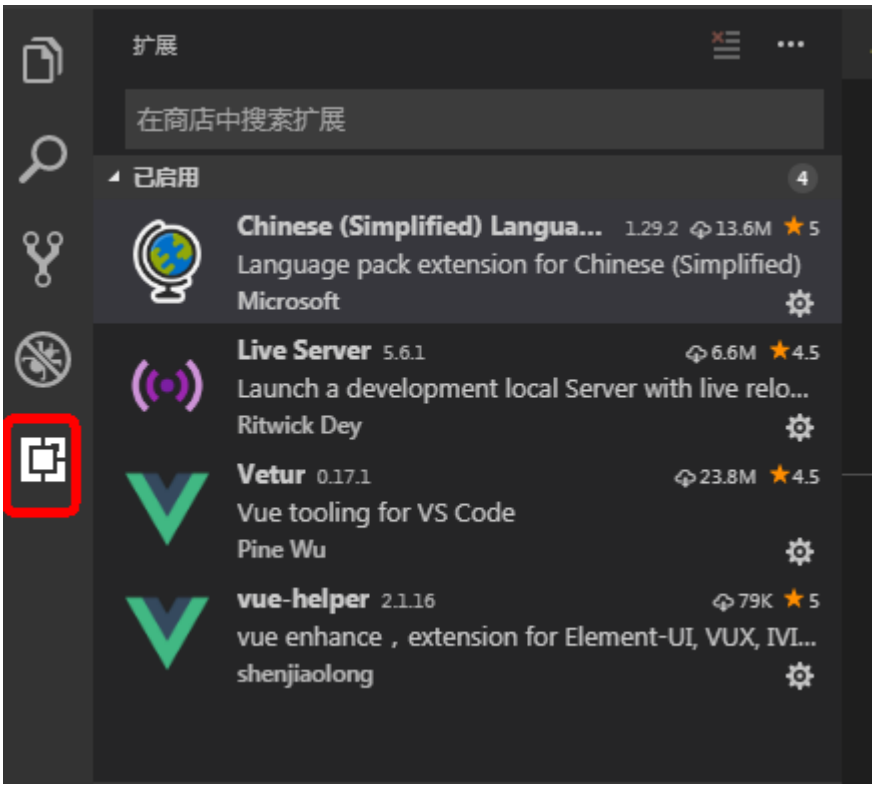
- 首先安装中文插件：Chinese (Simplified) Language Pack for Visual Studio Code
- 右下角弹出是否重启vs，点击“yes”
- 有些机器重启后如果界面没有变化，则 点击 左边栏Manage -> Command Paletet...【Ctrl+Shift+p】
- 在搜索框中输入“configure display language”，回车
- 打开locale.json文件，修改文件下的属性 "locale":"zh-cn"

```
{
    // 定义 VS Code 的显示语言。
    // 请参阅 https://go.microsoft.com/fwlink/?LinkId=761051，了解支持的语言列表。

    "locale":"zh-cn" // 更改将在重新启动 VS Code 之后生效。
}
```

- 重启vs

# 2、插件安装

为方便后续开发，建议安装如下插件（红色矩形框标记的插件）



## 3、创建项目

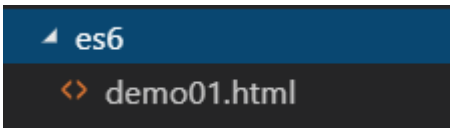
vscode本身没有新建项目的选项，所以要先创建一个空的文件夹，如project_xxxx。

然后打开vscode，再在vscode里面选择 File -> Open Folder 打开文件夹，这样才可以创建项目。

## 4、保存工作区

打开文件夹后，选择"文件 -> 将工作区另存为..."，为工作区文件起一个名字，存储在刚才的文件夹下即可

## 5、新建文件夹和网页




## 6、预览网页

**以文件路径方式打开网页预览**

需要安装"open in browser"插件：

文件右键 -> Open In Default Browser

**以服务器方式打开网页预览**

需要安装"Live Server"插件：

文件右键 -> Open with Live Server

## 7、设置字体大小

左边栏Manage -> settings -> 搜索 "font" -> Font size

# 8、开启完整的Emmet语法支持

设置中搜索 Emmet：启用如下选项，必要时重启vs

自学参考：http://es6.ruanyifeng.com/

# 一、ECMAScript 6 简介

ECMAScript 6.0（以下简称 ES6）是 JavaScript 语言的下一代标准，已经在 2015 年 6 月正式发布了。它的目标，是使得 JavaScript 语言可以用来编写复杂的大型应用程序，成为企业级开发语言。

## 1、ECMAScript 和 JavaScript 的关系

一个常见的问题是，ECMAScript 和 JavaScript 到底是什么关系？

要讲清楚这个问题，需要回顾历史。1996 年 11 月，JavaScript 的创造者 Netscape 公司，决定将 JavaScript 提交给标准化组织 ECMA，希望这种语言能够成为国际标准。次年，ECMA 发布 262 号标准文件（ECMA-262）的第一版，规定了浏览器脚本语言的标准，并将这种语言称为 ECMAScript，这个版本就是 1.0 版。

因此，ECMAScript 和 JavaScript 的关系是，前者是后者的规格，后者是前者的一种实现（另外的 ECMAScript 方言还有 Jscript 和 ActionScript）

## 2、ES6 与 ECMAScript 2015 的关系

ECMAScript 2015（简称 ES2015）这个词，也是经常可以看到的。它与 ES6 是什么关系呢？

2011 年，ECMAScript 5.1 版发布后，就开始制定 6.0 版了。因此，ES6 这个词的原意，就是指 JavaScript 语言的下一个版本。

ES6 的第一个版本，在 2015 年 6 月发布，正式名称是《ECMAScript 2015 标准》（简称 ES2015）。

2016 年 6 月，小幅修订的《ECMAScript 2016 标准》（简称 ES2016）如期发布，这个版本可以看作是 ES6.1 版，因为两者的差异非常小，基本上是同一个标准。根据计划，2017 年 6 月发布 ES2017 标准。

因此，ES6 既是一个历史名词，也是一个泛指，含义是 5.1 版以后的 JavaScript 的下一代标准，涵盖了 ES2015、ES2016、ES2017 等等，而 ES2015 则是正式名称，特指该年发布的正式版本的语言标准。本书中提到 ES6 的地方，一般是指 ES2015 标准，但有时也是泛指“下一代 JavaScript 语言”。

# 二、基本语法

ES标准中不包含 DOM 和 BOM的定义，只涵盖基本数据类型、关键字、语句、运算符、内建对象、内建函数等通用语法。

本部分只学习前端开发中ES6的最少必要知识，方便后面项目开发中对代码的理解。

## 1、let声明变量

创建 let.html

```
// var 声明的变量没有局部作用域
// let 声明的变量  有局部作用域
{
var a = 0
let b = 1
}
console.log(a)  // 0
console.log(b)  // ReferenceError: b is not defined
```

```
// var 可以声明多次
// let 只能声明一次
var m = 1
var m = 2
let n = 3
let n = 4
console.log(m)  // 2
console.log(n)  // Identifier 'n' has already been declared
```

## 2、const声明常量（只读变量）

创建 const.html

```
// 1、声明之后不允许改变
const PI = "3.1415926"
PI = 3  // TypeError: Assignment to constant variable.
```

```
// 2、一但声明必须初始化，否则会报错
const MY_AGE  // SyntaxError: Missing initializer in const declaration
```

## 3、解构赋值

创建 解构赋值.html

解构赋值是对赋值运算符的扩展。

他是一种针对数组或者对象进行模式匹配，然后对其中的变量进行赋值。

在代码书写上简洁且易读，语义更加清晰明了；也方便了复杂对象中数据字段获取。

```
//1、数组解构
// 传统
let a = 1, b = 2, c = 3
console.log(a, b, c)
// ES6
let [x, y, z] = [1, 2, 3]
console.log(x, y, z)
```

```
//2、对象解构
let user = {name: 'Helen', age: 18}
// 传统
let name1 = user.name
let age1 = user.age
console.log(name1, age1)
// ES6
let { name, age } =  user//注意：结构的变量必须是user中的属性
console.log(name, age)
```

# 4、模板字符串

创建 模板字符串.html

模板字符串相当于加强版的字符串，用反引号 `,除了作为普通字符串，还可以用来定义多行字符串，还可以在字符串中加入变量和表达式。

```
// 1、多行字符串
let string1 =  `Hey,
can you stop angry now?`
console.log(string1)
// Hey,
// can you stop angry now?
```

```
// 2、字符串插入变量和表达式。变量名写在 ${} 中，${} 中可以放入 JavaScript 表达式。
let name = "Mike"
let age = 27
let info = `My Name is ${name},I am ${age+1} years old next year.`
console.log(info)
// My Name is Mike,I am 28 years old next year.
```

```
// 3、字符串中调用函数
function f(){
    return "have fun!"
}
let string2 = `Game start,${f()}`
console.log(string2);  // Game start,have fun!
```

## 5、声明对象简写

创建 声明对象简写.html

```
const age = 12
const name = "Amy"

// 传统
const person1 = {age: age, name: name}
console.log(person1)

// ES6
const person2 = {age, name}
console.log(person2) //{age: 12, name: "Amy"}
```

## 6、定义方法简写

.html

创建 定义方法简写

```
// 传统
const person1 = {
    sayHi:function(){
        console.log("Hi")
    }
}
person1.sayHi();//"Hi"


// ES6
const person2 = {
    sayHi(){
        console.log("Hi")
    }
}
person2.sayHi()  //"Hi"
```

# 7、对象拓展运算符

创建 对象拓展运算符.html

拓展运算符（...）用于取出参数对象所有可遍历属性然后拷贝到当前对象。

```
// 1、拷贝对象
let person1 = {name: "Amy", age: 15}
let someone = { ...person1 }
console.log(someone)  //{name: "Amy", age: 15}
```

```
// 2、合并对象
let age = {age: 15}
let name = {name: "Amy"}
let person2 = {...age, ...name}
console.log(person2)  //{age: 15, name: "Amy"}
```

# 8、箭头函数

创建 箭头函数.html

箭头函数提供了一种更加简洁的函数书写方式。基本语法是：

```
参数 => 函数体
```

```
// 传统
var f1 = function(a){
    return a
}
console.log(f1(1))


// ES6
var f2 = a => a
console.log(f2(1))
```

```
// 当箭头函数没有参数或者有多个参数，要用 () 括起来。
// 当箭头函数函数体有多行语句，用 {} 包裹起来，表示代码块，
// 当只有一行语句，并且需要返回结果时，可以省略 {} ，结果会自动返回。
var f3 = (a,b) => {
    let result = a+b
    return result
}
console.log(f3(6,2))  // 8

// 前面代码相当于：
var f4 = (a,b) => a+b
```

箭头函数多用于匿名函数的定义

# 一、介绍

## 1、Vue.js 是什么

Vue (读音 /vjuː/，类似于 view) 是一套用于构建用户界面的渐进式框架。

Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。

官方网站：https://cn.vuejs.org

## 2、初始Vue.js

创建 demo.html

```
<!-- id标识vue作用的范围 -->
<div id="app">
    <!-- {{}} 插值表达式，绑定vue中的data数据 -->
    {{ message }}
</div>
<script src="vue.min.js"></script>
<script>

    // 创建一个vue对象
    new Vue({
        el: '#app',//绑定vue作用的范围
        data: {//定义页面中显示的模型数据
            message: 'Hello Vue!'
        }
    })

</script>
```

这就是声明式渲染：Vue.js 的核心是一个允许采用简洁的模板语法来声明式地将数据渲染进 DOM 的系统

这里的核心思想就是没有繁琐的DOM操作，例如jQuery中，我们需要先找到div节点，获取到DOM对

象，然后进行一系列的节点操作

**在vs code中创建代码片段：**

文件 => 首选项 => 用户代码片段 => 新建全局代码片段/或文件夹代码片段：vue-html.code-snippets

**注意：制作代码片段的时候，字符串中如果包含文件中复制过来的“Tab”键的空格，要换成“空格键”的空格**

```
{
    "vue htm": {
        "scope": "html",
        "prefix": "vuehtml",
        "body": [
            "<!DOCTYPE html>",
            "<html lang=\"en\">",
            "",
            "<head>",
            "    <meta charset=\"UTF-8\">",
            "    <meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">",
            "    <meta http-equiv=\"X-UA-Compatible\" content=\"ie=edge\">",
            "    <title>Document</title>",
            "</head>",
            "",
            "<body>",
            "    <div id=\"app\">",
            "",
            "    </div>",
            "    <script src=\"vue.min.js\"></script>",
            "    <script>",
            "        new Vue({",
            "            el: '#app',",
            "            data: {",
            "                $1",
            "            }",
            "        })",
            "    </script>",
            "</body>",
            "",
            "</html>",
        ],
```

```
        "description": "my vue template in html"
    }
}
```

# 二、基本语法

## 1、基本数据渲染和指令

创建 01-基本数据渲染和指令.html

你看到的 v-bind 特性被称为指令。指令带有前缀 v-

除了使用插值表达式{{}}进行数据渲染，也可以使用 v-bind指令，它的简写的形式就是一个冒号（:）

```
data: {
    content: '我是标题',
    message: '页面加载于 ' + new Date().toLocaleString()
}
```

```
<!-- 如果要将模型数据绑定在html属性中，则使用 v-bind 指令
     此时title中显示的是模型数据
-->
<h1 v-bind:title="message">
    {{content}}
</h1>

<!-- v-bind 指令的简写形式：  冒号（:）  -->
<h1 :title="message">
    {{content}}
</h1>
```

## 2、双向数据绑定

创建 02-双向数据绑定.html

双向数据绑定和单向数据绑定：使用 v-model 进行双向数据绑定

```
data: {
    searchMap:{
        keyWord: '尚硅谷'
    }
}
```

```
<!-- v-bind:value只能进行单向的数据渲染 -->
<input type="text" v-bind:value="searchMap.keyWord">
<!-- v-model 可以进行双向的数据绑定  -->
<input type="text" v-model="searchMap.keyWord">

<p>您要查询的是：{{searchMap.keyWord}}</p>
```

# 3、事件

创建 03-事件.html

需求：点击查询按钮，按照输入框中输入的内容查找公司相关信息

在前面的例子基础上，data节点中增加 result，增加 methods节点 并定义 search方法

```
data: {
     searchMap:{
         keyWord: '尚硅谷'
     },
     //查询结果
     result: {}
},
methods:{
    search(){
        console.log('search')
        //TODO
    }
}
```

html中增加 button 和 p

使用 v-on 进行数件处理，v-on:click 表示处理鼠标点击事件，事件调用的方法定义在 vue 对象声明的 methods 节点中

```
<!-- v-on 指令绑定事件，click指定绑定的事件类型，事件发生时调用vue中methods节点中定义的方法 -->
<button v-on:click="search()">查询</button>

<p>您要查询的是：{{searchMap.keyWord}}</p>
<p><a v-bind:href="result.site" target="_blank">{{result.title}}</a></p>
```

完善search方法

```
search(){
    console.log('search');
    this.result = {
        "title":"尚硅谷",
        "site":"http://www.atguigu.com"
    }
}
```

简写

```
<!-- v-on 指令的简写形式 @ -->
<button @click="search()">查询</button>
```

# 4、修饰符

创建 04-修饰符.html

修饰符 (Modifiers) 是以半角句号（.）指明的特殊后缀，用于指出一个指令应该以特殊方式绑定。

例如，.prevent 修饰符告诉 v-on 指令对于触发的事件调用 event.preventDefault()：

即阻止事件原本的默认行为

```
data: {
```

```
    user: {}
}
```

```
<!-- 修饰符用于指出一个指令应该以特殊方式绑定。
     这里的 .prevent 修饰符告诉 v-on 指令对于触发的事件调用js的event.preventDefault():
     即阻止表单提交的默认行为 -->
<form action="save" v-on:submit.prevent="onSubmit">
    <label for="username">
        <input type="text" id="username" v-model="user.username">
        <button type="submit">保存</button>
    </label>
</form>
```

```
methods: {
    onSubmit() {
        if (this.user.username) {
            console.log('提交表单')
        } else {
            alert('请输入用户名')
        }
    }
}
```

# 5、条件渲染

创建 05-条件渲染.html

v-if：条件指令

```
data: {
    ok: false
}
```

注意：单个复选框绑定到布尔值

```
<input type="checkbox" v-model="ok">同意许可协议
<!-- v:if条件指令：还有v-else、v-else-if 切换开销大 -->
<h1 v-if="ok">if: Lorem ipsum dolor sit amet.</h1>
<h1 v-else>no</h1>
```

v-show：条件指令

使用v-show完成和上面相同的功能

```
<!-- v:show 条件指令 初始渲染开销大 -->
<h1 v-show="ok">show: Lorem ipsum dolor sit amet.</h1>
<h1 v-show="!ok">no</h1>
```

- `v-if` 是“真正”的条件渲染，因为它会确保在切换过程中条件块内的事件监听器和子组件适当地被销毁和重建。
- `v-if` 也是**惰性的**：如果在初始渲染时条件为假，则什么也不做——直到条件第一次变为真时，才会开始渲染条件块。
- 相比之下，`v-show` 就简单得多——不管初始条件是什么，元素总是会被渲染，并且只是简单地基于 CSS 进行切换。
- 一般来说，`v-if` 有更高的切换开销，而 `v-show` 有更高的初始渲染开销。因此，如果需要非常频繁地切换，则使用 `v-show` 较好；如果在运行时条件很少改变，则使用 `v-if` 较好。

# 6、列表渲染

创建 06-列表渲染.html

v-for：列表循环指令

例**1**：简单的列表渲染

```
<!-- 1、简单的列表渲染 -->
<ul>
    <li v-for="n in 10">{{ n }} </li>
</ul>
<ul>
    <!-- 如果想获取索引，则使用index关键字，注意，圆括号中的index必须放在后面 -->
    <li v-for="(n, index) in 5">{{ n }} - {{ index }} </li>
```

```
</ul>
```

例**2**：遍历数据列表

```
data: {
    userList: [
        { id: 1, username: 'helen', age: 18 },
        { id: 2, username: 'peter', age: 28 },
        { id: 3, username: 'andy', age: 38 }
    ]
}
```

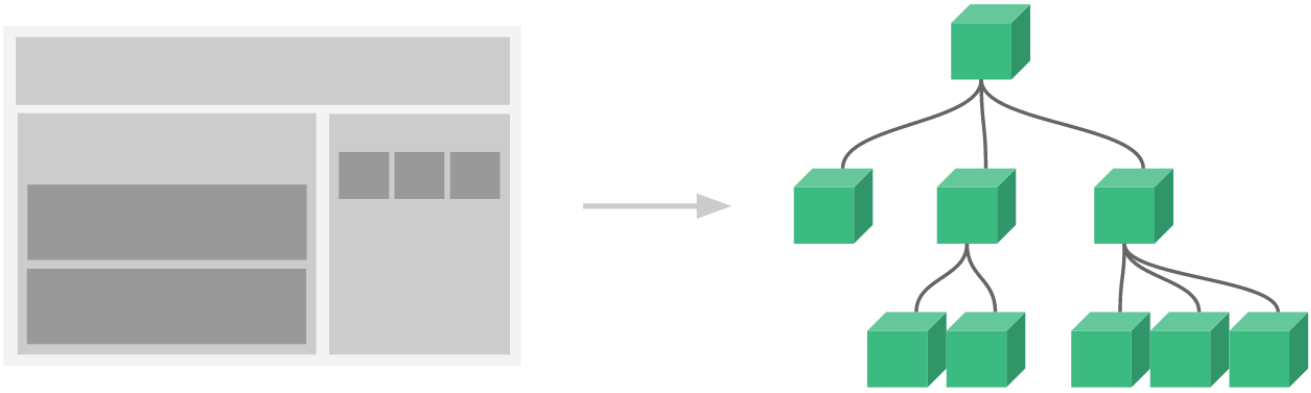
```
<!-- 2、遍历数据列表 -->
<table border="1">
    <!-- <tr v-for="item in userList"></tr> -->
    <tr v-for="(item, index) in userList">
        <td>{{index}}</td>
        <td>{{item.id}}</td>
        <td>{{item.username}}</td>
        <td>{{item.age}}</td>
    </tr>
</table>
```

# 一、组件（重点）

组件（Component）是 Vue.js 最强大的功能之一。

组件可以扩展 HTML 元素，封装可重用的代码。

组件系统让我们可以用独立可复用的小组件来构建大型应用，几乎任意类型的应用的界面都可以抽象为一个组件树：



## 1、局部组件

创建 01-1-局部组件.html

定义组件

```
var app = new Vue({
    el: '#app',
    // 定义局部组件，这里可以定义多个局部组件
    components: {
        //组件的名字
        'Navbar': {
            //组件的内容
            template: '<ul><li>首页</li><li>学员管理</li></ul>'
        }
    }
})
```

使用组件

```
<div id="app">
    <Navbar></Navbar>
</div>
```

## 2、全局组件

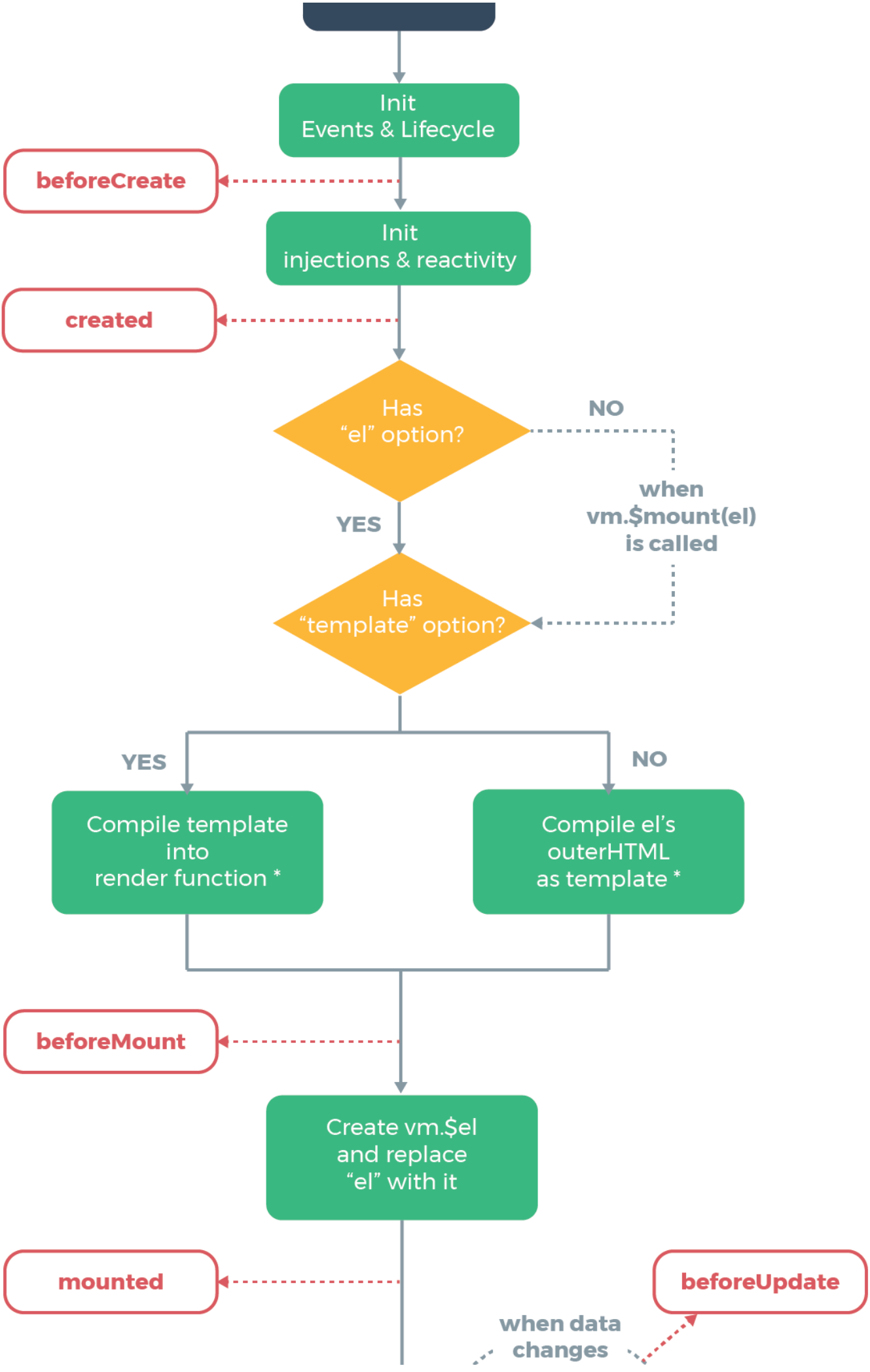
创建 01-2-全局组件.html

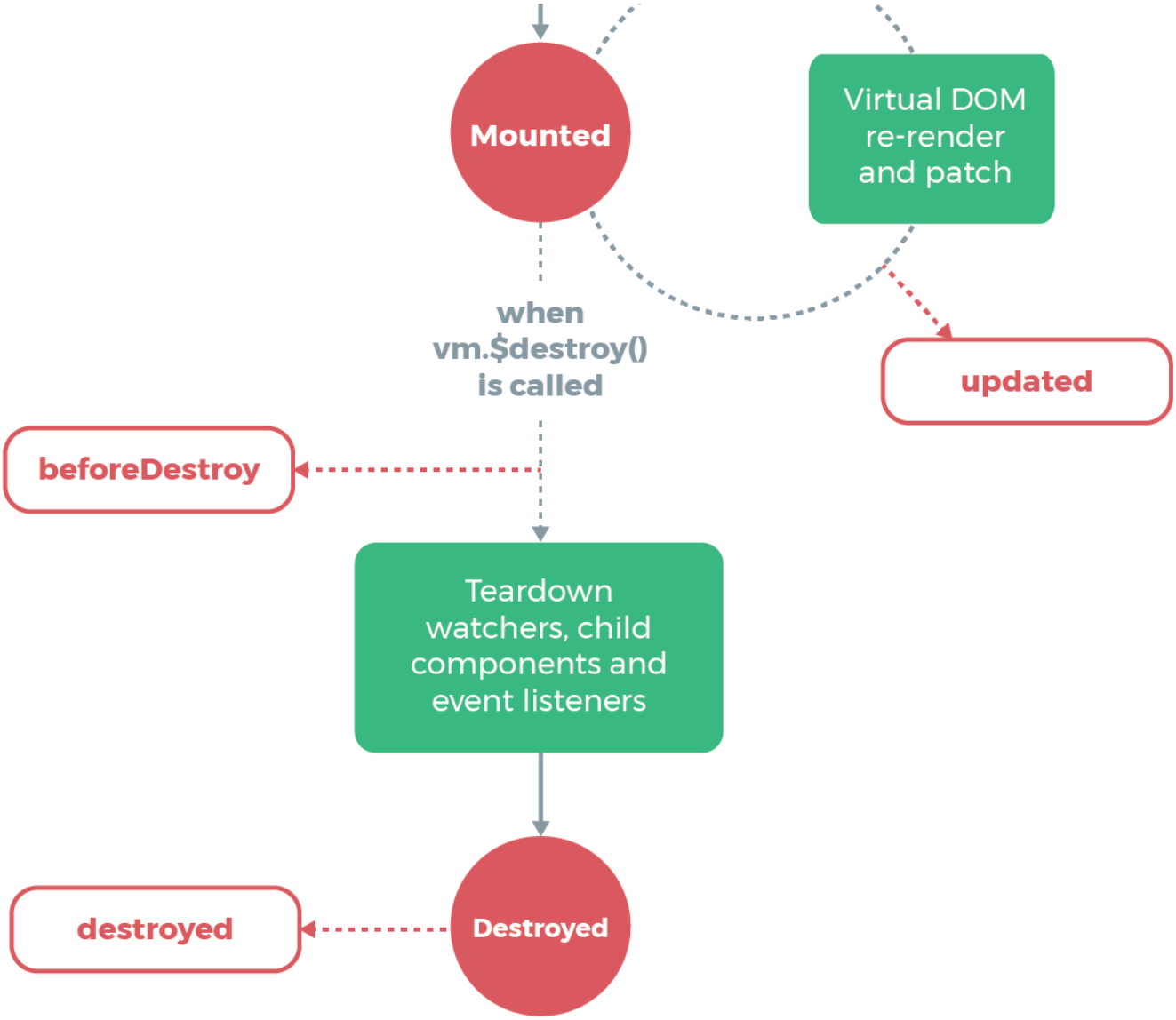
定义全局组件：components/Navbar.js

```
// 定义全局组件
Vue.component('Navbar', {
    template: '<ul><li>首页</li><li>学员管理</li><li>讲师管理</li></ul>'
})
```

```
<div id="app">
    <Navbar></Navbar>
</div>
<script src="vue.min.js"></script>
<script src="components/Navbar.js"></script>
<script>
    var app = new Vue({
        el: '#app'
    })
</script>
```

# 二、实例生命周期


Init
Events & Lifecycle
beforeCreate
Init
injections & reactivity
created
Has
“el” option?
NO
when
vm.$mount(el)
is called
YES
Has
“template” option?
YES
NO
Compile template
into
render function *
Compile el’s
outerHTML
as template *
beforeMount
Create vm.$el
and replace
“el” with it
mounted
beforeUpdate
when data
changes




* template compilation is performed ahead-of-time if using a build step, e.g. single-file components

创建 03-vue实例的生命周期.html

```
data: {
    message: '床前明月光'
},
methods: {
    show() {
        console.log('执行show方法')
    },
    update() {
```

```
        this.message = '玻璃好上霜'
    }
},
```

```
<button @click="update">update</button>
<h3 id="h3">{{ message }}</h3>
```

分析生命周期相关方法的执行时机

```
//===创建时的四个事件
beforeCreate() { // 第一个被执行的钩子方法：实例被创建出来之前执行
    console.log(this.message) //undefined
    this.show() //TypeError: this.show is not a function
    // beforeCreate执行时，data 和 methods 中的 数据都还没有没初始化
},
created() { // 第二个被执行的钩子方法
    console.log(this.message) //床前明月光
    this.show() //执行show方法
    // created执行时，data 和 methods 都已经被初始化好了！
    // 如果要调用 methods 中的方法，或者操作 data 中的数据，最早，只能在 created 中操作
},
beforeMount() { // 第三个被执行的钩子方法
    console.log(document.getElementById('h3').innerText) //{{ message }}
    // beforeMount执行时，模板已经在内存中编辑完成了，尚未被渲染到页面中
},
mounted() { // 第四个被执行的钩子方法
    console.log(document.getElementById('h3').innerText) //床前明月光
    // 内存中的模板已经渲染到页面，用户已经可以看见内容
},


//===运行中的两个事件
beforeUpdate() { // 数据更新的前一刻
    console.log('界面显示的内容：' + document.getElementById('h3').innerText)
    console.log('data 中的 message 数据是：' + this.message)
    // beforeUpdate执行时，内存中的数据已更新，但是页面尚未被渲染
},
updated() {
    console.log('界面显示的内容：' + document.getElementById('h3').innerText)
```

```
    console.log('data 中的 message 数据是: ' + this.message)
    // updated执行时，内存中的数据已更新，并且页面已经被渲染
}
```

# 四、路由

Vue.js 路由允许我们通过不同的 URL 访问不同的内容。

通过 Vue.js 可以实现多视图的单页Web应用（single page web application，SPA）。

Vue.js 路由需要载入 vue-router 库

创建 04-路由.html

## 1、引入js

```
<script src="vue.min.js"></script>
<script src="vue-router.min.js"></script>
```

## 2、编写html

```
<div id="app">
    <h1>Hello App!</h1>
    <p>
        <!-- 使用 router-link 组件来导航. -->
        <!-- 通过传入 `to` 属性指定链接. -->
        <!-- <router-link> 默认会被渲染成一个 `<a>` 标签 -->
        <router-link to="/">首页</router-link>
        <router-link to="/student">会员管理</router-link>
        <router-link to="/teacher">讲师管理</router-link>
    </p>
    <!-- 路由出口 -->
    <!-- 路由匹配到的组件将渲染在这里 -->
    <router-view></router-view>
</div>
```

## 3、编写js

```
<script>
    // 1. 定义（路由）组件。
    // 可以从其他文件 import 进来
    const Welcome = { template: '<div>欢迎</div>' }
    const Student = { template: '<div>student list</div>' }
    const Teacher = { template: '<div>teacher list</div>' }

    // 2. 定义路由
    // 每个路由应该映射一个组件。
    const routes = [
        { path: '/', redirect: '/welcome' }, //设置默认指向的路径
        { path: '/welcome', component: Welcome },
        { path: '/student', component: Student },
        { path: '/teacher', component: Teacher }
    ]

    // 3. 创建 router 实例，然后传 `routes` 配置
    const router = new VueRouter({
        routes //  （缩写）相当于 routes: routes
    })

    // 4. 创建和挂载根实例。
    // 从而让整个应用都有路由功能
    const app = new Vue({
        el: '#app',
        router
    })

    // 现在，应用已经启动了！
</script>
```

# 五、axios

axios是独立于vue的一个项目，基于promise用于浏览器和node.js的http客户端

- 在浏览器中可以帮助我们完成 ajax请求的发送

- 在node.js中可以向远程接口发送请求

## 获取数据

```
<script src="vue.min.js"></script>
<script src="axios.min.js"></script>
```

注意：测试时需要开启后端服务器，并且后端开启跨域访问权限

```
var app = new Vue({
    el: '#app',
    data: {
        memberList: []//数组
    },
    created() {
        this.getList()
    },

    methods: {

        getList(id) {
            //vm = this
            axios.get('http://localhost:8081/admin/ucenter/member')
            .then(response => {
                console.log(response)
                this.memberList = response.data.data.items
            })
            .catch(error => {
                console.log(error)
            })
        }
    }
})
```

### 控制台查看输出

## 2、显示数据

```
<div id="app">
    <table border="1">
        <tr>
            <td>id</td>
            <td>姓名</td>
        </tr>
        <tr v-for="item in memberList">
            <td>{{item.memberId}}</td>
            <td>{{item.nickname}}</td>
        </td>
    </tr>
</table>
</div>
```

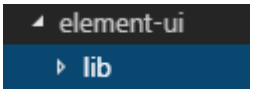
# 六、element-ui：

element-ui 是饿了么前端出品的基于 Vue.js的 后台组件库，方便程序员进行页面快速布局和构建

官网： http://element-cn.eleme.io/#/zh-CN

创建 06-element-ui.html

将element-ui引入到项目




## 1、引入css

```
<!-- import CSS -->
<link rel="stylesheet" href="element-ui/lib/theme-chalk/index.css">
```

## 2、引入js

```
<!-- import Vue before Element -->
<script src="vue.min.js"></script>
<!-- import JavaScript -->
<script src="element-ui/lib/index.js"></script>
```

## 3、编写html

```
<div id="app">
    <el-button @click="visible = true">Button</el-button>
    <el-dialog :visible.sync="visible" title="Hello world">
        <p>Try Element</p>
    </el-dialog>
</div>
```

关于.sync的扩展阅读

https://www.jianshu.com/p/d42c508ea9de

## 4、编写js

```
<script>
    new Vue({
      el: '#app',
      data: function () {//定义Vue中data的另一种方式
        return { visible: false }
      }
    })
</script>
```

测试

其他ui组件我们在项目中学习

# 一、简介

## 1、什么是Node.js

简单的说 Node.js 就是运行在服务端的 JavaScript。

Node.js是一个事件驱动I/O服务端JavaScript环境，基于Google的V8引擎，V8引擎执行Javascript的速度非常快，性能非常好。

## 2、Node.js有什么用

如果你是一个前端程序员，你不懂得像PHP、Python或Ruby等动态编程语言，然后你想创建自己的服务，那么Node.js是一个非常好的选择。

Node.js 是运行在服务端的 JavaScript，如果你熟悉Javascript，那么你将会很容易的学会Node.js。

当然，如果你是后端程序员，想部署一些高性能的服务，那么学习Node.js也是一个非常好的选择。

# 二、安装

## 1、下载

官网：https://nodejs.org/en/

中文网：http://nodejs.cn/

LTS：长期支持版本

Current：最新版

## 2、安装

## 3、查看版本

```
node -v
```

# 三、快速入门

## 1、创建文件夹nodejs

## 2、控制台程序

创建 01-控制台程序.js

```
console.log('Hello Node.js')
```

打开命令行终端：Ctrl + Shift + y

进入到程序所在的目录，输入

```
node 01-控制台程序.js
```

浏览器的内核包括两部分核心：

- DOM渲染引擎；
- js解析器（js引擎）
- js运行在浏览器中的内核中的js引擎内部

Node.js是脱离浏览器环境运行的JavaScript程序，基于V8 引擎（Chrome 的 JavaScript的引擎）

## 3、服务器端应用开发（了解）

创建 02-server-app.js

```
const http = require('http');
http.createServer(function (request, response) {
    // 发送 HTTP 头部
    // HTTP 状态值: 200 : OK
    // 内容类型: text/plain
    response.writeHead(200, {'Content-Type': 'text/plain'});
    // 发送响应数据 "Hello World"
    response.end('Hello Server');
```

```
}).listen(8888);
// 终端打印如下信息
console.log('Server running at http://127.0.0.1:8888/');
```

运行服务器程序

```
node 02-server-app.js
```

服务器启动成功后，在浏览器中输入：http://localhost:8888/ 查看webserver成功运行，并输出html页面

停止服务：ctrl + c

# 一、简介

## 1、什么是NPM

NPM全称Node Package Manager，是Node.js包管理工具，是全球最大的模块生态系统，里面所有的模块都是开源免费的；也是Node.js的包管理工具，相当于前端的Maven 。

## 2、NPM工具的安装位置

我们通过npm 可以很方便地下载js库，管理前端工程。

Node.js默认安装的npm包和工具的位置：Node.js目录\node_modules

- 在这个目录下你可以看见 npm目录，npm本身就是被NPM包管理器管理的一个工具，说明Node.js已经集成了npm工具

```
#在命令提示符输入 npm -v 可查看当前npm版本
npm -v
```

# 二、使用npm管理项目

## 1、创建文件夹npm

## 2、项目初始化

```
#建立一个空文件夹，在命令提示符进入该文件夹  执行命令初始化
npm init
#按照提示输入相关信息，如果是用默认值则直接回车即可。
#name: 项目名称
#version: 项目版本号
#description: 项目描述
#keywords: {Array}关键词，便于用户搜索到我们的项目
#最后会生成package.json文件，这个是包的配置文件，相当于maven的pom.xml
#我们之后也可以根据需要进行修改。
```

```
#如果想直接生成 package.json 文件，那么可以使用命令
npm init -y
```

## 2、修改npm镜像

NPM官方的管理的包都是从 http://npmjs.com下载的，但是这个网站在国内速度很慢。

这里推荐使用淘宝 NPM 镜像 http://npm.taobao.org/ ，淘宝 NPM 镜像是一个完整 npmjs.com 镜像，同步频率目前为 10分钟一次，以保证尽量与官方服务同步。

**设置镜像地址：**

```
#经过下面的配置，以后所有的 npm install 都会经过淘宝的镜像地址下载
npm config set registry https://registry.npm.taobao.org
#查看npm配置信息
npm config list
```

## 3、npm install命令的使用

```
#使用 npm install 安装依赖包的最新版，
#模块安装的位置：项目目录\node_modules
#安装会自动在项目目录下添加 package-lock.json文件，这个文件帮助锁定安装包的版本
#同时package.json 文件中，依赖包会被添加到dependencies节点下，类似maven中的<dependencies>
npm install jquery
#npm管理的项目在备份和传输的时候一般不携带node_modules文件夹
npm install #根据package.json中的配置下载依赖，初始化项目
#如果安装时想指定特定的版本
npm install jquery@2.1.x
#devDependencies节点：开发时的依赖包，项目打包到生产环境的时候不包含的依赖
#使用 -D参数将依赖添加到devDependencies节点
npm install --save-dev eslint
#或
npm install -D eslint
#全局安装
#Node.js全局安装的npm包和工具的位置：用户目录\AppData\Roaming\npm\node_modules
#一些命令行工具常使用全局安装的方式
npm install -g webpack
```

## 4、其它命令

```
#更新包（更新到最新版本）
npm update 包名
#全局更新
npm update -g 包名
#卸载包
npm uninstall 包名
#全局卸载
npm uninstall -g 包名
```

# 一、简介

Babel是一个广泛使用的转码器，可以将ES6代码转为ES5代码，从而在现有环境执行执行。

这意味着，你可以现在就用 ES6 编写程序，而不用担心现有环境是否支持。

# 二、安装

## 安装命令行转码工具

Babel提供babel-cli工具，用于命令行转码。它的安装命令如下：

```
npm install --global babel-cli

#查看是否安装成功
babel --version
```

# 三、Babel的使用

## 1、初始化项目

```
npm init -y
```

## 2、创建文件

src/example.js

下面是一段ES6代码：

```
// 转码前
// 定义数据
```

```
let input = [1, 2, 3]
// 将数组的每个元素 +1
input = input.map(item => item + 1)
console.log(input)
```

## 2、配置.babelrc

Babel的配置文件是.babelrc，存放在项目的根目录下，该文件用来设置转码规则和插件，基本格式如下。

```
{
    "presets": [],
    "plugins": []
}
```

presets字段设定转码规则，将es2015规则加入 .babelrc：

```
{
    "presets": ["es2015"],
    "plugins": []
}
```

## 3、安装转码器

在项目中安装

```
npm install --save-dev babel-preset-es2015
```

## 4、转码

```
# 转码结果写入一个文件
mkdir dist1
# --out-file 或 -o 参数指定输出文件
```

```
babel src/example.js --out-file dist1/compiled.js
# 或者
babel src/example.js -o dist1/compiled.js

# 整个目录转码
mkdir dist2
# --out-dir 或 -d 参数指定输出目录
babel src --out-dir dist2
# 或者
babel src -d dist2
```

# 一、模块化简介

## 1、模块化产生的背景

随着网站逐渐变成"互联网应用程序"，嵌入网页的Javascript代码越来越庞大，越来越复杂。

Javascript模块化编程，已经成为一个迫切的需求。理想情况下，开发者只需要实现核心的业务逻辑，其他都可以加载别人已经写好的模块。

但是，Javascript不是一种模块化编程语言，它不支持"类"（class），包（package）等概念，更遑论"模块"（module）了。

## 2、什么是模块化开发

传统非模块化开发有如下的缺点：

- 命名冲突
- 文件依赖

模块化规范：

- CommonJS模块化规范
- ES6模块化规范

# 二、CommonJS模块规范

每个文件就是一个模块，有自己的作用域。在一个文件里面定义的变量、函数、类，都是私有的，对其他文件不可见。

# 1、创建“module”文件夹

# 2、导出模块

创建 common-js模块化/四则运算.js

```
// 定义成员:
const sum = function(a,b){
    return parseInt(a) + parseInt(b)
}
const subtract = function(a,b){
    return parseInt(a) - parseInt(b)
}
const multiply = function(a,b){
    return parseInt(a) * parseInt(b)
}
const divide = function(a,b){
    return parseInt(a) / parseInt(b)
}
```

导出模块中的成员

```
// 导出成员:
module.exports = {
    sum: sum,
    subtract: subtract,
    multiply: multiply,
    divide: divide
}
```

简写

```
//简写
module.exports = {
    sum,
```

```
    subtract,
    multiply,
    divide
}
```

## 3、导入模块

创建 common-js模块化/引入模块.js

```
//引入模块，注意：当前路径必须写 ./
const m = require('./四则运算.js')
console.log(m)

const result1 = m.sum(1, 2)
const result2 = m.subtract(1, 2)
console.log(result1, result2)
```

## 4、运行程序

```
node common-js模块化/引入模块.js
```

CommonJS使用 exports 和require 来导出、导入模块。

# 三、ES6模块化规范

ES6使用 export 和 import 来导出、导入模块。

## 1、导出模块

创建 es6模块化/userApi.js

```
export function getList() {
    console.log('获取数据列表')
}
```

```
export function save() {
    console.log('保存数据')
}
```

## 2、导入模块

创建 es6模块化/userComponent.js

```
//只取需要的方法即可，多个方法用逗号分隔
import { getList, save } from "./userApi.js"
getList()
save()
```

**注意：这时的程序无法运行的，因为ES6的模块化无法在Node.js中执行，需要用Babel编辑成ES5后再执行。**

## 3、运行程序

```
node es6模块化-dist/userComponent.js
```

# 四、ES6模块化的另一种写法

## 1、导出模块

创建 es6模块化/userApi2.js

```
export default {
    getList() {
        console.log('获取数据列表2')
    },

    save() {
        console.log('保存数据2')
    }
```

```
}
```

## 2、导入模块

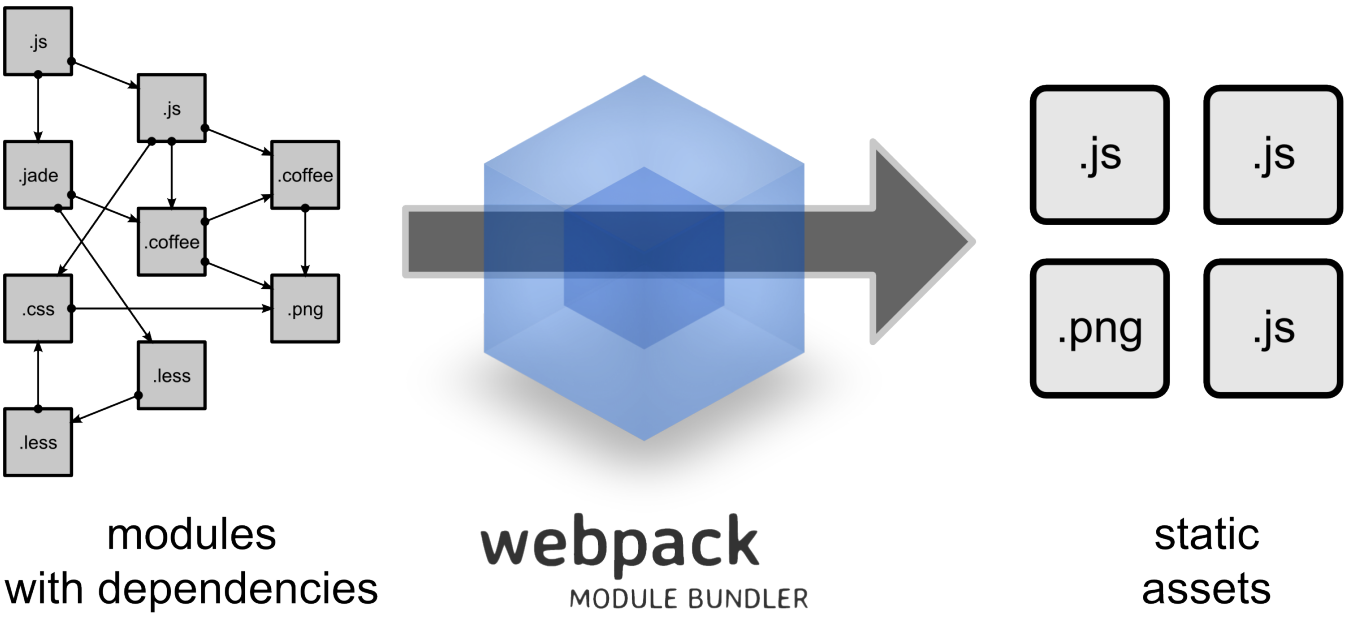
创建 es6模块化/userComponent2.js

```
import user from "./userApi2.js"
user.getList()
user.save()
```

# 一、什么是Webpack

Webpack 是一个前端资源加载/打包工具。它将根据模块的依赖关系进行静态分析，然后将这些模块按照指定的规则生成对应的静态资源。

从图中我们可以看出，Webpack 可以将多种静态资源 js、css、less 转换成一个静态文件，减少了页面的请求。




# 二、Webpack安装

## 1、全局安装

```
npm install -g webpack webpack-cli
```

## 2、安装后查看版本号

```
webpack -v
```

# 三、初始化项目

## 1、创建webpack文件夹

进入webpack目录，执行命令

```
npm init -y
```

## 2、创建src文件夹

## 3、src下创建common.js

```
exports.info = function (str) {
    document.write(str);
}
```

## 4、src下创建utils.js

```
exports.add = function (a, b) {
    return a + b;
}
```

## 5、src下创建main.js

```
const common = require('./common');
const utils = require('./utils');

common.info('Hello world!' + utils.add(100, 200));
```

# 四、JS打包

## 1、webpack目录下创建配置文件webpack.config.js

以下配置的意思是：读取当前项目目录下src文件夹中的main.js（入口文件）内容，分析资源依赖，把相关的js文件打包，打包后的文件放入当前目录的dist文件夹下，打包后的js文件名为bundle.js

```
const path = require("path"); //Node.js内置模块
module.exports = {
    entry: './src/main.js', //配置入口文件
    output: {
        path: path.resolve(__dirname, './dist'), //输出路径，__dirname：当前文件所在路径
        filename: 'bundle.js' //输出文件
    }
}
```

## 2、命令行执行编译命令

```
webpack #有黄色警告
webpack --mode=development #没有警告
#执行后查看bundle.js 里面包含了上面两个js文件的内容并惊醒了代码压缩
```

也可以配置项目的npm运行命令，修改package.json文件

```
"scripts": {
    //...,
    "dev": "webpack --mode=development"
 }
```

运行npm命令执行打包

```
npm run dev
```

## 3、webpack目录下创建index.html

引用bundle.js

```
<body>
    <script src="dist/bundle.js"></script>
</body>
```

## 4、浏览器中查看index.html

# 五、CSS打包

## 1、安装style-loader和 css-loader

Webpack 本身只能处理 JavaScript 模块，如果要处理其他类型的文件，就需要使用 loader 进行转换。

Loader 可以理解为是模块和资源的转换器。

首先我们需要安装相关Loader插件，css-loader 是将 css 装载到 javascript；style-loader 是让 javascript 认识css

```
npm install --save-dev style-loader css-loader
```

## 2、修改webpack.config.js

```
const path = require("path"); //Node.js内置模块
module.exports = {
    //...,
    output:{},
    module: {
        rules: [
            {
                test: /\.css$/,    //打包规则应用到以css结尾的文件上
                use: ['style-loader', 'css-loader']
            }
```

```
        ]
    }
}
```

## 3、在src文件夹创建style.css

```
body{
    background:pink;
}
```

## 4、修改main.js

在第一行引入style.css

```
require('./style.css');
```

## 5、浏览器中查看index.html

看看背景是不是变成粉色啦?

# 一、vue-element-admin

## 1、简介

而vue-element-admin是基于element-ui 的一套后台管理系统集成方案。

**功能：**https://panjiachen.github.io/vue-element-admin-site/zh/guide/#功能

**GitHub**地址：https://github.com/PanJiaChen/vue-element-admin

项目在线预览：https://panjiachen.gitee.io/vue-element-admin

## 2、安装

```
# 解压压缩包
# 进入目录
cd vue-element-admin-master
# 安装依赖
npm install
# 启动。执行后，浏览器自动弹出并访问http://localhost:9527/
npm run dev
```

# 二、vue-admin-template

## 1、简介

vueAdmin-template是基于vue-element-admin的一套后台管理系统基础模板（最少精简版），可作为模板进行二次开发。

**GitHub**地址：https://github.com/PanJiaChen/vue-admin-template

**建议：**你可以在 vue-admin-template 的基础上进行二次开发，把 vue-element-admin当做工具箱，想要什么功能或者组件就去 vue-element-admin 那里复制过来。

## 2、安装

```
```

```
# 解压压缩包
# 进入目录
cd vue-admin-template-master
# 安装依赖
npm install
# 启动。执行后，浏览器自动弹出并访问http://localhost:9528/
npm run dev
```

# 一、项目的创建和基本配置

## 1、创建项目

将vue-admin-template-master重命名为guli-admin

## 2、修改项目信息

package.json

```
{
    "name": "guli-admin",
    ......
    "description": "谷粒学院后台管理系统",
    "author": "Helen <55317332@qq.com>",
    ......
}
```

## 3、如果需要修改端口号

config/index.js中修改

```
port: 9528
```

## 4、项目的目录结构

```
.
├── build // 构建脚本
├── config // 全局配置
├── node_modules // 项目依赖模块
├── src //项目源代码
├── static // 静态资源
└── package.jspon // 项目信息和依赖配置
```

```
src
├── api // 各种接口
├── assets // 图片等资源
├── components // 各种公共组件，非公共组件在各自view下维护
├── icons //svg icon
├── router // 路由表
├── store // 存储
├── styles // 各种样式
├── utils // 公共工具，非公共工具，在各自view下维护
├── views // 各种layout
├── App.vue //***项目顶层组件***
├── main.js //***项目入口文件***
└── permission.js //认证入口
```

## 5、运行项目

```
npm run dev
```

# 二、登录页修改

src/views/login/index.vue（登录组件）

4行

```
<h3 class="title">谷粒学院后台管理系统</h3>
```

28行

```
<el-button :loading="loading" type="primary" style="width:100%;"
@click.native.prevent="handleLogin">
    登录
</el-button>
```

# 三、页面零星修改

## 1、标题

index.html（项目的html入口）

```
<title>谷粒学院后台管理系统</title>
```

修改后热部署功能，浏览器自动刷新

## 2、国际化设置

打开 src/main.js（项目的js入口），第7行，修改语言为 zh-CN，使用中文语言环境，例如：日期时间组件

```
import locale from 'element-ui/lib/locale/lang/zh-CN' // lang i18n
```

## 3、icon

复制 favicon.ico 到根目录

## 4、导航栏文字

src/views/layout/components（当前项目的布局组件）

src/views/layout/components/Navbar.vue

13行

```
<el-dropdown-item>
    首页
</el-dropdown-item>
```

17行

```
<span style="display:block;" @click="logout">退出</span>
```

## 5、面包屑文字

src/components（可以在很多项目中复用的通用组件）

src/components/Breadcrumb/index.vue

38行

```
meta: { title: '首页' }
```

# 四、Eslint语法规范型检查

## 1、ESLint简介

JavaScript 是一个动态的弱类型语言，在开发中比较容易出错。因为没有编译程序，为了寻找 JavaScript 代码错误通常需要在执行过程中不断调适。

ESLint 是一个语法规则和代码风格的检查工具，可以用来保证写出语法正确、风格统一的代码。让程序员在编码的过程中发现问题而不是在执行的过程中。

## 2、语法规则

ESLint 内置了一些规则，也可以在使用过程中自定义规则。

本项目的语法规则包括：两个字符缩进，必须使用单引号，不能使用双引号，语句后不可以写分号，代码段之间必须有一个空行等。

## 3、确认开启语法检查

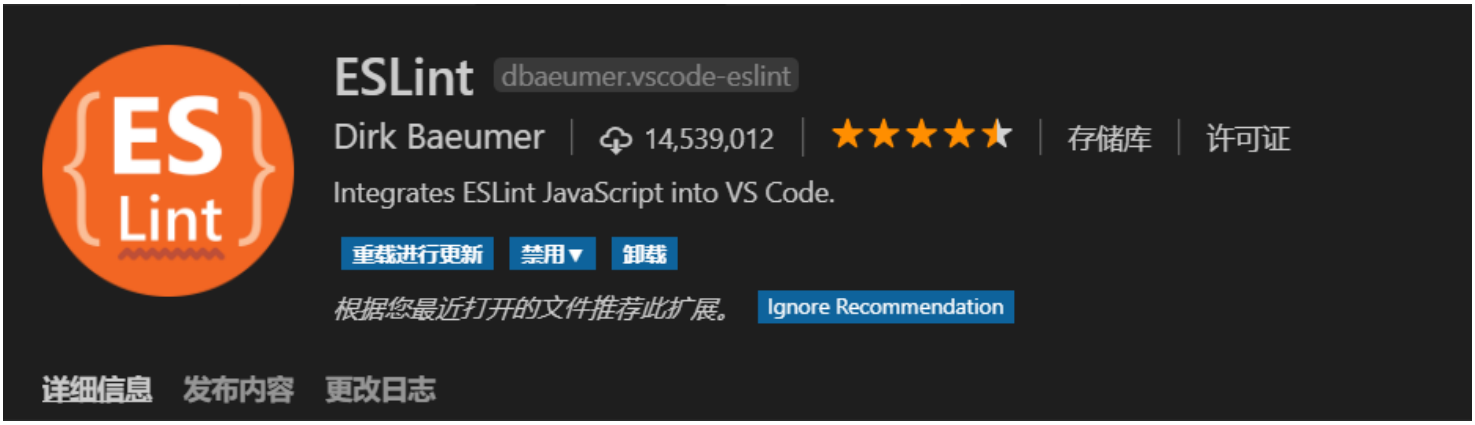
打开 config/index.js，配置是否开启语法检查

```
useEslint: true,
```

可以关闭语法检查，建议开启，养成良好的编程习惯。

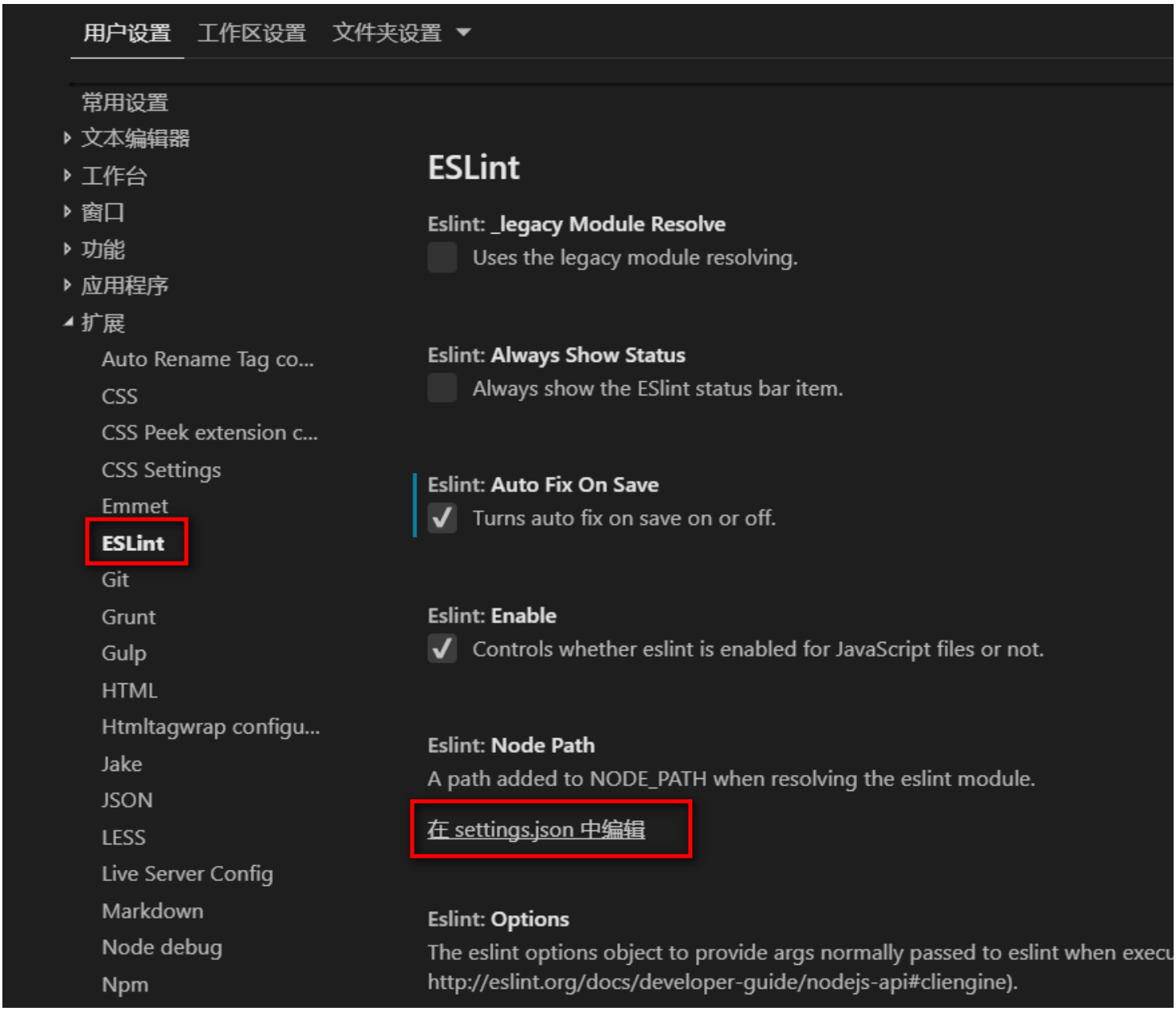
## 4、ESLint插件安装

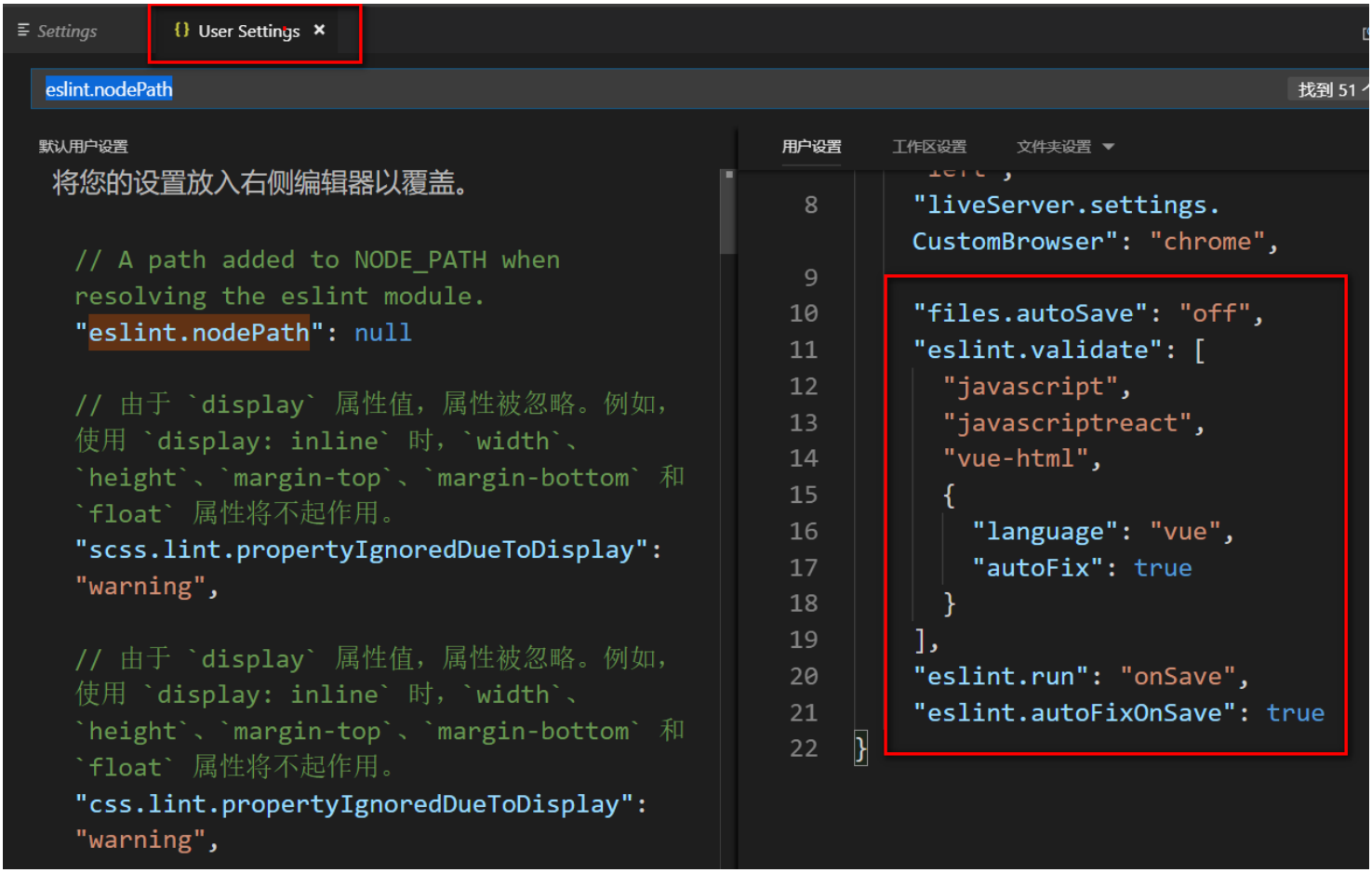
vs code的ESLint插件，能帮助我们自动整理代码格式




## 5、插件的扩展设置

选择vs code左下角的"设置"， 打开 VSCode 配置文件,添加如下配置






```
"files.autoSave": "off",
"eslint.validate": [
  "javascript",
  "javascriptreact",
  "vue-html",
  {
    "language": "vue",
    "autoFix": true
  }
],
"eslint.run": "onSave",
"eslint.autoFixOnSave": true
```

# 一、项目的创建和基本配置

## 1、创建项目

将vue-admin-template-master重命名为guli-admin

## 2、修改项目信息

package.json

```
{
    "name": "guli-admin",
    ......
    "description": "谷粒学院后台管理系统",
    "author": "Helen <55317332@qq.com>",
    ......
}
```

## 3、如果需要修改端口号

config/index.js中修改

```
port: 9528
```

## 4、项目的目录结构

```
.
├── build // 构建脚本
├── config // 全局配置
├── node_modules // 项目依赖模块
├── src //项目源代码
├── static // 静态资源
└── package.jspon // 项目信息和依赖配置
```

```
src
├── api // 各种接口
├── assets // 图片等资源
├── components // 各种公共组件，非公共组件在各自view下维护
├── icons //svg icon
├── router // 路由表
├── store // 存储
├── styles // 各种样式
├── utils // 公共工具，非公共工具，在各自view下维护
├── views // 各种layout
├── App.vue //***项目顶层组件***
├── main.js //***项目入口文件***
└── permission.js //认证入口
```

## 5、运行项目

```
npm run dev
```

# 二、登录页修改

src/views/login/index.vue（登录组件）

4行

```
<h3 class="title">谷粒学院后台管理系统</h3>
```

28行

```
<el-button :loading="loading" type="primary" style="width:100%;"
@click.native.prevent="handleLogin">
    登录
</el-button>
```

# 三、页面零星修改

## 1、标题

index.html（项目的html入口）

```
<title>谷粒学院后台管理系统</title>
```

修改后热部署功能，浏览器自动刷新

## 2、国际化设置

打开 src/main.js（项目的js入口），第7行，修改语言为 zh-CN，使用中文语言环境，例如：日期时间组件

```
import locale from 'element-ui/lib/locale/lang/zh-CN' // lang i18n
```

## 3、icon

复制 favicon.ico 到根目录

## 4、导航栏文字

src/views/layout/components（当前项目的布局组件）

src/views/layout/components/Navbar.vue

13行

```
<el-dropdown-item>
    首页
</el-dropdown-item>
```

17行

```
<span style="display:block;" @click="logout">退出</span>
```

## 5、面包屑文字

src/components（可以在很多项目中复用的通用组件）

src/components/Breadcrumb/index.vue

38行

```
meta: { title: '首页' }
```

# 四、Eslint语法规范型检查

## 1、ESLint简介

JavaScript 是一个动态的弱类型语言，在开发中比较容易出错。因为没有编译程序，为了寻找 JavaScript 代码错误通常需要在执行过程中不断调适。

ESLint 是一个语法规则和代码风格的检查工具，可以用来保证写出语法正确、风格统一的代码。让程序员在编码的过程中发现问题而不是在执行的过程中。

## 2、语法规则

ESLint 内置了一些规则，也可以在使用过程中自定义规则。

本项目的语法规则包括：两个字符缩进，必须使用单引号，不能使用双引号，语句后不可以写分号，代码段之间必须有一个空行等。

## 3、确认开启语法检查

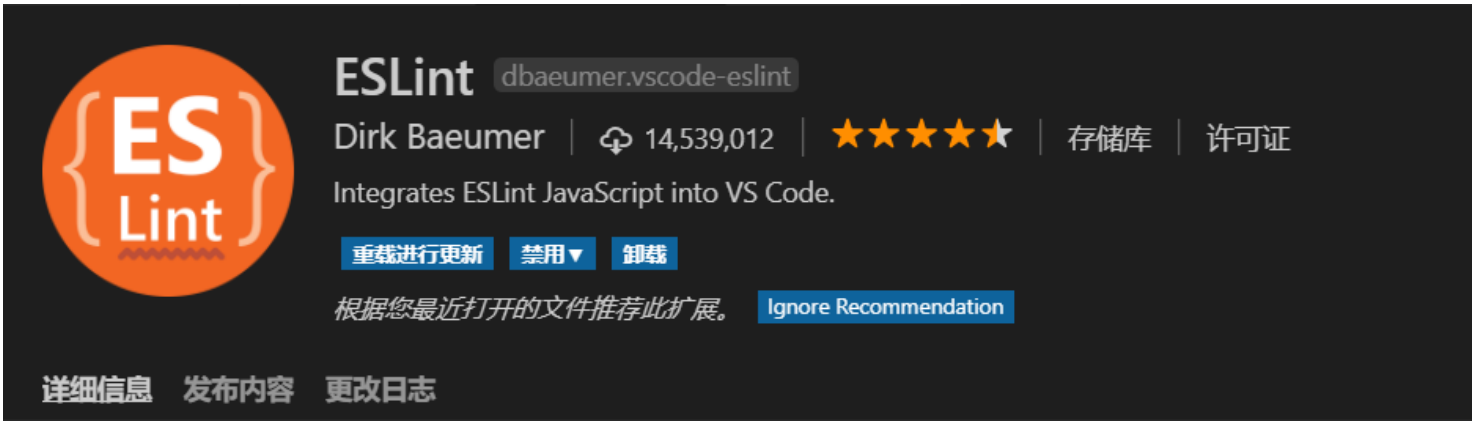
打开 config/index.js，配置是否开启语法检查

```
useEslint: true,
```

可以关闭语法检查，建议开启，养成良好的编程习惯。

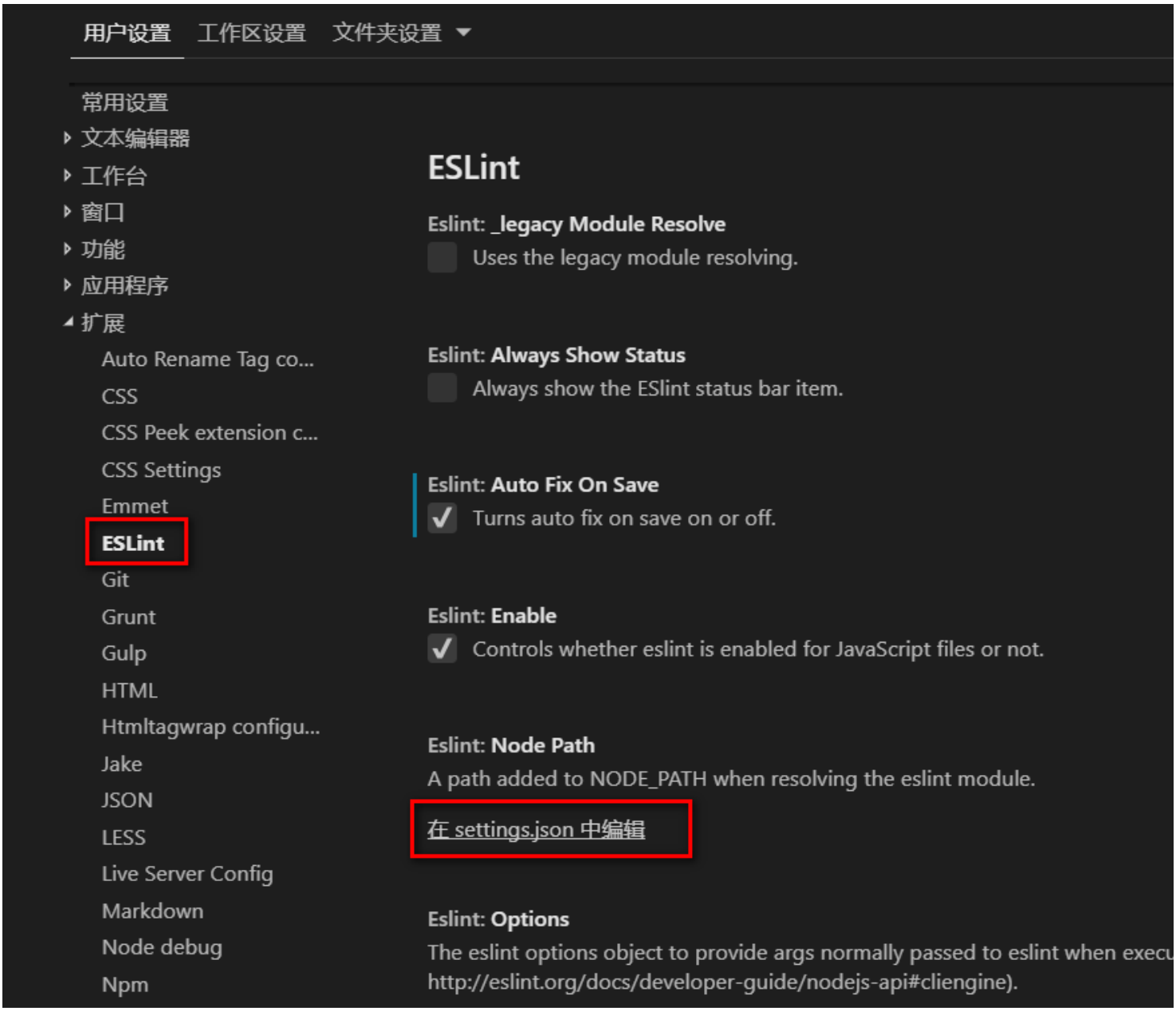
## 4、ESLint插件安装

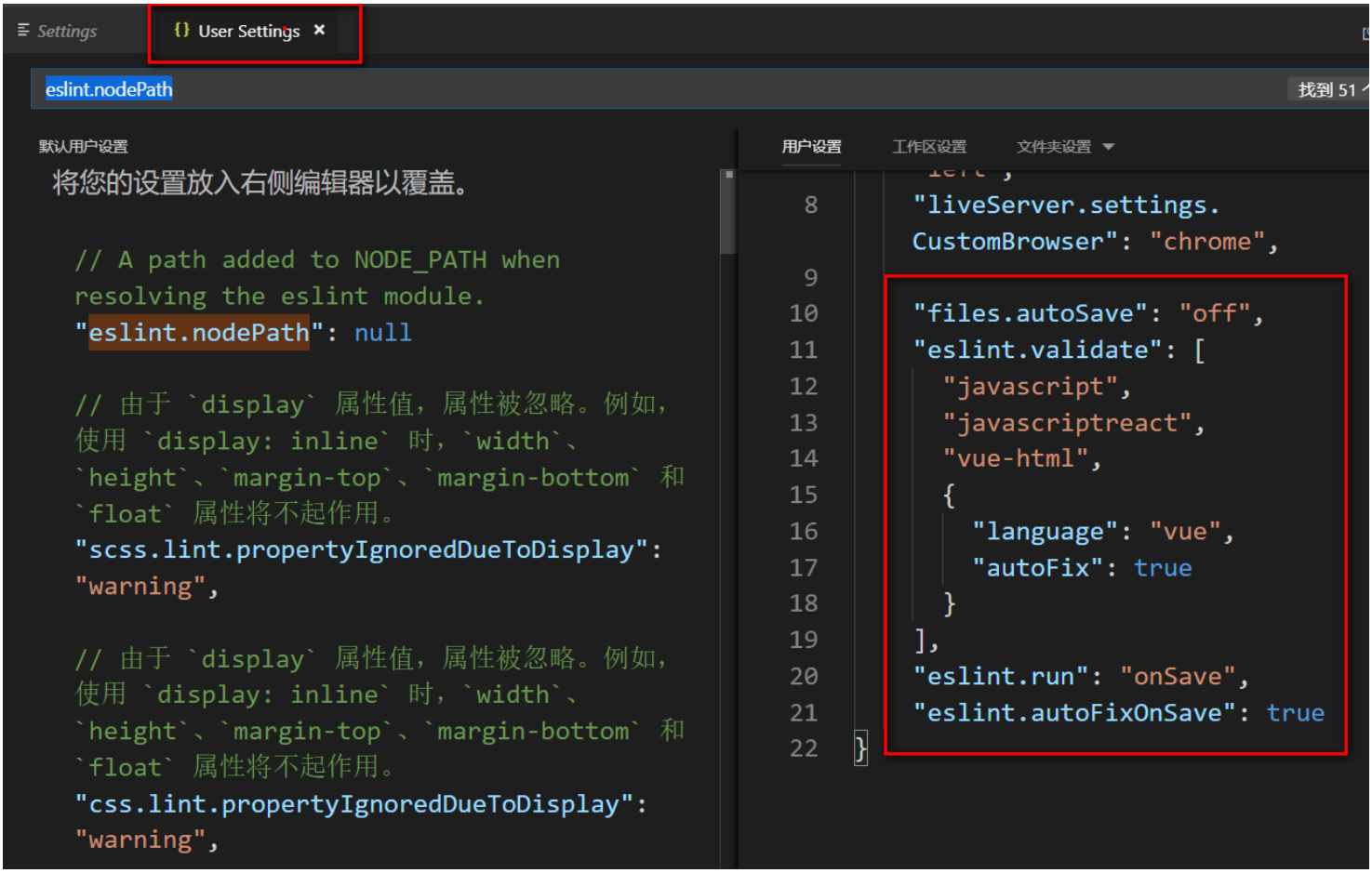
vs code的ESLint插件，能帮助我们自动整理代码格式




## 5、插件的扩展设置

选择vs code左下角的"设置"， 打开 VSCode 配置文件,添加如下配置






```
"files.autoSave": "off",
"eslint.validate": [
  "javascript",
  "javascriptreact",
  "vue-html",
  {
    "language": "vue",
    "autoFix": true
  }
],
"eslint.run": "onSave",
"eslint.autoFixOnSave": true
```

# 一、后台系统路由实现分析

## 1、入口文件中调用路由

src/main.js

```
......
import router from './router' //引入路由模块
......
new Vue({
  el: '#app',
  router, //挂载路由
  store,
  render: h => h(App)
})
```

## 2、路由模块中定义路由

src/router/index.js

```
......
export const constantRouterMap = [
  ......
]
export default new Router({
  ......
  routes: constantRouterMap
})
```

# 二、谷粒学院路由定义

## 1、复制icon图标

将vue-element-admin/src/icons/svg 中的图标复制到 guli-admin项目中

# 2、修改路由

修改 src/router/index.js 文件，重新定义constantRouterMap

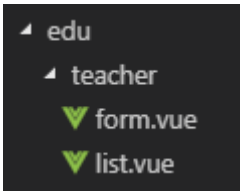
**注意：**每个路由的name不能相同

```
 export const constantRouterMap = [
  { path: '/login', component: () => import('@/views/login/index'), hidden:
true },
  { path: '/404', component: () => import('@/views/404'), hidden: true },
  // 首页
  {
    path: '/',
    component: Layout,
    redirect: '/dashboard',
    name: 'Dashboard',
    children: [{
      path: 'dashboard',
      component: () => import('@/views/dashboard/index'),
      meta: { title: '谷粒学院后台首页', icon: 'dashboard' }
    }]
  },
  // 讲师管理
  {
    path: '/edu/teacher',
    component: Layout,
    redirect: '/edu/teacher/list',
    name: 'Teacher',
    meta: { title: '讲师管理', icon: 'peoples' },
    children: [
      {
        path: 'list',
        name: 'EduTeacherList',
        component: () => import('@/views/edu/teacher/list'),
        meta: { title: '讲师列表' }
      },
      {
        path: 'create',
        name: 'EduTeacherCreate',
        component: () => import('@/views/edu/teacher/form'),
        meta: { title: '添加讲师' }
      },
      {
        path: 'edit/:id',
```

```
        name: 'EduTeacherEdit',
        component: () => import('@/views/edu/teacher/form'),
        meta: { title: '编辑讲师', noCache: true },
        hidden: true
      }
    ]
  },
  { path: '*', redirect: '/404', hidden: true }
]
```

## 3、创建vue组件

在src/views文件夹下创建以下文件夹和文件




## 4、form.vue

```
<template>
  <div class="app-container">
    讲师表单
  </div>
</template>
```

## 5、list.vue

```
<template>
  <div class="app-container">
    讲师列表
  </div>
</template>
```

# 一、项目中的Easy Mock

config/dev.env.js 中BASE_API 为项目的easymock地址，目前具有模拟登录、登出、获取用户信息的功能

```
BASE_API: '"https://easy-mock.com/mock/5950a2419adc231f356a6636/vue-admin"',
```

**登录：**/user/login

**获取用户信息：**/user/info?token=admin

**登出：**/user/logout

config/dev.env.js，只有一个api地址的配置位置，而我们实际的后端有很多微服务，所以接口地址有很多，

我们可以使用nginx反向代理让不同的api路径分发到不同的api服务器中

# 二、配置nginx反向代理

## 1、安装window版的nginx

将nginx-1.12.0.zip解压到开发目录中

如：E:\development\nginx-1.12.0-guli-api

双击nginx.exe运行nginx

访问：localhost

## 2、配置nginx代理

在Nginx中配置对应的微服务服务器地址即可

注意：最好修改默认的 80端口到81

```
http {
    server {
        listen       81;
        ......
    }
```

```
    ......
    server {
        listen 8201;
        server_name localhost;
        location ~ /edu/ {
             proxy_pass http://localhost:8101;
        }

        location ~ /user/ {
             rewrite /(.+)$ /mock/5950a2419adc231f356a6636/vue-admin/$1
 break;
             proxy_pass https://www.easy-mock.com;
        }
    }
}
```

## 3、重启nginx

```
nginx -s reload
```

## 4、测试

访问讲师列表接口：http://localhost:8201/admin/edu/teacher

访问获取用户信息接口：http://localhost:8201/user/info?token=admin

# 三、配置开发环境

## 1、修改config/dev.env.js

```
BASE_API: '"http://127.0.0.1:8201"'
```

## 2、重启前端程序

修改配置文件后，需要手动重启前端程序

# 一、分页列表

## 1、定义api

创建文件 src/api/edu/teacher.js

```
import request from '@/utils/request'
const api_name = '/admin/edu/teacher'
export default {
  getPageList(page, limit, searchObj) {
    return request({
      url: `${api_name}/${page}/${limit}`,
      method: 'get',
      data: searchObj
    })
  }
}
```

## 2、初始化vue组件

src/views/edu/teacher/list.vue

```
<template>
  <div class="app-container">
    讲师列表
  </div>
</template>
<script>
import teacher from '@/api/edu/teacher'
export default {
  data() {// 定义数据
      return {}
  },

  created() { // 当页面加载时获取数据
    this.fetchData()
  },

```

```
  methods: {
    fetchData() { // 调用api层获取数据库中的数据
      console.log('加载列表')
    }
  }
}
</script>
```

## 3、定义data

```
  data() {
    return {
      listLoading: true, // 是否显示loading信息
      list: null, // 数据列表
      total: 0, // 总记录数
      page: 1, // 页码
      limit: 10, // 每页记录数
      searchObj: {}// 查询条件
    }
  },
```

## 4、定义methods

```
  methods: {
    fetchData(page = 1) { // 调用api层获取数据库中的数据
      console.log('加载列表')
      this.page = page
      this.listLoading = true
      teacher.getPageList(this.page, this.limit, this.searchObj).then(response
=> {
        // debugger 设置断点调试
        if (response.success === true) {
          this.list = response.data.rows
          this.total = response.data.total
        }
        this.listLoading = false
      })
    }
  }
```

## 5、表格渲染

```
    <!-- 表格 -->
    <el-table
      v-loading="listLoading"
      :data="list"
      element-loading-text="数据加载中"
      border
      fit
      highlight-current-row>
      <el-table-column
        label="序号"
        width="70"
        align="center">
        <template slot-scope="scope">
          {{ (page - 1) * limit + scope.$index + 1 }}
        </template>
      </el-table-column>
      <el-table-column prop="name" label="名称" width="80" />
      <el-table-column label="头衔" width="80">
        <template slot-scope="scope">
          {{ scope.row.level===1?'高级讲师':'首席讲师' }}
        </template>
      </el-table-column>
      <el-table-column prop="intro" label="资历" />
      <el-table-column prop="gmtCreate" label="添加时间" width="160"/>
      <el-table-column prop="sort" label="排序" width="60" />
      <el-table-column label="操作" width="200" align="center">
        <template slot-scope="scope">
          <router-link :to="'/edu/teacher/edit/'+scope.row.id">
            <el-button type="primary" size="mini" icon="el-icon-edit">修改</el-
button>
          </router-link>
          <el-button type="danger" size="mini" icon="el-icon-delete"
@click="removeDataById(scope.row.id)">删除</el-button>
        </template>
      </el-table-column>
    </el-table>
```

# 6、分页组件

```
<!-- 分页 -->
 <el-pagination
   :current-page="page"
   :page-size="limit"
   :total="total"
   style="padding: 30px 0; text-align: center;"
   layout="total, prev, pager, next, jumper"
   @current-change="fetchData"
 />
```

# 7、顶部查询表单

**注意:**

element-ui的 date-picker组件默认绑定的时间值是默认世界标准时间，和中国时间差8小时

设置 value-format="yyyy-MM-dd HH:mm:ss" 改变绑定的值

```
<!--查询表单-->
<el-form :inline="true" class="demo-form-inline">
  <el-form-item>
    <el-input v-model="searchObj.name" placeholder="讲师名"/>
  </el-form-item>
  <el-form-item>
    <el-select v-model="searchObj.level" clearable placeholder="讲师头衔">
      <el-option :value="1" label="高级讲师"/>
      <el-option :value="2" label="首席讲师"/>
    </el-select>
  </el-form-item>
  <el-form-item label="添加时间">
    <el-date-picker
      v-model="searchObj.begin"
      type="datetime"
      placeholder="选择开始时间"
      value-format="yyyy-MM-dd HH:mm:ss"
      default-time="00:00:00"
    />
  </el-form-item>
  <el-form-item>
    <el-date-picker
```

```
          v-model="searchObj.end"
          type="datetime"
          placeholder="选择截止时间"
          value-format="yyyy-MM-dd HH:mm:ss"
          default-time="00:00:00"
        />
      </el-form-item>
      <el-button type="primary" icon="el-icon-search" @click="fetchData()">查询</el-button>
      <el-button type="default" @click="resetData()">清空</el-button>
    </el-form>
```

清空方法

```
resetData() {
    this.searchObj = {}
    this.fetchData()
}
```

## 8、测试

# 二、删除

## 1、定义api

src/api/edu/teacher.js

```
removeById(teacherId) {
    return request({
        url: `${api_name}/${teacherId}`,
        method: 'delete'
    })
}
```

## 2、定义methods

src/views/edu/teacher/list.vue

使用MessageBox 弹框组件

```
removeDataById(id) {
    // debugger
    // console.log(memberId)
    this.$confirm('此操作将永久删除该记录, 是否继续?', '提示', {
        confirmButtonText: '确定',
        cancelButtonText: '取消',
        type: 'warning'
    }).then(() => {
        return teacher.removeById(id)
    }).then(() => {
        this.fetchData()
        this.$message({
            type: 'success',
            message: '删除成功!'
        })
    }).catch((response) => { // 失败
        if (response === 'cancel') {
            this.$message({
                type: 'info',
                message: '已取消删除'
            })
        } else {
            this.$message({
                type: 'error',
                message: '删除失败'
            })
        }
    })
}
```

# 一、新增

## 1、定义api

src/api/edu/teacher.js

```
save(teacher) {
    return request({
        url: api_name,
        method: 'post',
        data: teacher
    })
}
```

## 2、初始化组件

src/views/edu/teacher/form.vue

html

```
<template>
  <div class="app-container">
    <el-form label-width="120px">
      <el-form-item label="讲师名称">
        <el-input v-model="teacher.name"/>
      </el-form-item>
      <el-form-item label="讲师排序">
        <el-input-number v-model="teacher.sort" controls-position="right"
min="0"/>
      </el-form-item>
      <el-form-item label="讲师头衔">
        <el-select v-model="teacher.level" clearable placeholder="请选择">
          <!--
            数据类型一定要和取出的json中的一致，否则没法回填
            因此，这里value使用动态绑定的值，保证其数据类型是number
          -->
          <el-option :value="1" label="高级讲师"/>
          <el-option :value="2" label="首席讲师"/>
        </el-select>
```

```
      </el-form-item>
      <el-form-item label="讲师资历">
        <el-input v-model="teacher.career"/>
      </el-form-item>
      <el-form-item label="讲师简介">
        <el-input v-model="teacher.intro" :rows="10" type="textarea"/>
      </el-form-item>
      <!-- 讲师头像：TODO -->
      <el-form-item>
        <el-button :disabled="saveBtnDisabled" type="primary"
@click="saveOrUpdate">保存</el-button>
      </el-form-item>
    </el-form>
  </div>
</template>
```

js

```
<script>
export default {
  data() {
    return {
      teacher: {
        name: '',
        sort: 0,
        level: 1,
        career: '',
        intro: '',
        avatar: ''
      },
      saveBtnDisabled: false // 保存按钮是否禁用,
    }
  },
  methods: {
    saveOrUpdate() {
      this.saveBtnDisabled = true
      this.saveData()
    },
    // 保存
    saveData() {
    }
  }
}
```

```
</script>
```

## 3、实现新增功能

引入teacher api模块

```
import teacher from '@/api/edu/teacher'
```

完善save方法

```
    // 保存
    saveData() {
      teacher.save(this.teacher).then(response => {
        return this.$message({
          type: 'success',
          message: '保存成功!'
        })
      }).then(resposne => {
        this.$router.push({ path: '/edu/teacher' })
      }).catch((response) => {
        // console.log(response)
        this.$message({
          type: 'error',
          message: '保存失败'
        })
      })
    }
```

# 二、回显

## 1、定义api

src/api/edu/teacher.js

```
getById(id) {
    return request({
        url: `${api_name}/${id}`,
```

```
        method: 'get'
    })
}
```

## 2、组件中调用api

methods中定义fetchDataById

```
// 根据id查询记录
fetchDataById(id) {
    teacher.getById(id).then(response => {
        this.teacher = response.data.item
    }).catch((response) => {
        this.$message({
            type: 'error',
            message: '获取数据失败'
        })
    })
}
```

## 3、页面渲染前调用fetchDataById

```
  created() {
    console.log('created')
    if (this.$route.params && this.$route.params.id) {
      const id = this.$route.params.id
      this.fetchDataById(id)
    }
  }
```

# 三、更新

## 1、定义api

```
updateById(teacher) {
```

```
    return request({
        url: `${api_name}/${teacher.id}`,
        method: 'put',
        data: teacher
    })
}
```

## 2、组件中调用api

methods中定义updateData

```
// 根据id更新记录
updateData() {
    this.saveBtnDisabled = true
    teacher.updateById(this.teacher).then(response => {
        return this.$message({
            type: 'success',
            message: '修改成功!'
        })
    }).then(resposne => {
        this.$router.push({ path: '/edu/teacher' })
    }).catch((response) => {
        // console.log(response)
        this.$message({
            type: 'error',
            message: '保存失败'
        })
    })
}
```

## 3、完善saveOrUpdate方法

```
saveOrUpdate() {
    this.saveBtnDisabled = true
    if (!this.teacher.id) {
        this.saveData()
    } else {
        this.updateData()
    }
```

```
}
```

# 四、存在问题

vue-router导航切换 时，如果两个路由都渲染同个组件，组件会重（chong）用，

组件的生命周期钩子（created）不会再被调用, 使得组件的一些数据无法根据 path的改变得到更新

因此：

1、我们可以在watch中监听路由的变化，当路由变化时，重新调用created中的内容

2、在init方法中我们判断路由的变化，如果是修改路由，则从api获取表单数据，

如果是新增路由，则重新初始化表单数据

```
<script>
import teacher from '@/api/edu/teacher'
const defaultForm = {
  name: '',
  sort: 0,
  level: '',
  career: '',
  intro: '',
  avatar: ''
}
export default {
  data() {
    return {
      teacher: defaultForm,
      saveBtnDisabled: false // 保存按钮是否禁用,
    }
  },
  watch: {
    $route(to, from) {
      console.log('watch $route')
      this.init()
    }
  },
  created() {
    console.log('created')
    this.init()
  },
  methods: {
    init() {
```

```
      if (this.$route.params && this.$route.params.id) {
        const id = this.$route.params.id
        this.fetchDataById(id)
      } else {
        // 使用对象拓展运算符，拷贝对象，而不是引用，
        // 否则新增一条记录后，defaultForm就变成了之前新增的teacher的值
        this.teacher = { ...defaultForm }
      }
    },
    ......
  }
}
</script>
```

# 一、返回操作是否成功

## 1、删除业务逻辑

TeacherServiceImpl

```
@Override
public boolean removeById(Serializable id) {
    Integer result = baseMapper.deleteById(id);
    return null != result && result > 0;
}
```

TeacherAdminController

```
@ApiOperation(value = "根据ID删除讲师")
@DeleteMapping("{id}")
public R removeById(
    @ApiParam(name = "id", value = "讲师ID", required = true)
    @PathVariable String id){
    boolean result = teacherService.removeById(id);
    if(result){
        return R.ok();
    }else{
        return R.error().message("删除失败");
    }
}
```

## 2、前端

在catch中处理错误信息

# 一、对象存储OSS

为了解决海量数据存储与弹性扩容，项目中我们采用云存储的解决方案- 阿里云OSS。

## 1、开通“对象存储OSS”服务

（1）申请阿里云账号

（2）实名认证

（3）开通“对象存储OSS”服务

（4）进入管理控制台

## 2、创建Bucket

选择：标准存储、公共读、不开通


# 3、上传默认头像

创建文件夹avatar，上传默认的用户头像

## 4、创建RAM子用户


# 二、使用SDK


## 1、创建Mavaen项目

com.atguigu

aliyun-oss

## 2、pom

```xml
<dependencies>
    <!--aliyunOSS-->
    <dependency>
        <groupId>com.aliyun.oss</groupId>
        <artifactId>aliyun-sdk-oss</artifactId>
        <version>2.8.3</version>
    </dependency>

    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
    </dependency>
</dependencies>
```

## 3、找到编码时需要用到的常量值

（1）endpoint

（2）bucketName

（3）accessKeyId

（4）accessKeySecret

## 4、测试创建Bucket的连接


```java
package com.atguigu.oss;

```

```
public class OSSTest {

    // Endpoint以杭州为例，其它Region请按实际情况填写。
    String endpoint = "your endpoint";
    // 阿里云主账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM账号进行API访问
号。
    String accessKeyId = "your accessKeyId";
    String accessKeySecret = "your accessKeySecret";
    String bucketName = "guli-file";

    @Test
    public void testCreateBucket() {

        // 创建OSSClient实例。
        OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

        // 创建存储空间。
        ossClient.createBucket(bucketName);

        // 关闭OSSClient。
        ossClient.shutdown();
    }
}
```

## 5、判断存储空间是否存在


```
@Test
public void testExist() {
```

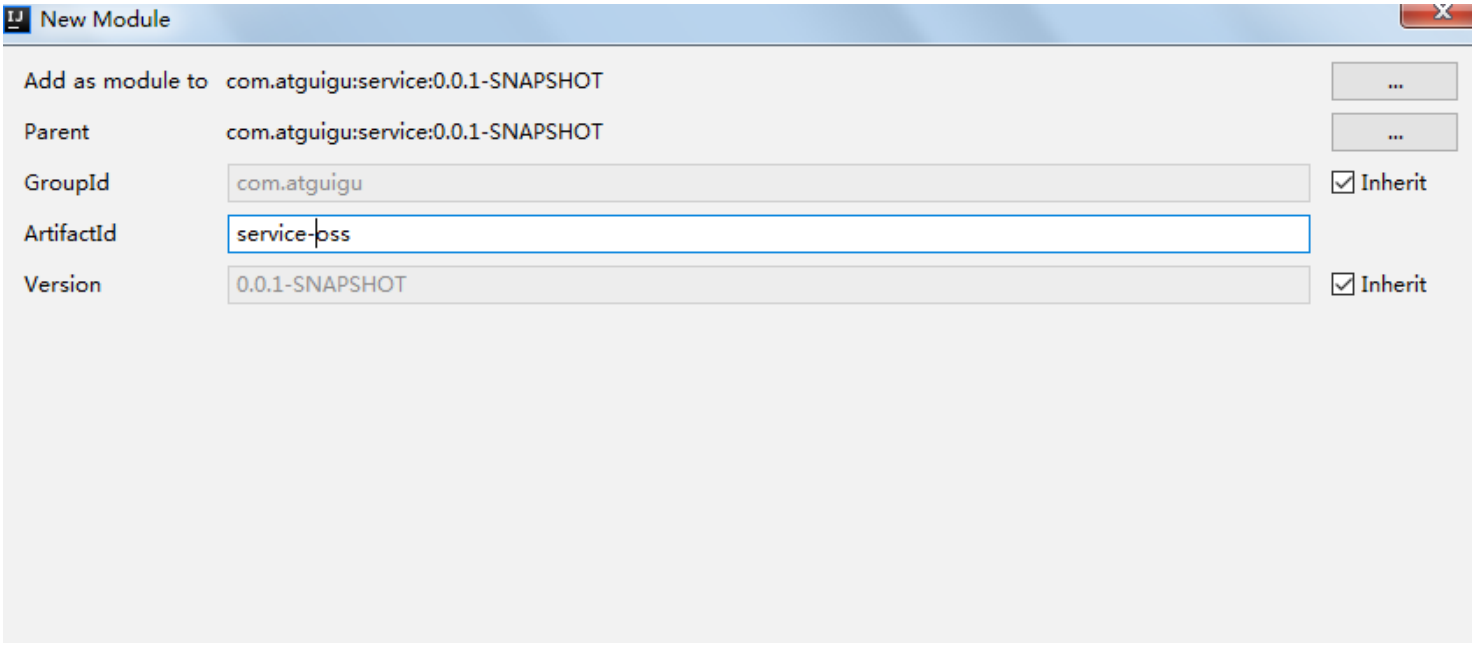
```
    // 创建OSSClient实例。
    OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

    boolean exists = ossClient.doesBucketExist(bucketName);
    System.out.println(exists);

    // 关闭OSSClient。
    ossClient.shutdown();
}
```

## 6、设置存储空间的访问权限

```
@Test
public void testAccessControl() {

    // 创建OSSClient实例。
    OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

    // 设置存储空间的访问权限为：公共读。
    ossClient.setBucketAcl(bucketName, CannedAccessControlList.PublicRead);

    // 关闭OSSClient。
    ossClient.shutdown();
}
```

# 一、新建云存储微服务

## 1、在service模块下创建子模块service-oss




## 2、配置pom.xml

service-oss上级模块service已经引入service的公共依赖，所以service-oss模块只需引入阿里云oss相关依赖即可，

service父模块已经引入了service-base模块，所以Swagger相关默认已经引入

```
<dependencies>
    <!-- 阿里云oss依赖 -->
    <dependency>
        <groupId>com.aliyun.oss</groupId>
        <artifactId>aliyun-sdk-oss</artifactId>
    </dependency>

    <!-- 日期工具栏依赖 -->
    <dependency>
        <groupId>joda-time</groupId>
        <artifactId>joda-time</artifactId>
    </dependency>
```

```
</dependencies>
```

# 3、配置application.properties

```
#服务端口
server.port=8002
#服务名
spring.application.name=service-oss

#环境设置：dev、test、prod
spring.profiles.active=dev

#阿里云 OSS
#不同的服务器，地址不同
aliyun.oss.file.endpoint=your endpoint
aliyun.oss.file.keyid=your accessKeyId
aliyun.oss.file.keysecret=your accessKeySecret
#bucket可以在控制台创建，也可以使用java代码创建
aliyun.oss.file.bucketname=guli-file
```

# 4、logback-spring.xml

# 5、创建启动类

创建OssApplication.java

```
package com.guli.oss;
@SpringBootApplication
@ComponentScan({"com.atguigu"})
public class OssApplication {

    public static void main(String[] args) {
        SpringApplication.run(OssApplication.class, args);
    }
}
```

# 6、启动项目

报错

```
***************************
APPLICATION FAILED TO START
***************************

Description:

Failed to configure a DataSource: 'url' attribute is not specified and no embedded datasource could be configured.

Reason: Failed to determine a suitable driver class


Action:

Consider the following:
    If you want an embedded database (H2, HSQL or Derby), please put it on the classpath.
    If you have database settings to be loaded from a particular profile you may need to activate it (no profiles are
```

spring boot 会默认加载org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration这个类，

而DataSourceAutoConfiguration类使用了@Configuration注解向spring注入了dataSource bean，又因为项目（oss模块）中并没有关于dataSource相关的配置信息，所以当spring创建dataSource bean时因缺少相关的信息就会报错。

解决办法：

方法1、在@SpringBootApplication注解上加上exclude，解除自动加载DataSourceAutoConfiguration

```
@SpringBootApplication(exclude = DataSourceAutoConfiguration.class)
```

# 二、实现文件上传

## 1、从配置文件读取常量

创建常量读取工具类：ConstantPropertiesUtil.java

使用@Value读取application.properties里的配置内容

用spring的 InitializingBean 的 afterPropertiesSet 来初始化配置信息，这个方法将在所有的属性被初始化后调用。

```java
/**
 * 常量类，读取配置文件application.properties中的配置
 */
@Component
//@PropertySource("classpath:application.properties")
public class ConstantPropertiesUtil implements InitializingBean {

    @Value("${aliyun.oss.file.endpoint}")
    private String endpoint;

    @Value("${aliyun.oss.file.keyid}")
    private String keyId;

    @Value("${aliyun.oss.file.keysecret}")
    private String keySecret;

    @Value("${aliyun.oss.file.filehost}")
    private String fileHost;

    @Value("${aliyun.oss.file.bucketname}")
    private String bucketName;

    public static String END_POINT;
    public static String ACCESS_KEY_ID;
    public static String ACCESS_KEY_SECRET;
    public static String BUCKET_NAME;
    public static String FILE_HOST ;

    @Override
    public void afterPropertiesSet() throws Exception {
        END_POINT = endpoint;
        ACCESS_KEY_ID = keyId;
        ACCESS_KEY_SECRET = keySecret;
        BUCKET_NAME = bucketName;
        FILE_HOST = fileHost;
    }
}
```

## 2、文件上传

创建Service接口：FileService.java

```java
public interface FileService {

    /**
     * 文件上传至阿里云
     * @param file
     * @return
     */
    String upload(MultipartFile file);
}
```

## 实现：FileServiceImpl.java

参考SDK中的：Java->上传文件->简单上传->流式上传->上传文件流

- 上传文件流

  以下代码用于上传文件流：

```java
// Endpoint以杭州为例，其它Region请按实际情况填写。
String endpoint = "http://oss-cn-hangzhou.aliyuncs.com";
// 云账号AccessKey有所有API访问权限，建议遵循阿里云安全最佳实践，创建并使用RAM子账号进行API访问或日常运维，
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";

// 创建OSSClient实例。
OSSClient ossClient = new OSSClient(endpoint, accessKeyId, accessKeySecret);

// 上传文件流。
InputStream inputStream = new FileInputStream("<yourlocalFile>");
ossClient.putObject("<yourBucketName>", "<yourObjectName>", inputStream);

// 关闭OSSClient。
ossClient.shutdown();
```

```java
public class FileServiceImpl implements FileService {

    @Override
    public String upload(MultipartFile file) {

        //获取阿里云存储相关常量
        String endPoint = ConstantPropertiesUtil.END_POINT;
        String accessKeyId = ConstantPropertiesUtil.ACCESS_KEY_ID;
```

```
        String accessKeySecret = ConstantPropertiesUtil.ACCESS_KEY_SECRET;
        String bucketName = ConstantPropertiesUtil.BUCKET_NAME;
        String fileHost = ConstantPropertiesUtil.FILE_HOST;

        String uploadUrl = null;

        try {
            //判断oss实例是否存在：如果不存在则创建，如果存在则获取
            OSSClient ossClient = new OSSClient(endPoint, accessKeyId,
accessKeySecret);
            if (!ossClient.doesBucketExist(bucketName)) {
                //创建bucket
                ossClient.createBucket(bucketName);
                //设置oss实例的访问权限：公共读
                ossClient.setBucketAcl(bucketName, CannedAccessControlList.PublicRead);
            }

            //获取上传文件流
            InputStream inputStream = file.getInputStream();

            //构建日期路径：avatar/2019/02/26/文件名
            String filePath = new DateTime().toString("yyyy/MM/dd");

            //文件名：uuid.扩展名
            String original = file.getOriginalFilename();
            String fileName = UUID.randomUUID().toString();
            String fileType = original.substring(original.lastIndexOf("."));
            String newName = fileName + fileType;
            String fileUrl = fileHost + "/" + filePath + "/" + newName;

            //文件上传至阿里云
            ossClient.putObject(bucketName, fileUrl, inputStream);

            // 关闭OSSClient。
            ossClient.shutdown();

            //获取url地址
            uploadUrl = "http://" + bucketName + "." + endPoint + "/" + fileUrl;

        } catch (IOException e) {
            throw new GuliException(ResultCodeEnum.FILE_UPLOAD_ERROR);
        }

```

```
        return uploadUrl;
    }
}
```

# 3、控制层

创建controller：FileUploadController.java

```
package com.guli.oss.controller;

@Api(description="阿里云文件管理")
@CrossOrigin //跨域
@RestController
@RequestMapping("/admin/oss/file")
public class FileController {

    @Autowired
    private FileService fileService;

    /**
     * 文件上传
     *
     * @param file
     */
    @ApiOperation(value = "文件上传")
    @PostMapping("upload")
    public R upload(
            @ApiParam(name = "file", value = "文件", required = true)
            @RequestParam("file") MultipartFile file) {

        String uploadUrl = fileService.upload(file);
        //返回r对象
        return R.ok().message("文件上传成功").data("url", uploadUrl);

    }
}
```

# 4、重启oss服务

# 5、Swagger中测试文件上传

# 6、配置nginx反向代理

将接口地址加入nginx配置

```
location ~ /eduoss/ {
    proxy_pass http://localhost:8001;
}
```

# 一、前端整合图片上传组件

## 1、复制头像上传组件

**从vue-element-admin复制组件：**

vue-element-admin/src/components/ImageCropper

vue-element-admin/src/components/PanThumb

## 2、前端参考实现

src/views/components-demo/avatarUpload.vue

## 3、前端添加文件上传组件

**src/views/edu/teacher/form.vue**

template：

```
<!-- 讲师头像 -->
<el-form-item label="讲师头像">
    <!-- 头衔缩略图 -->
    <pan-thumb :image="teacher.avatar"/>
    <!-- 文件上传按钮 -->
    <el-button type="primary" icon="el-icon-upload"
@click="imagecropperShow=true">更换头像
    </el-button>
    <!--
v-show：是否显示上传组件
:key：类似于id，如果一个页面多个图片上传控件，可以做区分
:url：后台上传的url地址
@close：关闭上传组件
@crop-upload-success：上传成功后的回调 -->
    <image-cropper
                    v-show="imagecropperShow"
                    :width="300"
                    :height="300"
                    :key="imagecropperKey"
                    :url="BASE_API+'/admin/oss/file/upload'"
                    field="file"
                    @close="close"
```

```
                    @crop-upload-success="cropSuccess"/>
</el-form-item>
```

引入组件模块

```
import ImageCropper from '@/components/ImageCropper'
import PanThumb from '@/components/PanThumb'
```

# 4、设置默认头像

config/dev.env.js中添加阿里云oss bucket地址

```
OSS_PATH: '"https://guli-file.oss-cn-beijing.aliyuncs.com"'
```

组件中初始化头像默认地址

```
const defaultForm = {
  ......,
  avatar: process.env.OSS_PATH + '/avatar/default.jpg'
}
```

# 5、js脚本实现上传和图片回显

```
export default {
  components: { ImageCropper, PanThumb },
  data() {
    return {
      //其它数据模型
      ......,

      BASE_API: process.env.BASE_API, // 接口API地址
      imagecropperShow: false, // 是否显示上传组件
      imagecropperKey: 0 // 上传组件id
    }
  },
```

```
13
14    ......,
15
16    methods: {
17      //其他函数
18      ......,
19      // 上传成功后的回调函数
20      cropSuccess(data) {
21        console.log(data)
22        this.imagecropperShow = false
23        this.teacher.avatar = data.url
24        // 上传成功后，重新打开上传组件时初始化组件，否则显示上一次的上传结果
25        this.imagecropperKey = this.imagecropperKey + 1
26      },
27      // 关闭上传组件
28      close() {
29        this.imagecropperShow = false
30        // 上传失败后，重新打开上传组件时初始化组件，否则显示上一次的上传结果
31        this.imagecropperKey = this.imagecropperKey + 1
32      }
33    }
34  }
35
36
```

# 二、测试文件上传

前后端联调

# 一、Excel导入导出的应用场景

1、数据导入：减轻录入工作量

2、数据导出：统计信息归档

3、数据传输：异构系统之间数据传输

# 二、EasyExcel简介

## 1、EasyExcel特点

- Java领域解析、生成Excel比较有名的框架有Apache poi、jxl等。但他们都存在一个严重的问题就是非常的耗内存。如果你的系统并发量不大的话可能还行，但是一旦并发上来后一定会OOM或者JVM频繁的full gc。
- EasyExcel是阿里巴巴开源的一个excel处理框架，**以使用简单、节省内存著称**。EasyExcel能大大减少占用内存的主要原因是在解析Excel时没有将文件数据一次性全部加载到内存中，而是从磁盘上一行行读取数据，逐个解析。
- EasyExcel采用一行一行的解析模式，并将一行的解析结果以观察者的模式通知处理（AnalysisEventListener）。

# 一、创建项目，实现EasyExcel对Excel写操作

## 1、创建一个普通的maven项目

项目名：excel-easydemo

## 2、pom中引入xml相关依赖

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/com.alibaba/easyexcel -->
    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>easyexcel</artifactId>
        <version>2.1.1</version>
    </dependency>
</dependencies>
```

## 3、创建实体类

### 设置表头和添加的数据字段

```
import com.alibaba.excel.annotation.ExcelProperty;

//设置表头和添加的数据字段
public class DemoData {
    //设置表头名称
    @ExcelProperty("学生编号")
    private int sno;

    //设置表头名称
    @ExcelProperty("学生姓名")
    private String sname;

    public int getSno() {
        return sno;
    }
```

```
16
17      public void setSno(int sno) {
18          this.sno = sno;
19      }
20
21      public String getSname() {
22          return sname;
23      }
24
25      public void setSname(String sname) {
26          this.sname = sname;
27      }
28
29      @Override
30      public String toString() {
31          return "DemoData{" +
32                  "sno=" + sno +
33                  ", sname='" + sname + '\'' +
34                  '}';
35      }
36  }
```

## 4、实现写操作

（1）创建方法循环设置要添加到**Excel**的数据

```
1  //循环设置要添加的数据，最终封装到list集合中
2  private static List<DemoData> data() {
3      List<DemoData> list = new ArrayList<DemoData>();
4      for (int i = 0; i < 10; i++) {
5          DemoData data = new DemoData();
6          data.setSno(i);
7          data.setSname("张三"+i);
8          list.add(data);
9      }
10     return list;
11 }
```

（2）实现最终的添加操作（写法一）

```
public static void main(String[] args) throws Exception {
    // 写法1
    String fileName = "F:\\11.xlsx";
    // 这里 需要指定写用哪个class去写，然后写到第一个sheet，名字为模板 然后文件流会自动关闭
    // 如果这里想使用03 则 传入excelType参数即可
    EasyExcel.write(fileName, DemoData.class).sheet("写入方法一").doWrite(data());
}
```

## （3）实现最终的添加操作（写法二）

```
public static void main(String[] args) throws Exception {
    // 写法2，方法二需要手动关闭流
    String fileName = "F:\\112.xlsx";
    // 这里 需要指定写用哪个class去写
    ExcelWriter excelWriter = EasyExcel.write(fileName, DemoData.class).build();
    WriteSheet writeSheet = EasyExcel.writerSheet("写入方法二").build();
    excelWriter.write(data(), writeSheet);
    /// 千万别忘记finish 会帮忙关闭流
    excelWriter.finish();
}
```

# 一、实现EasyExcel对Excel读操作

## 1、创建实体类

```java
import com.alibaba.excel.annotation.ExcelProperty;
public class ReadData {
    //设置列对应的属性
    @ExcelProperty(index = 0)
    private int sid;

    //设置列对应的属性
    @ExcelProperty(index = 1)
    private String sname;

    public int getSid() {
        return sid;
    }
    public void setSid(int sid) {
        this.sid = sid;
    }
    public String getSname() {
        return sname;
    }
    public void setSname(String sname) {
        this.sname = sname;
    }
    @Override
    public String toString() {
        return "ReadData{" +
                "sid=" + sid +
                ", sname='" + sname + '\'' +
                '}';
    }
}
```

## 2、创建读取操作的监听器

```java
import com.alibaba.excel.context.AnalysisContext;
import com.alibaba.excel.event.AnalysisEventListener;
import com.alibaba.excel.exception.ExcelDataConvertException;
import com.sun.scenario.effect.impl.sw.sse.SSEBlend_SRC_OUTPeer;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

//创建读取excel监听器
public class ExcelListener extends AnalysisEventListener<ReadData> {

    //创建list集合封装最终的数据
    List<ReadData> list = new ArrayList<ReadData>();

    //一行一行去读取excle内容
    @Override
    public void invoke(ReadData user, AnalysisContext analysisContext) {
       System.out.println("***"+user);
        list.add(user);
    }

    //读取excel表头信息
    @Override
    public void invokeHeadMap(Map<Integer, String> headMap, AnalysisContext
context) {
        System.out.println("表头信息: "+headMap);
    }

    //读取完成后执行
    @Override
    public void doAfterAllAnalysed(AnalysisContext analysisContext) {
    }
}
```

## 3、调用实现最终的读取

```java
   public static void main(String[] args) throws Exception {

        // 写法1:
```

```
        String fileName = "F:\\01.xlsx";
        // 这里 需要指定读用哪个class去读，然后读取第一个sheet 文件流会自动关闭
        EasyExcel.read(fileName, ReadData.class, new
ExcelListener()).sheet().doRead();

        // 写法2:
        InputStream in = new BufferedInputStream(new
FileInputStream("F:\\01.xlsx"));
        ExcelReader excelReader = EasyExcel.read(in, ReadData.class, new
ExcelListener()).build();
        ReadSheet readSheet = EasyExcel.readSheet(0).build();
        excelReader.read(readSheet);
        // 这里千万别忘记关闭，读的时候会创建临时文件，到时磁盘会崩的
        excelReader.finish();
}
```

# 一、Excel模板

## 1、编辑Excel模板

## 2、将文件上传至阿里云OSS

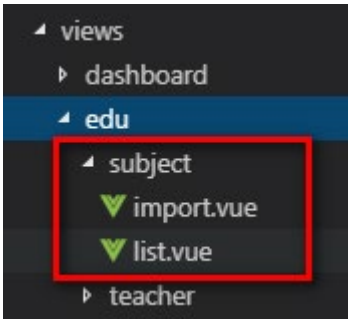

# 二、配置路由

## 1、添加路由

```
// 课程分类管理
{
  path: '/edu/subject',
  component: Layout,
  redirect: '/edu/subject/list',
  name: 'Subject',
  meta: { title: '课程分类管理', icon: 'nested' },
  children: [
    {
      path: 'list',
      name: 'EduSubjectList',
      component: () => import('@/views/edu/subject/list'),
      meta: { title: '课程分类列表' }
    },
    {
      path: 'import',
      name: 'EduSubjectImport',
```

```
      component: () => import('@/views/edu/subject/import'),
      meta: { title: '导入课程分类' }
    }
  ]
},
```

## 2、添加vue组件




# 三、表单组件import.vue

## 1、js定义数据

```
<script>
export default {

  data() {
    return {
      BASE_API: process.env.BASE_API, // 接口API地址
      OSS_PATH: process.env.OSS_PATH, // 阿里云OSS地址
      fileUploadBtnText: '上传到服务器', // 按钮文字
      importBtnDisabled: false, // 按钮是否禁用,
      loading: false
    }
  }
}
</script>
```

## 2、template

```
<template>
  <div class="app-container">
    <el-form label-width="120px">
      <el-form-item label="信息描述">
        <el-tag type="info">excel模版说明</el-tag>
        <el-tag>
          <i class="el-icon-download"/>
          <a :href="OSS_PATH +
'/excel/%E8%AF%BE%E7%A8%8B%E5%88%86%E7%B1%BB%E5%88%97%E8%A1%A8%E6%A8%A1%E6%9D%BF.x
ls'">点击下载模版</a>
        </el-tag>

      </el-form-item>

      <el-form-item label="选择Excel">
        <el-upload
          ref="upload"
          :auto-upload="false"
          :on-success="fileUploadSuccess"
          :on-error="fileUploadError"
          :disabled="importBtnDisabled"
          :limit="1"
          :action="BASE_API+'/admin/edu/subject/import'"
          name="file"
          accept="application/vnd.ms-excel">
          <el-button slot="trigger" size="small" type="primary">选取文件</el-
button>
          <el-button
            :loading="loading"
            style="margin-left: 10px;"
            size="small"
            type="success"
            @click="submitUpload">{{ fileUploadBtnText }}</el-button>
        </el-upload>
      </el-form-item>
    </el-form>
  </div>
</template>
```

## 3、js上传方法

```
methods: {
    submitUpload() {
      this.fileUploadBtnText = '正在上传'
      this.importBtnDisabled = true
      this.loading = true
      this.$refs.upload.submit()
    },

    fileUploadSuccess(response) {

    },

    fileUploadError(response) {

    }
}
```

## 4、回调函数

```
fileUploadSuccess(response) {
    if (response.success === true) {
    this.fileUploadBtnText = '导入成功'
    this.loading = false
    this.$message({
        type: 'success',
        message: response.message
    })
    }
},

fileUploadError(response) {
    this.fileUploadBtnText = '导入失败'
    this.loading = false
    this.$message({
    type: 'error',
    message: '导入失败'
    })
}
```

# 一、添加依赖

## 1、service-edu模块配置依赖

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/com.alibaba/easyexcel -->
    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>easyexcel</artifactId>
        <version>2.1.1</version>
    </dependency>
</dependencies>

```

# 二、业务处理

# 1、SubjectAdminController

```
package com.guli.edu.controller.admin;
@Api(description="课程分类管理")
@CrossOrigin //跨域
@RestController
@RequestMapping("/eduservice/subject")
public class SubjectAdminController {

    @Autowired
    private SubjectService subjectService;

    //添加课程分类
    @ApiOperation(value = "Excel批量导入")
    @PostMapping("addSubject")
    public R addSubject(MultipartFile file) {
        //1 获取上传的excel文件 MultipartFile
        //返回错误提示信息
        subjectService.importSubjectData(file,subjectService);
        //判断返回集合是否为空
```

```
        return R.ok();
    }
}
```

## 2、创建和Excel对应的实体类

```
import com.alibaba.excel.annotation.ExcelProperty;
import lombok.Data;

@Data
public class ExcelSubjectData {
    @ExcelProperty(index = 0)
    private int oneSubjectName;

    @ExcelProperty(index = 1)
    private String twoSubjectName;
}
```

## 3、SubjectService

（1）接口

```
void batchImport(MultipartFile file);
```

（2）实现类

```
//添加课程分类
//poi读取excel内容
@Override
public void importSubjectData(MultipartFile file,EduSubjectService subjectService)
{
    try {
        //1 获取文件输入流
        InputStream inputStream = file.getInputStream();

        // 这里 需要指定读用哪个class去读，然后读取第一个sheet 文件流会自动关闭
        EasyExcel.read(inputStream, ExcelSubjectData.class, new
```

```
SubjectExcelListener(subjectService)).sheet().doRead();
    }catch(Exception e) {
        e.printStackTrace();
        throw new GuliException(20002,"添加课程分类失败");
    }
}
```

# 4、创建读取Excel监听器

```
import com.alibaba.excel.context.AnalysisContext;
import com.alibaba.excel.event.AnalysisEventListener;
import com.atguigu.eduservice.entity.EduSubject;
import com.atguigu.eduservice.entity.vo.ExcelSubjectData;
import com.atguigu.eduservice.service.EduSubjectService;
import com.atguigu.servicebase.handler.GuliException;
import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

public class SubjectExcelListener extends AnalysisEventListener<ExcelSubjectData>
{

    public EduSubjectService subjectService;

    public SubjectExcelListener() {}
    //创建有参数构造，传递subjectService用于操作数据库
    public SubjectExcelListener(EduSubjectService subjectService) {
        this.subjectService = subjectService;
    }

    //一行一行去读取excle内容
    @Override
    public void invoke(ExcelSubjectData user, AnalysisContext analysisContext) {
        if(user == null) {
            throw new GuliException(20001,"添加失败");
        }
        //添加一级分类
        EduSubject existOneSubject =
this.existOneSubject(subjectService,user.getOneSubjectName());
```

```
        if(existOneSubject == null) {//没有相同的
            existOneSubject = new EduSubject();
            existOneSubject.setTitle(user.getOneSubjectName());
            existOneSubject.setParentId("0");
            subjectService.save(existOneSubject);
        }

        //获取一级分类id值
        String pid = existOneSubject.getId();

        //添加二级分类
        EduSubject existTwoSubject =
this.existTwoSubject(subjectService,user.getTwoSubjectName(), pid);
        if(existTwoSubject == null) {
            existTwoSubject = new EduSubject();
            existTwoSubject.setTitle(user.getTwoSubjectName());
            existTwoSubject.setParentId(pid);
            subjectService.save(existTwoSubject);
        }
    }

    //读取excel表头信息
    @Override
    public void invokeHeadMap(Map<Integer, String> headMap, AnalysisContext
context) {
        System.out.println("表头信息: "+headMap);
    }

    //读取完成后执行
    @Override
    public void doAfterAllAnalysed(AnalysisContext analysisContext) {}

    //判断一级分类是否重复
    private EduSubject existTwoSubject(EduSubjectService subjectService,String
name,String pid) {
        QueryWrapper<EduSubject> wrapper = new QueryWrapper<>();
        wrapper.eq("title",name);
        wrapper.eq("parent_id",pid);
        EduSubject eduSubject = subjectService.getOne(wrapper);
        return eduSubject;
    }

    //判断一级分类是否重复
```

```
71      private EduSubject existOneSubject(EduSubjectService subjectService,String
name) {
72          QueryWrapper<EduSubject> wrapper = new QueryWrapper<>();
73          wrapper.eq("title",name);
74          wrapper.eq("parent_id","0");
75          EduSubject eduSubject = subjectService.getOne(wrapper);
76          return eduSubject;
77      }
78  }
```

# 一、前端实现

## 1、参考 views/tree/index.vue

## 2、创建api

api/edu/subject.js

```
import request from '@/utils/request'

const api_name = '/admin/edu/subject'

export default {

  getNestedTreeList() {
    return request({
      url: `${api_name}`,
      method: 'get'
    })
  }
}
```

## 3、list.vue

```
<template>
  <div class="app-container">
    <el-input v-model="filterText" placeholder="Filter keyword" style="margin-bottom:30px;"
/>

    <el-tree
      ref="subjectTree"
      :data="subjectList"
      :props="defaultProps"
      :filter-node-method="filterNode"
      class="filter-tree"
```

```
      default-expand-all
    />

  </div>
</template>

<script>
import subject from '@/api/edu/subject'
export default {

  data() {
    return {
      filterText: '',
      subjectList: [],
      defaultProps: {
        children: 'children',
        label: 'title'
      }
    }
  },
  watch: {
    filterText(val) {
      this.$refs.subjectTree.filter(val)
    }
  },

  created() {
    this.fetchNodeList()
  },

  methods: {
    fetchNodeList() {
      subject.getNestedTreeList().then(response => {
        if (response.success === true) {
          this.subjectList = response.data.items
        }
      })
    },
    filterNode(value, data) {
      if (!value) return true
      return data.title.indexOf(value) !== -1
    }
  }
```

```
}
</script>
```

# 二、后端实现

## 1、创建vo

```
package com.guli.edu.vo;
@Data
public class SubjectVo {

    private String id;
    private String title;
}
```

```
package com.guli.edu.vo;
@Data
public class SubjectNestedVo {

    private String id;
    private String title;
    private List<SubjectVo> children = new ArrayList<>();
}
```

## 2、创建controller

```
@ApiOperation(value = "嵌套数据列表")
@GetMapping("")
public R nestedList(){

    List<SubjectNestedVo> subjectNestedVoList = subjectService.nestedList();
    return R.ok().data("items", subjectNestedVoList);
}
```

# 3、创建service

接口

```
List<SubjectNestedVo> nestedList();
```

实现Final

```
@Override
public List<SubjectNestedVo> nestedList() {

    //最终要的到的数据列表
    ArrayList<SubjectNestedVo> subjectNestedVoArrayList = new ArrayList<>();

    //获取一级分类数据记录
    QueryWrapper<Subject> queryWrapper = new QueryWrapper<>();
    queryWrapper.eq("parent_id", 0);
    queryWrapper.orderByAsc("sort", "id");
    List<Subject> subjects = baseMapper.selectList(queryWrapper);

    //获取二级分类数据记录
    QueryWrapper<Subject> queryWrapper2 = new QueryWrapper<>();
    queryWrapper2.ne("parent_id", 0);
    queryWrapper2.orderByAsc("sort", "id");
    List<Subject> subSubjects = baseMapper.selectList(queryWrapper2);

    //填充一级分类vo数据
    int count = subjects.size();
    for (int i = 0; i < count; i++) {
        Subject subject = subjects.get(i);

        //创建一级类别vo对象
        SubjectNestedVo subjectNestedVo = new SubjectNestedVo();
        BeanUtils.copyProperties(subject, subjectNestedVo);
        subjectNestedVoArrayList.add(subjectNestedVo);

        //填充二级分类vo数据
        ArrayList<SubjectVo> subjectVoArrayList = new ArrayList<>();
        int count2 = subSubjects.size();
```

```
        for (int j = 0; j < count2; j++) {

            Subject subSubject = subSubjects.get(j);
            if(subject.getId().equals(subSubject.getParentId())){

                //创建二级类别vo对象
                SubjectVo subjectVo = new SubjectVo();
                BeanUtils.copyProperties(subSubject, subjectVo);
                subjectVoArrayList.add(subjectVo);
            }
        }
        subjectNestedVo.setChildren(subjectVoArrayList);
    }


    return subjectNestedVoArrayList;
}
```

# 三、优化前端过滤功能

```
filterNode(value, data) {
    if (!value) return true
    return data.title.toLowerCase().indexOf(value.toLowerCase()) !== -1
}
```

# 一、需求



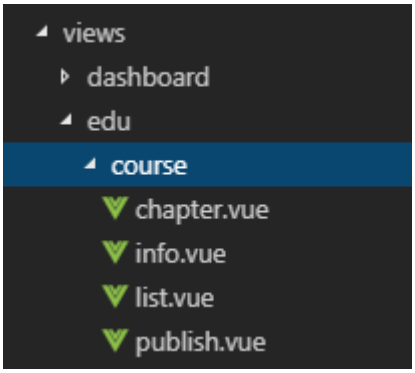

# 二、配置路由

## 1、添加路由

```
// 课程管理
{
  path: '/edu/course',
  component: Layout,
```

```
  redirect: '/edu/course/list',
  name: 'Course',
  meta: { title: '课程管理', icon: 'form' },
  children: [
    {
      path: 'list',
      name: 'EduCourseList',
      component: () => import('@/views/edu/course/list'),
      meta: { title: '课程列表' }
    },
    {
      path: 'info',
      name: 'EduCourseInfo',
      component: () => import('@/views/edu/course/info'),
      meta: { title: '发布课程' }
    },
    {
      path: 'info/:id',
      name: 'EduCourseInfoEdit',
      component: () => import('@/views/edu/course/info'),
      meta: { title: '编辑课程基本信息', noCache: true },
      hidden: true
    },
    {
      path: 'chapter/:id',
      name: 'EduCourseChapterEdit',
      component: () => import('@/views/edu/course/chapter'),
      meta: { title: '编辑课程大纲', noCache: true },
      hidden: true
    },
    {
      path: 'publish/:id',
      name: 'EduCoursePublishEdit',
      component: () => import('@/views/edu/course/publish'),
      meta: { title: '发布课程', noCache: true },
      hidden: true
    }
  ]
},
```

## 2、添加vue组件




# 三、整合步骤条组件

参考 http://element-cn.eleme.io/#/zh-CN/component/steps

## 1、课程信息页面

info.vue

```
<template>
  <div class="app-container">
    <h2 style="text-align: center;">发布新课程</h2>
    <el-steps :active="1" process-status="wait" align-center style="margin-bottom: 40px;">
      <el-step title="填写课程基本信息"/>
      <el-step title="创建课程大纲"/>
      <el-step title="提交审核"/>
    </el-steps>
    <el-form label-width="120px">
      <el-form-item>
        <el-button :disabled="saveBtnDisabled" type="primary" @click="next">保存并下一步</el-button>
      </el-form-item>
    </el-form>
  </div>
</template>
<script>
export default {
  data() {
    return {
      saveBtnDisabled: false // 保存按钮是否禁用
    }
  },
  created() {
```

```
    console.log('info created')
  },
  methods: {
    next() {
      console.log('next')
      this.$router.push({ path: '/edu/course/chapter/1' })
    }
  }
}
</script>
```

## 2、课程大纲页面

chapter.vue

```
<template>
  <div class="app-container">
    <h2 style="text-align: center;">发布新课程</h2>
    <el-steps :active="2" process-status="wait" align-center style="margin-bottom: 40px;">
      <el-step title="填写课程基本信息"/>
      <el-step title="创建课程大纲"/>
      <el-step title="提交审核"/>
    </el-steps>
    <el-form label-width="120px">
      <el-form-item>
        <el-button @click="previous">上一步</el-button>
        <el-button :disabled="saveBtnDisabled" type="primary" @click="next">下一步</el-button>
      </el-form-item>
    </el-form>
  </div>
</template>
<script>
export default {
  data() {
    return {
      saveBtnDisabled: false // 保存按钮是否禁用
    }
  },
  created() {
    console.log('chapter created')
```

```
  },
  methods: {
    previous() {
      console.log('previous')
      this.$router.push({ path: '/edu/course/info/1' })
    },
    next() {
      console.log('next')
      this.$router.push({ path: '/edu/course/publish/1' })
    }
  }
}
</script>
```

## 3、课程发布页面

publish.vue

```
<template>
  <div class="app-container">
    <h2 style="text-align: center;">发布新课程</h2>
    <el-steps :active="3" process-status="wait" align-center style="margin-bottom: 40px;">
      <el-step title="填写课程基本信息"/>
      <el-step title="创建课程大纲"/>
      <el-step title="提交审核"/>
    </el-steps>
    <el-form label-width="120px">
      <el-form-item>
        <el-button @click="previous">返回修改</el-button>
        <el-button :disabled="saveBtnDisabled" type="primary" @click="publish">发布课程</el-button>
      </el-form-item>
    </el-form>
  </div>
</template>
<script>
export default {
  data() {
    return {
      saveBtnDisabled: false // 保存按钮是否禁用
    }
```

```
  },
  created() {
    console.log('publish created')
  },
  methods: {
    previous() {
      console.log('previous')
      this.$router.push({ path: '/edu/course/chapter/1' })
    },
    publish() {
      console.log('publish')
      this.$router.push({ path: '/edu/course/list' })
    }
  }
}
</script>
```

# 一、后台api

## 1、定义form表单对象

CourseInfoForm.java

```
package com.guli.edu.form;

@ApiModel(value = "课程基本信息", description = "编辑课程基本信息的表单对象")
@Data
public class CourseInfoForm implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "课程ID")
    private String id;

    @ApiModelProperty(value = "课程讲师ID")
    private String teacherId;

    @ApiModelProperty(value = "课程专业ID")
    private String subjectId;

    @ApiModelProperty(value = "课程标题")
    private String title;

    @ApiModelProperty(value = "课程销售价格，设置为0则可免费观看")
    private BigDecimal price;

    @ApiModelProperty(value = "总课时")
    private Integer lessonNum;

    @ApiModelProperty(value = "课程封面图片路径")
    private String cover;

    @ApiModelProperty(value = "课程简介")
    private String description;
}
```

## 2、修改CourseDescription主键生成策略

```
@ApiModelProperty(value = "课程ID")
@TableId(value = "id", type = IdType.INPUT)
private String id;
```

## 3、定义常量

实体类Course.Java中定义

```
public static final String COURSE_DRAFT = "Draft";
public static final String COURSE_NORMAL = "Normal";
```

## 4、定义控制层接口

CourseAdminController.java

```
package com.guli.edu.controller.admin;

@Api(description="课程管理")
@CrossOrigin //跨域
@RestController
@RequestMapping("/admin/edu/course")
public class CourseAdminController {

    @Autowired
    private CourseService courseService;

    @ApiOperation(value = "新增课程")
    @PostMapping("save-course-info")
    public R saveCourseInfo(
            @ApiParam(name = "CourseInfoForm", value = "课程基本信息", required =
true)
            @RequestBody CourseInfoForm courseInfoForm){

        String courseId = courseService.saveCourseInfo(courseInfoForm);
```

```
        if(!StringUtils.isEmpty(courseId)){
            return R.ok().data("courseId", courseId);
        }else{
            return R.error().message("保存失败");
        }
    }
}
```

## 5、定义业务层方法

接口：CourseService.java

```
/**
     * 保存课程和课程详情信息
     * @param courseInfoForm
     * @return 新生成的课程id
     */
String saveCourseInfo(CourseInfoForm courseInfoForm);
```

实现：CourseServiceImpl.java

```
@Autowired
private CourseDescriptionService courseDescriptionService;

@Override
public String saveCourseInfo(CourseInfoForm courseInfoForm) {

    //保存课程基本信息
    Course course = new Course();
    course.setStatus(Course.COURSE_DRAFT);
    BeanUtils.copyProperties(courseInfoForm, course);
    boolean resultCourseInfo = this.save(course);
    if(!resultCourseInfo){
        throw new GuliException(20001, "课程信息保存失败");
    }

    //保存课程详情信息
    CourseDescription courseDescription = new CourseDescription();
    courseDescription.setDescription(courseInfoForm.getDescription());
```

```
    courseDescription.setId(course.getId());
    boolean resultDescription = courseDescriptionService.save(courseDescription);
    if(!resultDescription){
        throw new GuliException(20001, "课程详情信息保存失败");
    }

    return course.getId();
}
```

## 6、Swagger测试

# 二、前端实现

## 1、定义api

```
import request from '@/utils/request'

const api_name = '/admin/edu/course'

export default {
  saveCourseInfo(courseInfo) {
    return request({
      url: `${api_name}/save-course-info`,
      method: 'post',
      data: courseInfo
    })
  }
}
```

## 2、组件模板

```
<el-form label-width="120px">

  <el-form-item label="课程标题">
```

```
    <el-input v-model="courseInfo.title" placeholder=" 示例：机器学习项目课：从基础到搭建项目视
  </el-form-item>

  <!-- 所属分类 TODO -->

  <!-- 课程讲师 TODO -->

  <el-form-item label="总课时">
    <el-input-number :min="0" v-model="courseInfo.lessonNum" controls-position="right" plac
数"/>
  </el-form-item>

  <!-- 课程简介 TODO -->

  <!-- 课程封面 TODO -->

  <el-form-item label="课程价格">
    <el-input-number :min="0" v-model="courseInfo.price" controls-position="right" placehol
  </el-form-item>

  <el-form-item>
    <el-button :disabled="saveBtnDisabled" type="primary" @click="next">保存并下一步</el-but
  </el-form-item>
</el-form>
```

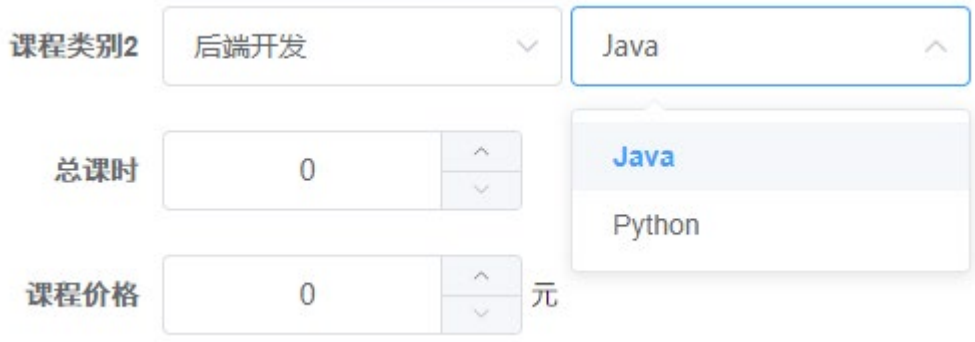
# 3、组件js

```
<script>
import course from '@/api/edu/course'

const defaultForm = {
  title: '',
  subjectId: '',
  teacherId: '',
  lessonNum: 0,
  description: '',
  cover: '',
  price: 0
}

```

```
export default {
  data() {
    return {
      courseInfo: defaultForm,
      saveBtnDisabled: false // 保存按钮是否禁用
    }
  },

  watch: {
    $route(to, from) {
      console.log('watch $route')
      this.init()
    }
  },

  created() {
    console.log('info created')
    this.init()
  },

  methods: {

    init() {
      if (this.$route.params && this.$route.params.id) {
        const id = this.$route.params.id
        console.log(id)
      } else {
        this.courseInfo = { ...defaultForm }
      }
    },

    next() {
      console.log('next')
      this.saveBtnDisabled = true
      if (!this.courseInfo.id) {
        this.saveData()
      } else {
        this.updateData()
      }
    },

    // 保存
    saveData() {
```

```
      course.saveCourseInfo(this.courseInfo).then(response => {
        this.$message({
          type: 'success',
          message: '保存成功!'
        })
        return response// 将响应结果传递给then
      }).then(response => {
        this.$router.push({ path: '/edu/course/chapter/' + response.data.courseId
})
      }).catch((response) => {
        this.$message({
          type: 'error',
          message: response.message
        })
      })
    },

    updateData() {
      this.$router.push({ path: '/edu/course/chapter/1' })
    }
  }
}
</script>
```

# 一、需求




# 二、获取一级分类

## 1、组件数据定义

定义在data中

```
subjectNestedList: [],//一级分类列表
subSubjectList: []//二级分类列表
```

## 2、组件模板

```
<!-- 所属分类：级联下拉列表 -->
<!-- 一级分类 -->
<el-form-item label="课程类别">
  <el-select
    v-model="courseInfo.subjectParentId"
    placeholder="请选择">
    <el-option
      v-for="subject in subjectNestedList"
      :key="subject.id"
      :label="subject.title"
      :value="subject.id"/>
  </el-select>
</el-form-item>
```

## 3、组件脚本

表单初始化时获取一级分类嵌套列表，引入subject api

```
import subject from '@/api/edu/subject'
```

定义方法

```
init() {
  ......
  // 初始化分类列表
  this.initSubjectList()
},
initSubjectList() {
  subject.getNestedTreeList().then(response => {
    this.subjectNestedList = response.data.items
  })
},
```

# 三、级联显示二级分类

## 1、组件模板

```
<!-- 二级分类 -->
<el-select v-model="courseInfo.subjectId" placeholder="请选择">
  <el-option
    v-for="subject in subSubjectList"
    :key="subject.value"
    :label="subject.title"
    :value="subject.id"/>
</el-select>
```

## 2、注册change事件

在一级分类的<el-select>组件中注册change事件

```
<el-select @change="subjectLevelOneChanged" ......
```

## 3、定义change事件方法

```
subjectLevelOneChanged(value) {
    console.log(value)
    for (let i = 0; i < this.subjectNestedList.length; i++) {
        if (this.subjectNestedList[i].id === value) {
            this.subSubjectList = this.subjectNestedList[i].children
            this.courseInfo.subjectId = ''
        }
    }
},
```

# 一、前端实现

## 1、组件模板

```
<!-- 课程讲师 -->
<el-form-item label="课程讲师">
  <el-select
    v-model="courseInfo.teacherId"
    placeholder="请选择">
    <el-option
      v-for="teacher in teacherList"
      :key="teacher.id"
      :label="teacher.name"
      :value="teacher.id"/>
  </el-select>
</el-form-item>
```

## 2、定义api

api/edu/teacher.js

```
getList() {
    return request({
        url: api_name,
        method: 'get'
    })
},
```

组件中引入teacher api

```
import teacher from '@/api/edu/teacher'
```

## 3、组件脚本

定义data

```
teacherList: [] // 讲师列表
```

表单初始化时获取讲师列表

```
init() {
  ......
  // 获取讲师列表
  this.initTeacherList()
},
initTeacherList() {
  teacher.getList().then(response => {
    this.teacherList = response.data.items
  })
},
```

# 一、Tinymce可视化编辑器

参考

https://panjiachen.gitee.io/vue-element-admin/#/components/tinymce

https://panjiachen.gitee.io/vue-element-admin/#/example/create

# 二、组件初始化

Tinymce是一个传统javascript插件，默认不能用于Vue.js因此需要做一些特殊的整合步骤

## 1、复制脚本库

将脚本库复制到项目的static目录下（在vue-element-admin-master的static路径下）

## 2、配置html变量

在 guli-admin/build/webpack.dev.conf.js 中添加配置

使在html页面中可是使用这里定义的BASE_URL变量

```
new HtmlWebpackPlugin({
    ......,
    templateParameters: {
        BASE_URL: config.dev.assetsPublicPath + config.dev.assetsSubDirectory
    }
})
```

## 3、引入js脚本

在guli-admin/index.html 中引入js脚本

```
<script src=<%= BASE_URL %>/tinymce4.7.5/tinymce.min.js></script>
<script src=<%= BASE_URL %>/tinymce4.7.5/langs/zh_CN.js></script>
```

# 三、组件引入

为了让Tinymce能用于Vue.js项目，vue-element-admin-master对Tinymce进行了封装，下面我们将它引入到我们的课程信息页面

## 1、复制组件

src/components/Tinymce

## 2、引入组件

课程信息组件中引入 Tinymce

```
import Tinymce from '@/components/Tinymce'
export default {
  components: { Tinymce },
  ......
}
```

## 3、组件模板

```
<!-- 课程简介-->
<el-form-item label="课程简介">
    <tinymce :height="300" v-model="courseInfo.description"/>
</el-form-item>
```

## 4、组件样式

在info.vue文件的最后添加如下代码，调整上传图片按钮的高度

```
<style scoped>
.tinymce-container {
  line-height: 29px;
}
</style>
```

# 5、图片的base64编码

Tinymce中的图片上传功能直接存储的是图片的base64编码，因此无需图片服务器

# 一、整合上传组件

参考 http://element-cn.eleme.io/#/zh-CN/component/upload 用户头像上传

## 1、上传默认封面

创建文件夹cover，上传默认的课程封面


## 2、定义默认封面

```
const defaultForm = {
  ......,
  cover: process.env.OSS_PATH + '/cover/default.gif',
  ......
}
```

## 3、定义data数据

```
BASE_API: process.env.BASE_API // 接口API地址
```

## 4、组件模板

在info.vue中添加上传组件模板

```
<!-- 课程封面-->
<el-form-item label="课程封面">

```

```
  <el-upload
    :show-file-list="false"
    :on-success="handleAvatarSuccess"
    :before-upload="beforeAvatarUpload"
    :action="BASE_API+'/admin/oss/file/upload?host=cover'"
    class="avatar-uploader">
    <img :src="courseInfo.cover">
  </el-upload>

</el-form-item>
```

## 5、结果回调

```
handleAvatarSuccess(res, file) {
  console.log(res)// 上传响应
  console.log(URL.createObjectURL(file.raw))// base64编码
  this.courseInfo.cover = res.data.url
},

beforeAvatarUpload(file) {
  const isJPG = file.type === 'image/jpeg'
  const isLt2M = file.size / 1024 / 1024 < 2

  if (!isJPG) {
    this.$message.error('上传头像图片只能是 JPG 格式!')
  }
  if (!isLt2M) {
    this.$message.error('上传头像图片大小不能超过 2MB!')
  }
  return isJPG && isLt2M
}
```

# 二、修改后端api

## 1、修改上传controller

添加host可选参数

```java
/**
     * 文件上传
     *
     * @param file
     */
@ApiOperation(value = "文件上传")
@PostMapping("upload")
public R upload(
    @ApiParam(name = "file", value = "文件", required = true)
    @RequestParam("file") MultipartFile file,

    @ApiParam(name = "host", value = "文件上传路径", required = false)) {

    String uploadUrl = fileService.upload(file);
    //返回r对象
    return R.ok().message("文件上传成功").data("url", uploadUrl);

}
```

## 2、综合测试

# 一、后端实现

## 1、业务层

接口：CourseService.java

```
CourseInfoForm getCourseInfoFormById(String id);
```

实现：CourseServiceImpl.java

```
@Override
public CourseInfoForm getCourseInfoFormById(String id) {

    Course course = this.getById(id);
    if(course == null){
        throw new GuliException(20001, "数据不存在");
    }
    CourseInfoForm courseInfoForm = new CourseInfoForm();
    BeanUtils.copyProperties(course, courseInfoForm);

    CourseDescription courseDescription = courseDescriptionService.getById(id);
    if(course != null){
        courseInfoForm.setDescription(courseDescription.getDescription());
    }

    return courseInfoForm;
}
```

## 2、web层

```
@ApiOperation(value = "根据ID查询课程")
@GetMapping("course-info/{id}")
public R getById(
    @ApiParam(name = "id", value = "课程ID", required = true)
    @PathVariable String id){
```

```
    CourseInfoForm courseInfoForm = courseService.getCourseInfoFormById(id);
    return R.ok().data("item", courseInfoForm);
}
```

## 3、Swagger中测试

# 二、前端实现

## 1、定义api

api/edu/course.js

```
getCourseInfoById(id) {
    return request({
        url: `${api_name}/course-info/${id}`,
        method: 'get'
    })
}
```

## 2、组件js

```
init() {
    if (this.$route.params && this.$route.params.id) {
        const id = this.$route.params.id
        //根据id获取课程基本信息
        this.fetchCourseInfoById(id)
    }

    ......
},

fetchCourseInfoById(id) {
  course.getCourseInfoById(id).then(response => {
    this.courseInfo = response.data.item
```

```
  }).catch((response) => {
    this.$message({
      type: 'error',
      message: response.message
    })
  })
},
```

# 三、解决级联下拉菜单回显问题

## 1、数据库中增加冗余列

```
subject_parent_id 课程专业父级ID
```

## 2、pojo中增加属性

entity.Course.java

form.CourseInfo.java

```
@ApiModelProperty(value = "课程专业父级ID")
private String subjectParentId;
```

## 3、vue组件中绑定数据

edu/course/infoinfo.vue

```
<el-select v-model="courseInfo.subjectParentId" ......
```

## 4、修改init方法

将 this.initSubjectList() 和 this.initTeacherList()移至else

```

```

```
init() {
  if (this.$route.params && this.$route.params.id) {
    const id = this.$route.params.id
    // 根据id获取课程基本信息
    this.fetchCourseInfoById(id)
  } else {
    this.courseInfo = { ...defaultForm }
    // 初始化分类列表
    this.initSubjectList()
    // 获取讲师列表
    this.initTeacherList()
  }
},
```

# 5、修改fetchCourseInfoById方法

```
fetchCourseInfoById(id) {
  course.getCourseInfoById(id).then(responseCourse => {
    this.courseInfo = responseCourse.data.item
    // 初始化分类列表
    subject.getNestedTreeList().then(responseSubject => {
      this.subjectNestedList = responseSubject.data.items
      for (let i = 0; i < this.subjectNestedList.length; i++) {
        if (this.subjectNestedList[i].id === this.courseInfo.subjectParentId) {
          this.subSubjectList = this.subjectNestedList[i].children
        }
      }
    })

    // 获取讲师列表
    this.initTeacherList()
  }).catch((response) => {
    this.$message({
      type: 'error',
      message: response.message
    })
  })
},
```

# 一、后端实现

## 1、业务层

接口：CourseService.java

```
void updateCourseInfoById(CourseInfoForm courseInfoForm);
```

实现：CourseServiceImpl.java

```
@Override
public void updateCourseInfoById(CourseInfoForm courseInfoForm) {
    //保存课程基本信息
    Course course = new Course();
    BeanUtils.copyProperties(courseInfoForm, course);
    boolean resultCourseInfo = this.updateById(course);
    if(!resultCourseInfo){
        throw new GuliException(20001, "课程信息保存失败");
    }
    //保存课程详情信息
    CourseDescription courseDescription = new CourseDescription();
    courseDescription.setDescription(courseInfoForm.getDescription());
    courseDescription.setId(course.getId());
    boolean resultDescription =
courseDescriptionService.updateById(courseDescription);
    if(!resultDescription){
        throw new GuliException(20001, "课程详情信息保存失败");
    }
}
```

## 2、web层

```
@ApiOperation(value = "更新课程")
@PutMapping("update-course-info/{id}")
public R updateCourseInfoById(
    @ApiParam(name = "CourseInfoForm", value = "课程基本信息", required = true)
```

```
    @RequestBody CourseInfoForm courseInfoForm,
    @ApiParam(name = "id", value = "课程ID", required = true)
    @PathVariable String id){
    courseService.updateCourseInfoById(courseInfoForm);
    return R.ok();
}
```

# 二、前端实现

## 1、定义api

course.js

```
updateCourseInfoById(courseInfo) {
  return request({
    url: `${api_name}/update-course-info/${courseInfo.id}`,
    method: 'put',
    data: courseInfo
  })
}
```

## 2、组件js

info.vue

```
updateData() {
    this.saveBtnDisabled = true
    course.updateCourseInfoById(this.courseInfo).then(response => {
        this.$message({
            type: 'success',
            message: '修改成功!'
        })
        return response// 将响应结果传递给then
    }).then(response => {
        this.$router.push({ path: '/edu/course/chapter/' +
response.data.courseId })
    }).catch((response) => {
        // console.log(response)
        this.$message({
```

```
            type: 'error',
            message: '保存失败'
        })
    })
},
```

# 一、后端实现

## 1、定义vo

ChapterVo

```
package com.guli.edu.vo;
@ApiModel(value = "章节信息")
@Data
public class ChapterVo implements Serializable {

    private static final long serialVersionUID = 1L;

    private String id;
    private String title;
    private List<VideoVo> children = new ArrayList<>();
}
```

VideoVo

```
package com.guli.edu.vo;
@ApiModel(value = "课时信息")
@Data
public class VideoVo implements Serializable {

    private static final long serialVersionUID = 1L;

    private String id;
    private String title;
    private Boolean free;
}
```

## 2、服务层

接口

```java
package com.guli.edu.service;
public interface ChapterService extends IService<Chapter> {
    List<ChapterVo> nestedList(String courseId);
}
```

实现

```java
package com.guli.edu.service.impl;

@Service
public class ChapterServiceImpl extends ServiceImpl<ChapterMapper, Chapter> implements Chap
{

    @Autowired
    private VideoService videoService;

    @Override
    public List<ChapterVo> nestedList(String courseId) {

        //最终要的到的数据列表
        ArrayList<ChapterVo> chapterVoArrayList = new ArrayList<>();

        //获取章节信息
        QueryWrapper<Chapter> queryWrapper1 = new QueryWrapper<>();
        queryWrapper1.eq("course_id", courseId);
        queryWrapper1.orderByAsc("sort", "id");
        List<Chapter> chapters = baseMapper.selectList(queryWrapper1);

        //获取课时信息
        QueryWrapper<Video> queryWrapper2 = new QueryWrapper<>();
        queryWrapper2.eq("course_id", courseId);
        queryWrapper2.orderByAsc("sort", "id");
        List<Video> videos = videoService.list(queryWrapper2);

        //填充章节vo数据
        int count1 = chapters.size();
        for (int i = 0; i < count1; i++) {
            Chapter chapter = chapters.get(i);

            //创建章节vo对象
            ChapterVo chapterVo = new ChapterVo();
```

```
            BeanUtils.copyProperties(chapter, chapterVo);
            chapterVoArrayList.add(chapterVo);

            //填充课时vo数据
            ArrayList<VideoVo> videoVoArrayList = new ArrayList<>();
            int count2 = videos.size();
            for (int j = 0; j < count2; j++) {

                Video video = videos.get(j);
                if(chapter.getId().equals(video.getChapterId())){

                    //创建课时vo对象
                    VideoVo videoVo = new VideoVo();
                    BeanUtils.copyProperties(video, videoVo);
                    videoVoArrayList.add(videoVo);
                }
            }
            chapterVo.setChildren(videoVoArrayList);
        }

        return chapterVoArrayList;
    }
}
```

# 3、web层

```
package com.guli.edu.controller.admin;

@Api(description="课程章节管理")
@CrossOrigin //跨域
@RestController
@RequestMapping("/admin/edu/chapter")
public class ChapterAdminController {

    @Autowired
    private ChapterService chapterService;

    @ApiOperation(value = "嵌套章节数据列表")
    @GetMapping("nested-list/{courseId}")
    public R nestedListByCourseId(
```

```
            @ApiParam(name = "courseId", value = "课程ID", required = true)
            @PathVariable String courseId){

        List<ChapterVo> chapterVoList = chapterService.nestedList(courseId);
        return R.ok().data("items", chapterVoList);
    }
}
```

## 4、Swagger测试

# 二、前端实现

## 1、定义api

chapter.js

```
import request from '@/utils/request'

const api_name = '/admin/edu/chapter'

export default {

  getNestedTreeList(courseId) {
    return request({
      url: `${api_name}/nested-list/${courseId}`,
      method: 'get'
    })
  }
}
```

## 2、定义组件脚本

定义data

```
courseId: '', // 所属课程
```

章节嵌套课时列表

```
chapterNestedList: [] //
```

created中调用init方法

```
created() {
    console.log('chapter created')
    this.init()
},
```

定义相关methods获取章节和课时列表

```
init() {
  if (this.$route.params && this.$route.params.id) {
    this.courseId = this.$route.params.id
    // 根据id获取课程基本信息
    this.fetchChapterNestedListByCourseId()
  }
},

fetchChapterNestedListByCourseId() {
  chapter.getNestedTreeList(this.courseId).then(response => {
    this.chapterNestedList = response.data.items
  })
},
```

# 3、定义组件模板

```
<el-button type="text">添加章节</el-button>
<!-- 章节 -->
<ul class="chanpterList">
    <li
        v-for="chapter in chapterNestedList"
        :key="chapter.id">
        <p>
            {{ chapter.title }}

            <span class="acts">
```

```
                <el-button type="text">添加课时</el-button>
                <el-button style="" type="text">编辑</el-button>
                <el-button type="text">删除</el-button>
            </span>
        </p>

        <!-- 视频 -->
        <ul class="chanpterList videoList">
            <li
                v-for="video in chapter.children"
                :key="video.id">
                <p>{{ video.title }}
                    <span class="acts">
                        <el-button type="text">编辑</el-button>
                        <el-button type="text">删除</el-button>
                    </span>
                </p>
            </li>
        </ul>
    </li>
</ul>
<div>
    <el-button @click="previous">上一步</el-button>
    <el-button :disabled="saveBtnDisabled" type="primary" @click="next">下一步</el-button>
</div>
```

# 4、定义样式

将样式的定义放在页面的最后

scope表示这里定义的样式只在当前页面范围内生效，不会污染到其他的页面

```
<style scoped>
.chanpterList{
    position: relative;
    list-style: none;
    margin: 0;
    padding: 0;
}
.chanpterList li{
```

```
  position: relative;
}
.chanpterList p{
  float: left;
  font-size: 20px;
  margin: 10px 0;
  padding: 10px;
  height: 70px;
  line-height: 50px;
  width: 100%;
  border: 1px solid #DDD;
}
.chanpterList .acts {
    float: right;
    font-size: 14px;
}

.videoList{
  padding-left: 50px;
}
.videoList p{
  float: left;
  font-size: 14px;
  margin: 10px 0;
  padding: 10px;
  height: 50px;
  line-height: 30px;
  width: 100%;
  border: 1px dotted #DDD;
}

</style>
```

# 一、新增章节

web层

```
@ApiOperation(value = "新增章节")
@PostMapping
public R save(
    @ApiParam(name = "chapterVo", value = "章节对象", required = true)
    @RequestBody Chapter chapter){
    chapterService.save(chapter);
    return R.ok();
}
```

# 二、根据id查询

web层

```
@ApiOperation(value = "根据ID查询章节")
@GetMapping("{id}")
public R getById(
    @ApiParam(name = "id", value = "章节ID", required = true)
    @PathVariable String id){
    Chapter chapter = chapterService.getById(id);
    return R.ok().data("item", chapter);
}
```

# 三、更新

web层

```
@ApiOperation(value = "根据ID修改章节")
@PutMapping("{id}")
public R updateById(
    @ApiParam(name = "id", value = "章节ID", required = true)
    @PathVariable String id,
    @ApiParam(name = "chapter", value = "章节对象", required = true)
```

```
    @RequestBody Chapter chapter){
    chapter.setId(id);
    chapterService.updateById(chapter);
    return R.ok();
}
```

# 四、删除

## 1、web层

```
@ApiOperation(value = "根据ID删除章节")
@DeleteMapping("{id}")
public R removeById(
    @ApiParam(name = "id", value = "章节ID", required = true)
    @PathVariable String id){
    boolean result = chapterService.removeChapterById(id);
    if(result){
        return R.ok();
    }else{
        return R.error().message("删除失败");
    }
}
```

## 2、Service

ChapterService层：接口

```
boolean removeChapterById(String id);
```

ChapterService层：实现

```
@Override
public boolean removeChapterById(String id) {
    //根据id查询是否存在视频，如果有则提示用户尚有子节点
    if(videoService.getCountByChapterId(id)){
        throw new GuliException(20001,"该分章节下存在视频课程，请先删除视频课程");
```

```
    }
    Integer result = baseMapper.deleteById(id);
    return null != result && result > 0;
}
```

VideoService：接口

```
boolean getCountByChapterId(String chapterId);
```

VideoService：实现

```
@Override
public boolean getCountByChapterId(String chapterId) {
    QueryWrapper<Video> queryWrapper = new QueryWrapper<>();
    queryWrapper.eq("chapter_id", chapterId);
    Integer count = baseMapper.selectCount(queryWrapper);
    return null != count && count > 0;
}
```

# 五、Swagger测试

# 一、定义api

```
removeById(id) {
  return request({
    url: `${api_name}/${id}`,
    method: 'delete'
  })
},
save(chapter) {
  return request({
    url: api_name,
    method: 'post',
    data: chapter
  })
},
getById(id) {
  return request({
    url: `${api_name}/${id}`,
    method: 'get'
  })
},
updateById(chapter) {
  return request({
    url: `${api_name}/${chapter.id}`,
    method: 'put',
    data: chapter
  })
}
```

# 二、新增章节页面功能

## 1、定义data数据

```
dialogChapterFormVisible: false, //是否显示章节表单
chapter: {// 章节对象
  title: '',
  sort: 0
```

```
}
```

## 2、添加章节按钮

```
<el-button type="text" @click="dialogChapterFormVisible = true">添加章节</el-button>
```

## 3、章节表单dialog

```
<!-- 添加和修改章节表单 -->
<el-dialog :visible.sync="dialogChapterFormVisible" title="添加章节">
    <el-form :model="chapter" label-width="120px">
        <el-form-item label="章节标题">
            <el-input v-model="chapter.title"/>
        </el-form-item>
        <el-form-item label="章节排序">
            <el-input-number v-model="chapter.sort" :min="0" controls-position="right"/>
        </el-form-item>
    </el-form>
    <div slot="footer" class="dialog-footer">
        <el-button @click="dialogChapterFormVisible = false">取 消</el-button>
        <el-button type="primary" @click="saveOrUpdate">确 定</el-button>
    </div>
</el-dialog>
```

## 4、添加章节methods

```
saveOrUpdate() {
  this.saveBtnDisabled = true
  if (!this.chapter.id) {
    this.saveData()
  } else {
    this.updateData()
  }
```

```
},
saveData() {
  this.chapter.courseId = this.courseId
  chapter.save(this.chapter).then(response => {
    this.$message({
      type: 'success',
      message: '保存成功!'
    })
    this.helpSave()
  }).catch((response) => {
    this.$message({
      type: 'error',
      message: response.message
    })
  })
},
updateData() {
},

helpSave(){
  this.dialogChapterFormVisible = false// 如果保存成功则关闭对话框
  this.fetchChapterNestedListByCourseId()// 刷新列表
  this.chapter.title = ''// 重置章节标题
  this.chapter.sort = 0// 重置章节标题
  this.saveBtnDisabled = false
},
```

# 三、修改章节信息

## 1、编辑章节按钮

```
<el-button type="text" @click="editChapter(chapter.id)">编辑</el-button>
```

## 2、定义编辑方法

```
editChapter(chapterId) {
    this.dialogChapterFormVisible = true
    chapter.getById(chapterId).then(response => {
```

```
        this.chapter = response.data.item
    })
},
```

## 3、定义更新方法

```
updateData() {
  chapter.updateById(this.chapter).then(response => {
    this.$message({
      type: 'success',
      message: '修改成功!'
    })
    this.helpSave()
  }).catch((response) => {
    // console.log(response)
    this.$message({
      type: 'error',
      message: response.message
    })
  })
},
```

# 四、删除章节

## 1、按钮

```
<el-button type="text" @click="removeChapter(chapter.id)">删除</el-button>
```

## 2、定义删除方法

```
removeChapter(chapterId) {
  this.$confirm('此操作将永久删除该记录，是否继续?', '提示', {
    confirmButtonText: '确定',
    cancelButtonText: '取消',
```

```
    type: 'warning'
  }).then(() => {
    return chapter.removeById(chapterId)
  }).then(() => {
    this.fetchChapterNestedListByCourseId()// 刷新列表
    this.$message({
      type: 'success',
      message: '删除成功!'
    })
  }).catch((response) => { // 失败
    if (response === 'cancel') {
      this.$message({
        type: 'info',
        message: '已取消删除'
      })
    } else {
      this.$message({
        type: 'error',
        message: response.message
      })
    }
  })
},
```

# 一、定义Form表单对象

VideoInfoForm.java

```
package com.guli.edu.form;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;
/**
 * @author helen
 * @since 2019/3/5
 */
@ApiModel(value = "课时基本信息", description = "编辑课时基本信息的表单对象")
@Data
public class VideoInfoForm {
    @ApiModelProperty(value = "视频ID")
    private String id;
    @ApiModelProperty(value = "节点名称")
    private String title;
    @ApiModelProperty(value = "课程ID")
    private String courseId;
    @ApiModelProperty(value = "章节ID")
    private String chapterId;
    @ApiModelProperty(value = "视频资源")
    private String videoSourceId;
    @ApiModelProperty(value = "显示排序")
    private Integer sort;

    @ApiModelProperty(value = "是否可以试听：0默认 1免费")
    private Boolean free;
}
```

# 二、课时保存

## 1、web层接口的定义

VideoAdminController.java

```
```

```java
package com.guli.edu.controller.admin;
@Api(description="课时管理")
@CrossOrigin //跨域
@RestController
@RequestMapping("/admin/edu/video")
public class VideoAdminController {
    @Autowired
    private VideoService videoService;
    @ApiOperation(value = "新增课时")
    @PostMapping("save-video-info")
    public R save(
            @ApiParam(name = "videoForm", value = "课时对象", required = true)
            @RequestBody VideoInfoForm videoInfoForm){
        videoService.saveVideoInfo(videoInfoForm);
        return R.ok();
    }
}
```

## 2、业务层

VideoService.java

```java
void saveVideoInfo(VideoInfoForm videoInfoForm);
```

VideoServiceImpl.java

```java
@Override
public void saveVideoInfo(VideoInfoForm videoInfoForm) {
    Video video = new Video();
    BeanUtils.copyProperties(videoInfoForm, video);
    boolean result = this.save(video);
    if(!result){
        throw new GuliException(20001, "课时信息保存失败");
    }
}
```

# 三、课时的修改

# 1、web层接口的定义

VideoAdminController.java

```
@ApiOperation(value = "根据ID查询课时")
@GetMapping("video-info/{id}")
public R getVideInfoById(
    @ApiParam(name = "id", value = "课时ID", required = true)
    @PathVariable String id){
    VideoInfoForm videoInfoForm = videoService.getVideoInfoFormById(id);
    return R.ok().data("item", videoInfoForm);
}
@ApiOperation(value = "更新课时")
@PutMapping("update-video-info/{id}")
public R updateCourseInfoById(
    @ApiParam(name = "VideoInfoForm", value = "课时基本信息", required = true)
    @RequestBody VideoInfoForm videoInfoForm,
    @ApiParam(name = "id", value = "课时ID", required = true)
    @PathVariable String id){
    videoService.updateVideoInfoById(videoInfoForm);
    return R.ok();
}
```

# 2、业务层

VideoService.java

```
VideoInfoForm getVideoInfoFormById(String id);
void updateVideoInfoById(VideoInfoForm videoInfoForm);
```

VideoServiceImpl.java

```
@Override
public VideoInfoForm getVideoInfoFormById(String id) {
    //从video表中取数据
    Video video = this.getById(id);
    if(video == null){
        throw new GuliException(20001, "数据不存在");
    }
```

```
    //创建videoInfoForm对象
    VideoInfoForm videoInfoForm = new VideoInfoForm();
    BeanUtils.copyProperties(video, videoInfoForm);
    return videoInfoForm;
}
@Override
public void updateVideoInfoById(VideoInfoForm videoInfoForm) {
    //保存课时基本信息
    Video video = new Video();
    BeanUtils.copyProperties(videoInfoForm, video);
    boolean result = this.updateById(video);
    if(!result){
        throw new GuliException(20001, "课时信息保存失败");
    }
}
```

# 四、课时的删除

## 1、web层接口的定义

VideoAdminController.java

```
@ApiOperation(value = "根据ID删除课时")
@DeleteMapping("{id}")
public R removeById(
    @ApiParam(name = "id", value = "课时ID", required = true)
    @PathVariable String id){
    boolean result = videoService.removeVideoById(id);
    if(result){
        return R.ok();
    }else{
        return R.error().message("删除失败");
    }
}
```

## 2、业务层

VideoService.java

```
```

```java
boolean removeVideoById(String id);
```

VideoServiceImpl.java

```java
@Override
public boolean removeVideoById(String id) {
    //删除视频资源 TODO
    Integer result = baseMapper.deleteById(id);
    return null != result && result > 0;
}
```

# 一、定义api

创建video.js

参考course.js

```
import request from '@/utils/request'
const api_name = '/admin/edu/video'
export default {
  saveVideoInfo(videoInfo) {
    return request({
      url: `${api_name}/save-video-info`,
      method: 'post',
      data: videoInfo
    })
  },
  getVideoInfoById(id) {
    return request({
      url: `${api_name}/video-info/${id}`,
      method: 'get'
    })
  },
  updateVideoInfoById(videoInfo) {
    return request({
      url: `${api_name}/update-video-info/${videoInfo.id}`,
      method: 'put',
      data: videoInfo
    })
  },
  removeById(id) {
    return request({
      url: `${api_name}/${id}`,
      method: 'delete'
    })
  }
}
```

# 二、新增课时页面功能

# 1、定义data数据

```
saveVideoBtnDisabled: false, // 课时按钮是否禁用
dialogVideoFormVisible: false, // 是否显示课时表单
chapterId: '', // 课时所在的章节id
video: {// 课时对象
  title: '',
  sort: 0,
  free: 0,
  videoSourceId: ''
},
```

# 2、添加课时按钮

```
<el-button type="text" @click="dialogVideoFormVisible = true; chapterId =
chapter.id">添加课时</el-button>
```

# 3、课时表单dialog

```
<!-- 添加和修改课时表单 -->
<el-dialog :visible.sync="dialogVideoFormVisible" title="添加课时">
  <el-form :model="video" label-width="120px">
    <el-form-item label="课时标题">
      <el-input v-model="video.title"/>
    </el-form-item>
    <el-form-item label="课时排序">
      <el-input-number v-model="video.sort" :min="0" controls-
position="right"/>
    </el-form-item>
    <el-form-item label="是否免费">
      <el-radio-group v-model="video.free">
        <el-radio :label="true">免费</el-radio>
        <el-radio :label="false">默认</el-radio>
      </el-radio-group>
    </el-form-item>
    <el-form-item label="上传视频">
      <!-- TODO -->
```

```
    </el-form-item>
  </el-form>
  <div slot="footer" class="dialog-footer">
    <el-button @click="dialogVideoFormVisible = false">取 消</el-button>
    <el-button :disabled="saveVideoBtnDisabled" type="primary"
@click="saveOrUpdateVideo">确 定</el-button>
  </div>
</el-dialog>
```

# 4、添加课时methods

引入video模块

```
import video from '@/api/edu/video'
```

方法的定义

```
saveOrUpdateVideo() {
  this.saveVideoBtnDisabled = true
  if (!this.video.id) {
    this.saveDataVideo()
  } else {
    this.updateDataVideo()
  }
},
saveDataVideo() {
  this.video.courseId = this.courseId
  this.video.chapterId = this.chapterId
  video.saveVideoInfo(this.video).then(response => {
    this.$message({
      type: 'success',
      message: '保存成功!'
    })
    this.helpSaveVideo()
  })
},
updateDataVideo() {
},
helpSaveVideo() {
  this.dialogVideoFormVisible = false// 如果保存成功则关闭对话框
```

```
  this.fetchChapterNestedListByCourseId()// 刷新列表
  this.video.title = ''// 重置章节标题
  this.video.sort = 0// 重置章节标题
  this.video.videoSourceId = ''// 重置视频资源id
  this.saveVideoBtnDisabled = false
},
```

# 三、修改课时信息

## 1、编辑课时按钮

```
<el-button type="text" @click="editVideo(video.id)">编辑</el-button>
```

## 2、定义编辑方法

```
editVideo(videoId) {
  this.dialogVideoFormVisible = true
  video.getVideoInfoById(videoId).then(response => {
    this.video = response.data.item
  })
},
```

## 3、定义更新方法

```
updateDataVideo() {
  video.updateVideoInfoById(this.video).then(response => {
    this.$message({
      type: 'success',
      message: '修改成功!'
    })
    this.helpSaveVideo()
  })
},
```

# 四、删除课时

## 1、按钮

```
<el-button type="text" @click="removeVideo(video.id)">删除</el-button>
```

## 2、定义删除方法

```
removeVideo(videoId) {
  this.$confirm('此操作将永久删除该记录, 是否继续?', '提示', {
    confirmButtonText: '确定',
    cancelButtonText: '取消',
    type: 'warning'
  }).then(() => {
    return video.removeById(videoId)
  }).then(() => {
    this.fetchChapterNestedListByCourseId()// 刷新列表
    this.$message({
      type: 'success',
      message: '删除成功!'
    })
  }).catch((response) => { // 失败
    if (response === 'cancel') {
      this.$message({
        type: 'info',
        message: '已取消删除'
      })
    }
  })
}
```

# 一、前端代码

## 1、定义api

分析这个页面一共有两个远程方法：一个是根基课程id获取课程基本预览信息，第二个是发布课程

```
getCoursePublishInfoById(id) {
  return request({
    url: `${api_name}/course-publish-info/${id}`,
    method: 'get'
  })
},

publishCourse(id) {
  return request({
    url: `${api_name}/publish-course/${id}`,
    method: 'put'
  })
}
```

## 2、定义数据模型

```
data() {
    return {
        saveBtnDisabled: false, // 保存按钮是否禁用
        courseId: '', // 所属课程
        coursePublish: {}
    }
},
```

## 3、完善步骤导航

edu/course/chapter.js

```
```

```
previous() {
  console.log('previous')
  this.$router.push({ path: '/edu/course/info/' + this.courseId })
},

next() {
  console.log('next')
  this.$router.push({ path: '/edu/course/publish/' + this.courseId })
}
```

edu/course/pubish.js

```
<div>
    <el-button @click="previous">返回修改</el-button>
    <el-button :disabled="saveBtnDisabled" type="primary" @click="publish">发布课程</el-button>
</div>
```

```
previous() {
  console.log('previous')
  this.$router.push({ path: '/edu/course/chapter/' + this.courseId  })
},

publish() {
  console.log('publish')
  course.publishCourse(this.courseId).then(response => {
    this.$router.push({ path: '/edu/course/list' })
  })
}
```

# 4、组件方法定义

import

```
import course from '@/api/edu/course'
```

created

```
created() {
    console.log('chapter created')
    this.init()
},
```

获取数据的方法

```
init() {
  if (this.$route.params && this.$route.params.id) {
    this.courseId = this.$route.params.id
    // 根据id获取课程基本信息
    this.fetchCoursePublishInfoById()
  }
},

fetchCoursePublishInfoById() {
  course.getCoursePublishInfoById(this.courseId).then(response => {
    this.coursePublish = response.data.item
  })
},
```

# 5、组件模板

```
<template>

  <div class="app-container">

    <h2 style="text-align: center;">发布新课程</h2>

    <el-steps :active="3" process-status="wait" align-center style="margin-bottom:
40px;">
      <el-step title="填写课程基本信息"/>
      <el-step title="创建课程大纲"/>
      <el-step title="发布课程"/>
    </el-steps>

```

```
    <div class="ccInfo">
      <img :src="coursePublish.cover">
      <div class="main">
        <h2>{{ coursePublish.title }}</h2>
        <p class="gray"><span>共{{ coursePublish.lessonNum }}课时</span></p>
        <p><span>所属分类：{{ coursePublish.subjectLevelOne }} — {{ coursePublish.subjectLevelTwo }}</span></p>
        <p>课程讲师：{{ coursePublish.teacherName }}</p>
        <h3 class="red">￥{{ coursePublish.price }}</h3>
      </div>
    </div>

    <div>
      <el-button @click="previous">返回修改</el-button>
      <el-button :disabled="saveBtnDisabled" type="primary" @click="publish">发布课程</el-button>
    </div>
  </div>
</template>
```

## 6、css样式

```
<style scoped>
.ccInfo {
    background: #f5f5f5;
    padding: 20px;
    overflow: hidden;
    border: 1px dashed #DDD;
    margin-bottom: 40px;
    position: relative;
}
.ccInfo img {
    background: #d6d6d6;
    width: 500px;
    height: 278px;
    display: block;
    float: left;
    border: none;
}
.ccInfo .main {
```

```
    margin-left: 520px;
}

.ccInfo .main h2 {
    font-size: 28px;
    margin-bottom: 30px;
    line-height: 1;
    font-weight: normal;
}
.ccInfo .main p {
    margin-bottom: 10px;
    word-wrap: break-word;
    line-height: 24px;
    max-height: 48px;
    overflow: hidden;
}

.ccInfo .main p {
    margin-bottom: 10px;
    word-wrap: break-word;
    line-height: 24px;
    max-height: 48px;
    overflow: hidden;
}
.ccInfo .main h3 {
    left: 540px;
    bottom: 20px;
    line-height: 1;
    font-size: 28px;
    color: #d32f24;
    font-weight: normal;
    position: absolute;
}
</style>
```

# 一、根据id查询课程发布信息

方式一：业务层组装多个表多次的查询结果

方式二：数据访问层进行关联查询

我们使用第二种方式实现

## 1、定义vo

```
package com.guli.edu.vo;

@ApiModel(value = "课程发布信息")
@Data
public class CoursePublishVo  implements Serializable {

    private static final long serialVersionUID = 1L;

    private String title;
    private String cover;
    private Integer lessonNum;
    private String subjectLevelOne;
    private String subjectLevelTwo;
    private String teacherName;
    private String price;//只用于显示
}
```

## 2、数据访问层

接口：CourseMapper.java

```
package com.guli.edu.mapper;
public interface CourseMapper extends BaseMapper<Course> {
    CoursePublishVo selectCoursePublishVoById(String id);
}
```

实现：CourseMapper.xml

```
<select id="getCoursePublishVoById" resultType="com.guli.edu.vo.CoursePublishVo">
    SELECT
        c.title,
        c.cover,
        c.lesson_num AS lessonNum,
        CONVERT(c.price, DECIMAL(8,2)) AS price,
        s1.title AS subjectLevelOne,
        s2.title AS subjectLevelTwo,
        t.name AS teacherName
    FROM
        edu_course c
        LEFT JOIN edu_teacher t ON c.teacher_id = t.id
        LEFT JOIN edu_subject s1 ON c.subject_parent_id = s1.id
        LEFT JOIN edu_subject s2 ON c.subject_id = s2.id
    WHERE
        c.id = #{id}
</select>
```

## 3、业务层

接口：CourseService.java

```
CoursePublishVo getCoursePublishVoById(String id);
```

实现：CourseServiceImpl.java

```
@Override
public CoursePublishVo getCoursePublishVoById(String id) {
    return baseMapper.getCoursePublishVoById(id);
}
```

## 4、web层

```

```

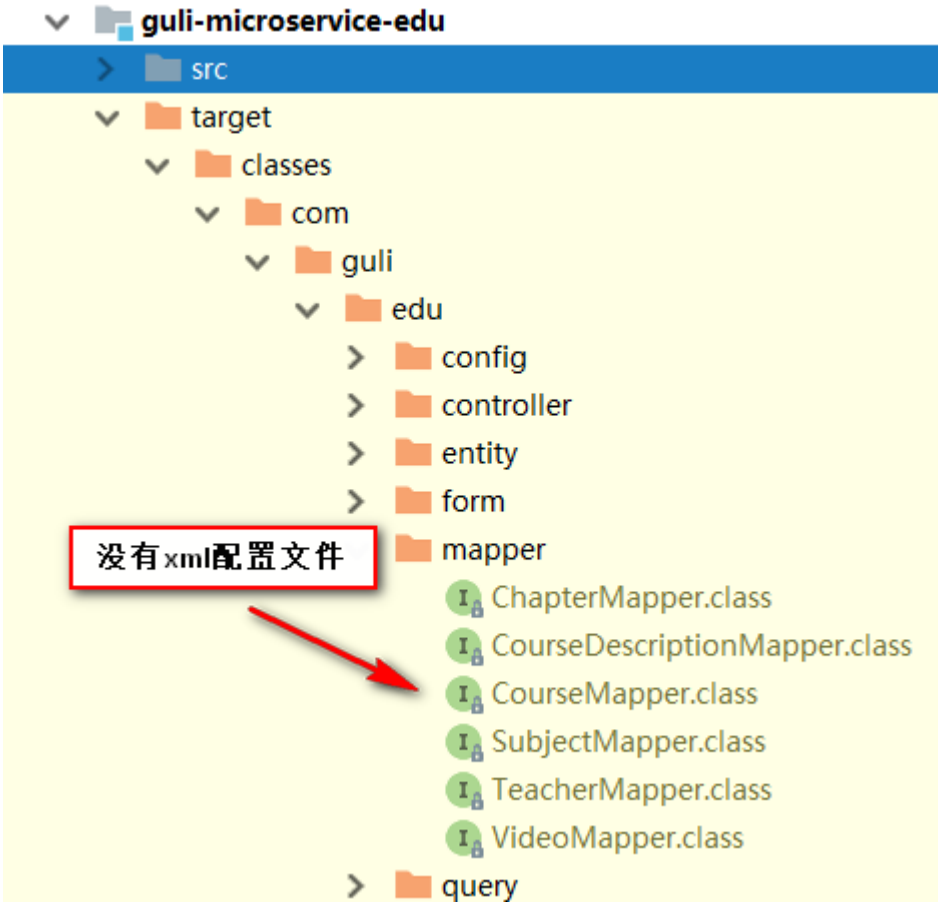
```java
@ApiOperation(value = "根据ID获取课程发布信息")
@GetMapping("course-publish-info/{id}")
public R getCoursePublishVoById(
    @ApiParam(name = "id", value = "课程ID", required = true)
    @PathVariable String id){

    CoursePublishVo courseInfoForm = courseService.getCoursePublishVoById(id);
    return R.ok().data("item", courseInfoForm);
}
```

测试：报告异常

AbstractHandlerExceptionResolver.java:194 |org.springframework.web.servlet.mvc.method.annotation.ExceptionHandlerExceptionResolver |Resolved exception caused by handler execution: org.apache.ibatis.binding.BindingException: Invalid bound statement (not found): com.guli.edu.mapper.CourseMapper.getCoursePublishVoById

问题分析：

dao层编译后只有class文件，没有mapper.xml，因为maven工程在默认情况下src/main/java目录下的所有资源文件是不发布到target目录下的，

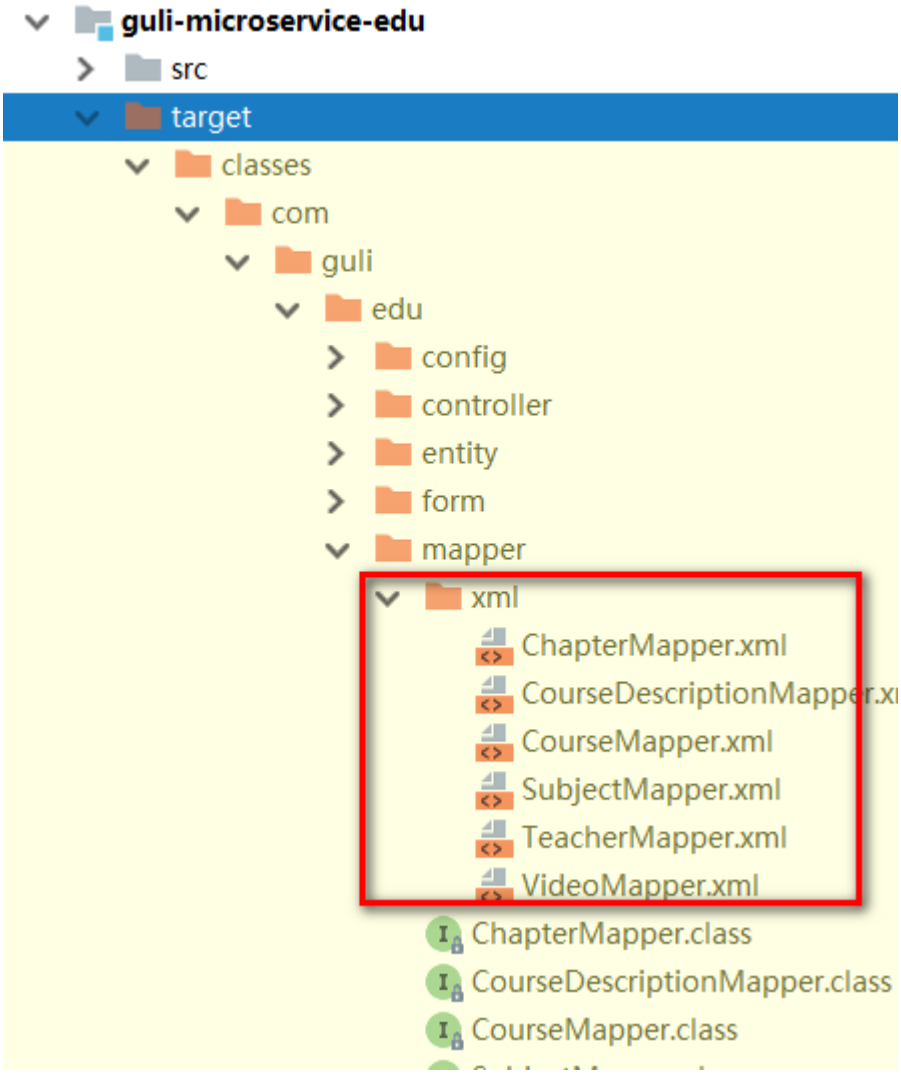


解决方案:

1、在guli_edu的pom中配置如下节点

```
<!-- 项目打包时会将java目录中的*.xml文件也进行打包 -->
<build>
    <resources>
        <resource>
            <directory>src/main/java</directory>
            <includes>
                <include>**/*.xml</include>
            </includes>
            <filtering>false</filtering>
        </resource>
    </resources>
</build>
```

重新打包项目会发现target目录下出现了xml文件夹




2、在Spring Boot配置文件中添加配置

```
#配置mapper xml文件的路径
mybatis-plus.mapper-locations=classpath:com/guli/edu/mapper/xml/*.xml
```

# 二、根据id发布课程

## 1、web层

```
@ApiOperation(value = "根据id发布课程")
@PutMapping("publish-course/{id}")
public R publishCourseById(
    @ApiParam(name = "id", value = "课程ID", required = true)
```

```
    @PathVariable String id){

    courseService.publishCourseById(id);
    return R.ok();
}
```

## 2、service层

接口

```
void publishCourseById(String id);
```

实现

```
@Override
public boolean publishCourseById(String id) {
    Course course = new Course();
    course.setId(id);
    course.setStatus(Course.COURSE_NORMAL);
    Integer count = baseMapper.updateById(course);
    return null != count && count > 0;
}
```

# 一、后端实现

## 1、定义搜索对象

CourseQuery

```
package com.guli.edu.query;
@ApiModel(value = "Course查询对象", description = "课程查询对象封装")
@Data
public class CourseQuery implements Serializable {
    private static final long serialVersionUID = 1L;
    @ApiModelProperty(value = "课程名称")
    private String title;
    @ApiModelProperty(value = "讲师id")
    private String teacherId;
    @ApiModelProperty(value = "一级类别id")
    private String subjectParentId;
    @ApiModelProperty(value = "二级类别id")
    private String subjectId;
}
```

## 2、定义service方法

接口

```
void pageQuery(Page<Course> pageParam, CourseQuery courseQuery);
```

实现

```
@Override
public void pageQuery(Page<Course> pageParam, CourseQuery courseQuery) {
    QueryWrapper<Course> queryWrapper = new QueryWrapper<>();
    queryWrapper.orderByDesc("gmt_create");
    if (courseQuery == null){
        baseMapper.selectPage(pageParam, queryWrapper);
        return;
```

```
    }
    String title = courseQuery.getTitle();
    String teacherId = courseQuery.getTeacherId();
    String subjectParentId = courseQuery.getSubjectParentId();
    String subjectId = courseQuery.getSubjectId();
    if (!StringUtils.isEmpty(title)) {
        queryWrapper.like("title", title);
    }
    if (!StringUtils.isEmpty(teacherId) ) {
        queryWrapper.eq("teacher_id", teacherId);
    }
    if (!StringUtils.isEmpty(subjectParentId)) {
        queryWrapper.ge("subject_parent_id", subjectParentId);
    }
    if (!StringUtils.isEmpty(subjectId)) {
        queryWrapper.ge("subject_id", subjectId);
    }
    baseMapper.selectPage(pageParam, queryWrapper);
}
```

# 3、定义web层方法

```
@ApiOperation(value = "分页课程列表")
@GetMapping("{page}/{limit}")
public R pageQuery(
        @ApiParam(name = "page", value = "当前页码", required = true)
        @PathVariable Long page,
        @ApiParam(name = "limit", value = "每页记录数", required = true)
        @PathVariable Long limit,
        @ApiParam(name = "courseQuery", value = "查询对象", required = false)
                CourseQuery courseQuery){
    Page<Course> pageParam = new Page<>(page, limit);
    courseService.pageQuery(pageParam, courseQuery);
    List<Course> records = pageParam.getRecords();
    long total = pageParam.getTotal();
    return  R.ok().data("total", total).data("rows", records);
}
```

# 二、前端分页查询列表

## 1、定义api

course.js

```
getPageList(page, limit, searchObj) {
  return request({
    url: `${api_name}/${page}/${limit}`,
    method: 'get',
    params: searchObj
  })
},
```

## 2、组件中的js

src/views/edu/list.vue

```
<script>
import course from '@/api/edu/course'
import teacher from '@/api/edu/teacher'
import subject from '@/api/edu/subject'
export default {
  data() {
    return {
      listLoading: true, // 是否显示loading信息
      list: null, // 数据列表
      total: 0, // 总记录数
      page: 1, // 页码
      limit: 10, // 每页记录数
      searchObj: {
        subjectParentId: '',
        subjectId: '',
        title: '',
        teacherId: ''
      }, // 查询条件
      teacherList: [], // 讲师列表
      subjectNestedList: [], // 一级分类列表
      subSubjectList: [] // 二级分类列表,
    }
```

```
  },
  created() { // 当页面加载时获取数据
    this.fetchData()
    // 初始化分类列表
    this.initSubjectList()
    // 获取讲师列表
    this.initTeacherList()
  },
  methods: {
    fetchData(page = 1) { // 调用api层获取数据库中的数据
      console.log('加载列表')
      // 当点击分页组件的切换按钮的时候，会传输一个当前页码的参数page
      // 解决分页无效问题
      this.page = page
      this.listLoading = true
      course.getPageList(this.page, this.limit, this.searchObj).then(response
=> {
        // debugger 设置断点调试
        if (response.success === true) {
          this.list = response.data.rows
          this.total = response.data.total
        }
        this.listLoading = false
      })
    },
    initTeacherList() {
      teacher.getList().then(response => {
        this.teacherList = response.data.items
      })
    },
    initSubjectList() {
      subject.getNestedTreeList().then(response => {
        this.subjectNestedList = response.data.items
      })
    },
    subjectLevelOneChanged(value) {
      for (let i = 0; i < this.subjectNestedList.length; i++) {
        if (this.subjectNestedList[i].id === value) {
          this.subSubjectList = this.subjectNestedList[i].children
          this.searchObj.subjectId = ''
        }
      }
    },
    resetData() {
      this.searchObj = {}
```

```
      this.subSubjectList = [] // 二级分类列表
      this.fetchData()
    }
  }
}
</script>
```

## 3、组件模板

查询表单

```
<!--查询表单-->
<el-form :inline="true" class="demo-form-inline">
  <!-- 所属分类：级联下拉列表 -->
  <!-- 一级分类 -->
  <el-form-item label="课程类别">
    <el-select
      v-model="searchObj.subjectParentId"
      placeholder="请选择"
      @change="subjectLevelOneChanged">
      <el-option
        v-for="subject in subjectNestedList"
        :key="subject.id"
        :label="subject.title"
        :value="subject.id"/>
    </el-select>
    <!-- 二级分类 -->
    <el-select v-model="searchObj.subjectId" placeholder="请选择">
      <el-option
        v-for="subject in subSubjectList"
        :key="subject.id"
        :label="subject.title"
        :value="subject.id"/>
    </el-select>
  </el-form-item>
  <!-- 标题 -->
  <el-form-item>
    <el-input v-model="searchObj.title" placeholder="课程标题"/>
  </el-form-item>
  <!-- 讲师 -->
  <el-form-item>
    <el-select
```

```
      v-model="searchObj.teacherId"
      placeholder="请选择讲师">
      <el-option
        v-for="teacher in teacherList"
        :key="teacher.id"
        :label="teacher.name"
        :value="teacher.id"/>
    </el-select>
  </el-form-item>
  <el-button type="primary" icon="el-icon-search" @click="fetchData()">查
询</el-button>
  <el-button type="default" @click="resetData()">清空</el-button>
</el-form>
```

表格和分页

表格添加了 row-class-name="myClassList" 样式定义

```
<!-- 表格 -->
<el-table
  v-loading="listLoading"
  :data="list"
  element-loading-text="数据加载中"
  border
  fit
  highlight-current-row
  row-class-name="myClassList">
  <el-table-column
    label="序号"
    width="70"
    align="center">
    <template slot-scope="scope">
      {{ (page - 1) * limit + scope.$index + 1 }}
    </template>
  </el-table-column>
  <el-table-column label="课程信息" width="470" align="center">
    <template slot-scope="scope">
      <div class="info">
        <div class="pic">
          <img :src="scope.row.cover" alt="scope.row.title" width="150px">
        </div>
        <div class="title">
          <a href="">{{ scope.row.title }}</a>
                                    课时
```

```
          <p>{{ scope.row.lessonNum }}    </p>
        </div>
      </div>
    </template>
  </el-table-column>
  <el-table-column label="创建时间" align="center">
    <template slot-scope="scope">
      {{ scope.row.gmtCreate.substr(0, 10) }}
    </template>
  </el-table-column>
  <el-table-column label="发布时间" align="center">
    <template slot-scope="scope">
      {{ scope.row.gmtModified.substr(0, 10) }}
    </template>
  </el-table-column>
  <el-table-column label="价格" width="100" align="center" >
    <template slot-scope="scope">
      {{ Number(scope.row.price) === 0 ? '免费' :
      '¥' + scope.row.price.toFixed(2) }}
    </template>
  </el-table-column>
  <el-table-column prop="buyCount" label="付费学员" width="100" align="center"
>
    <template slot-scope="scope">
      {{ scope.row.buyCount }}人
    </template>
  </el-table-column>
  <el-table-column label="操作" width="150" align="center">
    <template slot-scope="scope">
      <router-link :to="'/edu/course/info/'+scope.row.id">
        <el-button type="text" size="mini" icon="el-icon-edit">编辑课程信
息</el-button>
      </router-link>
      <router-link :to="'/edu/course/chapter/'+scope.row.id">
        <el-button type="text" size="mini" icon="el-icon-edit">编辑课程大
纲</el-button>
      </router-link>
      <el-button type="text" size="mini" icon="el-icon-delete">删除</el-button>
    </template>
  </el-table-column>
</el-table>
<!-- 分页 -->
<el-pagination
  :current-page="page"
  :page-size="limit"
```

```
  :total="total"
  style="padding: 30px 0; text-align: center;"
  layout="total, prev, pager, next, jumper"
  @current-change="fetchData"
/>
```

# 4、css的定义

```
<style scoped>
.myClassList .info {
    width: 450px;
    overflow: hidden;
}
.myClassList .info .pic {
    width: 150px;
    height: 90px;
    overflow: hidden;
    float: left;
}
.myClassList .info .pic a {
    display: block;
    width: 100%;
    height: 100%;
    margin: 0;
    padding: 0;
}
.myClassList .info .pic img {
    display: block;
    width: 100%;
}
.myClassList td .info .title {
    width: 280px;
    float: right;
    height: 90px;
}
.myClassList td .info .title a {
    display: block;
    height: 48px;
    line-height: 24px;
    overflow: hidden;
    color: #00baf2;
    margin-bottom: 12px;
```

```
}
.myClassList td .info .title p {
    line-height: 20px;
    margin-top: 5px;
    color: #818181;
}
</style>
```

# 一、后端实现

## 1、web层

定义删除api方法：CourseAdminController.java

```
@ApiOperation(value = "根据ID删除课程")
@DeleteMapping("{id}")
public R removeById(
    @ApiParam(name = "id", value = "课程ID", required = true)
    @PathVariable String id){
    boolean result = courseService.removeCourseById(id);
    if(result){
        return R.ok();
    }else{
        return R.error().message("删除失败");
    }
}
```

## 2、service层

如果用户确定删除，则首先删除video记录，然后删除chapter记录，最后删除Course记录

### 2.1、在VideoService中定义根据courseId删除video业务方法

接口

```
boolean removeByCourseId(String courseId);
```

实现

```
@Override
public boolean removeByCourseId(String courseId) {
    QueryWrapper<Video> queryWrapper = new QueryWrapper<>();
    queryWrapper.eq("course_id", courseId);
    Integer count = baseMapper.delete(queryWrapper);
    return null != count && count > 0;
```

```
}
```

### 2.2、在ChapterService中定义根据courseId删除chapter业务方法

接口

```
boolean removeByCourseId(String courseId);
```

实现

```
@Override
public boolean removeByCourseId(String courseId) {
    QueryWrapper<Chapter> queryWrapper = new QueryWrapper<>();
    queryWrapper.eq("course_id", courseId);
    Integer count = baseMapper.delete(queryWrapper);
    return null != count && count > 0;
}
```

### 2.3、删除当前course记录

接口：CourseService.java

```
boolean removeCourseById(String id);
```

实现：CourseServiceImpl.java

```
@Override
public boolean removeCourseById(String id) {
    //根据id删除所有视频
    videoService.removeByCourseId(id);
    //根据id删除所有章节
    chapterService.removeByCourseId(id);
    Integer result = baseMapper.deleteById(id);
    return null != result && result > 0;
}
```

# 二、前端实现

## 1、定义api

course.js中添加删除方法

```
removeById(id) {
    return request({
        url: `${api_name}/${id}`,
        method: 'delete'
    })
}
```

## 2、修改删除按钮

src/api/edu/course.js 删除按钮注册click事件

```
<el-button type="text" size="mini" icon="el-icon-delete"
@click="removeDataById(scope.row.id)">删除</el-button>
```

## 3、编写删除方法

```
removeDataById(id) {
    // debugger
    this.$confirm('此操作将永久删除该课程，以及该课程下的章节和视频，是否继续?', '提
示', {
        confirmButtonText: '确定',
        cancelButtonText: '取消',
        type: 'warning'
    }).then(() => {
        return course.removeById(id)
    }).then(() => {
        this.fetchData()
```

```
        this.$message({
            type: 'success',
            message: '删除成功!'
        })
    }).catch((response) => { // 失败
        if (response === 'cancel') {
            this.$message({
                type: 'info',
                message: '已取消删除'
            })
        }
    })
}
```

# 一、阿里云视频点播技术能力盘点

参考文章：

https://blog.csdn.net/qq_33857573/article/details/79564255

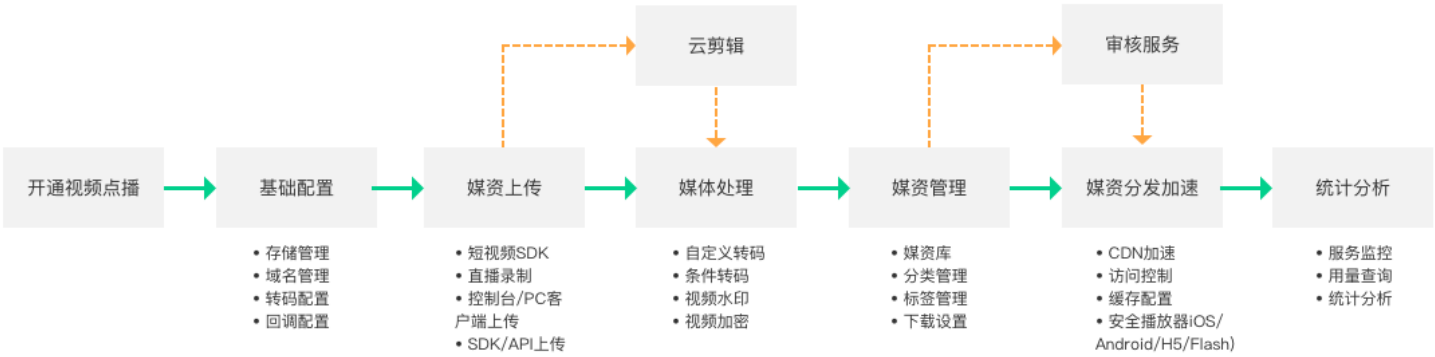
视频点播（ApsaraVideo for VoD）是集音视频采集、编辑、上传、自动化转码处理、媒体资源管理、分发加速于一体的一站式音视频点播解决方案。


## 1、应用场景

- 音视频网站：无论是初创视频服务企业，还是已拥有海量视频资源，可定制化的点播服务帮助客户快速搭建拥有极致观看体验、安全可靠的视频点播应用。
- 短视频：集音视频拍摄、特效编辑、本地转码、高速上传、自动化云端转码、媒体资源管理、分发加速、播放于一体的完整短视频解决方案。目前已帮助1000+APP快速实现手机短视频功能。
- 直播转点播：将直播流同步录制为点播视频，用于回看。并支持媒资管理、媒体处理（转码及内容审核/智能首图等AI处理）、内容制作（云剪辑）、CDN分发加速等一系列操作。
- 在线教育：为在线教育客户提供简单易用、安全可靠的视频点播服务。可通过控制台/API等多种方式上传教学视频，强大的转码能力保证视频可以快速发布，覆盖全网的加速节点保证学生观看的流畅度。防盗链、视频加密等版权保护方案保护教学内容不被窃取。
- 视频生产制作：提供在线可视化剪辑平台及丰富的OpenAPI，帮助客户高效处理、制作视频内容。除基础的剪切拼接、混音、遮标、特效、合成等一系列功能外，依托云剪辑及点播一体化服务还可实现标准化、智能化剪辑生产，大大降低视频制作的槛，缩短制作时间，提升内容生产效率。
- 内容审核：应用于短视频平台、传媒行业审核等场景，帮助客户从从语音、文字、视觉等多维度精

准识别视频、封面、标题或评论的违禁内容进行AI智能审核与人工审核。

## 2、功能介绍




# 二、开通视频点播云平台

## 1、选择视频点播服务

产品->企业应用->视频云->视频点播

## 2、开通视频点播


## 3、选择按使用流量计费

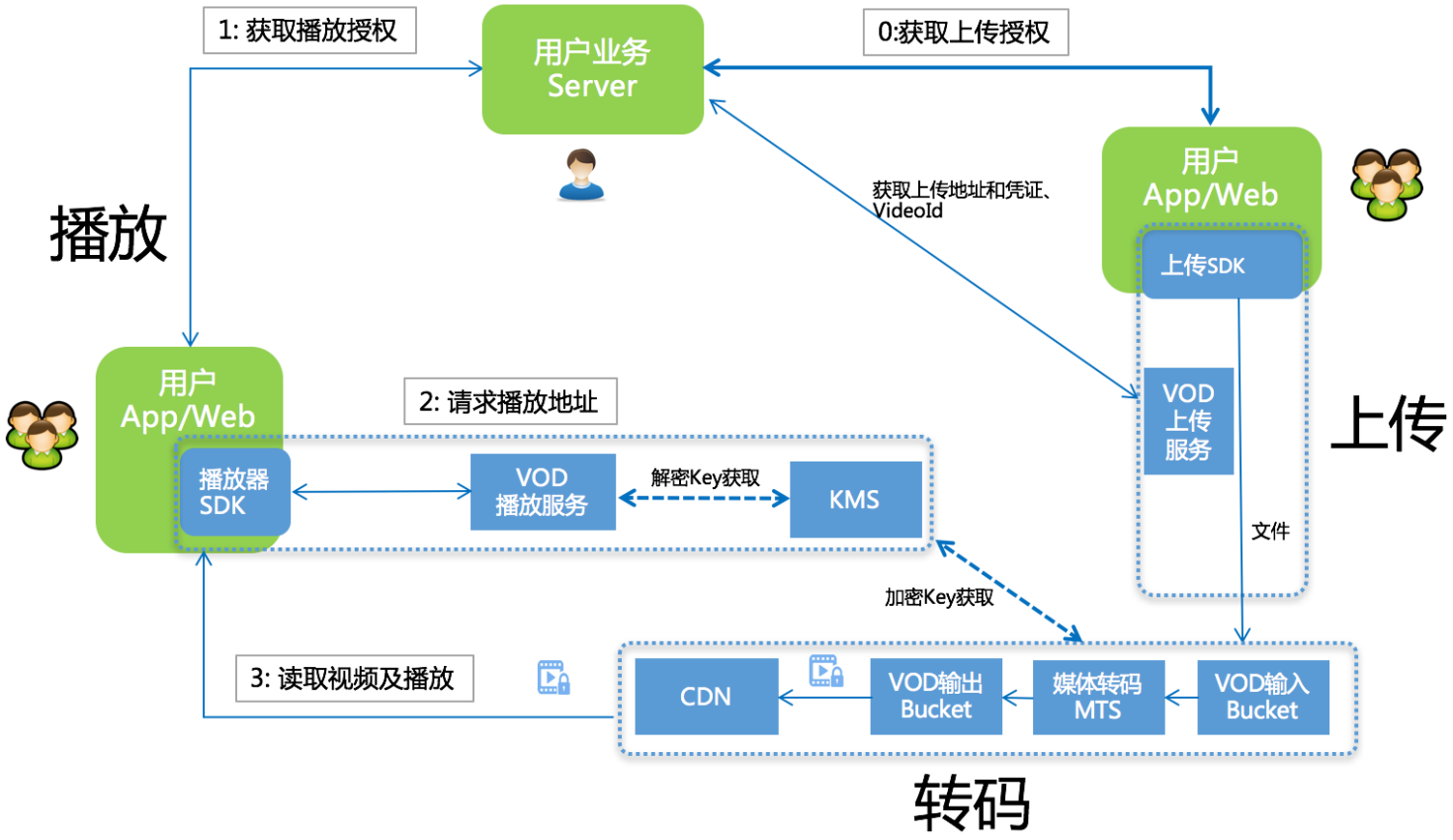

## 4、资费说明

https://www.aliyun.com/price/product?spm=a2c4g.11186623.2.12.7fbd59b9vmXVN6#/vod/detail

- 后付费
- 套餐包
- 欠费说明
- 计费案例：https://help.aliyun.com/document_detail/64032.html?spm=a2c4g.11186623.4.3.363db1bcfdvxB5

## 5、整体流程

使用视频点播实现音视频上传、存储、处理和播放的整体流程如下：




- 用户获取上传授权。
- VoD下发 上传地址和凭证 及 VideoId。
- 用户上传视频保存视频ID(VideoId)。
- 用户服务端获取播放凭证。
- VoD下发带时效的播放凭证。
- 用户服务端将播放凭证下发给客户端完成视频播放。

# 三、视频点播服务的基本使用

完整的参考文档

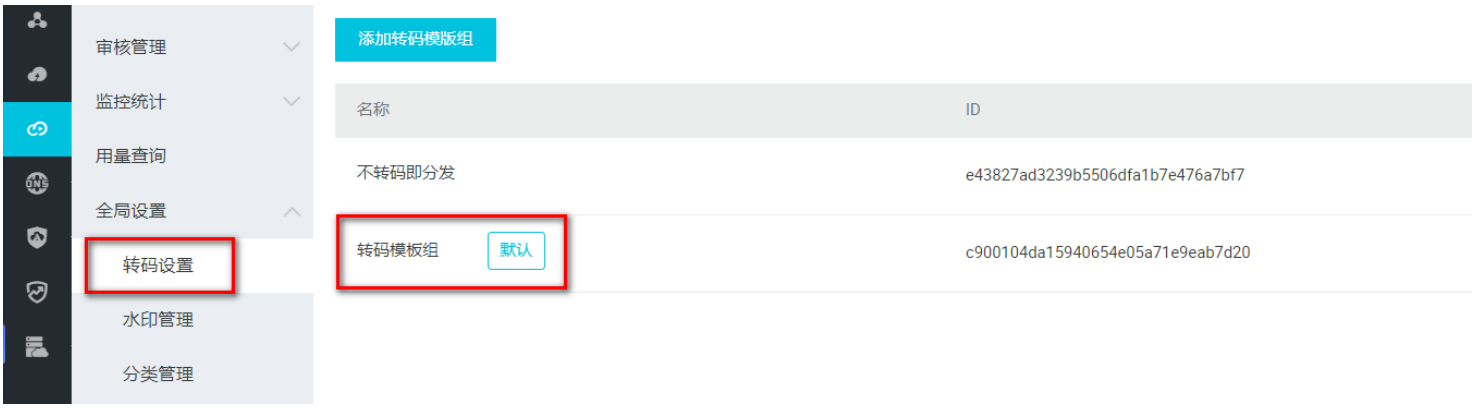
https://help.aliyun.com/product/29932.html?spm=a2c4g.11186623.6.540.3c356a58OEmVZJ

## 1、设置转码格式

选择全局设置 > 转码设置，单击添加转码模板组。

在视频转码模板组页面，根据业务需求选择封装格式和清晰度。

或直接将已有的模板设置为默认即可




## 2、分类管理

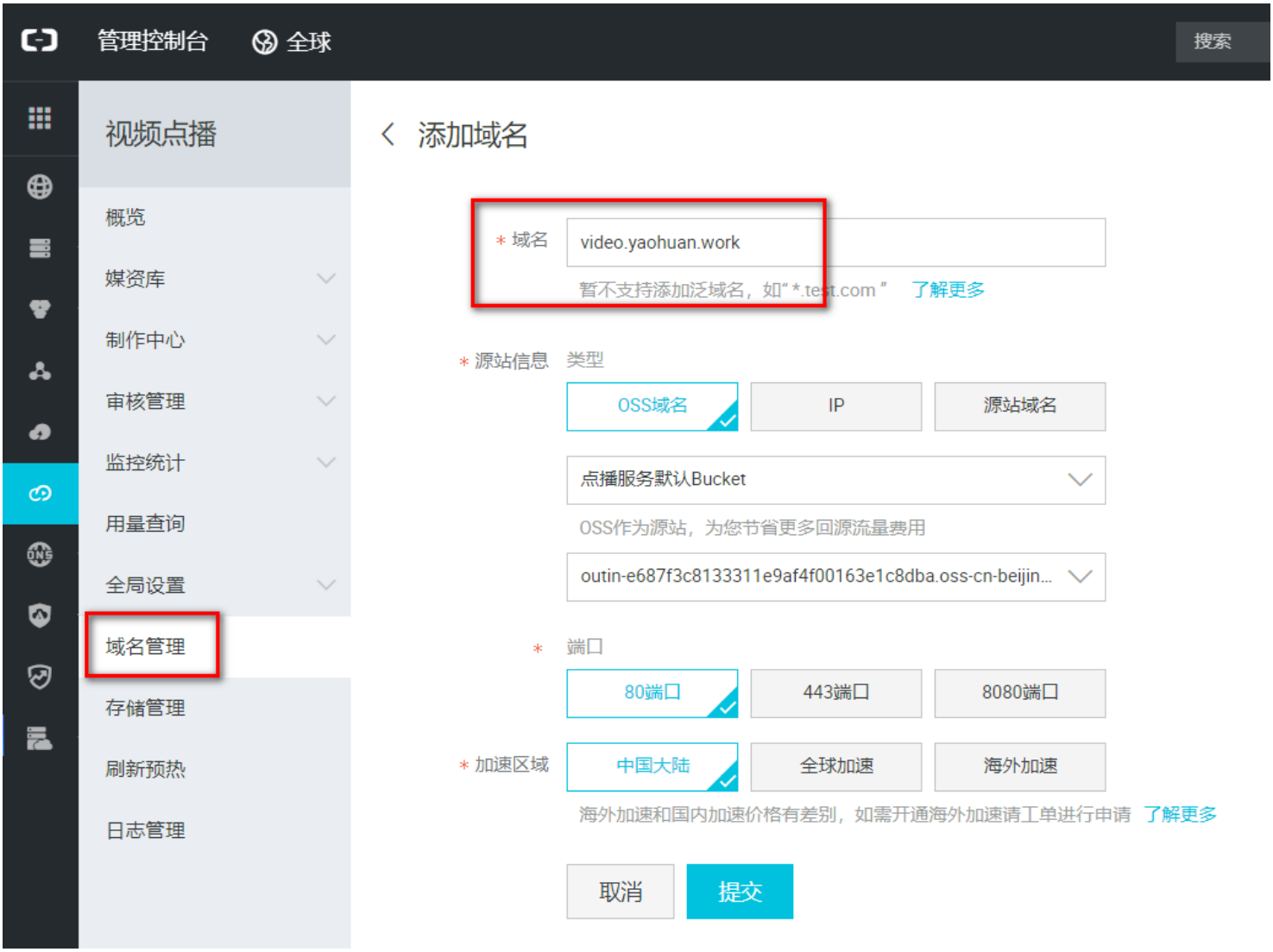
选择全局设置 > 分类管理

## 3、上传视频文件

选择媒资库 > 音视频，单击上传音视频

## 4、配置域名

音视频上传完成后，必须配一个已备案的域名，并完成CNAME绑定




得到CNAME

| 域名 | CNAME | 状态 |
| --- | --- | --- |
| video.yaohuan.work | video.yaohuan.work.w.kunlunca.com | 正常运行 |

在购买域名的服务商处的管理控制台配置域名解析

| 记录类型 | 主机记录 | 解析线路(isp) | 记录值 | MX优先级 | TTL | 状态 |
| --- | --- | --- | --- | --- | --- | --- |
| CNAME | video | 默认 | video.yaohuan.work.w.kunlunca.com | -- | 10 分钟 | 正常 |

# 5、在控制台查看视频

此时视频可以在阿里云控制台播放

# 6、获取web播放器代码


# 一、服务端SDK

## 1、简介

sdk的方式将api进行了进一步的封装，不用自己创建工具类。

我们可以基于服务端SDK编写代码来调用点播API，实现对点播产品和服务的快速操作。

## 2、功能介绍

- SDK封装了对API的调用请求和响应，避免自行计算较为繁琐的 API签名。
- 支持所有点播服务的API，并提供了相应的示例代码。
- 支持7种开发语言，包括：Java、Python、PHP、.NET、Node.js、Go、C/C++。
- 通常在发布新的API后，我们会及时同步更新SDK，所以即便您没有找到对应API的示例代码，也可以参考旧的示例自行实现调用。

# 二、使用SDK

## 1、安装

参考文档：https://help.aliyun.com/document_detail/57756.html

添加maven仓库的配置和依赖到pom

```
<repositories>
    <repository>
        <id>sonatype-nexus-staging</id>
        <name>Sonatype Nexus Staging</name>
        <url>https://oss.sonatype.org/service/local/staging/deploy/maven2/</url>
        <releases>
            <enabled>true</enabled>
        </releases>
        <snapshots>
            <enabled>true</enabled>
        </snapshots>
```

```
    </repository>
</repositories>
```

```
  <dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-core</artifactId>
    <version>4.3.3</version>
  </dependency>
  <dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-vod</artifactId>
    <version>2.15.5</version>
  </dependency>
  <dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.2</version>
  </dependency>

```

## 2、初始化

参考文档：https://help.aliyun.com/document_detail/61062.html

根据文档示例创建 AliyunVODSDKUtils.java

```
package com.atguigu.aliyunvod.util;

public class AliyunVodSDKUtils {

    public static DefaultAcsClient initVodClient(String accessKeyId, String accessKeySecret) throws ClientException {
        String regionId = "cn-shanghai";  // 点播服务接入区域
        DefaultProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKeySecret);
        DefaultAcsClient client = new DefaultAcsClient(profile);
        return client;
    }
}
```

## 3、创建测试类

创建 VodSdkTest.java

```
package com.atguigu.aliyunvod;

public class VodSdkTest {

    String accessKeyId = "你的accessKeyId";
    String accessKeySecret = "你的accessKeySecret";

}
```

# 三、创建测试用例

参考文档：https://help.aliyun.com/document_detail/61064.html

## 1、获取视频播放凭证

根据文档中的代码，修改如下

```
/**
     * 获取视频播放凭证
     * @throws ClientException
     */
@Test
public void testGetVideoPlayAuth() throws ClientException {

    //初始化客户端、请求对象和相应对象
    DefaultAcsClient client = AliyunVodSDKUtils.initVodClient(accessKeyId,
accessKeySecret);
    GetVideoPlayAuthRequest request = new GetVideoPlayAuthRequest();
    GetVideoPlayAuthResponse response = new GetVideoPlayAuthResponse();

    try {

        //设置请求参数
        request.setVideoId("视频ID");
```

```
        //获取请求响应
        response = client.getAcsResponse(request);

        //输出请求结果
        //播放凭证
        System.out.print("PlayAuth = " + response.getPlayAuth() + "\n");
        //VideoMeta信息
        System.out.print("VideoMeta.Title = " + response.getVideoMeta().getTitle()
+ "\n");
    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }

    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

## 2、获取视频播放地址

```
/**
     * 获取视频播放地址
     * @throws ClientException
     */
@Test
public void testGetPlayInfo() throws ClientException {

    //初始化客户端、请求对象和相应对象
    DefaultAcsClient client = AliyunVodSDKUtils.initVodClient(accessKeyId,
accessKeySecret);
    GetPlayInfoRequest request = new GetPlayInfoRequest();
    GetPlayInfoResponse response = new GetPlayInfoResponse();

    try {

        //设置请求参数
        //注意：这里只能获取非加密视频的播放地址
        request.setVideoId("视频ID");
        //获取请求响应
        response = client.getAcsResponse(request);

        //输出请求结果
```

```
        List<GetPlayInfoResponse.PlayInfo> playInfoList =
response.getPlayInfoList();
        //播放地址
        for (GetPlayInfoResponse.PlayInfo playInfo : playInfoList) {
            System.out.print("PlayInfo.PlayURL = " + playInfo.getPlayURL() +
"\n");
        }
        //Base信息
        System.out.print("VideoBase.Title = " + response.getVideoBase().getTitle()
+ "\n");

    } catch (Exception e) {
        System.out.print("ErrorMessage = " + e.getLocalizedMessage());
    }

    System.out.print("RequestId = " + response.getRequestId() + "\n");
}
```

参考文档：https://help.aliyun.com/document_detail/53406.html

# 一、安装SDK

## 1、配置pom

```
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-core</artifactId>
    <version>4.3.3</version>
</dependency>
<dependency>
    <groupId>com.aliyun.oss</groupId>
    <artifactId>aliyun-sdk-oss</artifactId>
    <version>3.1.0</version>
</dependency>
 <dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-vod</artifactId>
    <version>2.15.2</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.28</version>
</dependency>
<dependency>
    <groupId>org.json</groupId>
    <artifactId>json</artifactId>
    <version>20170516</version>
</dependency>
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.2</version>
</dependency>
```

## 2、安装非开源jar包

以 1.4.9 版本为例，步骤如下:

1.下载Java示例代码VODUploadDemo-java-1.4.9.zip开发包(包含示例代码和所需jar包)，见 视频上传SDK下载；

2.将解压后lib目录下的所有jar文件拷贝至您的项目中;

3.SDK依赖的jar包版本说明

注意：以下列举出部分依赖jar包的版本，您可直接在您的项目中添加maven依赖，也可以将VODUploadDemo-java-1.4.9.zip包中的所有jar包引入您的项目中使用。其中 aliyun-java-vod-upload-1.4.9.jar 还未正式开源，请您直接引入jar包至您的项目中使用。

在本地Maven仓库中安装jar包：

下载视频上传SDK，解压，命令行进入lib目录，执行以下代码

```
mvn install:install-file -DgroupId=com.aliyun -DartifactId=aliyun-sdk-vod-upload -Dversion=1.4.11 -Dpackaging=jar -Dfile=aliyun-java-vod-upload-1.4.11.jar
```

然后在pom中引入jar包

```
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-sdk-vod-upload</artifactId>
    <version>1.4.11</version>
</dependency>
```

# 二、测试

## 1、创建测试文件

```
package com.atguigu.aliyunvod;
public class UploadTest {

    //账号AK信息请填写(必选)
    private static final String accessKeyId = "你的accessKeyId";
    //账号AK信息请填写(必选)
    private static final String accessKeySecret = "你的accessKeySecret";
}
```

## 2、测试本地文件上传

```
/**
     * 视频上传
     */
@Test
public void testUploadVideo(){

    //1.音视频上传-本地文件上传
    //视频标题(必选)
    String title = "3 - How Does Project Submission Work - upload by sdk";
    //本地文件上传和文件流上传时，文件名称为上传文件绝对路径，如:/User/sample/文件名称.mp4 (必选)
    //文件名必须包含扩展名
    String fileName = "E:/共享/资源/课程视频/3 - How Does Project Submission Work.mp4";
    //本地文件上传
    UploadVideoRequest request = new UploadVideoRequest(accessKeyId, accessKeySecret, title, fileName);
    /* 可指定分片上传时每个分片的大小，默认为1M字节 */
    request.setPartSize(1 * 1024 * 1024L);
    /* 可指定分片上传时的并发线程数，默认为1，(注：该配置会占用服务器CPU资源，需根据服务器情况指定）*/
    request.setTaskNum(1);
    /* 是否开启断点续传，默认断点续传功能关闭。当网络不稳定或者程序崩溃时，再次发起相同上传请求，可以继续未完成的上传任务，适用于超时3000秒仍不能上传完成的大文件。
        注意：断点续传开启后，会在上传过程中将上传位置写入本地磁盘文件，影响文件上传速度，请您根据实际情况选择是否开启*/
    request.setEnableCheckpoint(false);

    UploadVideoImpl uploader = new UploadVideoImpl();
    UploadVideoResponse response = uploader.uploadVideo(request);
    System.out.print("RequestId=" + response.getRequestId() + "\n");  //请求视频点播服务的请求ID
    if (response.isSuccess()) {
        System.out.print("VideoId=" + response.getVideoId() + "\n");
    } else {
        /* 如果设置回调URL无效，不影响视频上传，可以返回VideoId同时会返回错误码。其他情况上传失败时，VideoId为空，此时需要根据返回错误码分析具体错误原因 */
        System.out.print("VideoId=" + response.getVideoId() + "\n");
```

```
        System.out.print("ErrorCode=" + response.getCode() + "\n");
        System.out.print("ErrorMessage=" + response.getMessage() + "\n");
    }

}
```

# 一、创建视频点播微服务

## 1、创建微服务模块

Artifact：service-vod

## 2、pom

### （1）service-vod中引入依赖

```
<dependencies>
    <dependency>
        <groupId>com.aliyun</groupId>
        <artifactId>aliyun-java-sdk-core</artifactId>
    </dependency>
    <dependency>
        <groupId>com.aliyun.oss</groupId>
        <artifactId>aliyun-sdk-oss</artifactId>
    </dependency>
    <dependency>
        <groupId>com.aliyun</groupId>
        <artifactId>aliyun-java-sdk-vod</artifactId>
    </dependency>
    <dependency>
        <groupId>com.aliyun</groupId>
        <artifactId>aliyun-sdk-vod-upload</artifactId>
    </dependency>
    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>fastjson</artifactId>
    </dependency>
    <dependency>
        <groupId>org.json</groupId>
        <artifactId>json</artifactId>
    </dependency>
    <dependency>
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
    </dependency>

```

```
    <dependency>
        <groupId>joda-time</groupId>
        <artifactId>joda-time</artifactId>
    </dependency>
</dependencies>
```

# 3、application.properties

```
# 服务端口
server.port=8003
# 服务名
spring.application.name=service-vod

# 环境设置：dev、test、prod
spring.profiles.active=dev

#阿里云 vod
#不同的服务器，地址不同
aliyun.vod.file.keyid=your accessKeyId
aliyun.vod.file.keysecret=your accessKeySecret

# 最大上传单个文件大小：默认1M
spring.servlet.multipart.max-file-size=1024MB
# 最大置总上传的数据大小 ：默认10M
spring.servlet.multipart.max-request-size=1024MB
```

# 4、logback.xml

# 5、启动类

VodApplication.java

```
package com.guli.vod;

@SpringBootApplication(exclude = DataSourceAutoConfiguration.class)
```

```
@ComponentScan(basePackages={"com.atguigu"})
public class VodApplication {

    public static void main(String[] args) {
        SpringApplication.run(VodApplication.class, args);
    }
}
```

# 二、整合阿里云vod实现视频上传

## 1、创建常量类

ConstantPropertiesUtil.java

```
package com.guli.vod.util;

@Component
//@PropertySource("classpath:application.properties")
public class ConstantPropertiesUtil implements InitializingBean {

    @Value("${aliyun.vod.file.keyid}")
    private String keyId;

    @Value("${aliyun.vod.file.keysecret}")
    private String keySecret;

    public static String ACCESS_KEY_ID;
    public static String ACCESS_KEY_SECRET;

    @Override
    public void afterPropertiesSet() throws Exception {
        ACCESS_KEY_ID = keyId;
        ACCESS_KEY_SECRET = keySecret;
    }
}
```

## 2、复制工具类到项目中

AliyunVodSDKUtils.java

```
package com.guli.vod.util;

public class AliyunVodSDKUtils {
    public static DefaultAcsClient initVodClient(String accessKeyId, String accessKeySecret) throws ClientException {
        String regionId = "cn-shanghai";  // 点播服务接入区域
        DefaultProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKeySecret);
        DefaultAcsClient client = new DefaultAcsClient(profile);
        return client;
    }
}
```

# 3、配置swagger

web和admin

# 4、创建service

接口：VideoService.java

```
package com.guli.vod.service;
public interface VideoService {
    String uploadVideo(MultipartFile file);
}
```

实现：VideoServiceImpl.java

```
package com.guli.vod.service.impl;

@Service
public class VideoServiceImpl implements VideoService {

    @Override
    public String uploadVideo(MultipartFile file) {

```

```
        try {
            InputStream inputStream = file.getInputStream();
            String originalFilename = file.getOriginalFilename();
            String title = originalFilename.substring(0,
originalFilename.lastIndexOf("."));

            UploadStreamRequest request = new UploadStreamRequest(
                    ConstantPropertiesUtil.ACCESS_KEY_ID,
                    ConstantPropertiesUtil.ACCESS_KEY_SECRET,
                    title, originalFilename, inputStream);

            UploadVideoImpl uploader = new UploadVideoImpl();
            UploadStreamResponse response = uploader.uploadStream(request);

            //如果设置回调URL无效，不影响视频上传，可以返回VideoId同时会返回错误码。
            // 其他情况上传失败时，VideoId为空，此时需要根据返回错误码分析具体错误原因
            String videoId = response.getVideoId();
            if (!response.isSuccess()) {
                String errorMessage = "阿里云上传错误: " + "code: " +
response.getCode() + ", message: " + response.getMessage();
                log.warn(errorMessage);
                if(StringUtils.isEmpty(videoId)){
                    throw new GuliException(20001, errorMessage);
                }
            }

            return videoId;
        } catch (IOException e) {
            throw new GuliException(20001, "guli vod 服务上传失败");
        }
    }
}
```

# 5、创建controller

VideoAdminController.java

```
package com.guli.vod.controller.admin;

@Api(description="阿里云视频点播微服务")
@CrossOrigin //跨域
```

```java
@RestController
@RequestMapping("/admin/vod/video")
public class VideoAdminController {

    @Autowired
    private VideoService videoService;

    @PostMapping("upload")
    public R uploadVideo(
            @ApiParam(name = "file", value = "文件", required = true)
            @RequestParam("file") MultipartFile file) throws Exception {

        String videoId = videoService.uploadVideo(file);
        return R.ok().message("视频上传成功").data("videoId", videoId);
    }
}
```

## 6、启动后端vod微服务

## 7、swagger测试

# 一、配置nginx反向代理

将接口地址加入nginx配置

```
location ~ /vod/ {
    proxy_pass http://localhost:8003;
}
```

配置nginx上传文件大小，否则上传时会有 413 (Request Entity Too Large) 异常

打开nginx主配置文件nginx.conf，找到http{}，添加

```
client_max_body_size 1024m;
```

重启nginx

```
nginx -s reload
```

# 二、前端实现

## 1、数据定义

```
fileList: [],//上传文件列表
BASE_API: process.env.BASE_API // 接口API地址
```

## 2、整合上传组件

```
<el-form-item label="上传视频">
    <el-upload
            :on-success="handleVodUploadSuccess"
            :on-remove="handleVodRemove"
```

```
            :before-remove="beforeVodRemove"
            :on-exceed="handleUploadExceed"
            :file-list="fileList"
            :action="BASE_API+'/admin/vod/video/upload'"
            :limit="1"
            class="upload-demo">
    <el-button size="small" type="primary">上传视频</el-button>
    <el-tooltip placement="right-end">
        <div slot="content">最大支持1G，<br>
            支持3GP、ASF、AVI、DAT、DV、FLV、F4V、<br>
            GIF、M2T、M4V、MJ2、MJPEG、MKV、MOV、MP4、<br>
            MPE、MPG、MPEG、MTS、OGG、QT、RM、RMVB、<br>
            SWF、TS、VOB、WMV、WEBM 等视频格式上传</div>
        <i class="el-icon-question"/>
    </el-tooltip>
    </el-upload>
</el-form-item>
```

## 3、方法定义

```
//成功回调
handleVodUploadSuccess(response, file, fileList) {
  this.video.videoSourceId = response.data.videoId
},
//视图上传多于一个视频
handleUploadExceed(files, fileList) {
  this.$message.warning('想要重新上传视频，请先删除已上传的视频')
},
```

文档：服务端SDK->Java SDK->媒资管理

https://help.aliyun.com/document_detail/61065.html?spm=a2c4g.11186623.6.831.654b3815cIxvma#h2--div-id-deletevideo-div-7

# 一、后端

## 1、service

接口

```
void removeVideo(String videoId);
```

实现

```
@Override
public void removeVideo(String videoId) {
    try{
        DefaultAcsClient client = AliyunVodSDKUtils.initVodClient(
            ConstantPropertiesUtil.ACCESS_KEY_ID,
            ConstantPropertiesUtil.ACCESS_KEY_SECRET);
        DeleteVideoRequest request = new DeleteVideoRequest();
        request.setVideoIds(videoId);
        DeleteVideoResponse response = client.getAcsResponse(request);
        System.out.print("RequestId = " + response.getRequestId() + "\n");
    }catch (ClientException e){
        throw new GuliException(20001, "视频删除失败");
    }
}
```

## 2、controller

```
@DeleteMapping("{videoId}")
public R removeVideo(@ApiParam(name = "videoId", value = "云端视频id", required
= true)
```

```
                    @PathVariable String videoId){
    videoService.removeVideo(videoId);
    return R.ok().message("视频删除成功");
}
```

# 二、前端

## 1、定义api

api/edu/vod.js

```
import request from '@/utils/request'
const api_name = '/admin/vod/video'
export default {
  removeById(id) {
    return request({
      url: `${api_name}/${id}`,
      method: 'delete'
    })
  }
}
```

## 2、组件方法

views/edu/course/chapter.vue

```
import vod from '@/api/edu/vod'
```

```
beforeVodRemove(file, fileList) {
  return this.$confirm(`确定移除 ${file.name}？`)
},
handleVodRemove(file, fileList) {
  console.log(file)
  vod.removeById(this.video.videoSourceId).then(response=>{
    this.$message({
      type: 'success',
```

```
      message: response.message
    })
  })
},
```

# 一、数据库优化冗余字段

## 1、video表中添加一列

video_original_name varchar 100 原始文件名称

## 2、pojo中定义新增字段

Video.java、VideoVo.java、VideoForm.java

```
@ApiModelProperty(value = "云服务器上存储的视频文件名称")
private String videoOriginalName;
```

# 二、前端修改

## 1、chapter.vue

添加videoOriginalName的数据定义

```
video: {// 课时对象
  title: '',
  sort: 0,
  free: false,
  videoSourceId: '',
  videoOriginalName: ''
},
```

## 2、上传成功回调

添加对videoOriginalName的赋值

```
handleVodUploadSuccess(response, file, fileList) {
    this.video.videoSourceId = response.data.videoId
```

```
    this.video.videoOriginalName = file.name;
},
```

## 3、修改回调Video

```
editVideo(videoId) {
    this.dialogVideoFormVisible = true
        video.getVideoInfoById(videoId).then(response => {
        this.video = response.data.item
        this.fileList = [{'name': this.video.videoOriginalName}]
    })
},
```

## 4、删除云端video回调

```
handleVodRemove(file, fileList) {
    console.log(file)
    vod.removeById(this.video.videoSourceId).then(response => {
        this.video.videoSourceId = ''
        this.video.videoOriginalName = ''
        this.fileList = []
        this.$message({
            type: 'success',
            message: response.message
        })
    })
},
```

# 一、什么是微服务

## 1、微服务的由来

微服务最早由Martin Fowler与James Lewis于2014年共同提出，微服务架构风格是一种使用一套小服务来开发单个应用的方式途径，每个服务运行在自己的进程中，并使用轻量级机制通信，通常是HTTP API，这些服务基于业务能力构建，并能够通过自动化部署机制来独立部署，这些服务使用不同的编程语言实现，以及不同数据存储技术，并保持最低限度的集中式管理。

## 2、为什么需要微服务

在传统的IT行业软件大多都是各种独立系统的堆砌，这些系统的问题总结来说就是扩展性差，可靠性不高，维护成本高。到后面引入了SOA服务化，但是，由于 SOA 早期均使用了总线模式，这种总线模式是与某种技术栈强绑定的，比如：J2EE。这导致很多企业的遗留系统很难对接，切换时间太长，成本太高，新系统稳定性的收敛也需要一些时间。

## 3、微服务与单体架构区别

（1）单体架构所有的模块全都耦合在一块，代码量大，维护困难。

微服务每个模块就相当于一个单独的项目，代码量明显减少，遇到问题也相对来说比较好解决。

（2）单体架构所有的模块都共用一个数据库，存储方式比较单一。

微服务每个模块都可以使用不同的存储方式（比如有的用redis，有的用mysql等），数据库也是单个模块对应自己的数据库。

（3）单体架构所有的模块开发所使用的技术一样。

微服务每个模块都可以使用不同的开发技术，开发模式更灵活。

## 4、微服务本质

（1）微服务，关键其实不仅仅是微服务本身，而是系统要提供一套基础的架构，这种架构使得微服务可以独立的部署、运行、升级，不仅如此，这个系统架构还让微服务与微服务之间在结构上“松耦合”，而在功能上则表现为一个统一的整体。这种所谓的“统一的整体”表现出来的是统一风格的界面，统一的权限管理，统一的安全策略，统一的上线过程，统一的日志和审计方法，统一的调度方式，统一的访问入口等

等。

（2）微服务的目的是有效的拆分应用，实现敏捷开发和部署 。

（3）微服务提倡的理念团队间应该是 inter-operate, not integrate 。inter-operate是定义好系统的边界和接口，在一个团队内全栈，让团队自治，原因就是因为如果团队按照这样的方式组建，将沟通的成本维持在系统内部，每个子系统就会更加内聚，彼此的依赖耦合能变弱，跨系统的沟通成本也就能降低。

## 5、什么样的项目适合微服务

微服务可以按照业务功能本身的独立性来划分，如果系统提供的业务是非常底层的，如：操作系统内核、存储系统、网络系统、数据库系统等等，这类系统都偏底层，功能和功能之间有着紧密的配合关系，如果强制拆分为较小的服务单元，会让集成工作量急剧上升，并且这种人为的切割无法带来业务上的真正的隔离，所以无法做到独立部署和运行，也就不适合做成微服务了。

## 6、微服务开发框架

目前微服务的开发框架，最常用的有以下四个：

Spring Cloud：http://projects.spring.io/spring-cloud（现在非常流行的微服务架构）

Dubbo：http：//dubbo.io

Dropwizard：http://www.dropwizard.io （关注单个微服务的开发）

Consul、etcd&etc.（微服务的模块）

## 7、什么是Spring Cloud

Spring Cloud是一系列框架的集合。它利用Spring Boot的开发便利性简化了分布式系统基础设施的开发，如服务发现、服务注册、配置中心、消息总线、负载均衡、 熔断器、数据监控等，都可以用Spring Boot的开发风格做到一键启动和部署。Spring并没有重复制造轮子，它只是将目前各家公司开发的比较成熟、经得起实际考验的服务框架组合起来，通过SpringBoot风格进行再封装屏蔽掉了复杂的配置和实现原理，最终给开发者留出了一套简单易懂、易部署和易维护的分布式系统开发工具包

## 8、Spring Cloud和Spring Boot是什么关系

Spring Boot 是 Spring 的一套快速配置脚手架，可以基于Spring Boot 快速开发单个微服务，Spring Cloud是一个基于Spring Boot实现的开发工具；Spring Boot专注于快速、方便集成的单个微服务个体，Spring Cloud关注全局的服务治理框架； Spring Boot使用了默认大于配置的理念，很多集成方案已

经帮你选择好了，能不配置就不配置，Spring Cloud很大的一部分是基于Spring Boot来实现，必须基于Spring Boot开发。可以单独使用Spring Boot开发项目，但是Spring Cloud离不开 Spring Boot。

## 9、Spring Cloud相关基础服务组件

服务发现——Netflix Eureka （Nacos）

服务调用——Netflix Feign

熔断器——Netflix Hystrix

服务网关——Spring Cloud GateWay

分布式配置——Spring Cloud Config （Nacos）

消息总线 —— Spring Cloud Bus （Nacos）

## 10、Spring Cloud的版本

Spring Cloud并没有熟悉的数字版本号，而是对应一个开发代号。

| **Cloud**代号 | **Boot**版本**(train)** | **Boot**版本**(tested)** | **lifecycle** |
|---|---|---|---|
| Angle | 1.2.x | incompatible with 1.3 | EOL in July 2017 |
| Brixton | 1.3.x | 1.4.x | 2017-07卒 |
| Camden | 1.4.x | 1.5.x | - |
| Dalston | 1.5.x | not expected 2.x | - |
| Edgware | 1.5.x | not expected 2.x | - |
| Finchley | 2.0.x | not expected 1.5.x | - |
| Greenwich | **2.1.x** | | |
| Hoxton | 2.2.x | | |

开发代号看似没有什么规律，但实际上首字母是有顺序的，比如：Dalston版本，我们可以简称 D 版本，对应的 Edgware 版本我们可以简称 E 版本。

### 小版本

Spring Cloud 小版本分为:

SNAPSHOT： 快照版本，随时可能修改

M： MileStone，M1表示第1个里程碑版本，一般同时标注PRE，表示预览版版。

SR： Service Release，SR1表示第1个正式版本，一般同时标注GA：(GenerallyAvailable),表示稳定版本。

# 一、Nacos

## 1、基本概念

（**1**）Nacos 是阿里巴巴推出来的一个新开源项目，是一个更易于构建云原生应用的动态服务发现、配置管理和服务管理平台。Nacos 致力于帮助您发现、配置和管理微服务。Nacos 提供了一组简单易用的特性集，帮助您快速实现动态服务发现、服务配置、服务元数据及流量管理。Nacos 帮助您更敏捷和容易地构建、交付和管理微服务平台。 Nacos 是构建以“服务”为中心的现代应用架构 (例如微服务范式、云原生范式) 的服务基础设施。

**（2）**常见的注册中心：

1. Eureka（原生，2.0遇到性能瓶颈，停止维护）

2. Zookeeper（支持，专业的独立产品。例如：dubbo）

3. Consul（原生，GO语言开发）

4. Nacos

相对于 Spring Cloud Eureka 来说，Nacos 更强大。Nacos = Spring Cloud Eureka + Spring Cloud Config

Nacos 可以与 Spring, Spring Boot, Spring Cloud 集成，并能代替 Spring Cloud Eureka, Spring Cloud Config

- 通过 Nacos Server 和 spring-cloud-starter-alibaba-nacos-discovery 实现服务的注册与发现。

（**3**）Nacos是以服务为主要服务对象的中间件，Nacos支持所有主流的服务发现、配置和管理。

Nacos主要提供以下四大功能：

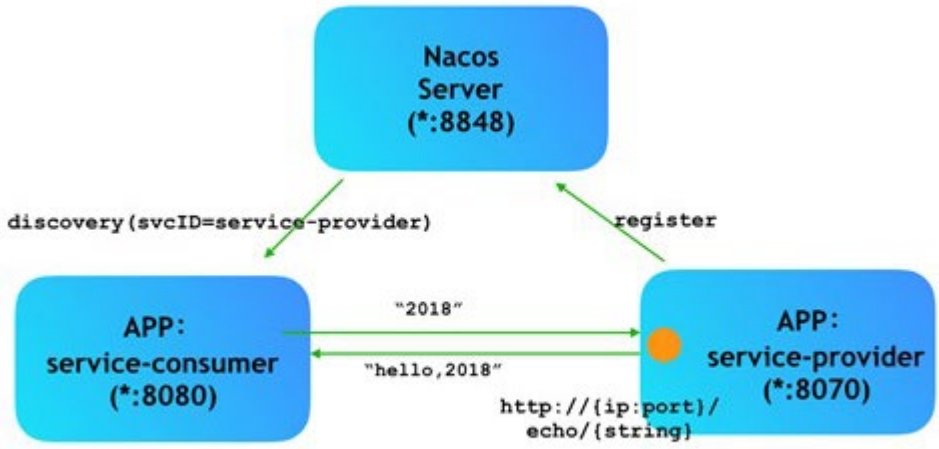
1. 服务发现和服务健康监测

2. 动态配置服务

3. 动态DNS服务

4. 服务及其元数据管理

（**4**）Nacos结构图




# 2、Nacos下载和安装

（**1**）下载地址和版本

下载地址：https://github.com/alibaba/nacos/releases

下载版本：nacos-server-1.1.4.tar.gz或nacos-server-1.1.4.zip，解压任意目录即可

（**2**）启动**nacos**服务

- Linux/Unix/Mac

启动命令(standalone代表着单机模式运行，非集群模式)

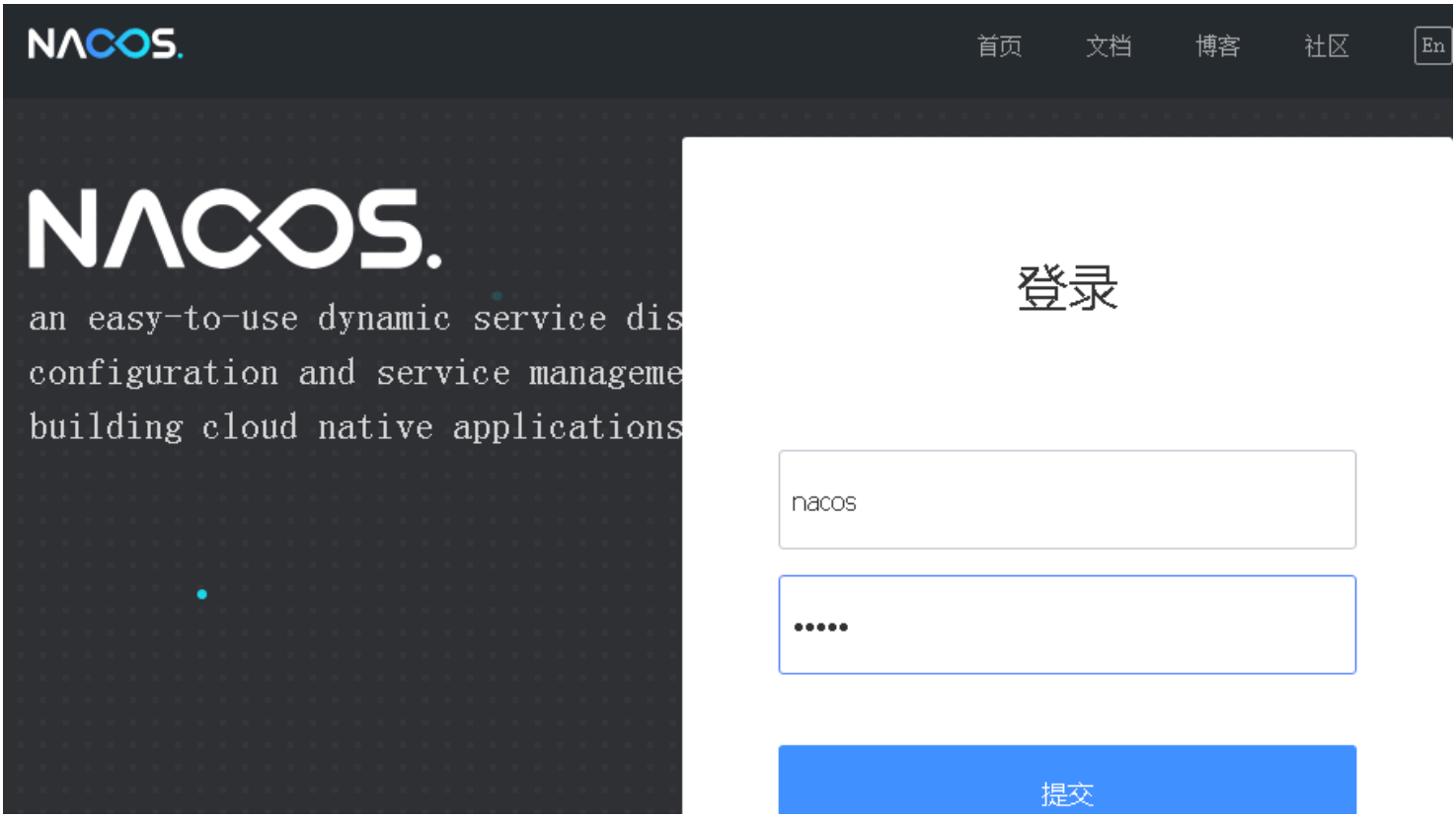
启动命令：sh startup.sh -m standalone

- Windows

启动命令：cmd startup.cmd 或者双击startup.cmd运行文件。

访问：http://localhost:8848/nacos

用户名密码：nacos/nacos





# 二、服务注册（service_edu为例）

把service-edu微服务注册到注册中心中，service-vod步骤相同

# 1、在service模块配置pom

配置Nacos客户端的pom依赖

```
<!--服务注册-->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
</dependency>
```

# 2、添加服务配置信息

配置application.properties，在客户端微服务中添加注册Nacos服务的配置信息

```
# nacos服务地址
spring.cloud.nacos.discovery.server-addr=127.0.0.1:8848
```

# 3、添加Nacos客户端注解

在客户端微服务启动类中添加注解

```
@EnableDiscoveryClient
```

# 4、启动客户端微服务

启动注册中心

启动已注册的微服务，可以在Nacos服务列表中看到被注册的微服务


NACOS
首页
文档
博客
社区
En
nacos
NACOS 1.1.4
public
配置管理
配置列表
历史版本
监听查询
服务管理
服务列表
订阅者列表
命名空间
集群管理
服务列表 | public
服务名称
请输入服务名称
分组名称
请输入分组名称
隐藏空服务:
查询
创建服务
服务名
分组名称
集群数目
实例数
健康实例数
触发保护阈值
操作
service-edu
DEFAULT_GROUP
1
1
1
false
详情 | 示例代码 | 删除

# 一、Feign

## 1、基本概念

- Feign是Netflix开发的声明式、模板化的HTTP客户端， Feign可以帮助我们更快捷、优雅地调用HTTP API。
- Feign支持多种注解，例如Feign自带的注解或者JAX-RS注解等。
- Spring Cloud对Feign进行了增强，使Feign支持了Spring MVC注解，并整合了Ribbon和Eureka，从而让Feign的使用更加方便。
- Spring Cloud Feign是基于Netflix feign实现，整合了Spring Cloud Ribbon和Spring Cloud Hystrix，除了提供这两者的强大功能外，还提供了一种声明式的Web服务客户端定义的方式。
- Spring Cloud Feign帮助我们定义和实现依赖服务接口的定义。在Spring Cloud feign的实现下，只需要创建一个接口并用注解方式配置它，即可完成服务提供方的接口绑定，简化了在使用Spring Cloud Ribbon时自行封装服务调用客户端的开发量。

# 二、实现服务调用

## 1、需求

删除课时的同时删除云端视频

## 2、在service模块添加pom依赖

```
<!--服务调用-->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>
```

## 3、在调用端的启动类添加注解

```
@EnableFeignClients
```

## 4、创建包和接口

创建client包

@FeignClient注解用于指定从哪个服务中调用功能 ， 名称与被调用的服务名保持一致。

@GetMapping注解用于对被调用的微服务进行地址映射。

@PathVariable注解一定要指定参数名称，否则出错

@Component注解防止，在其他位置注入CodClient时idea报错

```
package com.guli.edu.client;

@FeignClient("service-vod")
@Component
public interface VodClient {
    @DeleteMapping(value = "/eduvod/vod/video/{videoId}")
    public R removeVideo(@PathVariable("videoId") String videoId);
}
```

## 5、调用微服务

在调用端的`VideoServiceImpl`中调用client中的方法

```
@Override
public boolean removeVideoById(String id) {

    //查询云端视频id
    Video video = baseMapper.selectById(id);
    String videoSourceId = video.getVideoSourceId();
    //删除视频资源
    if(!StringUtils.isEmpty(videoSourceId)){
        vodClient.removeVideo(videoSourceId);
    }

    Integer result = baseMapper.deleteById(id);
    return null != result && result > 0;
}
```

# 6、测试

启动相关微服务

测试删除课时的功能

**需求**

删除课程的同时删除云端视频

# 一、vod服务

## 1、业务

业务接口：VideoService.java

```
void removeVideoList(List<String> videoIdList);
```

业务实现：VideoServiceImpl.java

```
@Override
public void removeVideoList(List<String> videoIdList) {
    try {
        //初始化
        DefaultAcsClient client = AliyunVodSDKUtils.initVodClient(
            ConstantPropertiesUtil.ACCESS_KEY_ID,
            ConstantPropertiesUtil.ACCESS_KEY_SECRET);

        //创建请求对象
        //一次只能批量删20个
        String str =
org.apache.commons.lang.StringUtils.join(videoIdList.toArray(), ",");
        DeleteVideoRequest request = new DeleteVideoRequest();
        request.setVideoIds(str);

        //获取响应
        DeleteVideoResponse response = client.getAcsResponse(request);

        System.out.print("RequestId = " + response.getRequestId() + "\n");

    } catch (ClientException e) {
        throw new GuliException(20001, "视频删除失败");
    }
}
```

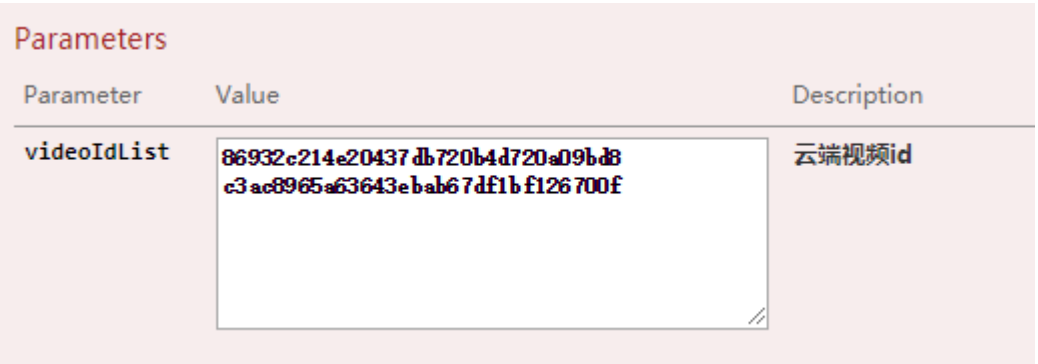
## 2、web层接口

controller：VideoAdminController.java

```
/**
     * 批量删除视频
     * @param videoIdList
     * @return
     */
@DeleteMapping("delete-batch")
public R removeVideoList(
    @ApiParam(name = "videoIdList", value = "云端视频id", required = true)
    @RequestParam("videoIdList") List videoIdList){

    videoService.removeVideoList(videoIdList);
    return R.ok().message("视频删除成功");
}
```

## 3、Swagger测试

输入多个id，每个一行




# 二、edu服务

## 1、client

VodClient.java

```
@DeleteMapping(value = "/admin/vod/video/delete-batch")
```

```
public R removeVideoList(@RequestParam("videoIdList") List<String> videoIdList);
```

## 2、业务

VideoServiceImpl.java

```
@Override
public boolean removeByCourseId(String courseId) {

    //根据课程id查询所有视频列表
    QueryWrapper<Video> queryWrapper = new QueryWrapper<>();
    queryWrapper.eq("course_id", courseId);
    queryWrapper.select("video_source_id");
    List<Video> videoList = baseMapper.selectList(queryWrapper);

    //得到所有视频列表的云端原始视频id
    List<String> videoSourceIdList = new ArrayList<>();
    for (int i = 0; i < videoList.size(); i++) {
        Video video = videoList.get(i);
        String videoSourceId = video.getVideoSourceId();
        if(!StringUtils.isEmpty(videoSourceId)){
            videoSourceIdList.add(videoSourceId);
        }
    }

    //调用vod服务删除远程视频
    if(videoSourceIdList.size() > 0){
        vodClient.removeVideoList(videoSourceIdList);
    }

    //删除video表示的记录
    QueryWrapper<Video> queryWrapper2 = new QueryWrapper<>();
    queryWrapper2.eq("course_id", courseId);
    Integer count = baseMapper.delete(queryWrapper2);
    return null != count && count > 0;
}

```

CourseServiceImpl.java

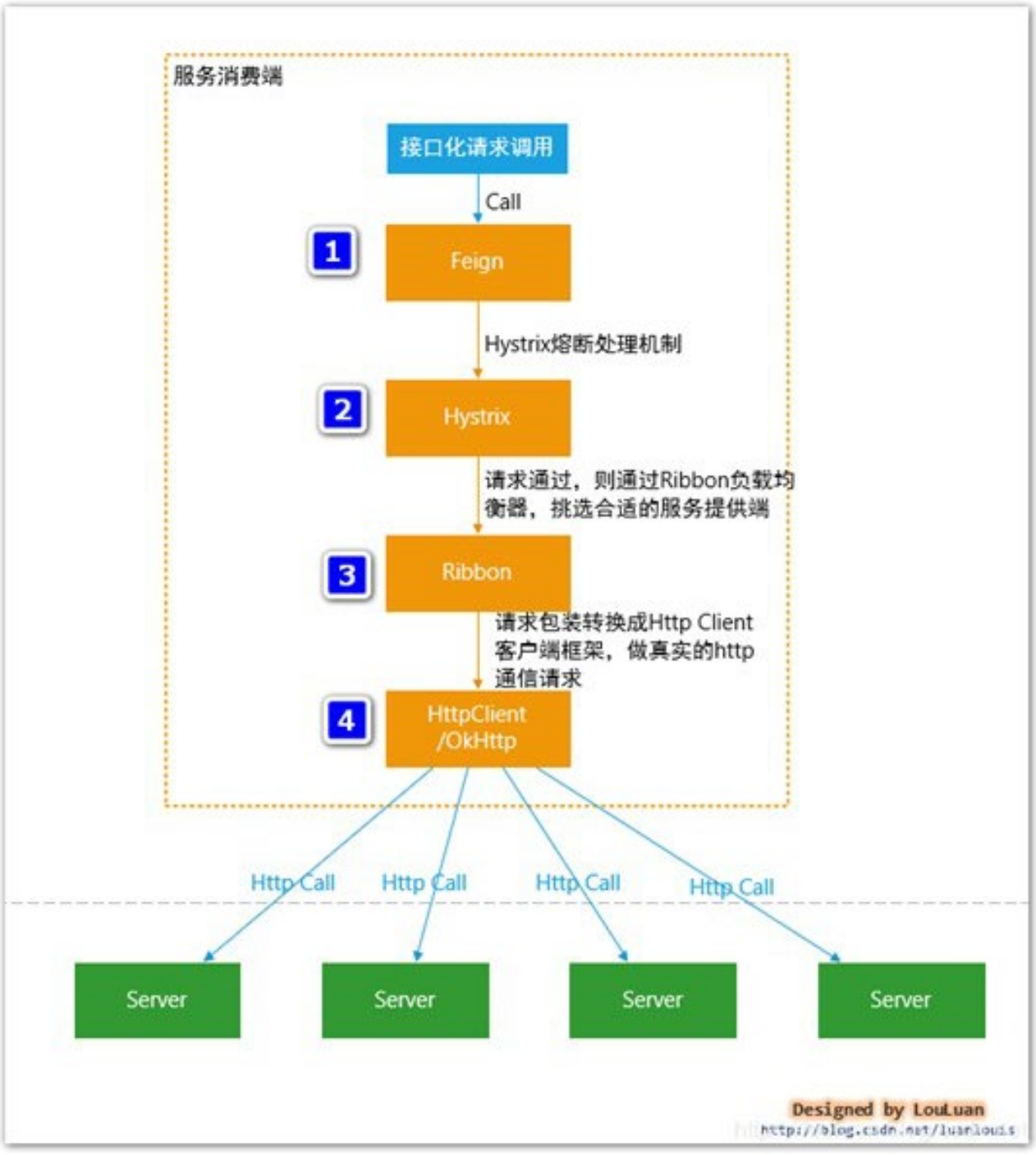
```java
@Override
public boolean removeCourseById(String id) {

    //根据id删除所有视频
    videoService.removeByCourseId(id);

    //根据id删除所有章节
    chapterService.removeByCourseId(id);

    //根据id删除所有课程详情
    courseDescriptionService.removeById(id);

    //删除封面 TODO 独立完成

    Integer result = baseMapper.deleteById(id);
    return null != result && result > 0;
}

```

# 一、Hystrix基本概念

## 1、Spring Cloud调用接口过程

Spring Cloud 在接口调用上，大致会经过如下几个组件配合：

Feign ----->Hystrix —>Ribbon —>Http Client（apache http components 或者 Okhttp） 具体交互流程上，如下图所示：




（1）接口化请求调用当调用被@FeignClient注解修饰的接口时，在框架内部，将请求转换成Feign的请求实例feign.Request，交由Feign框架处理。

(2) **Feign**：转化请求Feign是一个http请求调用的轻量级框架，可以以Java接口注解的方式调用Http请求，封装了Http调用流程。

(3) **Hystrix**：熔断处理机制 Feign的调用关系，会被Hystrix代理拦截，对每一个Feign调用请求，Hystrix都会将其包装成HystrixCommand,参与Hystrix的流控和熔断规则。如果请求判断需要熔断，则Hystrix直接熔断，抛出异常或者使用FallbackFactory返回熔断Fallback结果；如果通过，则将调用请求传递给Ribbon组件。

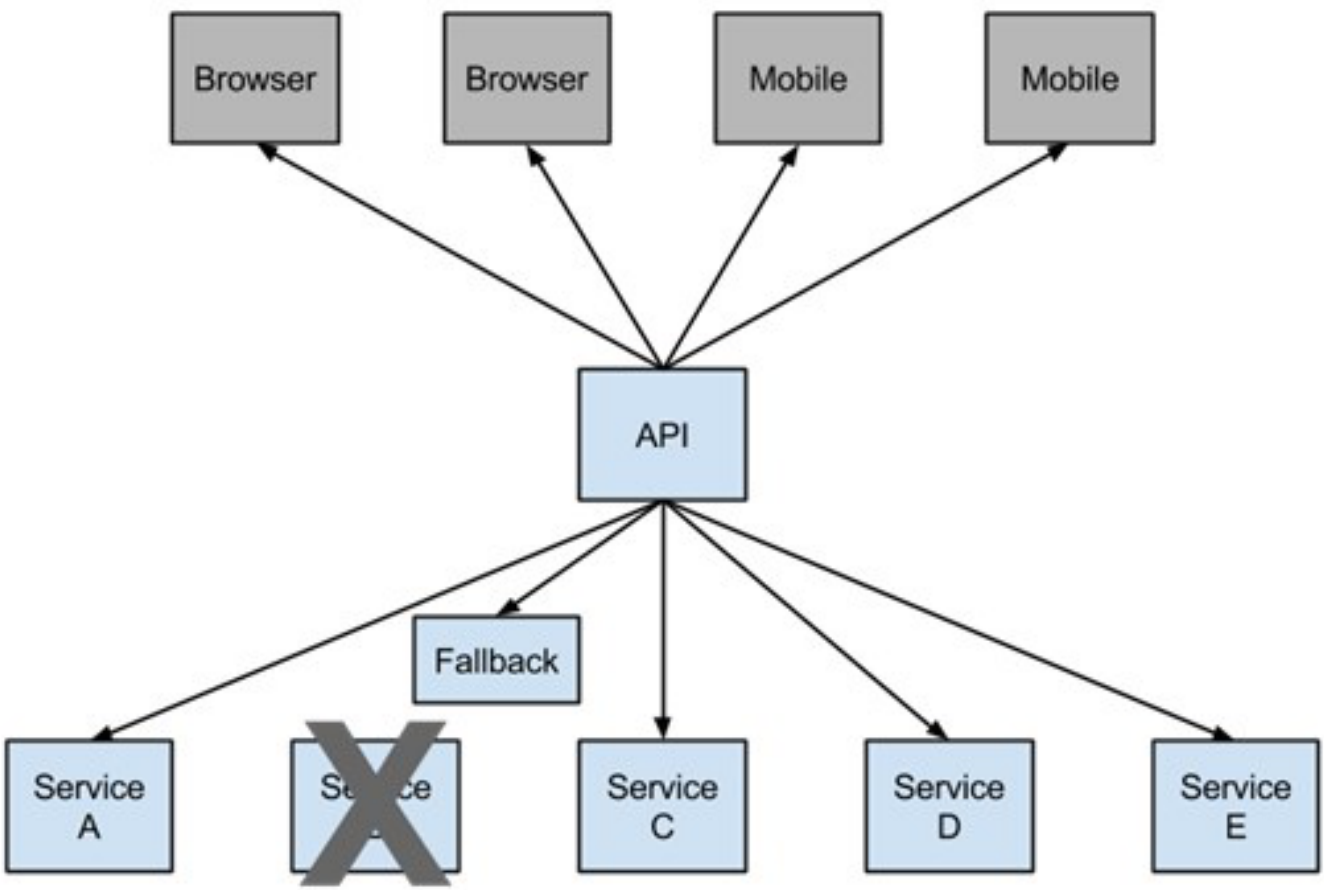
(4) **Ribbon**：服务地址选择 当请求传递到Ribbon之后,Ribbon会根据自身维护的服务列表，根据服务的服务质量，如平均响应时间，Load等，结合特定的规则，从列表中挑选合适的服务实例，选择好机器之后，然后将机器实例的信息请求传递给Http Client客户端，HttpClient客户端来执行真正的Http接口调用；

(5) **HttpClient**：Http客户端，真正执行Http调用根据上层Ribbon传递过来的请求，已经指定了服务地址，则HttpClient开始执行真正的Http请求

## 2、Hystrix概念

Hystrix 是一个供分布式系统使用，提供延迟和容错功能，保证复杂的分布系统在面临不可避免的失败时，仍能有其弹性。

比如系统中有很多服务，当某些服务不稳定的时候，使用这些服务的用户线程将会阻塞，如果没有隔离机制，系统随时就有可能会挂掉，从而带来很大的风险。SpringCloud使用Hystrix组件提供断路器、资源隔离与自我修复功能。下图表示服务B触发了断路器，阻止了级联失败



# 二、feign结合Hystrix使用

## 改造service-edu模块

## 1、在service的pom中添加依赖

```
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-ribbon</artifactId>
        </dependency>

        <!--hystrix依赖，主要是用  @HystrixCommand -->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
        </dependency>

        <!--服务注册-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
        </dependency>
        <!--服务调用-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-openfeign</artifactId>
        </dependency>
```

## 2、在配置文件中添加hystrix配置

```

#开启熔断机制
feign.hystrix.enabled=true
# 设置hystrix超时时间，默认1000ms
hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds=6000
```

## 3、在service-edu的client包里面创建熔断器的实现类

```
@Component
public class VodFileDegradeFeignClient implements VodClient {
    @Override
    public R removeVideo(String videoId) {
        return R.error().message("time out");
    }

    @Override
    public R removeVideoList(List videoIdList) {
        return R.error().message("time out");
    }
}
```

## 4、修改VodClient接口的注解

```
@FeignClient(name = "service-vod", fallback = VodFileDegradeFeignClient.class)
@Component
public interface VodClient {
    @DeleteMapping(value = "/eduvod/vod/{videoId}")
    public R removeVideo(@PathVariable("videoId") String videoId);

    @DeleteMapping(value = "/eduvod/vod/delete-batch")
    public R removeVideoList(@RequestParam("videoIdList") List videoIdList);
}
```

## 5、测试熔断器效果

# 一、服务端渲染技术NUXT

## 1、什么是服务端渲染

服务端渲染又称SSR (Server Side Render)是在服务端完成页面的内容，而不是在客户端通过AJAX获取数据。

服务器端渲染(SSR)的优势主要在于：**更好的 SEO**，由于搜索引擎爬虫抓取工具可以直接查看完全渲染的页面。

如果你的应用程序初始展示 loading 菊花图，然后通过 Ajax 获取内容，抓取工具并不会等待异步完成后再进行页面内容的抓取。也就是说，如果 SEO 对你的站点至关重要，而你的页面又是异步获取内容，则你可能需要服务器端渲染(SSR)解决此问题。

另外，使用服务器端渲染，我们可以获得更快的内容到达时间(time-to-content)，无需等待所有的 JavaScript 都完成下载并执行，产生更好的用户体验，对于那些「内容到达时间(time-to-content)与转化率直接相关」的应用程序而言，服务器端渲染(SSR)至关重要。

## 2、什么是NUXT

Nuxt.js 是一个基于 Vue.js 的轻量级应用框架,可用来创建服务端渲染 (SSR) 应用,也可充当静态站点引擎生成静态站点应用,具有优雅的代码结构分层和热加载等特性。

官网网站：

https://zh.nuxtjs.org/

# 二、NUXT环境初始化

## 1、下载压缩包

https://github.com/nuxt-community/starter-template/archive/master.zip

## 2、解压

将template中的内容复制到 guli

## 3、安装ESLint

将guli-admin项目下的.eslintrc.js配置文件复制到当前项目下

# 4、修改package.json

name、description、author（必须修改这里，否则项目无法安装）

```
"name": "guli",
"version": "1.0.0",
"description": "谷粒学院前台网站",
"author": "Helen <55317332@qq.com>",
```

# 5、修改nuxt.config.js

修改title: '{{ name }}'、content: '{{escape description }}'

这里的设置最后会显示在页面标题栏和meta数据中

```
head: {
    title: '谷粒学院 - Java视频|HTML5视频|前端视频|Python视频|大数据视频-自学拿1万+月薪的IT在线视频课程，谷粉力挺，老学员为你推荐',
    meta: [
      { charset: 'utf-8' },
      { name: 'viewport', content: 'width=device-width, initial-scale=1' },
      { hid: 'keywords', name: 'keywords', content: '谷粒学院,IT在线视频教程,Java视频,HTML5视频,前端视频,Python视频,大数据视频' },
      { hid: 'description', name: 'description', content: '谷粒学院是国内领先的IT在线视频学习平台、职业教育平台。截止目前,谷粒学院线上、线下学习人次数以万计！会同上百个知名开发团队联合制定的Java、HTML5前端、大数据、Python等视频课程，被广大学习者及IT工程师誉为:业界最适合自学、代码量最大、案例最多、实战性最强、技术最前沿的IT系列视频课程！ ' }
    ],
    link: [
      { rel: 'icon', type: 'image/x-icon', href: '/favicon.ico' }
    ]
},
```

# 6、在命令提示终端中进入项目目录

# 7、安装依赖

```
npm install
```

```
e:\work\test0826\edu-front0826>npm install
npm WARN deprecated core-js@2.6.11: core-js@<3 is no longer maintained and not recommended for usage due to the number of is
 Please, upgrade your dependencies to the actual version of core-js@3.
[ ..................] \ fetchMetadata: WARN deprecated core-js@2.6.11: core-js@<3 is no longer maintained and not recommended
```

## 8、测试运行

```
npm run dev
```

## 9、NUXT目录结构

（1）资源目录 assets

用于组织未编译的静态资源如 LESS、SASS 或 JavaScript。

（2）组件目录 components

用于组织应用的 Vue.js 组件。Nuxt.js 不会扩展增强该目录下 Vue.js 组件，即这些组件不会像页面组件那样有 asyncData 方法的特性。

（3）布局目录 layouts

用于组织应用的布局组件。

（4）页面目录 pages

用于组织应用的路由及视图。Nuxt.js 框架读取该目录下所有的 .vue 文件并自动生成对应的路由配置。

（5）插件目录 plugins

用于组织那些需要在 根vue.js应用 实例化之前需要运行的 Javascript 插件。

（6）nuxt.config.js 文件

nuxt.config.js 文件用于组织Nuxt.js 应用的个性化配置，以便覆盖默认配置。

# 三、幻灯片插件

## 1、安装插件

```
npm install vue-awesome-swiper
```

## 2、配置插件

在 plugins 文件夹下新建文件 nuxt-swiper-plugin.js，内容是

```
import Vue from 'vue'
import VueAwesomeSwiper from 'vue-awesome-swiper/dist/ssr'

Vue.use(VueAwesomeSwiper)
```

在 nuxt.config.js 文件中配置插件

将 plugins 和 css节点 复制到 module.exports节点下

```
module.exports = {
  // some nuxt config...
  plugins: [
    { src: '~/plugins/nuxt-swiper-plugin.js', ssr: false }
  ],

  css: [
    'swiper/dist/css/swiper.css'
  ]
}
```

# 一、页面布局

## 1、复制静态资源

将静态原型中的css、img、js、photo目录拷贝至assets目录下

将favicon.ico复制到static目录下

## 2、定义布局

我们可以把页头和页尾提取出来，形成布局页

修改layouts目录下default.vue，从静态页面中复制首页，修改了原始文件中的资源路径为~/assets/，将主内容区域的内容替换成<nuxt />

内容如下：

```
<template>
  <div>
    <!-- 页头部分 -->

    <!-- 内容的区域 -->
    <nuxt/>

    <!-- 页尾部分 -->
  </div>
</template>
```

完整的内容如下

```
<template>
  <div class="in-wrap">
    <!-- 公共头引入 -->
    <header id="header">
      <section class="container">
        <h1 id="logo">
          <a href="#" title="谷粒学院">
            <img src="~/assets/img/logo.png" width="100%" alt="谷粒学院">
          </a>
```

```
      </h1>
      <div class="h-r-nsl">
        <ul class="nav">
          <router-link to="/" tag="li" active-class="current" exact>
            <a>首页</a>
          </router-link>
          <router-link to="/course" tag="li" active-class="current">
            <a>课程</a>
          </router-link>
          <router-link to="/teacher" tag="li" active-class="current">
            <a>名师</a>
          </router-link>
          <router-link to="/article" tag="li" active-class="current">
            <a>文章</a>
          </router-link>
          <router-link to="/qa" tag="li" active-class="current">
            <a>问答</a>
          </router-link>
        </ul>
        <!-- / nav -->
        <ul class="h-r-login">
          <li id="no-login">
            <a href="/sing_in" title="登录">
              <em class="icon18 login-icon"> </em>
              <span class="vam ml5">登录</span>
            </a>
            |
            <a href="/sign_up" title="注册">
              <span class="vam ml5">注册</span>
            </a>
          </li>
          <li class="mr10 undis" id="is-login-one">
            <a href="#" title="消息" id="headerMsgCountId">
              <em class="icon18 news-icon"> </em>
            </a>
            <q class="red-point" style="display: none"> </q>
          </li>
          <li class="h-r-user undis" id="is-login-two">
            <a href="#" title>
              <img
                src="~/assets/img/avatar-boy.gif"
                width="30"
                height="30"
```

```
                  class="vam picImg"
                  alt
                >
                <span class="vam disIb" id="userName"></span>
              </a>
              <a href="javascript:void(0)" title="退出" onclick="exit();"
class="ml5">退出</a>
            </li>
            <!-- /未登录显示第1 li; 登录后显示第2, 3 li -->
          </ul>
          <aside class="h-r-search">
            <form action="#" method="post">
              <label class="h-r-s-box">
                <input type="text" placeholder="输入你想学的课程"
name="queryCourse.courseName" value>
                <button type="submit" class="s-btn">
                  <em class="icon18"> </em>
                </button>
              </label>
            </form>
          </aside>
        </div>
        <aside class="mw-nav-btn">
          <div class="mw-nav-icon"></div>
        </aside>
        <div class="clear"></div>
      </section>
    </header>
    <!-- /公共头引入 -->

    <nuxt/>

    <!-- 公共底引入 -->
    <footer id="footer">
      <section class="container">
        <div class>
          <h4 class="hLh30">
            <span class="fsize18 f-fM c-999">友情链接</span>
          </h4>
          <ul class="of flink-list">
            <li>
              <a href="http://www.atguigu.com/" title="尚硅谷" target="_blank">尚硅
谷</a>
```

```
            </li>
          </ul>
          <div class="clear"></div>
        </div>
        <div class="b-foot">
          <section class="fl col-7">
            <section class="mr20">
              <section class="b-f-link">
                <a href="#" title="关于我们" target="_blank">关于我们</a>|
                <a href="#" title="联系我们" target="_blank">联系我们</a>|
                <a href="#" title="帮助中心" target="_blank">帮助中心</a>|
                <a href="#" title="资源下载" target="_blank">资源下载</a>|
                <span>服务热线：010-56253825(北京) 0755-85293825(深圳)</span>
                <span>Email：info@atguigu.com</span>
              </section>
              <section class="b-f-link mt10">
                <span>©2018课程版权均归谷粒学院所有 京ICP备17055252号</span>
              </section>
            </section>
          </section>
          <aside class="fl col-3 tac mt15">
            <section class="gf-tx">
              <span>
                <img src="~/assets/img/wx-icon.png" alt>
              </span>
            </section>
            <section class="gf-tx">
              <span>
                <img src="~/assets/img/wb-icon.png" alt>
              </span>
            </section>
          </aside>
          <div class="clear"></div>
        </div>
      </section>
    </footer>
    <!-- /公共底引入 -->
  </div>
</template>
<script>
import "~/assets/css/reset.css";
import "~/assets/css/theme.css";
import "~/assets/css/global.css";
```

```
import "~/assets/css/web.css";

export default {};
</script>
```

## 3、定义首页面

（不包含幻灯片）

修改pages/index.vue：

修改了原始文件中的资源路径为~/assets/

内容如下：

```
<template>

  <div>
    <!-- 幻灯片 开始 -->
    <!-- 幻灯片 结束 -->

     <div id="aCoursesList">
      <!-- 网校课程 开始 -->
      <div>
        <section class="container">
          <header class="comm-title">
            <h2 class="tac">
              <span class="c-333">热门课程</span>
            </h2>
          </header>
          <div>
            <article class="comm-course-list">
              <ul class="of" id="bna">
                <li>
                  <div class="cc-l-wrap">
                    <section class="course-img">
                      <img
                        src="~/assets/photo/course/1442295592705.jpg"
                        class="img-responsive"
                        alt="听力口语"
                      >
                      <div class="cc-mask">
```

```
                    <a href="#" title="开始学习" class="comm-btn c-btn-1">开始学习</a>
                  </div>
                </section>
                <h3 class="hLh30 txtOf mt10">
                  <a href="#" title="听力口语" class="course-title fsize18 c-333">听力口语</a>
                </h3>
                <section class="mt10 hLh20 of">
                  <span class="fr jgTag bg-green">
                    <i class="c-fff fsize12 f-fA">免费</i>
                  </span>
                  <span class="fl jgAttr c-ccc f-fA">
                    <i class="c-999 f-fA">9634人学习</i>
                    |
                    <i class="c-999 f-fA">9634评论</i>
                  </span>
                </section>
              </div>
            </li>
            <li>
              <div class="cc-l-wrap">
                <section class="course-img">
                  <img
                    src="~/assets/photo/course/1442295581911.jpg"
                    class="img-responsive"
                    alt="Java精品课程"
                  >
                  <div class="cc-mask">
                    <a href="#" title="开始学习" class="comm-btn c-btn-1">开始学习</a>
                  </div>
                </section>
                <h3 class="hLh30 txtOf mt10">
                  <a href="#" title="Java精品课程" class="course-title fsize18 c-333">Java精品课程</a>
                </h3>
                <section class="mt10 hLh20 of">
                  <span class="fr jgTag bg-green">
                    <i class="c-fff fsize12 f-fA">免费</i>
                  </span>
                  <span class="fl jgAttr c-ccc f-fA">
                    <i class="c-999 f-fA">501人学习</i>
```

```
                        |
                        <i class="c-999 f-fA">501评论</i>
                      </span>
                    </section>
                  </div>
                </li>
                <li>
                  <div class="cc-l-wrap">
                    <section class="course-img">
                      <img
                        src="~/assets/photo/course/1442295604295.jpg"
                        class="img-responsive"
                        alt="C4D零基础"
                      >
                      <div class="cc-mask">
                        <a href="#" title="开始学习" class="comm-btn c-btn-1">开始学习</a>
                      </div>
                    </section>
                    <h3 class="hLh30 txtOf mt10">
                      <a href="#" title="C4D零基础" class="course-title fsize18 c-333">C4D零基础</a>
                    </h3>
                    <section class="mt10 hLh20 of">
                      <span class="fr jgTag bg-green">
                        <i class="c-fff fsize12 f-fA">免费</i>
                      </span>
                      <span class="fl jgAttr c-ccc f-fA">
                        <i class="c-999 f-fA">300人学习</i>
                        |
                        <i class="c-999 f-fA">300评论</i>
                      </span>
                    </section>
                  </div>
                </li>
                <li>
                  <div class="cc-l-wrap">
                    <section class="course-img">
                      <img
                        src="~/assets/photo/course/1442302831779.jpg"
                        class="img-responsive"
                        alt="数学给宝宝带来的兴趣"
                      >
```

```
                    <div class="cc-mask">
                      <a href="#" title="开始学习" class="comm-btn c-btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="#" title="数学给宝宝带来的兴趣" class="course-title fsize18 c-333">数学给宝宝带来的兴趣</a>
                  </h3>
                  <section class="mt10 hLh20 of">
                    <span class="fr jgTag bg-green">
                      <i class="c-fff fsize12 f-fA">免费</i>
                    </span>
                    <span class="fl jgAttr c-ccc f-fA">
                      <i class="c-999 f-fA">256人学习</i>
                      |
                      <i class="c-999 f-fA">256评论</i>
                    </span>
                  </section>
                </div>
              </li>
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img
                      src="~/assets/photo/course/1442295455437.jpg"
                      class="img-responsive"
                      alt="零基础入门学习Python课程学习"
                    >
                    <div class="cc-mask">
                      <a href="#" title="开始学习" class="comm-btn c-btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a
                      href="#"
                      title="零基础入门学习Python课程学习"
                      class="course-title fsize18 c-333"
                    >零基础入门学习Python课程学习</a>
                  </h3>
                  <section class="mt10 hLh20 of">
                    <span class="fr jgTag bg-green">
```

```
                        <i class="c-fff fsize12 f-fA">免费</i>
                      </span>
                      <span class="fl jgAttr c-ccc f-fA">
                        <i class="c-999 f-fA">137人学习</i>
                        |
                        <i class="c-999 f-fA">137评论</i>
                      </span>
                    </section>
                  </div>
                </li>
                <li>
                  <div class="cc-l-wrap">
                    <section class="course-img">
                      <img
                        src="~/assets/photo/course/1442295570359.jpg"
                        class="img-responsive"
                        alt="MySql从入门到精通"
                      >
                      <div class="cc-mask">
                        <a href="#" title="开始学习" class="comm-btn c-btn-1">开始学习</a>
                      </div>
                    </section>
                    <h3 class="hLh30 txtOf mt10">
                      <a href="#" title="MySql从入门到精通" class="course-title fsize18 c-333">MySql从入门到精通</a>
                    </h3>
                    <section class="mt10 hLh20 of">
                      <span class="fr jgTag bg-green">
                        <i class="c-fff fsize12 f-fA">免费</i>
                      </span>
                      <span class="fl jgAttr c-ccc f-fA">
                        <i class="c-999 f-fA">125人学习</i>
                        |
                        <i class="c-999 f-fA">125评论</i>
                      </span>
                    </section>
                  </div>
                </li>
                <li>
                  <div class="cc-l-wrap">
                    <section class="course-img">
                      <img
```

```
                        src="~/assets/photo/course/1442302852837.jpg"
                        class="img-responsive"
                        alt="搜索引擎优化技术"
                      >
                      <div class="cc-mask">
                        <a href="#" title="开始学习" class="comm-btn c-btn-1">开始学习</a>
                      </div>
                    </section>
                    <h3 class="hLh30 txtOf mt10">
                      <a href="#" title="搜索引擎优化技术" class="course-title fsize18 c-333">搜索引擎优化技术</a>
                    </h3>
                    <section class="mt10 hLh20 of">
                      <span class="fr jgTag bg-green">
                        <i class="c-fff fsize12 f-fA">免费</i>
                      </span>
                      <span class="fl jgAttr c-ccc f-fA">
                        <i class="c-999 f-fA">123人学习</i>
                        |
                        <i class="c-999 f-fA">123评论</i>
                      </span>
                    </section>
                  </div>
                </li>
                <li>
                  <div class="cc-l-wrap">
                    <section class="course-img">
                      <img
                        src="~/assets/photo/course/1442295379715.jpg"
                        class="img-responsive"
                        alt="20世纪西方音乐"
                      >
                      <div class="cc-mask">
                        <a href="#" title="开始学习" class="comm-btn c-btn-1">开始学习</a>
                      </div>
                    </section>
                    <h3 class="hLh30 txtOf mt10">
                      <a href="#" title="20世纪西方音乐" class="course-title fsize18 c-333">20世纪西方音乐</a>
                    </h3>
                    <section class="mt10 hLh20 of">
```

```
                    <span class="fr jgTag bg-green">
                      <i class="c-fff fsize12 f-fA">免费</i>
                    </span>
                    <span class="fl jgAttr c-ccc f-fA">
                      <i class="c-999 f-fA">34人学习</i>
                      |
                      <i class="c-999 f-fA">34评论</i>
                    </span>
                  </section>
                </div>
              </li>
            </ul>
            <div class="clear"></div>
          </article>
          <section class="tac pt20">
            <a href="#" title="全部课程" class="comm-btn c-btn-2">全部课程</a>
          </section>
        </div>
      </section>
    </div>
    <!-- /网校课程 结束 -->
    <!-- 网校名师 开始 -->
    <div>
      <section class="container">
        <header class="comm-title">
          <h2 class="tac">
            <span class="c-333">名师大咖</span>
          </h2>
        </header>
        <div>
          <article class="i-teacher-list">
            <ul class="of">
              <li>
                <section class="i-teach-wrap">
                  <div class="i-teach-pic">
                    <a href="/teacher/1" title="姚晨">
                      <img alt="姚晨"
src="~/assets/photo/teacher/1442297885942.jpg">
                    </a>
                  </div>
                  <div class="mt10 hLh30 txtOf tac">
                    <a href="/teacher/1" title="姚晨" class="fsize18 c-666">姚
晨</a>
```

```
                    </div>
                    <div class="hLh30 txtOf tac">
                      <span class="fsize14 c-999">北京师范大学法学院副教授</span>
                    </div>
                    <div class="mt15 i-q-txt">
                      <p
                        class="c-999 f-fA"
                      >北京师范大学法学院副教授、清华大学法学博士。自2004年至今已有9年的司法考试培训经验。长期从事司法考试辅导，深知命题规律，了解解题技巧。内容把握准确，授课重点明确，层次分明，调理清晰，将法条法理与案例有机融合，强调综合，深入浅出。</p>
                    </div>
                  </section>
                </li>
                <li>
                  <section class="i-teach-wrap">
                    <div class="i-teach-pic">
                      <a href="/teacher/1" title="谢娜">
                        <img alt="谢娜"
src="~/assets/photo/teacher/1442297919077.jpg">
                      </a>
                    </div>
                    <div class="mt10 hLh30 txtOf tac">
                      <a href="/teacher/1" title="谢娜" class="fsize18 c-666">谢娜</a>
                    </div>
                    <div class="hLh30 txtOf tac">
                      <span class="fsize14 c-999">资深课程设计专家，专注10年AACTP美国培训协会认证导师</span>
                    </div>
                    <div class="mt15 i-q-txt">
                      <p
                        class="c-999 f-fA"
                      >十年课程研发和培训咨询经验，曾任国企人力资源经理、大型外企培训经理，负责企业大学和培训体系搭建；曾任专业培训机构高级顾问、研发部总监，为包括广东移动、东莞移动、深圳移动、南方电网、工商银行、农业银行、民生银行、邮储银行、TCL集团、清华大学继续教育学院、中天路桥、广西扬翔股份等超过200家企业提供过培训与咨询服务，并担任近50个大型项目的总负责人。</p>
                    </div>
                  </section>
                </li>
                <li>
                  <section class="i-teach-wrap">
                    <div class="i-teach-pic">
                      <a href="/teacher/1" title="刘德华">
```

```
                        <img alt="刘德华"
src="~/assets/photo/teacher/1442297927029.jpg">
                      </a>
                    </div>
                    <div class="mt10 hLh30 txtOf tac">
                      <a href="/teacher/1" title="刘德华" class="fsize18 c-666">刘
德华</a>
                    </div>
                    <div class="hLh30 txtOf tac">
                      <span class="fsize14 c-999">上海师范大学法学院副教授</span>
                    </div>
                    <div class="mt15 i-q-txt">
                      <p
                        class="c-999 f-fA"
                      >上海师范大学法学院副教授、清华大学法学博士。自2004年至今已有9年
的司法考试培训经验。长期从事司法考试辅导，深知命题规律，了解解题技巧。内容把握准确，授课重
点明确，层次分明，调理清晰，将法条法理与案例有机融合，强调综合，深入浅出。</p>
                    </div>
                  </section>
                </li>
                <li>
                  <section class="i-teach-wrap">
                    <div class="i-teach-pic">
                      <a href="/teacher/1" title="周润发">
                        <img alt="周润发"
src="~/assets/photo/teacher/1442297935589.jpg">
                      </a>
                    </div>
                    <div class="mt10 hLh30 txtOf tac">
                      <a href="/teacher/1" title="周润发" class="fsize18 c-666">周
润发</a>
                    </div>
                    <div class="hLh30 txtOf tac">
                      <span class="fsize14 c-999">考研政治辅导实战派专家，全国考研政
治命题研究组核心成员。</span>
                    </div>
                    <div class="mt15 i-q-txt">
                      <p
                        class="c-999 f-fA"
                      >法学博士，北京师范大学马克思主义学院副教授，专攻毛泽东思想概论、
邓小平理论，长期从事考研辅导。出版著作两部，发表学术论文30余篇，主持国家社会科学基金项目和
教育部重大课题子课题各一项，参与中央实施马克思主义理论研究和建设工程项目。</p>
                    </div>
                  </section>
```

```
                </li>
              </ul>
              <div class="clear"></div>
            </article>
            <section class="tac pt20">
              <a href="#" title="全部讲师" class="comm-btn c-btn-2">全部讲师</a>
            </section>
          </div>
        </section>
      </div>
      <!-- /网校名师 结束 -->
    </div>
  </div>
</template>

<script>
export default {

}
</script>
```

# 4、幻灯片插件

```
<!-- 幻灯片 开始 -->
<div v-swiper:mySwiper="swiperOption">
    <div class="swiper-wrapper">
        <div class="swiper-slide" style="background: #040B1B;">
            <a target="_blank" href="/">
                <img src="~/assets/photo/banner/1525939573202.jpg" alt="首页banner">
            </a>
        </div>
        <div class="swiper-slide" style="background: #040B1B;">
            <a target="_blank" href="/">
                <img src="~/assets/photo/banner/1525939573202.jpg" alt="首页banner">
            </a>
        </div>
    </div>
    <div class="swiper-pagination swiper-pagination-white"></div>
```

```
    <div class="swiper-button-prev swiper-button-white" slot="button-prev"></div>
    <div class="swiper-button-next swiper-button-white" slot="button-next"></div>
</div>
<!-- 幻灯片 结束 -->
```

script

```
<script>
export default {
  data () {
    return {
      swiperOption: {
        //配置分页
        pagination: {
          el: '.swiper-pagination'//分页的dom节点
        },
        //配置导航
        navigation: {
          nextEl: '.swiper-button-next',//下一页dom节点
          prevEl: '.swiper-button-prev'//前一页dom节点
        }
      }
    }
  }
}
</script>
```

# 二、路由

## 1、固定路由

（1）使用**router-link**构建路由，地址是**/course**

```
<router-link to="/course" tag="li" active-class="current">
  <a>课程</a>
</router-link>
```

**(2) 在page目录创建文件夹course ， 在course目录创建index.vue**

```vue
<template>
  <div>
    课程列表
  </div>
</template>
```

**点击导航，测试路由**

## 2、动态路由

### （1）创建方式

如果我们需要根据id查询一条记录，就需要使用动态路由。NUXT的动态路由是以下划线开头的vue文件，参数名为下划线后边的文件名

在pages下的course目录下创建_id.vue

```vue
<template>
  <div>
    讲师详情
  </div>
</template>
```

# 三、封装axios

我们可以参考guli-admin将axios操作封装起来

下载axios，使用命令 **npm install axios**

创建utils文件夹，utils下创建request.js

```
import axios from 'axios'
// 创建axios实例
const service = axios.create({
  baseURL: 'http://localhost:8201', // api的base_url
  timeout: 20000 // 请求超时时间
```

```
})
export default service
```

# 一、列表页面

创建 pages/teacher/index.vue

```
<template>
  <div id="aCoursesList" class="bg-fa of">
    <!-- 讲师列表 开始 -->
    <section class="container">
      <header class="comm-title all-teacher-title">
        <h2 class="fl tac">
          <span class="c-333">全部讲师</span>
        </h2>
        <section class="c-tab-title">
          <a id="subjectAll" title="全部" href="#">全部</a>
          <!-- <c:forEach var="subject" items="${subjectList }">
                          <a id="${subject.subjectId}"
title="${subject.subjectName }" href="javascript:void(0)"
onclick="submitForm(${subject.subjectId})">${subject.subjectName }</a>
          </c:forEach>-->
        </section>
      </header>
      <section class="c-sort-box unBr">
        <div>
          <!-- /无数据提示 开始-->
          <section class="no-data-wrap">
            <em class="icon30 no-data-ico"> </em>
            <span class="c-666 fsize14 ml10 vam">没有相关数据，小编正在努力整理
中...</span>
          </section>
          <!-- /无数据提示 结束-->
          <article class="i-teacher-list">
            <ul class="of">
              <li>
                <section class="i-teach-wrap">
                  <div class="i-teach-pic">
                    <a href="/teacher/1" title="姚晨" target="_blank">
                      <img src="~/assets/photo/teacher/1442297885942.jpg" alt>
                    </a>
                  </div>
                  <div class="mt10 hLh30 txtOf tac">
                    <a href="/teacher/1" title="姚晨" target="_blank"
class="fsize18 c-666">姚晨</a>
```

```
                </div>
                <div class="hLh30 txtOf tac">
                  <span class="fsize14 c-999">北京师范大学法学院副教授、清华大学法学博士。自2004年至今已有9年的司法考试培训经验。长期从事司法考试辅导，深知命题规律，了解解题技巧。内容把握准确，授课重点明确，层次分明，调理清晰，将法条法理与案例有机融合，强调综合，深入浅出。</span>
                </div>
                <div class="mt15 i-q-txt">
                  <p class="c-999 f-fA">北京师范大学法学院副教授</p>
                </div>
              </section>
            </li>
            <li>
              <section class="i-teach-wrap">
                <div class="i-teach-pic">
                  <a href="/teacher/1" title="谢娜" target="_blank">
                    <img src="~/assets/photo/teacher/1442297919077.jpg" alt>
                  </a>
                </div>
                <div class="mt10 hLh30 txtOf tac">
                  <a href="/teacher/1" title="谢娜" target="_blank" class="fsize18 c-666">谢娜</a>
                </div>
                <div class="hLh30 txtOf tac">
                  <span class="fsize14 c-999">十年课程研发和培训咨询经验，曾任国企人力资源经理、大型外企培训经理，负责企业大学和培训体系搭建；曾任专业培训机构高级顾问、研发部总监，为包括广东移动、东莞移动、深圳移动、南方电网、工商银行、农业银行、民生银行、邮储银行、TCL集团、清华大学继续教育学院、中天路桥、广西扬翔股份等超过200家企业提供过培训与咨询服务，并担任近50个大型项目的总负责人。</span>
                </div>
                <div class="mt15 i-q-txt">
                  <p class="c-999 f-fA">资深课程设计专家，专注10年AACTP美国培训协会认证导师</p>
                </div>
              </section>
            </li>
            <li>
              <section class="i-teach-wrap">
                <div class="i-teach-pic">
                  <a href="/teacher/1" title="刘德华" target="_blank">
                    <img src="~/assets/photo/teacher/1442297927029.jpg" alt>
                  </a>
                </div>
                <div class="mt10 hLh30 txtOf tac">
                  <a href="/teacher/1" title="刘德华" target="_blank"
```

```
class="fsize18 c-666">刘德华</a>
                  </div>
                  <div class="hLh30 txtOf tac">
                    <span class="fsize14 c-999">上海师范大学法学院副教授、清华大学法学博士。自2004年至今已有9年的司法考试培训经验。长期从事司法考试辅导，深知命题规律，了解解题技巧。内容把握准确，授课重点明确，层次分明，调理清晰，将法条法理与案例有机融合，强调综合，深入浅出。</span>
                  </div>
                  <div class="mt15 i-q-txt">
                    <p class="c-999 f-fA">上海师范大学法学院副教授</p>
                  </div>
                </section>
              </li>
              <li>
                <section class="i-teach-wrap">
                  <div class="i-teach-pic">
                    <a href="/teacher/1" title="周润发" target="_blank">
                      <img src="~/assets/photo/teacher/1442297935589.jpg" alt>
                    </a>
                  </div>
                  <div class="mt10 hLh30 txtOf tac">
                    <a href="/teacher/1" title="周润发" target="_blank"
class="fsize18 c-666">周润发</a>
                  </div>
                  <div class="hLh30 txtOf tac">
                    <span class="fsize14 c-999">法学博士，北京师范大学马克思主义学院副教授，专攻毛泽东思想概论、邓小平理论，长期从事考研辅导。出版著作两部，发表学术论文30余篇，主持国家社会科学基金项目和教育部重大课题子课题各一项，参与中央实施马克思主义理论研究和建设工程项目。</span>
                  </div>
                  <div class="mt15 i-q-txt">
                    <p class="c-999 f-fA">考研政治辅导实战派专家，全国考研政治命题研究组核心成员。</p>
                  </div>
                </section>
              </li>
              <li>
                <section class="i-teach-wrap">
                  <div class="i-teach-pic">
                    <a href="/teacher/1" title="钟汉良" target="_blank">
                      <img src="~/assets/photo/teacher/1442298121626.jpg" alt>
                    </a>
                  </div>
                  <div class="mt10 hLh30 txtOf tac">
                    <a href="/teacher/1" title="钟汉良" target="_blank"
```

```
class="fsize18 c-666">钟汉良</a>
                  </div>
                  <div class="hLh30 txtOf tac">
                    <span class="fsize14 c-999">具备深厚的数学思维功底、丰富的小学教育经验，授课风格生动活泼，擅长用形象生动的比喻帮助理解、简单易懂的语言讲解难题，深受学生喜欢</span>
                  </div>
                  <div class="mt15 i-q-txt">
                    <p class="c-999 f-fA">毕业于师范大学数学系，热爱教育事业，执教数学思维6年有余</p>
                  </div>
                </section>
              </li>
              <li>
                <section class="i-teach-wrap">
                  <div class="i-teach-pic">
                    <a href="/teacher/1" title="唐嫣" target="_blank">
                      <img src="~/assets/photo/teacher/1442297957332.jpg" alt>
                    </a>
                  </div>
                  <div class="mt10 hLh30 txtOf tac">
                    <a href="/teacher/1" title="唐嫣" target="_blank"
class="fsize18 c-666">唐嫣</a>
                  </div>
                  <div class="hLh30 txtOf tac">
                    <span class="fsize14 c-999">中国科学院数学与系统科学研究院应用数学专业博士，研究方向为数字图像处理，中国工业与应用数学学会会员。参与全国教育科学“十五”规划重点课题“信息化进程中的教育技术发展研究”的子课题“基与课程改革的资源开发与应用”，以及全国“十五”科研规划全国重点项目“掌上型信息技术产品在教学中的运用和开发研究”的子课题“用技术学数学”。</span>
                  </div>
                  <div class="mt15 i-q-txt">
                    <p class="c-999 f-fA">中国人民大学附属中学数学一级教师</p>
                  </div>
                </section>
              </li>
              <li>
                <section class="i-teach-wrap">
                  <div class="i-teach-pic">
                    <a href="/teacher/1" title="周杰伦" target="_blank">
                      <img src="~/assets/photo/teacher/1442297969808.jpg" alt>
                    </a>
                  </div>
                  <div class="mt10 hLh30 txtOf tac">
                    <a href="/teacher/1" title="周杰伦" target="_blank"
```

```
class="fsize18 c-666">周杰伦</a>
                </div>
                <div class="hLh30 txtOf tac">
                  <span class="fsize14 c-999">中教一级职称。讲课极具亲和
力。</span>
                </div>
                <div class="mt15 i-q-txt">
                  <p class="c-999 f-fA">毕业于北京大学数学系</p>
                </div>
              </section>
            </li>
            <li>
              <section class="i-teach-wrap">
                <div class="i-teach-pic">
                  <a href="/teacher/1" title="陈伟霆" target="_blank">
                    <img src="~/assets/photo/teacher/1442297977255.jpg" alt>
                  </a>
                </div>
                <div class="mt10 hLh30 txtOf tac">
                  <a href="/teacher/1" title="陈伟霆" target="_blank"
class="fsize18 c-666">陈伟霆</a>
                </div>
                <div class="hLh30 txtOf tac">
                  <span
                    class="fsize14 c-999"
                  >政治学博士、管理学博士后，北京师范大学马克思主义学院副教授。多
年来总结出了一套行之有效的应试技巧与答题方法，针对性和实用性极强，能帮助考生在轻松中应
考，在激励的竞争中取得高分，脱颖而出。</span>
                </div>
                <div class="mt15 i-q-txt">
                  <p class="c-999 f-fA">长期从事考研政治课讲授和考研命题趋势与
应试对策研究。考研辅导新锐派的代表。</p>
                </div>
              </section>
            </li>
          </ul>
          <div class="clear"></div>
        </article>
      </div>
      <!-- 公共分页 开始 -->
      <div>
        <div class="paging">
          <!-- undisable这个class是否存在，取决于数据属性hasPrevious -->
          <a href="#" title="首页">首</a>
          <a href="#" title="前一页">&lt;</a>
```

```
            <a href="#" title="第1页" class="current undisable">1</a>
            <a href="#" title="第2页">2</a>
            <a href="#" title="后一页">&gt;</a>
            <a href="#" title="末页">末</a>
            <div class="clear"></div>
          </div>
        </div>
        <!-- 公共分页 结束 -->
      </section>
    </section>
    <!-- /讲师列表 结束 -->
  </div>
</template>
<script>
export default {};
</script>
```

# 二、详情页面

创建 pages/teacher/_id.vue

修改资源路径为~/assets/

```
<template>
  <div id="aCoursesList" class="bg-fa of">
    <!-- 讲师介绍 开始 -->
    <section class="container">
      <header class="comm-title">
        <h2 class="fl tac">
          <span class="c-333">讲师介绍</span>
        </h2>
      </header>
      <div class="t-infor-wrap">
        <!-- 讲师基本信息 -->
        <section class="fl t-infor-box c-desc-content">
          <div class="mt20 ml20">
            <section class="t-infor-pic">
              <img src="~/assets/photo/teacher/1442297885942.jpg">
            </section>
            <h3 class="hLh30">
              <span class="fsize24 c-333">姚晨 高级讲师</span>
            </h3>
            <section class="mt10">
```

```
              <span class="t-tag-bg">北京师范大学法学院副教授</span>
            </section>
            <section class="t-infor-txt">
              <p
                class="mt20"
              >北京师范大学法学院副教授、清华大学法学博士。自2004年至今已有9年的司法考试培训经验。长期从事司法考试辅导，深知命题规律，了解解题技巧。内容把握准确，授课重点明确，层次分明，调理清晰，将法条法理与案例有机融合，强调综合，深入浅出。</p>
            </section>
            <div class="clear"></div>
          </div>
        </section>
        <div class="clear"></div>
      </div>
      <section class="mt30">
        <div>
          <header class="comm-title all-teacher-title c-course-content">
            <h2 class="fl tac">
              <span class="c-333">主讲课程</span>
            </h2>
            <section class="c-tab-title">
              <a href="javascript: void(0)"> </a>
            </section>
          </header>
          <!-- /无数据提示 开始-->
          <section class="no-data-wrap">
            <em class="icon30 no-data-ico"> </em>
            <span class="c-666 fsize14 ml10 vam">没有相关数据，小编正在努力整理中...</span>
          </section>
          <!-- /无数据提示 结束-->
          <article class="comm-course-list">
            <ul class="of">
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442295455437.jpg" class="img-responsive" >
                    <div class="cc-mask">
                      <a href="#" title="开始学习" target="_blank" class="comm-btn c-btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="#" title="零基础入门学习Python课程学习"
```

```
target="_blank" class="course-title fsize18 c-333">零基础入门学习Python课程学习</a>
                  </h3>
                </div>
              </li>
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442295472860.jpg" class="img-responsive" >
                    <div class="cc-mask">
                      <a href="#" title="开始学习" target="_blank" class="comm-btn c-btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="#" title="影想力摄影小课堂" target="_blank" class="course-title fsize18 c-333">影想力摄影小课堂</a>
                  </h3>
                </div>
              </li>
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442302831779.jpg" class="img-responsive" >
                    <div class="cc-mask">
                      <a href="#" title="开始学习" target="_blank" class="comm-btn c-btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="#" title="数学给宝宝带来的兴趣" target="_blank" class="course-title fsize18 c-333">数学给宝宝带来的兴趣</a>
                  </h3>
                </div>
              </li>
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442295506745.jpg" class="img-responsive" >
                    <div class="cc-mask">
                      <a href="#" title="开始学习" target="_blank" class="comm-btn c-btn-1">开始学习</a>
```

```
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="#" title="国家教师资格考试专用" target="_blank"
class="course-title fsize18 c-333">国家教师资格考试专用</a>
                  </h3>
                </div>
              </li>
            </ul>
            <div class="clear"></div>
          </article>
        </div>
      </section>
    </section>
    <!-- /讲师介绍 结束 -->
  </div>
</template>
<script>
export default {};
</script>
```

# 一、列表页面

创建 pages/course/index.vue

```
<template>
  <div id="aCoursesList" class="bg-fa of">
    <!-- /课程列表 开始 -->
    <section class="container">
      <header class="comm-title">
        <h2 class="fl tac">
          <span class="c-333">全部课程</span>
        </h2>
      </header>
      <section class="c-sort-box">
        <section class="c-s-dl">
          <dl>
            <dt>
              <span class="c-999 fsize14">课程类别</span>
            </dt>
            <dd class="c-s-dl-li">
              <ul class="clearfix">
                <li>
                  <a title="全部" href="#">全部</a>
                </li>
                <li>
                  <a title="数据库" href="#">数据库</a>
                </li>
                <li class="current">
                  <a title="外语考试" href="#">外语考试</a>
                </li>
                <li>
                  <a title="教师资格证" href="#">教师资格证</a>
                </li>
                <li>
                  <a title="公务员" href="#">公务员</a>
                </li>
                <li>
                  <a title="移动开发" href="#">移动开发</a>
                </li>
                <li>
                  <a title="操作系统" href="#">操作系统</a>
                </li>
```

```
            </ul>
          </dd>
        </dl>
        <dl>
          <dt>
            <span class="c-999 fsize14"></span>
          </dt>
          <dd class="c-s-dl-li">
            <ul class="clearfix">
              <li>
                <a title="职称英语" href="#">职称英语</a>
              </li>
              <li>
                <a title="英语四级" href="#">英语四级</a>
              </li>
              <li>
                <a title="英语六级" href="#">英语六级</a>
              </li>
            </ul>
          </dd>
        </dl>
        <div class="clear"></div>
      </section>
      <div class="js-wrap">
        <section class="fr">
          <span class="c-ccc">
            <i class="c-master f-fM">1</i>/
            <i class="c-666 f-fM">1</i>
          </span>
        </section>
        <section class="fl">
          <ol class="js-tap clearfix">
            <li>
              <a title="关注度" href="#">关注度</a>
            </li>
            <li>
              <a title="最新" href="#">最新</a>
            </li>
            <li class="current bg-orange">
              <a title="价格" href="#">价格 
                <span>↓</span>
              </a>
            </li>
          </ol>
        </section>
```

```
        </div>
        <div class="mt40">
          <!-- /无数据提示 开始-->
          <section class="no-data-wrap">
            <em class="icon30 no-data-ico"> </em>
            <span class="c-666 fsize14 ml10 vam">没有相关数据，小编正在努力整理中...</span>
          </section>
          <!-- /无数据提示 结束-->
          <article class="comm-course-list">
            <ul class="of" id="bna">
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442295592705.jpg" class="img-responsive" alt="听力口语">
                    <div class="cc-mask">
                      <a href="/course/1" title="开始学习" class="comm-btn c-btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="/course/1" title="听力口语" class="course-title fsize18 c-333">听力口语</a>
                  </h3>
                  <section class="mt10 hLh20 of">
                    <span class="fr jgTag bg-green">
                      <i class="c-fff fsize12 f-fA">免费</i>
                    </span>
                    <span class="fl jgAttr c-ccc f-fA">
                      <i class="c-999 f-fA">9634人学习</i>
                      |
                      <i class="c-999 f-fA">9634评论</i>
                    </span>
                  </section>
                </div>
              </li>
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442295581911.jpg" class="img-responsive" alt="Java精品课程">
                    <div class="cc-mask">
                      <a href="/course/1" title="开始学习" class="comm-btn c-btn-1">开始学习</a>
```

```
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="/course/1" title="Java精品课程" class="course-
title fsize18 c-333">Java精品课程</a>
                  </h3>
                  <section class="mt10 hLh20 of">
                    <span class="fr jgTag bg-green">
                      <i class="c-fff fsize12 f-fA">免费</i>
                    </span>
                    <span class="fl jgAttr c-ccc f-fA">
                      <i class="c-999 f-fA">501人学习</i>
                      |
                      <i class="c-999 f-fA">501评论</i>
                    </span>
                  </section>
                </div>
              </li>
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442295604295.jpg"
class="img-responsive" alt="C4D零基础">
                    <div class="cc-mask">
                      <a href="/course/1" title="开始学习" class="comm-btn c-
btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="/course/1" title="C4D零基础" class="course-title
fsize18 c-333">C4D零基础</a>
                  </h3>
                  <section class="mt10 hLh20 of">
                    <span class="fr jgTag bg-green">
                      <i class="c-fff fsize12 f-fA">免费</i>
                    </span>
                    <span class="fl jgAttr c-ccc f-fA">
                      <i class="c-999 f-fA">300人学习</i>
                      |
                      <i class="c-999 f-fA">300评论</i>
                    </span>
                  </section>
                </div>
              </li>
              <li>
```

```
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img
                      src="~/assets/photo/course/1442302831779.jpg"
                      class="img-responsive"
                      alt="数学给宝宝带来的兴趣"
                    >
                    <div class="cc-mask">
                      <a href="/course/1" title="开始学习" class="comm-btn c-
btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="/course/1" title="数学给宝宝带来的兴趣"
class="course-title fsize18 c-333">数学给宝宝带来的兴趣</a>
                  </h3>
                  <section class="mt10 hLh20 of">
                    <span class="fr jgTag bg-green">
                      <i class="c-fff fsize12 f-fA">免费</i>
                    </span>
                    <span class="fl jgAttr c-ccc f-fA">
                      <i class="c-999 f-fA">256人学习</i>
                      |
                      <i class="c-999 f-fA">256评论</i>
                    </span>
                  </section>
                </div>
              </li>
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img
                      src="~/assets/photo/course/1442295455437.jpg"
                      class="img-responsive"
                      alt="零基础入门学习Python课程学习"
                    >
                    <div class="cc-mask">
                      <a href="/course/1" title="开始学习" class="comm-btn c-
btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a
                      href="/course/1"
                      title="零基础入门学习Python课程学习"
```

```
                    class="course-title fsize18 c-333"
                  >零基础入门学习Python课程学习</a>
                </h3>
                <section class="mt10 hLh20 of">
                  <span class="fr jgTag bg-green">
                    <i class="c-fff fsize12 f-fA">免费</i>
                  </span>
                  <span class="fl jgAttr c-ccc f-fA">
                    <i class="c-999 f-fA">137人学习</i>
                    |
                    <i class="c-999 f-fA">137评论</i>
                  </span>
                </section>
              </div>
            </li>
            <li>
              <div class="cc-l-wrap">
                <section class="course-img">
                  <img
                    src="~/assets/photo/course/1442295570359.jpg"
                    class="img-responsive"
                    alt="MySql从入门到精通"
                  >
                  <div class="cc-mask">
                    <a href="/course/1" title="开始学习" class="comm-btn c-
btn-1">开始学习</a>
                  </div>
                </section>
                <h3 class="hLh30 txtOf mt10">
                  <a href="/course/1" title="MySql从入门到精通"
class="course-title fsize18 c-333">MySql从入门到精通</a>
                </h3>
                <section class="mt10 hLh20 of">
                  <span class="fr jgTag bg-green">
                    <i class="c-fff fsize12 f-fA">免费</i>
                  </span>
                  <span class="fl jgAttr c-ccc f-fA">
                    <i class="c-999 f-fA">125人学习</i>
                    |
                    <i class="c-999 f-fA">125评论</i>
                  </span>
                </section>
              </div>
            </li>
            <li>
```

```
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442302852837.jpg"
class="img-responsive" alt="搜索引擎优化技术">
                    <div class="cc-mask">
                      <a href="/course/1" title="开始学习" class="comm-btn c-
btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="/course/1" title="搜索引擎优化技术" class="course-
title fsize18 c-333">搜索引擎优化技术</a>
                  </h3>
                  <section class="mt10 hLh20 of">
                    <span class="fr jgTag bg-green">
                      <i class="c-fff fsize12 f-fA">免费</i>
                    </span>
                    <span class="fl jgAttr c-ccc f-fA">
                      <i class="c-999 f-fA">123人学习</i>
                      |
                      <i class="c-999 f-fA">123评论</i>
                    </span>
                  </section>
                </div>
              </li>
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442295379715.jpg"
class="img-responsive" alt="20世纪西方音乐">
                    <div class="cc-mask">
                      <a href="/course/1" title="开始学习" class="comm-btn c-
btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="/course/1" title="20世纪西方音乐" class="course-
title fsize18 c-333">20世纪西方音乐</a>
                  </h3>
                  <section class="mt10 hLh20 of">
                    <span class="fr jgTag bg-green">
                      <i class="c-fff fsize12 f-fA">免费</i>
                    </span>
                    <span class="fl jgAttr c-ccc f-fA">
                      <i class="c-999 f-fA">34人学习</i>
```

```
                    |
                    <i class="c-999 f-fA">34评论</i>
                  </span>
                </section>
              </div>
            </li>
          </ul>
          <div class="clear"></div>
        </article>
      </div>
      <!-- 公共分页 开始 -->
      <div>
        <div class="paging">
          <a class="undisable" title>首</a>
          <a id="backpage" class="undisable" href="#" title>&lt;</a>
          <a href="#" title class="current undisable">1</a>
          <a href="#" title>2</a>
          <a id="nextpage" href="#" title>&gt;</a>
          <a href="#" title>末</a>
          <div class="clear"></div>
        </div>
      </div>
      <!-- 公共分页 结束 -->
    </section>
  </section>
  <!-- /课程列表 结束 -->
 </div>
</template>
<script>
export default {};
</script>
```

# 二、详情页面

创建 pages/course/_id.vue

```
<template>
  <div id="aCoursesList" class="bg-fa of">
    <!-- /课程详情 开始 -->
    <section class="container">
      <section class="path-wrap txtOf hLh30">
```

```
        <a href="#" title class="c-999 fsize14">首页</a>
        \
        <a href="#" title class="c-999 fsize14">课程列表</a>
        \
        <span class="c-333 fsize14">Java精品课程</span>
      </section>
      <div>
        <article class="c-v-pic-wrap" style="height: 357px;">
          <section class="p-h-video-box" id="videoPlay">
            <img src="~/assets/photo/course/1442295581911.jpg" alt="Java精品课程" class="dis c-v-pic">
          </section>
        </article>
        <aside class="c-attr-wrap">
          <section class="ml20 mr15">
            <h2 class="hLh30 txtOf mt15">
              <span class="c-fff fsize24">Java精品课程</span>
            </h2>
            <section class="c-attr-jg">
              <span class="c-fff">价格: </span>
              <b class="c-yellow" style="font-size:24px;">￥0.00</b>
            </section>
            <section class="c-attr-mt c-attr-undis">
              <span class="c-fff fsize14">主讲:  唐嫣&amp;nbsp;&amp;nbsp;&amp;nbsp;</span>
            </section>
            <section class="c-attr-mt of">
              <span class="ml10 vam">
                <em class="icon18 scIcon"></em>
                <a class="c-fff vam" title="收藏" href="#" >收藏</a>
              </span>
            </section>
            <section class="c-attr-mt">
              <a href="#" title="立即观看" class="comm-btn c-btn-3">立即观看</a>
            </section>
          </section>
        </aside>
        <aside class="thr-attr-box">
          <ol class="thr-attr-ol clearfix">
            <li>
              <p>&amp;nbsp;</p>
              <aside>
                <span class="c-fff f-fM">购买数</span>
                <br>
                <h6 class="c-fff f-fM mt10">150</h6>
```

```
              </aside>
            </li>
            <li>
              <p> </p>
              <aside>
                <span class="c-fff f-fM">课时数</span>
                <br>
                <h6 class="c-fff f-fM mt10">20</h6>
              </aside>
            </li>
            <li>
              <p> </p>
              <aside>
                <span class="c-fff f-fM">浏览数</span>
                <br>
                <h6 class="c-fff f-fM mt10">501</h6>
              </aside>
            </li>
          </ol>
        </aside>
        <div class="clear"></div>
      </div>
      <!-- /课程封面介绍 -->
      <div class="mt20 c-infor-box">
        <article class="fl col-7">
          <section class="mr30">
            <div class="i-box">
              <div>
                <section id="c-i-tabTitle" class="c-infor-tabTitle c-tab-
title">
                  <a name="c-i" class="current" title="课程详情">课程详情</a>
                </section>
              </div>
              <article class="ml10 mr10 pt20">
                <div>
                  <h6 class="c-i-content c-infor-title">
                    <span>课程介绍</span>
                  </h6>
                  <div class="course-txt-body-wrap">
                    <section class="course-txt-body">
                      <p>
                        Java的发展历史，可追溯到1990年。当
时Sun Microsystem公司为了发展消费性电子产品而进行了一个名为Green的项目计划。该
计划
                        负责人是James Gosling。起初他以C++来写一种内嵌式软
```

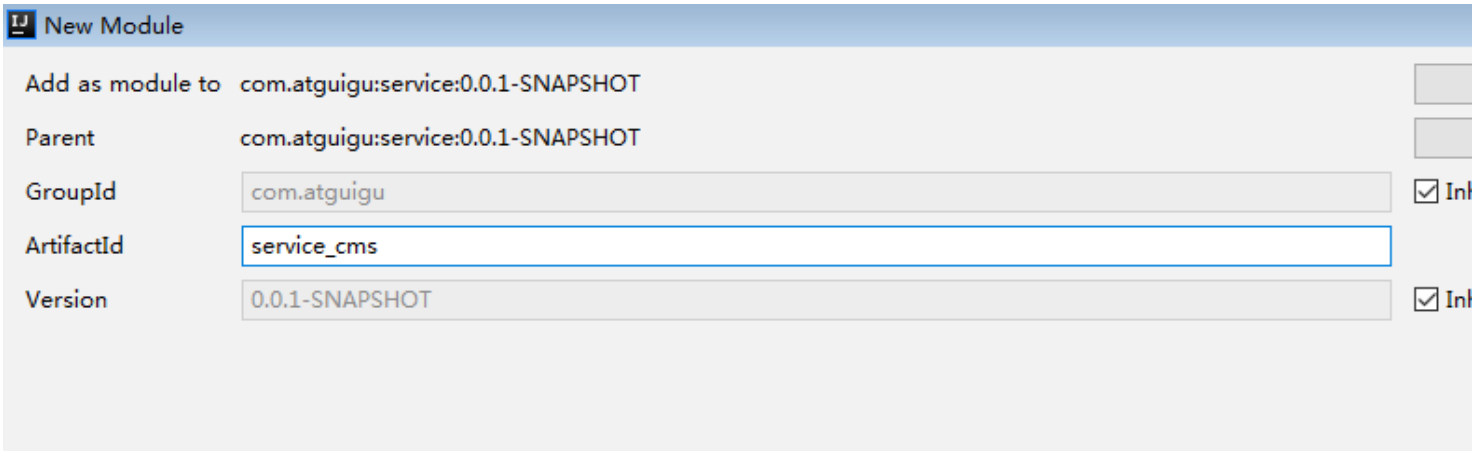
```
件，可以放在烤面包机或PAD等小型电子消费设备里，使得机器更聪明，具有人工智
                        能。但他发现C++并不适合完成这类任务！因为C++常会有使系统失效的程序错误，尤其是内存管理，需要程序设计师记录并管理内存资源。这给设计师们造成
                        极大的负担，并可能产生许多bugs。 
                        <br>为了解决所遇到的问题，Gosling决定要发展一种新的语言，来解决C++的潜在性危险问题，这个语言名叫Oak。Oak是一种可移植性语言，也就是一种平台独立语言，能够在各种芯片上运行。
                        <br>1994年，Oak技术日趋成熟，这时网络正开始蓬勃发展。Oak研发小组发现Oak很适合作为一种网络程序语言。因此发展了一个能与Oak配合的浏
                        览器--WebRunner，后更名为HotJava，它证明了Oak是一种能在网络上发展的程序语言。由于Oak商标已被注册，工程师们便想到以自己常
                        享用的咖啡(Java)来重新命名，并于Sun World 95中被发表出来。
                    </p>
                  </section>
                </div>
              </div>
              <!-- /课程介绍 -->
              <div class="mt50">
                <h6 class="c-g-content c-infor-title">
                  <span>课程大纲</span>
                </h6>
                <section class="mt20">
                  <div class="lh-menu-wrap">
                    <menu id="lh-menu" class="lh-menu mt10 mr10">
                      <ul>
                        <!-- 文件目录 -->
                        <li class="lh-menu-stair">
                          <a href="javascript: void(0)" title="第一章" class="current-1">
                            <em class="lh-menu-i-1 icon18 mr10"></em>第一章
                          </a>
                          <ol class="lh-menu-ol" style="display: block;">
                            <li class="lh-menu-second ml30">
                              <a href="#" title>
                                <span class="fr">
                                  <i class="free-icon vam mr10">免费试听</i>
                                </span>
                                <em class="lh-menu-i-2 icon16 mr5"> </em>第一节
                              </a>
                            </li>
                            <li class="lh-menu-second ml30">
                              <a href="#" title class="current-2">
                                <em class="lh-menu-i-2 icon16
```

```
mr5"> </em>第二节
                                </a>
                              </li>
                            </ol>
                          </li>
                        </ul>
                      </menu>
                    </div>
                  </section>
                </div>
                <!-- /课程大纲 -->
              </article>
            </div>
          </section>
        </article>
        <aside class="fl col-3">
          <div class="i-box">
            <div>
              <section class="c-infor-tabTitle c-tab-title">
                <a title href="javascript:void(0)">主讲讲师</a>
              </section>
              <section class="stud-act-list">
                <ul style="height: auto;">
                  <li>
                    <div class="u-face">
                      <a href="#">
                        <img src="~/assets/photo/teacher/1442297969808.jpg"
width="50" height="50" alt>
                      </a>
                    </div>
                    <section class="hLh30 txtOf">
                      <a class="c-333 fsize16 fl" href="#">周杰伦</a>
                    </section>
                    <section class="hLh20 txtOf">
                      <span class="c-999">毕业于北京大学数学系</span>
                    </section>
                  </li>
                </ul>
              </section>
            </div>
          </div>
        </aside>
        <div class="clear"></div>
      </div>
    </section>
```

```
    <!-- /课程详情 结束 -->
  </div>
</template>
<script>
export default {};
</script>

```

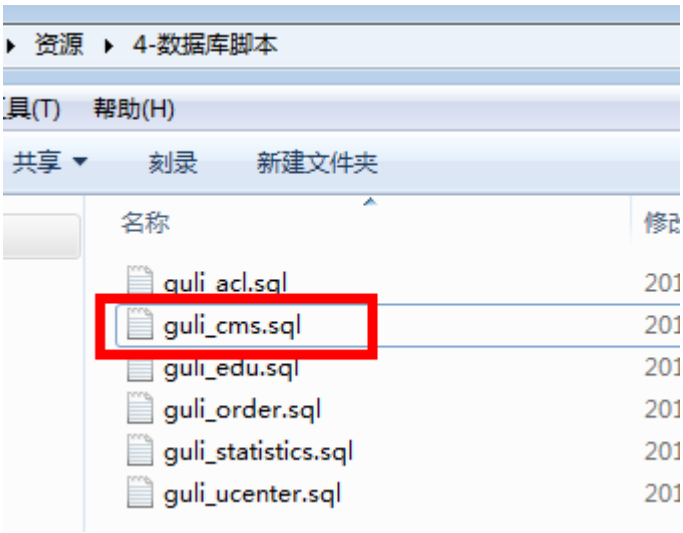
# 一、新建banner微服务

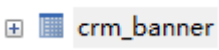
## 1、在service模块下创建子模块service-cms



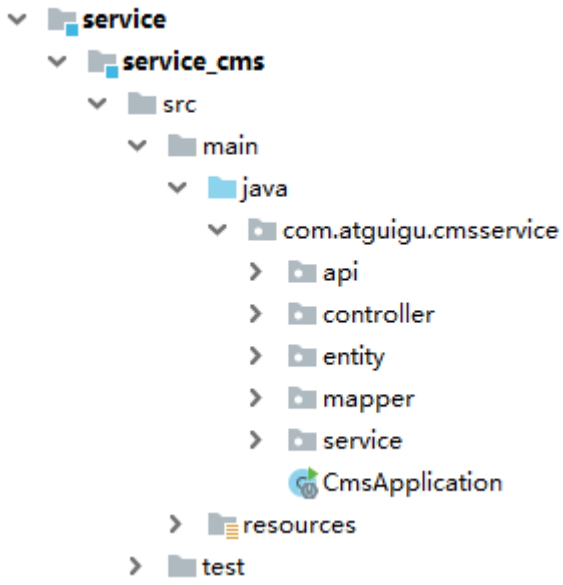

## 2、使用代码生成器生成banner代码

（1）创建**crm_banner**表




（2）生成代码




## 3、配置application.properties

```
# 服务端口
server.port=8004
# 服务名
spring.application.name=service-cms

# mysql数据库连接
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/guli?serverTimezone=GMT%2B8
spring.datasource.username=root
spring.datasource.password=root

#返回json的全局时间格式
spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
spring.jackson.time-zone=GMT+8

#配置mapper xml文件的路径
mybatis-plus.mapper-locations=classpath:com/atguigu/cmsservice/mapper/xml/*.xml

#mybatis日志
mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl
```

## 4、创建启动类

创建CmsApplication.java

```
@SpringBootApplication
@ComponentScan({"com.atguigu"}) //指定扫描位置
@MapperScan("com.atguigu.cmsservice.mapper")
public class CmsApplication {
    public static void main(String[] args) {
        SpringApplication.run(CmsApplication.class, args);
    }
}
```

# 二、创建banner服务接口

## 1、创建banner后台管理接口

**banner**后台分页查询、添加、修改、删除接口

```
@RestController
@RequestMapping("/eduservice/banner")
@CrossOrigin
public class CrmBannerController {

    @Autowired
    private CrmBannerService bannerService;

    @ApiOperation(value = "获取Banner分页列表")
    @GetMapping("{page}/{limit}")
    public R index(
            @ApiParam(name = "page", value = "当前页码", required = true)
            @PathVariable Long page,

            @ApiParam(name = "limit", value = "每页记录数", required = true)
            @PathVariable Long limit) {

        Page<CrmBanner> pageParam = new Page<>(page, limit);
        bannerService.pageBanner(pageParam,null);
        return R.ok().data("items", pageParam.getRecords()).data("total",
```

```
pageParam.getTotal());
    }

    @ApiOperation(value = "获取Banner")
    @GetMapping("get/{id}")
    public R get(@PathVariable String id) {
        CrmBanner banner = bannerService.getBannerById(id);
        return R.ok().data("item", banner);
    }

    @ApiOperation(value = "新增Banner")
    @PostMapping("save")
    public R save(@RequestBody CrmBanner banner) {
        bannerService.saveBanner(banner);
        return R.ok();
    }

    @ApiOperation(value = "修改Banner")
    @PutMapping("update")
    public R updateById(@RequestBody CrmBanner banner) {
        bannerService.updateBannerById(banner);
        return R.ok();
    }

    @ApiOperation(value = "删除Banner")
    @DeleteMapping("remove/{id}")
    public R remove(@PathVariable String id) {
        bannerService.removeBannerById(id);
        return R.ok();
    }
}
```

# 2、创建banner前台查询接口

### 首页获取banner数据接口

```
@RestController
@RequestMapping("/educms/banner")
@Api(description = "网站首页Banner列表")
@CrossOrigin //跨域
public class BannerApiController {
```

```
6
7       @Autowired
8       private CrmBannerService bannerService;
9
10      @ApiOperation(value = "获取首页banner")
11      @GetMapping("getAllBanner")
12      public R index() {
13          List<CrmBanner> list = bannerService.selectIndexList();
14          return R.ok().data("bannerList", list);
15      }
16
17  }
```

# 三、实现banner后台管理前端

实现banner后台的添加修改删除和分页查询操作，和其他后台管理模块类似

# 一、新建前端查询课程名师接口

## 1、在service-edu模块创建controller

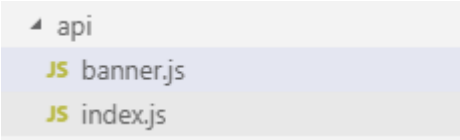
（1）查询最新前4条讲师数据

（2）查询最新前8条课程数据

```
@RestController
@RequestMapping("/eduservice/index")
@CrossOrigin
public class IndexController {

    @Autowired
    private EduCourseService courseService;
    @Autowired
    private EduTeacherService teacherService;

   //查询前8条热门课程，查询前4条名师
    @GetMapping("index")
    public R index() {
        //查询前8条热门课程
        QueryWrapper<EduCourse> wrapper = new QueryWrapper<>();
        wrapper.orderByDesc("id");
        wrapper.last("limit 8");
        List<EduCourse> eduList = courseService.list(wrapper);

        //查询前4条名师
        QueryWrapper<EduTeacher> wrapperTeacher = new QueryWrapper<>();
        wrapperTeacher.orderByDesc("id");
        wrapperTeacher.last("limit 4");
        List<EduTeacher> teacherList = teacherService.list(wrapperTeacher);

        return R.ok().data("eduList",eduList).data("teacherList",teacherList);
    }
}
```

# 一、首页banner数据显示

## 1、创建api文件夹，创建banner.js文件




### banner.js

```
import request from '@/utils/request'
export default {
  getList() {
    return request({
      url: `/educms/banner/getAllBanner`,
      method: 'get'
    })
  }
}
```

## 2、在首页面引入，调用实现

```
<script>
import banner from "@/api/banner"

export default {
  data () {
    return {
      swiperOption: {
        //配置分页
        pagination: {
          el: '.swiper-pagination'//分页的dom节点
        },
        //配置导航
        navigation: {
          nextEl: '.swiper-button-next',//下一页dom节点
          prevEl: '.swiper-button-prev'//前一页dom节点
```

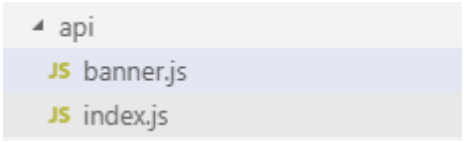
```
        }
      },
      bannerList: {}
    }
  },
  created() {
    this.initDataBanner()
  },
  methods:{
    initDataBanner() {
      banner.getList().then(response => {
        this.bannerList = response.data.data.bannerList
      })
    }
  }
}
</script>
```

## 3、在页面遍历显示banner

```
<!-- 幻灯片 开始 -->
<div v-swiper:mySwiper="swiperOption">
    <div class="swiper-wrapper">
        <div v-for="banner in bannerList" :key="banner.id" class="swiper-slide" style="background: #040B1B;">
            <a target="_blank" :href="banner.linkUrl">
              <img width="100%"
                :src="banner.imageUrl"
                :alt="banner.title"/>
            </a>
        </div>

    </div>
    <div class="swiper-pagination swiper-pagination-white"></div>
    <div class="swiper-button-prev swiper-button-white" slot="button-prev"></div>
    <div class="swiper-button-next swiper-button-white" slot="button-next"></div>
</div>
<!-- 幻灯片 结束 -->
```

# 二、首页显示课程和名师数据

## 1、创建api文件夹，创建index.js文件




### index.js

```
import request from '@/utils/request'
export default {
  getList() {
    return request({
      url: `/eduservice/index`,
      method: 'get'
    })
  }
}
```

## 2、在首页面引入，调用实现

```
<script>
import index from '@/api/index'
import banner from "@/api/banner"

export default {
  data () {
    return {
      swiperOption: {
        //配置分页
        pagination: {
          el: '.swiper-pagination'//分页的dom节点
        },
        //配置导航
        navigation: {
          nextEl: '.swiper-button-next',//下一页dom节点
          prevEl: '.swiper-button-prev'//前一页dom节点
```

```
        }
      },
      teacherList: {},
      courseList: {},
      bannerList: {}

    }
  },
  created() {
    this.initDataBanner()
    this.initDataObj()
  },
  methods:{
    initDataBanner() {
      banner.getList().then(response => {
        this.bannerList = response.data.data.bannerList
      })
    },

    initDataObj() {
      index.getList().then(response => {
        this.teacherList = response.data.data.teacherList
        this.courseList = response.data.data.courseList
      })
    }
  }
}
</script>
```

## 3、在页面遍历显示课程和名师

```
<div id="aCoursesList">
      <!-- 网校课程 开始 -->
      <div>
        <section class="container">
          <header class="comm-title">
            <h2 class="tac">
              <span class="c-333">热门课程</span>
            </h2>
          </header>
```

```
          <div>
            <article class="comm-course-list">
              <ul class="of" id="bna">
                <li v-for="(course, index) in courseList" v-bind:key="index">
                <div class="cc-l-wrap">
                    <section class="course-img">
<!-- ~/assets/photo/course/01.jpg -->
                        <img
                            :src="course.cover"
                            class="img-responsive"
                            :alt="course.title">
        <div class="cc-mask">
            <a :href="'/course/'+course.id" title="开始学习" class="comm-btn c-btn-1">开始学习</a>
            </div>
        </section>
                    <h3 class="hLh30 txtOf mt10">
                <a href="#" :title="course.title" class="course-title fsize18 c-333">{{course.title}}</a>
                    </h3>
                    <section class="mt10 hLh20 of">
                    <span class="fr jgTag bg-green" v-if="Number(course.price) === 0">
                    <i class="c-fff fsize12 f-fA">免费</i>
                    </span>
                    <span class="fr jgTag bg-green" v-else>
                    <i class="c-fff fsize12 f-fA"> ￥{{course.price}}</i>
                    </span>
                    <span class="fl jgAttr c-ccc f-fA">
                    <i class="c-999 f-fA">{{course.buyCount}} 人学习</i>
                                        |
                    <i class="c-999 f-fA">{{course.viewCount}} 人浏览</i>
                    </span>
                    </section>
                    </div>
                    </li>
              </ul>
              <div class="clear"></div>
            </article>
            <section class="tac pt20">
              <a href="#" title="全部课程" class="comm-btn c-btn-2">全部课程</a>
            </section>
          </div>
```

```
        </section>
      </div>
      <!-- /网校课程 结束 -->
      <!-- 网校名师 开始 -->
      <div>
        <section class="container">
          <header class="comm-title">
            <h2 class="tac">
              <span class="c-333">名师大咖</span>
            </h2>
          </header>
          <div>
            <article class="i-teacher-list">
              <ul class="of">
               <li v-for="(teacher,index) in teacherList" v-bind:key="index">
                <section class="i-teach-wrap">
                <div class="i-teach-pic">
                <a :href='"/teacher/"+teacher.id' :title="teacher.name">
                <img :alt="teacher.name" :src="teacher.avatar">
                </a>
                </div>
                <div class="mt10 hLh30 txtOf tac">
                <a :href='"/teacher/"+teacher.id' :title="teacher.name"
class="fsize18 c-666">{{teacher.name}}</a>
                </div>
                <div class="hLh30 txtOf tac">
                <span class="fsize14 c-999">{{teacher.intro}}</span>
                </div>
                <div class="mt15 i-q-txt">
                <p
                                        class="c-999 f-fA"
                >{{teacher.career}}</p>
                </div>
                </section>
                </li>
              </ul>
              <div class="clear"></div>
            </article>
            <section class="tac pt20">
              <a href="#" title="全部讲师" class="comm-btn c-btn-2">全部讲师</a>
            </section>
          </div>
        </section>
```

```
92      </div>
```

# 一、Redis介绍

Redis是当前比较热门的NOSQL系统之一，它是一个开源的使用ANSI c语言编写的key-value存储系统（区别于MySQL的二维表格的形式存储。）。和Memcache类似，但很大程度补偿了Memcache的不足。和Memcache一样，Redis数据都是缓存在计算机内存中，不同的是，Memcache只能将数据缓存到内存中，无法自动定期写入硬盘，这就表示，一断电或重启，内存清空，数据丢失。所以Memcache的应用场景适用于缓存无需持久化的数据。而Redis不同的是它会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，实现数据的持久化。

Redis的特点：

1，Redis读取的速度是110000次/s，写的速度是81000次/s；

2，原子 。Redis的所有操作都是原子性的，同时Redis还支持对几个操作全并后的原子性执行。

3，支持多种数据结构：string（字符串）；list（列表）；hash（哈希），set（集合）；zset(有序集合)

4，持久化，集群部署

5，支持过期时间，支持事务，消息订阅

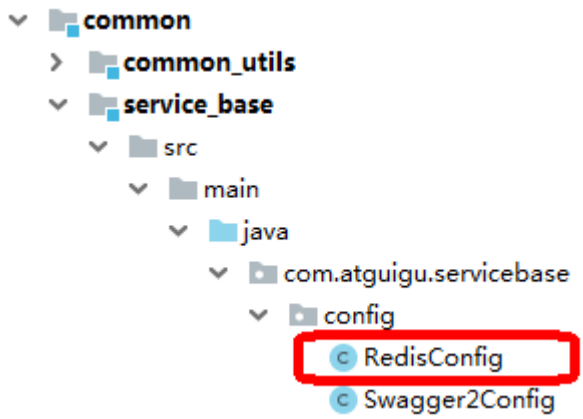
# 二、项目集成Redis

## 1、在common模块添加依赖

由于**redis**缓存是公共应用，所以我们把依赖与配置添加到了**common**模块下面，在**common**模块**pom.xml**下添加以下依赖

```
<!-- redis -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>

<!-- spring2.X集成redis所需common-pool2-->
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-pool2</artifactId>
    <version>2.6.0</version>
</dependency>
```

## 2、在service-base模块添加redis配置类




### RedisConfig.java

```
@EnableCaching
@Configuration
public class RedisConfig extends CachingConfigurerSupport {

    @Bean
    public RedisTemplate<String, Object> redisTemplate(RedisConnectionFactory factory) {
        RedisTemplate<String, Object> template = new RedisTemplate<>();
        RedisSerializer<String> redisSerializer = new StringRedisSerializer();
        Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = new Jackson2JsonRedisSerializer(Object.class);
        ObjectMapper om = new ObjectMapper();
        om.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY);
        om.enableDefaultTyping(ObjectMapper.DefaultTyping.NON_FINAL);
        jackson2JsonRedisSerializer.setObjectMapper(om);
        template.setConnectionFactory(factory);
        //key序列化方式
        template.setKeySerializer(redisSerializer);
        //value序列化
        template.setValueSerializer(jackson2JsonRedisSerializer);
        //value hashmap序列化
        template.setHashValueSerializer(jackson2JsonRedisSerializer);
        return template;
    }

```

```
    @Bean
    public CacheManager cacheManager(RedisConnectionFactory factory) {
        RedisSerializer<String> redisSerializer = new StringRedisSerializer();
        Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = new
Jackson2JsonRedisSerializer(Object.class);
        //解决查询缓存转换异常的问题
        ObjectMapper om = new ObjectMapper();
        om.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY);
        om.enableDefaultTyping(ObjectMapper.DefaultTyping.NON_FINAL);
        jackson2JsonRedisSerializer.setObjectMapper(om);
        // 配置序列化（解决乱码的问题）,过期时间600秒
        RedisCacheConfiguration config =
RedisCacheConfiguration.defaultCacheConfig()
                .entryTtl(Duration.ofSeconds(600))

.serializeKeysWith(RedisSerializationContext.SerializationPair.fromSerializer(redi
sSerializer))

.serializeValuesWith(RedisSerializationContext.SerializationPair.fromSerializer(ja
ckson2JsonRedisSerializer))
                .disableCachingNullValues();
        RedisCacheManager cacheManager = RedisCacheManager.builder(factory)
                .cacheDefaults(config)
                .build();
        return cacheManager;
    }
}
```

# 3、在接口中添加redis缓存

由于首页数据变化不是很频繁，而且首页访问量相对较大，所以我们有必要把首页接口数据缓存到redis缓存中，减少数据库压力和提高访问速度。

改造service-cms模块首页banner接口，首页课程与讲师接口类似

## 3.1 Spring Boot缓存注解

### （1）缓存@Cacheable

根据方法对其返回结果进行缓存，下次请求时，如果缓存存在，则直接读取缓存数据返回；如果缓存不存在，则执行方法，并把返回的结果存入缓存中。一般用在查询方法上。

查看源码，属性值如下：

| 属性/方法名 | 解释 |
| --- | --- |
| value | 缓存名，必填，它指定了你的缓存存放在哪块命名空间 |
| cacheNames | 与 value 差不多，二选一即可 |
| key | 可选属性，可以使用 SpEL 标签自定义缓存的key |

### （2）缓存@CachePut

使用该注解标志的方法，每次都会执行，并将结果存入指定的缓存中。其他方法可以直接从响应的缓存中读取缓存数据，而不需要再去查询数据库。一般用在新增方法上。

查看源码，属性值如下：

| 属性/方法名 | 解释 |
| --- | --- |
| value | 缓存名，必填，它指定了你的缓存存放在哪块命名空间 |
| cacheNames | 与 value 差不多，二选一即可 |
| key | 可选属性，可以使用 SpEL 标签自定义缓存的key |

### （3）缓存@CacheEvict

使用该注解标志的方法，会清空指定的缓存。一般用在更新或者删除方法上

查看源码，属性值如下：

| 属性/方法名 | 解释 |
| --- | --- |
| value | 缓存名，必填，它指定了你的缓存存放在哪块命名空间 |
| cacheNames | 与 value 差不多，二选一即可 |
| key | 可选属性，可以使用 SpEL 标签自定义缓存的key |
| allEntries | 是否清空所有缓存，默认为 false。如果指定为 true，则方法调用后将立即清空所有的缓存 |

| | |
|---|---|
| beforeInvocation | 是否在方法执行前就清空，默认为 false。如果指定为 true，则在方法执行前就会清空缓存 |

## 3.2 启动redis服务

```
[root@online bin]# cd bin
[root@online bin]# ls
dump.rdb          redis-check-aof  redis-cli       redis-server
redis-benchmark   redis-check-rdb  redis-sentinel
[root@online bin]# ./redis-server /etc/redis.conf
[root@online bin]# ./redis-cli
127.0.0.1:6379> keys *
1) "test"
2) "test4"
```

## 3.3连接redis服务可能遇到的问题

（1）**关闭liunx**防火墙

**(2) 找到redis**配置文件， **注释一行配置**

注释掉这句话

```
# bind 127.0.0.1
```

**(3**）如果出现下面错误提示

```
Caused by: io.lettuce.core.RedisConnectionException: DENIED Redis is running in protected mode because protected mode is enabled,
```

**修改 protected-mode  yes**

**改为**

```
protected-mode no
```

## 3.4 banner接口改造

（1）在service-cms模块配置文件添加redis配置

```
spring.redis.host=192.168.44.132
```

```
spring.redis.port=6379
spring.redis.database= 0
spring.redis.timeout=1800000

spring.redis.lettuce.pool.max-active=20
spring.redis.lettuce.pool.max-wait=-1
#最大阻塞等待时间(负数表示没限制)
spring.redis.lettuce.pool.max-idle=5
spring.redis.lettuce.pool.min-idle=0
```

（2）修改CrmBannerServiceImpl，添加redis缓存注解

```
@Service
public class CrmBannerServiceImpl extends ServiceImpl<CrmBannerMapper, CrmBanner>
implements CrmBannerService {

    @Cacheable(value = "banner", key = "'selectIndexList'")
    @Override
    public List<CrmBanner> selectIndexList() {
        List<CrmBanner> list = baseMapper.selectList(new
QueryWrapper<CrmBanner>().orderByDesc("sort"));
        return list;
    }

    @Override
    public void pageBanner(Page<CrmBanner> pageParam, Object o) {
        baseMapper.selectPage(pageParam,null);
    }

    @Override
    public CrmBanner getBannerById(String id) {
        return baseMapper.selectById(id);
    }

    @CacheEvict(value = "banner", allEntries=true)
    @Override
    public void saveBanner(CrmBanner banner) {
        baseMapper.insert(banner);
    }

```

```
    @CacheEvict(value = "banner", allEntries=true)
    @Override
    public void updateBannerById(CrmBanner banner) {
        baseMapper.updateById(banner);
    }

    @CacheEvict(value = "banner", allEntries=true)
    @Override
    public void removeBannerById(String id) {
        baseMapper.deleteById(id);
    }
}
```

（3）在**redis**添加了**key**

```
127.0.0.1:6379> keys *
1) "test"
2) "test4"
3) "banner::selectIndexList"
4) "1234"
5) "aa"
```

（4）通过源码查看到**key**生成的规则

```
package org.springframework.data.redis.cache;

@FunctionalInterface
public interface CacheKeyPrefix {
    String compute(String var1);

    static CacheKeyPrefix simple() {
        return (name) -> {
            return name + "::";
        };
    }
```

# 1. 用户登录业务介绍

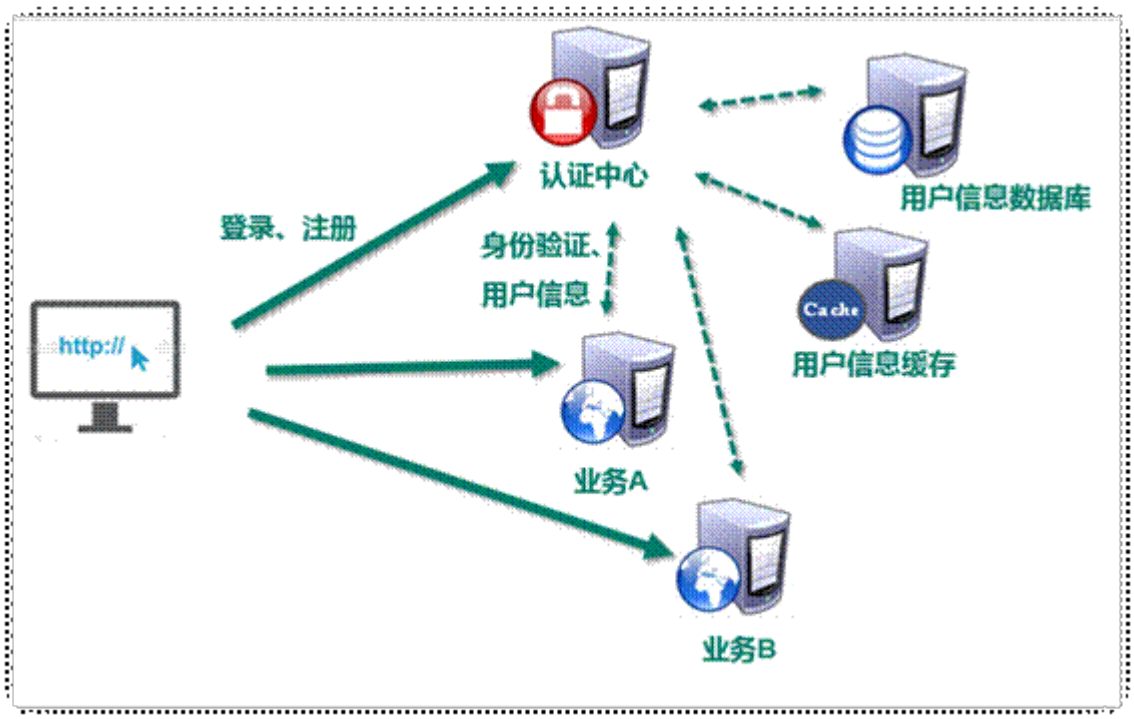
## 1.1. 单一服务器模式

早期单一服务器，用户认证。


缺点：单点性能压力，无法扩展

## 1.2. SSO(single sign on)模式

分布式，SSO(single sign on)模式




优点：

用户身份信息独立管理，更好的分布式管理。

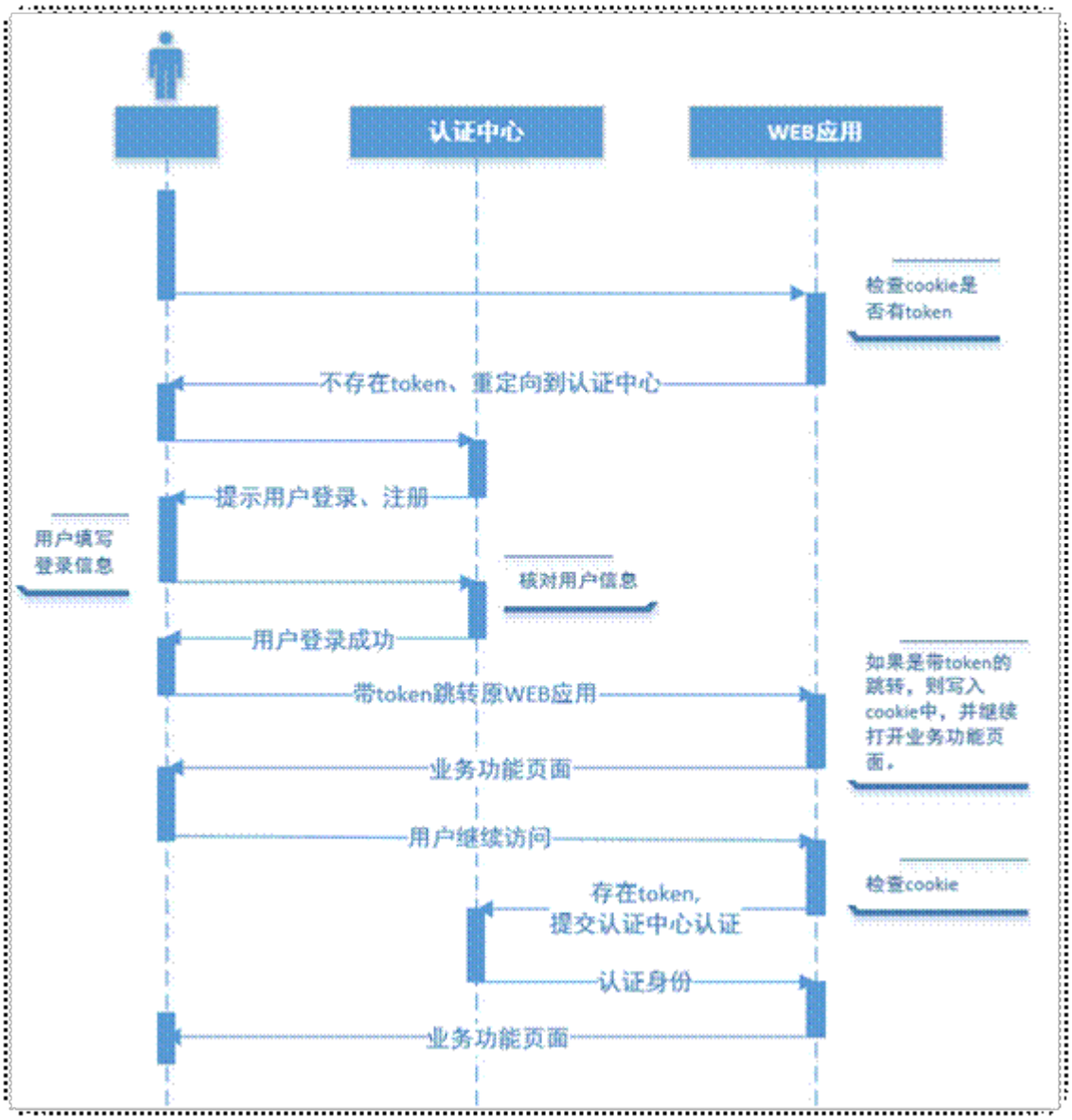
可以自己扩展安全策略

缺点：

认证服务器访问压力较大。

## 1.3. Token模式

业务流程图{用户访问业务时，必须登录的流程}




优点：

无状态： token无状态，session有状态的

基于标准化: 你的API可以采用标准化的 JSON Web Token (JWT)

缺点：

占用带宽

无法在服务器端销毁

注：基于微服务开发，选择token的形式相对较多，因此我使用token作为用户认证的标准

# 一、使用JWT进行跨域身份验证

## 1、传统用户身份验证


Internet服务无法与用户身份验证分开。一般过程如下：

1. 用户向服务器发送用户名和密码。
2. 验证服务器后，相关数据（如用户角色，登录时间等）将保存在当前会话中。
3. 服务器向用户返回session_id，session信息都会写入到用户的Cookie。
4. 用户的每个后续请求都将通过在Cookie中取出session_id传给服务器。
5. 服务器收到session_id并对比之前保存的数据，确认用户的身份。

这种模式最大的问题是，没有分布式架构，无法支持横向扩展。

## 2、解决方案

1. session广播
2. 将透明令牌存入cookie，将用户身份信息存入redis

**另外一种灵活的解决方案：**

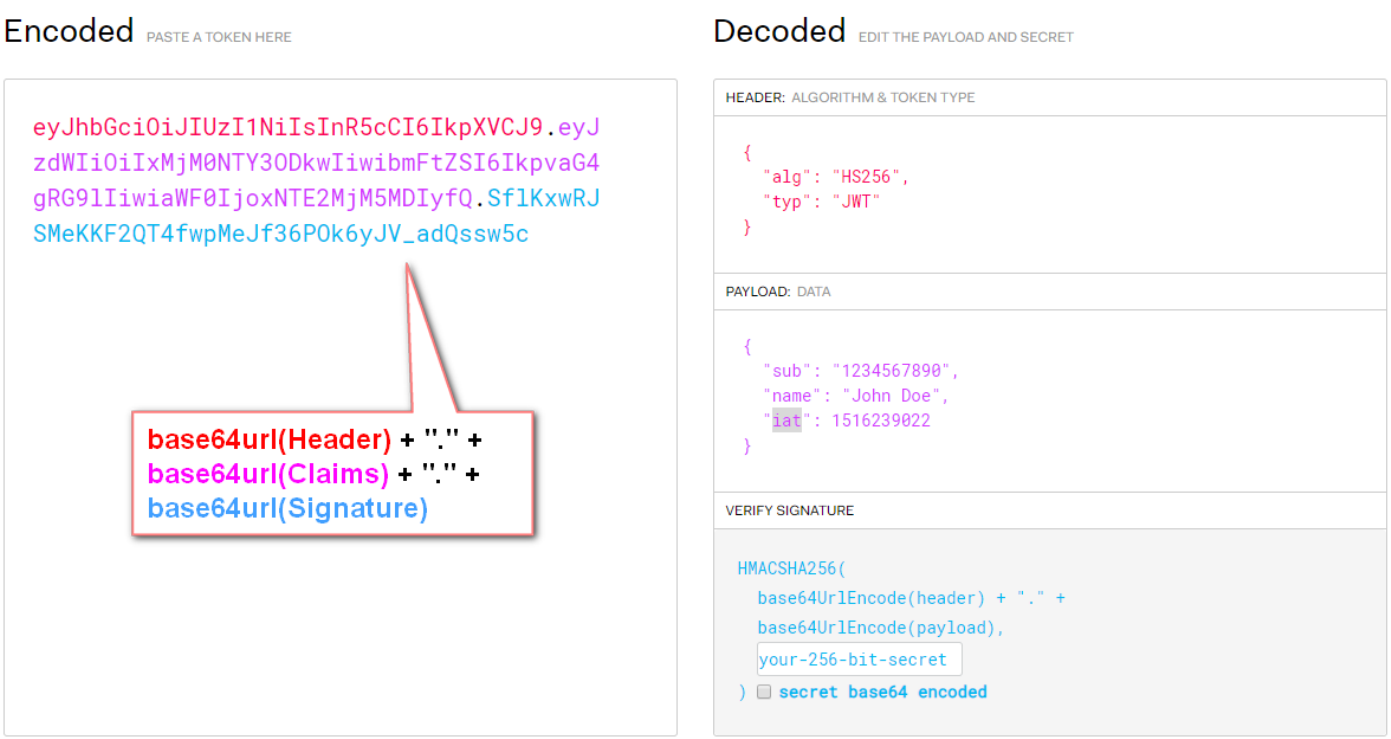
使用自包含令牌，通过客户端保存数据，而服务器不保存会话数据。 JWT是这种解决方案的代表。

# 二、JWT令牌

# 1、访问令牌的类型


# 2、JWT的组成

典型的，一个JWT看起来如下图：




该对象为一个很长的字符串，字符之间通过"."分隔符分为三个子串。

每一个子串表示了一个功能块，总共有以下三个部分：JWT头、有效载荷和签名

### JWT头

JWT头部分是一个描述JWT元数据的JSON对象，通常如下所示。

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

在上面的代码中，alg属性表示签名使用的算法，默认为HMAC SHA256（写为HS256）；typ属性表示令牌的类型，JWT令牌统一写为JWT。最后，使用Base64 URL算法将上述JSON对象转换为字符串保存。

### 有效载荷

有效载荷部分，是JWT的主体内容部分，也是一个JSON对象，包含需要传递的数据。 JWT指定七个默认字段供选择。

```
iss: 发行人
exp: 到期时间
sub: 主题
aud: 用户
nbf: 在此之前不可用
iat: 发布时间
jti: JWT ID用于标识该JWT
```

除以上默认字段外，我们还可以自定义私有字段，如下例：

```
{
  "sub": "1234567890",
  "name": "Helen",
  "admin": true
}
```

请注意，默认情况下JWT是未加密的，任何人都可以解读其内容，因此不要构建隐私信息字段，存放保密信息，以防止信息泄露。

JSON对象也使用Base64 URL算法转换为字符串保存。

### 签名哈希

签名哈希部分是对上面两部分数据签名，通过指定的算法生成哈希，以确保数据不会被篡改。

首先，需要指定一个密码（secret）。该密码仅仅为保存在服务器中，并且不能向用户公开。然后，使用

标头中指定的签名算法（默认情况下为HMAC SHA256）根据以下公式生成签名。

```
HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(claims), secret)
```

在计算出签名哈希后，JWT头，有效载荷和签名哈希的三个部分组合成一个字符串，每个部分用"."分隔，就构成整个JWT对象。

### Base64URL算法

如前所述，JWT头和有效载荷序列化的算法都用到了Base64URL。该算法和常见Base64算法类似，稍有差别。

作为令牌的JWT可以放在URL中（例如api.example/?token=xxx）。 Base64中用的三个字符是"+"，"/"和"="，由于在URL中有特殊含义，因此Base64URL中对他们做了替换："="去掉，"+"用"-"替换，"/"用"_"替换，这就是Base64URL算法。

## 3、JWT的原则

JWT的原则是在服务器身份验证之后，将生成一个JSON对象并将其发送回用户，如下所示。

```
{
  "sub": "1234567890",
  "name": "Helen",
  "admin": true
}
```

之后，当用户与服务器通信时，客户在请求中发回JSON对象。服务器仅依赖于这个JSON对象来标识用户。为了防止用户篡改数据，服务器将在生成对象时添加签名。

服务器不保存任何会话数据，即服务器变为无状态，使其更容易扩展。

## 4、JWT的用法

客户端接收服务器返回的JWT，将其存储在Cookie或localStorage中。

此后，客户端将在与服务器交互中都会带JWT。如果将它存储在Cookie中，就可以自动发送，但是不会跨域，因此一般是将它放入HTTP请求的Header Authorization字段中。当跨域时，也可以将JWT被放置于POST请求的数据主体中。

## 5、JWT问题和趋势

- JWT不仅可用于认证，还可用于信息交换。善用JWT有助于减少服务器请求数据库的次数。
- 生产的token可以包含基本信息，比如id、用户昵称、头像等信息，避免再次查库
- 存储在客户端，不占用服务端的内存资源
- JWT默认不加密，但可以加密。生成原始令牌后，可以再次对其进行加密。
- 当JWT未加密时，一些私密数据无法通过JWT传输。
- JWT的最大缺点是服务器不保存会话状态，所以在使用期间不可能取消令牌或更改令牌的权限。也就是说，一旦JWT签发，在有效期内将会一直有效。
- JWT本身包含认证信息，token是经过base64编码，所以可以解码，因此token加密前的对象不应该包含敏感信息，一旦信息泄露，任何人都可以获得令牌的所有权限。为了减少盗用，JWT的有效期不宜设置太长。对于某些重要操作，用户在使用时应该每次都进行进行身份验证。
- 为了减少盗用和窃取，JWT不建议使用HTTP协议来传输代码，而是使用加密的HTTPS协议进行传输。

# 三、整合JWT令牌

## 1、在common_utils模块中添加jwt工具依赖

在pom中添加

```
<dependencies>
    <!-- JWT-->
    <dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt</artifactId>
    </dependency>
</dependencies>
```

## 2、创建JWT工具类

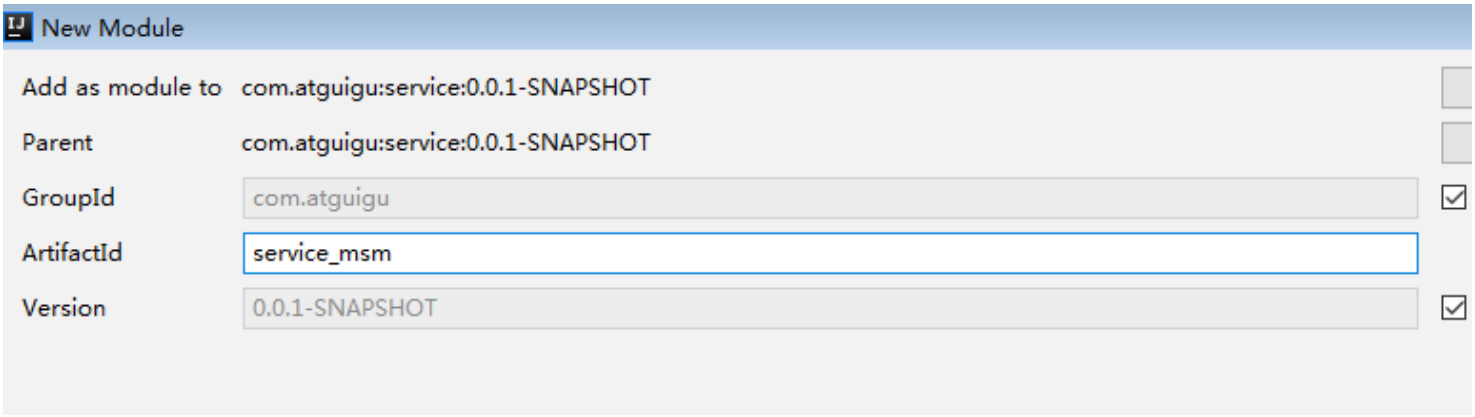
```
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jws;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
```

```java
import org.springframework.util.StringUtils;

import javax.servlet.http.HttpServletRequest;
import java.util.Date;

/**
 * @author
 */
public class JwtUtils {

    public static final long EXPIRE = 1000 * 60 * 60 * 24;
    public static final String APP_SECRET = "ukc8BDbRigUDaY6pZFfWus2jZWLPHO";

    public static String getJwtToken(String id, String nickname){

        String JwtToken = Jwts.builder()
                .setHeaderParam("typ", "JWT")
                .setHeaderParam("alg", "HS256")
                .setSubject("guli-user")
                .setIssuedAt(new Date())
                .setExpiration(new Date(System.currentTimeMillis() + EXPIRE))
                .claim("id", id)
                .claim("nickname", nickname)
                .signWith(SignatureAlgorithm.HS256, APP_SECRET)
                .compact();

        return JwtToken;
    }

    /**
     * 判断token是否存在与有效
     * @param jwtToken
     * @return
     */
    public static boolean checkToken(String jwtToken) {
        if(StringUtils.isEmpty(jwtToken)) return false;
        try {
            Jwts.parser().setSigningKey(APP_SECRET).parseClaimsJws(jwtToken);
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
        return true;
```

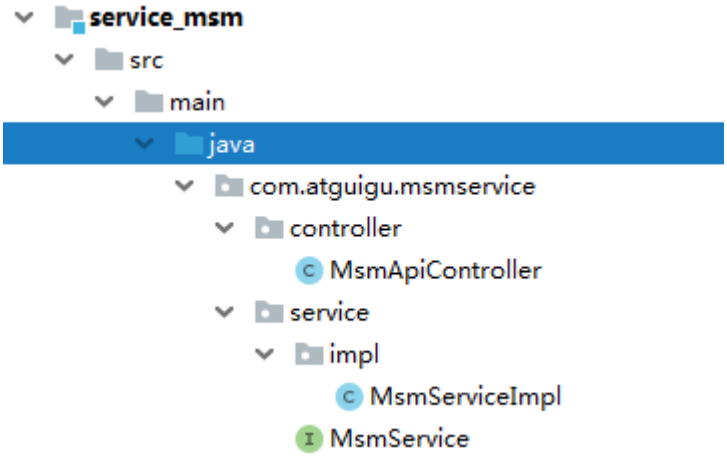
```
    }

    /**
     * 判断token是否存在与有效
     * @param request
     * @return
     */
    public static boolean checkToken(HttpServletRequest request) {
        try {
            String jwtToken = request.getHeader("token");
            if(StringUtils.isEmpty(jwtToken)) return false;
            Jwts.parser().setSigningKey(APP_SECRET).parseClaimsJws(jwtToken);
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
        return true;
    }

    /**
     * 根据token获取会员id
     * @param request
     * @return
     */
    public static String getMemberIdByJwtToken(HttpServletRequest request) {
        String jwtToken = request.getHeader("token");
        if(StringUtils.isEmpty(jwtToken)) return "";
        Jws<Claims> claimsJws =
Jwts.parser().setSigningKey(APP_SECRET).parseClaimsJws(jwtToken);
        Claims claims = claimsJws.getBody();
        return (String)claims.get("id");
    }
}
```

# 一、新建短信微服务

## 1、在service模块下创建子模块service-msm




## 2、创建controller和service代码




## 3、配置application.properties

```
# 服务端口
server.port=8006
# 服务名
spring.application.name=service-msm

# mysql数据库连接
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/guli?serverTimezone=GMT%2B8
spring.datasource.username=root
```

```
spring.datasource.password=root

spring.redis.host=192.168.44.131
spring.redis.port=6379
spring.redis.database= 0
spring.redis.timeout=1800000

spring.redis.lettuce.pool.max-active=20
spring.redis.lettuce.pool.max-wait=-1
  #最大阻塞等待时间(负数表示没限制)
spring.redis.lettuce.pool.max-idle=5
spring.redis.lettuce.pool.min-idle=0
  #最小空闲


#返回json的全局时间格式
spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
spring.jackson.time-zone=GMT+8

#配置mapper xml文件的路径
mybatis-plus.mapper-locations=classpath:com/atguigu/cmsservice/mapper/xml/*.xml

#mybatis日志
mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl
```

## 4、创建启动类

创建ServiceMsmApplication.java

```
@ComponentScan({"com.atguigu"})
@SpringBootApplication(exclude = DataSourceAutoConfiguration.class)//取消数据源自动配置
public class ServiceMsmApplication {
    public static void main(String[] args) {
        SpringApplication.run(ServiceMsmApplication.class, args);
    }
}
```

# 二、阿里云短信服务

帮助文档：

https://help.aliyun.com/product/44282.html?spm=5176.10629532.0.0.38311cbeYzBm73

# 1、开通阿里云短信服务



## 2、添加签名管理与模板管理

### （1）添加模板管理

选择 国内消息 - 模板管理 - 添加模板


点击 添加模板，进入到添加页面，输入模板信息


点击提交，等待审核，审核通过后可以使用

## （2）添加签名管理

**选择 国内消息 - 签名管理 - 添加签名**


**点击添加签名，进入添加页面，填入相关信息**

**注意：签名要写的有实际意义**


**点击提交，等待审核，审核通过后可以使**

# 三、编写发送短信接口

## 1、在service-msm的pom中引入依赖

```
<dependencies>
    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>fastjson</artifactId>
    </dependency>
    <dependency>
        <groupId>com.aliyun</groupId>
        <artifactId>aliyun-java-sdk-core</artifactId>
    </dependency>
</dependencies>
```

## 2、编写controller，根据手机号发送短信

```java
@RestController
@RequestMapping("/api/msm")
@CrossOrigin //跨域
public class MsmApiController {

    @Autowired
    private MsmService msmService;

    @Autowired
    private RedisTemplate<String, String> redisTemplate;

    @GetMapping(value = "/send/{phone}")
    public R code(@PathVariable String phone) {
        String code = redisTemplate.opsForValue().get(phone);
        if(!StringUtils.isEmpty(code)) return R.ok();

        code = RandomUtil.getFourBitRandom();
        Map<String,Object> param = new HashMap<>();
        param.put("code", code);
        boolean isSend = msmService.send(phone, "SMS_180051135", param);
        if(isSend) {
            redisTemplate.opsForValue().set(phone, code,5,TimeUnit.MINUTES);
            return R.ok();
        } else {
            return R.error().message("发送短信失败");
        }
    }
}
```

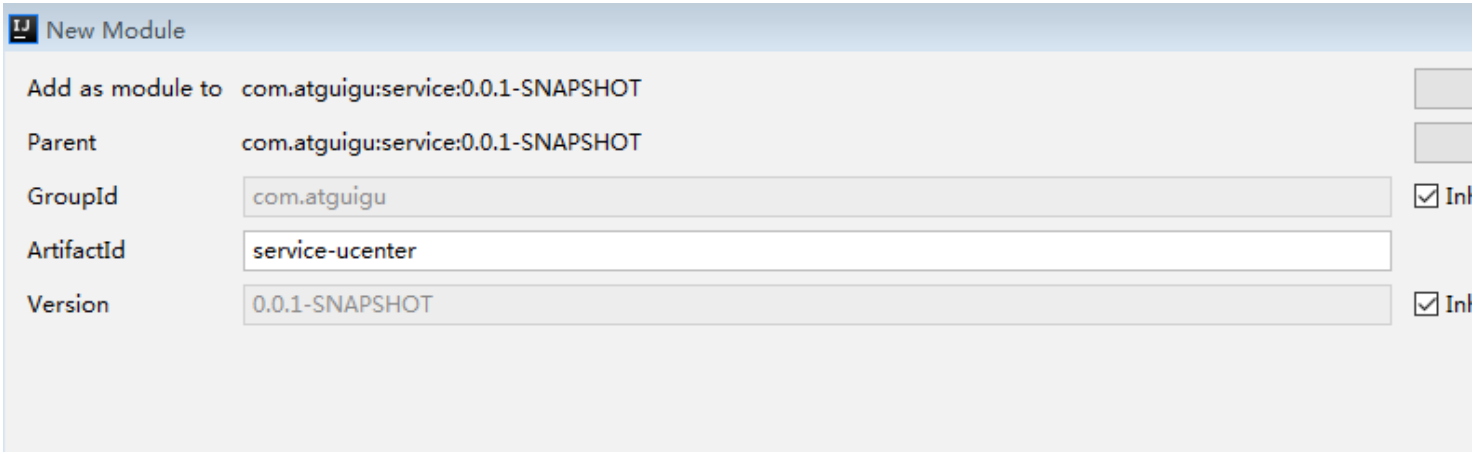
# 3、编写service

```java
 @Service
public class MsmServiceImpl implements MsmService {

    /**
     * 发送短信
     */
    public boolean send(String PhoneNumbers, String templateCode,
Map<String,Object> param) {

        if(StringUtils.isEmpty(PhoneNumbers)) return false;
```

```
10
11          DefaultProfile profile =
12                  DefaultProfile.getProfile("default", "LTAIq6nIPY09VROj",
"FQ7UcixT9wEqMv9F35nORPqKr8XkTF");
13          IAcsClient client = new DefaultAcsClient(profile);
14
15          CommonRequest request = new CommonRequest();
16          //request.setProtocol(ProtocolType.HTTPS);
17          request.setMethod(MethodType.POST);
18          request.setDomain("dysmsapi.aliyuncs.com");
19          request.setVersion("2017-05-25");
20          request.setAction("SendSms");
21
22          request.putQueryParameter("PhoneNumbers", PhoneNumbers);
23          request.putQueryParameter("SignName", "我的谷粒在线教育网站");
24          request.putQueryParameter("TemplateCode", templateCode);
25          request.putQueryParameter("TemplateParam",
JSONObject.toJSONString(param));
26
27          try {
28              CommonResponse response = client.getCommonResponse(request);
29              System.out.println(response.getData());
30              return response.getHttpResponse().isSuccess();
31          } catch (ServerException e) {
32              e.printStackTrace();
33          } catch (ClientException e) {
34              e.printStackTrace();
35          }
36          return false;
37      }
38  }
```

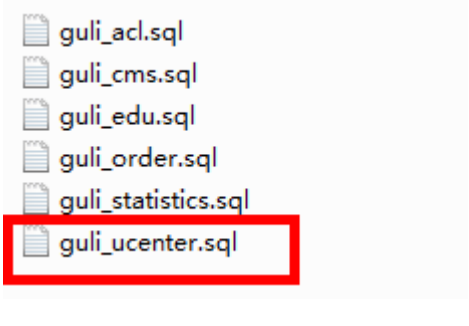
# 一、新建用户微服务

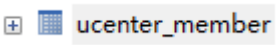
## 1、在service模块下创建子模块service-ucenter



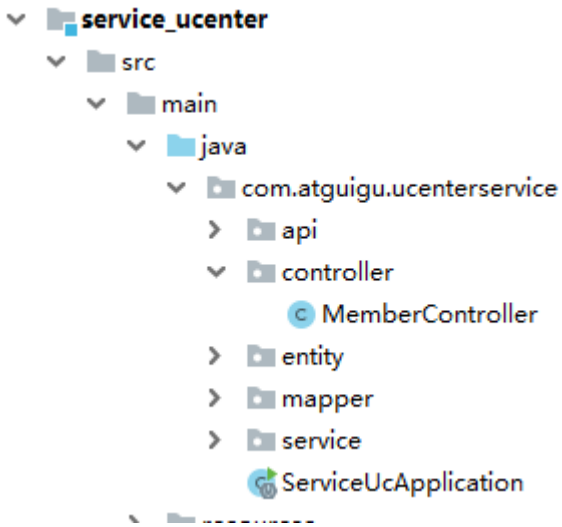

## 2、使用代码生成器生成代码

（1）创建**ucenter_member**表




（2）生成代码



# 3、配置application.properties

```
# 服务端口
server.port=8005
# 服务名
spring.application.name=service-ucenter

# mysql数据库连接
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/guli?serverTimezone=GMT%2B8
spring.datasource.username=root
spring.datasource.password=root

spring.redis.host=192.168.44.131
spring.redis.port=6379
spring.redis.database= 0
spring.redis.timeout=1800000
spring.redis.lettuce.pool.max-active=20
spring.redis.lettuce.pool.max-wait=-1
  #最大阻塞等待时间(负数表示没限制)
spring.redis.lettuce.pool.max-idle=5
spring.redis.lettuce.pool.min-idle=0
#最小空闲

#返回json的全局时间格式
spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
spring.jackson.time-zone=GMT+8

#配置mapper xml文件的路径
mybatis-plus.mapper-locations=classpath:com/atguigu/cmsservice/mapper/xml/*.xml

#mybatis日志
mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl
```

# 4、创建启动类

创建ServiceUcApplication.java

```
@ComponentScan({"com.atguigu"})
```

```
@SpringBootApplication//取消数据源自动配置
@MapperScan("com.atguigu.ucenterservice.mapper")
public class ServiceUcApplication {
    public static void main(String[] args) {
        SpringApplication.run(ServiceUcApplication.class, args);
    }
}
```

# 二、创建登录和注册接口

## 1、创建LoginVo和RegisterVo用于数据封装

### LoginVo

```
@Data
@ApiModel(value="登录对象", description="登录对象")
public class LoginVo {

    @ApiModelProperty(value = "手机号")
    private String mobile;

    @ApiModelProperty(value = "密码")
    private String password;
}
```

### RegisterVo

```
@Data
@ApiModel(value="注册对象", description="注册对象")
public class RegisterVo {

    @ApiModelProperty(value = "昵称")
    private String nickname;

    @ApiModelProperty(value = "手机号")
    private String mobile;

    @ApiModelProperty(value = "密码")
```

```
    private String password;

    @ApiModelProperty(value = "验证码")
    private String code;
}
```

# 2、创建controller编写登录和注册方法

### MemberApiController.java

```
@RestController
@RequestMapping("/ucenterservice/apimember")
@CrossOrigin
public class MemberApiController {

    @Autowired
    private MemberService memberService;

    @ApiOperation(value = "会员登录")
    @PostMapping("login")
    public R login(@RequestBody LoginVo loginVo) {
        String token = memberService.login(loginVo);
        return R.ok().data("token", token);
    }

    @ApiOperation(value = "会员注册")
    @PostMapping("register")
    public R register(@RequestBody RegisterVo registerVo){
        memberService.register(registerVo);
        return R.ok();
    }
}
```

# 3、创建service接口和实现类

```
@Service
public class MemberServiceImpl extends ServiceImpl<MemberMapper, Member>
implements MemberService {
```

```
3
4       @Autowired
5       private RedisTemplate<String, String> redisTemplate;
6
7       /**
8        * 会员登录
9        * @param loginVo
10       * @return
11       */
12      @Override
13      public String login(LoginVo loginVo) {
14          String mobile = loginVo.getMobile();
15          String password = loginVo.getPassword();
16
17          //校验参数
18          if(StringUtils.isEmpty(mobile) ||
19                  StringUtils.isEmpty(password) ||
20                  StringUtils.isEmpty(mobile)) {
21              throw new GuliException(20001,"error");
22          }
23
24          //获取会员
25          Member member = baseMapper.selectOne(new
QueryWrapper<Member>().eq("mobile", mobile));
26          if(null == member) {
27              throw new GuliException(20001,"error");
28          }
29
30          //校验密码
31          if(!MD5.encrypt(password).equals(member.getPassword())) {
32              throw new GuliException(20001,"error");
33          }
34
35          //校验是否被禁用
36          if(member.getIsDisabled()) {
37              throw new GuliException(20001,"error");
38          }
39
40          //使用JWT生成token字符串
41          String token = JwtUtils.getJwtToken(member.getId(), member.getNickname());
42          return token;
43      }
44
```

```
    /**
     * 会员注册
     * @param registerVo
     */
    @Override
    public void register(RegisterVo registerVo) {
        //获取注册信息，进行校验
        String nickname = registerVo.getNickname();
        String mobile = registerVo.getMobile();
        String password = registerVo.getPassword();
        String code = registerVo.getCode();

        //校验参数
        if(StringUtils.isEmpty(mobile) ||
                StringUtils.isEmpty(mobile) ||
                StringUtils.isEmpty(password) ||
                StringUtils.isEmpty(code)) {
            throw new GuliException(20001,"error");
        }

        //校验校验验证码
        //从redis获取发送的验证码
        String mobleCode = redisTemplate.opsForValue().get(mobile);
        if(!code.equals(mobleCode)) {
            throw new GuliException(20001,"error");
        }

        //查询数据库中是否存在相同的手机号码
        Integer count = baseMapper.selectCount(new
QueryWrapper<Member>().eq("mobile", mobile));
        if(count.intValue() > 0) {
            throw new GuliException(20001,"error");
        }

        //添加注册信息到数据库
        Member member = new Member();
        member.setNickname(nickname);
        member.setMobile(registerVo.getMobile());
        member.setPassword(MD5.encrypt(password));
        member.setIsDisabled(false);

member.setAvatar("http://thirdwx.qlogo.cn/mmopen/vi_32/DYAIOgq83eoj0hHXhgJNOTSOFsS
4uZs8x1ConecaVOB8eIl115xmJZcT4oCicvia7wMEufibKtTLqiaJeanU2Lpg3w/132");
```

```
        this.save(member);
    }
}
```

# 三、创建接口根据token获取用户信息

在**MemberApiController中创建方法**

```
@ApiOperation(value = "根据token获取登录信息")
    @GetMapping("auth/getLoginInfo")
    public R getLoginInfo(HttpServletRequest request){
        try {
            String memberId = JwtUtils.getMemberIdByJwtToken(request);
            LoginInfoVo loginInfoVo = memberService.getLoginInfo(memberId);
            return R.ok().data("item", loginInfoVo);
        }catch (Exception e){
            e.printStackTrace();
            throw new GuliException(20001,"error");
        }
    }
```

创建**service**

```
    @Override
    public LoginInfoVo getLoginInfo(String memberId) {
        Member member = baseMapper.selectById(memberId);
        LoginInfoVo loginInfoVo = new LoginInfoVo();
        BeanUtils.copyProperties(member, loginInfoVo);
        return loginInfoVo;
    }
```

# 一、在nuxt环境中安装插件

## 1、安装element-ui 和 vue-qriously

（1）执行命令安装

npm install element-ui

npm install vue-qriously

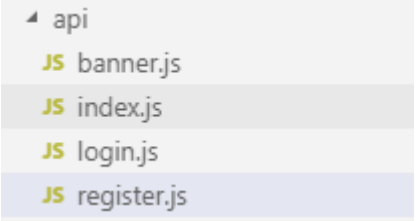
## 2、修改配置文件 nuxt-swiper-plugin.js，使用插件

nuxt-swiper-plugin.js

```
import Vue from 'vue'
import VueAwesomeSwiper from 'vue-awesome-swiper/dist/ssr'
import VueQriously from 'vue-qriously'
import ElementUI from 'element-ui' //element-ui的全部组件
import 'element-ui/lib/theme-chalk/index.css'//element-ui的css
Vue.use(ElementUI) //使用elementUI
Vue.use(VueQriously)
Vue.use(VueAwesomeSwiper)
```

# 二、用户注册功能前端整合

## 1、在api文件夹中创建注册的js文件，定义接口




**register.js**

```
import request from '@/utils/request'
```

```
export default {
  //根据手机号码发送短信
  getMobile(mobile) {
    return request({
      url: `/edumsm/send/${mobile}`,
      method: 'get'
    })
  },
  //用户注册
  submitRegister(formItem) {
    return request({
      url: `/ucenterservice/apimember/register`,
      method: 'post',
      data: formItem
    })
  }
}
```

# 2、在pages文件夹中创建注册页面，调用方法

## （1）在layouts创建布局页面

## sign.vue

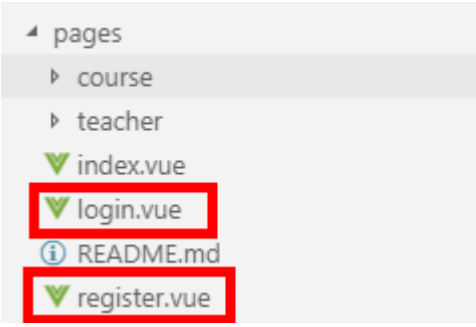
```
<template>
  <div class="sign">
    <!--标题-->
    <div class="logo">
      <img src="~/assets/img/logo.png" alt="logo">
    </div>
    <!--表单-->
    <nuxt/>
  </div>
</template>
```

## （2）创建注册页面

**修改layouts文件夹里面default.vue页面，修改登录和注册超链接地址**

```
<!-- / nav -->
<ul class="h-r-login">
  <li id="no-login">
    <a href="/login" title="登录">
      <em class="icon18 login-icon"> </em>
      <span class="vam ml5">登录</span>
    </a>
    |
    <a href="/register" title="注册">
      <span class="vam ml5">注册</span>
    </a>
  </li>
```

**在pages文件夹下，创建注册和登录页面**




**register.vue**

```
<template>
  <div class="main">
    <div class="title">
      <a href="/login">登录</a>
      <span>·</span>
      <a class="active" href="/register">注册</a>
    </div>

    <div class="sign-up-container">
      <el-form ref="userForm" :model="params">

        <el-form-item class="input-prepend restyle" prop="nickname" :rules="[{ required: true, message: '请输入你的昵称', trigger: 'blur' }]">
          <div>
            <el-input type="text" placeholder="你的昵称" v-
```

```
model="params.nickname"/>
            <i class="iconfont icon-user"/>
          </div>
        </el-form-item>

        <el-form-item class="input-prepend restyle no-radius" prop="mobile"
:rules="[{ required: true, message: '请输入手机号码', trigger: 'blur' },{validator:
checkPhone, trigger: 'blur'}]">
          <div>
            <el-input type="text" placeholder="手机号" v-model="params.mobile"/>
            <i class="iconfont icon-phone"/>
          </div>
        </el-form-item>

        <el-form-item class="input-prepend restyle no-radius" prop="code" :rules="
[{ required: true, message: '请输入验证码', trigger: 'blur' }]">
          <div style="width: 100%;display: block;float: left;position: relative">
            <el-input type="text" placeholder="验证码" v-model="params.code"/>
            <i class="iconfont icon-phone"/>
          </div>
          <div class="btn" style="position:absolute;right: 0;top: 6px;width:
40%;">
            <a href="javascript:" type="button" @click="getCodeFun()"
:value="codeTest" style="border: none;background-color: none">{{codeTest}}</a>
          </div>
        </el-form-item>

        <el-form-item class="input-prepend" prop="password" :rules="[{ required:
true, message: '请输入密码', trigger: 'blur' }]">
          <div>
            <el-input type="password" placeholder="设置密码" v-
model="params.password"/>
            <i class="iconfont icon-password"/>
          </div>
        </el-form-item>

        <div class="btn">
          <input type="button" class="sign-up-button" value="注册"
@click="submitRegister()">
        </div>
        <p class="sign-up-msg">
          点击 “注册” 即表示您同意并愿意遵守简书
          <br>
          <a target="_blank" href="http://www.jianshu.com/p/c44d171298ce">用户协
```

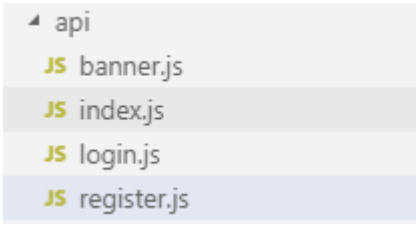
```
议</a>
          和
          <a target="_blank" href="http://www.jianshu.com/p/2ov8x3">隐私政策</a> 。
        </p>
      </el-form>
      <!-- 更多注册方式 -->
      <div class="more-sign">
        <h6>社交帐号直接注册</h6>
        <ul>
          <li><a id="weixin" class="weixin" target="_blank"
href="http://huaan.free.idcfengye.com/api/ucenter/wx/login"><i
            class="iconfont icon-weixin"/></a></li>
          <li><a id="qq" class="qq" target="_blank" href="#"><i class="iconfont
icon-qq"/></a></li>
        </ul>
      </div>
    </div>
  </div>
</template>

<script>
  import '~/assets/css/sign.css'
  import '~/assets/css/iconfont.css'

  import registerApi from '@/api/register'

  export default {
    layout: 'sign',
    data() {
      return {
        params: {
          mobile: '',
          code: '',
          nickname: '',
          password: ''
        },
        sending: true,      //是否发送验证码
        second: 60,         //倒计时间
        codeTest: '获取验证码'
      }
    },
    methods: {
      getCodeFun() {
```

```
        //sending = false
        //his.sending原为true,请求成功，!this.sending == true，主要是防止有人把disabled属性去掉，多次点击;
        if (!this.sending)
          return;

        //debugger
        // prop 换成你想监听的prop字段
        this.$refs.userForm.validateField('mobile', (errMsg) => {
          if (errMsg == '') {
            registerApi.getMobile(this.params.mobile).then(res => {
              this.sending = false;
              this.timeDown();
            });
          }
        })
      },

      timeDown() {
        let result = setInterval(() => {
          --this.second;
          this.codeTest = this.second
          if (this.second < 1) {
            clearInterval(result);
            this.sending = true;
            //this.disabled = false;
            this.second = 60;
            this.codeTest = "获取验证码"
          }
        }, 1000);

      },
      submitRegister() {
        this.$refs['userForm'].validate((valid) => {
          if (valid) {
            registerApi.submitRegister(this.params).then(response => {
              //提示注册成功
              this.$message({
                type: 'success',
                message: "注册成功"
              })
              this.$router.push({path: '/login'})
            })
```

```
          }
        })
      },

      checkPhone (rule, value, callback) {
        //debugger
        if (!(/^1[34578]\d{9}$/.test(value))) {
          return callback(new Error('手机号码格式不正确'))
        }
        return callback()
      }
    }
  }
</script>

```

## 三、用户登录功能前端整合

### 1、在api文件夹中创建登录的js文件，定义接口

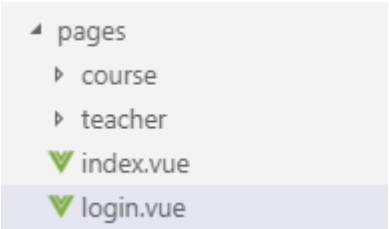



**login.js**

```
import request from '@/utils/request'
export default {
  //登录
  submitLogin(userInfo) {
    return request({
      url: `/ucenterservice/apimember/login`,
      method: 'post',
      data: userInfo
    })
  },
  //根据token获取用户信息
```

```
  getLoginInfo() {
    return request({
      url: `/ucenterservice/apimember/auth/getLoginInfo`,
      method: 'get',
     // headers: {'token': cookie.get('guli_token')}
    })
    //headers: {'token': cookie.get('guli_token')}
  }
}
```

## 2、在pages文件夹中创建登录页面，调用方法




### （1）安装**js-cookie**插件

```
npm install js-cookie
```

### （2）login.vue

```
<template>
  <div class="main">
    <div class="title">
      <a class="active" href="/login">登录</a>
      <span>·</span>
      <a href="/register">注册</a>
    </div>

    <div class="sign-up-container">
      <el-form ref="userForm" :model="user">

        <el-form-item class="input-prepend restyle" prop="mobile" :rules="[{
required: true, message: '请输入手机号码', trigger: 'blur' },{validator:
checkPhone, trigger: 'blur'}]">
          <div >
            <el-input type="text" placeholder="手机号" v-model="user.mobile"/>
```

```
            <i class="iconfont icon-phone" />
          </div>
        </el-form-item>

        <el-form-item class="input-prepend" prop="password" :rules="[{ required: true, message: '请输入密码', trigger: 'blur' }]">
          <div>
            <el-input type="password" placeholder="密码" v-model="user.password"/>
            <i class="iconfont icon-password"/>
          </div>
        </el-form-item>

        <div class="btn">
          <input type="button" class="sign-in-button" value="登录" @click="submitLogin()">
        </div>
      </el-form>
      <!-- 更多登录方式 -->
      <div class="more-sign">
        <h6>社交帐号登录</h6>
        <ul>
          <li><a id="weixin" class="weixin" target="_blank" href="http://qy.free.idcfengye.com/api/ucenter/weixinLogin/login"><i class="iconfont icon-weixin"/></a></li>
          <li><a id="qq" class="qq" target="_blank" href="#"><i class="iconfont icon-qq"/></a></li>
        </ul>
      </div>
    </div>

  </div>
</template>

<script>
  import '~/assets/css/sign.css'
  import '~/assets/css/iconfont.css'
  import cookie from 'js-cookie'

  import loginApi from '@/api/login'
  export default {
    layout: 'sign',

    data () {
      return {
```

```
        user:{
          mobile:'',
          password:''
        },
        loginInfo:{}
      }
    },

    methods: {
      submitLogin(){
            loginApi.submitLogin(this.user).then(response => {
              if(response.data.success){

                //把token存在cookie中、也可以放在localStorage中
                cookie.set('guli_token', response.data.data.token, { domain:
'localhost' })
                //登录成功根据token获取用户信息
                loginApi.getLoginInfo().then(response => {

                  this.loginInfo = response.data.data.item
                  //将用户信息记录cookie
                  cookie.set('guli_ucenter', this.loginInfo, { domain: 'localhost'
})
                  //跳转页面
                  window.location.href = "/";
                })
              }
            })
      },

      checkPhone (rule, value, callback) {
        //debugger
        if (!(/^1[34578]\d{9}$/.test(value))) {
          return callback(new Error('手机号码格式不正确'))
        }
        return callback()
      }
    }
  }
</script>
<style>
   .el-form-item__error{
    z-index: 9999999;
```

```
  }
</style>
```

**3**、在**request.js**添加拦截器，用于传递**token**信息

```
import axios from 'axios'
import { MessageBox, Message } from 'element-ui'
import cookie from 'js-cookie'

// 创建axios实例
const service = axios.create({
   //baseURL: 'http://qy.free.idcfengye.com/api', // api 的 base_url
  //baseURL: 'http://localhost:8210', // api 的 base_url
  baseURL: 'http://localhost:9001',
  timeout: 15000 // 请求超时时间

})

// http request 拦截器
service.interceptors.request.use(
  config => {
  //debugger
  if (cookie.get('guli_token')) {
    config.headers['token'] = cookie.get('guli_token');
  }
    return config
  },
  err => {
  return Promise.reject(err);
})
// http response 拦截器
service.interceptors.response.use(
  response => {
    //debugger
    if (response.data.code == 28004) {
        console.log("response.data.resultCode是28004")
        // 返回 错误代码-1 清除ticket信息并跳转到登录页面
        //debugger
        window.location.href="/login"
        return
    }else{
```

```
      if (response.data.code !== 20000) {
        //25000: 订单支付中，不做任何提示
        if(response.data.code != 25000) {
          Message({
            message: response.data.message || 'error',
            type: 'error',
            duration: 5 * 1000
          })
        }
      } else {
        return response;
      }
    }
  },
  error => {
    return Promise.reject(error.response)   // 返回接口返回的错误信息
});

export default service
```

4、修改**layouts**中的**default.vue**页面

（**1**）显示登录之后的用户信息

```
<script>
import "~/assets/css/reset.css";
import "~/assets/css/theme.css";
import "~/assets/css/global.css";
import "~/assets/css/web.css";

import cookie from 'js-cookie'
import userApi from '@/api/login'

export default {
  data() {
    return {
      token: '',
      loginInfo: {
        id: '',
        age: '',
        avatar: '',
        mobile: '',
```

```
        nickname: '',
        sex: ''
      }
    }
  },

  created() {
    this.showInfo()
  },

  methods: {
    showInfo() {
      //debugger
      var jsonStr = cookie.get("guli_ucenter");
      //alert(jsonStr)
      if (jsonStr) {
        this.loginInfo = JSON.parse(jsonStr)
      }
    },

    logout() {
      //debugger
      cookie.set('guli_ucenter', "", {domain: 'localhost'})
      cookie.set('guli_token', "", {domain: 'localhost'})

      //跳转页面
      window.location.href = "/"
    }
  }
}
</script>
```

（2）**default.vue**页面显示登录之后的用户信息

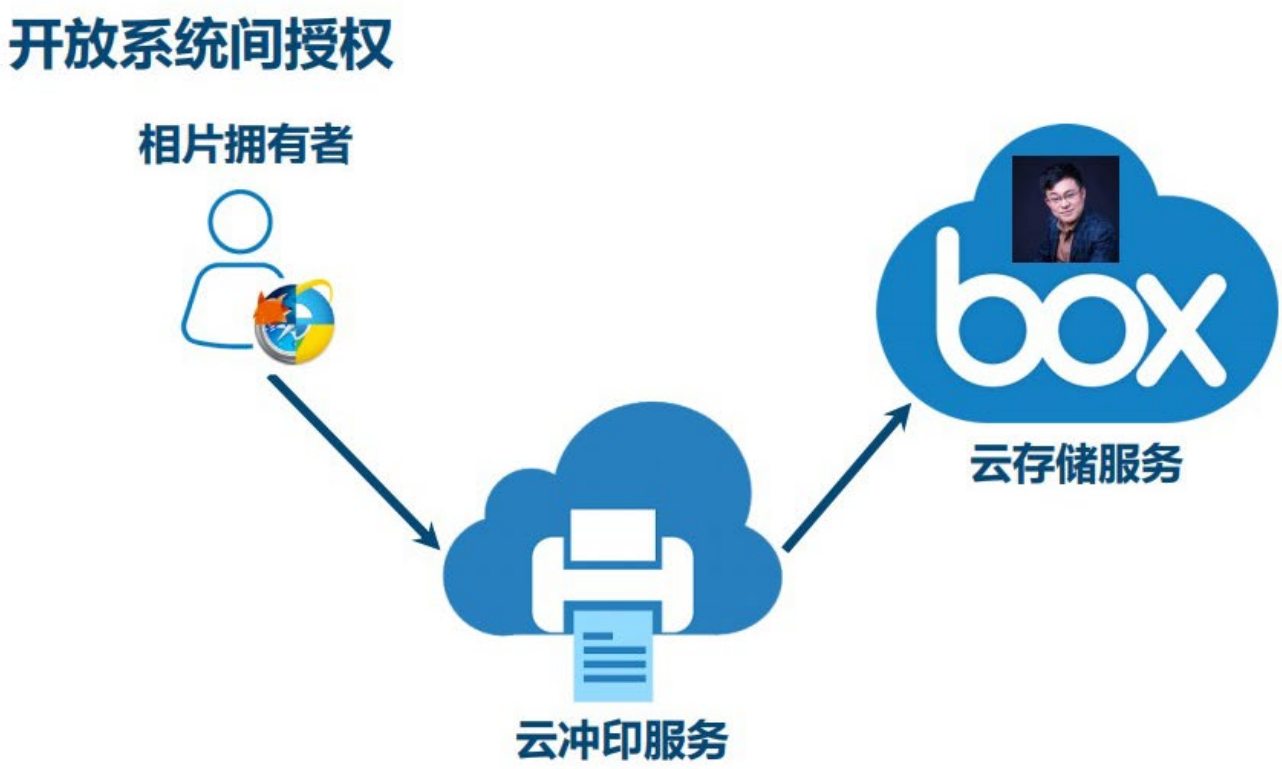
```
 <!-- / nav -->
<ul class="h-r-login">
    <li v-if="!loginInfo.id" id="no-login">
        <a href="/login" title="登录">
            <em class="icon18 login-icon"> </em>
            <span class="vam ml5">登录</span>
        </a>
```

```
        |
        <a href="/register" title="注册">
            <span class="vam ml5">注册</span>
        </a>
    </li>
    <li v-if="loginInfo.id" id="is-login-one" class="mr10">
        <a id="headerMsgCountId" href="#" title="消息">
            <em class="icon18 news-icon"> </em>
        </a>
        <q class="red-point" style="display: none"> </q>
    </li>
    <li v-if="loginInfo.id" id="is-login-two" class="h-r-user">
        <a href="/ucenter" title>
            <img
                 :src="loginInfo.avatar"
                 width="30"
                 height="30"
                 class="vam picImg"
                 alt
                 >
            <span id="userName" class="vam disIb">{{ loginInfo.nickname }}</span>
        </a>
        <a href="javascript:void(0);" title="退出" @click="logout()" class="ml5">退出</a>
    </li>
    <!-- /未登录显示第1 li; 登录后显示第2, 3 li -->
</ul>
```

# 一、OAuth2解决什么问题

## 1、OAuth2提出的背景

照片拥有者想要在云冲印服务上打印照片，云冲印服务需要访问云存储服务上的资源




## 2、图例

资源拥有者：照片拥有者

客户应用：云冲印

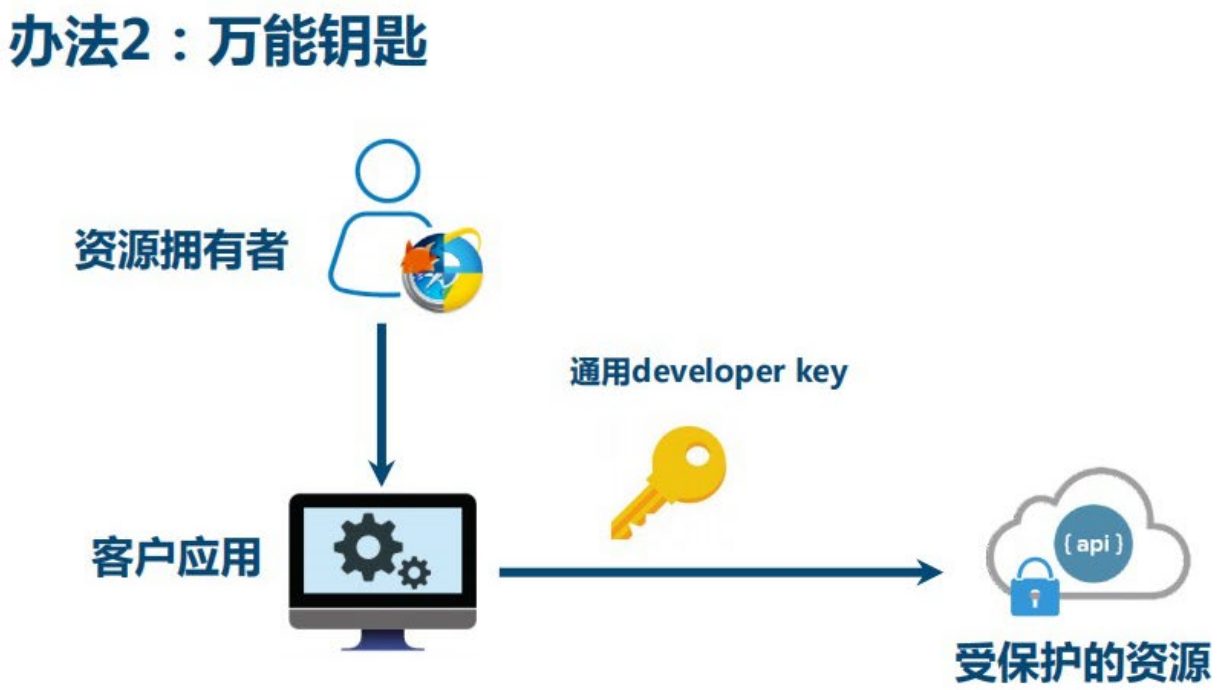
受保护的资源：照片


## 3、方式一：用户名密码复制

适用于同一公司内部的多个系统，不适用于不受信的第三方应用

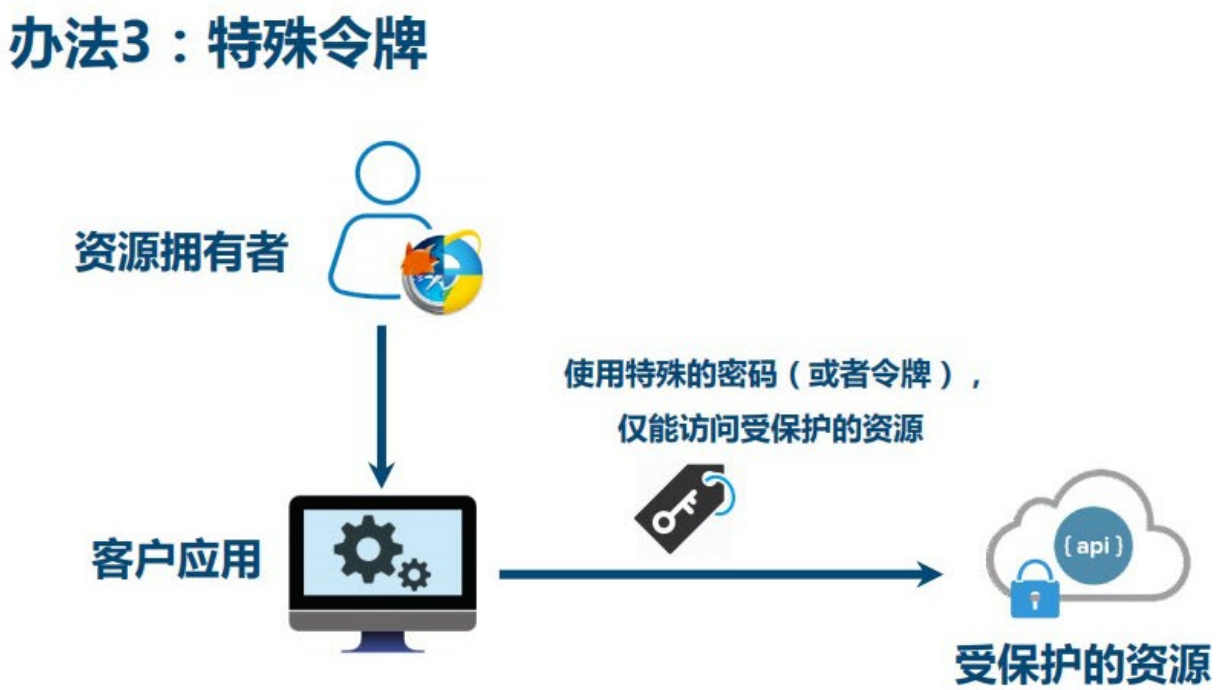

## 4、方式二：通用开发者key

适用于合作商或者授信的不同业务部门之间




## 5、方式三：办法令牌

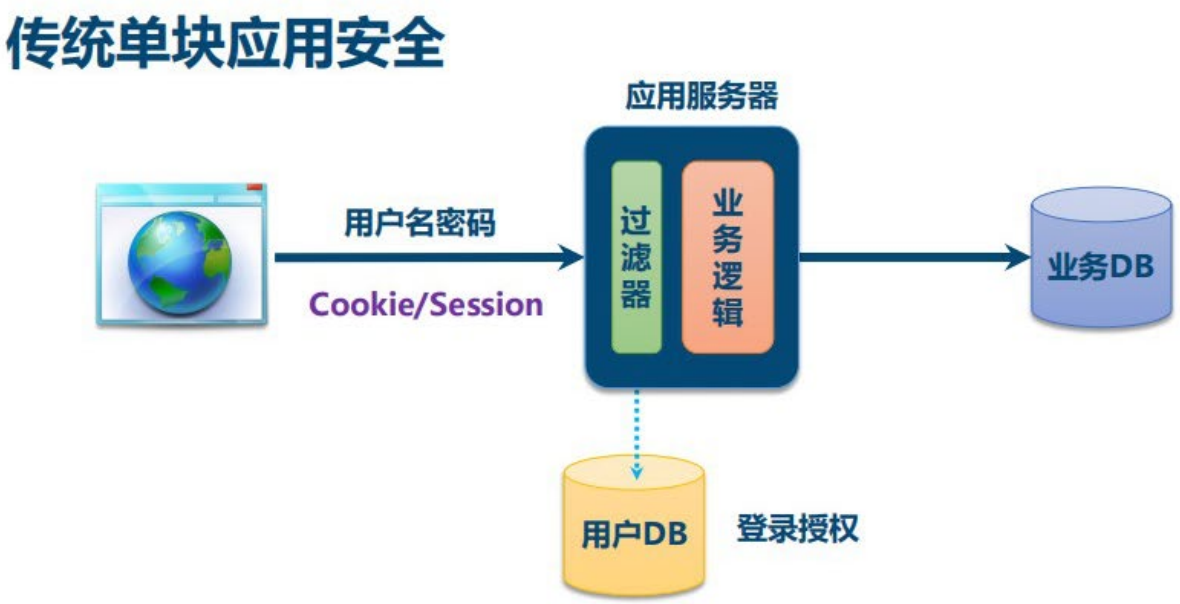
接近OAuth2方式，需要考虑如何管理令牌、颁发令牌、吊销令牌，需要统一的协议，因此就有了OAuth2协议



# 二、现代微服务安全

除了开放系统授权，OAuth2还可以应用于现代微服务安全

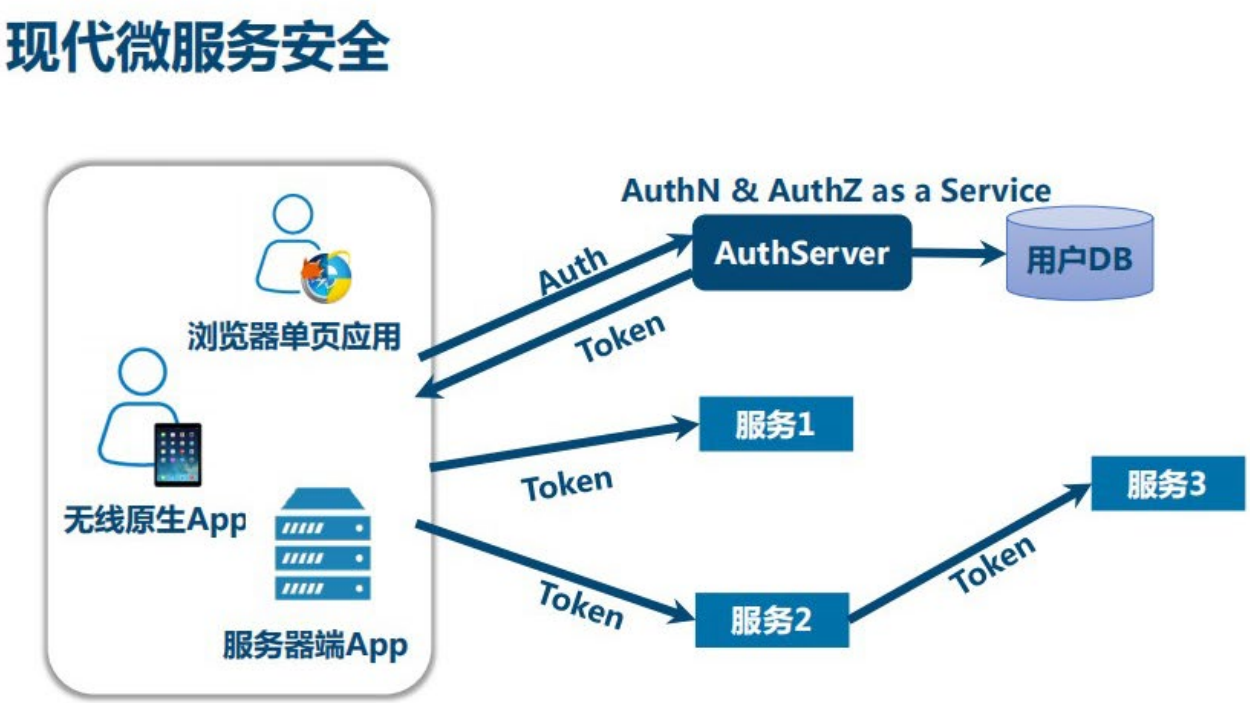
## 1、传统单块应用的安全




## 2、现代微服务安全

现代微服务中系统微服务化以及应用的形态和设备类型增多，不能用传统的登录方式

核心的技术不是用户名和密码，而是token，由AuthServer颁发token，用户使用token进行登录



## 3、典型的OAuth2应用


## 三、总结


## 四、OAuth2最简向导

川崎高彦：OAuth2领域专家，开发了一个OAuth2 sass服务，OAuth2 as Service，并且做成了一个公司再融资的过程中为了向投资人解释OAuth2是什么，于是写了一篇文章，《OAuth2最简向导》

# 一、什么是OAuth2

## 1、OAuth2正式定义


## 2、令牌的核心

## 令牌类比仆从钥匙(Valet Key)

给应用授予有限的访问权限，让应用能够代表用户去访问用户的数据


## 3、OAuth2的历史

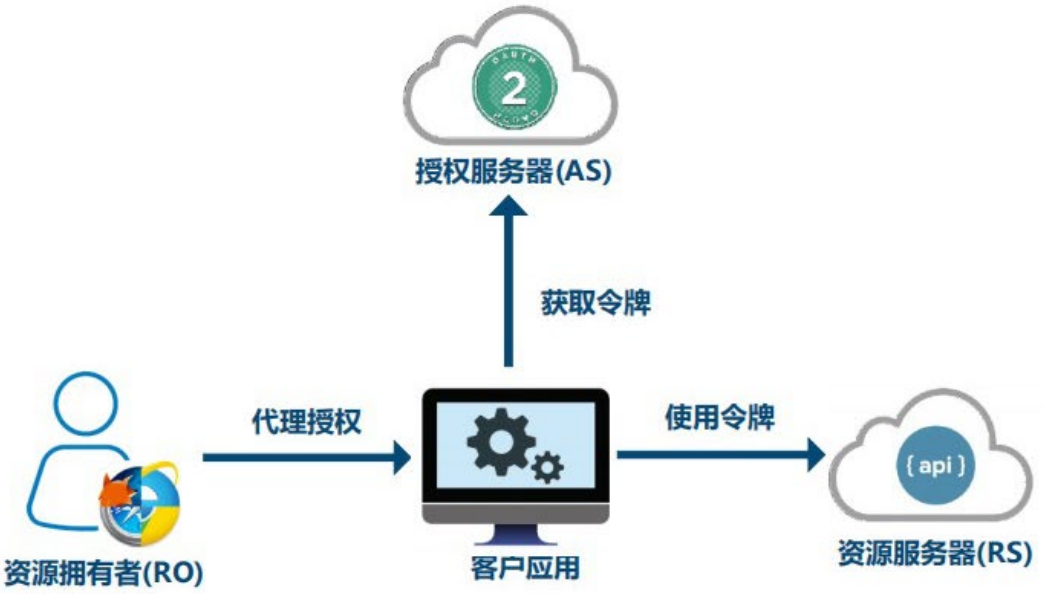
### OAuth 2.0历史


## 4、OAuth2的优势


## 5、OAuth2的不足


## 6、Auth2涉及的角色

## OAuth 2.0主要角色




# 7、OAuth2术语

## OAuth术语

| 客户应用 (Client Application) | 资源服务器 (Resource Server) | 授权服务器 (Authorized Server) | 资源拥有者 (Resource Owner) |
| --- | --- | --- | --- |
| 通常是一个Web或者无线应用，它需要访问用户的受保护资源 | 是一个web站点或者web service API，用户的受保护数据保存于此 | 在客户应用成功认证并获得授权之后，向客户应用颁发访问令牌Access Token | 资源的拥有人，想要分享某些资源给第三方应用 |


## 8、OAuth2令牌的类型


## 9、OAuth2的误解

## OAuth 2.0误解


# 二、回顾

## 回顾

**OAuth是一种在系统之间的代理授权(delegation authorization)协议**

**OAuth本质**
如何获取token
如何使用token

**OAuth使用代理协议的方式解决密码共享反模式问题**

**OAuth提供一个宽泛的协议框架，具体安全场景需要定制**

# 一、准备工作

https://open.weixin.qq.com

## 1、注册

## 2、邮箱激活

## 3、完善开发者资料

## 4、开发者资质认证

准备营业执照，1-2个工作日审批、300元

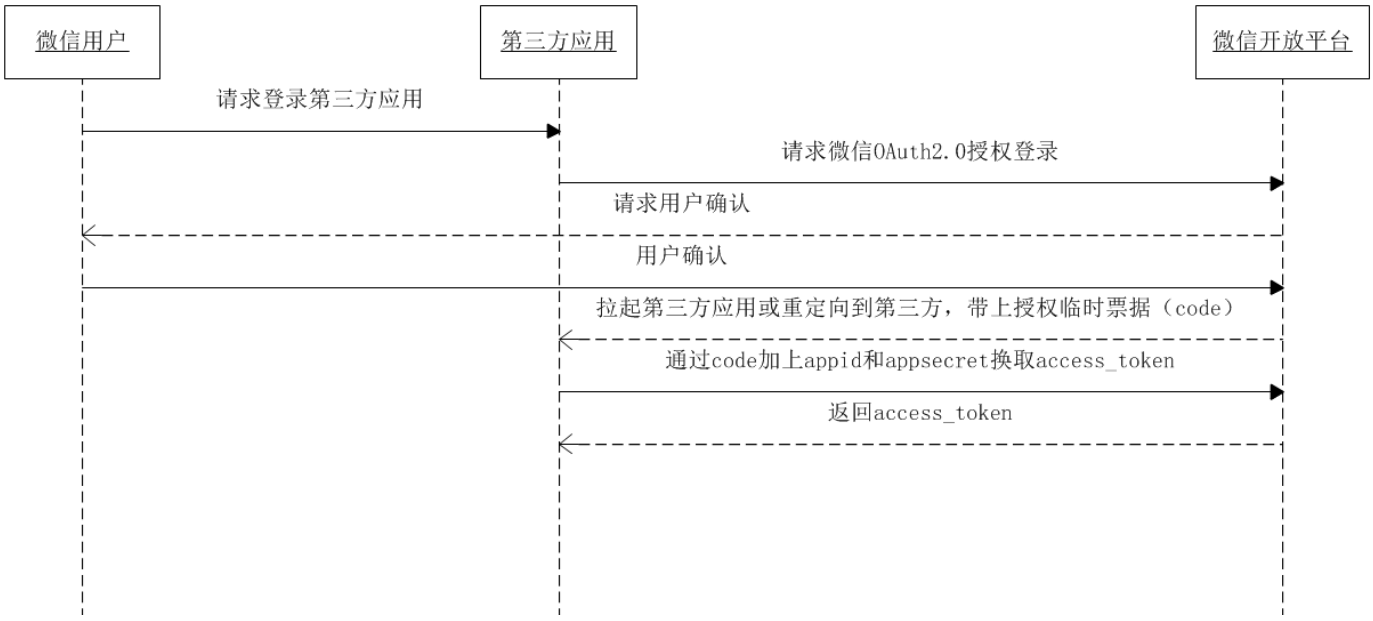
## 5、创建网站应用

提交审核，7个工作日审批

## 6、熟悉微信登录流程

参考文档：https://open.weixin.qq.com/cgi-bin/showdocument?action=dir_list&t=resource/res_list&verify=1&id=open1419316505&token=e547653f995d8f402704d5cb2945177dc8aa4e7e&lang=zh_CN

**获取access_token时序图**




# 二、后端开发

## 1、添加配置

application.properties添加相关配置信息

```
# 微信开放平台 appid
wx.open.app_id=你的appid
# 微信开放平台 appsecret
wx.open.app_secret=你的appsecret
# 微信开放平台 重定向url
wx.open.redirect_url=http://你的服务器名称/api/ucenter/wx/callback
```

## 2、创建常量类

创建util包，创建ConstantPropertiesUtil.java常量类

```
package com.guli.ucenter.util;

@Component
//@PropertySource("classpath:application.properties")
public class ConstantPropertiesUtil implements InitializingBean {

    @Value("${wx.open.app_id}")
    private String appId;

    @Value("${wx.open.app_secret}")
    private String appSecret;

    @Value("${wx.open.redirect_url}")
    private String redirectUrl;

    public static String WX_OPEN_APP_ID;
    public static String WX_OPEN_APP_SECRET;
    public static String WX_OPEN_REDIRECT_URL;

    @Override
    public void afterPropertiesSet() throws Exception {
        WX_OPEN_APP_ID = appId;
        WX_OPEN_APP_SECRET = appSecret;
        WX_OPEN_REDIRECT_URL = redirectUrl;
    }
}
```

## 3、创建controller

guli-microservice-ucenter微服务中创建api包

api包中创建WxApiController

```
package com.guli.ucenter.controller.api;
```

```
2
3  @CrossOrigin
4  @Controller//注意这里没有配置 @RestController
5  @RequestMapping("/api/ucenter/wx")
6  public class WxApiController {
7
8      @GetMapping("login")
9      public String genQrConnect(HttpSession session) {
10
11         // 微信开放平台授权baseUrl
12         String baseUrl = "https://open.weixin.qq.com/connect/qrconnect" +
13                 "?appid=%s" +
14                 "&redirect_uri=%s" +
15                 "&response_type=code" +
16                 "&scope=snsapi_login" +
17                 "&state=%s" +
18                 "#wechat_redirect";
19
20         // 回调地址
21         String redirectUrl = ConstantPropertiesUtil.WX_OPEN_REDIRECT_URL; //获取业务服务器重定向地址
22         try {
23             redirectUrl = URLEncoder.encode(redirectUrl, "UTF-8"); //url编码
24         } catch (UnsupportedEncodingException e) {
25             throw new GuliException(20001, e.getMessage());
26         }
27
28         // 防止csrf攻击（跨站请求伪造攻击）
29         //String state = UUID.randomUUID().toString().replaceAll("-", "");//一般情况下会使用一个随机数
30         String state = "imhelen";//为了让大家能够使用我搭建的外网的微信回调跳转服务器，这里填写你在ngrok的前置
   域名
31         System.out.println("state = " + state);
32
33         // 采用redis等进行缓存state 使用sessionId为key 30分钟后过期，可配置
34         //键："wechar-open-state-" + httpServletRequest.getSession().getId()
35         //值：satte
36         //过期时间：30分钟
37
38         //生成qrcodeUrl
39         String qrcodeUrl = String.format(
40                 baseUrl,
41                 ConstantPropertiesUtil.WX_OPEN_APP_ID,
42                 redirectUrl,
43                 state);
44
45         return "redirect:" + qrcodeUrl;
46     }
47 }
```

授权url参数说明

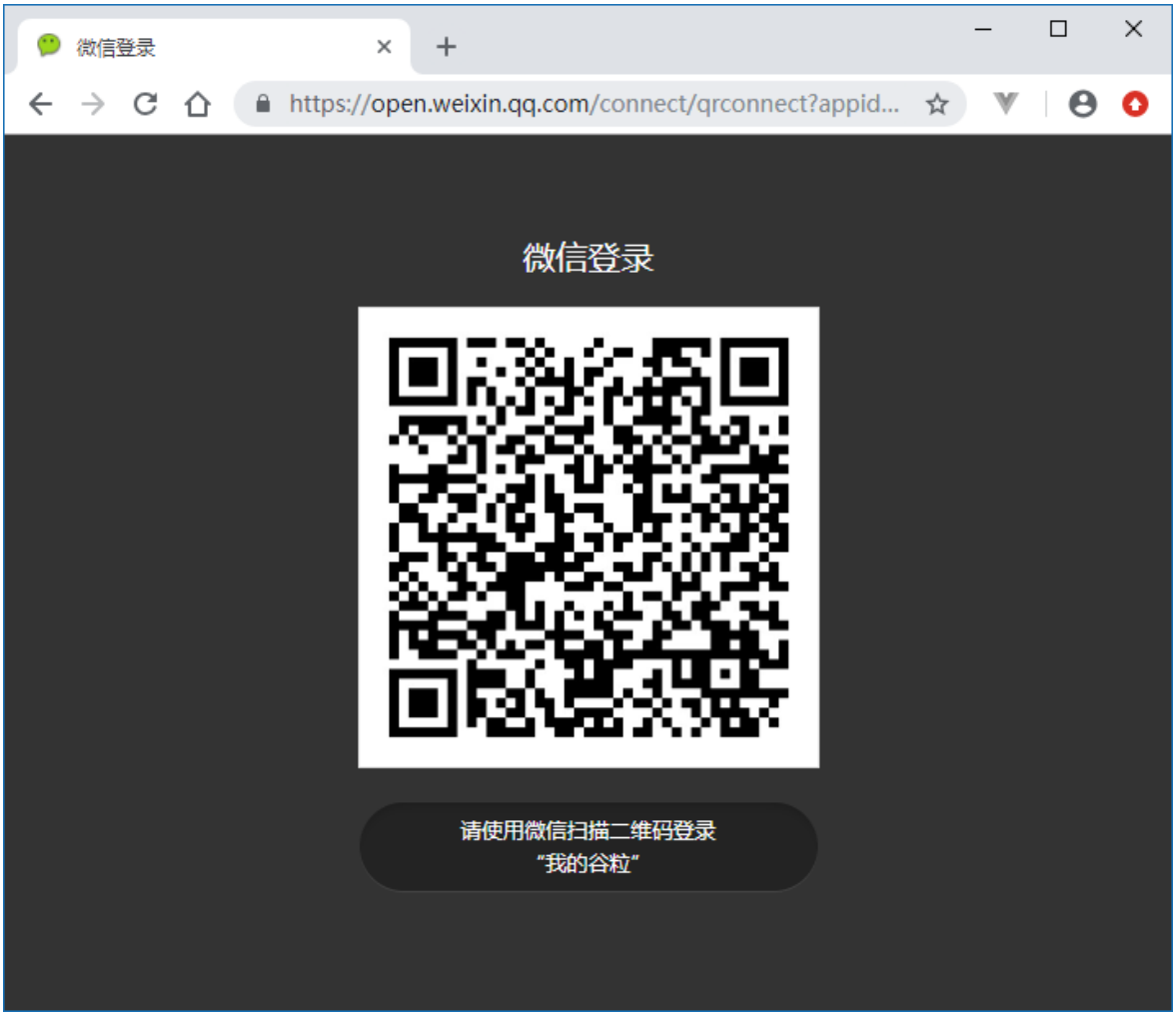
| 参数 | 是否必须 | 说明 |
| --- | --- | --- |
| appid | 是 | 应用唯一标识 |

| | | |
|---|---|---|
| redirect_uri | 是 | 请使用urlEncode对链接进行处理 |
| response_type | 是 | 填code |
| scope | 是 | 应用授权作用域，拥有多个作用域用逗号（,）分隔，网页应用目前仅填写snsapi_login即 |
| state | 否 | 用于保持请求和回调的状态，授权请求后原样带回给第三方。该参数可用于防止csrf攻击（跨站请求伪造攻击），建议第三方带上该参数，可设置为简单的随机数加session进行校验 |

## 4、测试

访问：http://localhost:8201/api/ucenter/wx/login

访问授权url后会得到一个微信登录二维码




用户扫描二维码会看到确认登录的页面


用户点击“确认登录”后，微信服务器会向谷粒学院的业务服务器发起回调，因此接下来我们需要开发回调controller

**注意：**如果没有正确的配置业务服务器的回调url，则会看到以下错误提示

# 一、准备工作

## 1、全局配置的跳转路径

```
# 微信开放平台 重定向url
wx.open.redirect_url=http://回调地址/api/ucenter/wx/callback
```

## 2、修改当前项目启动端口号为8150

## 3、测试回调是否可用

在WxApiController中添加方法

```
@GetMapping("callback")
public String callback(String code, String state, HttpSession session) {

    //得到授权临时票据code
    System.out.println("code = " + code);
    System.out.println("state = " + state);
}
```

# 二、后台开发

## 1、添加依赖

```
 <!--httpclient-->
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
</dependency>
<!--commons-io-->
```

```
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
</dependency>
<!--gson-->
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
</dependency>
```

## 2、创建httpclient工具类

放入util包

```
HttpClientUtils.java
```

## 3、创建回调controller方法

在WxApiController.java中添加如下方法

```
/**
* @param code
* @param state
* @return
*/
@GetMapping("callback")
public String callback(String code, String state){

    //得到授权临时票据code
    System.out.println(code);
    System.out.println(state);

    //从redis中将state获取出来，和当前传入的state作比较
    //如果一致则放行，如果不一致则抛出异常：非法访问

    //向认证服务器发送请求换取access_token
    String baseAccessTokenUrl =
"https://api.weixin.qq.com/sns/oauth2/access_token" +
```

```
        "?appid=%s" +
        "&secret=%s" +
        "&code=%s" +
        "&grant_type=authorization_code";

    String accessTokenUrl = String.format(baseAccessTokenUrl,
                                          ConstantPropertiesUtil.WX_OPEN_APP_ID,
ConstantPropertiesUtil.WX_OPEN_APP_SECRET,
                                          code);

    String result = null;
    try {
        result = HttpClientUtils.get(accessTokenUrl);
        System.out.println("accessToken=============" + result);
    } catch (Exception e) {
        throw new GuliException(20001, "获取access_token失败");
    }

    //解析json字符串
    Gson gson = new Gson();
    HashMap map = gson.fromJson(result, HashMap.class);
    String accessToken = (String)map.get("access_token");
    String openid = (String)map.get("openid");

    //查询数据库当前用用户是否曾经使用过微信登录
    Member member = memberService.getByOpenid(openid);
    if(member == null){
        System.out.println("新用户注册");

        //访问微信的资源服务器，获取用户信息
        String baseUserInfoUrl = "https://api.weixin.qq.com/sns/userinfo" +
            "?access_token=%s" +
            "&openid=%s";
        String userInfoUrl = String.format(baseUserInfoUrl, accessToken, openid);
        String resultUserInfo = null;
        try {
            resultUserInfo = HttpClientUtils.get(userInfoUrl);
            System.out.println("resultUserInfo==========" + resultUserInfo);
        } catch (Exception e) {
            throw new GuliException(20001, "获取用户信息失败");
        }

```

```
        //解析json
        HashMap<String, Object> mapUserInfo = gson.fromJson(resultUserInfo,
HashMap.class);
        String nickname = (String)mapUserInfo.get("nickname");
        String headimgurl = (String)mapUserInfo.get("headimgurl");

        //向数据库中插入一条记录
        member = new Member();
        member.setNickname(nickname);
        member.setOpenid(openid);
        member.setAvatar(headimgurl);
        memberService.save(member);
    }

    //TODO 登录

    return "redirect:http://localhost:3000";
}
```

# 4、业务层

业务接口：MemberService.java

```
Member getByOpenid(String openid);
```

业务实现：MemberServiceImpl.java

```
@Override
public Member getByOpenid(String openid) {

    QueryWrapper<Member> queryWrapper = new QueryWrapper<>();
    queryWrapper.eq("openid", openid);

    Member member = baseMapper.selectOne(queryWrapper);
    return member;
}
```

# 一、整合JWT令牌

## 1、callback中生成jwt

在WxApiController.java的callback方法的最后添加如下代码

```
// 生成jwt
String token = JwtUtils.geneJsonWebToken(member.getId(),member.getNickName());

//存入cookie
//CookieUtils.setCookie(request, response, "guli_jwt_token", token);

//因为端口号不同存在蛞蝓问题，cookie不能跨域，所以这里使用url重写
return "redirect:http://localhost:3000?token=" + token;
```

## 2、前端打印token

在layout/defaullt.vue中打印获取的token值

```
export default {

  created() {
    console.log(this.$route.query.token)
  }
}
```

# 1、修改default.vue页面脚本

```
export default {
  data() {
    return {
      token: '',
      loginInfo: {
        id: '',
        age: '',
        avatar: '',
        mobile: '',
        nickname: '',
        sex: ''
      }
    }
  },

  created() {
    this.token = this.$route.query.token
    if (this.token) {
      this.wxLogin()
    }

    this.showInfo()
  },

  methods: {
    showInfo() {
      //debugger
      var jsonStr = cookie.get("guli_ucenter");
      if (jsonStr) {
        this.loginInfo = JSON.parse(jsonStr)
      }
    },

    logout() {
      //debugger
      cookie.set('guli_ucenter', "", {domain: 'localhost'})
      cookie.set('guli_token', "", {domain: 'localhost'})

```

```
      //跳转页面
      window.location.href = "/"
    },

    wxLogin() {
      if (this.token == '') return
      //把token存在cookie中、也可以放在localStorage中
      cookie.set('guli_token', this.token, {domain: 'localhost'})
      cookie.set('guli_ucenter', '', {domain: 'localhost'})
      //登录成功根据token获取用户信息
      userApi.getLoginInfo().then(response => {
        this.loginInfo = response.data.data.item
        //将用户信息记录cookie
        cookie.set('guli_ucenter', this.loginInfo, {domain: 'localhost'})
      })
    }
  }
}
```

# 一、列表页面

创建 pages/teacher/index.vue

```
<template>
  <div id="aCoursesList" class="bg-fa of">
    <!-- 讲师列表 开始 -->
    <section class="container">
      <header class="comm-title all-teacher-title">
        <h2 class="fl tac">
          <span class="c-333">全部讲师</span>
        </h2>
        <section class="c-tab-title">
          <a id="subjectAll" title="全部" href="#">全部</a>
          <!-- <c:forEach var="subject" items="${subjectList }">
                          <a id="${subject.subjectId}"
title="${subject.subjectName }" href="javascript:void(0)"
onclick="submitForm(${subject.subjectId})">${subject.subjectName }</a>
          </c:forEach>-->
        </section>
      </header>
      <section class="c-sort-box unBr">
        <div>
          <!-- /无数据提示 开始-->
          <section class="no-data-wrap">
            <em class="icon30 no-data-ico"> </em>
            <span class="c-666 fsize14 ml10 vam">没有相关数据，小编正在努力整理
中...</span>
          </section>
          <!-- /无数据提示 结束-->
          <article class="i-teacher-list">
            <ul class="of">
              <li>
                <section class="i-teach-wrap">
                  <div class="i-teach-pic">
                    <a href="/teacher/1" title="姚晨" target="_blank">
                      <img src="~/assets/photo/teacher/1442297885942.jpg" alt>
                    </a>
                  </div>
                  <div class="mt10 hLh30 txtOf tac">
                    <a href="/teacher/1" title="姚晨" target="_blank"
class="fsize18 c-666">姚晨</a>
```

```
                </div>
                <div class="hLh30 txtOf tac">
                  <span class="fsize14 c-999">北京师范大学法学院副教授、清华大学法学博士。自2004年至今已有9年的司法考试培训经验。长期从事司法考试辅导，深知命题规律，了解解题技巧。内容把握准确，授课重点明确，层次分明，调理清晰，将法条法理与案例有机融合，强调综合，深入浅出。</span>
                </div>
                <div class="mt15 i-q-txt">
                  <p class="c-999 f-fA">北京师范大学法学院副教授</p>
                </div>
              </section>
            </li>
            <li>
              <section class="i-teach-wrap">
                <div class="i-teach-pic">
                  <a href="/teacher/1" title="谢娜" target="_blank">
                    <img src="~/assets/photo/teacher/1442297919077.jpg" alt>
                  </a>
                </div>
                <div class="mt10 hLh30 txtOf tac">
                  <a href="/teacher/1" title="谢娜" target="_blank" class="fsize18 c-666">谢娜</a>
                </div>
                <div class="hLh30 txtOf tac">
                  <span class="fsize14 c-999">十年课程研发和培训咨询经验，曾任国企人力资源经理、大型外企培训经理，负责企业大学和培训体系搭建；曾任专业培训机构高级顾问、研发部总监，为包括广东移动、东莞移动、深圳移动、南方电网、工商银行、农业银行、民生银行、邮储银行、TCL集团、清华大学继续教育学院、中天路桥、广西扬翔股份等超过200家企业提供过培训与咨询服务，并担任近50个大型项目的总负责人。</span>
                </div>
                <div class="mt15 i-q-txt">
                  <p class="c-999 f-fA">资深课程设计专家，专注10年AACTP美国培训协会认证导师</p>
                </div>
              </section>
            </li>
            <li>
              <section class="i-teach-wrap">
                <div class="i-teach-pic">
                  <a href="/teacher/1" title="刘德华" target="_blank">
                    <img src="~/assets/photo/teacher/1442297927029.jpg" alt>
                  </a>
                </div>
                <div class="mt10 hLh30 txtOf tac">
                  <a href="/teacher/1" title="刘德华" target="_blank"
```

```
class="fsize18 c-666">刘德华</a>
                  </div>
                  <div class="hLh30 txtOf tac">
                    <span class="fsize14 c-999">上海师范大学法学院副教授、清华大学法学博士。自2004年至今已有9年的司法考试培训经验。长期从事司法考试辅导，深知命题规律，了解解题技巧。内容把握准确，授课重点明确，层次分明，调理清晰，将法条法理与案例有机融合，强调综合，深入浅出。</span>
                  </div>
                  <div class="mt15 i-q-txt">
                    <p class="c-999 f-fA">上海师范大学法学院副教授</p>
                  </div>
                </section>
              </li>
              <li>
                <section class="i-teach-wrap">
                  <div class="i-teach-pic">
                    <a href="/teacher/1" title="周润发" target="_blank">
                      <img src="~/assets/photo/teacher/1442297935589.jpg" alt>
                    </a>
                  </div>
                  <div class="mt10 hLh30 txtOf tac">
                    <a href="/teacher/1" title="周润发" target="_blank"
class="fsize18 c-666">周润发</a>
                  </div>
                  <div class="hLh30 txtOf tac">
                    <span class="fsize14 c-999">法学博士，北京师范大学马克思主义学院副教授，专攻毛泽东思想概论、邓小平理论，长期从事考研辅导。出版著作两部，发表学术论文30余篇，主持国家社会科学基金项目和教育部重大课题子课题各一项，参与中央实施马克思主义理论研究和建设工程项目。</span>
                  </div>
                  <div class="mt15 i-q-txt">
                    <p class="c-999 f-fA">考研政治辅导实战派专家，全国考研政治命题研究组核心成员。</p>
                  </div>
                </section>
              </li>
              <li>
                <section class="i-teach-wrap">
                  <div class="i-teach-pic">
                    <a href="/teacher/1" title="钟汉良" target="_blank">
                      <img src="~/assets/photo/teacher/1442298121626.jpg" alt>
                    </a>
                  </div>
                  <div class="mt10 hLh30 txtOf tac">
                    <a href="/teacher/1" title="钟汉良" target="_blank"
```

```
class="fsize18 c-666">钟汉良</a>
                  </div>
                  <div class="hLh30 txtOf tac">
                    <span class="fsize14 c-999">具备深厚的数学思维功底、丰富的小学教育经验，授课风格生动活泼，擅长用形象生动的比喻帮助理解、简单易懂的语言讲解难题，深受学生喜欢</span>
                  </div>
                  <div class="mt15 i-q-txt">
                    <p class="c-999 f-fA">毕业于师范大学数学系，热爱教育事业，执教数学思维6年有余</p>
                  </div>
                </section>
              </li>
              <li>
                <section class="i-teach-wrap">
                  <div class="i-teach-pic">
                    <a href="/teacher/1" title="唐嫣" target="_blank">
                      <img src="~/assets/photo/teacher/1442297957332.jpg" alt>
                    </a>
                  </div>
                  <div class="mt10 hLh30 txtOf tac">
                    <a href="/teacher/1" title="唐嫣" target="_blank"
class="fsize18 c-666">唐嫣</a>
                  </div>
                  <div class="hLh30 txtOf tac">
                    <span class="fsize14 c-999">中国科学院数学与系统科学研究院应用数学专业博士，研究方向为数字图像处理，中国工业与应用数学学会会员。参与全国教育科学“十五”规划重点课题“信息化进程中的教育技术发展研究”的子课题“基与课程改革的资源开发与应用”，以及全国“十五”科研规划全国重点项目“掌上型信息技术产品在教学中的运用和开发研究”的子课题“用技术学数学”。</span>
                  </div>
                  <div class="mt15 i-q-txt">
                    <p class="c-999 f-fA">中国人民大学附属中学数学一级教师</p>
                  </div>
                </section>
              </li>
              <li>
                <section class="i-teach-wrap">
                  <div class="i-teach-pic">
                    <a href="/teacher/1" title="周杰伦" target="_blank">
                      <img src="~/assets/photo/teacher/1442297969808.jpg" alt>
                    </a>
                  </div>
                  <div class="mt10 hLh30 txtOf tac">
                    <a href="/teacher/1" title="周杰伦" target="_blank"
```

```
class="fsize18 c-666">周杰伦</a>
                </div>
                <div class="hLh30 txtOf tac">
                  <span class="fsize14 c-999">中教一级职称。讲课极具亲和
力。</span>
                </div>
                <div class="mt15 i-q-txt">
                  <p class="c-999 f-fA">毕业于北京大学数学系</p>
                </div>
              </section>
            </li>
            <li>
              <section class="i-teach-wrap">
                <div class="i-teach-pic">
                  <a href="/teacher/1" title="陈伟霆" target="_blank">
                    <img src="~/assets/photo/teacher/1442297977255.jpg" alt>
                  </a>
                </div>
                <div class="mt10 hLh30 txtOf tac">
                  <a href="/teacher/1" title="陈伟霆" target="_blank"
class="fsize18 c-666">陈伟霆</a>
                </div>
                <div class="hLh30 txtOf tac">
                  <span
                    class="fsize14 c-999"
                  >政治学博士、管理学博士后，北京师范大学马克思主义学院副教授。多
年来总结出了一套行之有效的应试技巧与答题方法，针对性和实用性极强，能帮助考生在轻松中应
考，在激励的竞争中取得高分，脱颖而出。</span>
                </div>
                <div class="mt15 i-q-txt">
                  <p class="c-999 f-fA">长期从事考研政治课讲授和考研命题趋势与
应试对策研究。考研辅导新锐派的代表。</p>
                </div>
              </section>
            </li>
          </ul>
          <div class="clear"></div>
        </article>
      </div>
      <!-- 公共分页 开始 -->
      <div>
        <div class="paging">
          <!-- undisable这个class是否存在，取决于数据属性hasPrevious -->
          <a href="#" title="首页">首</a>
          <a href="#" title="前一页">&lt;</a>
```

```
            <a href="#" title="第1页" class="current undisable">1</a>
            <a href="#" title="第2页">2</a>
            <a href="#" title="后一页">&gt;</a>
            <a href="#" title="末页">末</a>
            <div class="clear"></div>
          </div>
        </div>
        <!-- 公共分页 结束 -->
      </section>
    </section>
    <!-- /讲师列表 结束 -->
  </div>
</template>
<script>
export default {};
</script>
```

# 二、详情页面

创建 pages/teacher/_id.vue

修改资源路径为~/assets/

```
<template>
  <div id="aCoursesList" class="bg-fa of">
    <!-- 讲师介绍 开始 -->
    <section class="container">
      <header class="comm-title">
        <h2 class="fl tac">
          <span class="c-333">讲师介绍</span>
        </h2>
      </header>
      <div class="t-infor-wrap">
        <!-- 讲师基本信息 -->
        <section class="fl t-infor-box c-desc-content">
          <div class="mt20 ml20">
            <section class="t-infor-pic">
              <img src="~/assets/photo/teacher/1442297885942.jpg">
            </section>
            <h3 class="hLh30">
              <span class="fsize24 c-333">姚晨 高级讲师</span>
            </h3>
            <section class="mt10">
```

```
              <span class="t-tag-bg">北京师范大学法学院副教授</span>
            </section>
            <section class="t-infor-txt">
              <p
                class="mt20"
              >北京师范大学法学院副教授、清华大学法学博士。自2004年至今已有9年的司法考试培训经验。长期从事司法考试辅导，深知命题规律，了解解题技巧。内容把握准确，授课重点明确，层次分明，调理清晰，将法条法理与案例有机融合，强调综合，深入浅出。</p>
            </section>
            <div class="clear"></div>
          </div>
        </section>
        <div class="clear"></div>
      </div>
      <section class="mt30">
        <div>
          <header class="comm-title all-teacher-title c-course-content">
            <h2 class="fl tac">
              <span class="c-333">主讲课程</span>
            </h2>
            <section class="c-tab-title">
              <a href="javascript: void(0)"> </a>
            </section>
          </header>
          <!-- /无数据提示 开始-->
          <section class="no-data-wrap">
            <em class="icon30 no-data-ico"> </em>
            <span class="c-666 fsize14 ml10 vam">没有相关数据，小编正在努力整理中...</span>
          </section>
          <!-- /无数据提示 结束-->
          <article class="comm-course-list">
            <ul class="of">
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442295455437.jpg" class="img-responsive" >
                    <div class="cc-mask">
                      <a href="#" title="开始学习" target="_blank" class="comm-btn c-btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="#" title="零基础入门学习Python课程学习"
```

```
target="_blank" class="course-title fsize18 c-333">零基础入门学习Python课程学习</a>
                  </h3>
                </div>
              </li>
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442295472860.jpg" class="img-responsive" >
                    <div class="cc-mask">
                      <a href="#" title="开始学习" target="_blank" class="comm-btn c-btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="#" title="影想力摄影小课堂" target="_blank" class="course-title fsize18 c-333">影想力摄影小课堂</a>
                  </h3>
                </div>
              </li>
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442302831779.jpg" class="img-responsive" >
                    <div class="cc-mask">
                      <a href="#" title="开始学习" target="_blank" class="comm-btn c-btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="#" title="数学给宝宝带来的兴趣" target="_blank" class="course-title fsize18 c-333">数学给宝宝带来的兴趣</a>
                  </h3>
                </div>
              </li>
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442295506745.jpg" class="img-responsive" >
                    <div class="cc-mask">
                      <a href="#" title="开始学习" target="_blank" class="comm-btn c-btn-1">开始学习</a>
```

```
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="#" title="国家教师资格考试专用" target="_blank"
class="course-title fsize18 c-333">国家教师资格考试专用</a>
                  </h3>
                </div>
              </li>
            </ul>
            <div class="clear"></div>
          </article>
        </div>
      </section>
    </section>
    <!-- /讲师介绍 结束 -->
  </div>
</template>
<script>
export default {};
</script>
```

# 一、后端

## 1、TeacherService

接口

```
public Map<String, Object> pageListWeb(Page<Teacher> pageParam);
```

实现

```
@Override
public Map<String, Object> pageListWeb(Page<Teacher> pageParam) {
    QueryWrapper<Teacher> queryWrapper = new QueryWrapper<>();
    queryWrapper.orderByAsc("sort");
    baseMapper.selectPage(pageParam, queryWrapper);
    List<Teacher> records = pageParam.getRecords();
    long current = pageParam.getCurrent();
    long pages = pageParam.getPages();
    long size = pageParam.getSize();
    long total = pageParam.getTotal();
    boolean hasNext = pageParam.hasNext();
    boolean hasPrevious = pageParam.hasPrevious();
    Map<String, Object> map = new HashMap<String, Object>();
    map.put("items", records);
    map.put("current", current);
    map.put("pages", pages);
    map.put("size", size);
    map.put("total", total);
    map.put("hasNext", hasNext);
    map.put("hasPrevious", hasPrevious);
    return map;
}
```

## 2、TeacherController

```
@ApiOperation(value = "分页讲师列表")
```

```
@GetMapping(value = "{page}/{limit}")
public R pageList(
    @ApiParam(name = "page", value = "当前页码", required = true)
    @PathVariable Long page,
    @ApiParam(name = "limit", value = "每页记录数", required = true)
    @PathVariable Long limit){
    Page<Teacher> pageParam = new Page<Teacher>(page, limit);
    Map<String, Object> map = teacherService.pageListWeb(pageParam);
    return  R.ok().data(map);
}
```

## 3、swagger测试

# 二、前端列表js

## 1、创建api

创建文件夹api，api下创建teacher.js，用于封装讲师模块的请求

```
import request from '@/utils/request'
const api_name = '/edu/teacher'
export default {
  getPageList(page, limit) {
    return request({
      url: `${api_name}/${page}/${limit}`,
      method: 'get'
    })
  }
}
```

## 2、讲师列表组件中调用api

pages/teacher/index.vue

```
<script>
import teacher from "@/api/teacher"
```

```
export default {
  asyncData({ params, error }) {
    return teacher.getPageList(1, 8).then(response => {
      console.log(response.data.data);
      return { data: response.data.data }
    });
  },
};
</script>
```

# 三、页面渲染

## 1、页面模板

```
<template>
  <div id="aCoursesList" class="bg-fa of">
    <!-- 讲师列表 开始 -->
    <section class="container">
      <section class="c-sort-box unBr">
        <div>
          <!-- 无数据提示 开始-->
          <!-- /无数据提示 结束-->
          <!-- 数据列表 开始-->
          <!-- /数据列表 结束-->
        </div>
        <!-- 公共分页 开始 -->
        <!-- /公共分页 结束 -->
      </section>
    </section>
    <!-- /讲师列表 结束 -->
  </div>
</template>
```

## 2、无数据提示

添加：v-if="data.total==0"

```
<!-- /无数据提示 开始-->
```

```
<section class="no-data-wrap" v-if="data.total==0">
    <em class="icon30 no-data-ico"> </em>
    <span class="c-666 fsize14 ml10 vam">没有相关数据，小编正在努力整理中...</span>
</section>
<!-- /无数据提示 结束-->
```

## 3、列表

```
<!-- /数据列表 开始-->
<article v-if="data.total>0" class="i-teacher-list">
    <ul class="of">
        <li v-for="item in data.items" :key="item.id">
            <section class="i-teach-wrap">
                <div class="i-teach-pic">
                    <a :href="'/teacher/'+item.id" :title="item.name">
                        <img :src="item.avatar" :alt="item.name" height="142" hright="142">
                    </a>
                </div>
                <div class="mt10 hLh30 txtOf tac">
                    <a :href="'/teacher/'+item.id" :title="item.name" class="fsize18 c-666">{{ item.name }}</a>
                </div>
                <div class="hLh30 txtOf tac">
                    <span class="fsize14 c-999" >{{ item.career }}</span>
                </div>
                <div class="mt15 i-q-txt">
                    <p class="c-999 f-fA">{{ item.intro }}</p>
                </div>
            </section>
        </li>
    </ul>
    <div class="clear"/>
</article>
<!-- /数据列表 结束-->
```

## 4、测试

# 四、分页

## 1、分页方法

```
methods: {
    gotoPage(page){
        teacher.getPageList(page, 8).then(response => {
            this.data = response.data.data
        })
    }
}
```

## 2、分页页面渲染

```
<!-- 公共分页 开始 -->
<div>
  <div class="paging">
    <!-- undisable这个class是否存在，取决于数据属性hasPrevious -->
    <a
      :class="{undisable: !data.hasPrevious}"
      href="#"
      title="首页"
      @click.prevent="gotoPage(1)">首</a>
    <a
      :class="{undisable: !data.hasPrevious}"
      href="#"
      title="前一页"
      @click.prevent="gotoPage(data.current-1)">&lt;</a>
    <a
      v-for="page in data.pages"
      :key="page"
      :class="{current: data.current == page, undisable: data.current ==
page}"
      :title="'第'+page+'页'"
      href="#"
      @click.prevent="gotoPage(page)">{{ page }}</a>
    <a
      :class="{undisable: !data.hasNext}"
```

```
      href="#"
      title="后一页"
      @click.prevent="gotoPage(data.current+1)">&gt;</a>
    <a
      :class="{undisable: !data.hasNext}"
      href="#"
      title="末页"
      @click.prevent="gotoPage(data.pages)">末</a>
    <div class="clear"/>
  </div>
</div>
<!-- 公共分页 结束 -->
```

## 3、测试

# 一、后端

## 1、CourseService

根据讲师id查询讲师所讲课程列表

接口

```
List<Course> selectByTeacherId(String teacherId);
```

实现

```
/**
     * 根据讲师id查询当前讲师的课程列表
     * @param teacherId
     * @return
     */
@Override
public List<Course> selectByTeacherId(String teacherId) {
    QueryWrapper<Course> queryWrapper = new QueryWrapper<Course>();
    queryWrapper.eq("teacher_id", teacherId);
    //按照最后更新时间倒序排列
    queryWrapper.orderByDesc("gmt_modified");
    List<Course> courses = baseMapper.selectList(queryWrapper);
    return courses;
}
```

## 2、TeacherController

```
@Autowired
private CourseService courseService;
```

修改getById方法：

```

```

```
@ApiOperation(value = "根据ID查询讲师")
@GetMapping(value = "{id}")
public R getById(
        @ApiParam(name = "id", value = "讲师ID", required = true)
        @PathVariable String id){
    //查询讲师信息
    Teacher teacher = teacherService.getById(id);
    //根据讲师id查询这个讲师的课程列表
    List<Course> courseList = courseService.selectByTeacherId(id);
    return R.ok().data("teacher", teacher).data("courseList", courseList);
}
```

## 3、swagger测试

# 二、前端详情js

## 1、teacher api

api/teacher.js

```
getById(teacherId) {
    return request({
        url: `${api_name}/${teacherId}`,
        method: 'get'
    })
}
```

## 2、讲师详情页中调用api

pages/teacher/_id.vue

```
<script>
import teacher from "@/api/teacher"
export default {
  asyncData({ params, error }) {
    return teacher.getById(params.id).then(response => {
```

```
      console.log(response);
      return {
        teacher: response.data.data.teacher,
        courseList: response.data.data.courseList
      }
    })
  }
}
</script>
```

# 三、页面渲染

## 1、讲师基本信息模板

```
<template>
  <div id="aCoursesList" class="bg-fa of">
    <!-- 讲师介绍 开始 -->
    <section class="container">
      <header class="comm-title">
        <h2 class="fl tac">
          <span class="c-333">讲师介绍</span>
        </h2>
      </header>
      <div class="t-infor-wrap">
        <!-- 讲师基本信息 开始 -->
        <!-- /讲师基本信息 结束 -->
        <div class="clear"/>
      </div>
      <section class="mt30">
        <div>
          <header class="comm-title all-teacher-title c-course-content">
            <h2 class="fl tac">
              <span class="c-333">主讲课程</span>
            </h2>
            <section class="c-tab-title">
              <a href="javascript: void(0)"> </a>
            </section>
          </header>
          <!-- 无数据提示 开始-->
          <!-- /无数据提示 结束-->
          <!-- 课程列表 开始-->
```

```
          <!-- /课程列表 结束-->
        </div>
      </section>
    </section>
    <!-- /讲师介绍 结束 -->
  </div>
</template>
```

## 2、讲师详情显示

```
<!-- 讲师基本信息 开始 -->
<section class="fl t-infor-box c-desc-content">
    <div class="mt20 ml20">
        <section class="t-infor-pic">
            <img :src="teacher.avatar" :alt="teacher.name">
        </section>
        <h3 class="hLh30">
            <span class="fsize24 c-333">{{ teacher.name }}
                 
                {{ teacher.level===1?'高级讲师':'首席讲师' }}
            </span>
        </h3>
        <section class="mt10">
            <span class="t-tag-bg">{{ teacher.intro }}</span>
        </section>
        <section class="t-infor-txt">
            <p class="mt20">{{ teacher.career }}</p>
        </section>
        <div class="clear"/>
    </div>
</section>
<!-- /讲师基本信息 结束 -->
```

## 3、无数据提示

```
<!-- 无数据提示 开始-->
<section class="no-data-wrap" v-if="courseList.total==0">
    <em class="icon30 no-data-ico"> </em>
    <span class="c-666 fsize14 ml10 vam">没有相关数据，小编正在努力整理
```

```
中...</span>
</section>
<!-- /无数据提示 结束-->
```

## 4、当前讲师课程列表

```
<!-- 课程列表 开始-->
<article class="comm-course-list">
  <ul class="of">
    <li v-for="course in courseList" :key="course.id">
      <div class="cc-l-wrap">
        <section class="course-img">
          <img :src="course.cover" class="img-responsive">
          <div class="cc-mask">
            <a :href="'/course/'+course.id" title="开始学习" target="_blank"
class="comm-btn c-btn-1">开始学习</a>
          </div>
        </section>
        <h3 class="hLh30 txtOf mt10">
          <a
            :href="'/course/'+course.id"
            :title="course.title"
            target="_blank"
            class="course-title fsize18 c-333">{{ course.title }}</a>
        </h3>
      </div>
    </li>
  </ul>
  <div class="clear"/>
</article>
<!-- /课程列表 结束-->
```

## 5、测试

# 一、列表页面

创建 pages/course/index.vue

```
<template>
  <div id="aCoursesList" class="bg-fa of">
    <!-- /课程列表 开始 -->
    <section class="container">
      <header class="comm-title">
        <h2 class="fl tac">
          <span class="c-333">全部课程</span>
        </h2>
      </header>
      <section class="c-sort-box">
        <section class="c-s-dl">
          <dl>
            <dt>
              <span class="c-999 fsize14">课程类别</span>
            </dt>
            <dd class="c-s-dl-li">
              <ul class="clearfix">
                <li>
                  <a title="全部" href="#">全部</a>
                </li>
                <li>
                  <a title="数据库" href="#">数据库</a>
                </li>
                <li class="current">
                  <a title="外语考试" href="#">外语考试</a>
                </li>
                <li>
                  <a title="教师资格证" href="#">教师资格证</a>
                </li>
                <li>
                  <a title="公务员" href="#">公务员</a>
                </li>
                <li>
                  <a title="移动开发" href="#">移动开发</a>
                </li>
                <li>
                  <a title="操作系统" href="#">操作系统</a>
                </li>
```

```
            </ul>
          </dd>
        </dl>
        <dl>
          <dt>
            <span class="c-999 fsize14"></span>
          </dt>
          <dd class="c-s-dl-li">
            <ul class="clearfix">
              <li>
                <a title="职称英语" href="#">职称英语</a>
              </li>
              <li>
                <a title="英语四级" href="#">英语四级</a>
              </li>
              <li>
                <a title="英语六级" href="#">英语六级</a>
              </li>
            </ul>
          </dd>
        </dl>
        <div class="clear"></div>
      </section>
      <div class="js-wrap">
        <section class="fr">
          <span class="c-ccc">
            <i class="c-master f-fM">1</i>/
            <i class="c-666 f-fM">1</i>
          </span>
        </section>
        <section class="fl">
          <ol class="js-tap clearfix">
            <li>
              <a title="关注度" href="#">关注度</a>
            </li>
            <li>
              <a title="最新" href="#">最新</a>
            </li>
            <li class="current bg-orange">
              <a title="价格" href="#">价格 
                <span>↓</span>
              </a>
            </li>
          </ol>
        </section>
```

```
        </div>
        <div class="mt40">
          <!-- /无数据提示 开始-->
          <section class="no-data-wrap">
            <em class="icon30 no-data-ico"> </em>
            <span class="c-666 fsize14 ml10 vam">没有相关数据，小编正在努力整理中...</span>
          </section>
          <!-- /无数据提示 结束-->
          <article class="comm-course-list">
            <ul class="of" id="bna">
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442295592705.jpg" class="img-responsive" alt="听力口语">
                    <div class="cc-mask">
                      <a href="/course/1" title="开始学习" class="comm-btn c-btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="/course/1" title="听力口语" class="course-title fsize18 c-333">听力口语</a>
                  </h3>
                  <section class="mt10 hLh20 of">
                    <span class="fr jgTag bg-green">
                      <i class="c-fff fsize12 f-fA">免费</i>
                    </span>
                    <span class="fl jgAttr c-ccc f-fA">
                      <i class="c-999 f-fA">9634人学习</i>
                      |
                      <i class="c-999 f-fA">9634评论</i>
                    </span>
                  </section>
                </div>
              </li>
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442295581911.jpg" class="img-responsive" alt="Java精品课程">
                    <div class="cc-mask">
                      <a href="/course/1" title="开始学习" class="comm-btn c-btn-1">开始学习</a>
```

```
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="/course/1" title="Java精品课程" class="course-
title fsize18 c-333">Java精品课程</a>
                  </h3>
                  <section class="mt10 hLh20 of">
                    <span class="fr jgTag bg-green">
                      <i class="c-fff fsize12 f-fA">免费</i>
                    </span>
                    <span class="fl jgAttr c-ccc f-fA">
                      <i class="c-999 f-fA">501人学习</i>
                      |
                      <i class="c-999 f-fA">501评论</i>
                    </span>
                  </section>
                </div>
              </li>
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442295604295.jpg"
class="img-responsive" alt="C4D零基础">
                    <div class="cc-mask">
                      <a href="/course/1" title="开始学习" class="comm-btn c-
btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="/course/1" title="C4D零基础" class="course-title
fsize18 c-333">C4D零基础</a>
                  </h3>
                  <section class="mt10 hLh20 of">
                    <span class="fr jgTag bg-green">
                      <i class="c-fff fsize12 f-fA">免费</i>
                    </span>
                    <span class="fl jgAttr c-ccc f-fA">
                      <i class="c-999 f-fA">300人学习</i>
                      |
                      <i class="c-999 f-fA">300评论</i>
                    </span>
                  </section>
                </div>
              </li>
              <li>
```

```
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img
                      src="~/assets/photo/course/1442302831779.jpg"
                      class="img-responsive"
                      alt="数学给宝宝带来的兴趣"
                    >
                    <div class="cc-mask">
                      <a href="/course/1" title="开始学习" class="comm-btn c-
btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="/course/1" title="数学给宝宝带来的兴趣"
class="course-title fsize18 c-333">数学给宝宝带来的兴趣</a>
                  </h3>
                  <section class="mt10 hLh20 of">
                    <span class="fr jgTag bg-green">
                      <i class="c-fff fsize12 f-fA">免费</i>
                    </span>
                    <span class="fl jgAttr c-ccc f-fA">
                      <i class="c-999 f-fA">256人学习</i>
                      |
                      <i class="c-999 f-fA">256评论</i>
                    </span>
                  </section>
                </div>
              </li>
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img
                      src="~/assets/photo/course/1442295455437.jpg"
                      class="img-responsive"
                      alt="零基础入门学习Python课程学习"
                    >
                    <div class="cc-mask">
                      <a href="/course/1" title="开始学习" class="comm-btn c-
btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a
                      href="/course/1"
                      title="零基础入门学习Python课程学习"
```

```
                    class="course-title fsize18 c-333"
                  >零基础入门学习Python课程学习</a>
                </h3>
                <section class="mt10 hLh20 of">
                  <span class="fr jgTag bg-green">
                    <i class="c-fff fsize12 f-fA">免费</i>
                  </span>
                  <span class="fl jgAttr c-ccc f-fA">
                    <i class="c-999 f-fA">137人学习</i>
                    |
                    <i class="c-999 f-fA">137评论</i>
                  </span>
                </section>
              </div>
            </li>
            <li>
              <div class="cc-l-wrap">
                <section class="course-img">
                  <img
                    src="~/assets/photo/course/1442295570359.jpg"
                    class="img-responsive"
                    alt="MySql从入门到精通"
                  >
                  <div class="cc-mask">
                    <a href="/course/1" title="开始学习" class="comm-btn c-
btn-1">开始学习</a>
                  </div>
                </section>
                <h3 class="hLh30 txtOf mt10">
                  <a href="/course/1" title="MySql从入门到精通"
class="course-title fsize18 c-333">MySql从入门到精通</a>
                </h3>
                <section class="mt10 hLh20 of">
                  <span class="fr jgTag bg-green">
                    <i class="c-fff fsize12 f-fA">免费</i>
                  </span>
                  <span class="fl jgAttr c-ccc f-fA">
                    <i class="c-999 f-fA">125人学习</i>
                    |
                    <i class="c-999 f-fA">125评论</i>
                  </span>
                </section>
              </div>
            </li>
            <li>
```

```
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442302852837.jpg"
class="img-responsive" alt="搜索引擎优化技术">
                    <div class="cc-mask">
                      <a href="/course/1" title="开始学习" class="comm-btn c-
btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="/course/1" title="搜索引擎优化技术" class="course-
title fsize18 c-333">搜索引擎优化技术</a>
                  </h3>
                  <section class="mt10 hLh20 of">
                    <span class="fr jgTag bg-green">
                      <i class="c-fff fsize12 f-fA">免费</i>
                    </span>
                    <span class="fl jgAttr c-ccc f-fA">
                      <i class="c-999 f-fA">123人学习</i>
                      |
                      <i class="c-999 f-fA">123评论</i>
                    </span>
                  </section>
                </div>
              </li>
              <li>
                <div class="cc-l-wrap">
                  <section class="course-img">
                    <img src="~/assets/photo/course/1442295379715.jpg"
class="img-responsive" alt="20世纪西方音乐">
                    <div class="cc-mask">
                      <a href="/course/1" title="开始学习" class="comm-btn c-
btn-1">开始学习</a>
                    </div>
                  </section>
                  <h3 class="hLh30 txtOf mt10">
                    <a href="/course/1" title="20世纪西方音乐" class="course-
title fsize18 c-333">20世纪西方音乐</a>
                  </h3>
                  <section class="mt10 hLh20 of">
                    <span class="fr jgTag bg-green">
                      <i class="c-fff fsize12 f-fA">免费</i>
                    </span>
                    <span class="fl jgAttr c-ccc f-fA">
                      <i class="c-999 f-fA">34人学习</i>
```

```
                    |
                    <i class="c-999 f-fA">34评论</i>
                  </span>
                </section>
              </div>
            </li>
          </ul>
          <div class="clear"></div>
        </article>
      </div>
      <!-- 公共分页 开始 -->
      <div>
        <div class="paging">
          <a class="undisable" title>首</a>
          <a id="backpage" class="undisable" href="#" title>&lt;</a>
          <a href="#" title class="current undisable">1</a>
          <a href="#" title>2</a>
          <a id="nextpage" href="#" title>&gt;</a>
          <a href="#" title>末</a>
          <div class="clear"></div>
        </div>
      </div>
      <!-- 公共分页 结束 -->
    </section>
  </section>
  <!-- /课程列表 结束 -->
 </div>
</template>
<script>
export default {};
</script>
```

# 二、详情页面

创建 pages/course/_id.vue

```
<template>
  <div id="aCoursesList" class="bg-fa of">
    <!-- /课程详情 开始 -->
    <section class="container">
      <section class="path-wrap txtOf hLh30">
```

```
        <a href="#" title class="c-999 fsize14">首页</a>
        \
        <a href="#" title class="c-999 fsize14">课程列表</a>
        \
        <span class="c-333 fsize14">Java精品课程</span>
      </section>
      <div>
        <article class="c-v-pic-wrap" style="height: 357px;">
          <section class="p-h-video-box" id="videoPlay">
            <img src="~/assets/photo/course/1442295581911.jpg" alt="Java精品课程" class="dis c-v-pic">
          </section>
        </article>
        <aside class="c-attr-wrap">
          <section class="ml20 mr15">
            <h2 class="hLh30 txtOf mt15">
              <span class="c-fff fsize24">Java精品课程</span>
            </h2>
            <section class="c-attr-jg">
              <span class="c-fff">价格: </span>
              <b class="c-yellow" style="font-size:24px;">￥0.00</b>
            </section>
            <section class="c-attr-mt c-attr-undis">
              <span class="c-fff fsize14">主讲:  唐嫣&amp;nbsp;&amp;nbsp;&amp;nbsp;</span>
            </section>
            <section class="c-attr-mt of">
              <span class="ml10 vam">
                <em class="icon18 scIcon"></em>
                <a class="c-fff vam" title="收藏" href="#" >收藏</a>
              </span>
            </section>
            <section class="c-attr-mt">
              <a href="#" title="立即观看" class="comm-btn c-btn-3">立即观看</a>
            </section>
          </section>
        </aside>
        <aside class="thr-attr-box">
          <ol class="thr-attr-ol clearfix">
            <li>
              <p>&amp;nbsp;</p>
              <aside>
                <span class="c-fff f-fM">购买数</span>
                <br>
                <h6 class="c-fff f-fM mt10">150</h6>
```

```
              </aside>
            </li>
            <li>
              <p> </p>
              <aside>
                <span class="c-fff f-fM">课时数</span>
                <br>
                <h6 class="c-fff f-fM mt10">20</h6>
              </aside>
            </li>
            <li>
              <p> </p>
              <aside>
                <span class="c-fff f-fM">浏览数</span>
                <br>
                <h6 class="c-fff f-fM mt10">501</h6>
              </aside>
            </li>
          </ol>
        </aside>
        <div class="clear"></div>
      </div>
      <!-- /课程封面介绍 -->
      <div class="mt20 c-infor-box">
        <article class="fl col-7">
          <section class="mr30">
            <div class="i-box">
              <div>
                <section id="c-i-tabTitle" class="c-infor-tabTitle c-tab-
title">
                  <a name="c-i" class="current" title="课程详情">课程详情</a>
                </section>
              </div>
              <article class="ml10 mr10 pt20">
                <div>
                  <h6 class="c-i-content c-infor-title">
                    <span>课程介绍</span>
                  </h6>
                  <div class="course-txt-body-wrap">
                    <section class="course-txt-body">
                      <p>
                        Java的发展历史，可追溯到1990年。当
时Sun Microsystem公司为了发展消费性电子产品而进行了一个名为Green的项目计划。该
计划
                        负责人是James Gosling。起初他以C++来写一种内嵌式软
```

```
件，可以放在烤面包机或PAD等小型电子消费设备里，使得机器更聪明，具有人工智
                        能。但他发现C++并不适合完成这类任务！因为C++常会有使系统失效的程序错误，尤其是内存管理，需要程序设计师记录并管理内存资源。这给设计师们造成
                        极大的负担，并可能产生许多bugs。 
                        <br>为了解决所遇到的问题，Gosling决定要发展一种新的语言，来解决C++的潜在性危险问题，这个语言名叫Oak。Oak是一种可移植性语言，也就是一种平台独立语言，能够在各种芯片上运行。
                        <br>1994年，Oak技术日趋成熟，这时网络正开始蓬勃发展。Oak研发小组发现Oak很适合作为一种网络程序语言。因此发展了一个能与Oak配合的浏
                        览器--WebRunner，后更名为HotJava，它证明了Oak是一种能在网络上发展的程序语言。由于Oak商标已被注册，工程师们便想到以自己常
                        享用的咖啡(Java)来重新命名，并于Sun World 95中被发表出来。
                    </p>
                  </section>
                </div>
              </div>
              <!-- /课程介绍 -->
              <div class="mt50">
                <h6 class="c-g-content c-infor-title">
                  <span>课程大纲</span>
                </h6>
                <section class="mt20">
                  <div class="lh-menu-wrap">
                    <menu id="lh-menu" class="lh-menu mt10 mr10">
                      <ul>
                        <!-- 文件目录 -->
                        <li class="lh-menu-stair">
                          <a href="javascript: void(0)" title="第一章" class="current-1">
                            <em class="lh-menu-i-1 icon18 mr10"></em>第一章
                          </a>
                          <ol class="lh-menu-ol" style="display: block;">
                            <li class="lh-menu-second ml30">
                              <a href="#" title>
                                <span class="fr">
                                  <i class="free-icon vam mr10">免费试听</i>
                                </span>
                                <em class="lh-menu-i-2 icon16 mr5"> </em>第一节
                              </a>
                            </li>
                            <li class="lh-menu-second ml30">
                              <a href="#" title class="current-2">
                                <em class="lh-menu-i-2 icon16
```

```
mr5"> </em>第二节
                                </a>
                              </li>
                            </ol>
                          </li>
                        </ul>
                      </menu>
                    </div>
                  </section>
                </div>
                <!-- /课程大纲 -->
              </article>
            </div>
          </section>
        </article>
        <aside class="fl col-3">
          <div class="i-box">
            <div>
              <section class="c-infor-tabTitle c-tab-title">
                <a title href="javascript:void(0)">主讲讲师</a>
              </section>
              <section class="stud-act-list">
                <ul style="height: auto;">
                  <li>
                    <div class="u-face">
                      <a href="#">
                        <img src="~/assets/photo/teacher/1442297969808.jpg"
width="50" height="50" alt>
                      </a>
                    </div>
                    <section class="hLh30 txtOf">
                      <a class="c-333 fsize16 fl" href="#">周杰伦</a>
                    </section>
                    <section class="hLh20 txtOf">
                      <span class="c-999">毕业于北京大学数学系</span>
                    </section>
                  </li>
                </ul>
              </section>
            </div>
          </div>
        </aside>
        <div class="clear"></div>
      </div>
    </section>
```

```
    <!-- /课程详情 结束 -->
  </div>
</template>
<script>
export default {};
</script>

```

# 一、课程后端接口

## 1、课程列表

### （1）课程列表vo类

```
@ApiModel(value = "课程查询对象", description = "课程查询对象封装")
@Data
public class CourseQueryVo implements Serializable {
    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "课程名称")
    private String title;

    @ApiModelProperty(value = "讲师id")
    private String teacherId;

    @ApiModelProperty(value = "一级类别id")
    private String subjectParentId;

    @ApiModelProperty(value = "二级类别id")
    private String subjectId;

    @ApiModelProperty(value = "销量排序")
    private String buyCountSort;

    @ApiModelProperty(value = "最新时间排序")
    private String gmtCreateSort;

    @ApiModelProperty(value = "价格排序")
    private String priceSort;
}
```

### （2）课程列表**controller**

```
@ApiOperation(value = "分页课程列表")
```

```
@PostMapping(value = "{page}/{limit}")
public R pageList(
        @ApiParam(name = "page", value = "当前页码", required = true)
        @PathVariable Long page,

        @ApiParam(name = "limit", value = "每页记录数", required = true)
        @PathVariable Long limit,

        @ApiParam(name = "courseQuery", value = "查询对象", required = false)
                @RequestBody(required = false) CourseQueryVo courseQuery){
    Page<EduCourse> pageParam = new Page<EduCourse>(page, limit);
    Map<String, Object> map = courseService.pageListWeb(pageParam, courseQuery);
    return  R.ok().data(map);
}
```

### （3）课程列表**service**

```
@Override
public Map<String, Object> pageListWeb(Page<EduCourse> pageParam, CourseQueryVo
courseQuery) {
        QueryWrapper<EduCourse> queryWrapper = new QueryWrapper<>();
        if (!StringUtils.isEmpty(courseQuery.getSubjectParentId())) {
            queryWrapper.eq("subject_parent_id",
courseQuery.getSubjectParentId());
        }

        if (!StringUtils.isEmpty(courseQuery.getSubjectId())) {
            queryWrapper.eq("subject_id", courseQuery.getSubjectId());
        }

        if (!StringUtils.isEmpty(courseQuery.getBuyCountSort())) {
            queryWrapper.orderByDesc("buy_count");
        }

        if (!StringUtils.isEmpty(courseQuery.getGmtCreateSort())) {
            queryWrapper.orderByDesc("gmt_create");
        }

        if (!StringUtils.isEmpty(courseQuery.getPriceSort())) {
            queryWrapper.orderByDesc("price");
```

```
        }

        baseMapper.selectPage(pageParam, queryWrapper);

        List<EduCourse> records = pageParam.getRecords();
        long current = pageParam.getCurrent();
        long pages = pageParam.getPages();
        long size = pageParam.getSize();
        long total = pageParam.getTotal();
        boolean hasNext = pageParam.hasNext();
        boolean hasPrevious = pageParam.hasPrevious();

        Map<String, Object> map = new HashMap<String, Object>();
        map.put("items", records);
        map.put("current", current);
        map.put("pages", pages);
        map.put("size", size);
        map.put("total", total);
        map.put("hasNext", hasNext);
        map.put("hasPrevious", hasPrevious);

        return map;
}
```

# 二、课程列表前端

## 1、定义api

api/course.js

```
import request from '@/utils/request'
export default {
  getPageList(page, limit, searchObj) {
    return request({
      url: `/eduservice/edu/course/${page}/${limit}`,
      method: 'post',
      data: searchObj
    })
```

```
  },
  // 获取课程二级分类
  getNestedTreeList2() {
    return request({
      url: `/eduservice/edu/course/list2`,
      method: 'get'
    })
  }
}
```

## 2、页面调用接口

pages/course/index.vue

```
<script>
import course from '@/api/course'

export default {

  data () {
    return {
      page:1,
      data:{},
      subjectNestedList: [], // 一级分类列表
      subSubjectList: [], // 二级分类列表
      searchObj: {}, // 查询表单对象
      oneIndex:-1,
      twoIndex:-1,
      buyCountSort:"",
      gmtCreateSort:"",
      priceSort:""
    }
  },

  //加载完渲染时
  created () {
    //获取课程列表
    this.initCourse()

    //获取分类
    this.initSubject()
```

```
  },

  methods: {
    //查询课程列表
    initCourse(){
      course.getPageList(1, 8,this.searchObj).then(response => {
        this.data = response.data.data
      })
    },
    //查询所有一级分类
    initSubject(){
      //debugger
      course.getNestedTreeList2().then(response => {
        this.subjectNestedList = response.data.data.items
      })
    },

    //点击一级分类，显示对应的二级分类，查询数据
    searchOne(subjectParentId, index) {
      //debugger
      this.oneIndex = index
      this.twoIndex = -1

      this.searchObj.subjectId = "";
      this.subSubjectList = [];

      this.searchObj.subjectParentId = subjectParentId;
      this.gotoPage(this.page)

      for (let i = 0; i < this.subjectNestedList.length; i++) {
        if (this.subjectNestedList[i].id === subjectParentId) {
          this.subSubjectList = this.subjectNestedList[i].children
        }
      }
    },

    //点击二级分类，查询数据
    searchTwo(subjectId, index) {
      this.twoIndex = index
      this.searchObj.subjectId = subjectId;
      this.gotoPage(this.page)
    },
    //购买量查询
```

```
    searchBuyCount() {
      this.buyCountSort = "1";
      this.gmtCreateSort = "";
      this.priceSort = "";
      this.searchObj.buyCountSort = this.buyCountSort;
      this.searchObj.gmtCreateSort = this.gmtCreateSort;
      this.searchObj.priceSort = this.priceSort;
      this.gotoPage(this.page)
    },
    //更新时间查询
    searchGmtCreate() {
      debugger
      this.buyCountSort = "";
      this.gmtCreateSort = "1";
      this.priceSort = "";
      this.searchObj.buyCountSort = this.buyCountSort;
      this.searchObj.gmtCreateSort = this.gmtCreateSort;
      this.searchObj.priceSort = this.priceSort;
      this.gotoPage(this.page)
    },
    //价格查询
    searchPrice() {
      this.buyCountSort = "";
      this.gmtCreateSort = "";
      this.priceSort = "1";
      this.searchObj.buyCountSort = this.buyCountSort;
      this.searchObj.gmtCreateSort = this.gmtCreateSort;
      this.searchObj.priceSort = this.priceSort;
      this.gotoPage(this.page)
    },
    //分页查询
    gotoPage(page) {
      this.page = page
      course.getPageList(page, 8, this.searchObj).then(response => {
        this.data = response.data.data
      })
    }
  }
}
</script>
<style scoped>
  .active {
    background: #bdbdbd;
```

```
  }
  .hide {
    display: none;
  }
  .show {
    display: block;
  }
</style>

```

# 三、课程列表渲染

## 1、课程类别显示

```
<section class="c-s-dl">
  <dl>
    <dt>
      <span class="c-999 fsize14">课程类别</span>
    </dt>
    <dd class="c-s-dl-li">
      <ul class="clearfix">
        <li>
          <a title="全部" href="javascript:void(0);" @click="searchOne('')">全部</a>
        </li>
        <li v-for="(item,index) in subjectNestedList" v-bind:key="index" :class="{active:oneIndex==index}">
          <a :title="item.title" href="javascript:void(0);" @click="searchOne(item.id, index)">{{item.title}}</a>
        </li>
      </ul>
    </dd>
  </dl>
  <dl>
    <dt>
      <span class="c-999 fsize14"/>
    </dt>
    <dd class="c-s-dl-li">
      <ul class="clearfix">
        <li v-for="(item,index) in subSubjectList" v-bind:key="index" :class="
```

```
{active:twoIndex==index}">
24          <a :title="item.title" href="javascript:void(0);"
@click="searchTwo(item.id, index)">{{item.title}}</a>
25        </li>
26      </ul>
27    </dd>
28  </dl>
29  <div class="clear"/>
30 </section>
```

## 2、排序方式显示

```
1 <section class="fl">
2   <ol class="js-tap clearfix">
3     <li :class="{'current bg-orange':buyCountSort!=''}">
4       <a title="销量" href="javascript:void(0);" @click="searchBuyCount()">销量
5       <span :class="{hide:buyCountSort==''}">↓</span>
6       </a>
7     </li>
8     <li :class="{'current bg-orange':gmtCreateSort!=''}">
9       <a title="最新" href="javascript:void(0);" @click="searchGmtCreate()">最新
10      <span :class="{hide:gmtCreateSort==''}">↓</span>
11      </a>
12    </li>
13    <li :class="{'current bg-orange':priceSort!=''}">
14      <a title="价格" href="javascript:void(0);" @click="searchPrice()">价格 
15        <span :class="{hide:priceSort==''}">↓</span>
16      </a>
17    </li>
18  </ol>
19 </section>
```

## 3、无数据提示

添加：v-if="data.total==0"

```
1 <!-- /无数据提示 开始-->
2 <section class="no-data-wrap" v-if="data.total==0">
```

```
    <em class="icon30 no-data-ico"> </em>
    <span class="c-666 fsize14 ml10 vam">没有相关数据，小编正在努力整理中...</span>
</section>
<!-- /无数据提示 结束-->
```

## 4、列表

```
<!-- 数据列表 开始-->
<article v-if="data.total>0" class="comm-course-list">
    <ul id="bna" class="of">
        <li v-for="item in data.items" :key="item.id">
            <div class="cc-l-wrap">
                <section class="course-img">
                    <img :src="item.cover" class="img-responsive" alt="听力口语">
                    <div class="cc-mask">
                        <a :href="'/course/'+item.id" title="开始学习" class="comm-btn c-btn-1">开始学习</a>
                    </div>
                </section>
                <h3 class="hLh30 txtOf mt10">
                    <a :href="'/course/'+item.id" :title="item.title" class="course-title fsize18 c-333">{{ item.title }}</a>
                </h3>
                <section class="mt10 hLh20 of">
                    <span v-if="Number(item.price) === 0" class="fr jgTag bg-green">
                        <i class="c-fff fsize12 f-fA">免费</i>
                    </span>
                    <span class="fl jgAttr c-ccc f-fA">
                        <i class="c-999 f-fA">{{ item.viewCount }}人学习</i>
                        |
                        <i class="c-999 f-fA">9634评论</i>
                    </span>
                </section>
            </div>
        </li>
    </ul>
    <div class="clear"/>
</article>
<!-- /数据列表 结束-->
```

# 5、分页页面渲染

```
<div>
  <div class="paging">
    <!-- undisable这个class是否存在，取决于数据属性hasPrevious -->
    <a
      :class="{undisable: !data.hasPrevious}"
      href="#"
      title="首页"
      @click.prevent="gotoPage(1)">首</a>
    <a
      :class="{undisable: !data.hasPrevious}"
      href="#"
      title="前一页"
      @click.prevent="gotoPage(data.current-1)">&lt;</a>
    <a
      v-for="page in data.pages"
      :key="page"
      :class="{current: data.current == page, undisable: data.current == page}"
      :title="'第'+page+'页'"
      href="#"
      @click.prevent="gotoPage(page)">{{ page }}</a>
    <a
      :class="{undisable: !data.hasNext}"
      href="#"
      title="后一页"
      @click.prevent="gotoPage(data.current+1)">&gt;</a>
    <a
      :class="{undisable: !data.hasNext}"
      href="#"
      title="末页"
      @click.prevent="gotoPage(data.pages)">末</a>
    <div class="clear"/>
  </div>
</div>
```

# 一、vo对象的定义

在项目中很多时候需要把model转换成dto用于网站信息的展示，按前端的需要传递对象的数据，保证model对外是隐私的，例如密码之类的属性能很好地避免暴露在外，同时也会减小数据传输的体积。

CourseWebVo.java

```
package com.guli.edu.vo;

@ApiModel(value="课程信息", description="网站课程详情页需要的相关字段")
@Data
public class CourseWebVo implements Serializable {

    private static final long serialVersionUID = 1L;

    private String id;

    @ApiModelProperty(value = "课程标题")
    private String title;

    @ApiModelProperty(value = "课程销售价格，设置为0则可免费观看")
    private BigDecimal price;

    @ApiModelProperty(value = "总课时")
    private Integer lessonNum;

    @ApiModelProperty(value = "课程封面图片路径")
    private String cover;

    @ApiModelProperty(value = "销售数量")
    private Long buyCount;

    @ApiModelProperty(value = "浏览数量")
    private Long viewCount;

    @ApiModelProperty(value = "课程简介")
    private String description;

    @ApiModelProperty(value = "讲师ID")
    private String teacherId;

```

```
    @ApiModelProperty(value = "讲师姓名")
    private String teacherName;

    @ApiModelProperty(value = "讲师资历,一句话说明讲师")
    private String intro;

    @ApiModelProperty(value = "讲师头像")
    private String avatar;

    @ApiModelProperty(value = "课程类别ID")
    private String subjectLevelOneId;

    @ApiModelProperty(value = "类别名称")
    private String subjectLevelOne;

    @ApiModelProperty(value = "课程类别ID")
    private String subjectLevelTwoId;

    @ApiModelProperty(value = "类别名称")
    private String subjectLevelTwo;

}
```

# 二、课程和讲师信息的获取

## 1、Mapper中关联查询课程和讲师信息

CourseMapper.java

```
CourseWebVo selectInfoWebById(String courseId);
```

CourseMapper.xml

```
<select id="selectInfoWebById" resultType="com.guli.edu.vo.CourseWebVo">
  SELECT
    c.id,
    c.title,
    c.cover,
```

```
    CONVERT(c.price, DECIMAL(8,2)) AS price,
    c.lesson_num AS lessonNum,
    c.cover,
    c.buy_count AS buyCount,
    c.view_count AS viewCount,
    cd.description,

    t.id AS teacherId,
    t.name AS teacherName,
    t.intro,
    t.avatar,

    s1.id AS subjectLevelOneId,
    s1.title AS subjectLevelOne,
    s2.id AS subjectLevelTwoId,
    s2.title AS subjectLevelTwo

  FROM
    edu_course c
    LEFT JOIN edu_course_description cd ON c.id = cd.id
    LEFT JOIN edu_teacher t ON c.teacher_id = t.id
    LEFT JOIN edu_subject s1 ON c.subject_parent_id = s1.id
    LEFT JOIN edu_subject s2 ON c.subject_id = s2.id
  WHERE
    c.id = #{id}
</select>
```

## 2、业务层获取数据并更新浏览量

CourseService

接口

```
/**
     * 获取课程信息
     * @param id
     * @return
     */
CourseWebVo selectInfoWebById(String id);

```

```
/**
     * 更新课程浏览数
     * @param id
     */
void updatePageViewCount(String id);
```

实现

```
@Override
public CourseWebVo selectInfoWebById(String id) {
    this.updatePageViewCount(id);
    return baseMapper.selectInfoWebById(id);
}

@Override
public void updatePageViewCount(String id) {
    Course course = baseMapper.selectById(id);
    course.setViewCount(course.getViewCount() + 1);
    baseMapper.updateById(course);
}
```

# 3、接口层

CourseController

```
@Autowired
private ChapterService chapterService;

@ApiOperation(value = "根据ID查询课程")
@GetMapping(value = "{courseId}")
public R getById(
    @ApiParam(name = "courseId", value = "课程ID", required = true)
    @PathVariable String courseId){

    //查询课程信息和讲师信息
    CourseWebVo courseWebVo = courseService.selectInfoWebById(courseId);

    //查询当前课程的章节信息
    List<ChapterVo> chapterVoList = chapterService.nestedList(courseId);
```

```
15
16      return R.ok().data("course", courseWebVo).data("chapterVoList",
chapterVoList);
17  }
```

### 4、swagger测试

# 三、前端js

## 1、api/course.js

```
1  getById(courseId) {
2      return request({
3          url: `${api_name}/${courseId}`,
4          method: 'get'
5      })
6  }
```

## 2、pages/course/_id.vue

```
1  <script>
2  import course from "@/api/course"
3  export default {
4    asyncData({ params, error }) {
5      return course.getById(params.id).then(response => {
6        console.log(response);
7        return {
8          course: response.data.data.course,
9          chapterList: response.data.data.chapterVoList
10       }
11     })
12   }
13 }
14 </script>
```

# 四、页面模板

pages/course/_id.vue

## 1、template

```
<template>
  <div id="aCoursesList" class="bg-fa of">
    <!-- 课程详情 开始 -->
    <section class="container">

      <!-- 课程所属分类 开始 -->

      <!-- /课程所属分类 结束 -->

      <!-- 课程基本信息 开始 -->

      <!-- /课程基本信息 结束 -->

      <div class="mt20 c-infor-box">
        <article class="fl col-7">
          <section class="mr30">
            <div class="i-box">
              <div>
                <section id="c-i-tabTitle" class="c-infor-tabTitle c-tab-title">
                  <a name="c-i" class="current" title="课程详情">课程详情</a>
                </section>
              </div>
              <article class="ml10 mr10 pt20">

                <!-- 课程详情介绍 开始 -->

                <!-- /课程详情介绍 结束 -->

                <!-- 课程大纲 开始-->

                <!-- /课程大纲 结束 -->
              </article>
            </div>
          </section>
```

```
        </article>
        <aside class="fl col-3">
          <div class="i-box">
            <!-- 主讲讲师 开始-->

            <!-- /主讲讲师 结束 -->
          </div>
        </aside>
        <div class="clear"/>
      </div>
    </section>
    <!-- /课程详情 结束 -->
  </div>
</template>
```

## 2、课程所属分类

```
<!-- 课程所属分类 开始 -->
<section class="path-wrap txtOf hLh30">
    <a href="#" title class="c-999 fsize14">首页</a>
    \
    <a href="/course" title class="c-999 fsize14">课程列表</a>
    \
    <span class="c-333 fsize14">{{ course.subjectLevelOne }}</span>
    \
    <span class="c-333 fsize14">{{ course.subjectLevelTwo }}</span>
</section>
<!-- /课程所属分类 结束 -->
```

## 3、课程基本信息

```
<!-- 课程基本信息 开始 -->
<div>
    <article class="c-v-pic-wrap" style="height: 357px;">
        <section id="videoPlay" class="p-h-video-box">
            <img :src="course.cover" :alt="course.title" class="dis c-v-pic">
        </section>
```

```
    </article>
    <aside class="c-attr-wrap">
        <section class="ml20 mr15">
            <h2 class="hLh30 txtOf mt15">
                <span class="c-fff fsize24">{{ course.title }}</span>
            </h2>
            <section class="c-attr-jg">
                <span class="c-fff">价格: </span>
                <b class="c-yellow" style="font-size:24px;">￥{{ course.price
}}</b>
            </section>
            <section class="c-attr-mt c-attr-undis">
                <span class="c-fff fsize14">主讲:  {{ course.teacherName
}}   </span>
            </section>
            <section class="c-attr-mt of">
                <span class="ml10 vam">
                    <em class="icon18 scIcon"/>
                    <a class="c-fff vam" title="收藏" href="#" >收藏</a>
                </span>
            </section>
            <section class="c-attr-mt">
                <a href="#" title="立即观看" class="comm-btn c-btn-3">立即观看</a>
            </section>
        </section>
    </aside>
    <aside class="thr-attr-box">
        <ol class="thr-attr-ol clearfix">
            <li>
                <p> </p>
                <aside>
                    <span class="c-fff f-fM">购买数</span>
                    <br>
                    <h6 class="c-fff f-fM mt10">{{ course.buyCount }}</h6>
                </aside>
            </li>
            <li>
                <p> </p>
                <aside>
                    <span class="c-fff f-fM">课时数</span>
                    <br>
                    <h6 class="c-fff f-fM mt10">{{ course.lessonNum }}</h6>
                </aside>
```

```
            </li>
            <li>
                <p> </p>
                <aside>
                    <span class="c-fff f-fM">浏览数</span>
                    <br>
                    <h6 class="c-fff f-fM mt10">{{ course.viewCount }}</h6>
                </aside>
            </li>
        </ol>
    </aside>
    <div class="clear"/>
</div>
<!-- /课程基本信息 结束 -->
```

## 4、课程详情介绍

```
<!-- 课程详情介绍 开始 -->
<div>
    <h6 class="c-i-content c-infor-title">
        <span>课程介绍</span>
    </h6>
    <div class="course-txt-body-wrap">
        <section class="course-txt-body">
            <!-- 将内容中的html翻译过来 -->
            <p v-html="course.description">{{ course.description }}</p>
        </section>
    </div>
</div>
<!-- /课程详情介绍 结束 -->
```

## 5、课程大纲

```
<!-- 课程大纲 开始-->
<div class="mt50">
    <h6 class="c-g-content c-infor-title">
        <span>课程大纲</span>
```

```
    </h6>
    <section class="mt20">
        <div class="lh-menu-wrap">
            <menu id="lh-menu" class="lh-menu mt10 mr10">
                <ul>
                    <!-- 课程章节目录 -->
                    <li v-for="chapter in chapterList" :key="chapter.id"
class="lh-menu-stair">
                        <a :title="chapter.title" href="javascript: void(0)"
class="current-1">
                            <em class="lh-menu-i-1 icon18 mr10"/>{{ chapter.title
}}
                        </a>
                        <ol class="lh-menu-ol" style="display: block;">
                            <li v-for="video in chapter.children" :key="video.id"
class="lh-menu-second ml30">
                                <a href="#" title>
                                    <span v-if="video.free === true" class="fr">
                                        <i class="free-icon vam mr10">免费试听</i>
                                    </span>
                                    <em class="lh-menu-i-2 icon16
mr5"> </em>{{ video.title }}
                                </a>
                            </li>
                        </ol>
                    </li>
                </ul>
            </menu>
        </div>
    </section>
    <!-- /课程大纲 结束 -->
```

# 6、主讲讲师

```
<!-- 主讲讲师 开始-->
<div>
    <section class="c-infor-tabTitle c-tab-title">
        <a title href="javascript:void(0)">主讲讲师</a>
    </section>
    <section class="stud-act-list">
        <ul style="height: auto;">
```

```
            <li>
                <div class="u-face">
                    <a :href="'/teacher/'+course.teacherId" target="_blank">
                        <img :src="course.avatar" width="50" height="50" alt>
                    </a>
                </div>
                <section class="hLh30 txtOf">
                    <a :href="'/teacher/'+course.teacherId" class="c-333 fsize16 fl" target="_blank">{{ course.teacherName }}</a>
                </section>
                <section class="hLh20 txtOf">
                    <span class="c-999">{{ course.intro }}</span>
                </section>
            </li>
        </ul>
    </section>
</div>
<!-- /主讲讲师 结束 -->
```

# 一、获取播放地址播放

## 获取播放地址

参考文档：https://help.aliyun.com/document_detail/61064.html

前面的 03-使用服务端SDK 介绍了如何获取非加密视频的播放地址。直接使用03节的例子获取加密视频播放地址会返回如下错误信息

Currently only the AliyunVoDEncryption stream exists, you must use the Aliyun player to play or set the value of ResultType to Multiple.

目前只有AliyunVoDEncryption流存在，您必须使用Aliyun player来播放或将ResultType的值设置为Multiple。

因此在testGetPlayInfo测试方法中添加 ResultType 参数，并设置为true

```
privateParams.put("ResultType", "Multiple");
```

此种方式获取的视频文件不能直接播放，必须使用阿里云播放器播放

# 二、视频播放器

参考文档：https://help.aliyun.com/document_detail/61109.html

## 1、视频播放器介绍

阿里云播放器SDK（ApsaraVideo Player SDK）是阿里视频服务的重要一环，除了支持点播和直播的基础播放功能外，深度融合视频云业务，如支持视频的加密播放、安全下载、清晰度切换、直播答题等业务场景，为用户提供简单、快速、安全、稳定的视频播放服务。

## 2、集成视频播放器

参考文档：https://help.aliyun.com/document_detail/51991.html

参考 【播放器简单使用说明】一节

引入脚本文件和css文件

```
<link rel="stylesheet"
href="https://g.alicdn.com/de/prismplayer/2.8.1/skins/default/aliplayer-
```

```
min.css" />
<script charset="utf-8" type="text/javascript"
src="https://g.alicdn.com/de/prismplayer/2.8.1/aliplayer-min.js"></script>
```

初始化视频播放器

```
<body>
    <div  class="prism-player" id="J_prismPlayer"></div>
    <script>
        var player = new Aliplayer({
            id: 'J_prismPlayer',
            width: '100%',
            autoplay: false,
            cover: 'http://liveroom-img.oss-cn-qingdao.aliyuncs.com/logo.png',

            //播放配置
        },function(player){
            console.log('播放器创建好了。')
        });
    </script>
</body>
```

## 3、播放地址播放

在Aliplayer的配置参数中添加如下属性

```
//播放方式一：支持播放地址播放,此播放优先级最高，此种方式不能播放加密视频
source : '你的视频播放地址',
```

启动浏览器运行，测试视频的播放

## 4、播放凭证播放（推荐）

阿里云播放器支持通过播放凭证自动换取播放地址进行播放，接入方式更为简单，且安全性更高。播放凭证默认时效为100秒（最大为3000秒），只能用于获取指定视频的播放地址，不能混用或重复使用。如果凭证过期则无法获取播放地址，需要重新获取凭证。

```
encryptType:'1',//如果播放加密视频，则需设置encryptType=1，非加密视频无需设置此项
```

```
vid : '视频id',
playauth : '视频授权码',
```

注意：播放凭证有过期时间，默认值：100秒 。取值范围：**100~3000**。

设置播放凭证的有效期

在获取播放凭证的测试用例中添加如下代码

```
request.setAuthInfoTimeout(200L);
```

在线配置参考：https://player.alicdn.com/aliplayer/setting/setting.html

# 一、后端获取播放凭证

## 1、VideoController

service-vod微服务中创建 VideoController.java

controller中创建 getVideoPlayAuth 接口方法

```
package com.guli.vod.controller;

@Api(description="阿里云视频点播微服务")
@CrossOrigin //跨域
@RestController
@RequestMapping("/vod/video")
public class VideoController {

    @GetMapping("get-play-auth/{videoId}")
    public R getVideoPlayAuth(@PathVariable("videoId") String videoId) throws Exception {

        //获取阿里云存储相关常量
        String accessKeyId = ConstantPropertiesUtil.ACCESS_KEY_ID;
        String accessKeySecret = ConstantPropertiesUtil.ACCESS_KEY_SECRET;

        //初始化
        DefaultAcsClient client = AliyunVodSDKUtils.initVodClient(accessKeyId, accessKeySecret);

        //请求
        GetVideoPlayAuthRequest request = new GetVideoPlayAuthRequest();
        request.setVideoId(videoId);

        //响应
        GetVideoPlayAuthResponse response = client.getAcsResponse(request);

        //得到播放凭证
        String playAuth = response.getPlayAuth();

        //返回结果
        return R.ok().message("获取凭证成功").data("playAuth", playAuth);
```

```
    }
}
```

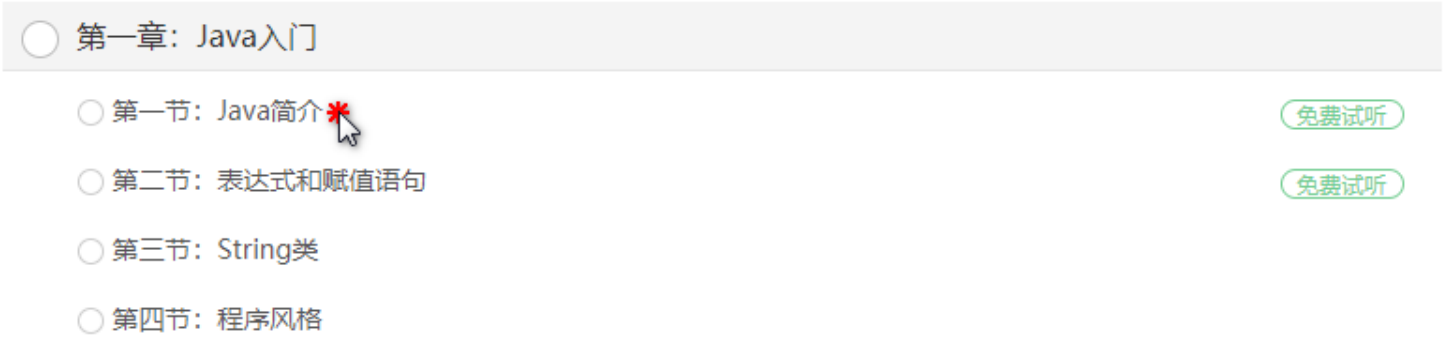
## 2、Swagger测试

# 二、前端播放器整合

## 1、点击播放超链接

course/_id.vue

修改课时目录超链接




```
<a
   :href="'/player/'+video.videoSourceId"
   :title="video.title"
   target="_blank">
```

## 2、layout

因为播放器的布局和其他页面的基本布局不一致，因此创建新的布局容器 layouts/video.vue

```
<template>
  <div class="guli-player">
    <div class="head">
      <a href="#" title="谷粒学院">
        <img class="logo" src="~/assets/img/logo.png" lt="谷粒学院">
```

```
    </a></div>
    <div class="body">
      <div class="content"><nuxt/></div>
    </div>
  </div>
</template>
<script>
export default {}
</script>

<style>
html,body{
  height:100%;
}
</style>

<style scoped>
.head {
  height: 50px;
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
}

.head .logo{
  height: 50px;
  margin-left: 10px;
}

.body {
  position: absolute;
  top: 50px;
  left: 0;
  right: 0;
  bottom: 0;
  overflow: hidden;
}
</style>
```

## 3、api

创建api模块 api/vod.js，从后端获取播放凭证

```
import request from '@/utils/request'
const api_name = '/vod/video'

export default {

  getPlayAuth(vid) {
    return request({
      url: `${api_name}/get-play-auth/${vid}`,
      method: 'get'
    })
  }

}
```

# 4、播放组件相关文档

集成文档：https://help.aliyun.com/document_detail/51991.html?spm=a2c4g.11186623.2.39.478e192b8VSdEn

在线配置：https://player.alicdn.com/aliplayer/setting/setting.html

功能展示：https://player.alicdn.com/aliplayer/presentation/index.html

# 5、创建播放页面

创建 pages/player/_vid.vue

（1）引入播放器js库和css样式

```
<template>
  <div>

    <!-- 阿里云视频播放器样式 -->
    <link rel="stylesheet"
href="https://g.alicdn.com/de/prismplayer/2.8.1/skins/default/aliplayer-min.css" >
    <!-- 阿里云视频播放器脚本 -->
    <script charset="utf-8" type="text/javascript"
```

```
src="https://g.alicdn.com/de/prismplayer/2.8.1/aliplayer-min.js" />

    <!-- 定义播放器dom -->
    <div id="J_prismPlayer" class="prism-player" />
  </div>
</template>
```

（2）获取播放凭证

```
<script>
import vod from '@/api/vod'
export default {

  layout: 'video',//应用video布局
  asyncData({ params, error }) {
    return vod.getPlayAuth(params.vid).then(response => {
      // console.log(response.data.data)
      return {
        vid: params.vid,
        playAuth: response.data.data.playAuth
      }
    })
  }
}
</script>
```

（3）创建播放器

```
 /**
 * 页面渲染完成时：此时js脚本已加载，Aliplayer已定义，可以使用
 * 如果在created生命周期函数中使用，Aliplayer is not defined错误
 */
mounted() {

    new Aliplayer({
        id: 'J_prismPlayer',
        vid: this.vid, // 视频id
        playauth: this.playAuth, // 播放凭证
        encryptType: '1', // 如果播放加密视频，则需设置encryptType=1，非加密视频无需设置此项
```

```
        width: '100%',
        height: '500px'
    }, function(player) {
        console.log('播放器创建成功')
    })
}
```

（4）其他常见的可选配置

```
// 以下可选设置
cover: 'http://guli.shop/photo/banner/1525939573202.jpg', // 封面
qualitySort: 'asc', // 清晰度排序

mediaType: 'video', // 返回音频还是视频
autoplay: false, // 自动播放
isLive: false, // 直播
rePlay: false, // 循环播放
preload: true,
controlBarVisibility: 'hover', // 控制条的显示方式：鼠标悬停
useH5Prism: true, // 播放器类型：html5
```

# 6、加入播放组件

功能展示：https://player.alicdn.com/aliplayer/presentation/index.html

# 一、数据库设计

## 1、数据库

edu_comment

## 2、数据表

```
guli_edu.sql
```

# 二、创建课程评论接口

## 1、在service-edu模块，生成课程评论代码

（1）使用**mp**代码生成器生成

## 2、在service-ucenter模块，创建接口

（1）实现用户**id**获取用户信息，返回用户信息对象

```
//根据token字符串获取用户信息
@PostMapping("getInfoUc/{id}")
public com.atguigu.commonutils.vo.UcenterMember getInfo(@PathVariable String id) {
        //根据用户id获取用户信息
        UcenterMember ucenterMember = memberService.getById(id);
        com.atguigu.commonutils.vo.UcenterMember memeber = new
com.atguigu.commonutils.vo.UcenterMember();
        BeanUtils.copyProperties(ucenterMember,memeber);
        return memeber;
}
```

## 3、创建课程评论controller

## （1）在service-edu模块创建client，实现微服务调用

```
@Component
@FeignClient(name="service-ucenter",fallback = UcenterClientImpl.class)
public interface UcenterClient {

    //根据用户id获取用户信息
    @GetMapping("/ucenterservice/member/getUcenterPay/{memberId}")
    public UcenterMemberPay getUcenterPay(@PathVariable("memberId") String memberId);
}

@Component
public class UcenterClientImpl implements UcenterClient {
    @Override
    public UcenterMemberPay getUcenterPay(String memberId) {
        return null;
    }
}
```

## （2）创建评论列表和添加评论接口

```
@RestController
@RequestMapping("/eduservice/comment")
@CrossOrigin
public class CommentFrontController {

    @Autowired
    private CommentService commentService;
    @Autowired
    private UcenterClient ucenterClient;

    //根据课程id查询评论列表
    @ApiOperation(value = "评论分页列表")
    @GetMapping("{page}/{limit}")
    public R index(
            @ApiParam(name = "page", value = "当前页码", required = true)
            @PathVariable Long page,
```

```
            @ApiParam(name = "limit", value = "每页记录数", required = true)
            @PathVariable Long limit,

            @ApiParam(name = "courseQuery", value = "查询对象", required = false)
                    String courseId) {
        Page<Comment> pageParam = new Page<>(page, limit);

        QueryWrapper<Comment> wrapper = new QueryWrapper<>();
        wrapper.eq("course_id",courseId);

        commentService.page(pageParam,wrapper);
        List<Comment> commentList = pageParam.getRecords();

        Map<String, Object> map = new HashMap<>();
        map.put("items", commentList);
        map.put("current", pageParam.getCurrent());
        map.put("pages", pageParam.getPages());
        map.put("size", pageParam.getSize());
        map.put("total", pageParam.getTotal());
        map.put("hasNext", pageParam.hasNext());
        map.put("hasPrevious", pageParam.hasPrevious());
        return R.ok().data(map);
    }

    @ApiOperation(value = "添加评论")
    @PostMapping("auth/save")
    public R save(@RequestBody Comment comment, HttpServletRequest request) {
        String memberId = JwtUtils.getMemberIdByJwtToken(request);
        if(StringUtils.isEmpty(memberId)) {
            return R.error().code(28004).message("请登录");
        }
        comment.setMemberId(memberId);

        UcenterMemberPay ucenterInfo = ucenterClient.getUcenterPay(memberId);

        comment.setNickname(ucenterInfo.getNickname());
        comment.setAvatar(ucenterInfo.getAvatar());

        commentService.save(comment);
        return R.ok();
    }
}
```

# 三、课程评论前端整合

## 1、在api创建commonedu.js

```
import request from '@/utils/request'

export default {

  getPageList(page, limit, courseId) {
    return request({
      url: `/eduservice/comment/${page}/${limit}`,
      method: 'get',
      params: {courseId}
    })
  },
  addComment(comment) {
    return request({
      url: `/eduservice/comment/auth/save`,
      method: 'post',
      data: comment
    })
  }
}
```

## 2、在课程详情页面，调用方法 _id.vue

```
import comment from '@/api/commonedu'

<script>
import course from '@/api/course'
import comment from '@/api/commonedu'
export default {

     //和页面异步开始的
  asyncData({ params, error }) {
    return {courseId: params.id}

```

```
  },
  data() {
    return {
      data:{},
      page:1,
      limit:4,
      total:10,
      comment:{
        content:'',
        courseId:''
      },
      courseInfo:{},
      chapterVideoList:[],
      isbuyCourse:false
    }
  },
  created() {
    this.initCourseInfo()
    this.initComment()
  },
  methods:{
    //获取课程详情
    initCourseInfo() {
      course.getCourseInfo(this.courseId)
            .then(response => {
              this.courseInfo=response.data.data.courseFrontInfo
              this.chapterVideoList=response.data.data.chapterVideoList
              this.isbuyCourse=response.data.data.isbuyCourse
            })
    },

    initComment(){
       comment.getPageList(this.page, this.limit, this.courseId).then(response =>
{
           this.data = response.data.data
       })
    },
    addComment(){
        this.comment.courseId = this.courseId
        this.comment.teacherId = this.courseInfo.teacherId
        comment.addComment(this.comment).then(response => {
            if(response.data.success){
                this.comment.content = ''
```

```
                this.initComment()
            }
        })
    },
    gotoPage(page){
          comment.getPageList(page, this.limit,this.courseId).then(response => {
              this.data = response.data.data
          })
      }
  }

};
</script>
```

## 3、在课程详情页面 _id.vue显示评论

```
<div class="mt50 commentHtml"><div>
      <h6 class="c-c-content c-infor-title" id="i-art-comment">
        <span class="commentTitle">课程评论</span>
      </h6>
      <section class="lh-bj-list pr mt20 replyhtml">
        <ul>
          <li class="unBr">
            <aside class="noter-pic">
              <img width="50" height="50" class="picImg" src="~/assets/img/avatar-boy.gif">
              </aside>
            <div class="of">
              <section class="n-reply-wrap">
                <fieldset>
                  <textarea name="" v-model="comment.content" placeholder="输入您要评论的文字" id="commentContent"></textarea>
                </fieldset>
                <p class="of mt5 tar pl10 pr10">
                  <span class="fl "><tt class="c-red commentContentmeg" style="display: none;"></tt></span>
                  <input type="button" @click="addComment()" value="回复" class="lh-reply-btn">
                </p>
              </section>
```

```
            </div>
          </li>
        </ul>
      </section>
      <section class="">
          <section class="question-list lh-bj-list pr">
            <ul class="pr10">
              <li v-for="(comment,index) in data.items" v-bind:key="index">
                  <aside class="noter-pic">
                    <img width="50" height="50" class="picImg"
:src="comment.avatar">
                  </aside>
                  <div class="of">
                    <span class="fl">
                    <font class="fsize12 c-blue">
                      {{comment.nickname}}</font>
                    <font class="fsize12 c-999 ml5">评论: </font></span>
                  </div>
                  <div class="noter-txt mt5">
                    <p>{{comment.content}}</p>
                  </div>
                  <div class="of mt5">
                    <span class="fr"><font class="fsize12 c-999
ml5">{{comment.gmtCreate}}</font></span>
                  </div>
                </li>

              </ul>
          </section>
        </section>

        <!-- 公共分页 开始 -->
        <div class="paging">
            <!-- undisable这个class是否存在, 取决于数据属性hasPrevious -->
            <a
            :class="{undisable: !data.hasPrevious}"
            href="#"
            title="首页"
            @click.prevent="gotoPage(1)">首</a>
            <a
            :class="{undisable: !data.hasPrevious}"
            href="#"
            title="前一页"
```

```
            @click.prevent="gotoPage(data.current-1)">&lt;</a>
            <a
            v-for="page in data.pages"
            :key="page"
            :class="{current: data.current == page, undisable: data.current ==
page}"
            :title="'第'+page+'页'"
            href="#"
            @click.prevent="gotoPage(page)">{{ page }}</a>
            <a
            :class="{undisable: !data.hasNext}"
            href="#"
            title="后一页"
            @click.prevent="gotoPage(data.current+1)">&gt;</a>
            <a
            :class="{undisable: !data.hasNext}"
            href="#"
            title="末页"
            @click.prevent="gotoPage(data.pages)">末</a>
            <div class="clear"/>
        </div>
        <!-- 公共分页 结束 -->
      </div>
    </div>
```

# 一、课程支付需求描述

## 1、课程支付说明

（1）课程分为免费课程和付费课程，如果是免费课程可以直接观看，如果是付费观看的课程，用户需下单支付后才可以观看

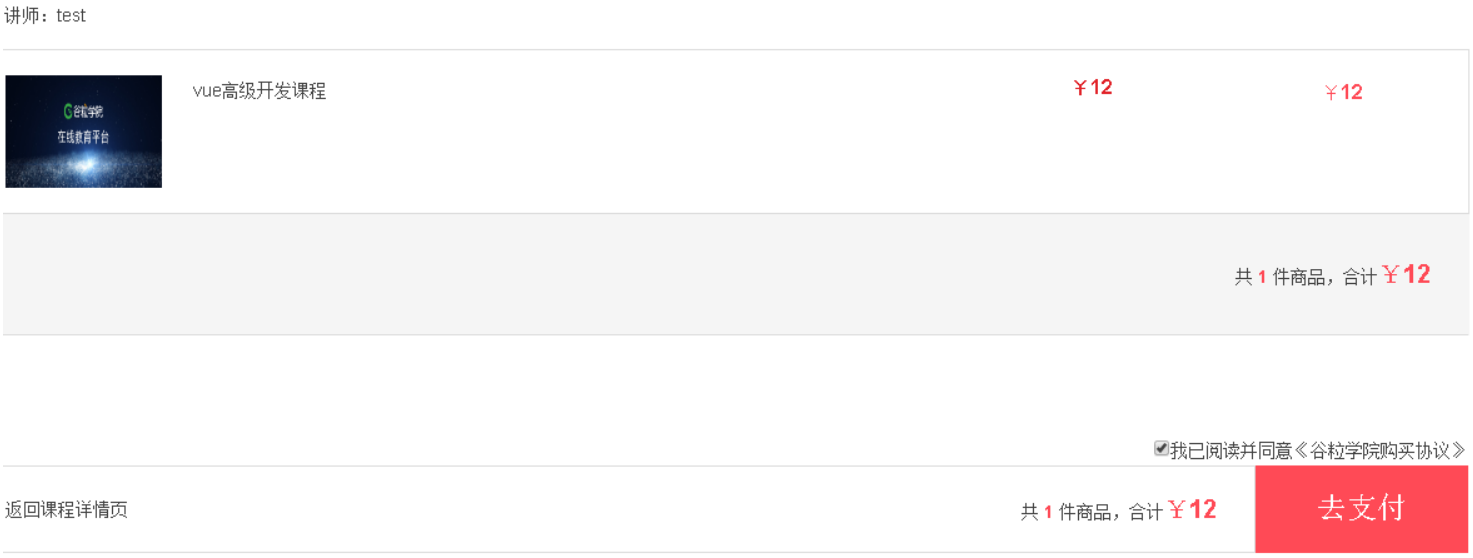

（2）如果是免费课程，在用户选择课程，进入到课程详情页面时候，直接显示“立即观看”，用户点击立即观看，可以切换到播放列表进行视频播放


## 2

# 、付费课程流程

（**1**）如果是付费课程，在用户选择课程，进入到课程详情页面时候，会显示 **“立即购买”**


（**2**）点击**“立即购买”**，会生成课程的订单，跳转到订单页面



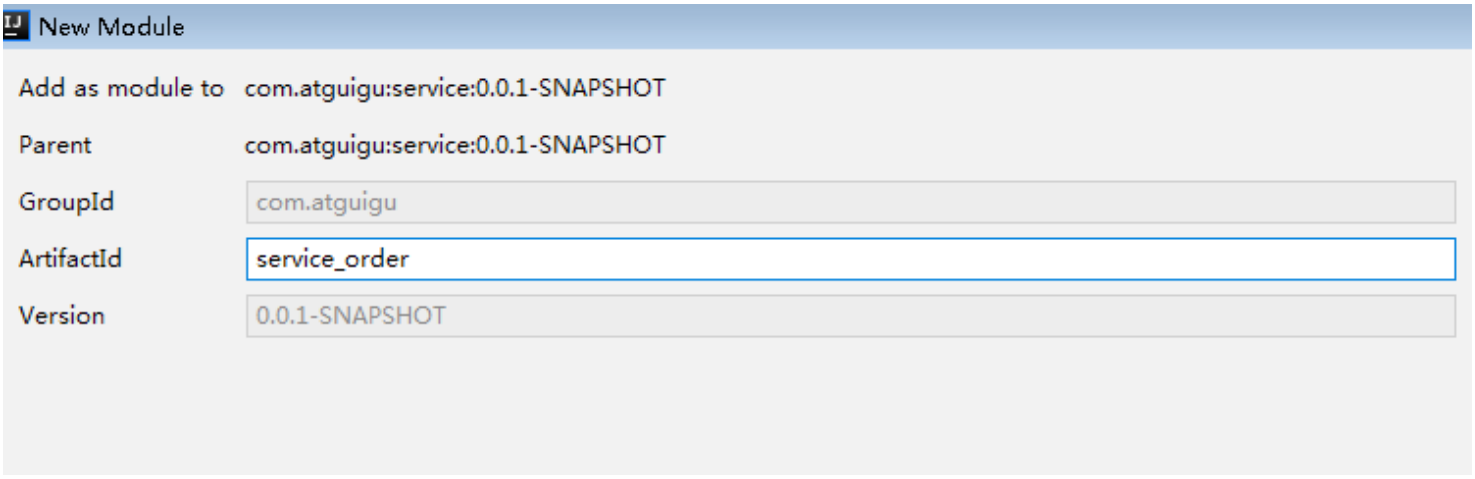

（**3**）点击**“去支付”**，会跳转到支付页面，生成微信扫描的二维码

订单提交成功，请您及时付款！订单号：20200109202147004


（**4**）使用微信扫描支付后，会跳转回到课程详情页面，同时显示**“**立即观看**”**

# 一、创建支付模块和准备

## 1、在service模块下创建子模块service_order

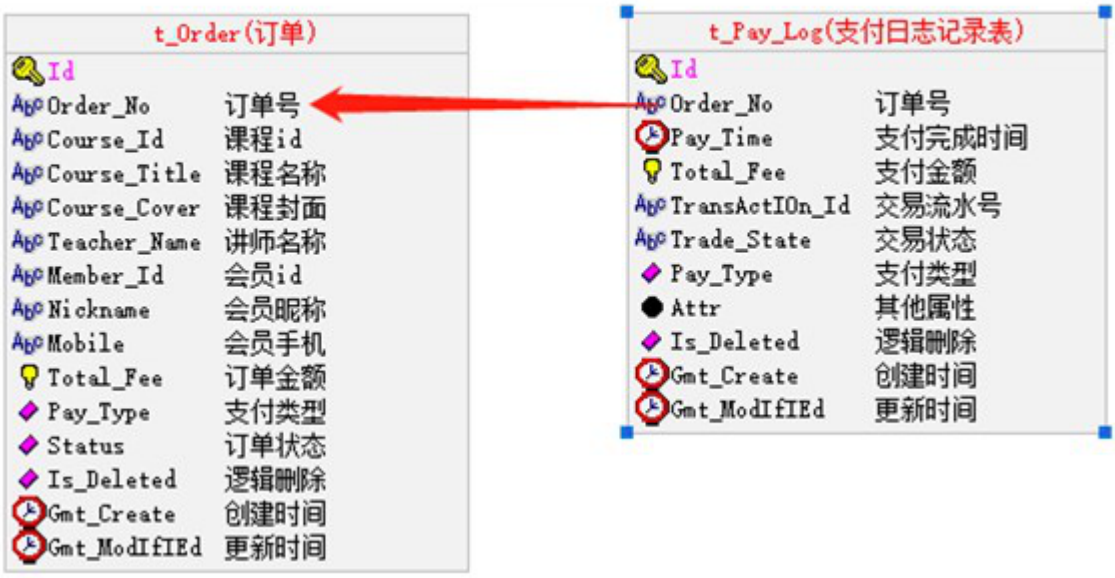



## 2、在service_order模块中引入依赖

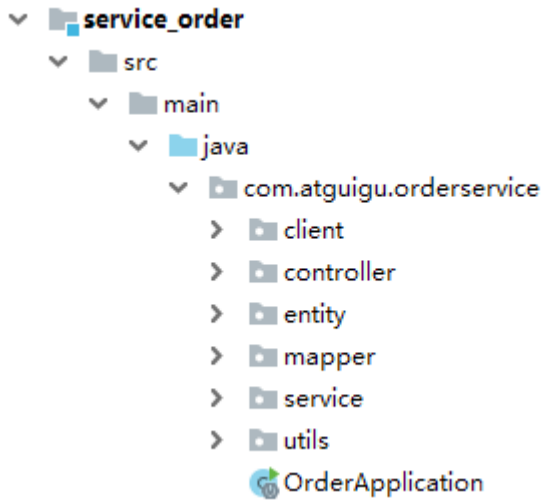
```xml
<dependencies>
    <dependency>
        <groupId>com.github.wxpay</groupId>
        <artifactId>wxpay-sdk</artifactId>
        <version>0.0.3</version>
    </dependency>

    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>fastjson</artifactId>
    </dependency>
</dependencies>
```

## 3、创建支付相关的表

导入 guli_order.sql




## 4、使用代码生成器生成相关代码




## 5、编写application.properties配置文件

```
# 服务端口
server.port=8007
# 服务名
spring.application.name=service-order

# mysql数据库连接
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/guli?serverTimezone=GMT%2B8
spring.datasource.username=root
spring.datasource.password=root

#返回json的全局时间格式
spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
```

```
spring.jackson.time-zone=GMT+8

#配置mapper xml文件的路径
mybatis-plus.mapper-locations=classpath:com/atguigu/orderservice/mapper/xml/*.xml

#mybatis日志
mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl

# nacos服务地址
spring.cloud.nacos.discovery.server-addr=127.0.0.1:8848

#开启熔断机制
feign.hystrix.enabled=true
# 设置hystrix超时时间，默认1000ms
hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds=3000
```

# 二、开发创建订单接口

## 1、编写订单controller

```
@RestController
@RequestMapping("/orderservice/order")
@CrossOrigin
public class TOrderController {

    @Autowired
    private TOrderService orderService;
    //根据课程id和用户id创建订单，返回订单id
    @PostMapping("createOrder/{courseId}")
    public R save(@PathVariable String courseId, HttpServletRequest request) {
        String orderId = orderService.saveOrder(courseId,
JwtUtils.getMemberIdByJwtToken(request));
        return R.ok().data("orderId", orderId);
    }
}
```

## 2、在service_edu创建接口

（1）实现根据课程id获取课程信息，返回课程信息对象

```
//根据课程id查询课程信息
@GetMapping("getDto/{courseId}")
public com.atguigu.commonutils.vo.CourseInfoForm getCourseInfoDto(@PathVariable
String courseId) {
        CourseInfoForm courseInfoForm = courseService.getCourseInfo(courseId);
        com.atguigu.commonutils.vo.CourseInfoForm courseInfo = new
com.atguigu.commonutils.vo.CourseInfoForm();
        BeanUtils.copyProperties(courseInfoForm,courseInfo);
        return courseInfo;
}
```

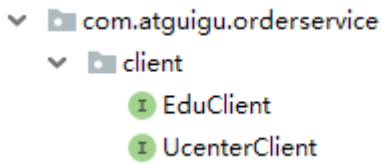
## 3、在service_ucenter创建接口

（1）实现用户id获取用户信息，返回用户信息对象

```
//根据token字符串获取用户信息
@PostMapping("getInfoUc/{id}")
public com.atguigu.commonutils.vo.UcenterMember getInfo(@PathVariable String id) {
        //根据用户id获取用户信息
        UcenterMember ucenterMember = memberService.getById(id);
        com.atguigu.commonutils.vo.UcenterMember memeber = new
com.atguigu.commonutils.vo.UcenterMember();
        BeanUtils.copyProperties(ucenterMember,memeber);
        return memeber;
}
```

## 4、编写订单service

（1）在service_order模块创建接口，实现远程调用




### EduClient

```
@Component
```

```
@FeignClient("service-edu")
public interface EduClient {
    //根据课程id查询课程信息
    @GetMapping("/eduservice/course/getDto/{courseId}")
    public com.atguigu.commonutils.vo.CourseInfoForm
getCourseInfoDto(@PathVariable("courseId") String courseId);
}
```

### UcenterClient

```
@Component
@FeignClient("service-ucenter")
public interface UcenterClient {
    //根据课程id查询课程信息
    @PostMapping("/ucenterservice/member/getInfoUc/{id}")
    public com.atguigu.commonutils.vo.UcenterMember getInfo(@PathVariable("id")
String id);
}
```

## （2）在service_order模块编写创建订单service

```
@Service
public class TOrderServiceImpl extends ServiceImpl<TOrderMapper, TOrder>
implements TOrderService {

    @Autowired
    private EduClient eduClient;

    @Autowired
    private UcenterClient ucenterClient;

    //创建订单
    @Override
    public String saveOrder(String courseId, String memberId) {
        //远程调用课程服务，根据课程id获取课程信息
        CourseInfoForm courseDto = eduClient.getCourseInfoDto(courseId);

```

```
        //远程调用用户服务，根据用户id获取用户信息
        UcenterMember ucenterMember = ucenterClient.getInfo(memberId);

        //创建订单
        TOrder order = new TOrder();
        order.setOrderNo(OrderNoUtil.getOrderNo());
        order.setCourseId(courseId);
        order.setCourseTitle(courseDto.getTitle());
        order.setCourseCover(courseDto.getCover());
        order.setTeacherName("test");
        order.setTotalFee(courseDto.getPrice());
        order.setMemberId(memberId);
        order.setMobile(ucenterMember.getMobile());
        order.setNickname(ucenterMember.getNickname());
        order.setStatus(0);
        order.setPayType(1);
        baseMapper.insert(order);

        return order.getOrderNo();
    }
}
```

# 三、开发获取订单接口

## 1、在订单controller创建根据id获取订单信息接口

```
    @GetMapping("getOrder/{orderId}")
    public R get(@PathVariable String orderId) {
        QueryWrapper<TOrder> wrapper = new QueryWrapper<>();
        wrapper.eq("order_no",orderId);
        TOrder order = orderService.getOne(wrapper);
        return R.ok().data("item", order);
    }
```

# 一、生成微信支付二维码

## 1、编写controller

```
@RestController
@RequestMapping("/orderservice/log")
@CrossOrigin
public class PayLogController {

    @Autowired
    private PayLogService payService;
    /**
     * 生成二维码
     *
     * @return
     */
    @GetMapping("/createNative/{orderNo}")
    public R createNative(@PathVariable String orderNo) {
        Map map = payService.createNative(orderNo);
        return R.ok().data(map);
    }
}
```

## 2、编写service

```
@Service
public class PayLogServiceImpl extends ServiceImpl<PayLogMapper, PayLog>
implements PayLogService {

    @Autowired
    private TOrderService orderService;

    @Override
    public Map createNative(String orderNo) {
        try {
            //根据订单id获取订单信息
            QueryWrapper<TOrder> wrapper = new QueryWrapper<>();
```

```
            wrapper.eq("order_no",orderNo);
            TOrder order = orderService.getOne(wrapper);

            Map m = new HashMap();
            //1、设置支付参数
            m.put("appid", "wx74862e0dfcf69954");
            m.put("mch_id", "1558950191");
            m.put("nonce_str", WXPayUtil.generateNonceStr());
            m.put("body", order.getCourseTitle());
            m.put("out_trade_no", orderNo);
            m.put("total_fee", order.getTotalFee().multiply(new
BigDecimal("100")).longValue()+"");
            m.put("spbill_create_ip", "127.0.0.1");
            m.put("notify_url",
"http://guli.shop/api/order/weixinPay/weixinNotify\n");
            m.put("trade_type", "NATIVE");

            //2、HTTPClient来根据URL访问第三方接口并且传递参数
            HttpClient client = new
HttpClient("https://api.mch.weixin.qq.com/pay/unifiedorder");

            //client设置参数
            client.setXmlParam(WXPayUtil.generateSignedXml(m,
"T6m9iK73b0kn9g5v426MKfHQH7X8rKwb"));
            client.setHttps(true);
            client.post();
            //3、返回第三方的数据
            String xml = client.getContent();
            Map<String, String> resultMap = WXPayUtil.xmlToMap(xml);
            //4、封装返回结果集

            Map map = new HashMap<>();
            map.put("out_trade_no", orderNo);
            map.put("course_id", order.getCourseId());
            map.put("total_fee", order.getTotalFee());
            map.put("result_code", resultMap.get("result_code"));
            map.put("code_url", resultMap.get("code_url"));

            //微信支付二维码2小时过期，可采取2小时未支付取消订单
            //redisTemplate.opsForValue().set(orderNo, map, 120,
TimeUnit.MINUTES);
            return map;
        } catch (Exception e) {
            e.printStackTrace();
```

```
            return new HashMap<>();
        }
    }
}
```

## 二、获取支付状态接口

### 1、编写controller

```
@GetMapping("/queryPayStatus/{orderNo}")
public R queryPayStatus(@PathVariable String orderNo) {
    //调用查询接口
    Map<String, String> map = payService.queryPayStatus(orderNo);
    if (map == null) {//出错
        return R.error().message("支付出错");
    }
    if (map.get("trade_state").equals("SUCCESS")) {//如果成功
        //更改订单状态
        payService.updateOrderStatus(map);
        return R.ok().message("支付成功");
    }

    return R.ok().code(25000).message("支付中");
}
```

### 2、编写service，更新订单状态

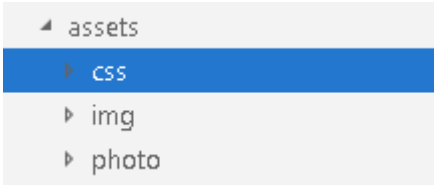
```
@Override
public void updateOrderStatus(Map<String, String> map) {
    //获取订单id
    String orderNo = map.get("out_trade_no");
    //根据订单id查询订单信息
    QueryWrapper<TOrder> wrapper = new QueryWrapper<>();
    wrapper.eq("order_no",orderNo);
    TOrder order = orderService.getOne(wrapper);

    if(order.getStatus().intValue() == 1) return;
```

```
    order.setStatus(1);
    orderService.updateById(order);

    //记录支付日志
    PayLog payLog=new PayLog();
    payLog.setOrderNo(order.getOrderNo());//支付订单号
    payLog.setPayTime(new Date());
    payLog.setPayType(1);//支付类型
    payLog.setTotalFee(order.getTotalFee());//总金额(分)
    payLog.setTradeState(map.get("trade_state"));//支付状态
    payLog.setTransactionId(map.get("transaction_id"));
    payLog.setAttr(JSONObject.toJSONString(map));
    baseMapper.insert(payLog);//插入到支付日志表
}


    @Override
    public Map queryPayStatus(String orderNo) {
        try {
            //1、封装参数
            Map m = new HashMap<>();
            m.put("appid", "wx74862e0dfcf69954");
            m.put("mch_id", "1558950191");
            m.put("out_trade_no", orderNo);
            m.put("nonce_str", WXPayUtil.generateNonceStr());

            //2、设置请求
            HttpClient client = new
HttpClient("https://api.mch.weixin.qq.com/pay/orderquery");
            client.setXmlParam(WXPayUtil.generateSignedXml(m,
"T6m9iK73b0kn9g5v426MKfHQH7X8rKwb"));
            client.setHttps(true);
            client.post();
            //3、返回第三方的数据
            String xml = client.getContent();
            Map<String, String> resultMap = WXPayUtil.xmlToMap(xml);
            //6、转成Map
            //7、返回
            return resultMap;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
```

```
52      }
```

# 一、页面样式修改

## 1、复制样式文件到**assets**




## 2、修改**default.vue**页面

```
import '~/assets/css/reset.css'
import '~/assets/css/theme.css'
import '~/assets/css/global.css'
import '~/assets/css/web.css'
import '~/assets/css/base.css'
import '~/assets/css/activity_tab.css'
import '~/assets/css/bottom_rec.css'
import '~/assets/css/nice_select.css'
import '~/assets/css/order.css'
import '~/assets/css/swiper-3.3.1.min.css'
import "~/assets/css/pages-weixinpay.css"
```

# 二、课程支付前端

## 1、在**api**文件夹下创建**order.js**文件

```
import request from '@/utils/request'

export default {

    //1、创建订单
    createOrder(cid) {
        return request({
            url: '/orderservice/order/createOrder/'+cid,
            method: 'post'
        })
    },
```

```
    //2、根据id获取订单
    getById(cid) {
        return request({
            url: '/orderservice/order/getOrder/'+cid,
            method: 'get'
        })
    },
    //3、生成微信支付二维码
    createNative(cid) {
        return request({
            url: '/orderservice/log/createNative/'+cid,
            method: 'get'
        })
    },
    //4、根据id获取订单支付状态
    queryPayStatus(cid) {
        return request({
            url: '/orderservice/log/queryPayStatus/'+cid,
            method: 'get'
        })
    }
}
```

## 2、在课程详情页面中添加创建订单方法

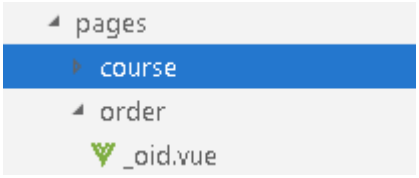
在“立即购买”位置添加事件

```
@click="createOrder()"
```

```
methods:{
    //根据课程id，调用接口方法生成订单
    createOrder(){
        order.createOrder(this.courseId).then(response => {
            if(response.data.success){
                //订单创建成功，跳转到订单页面
                this.$router.push({ path: '/order/'+ response.data.data.orderId })
            }
        })
    },
}
```

## 3、创建订单页面，显示订单信息

在**pages**下面创建**order**文件夹，创建**_oid.vue**页面




在**_oid.vue**页面调用方法，获取订单信息

（**1**）页面部分

```
<template>
  <div class="Page Confirm">
    <div class="Title">
      <h1 class="fl f18">订单确认</h1>
      <img src="~/assets/img/cart_setp2.png" class="fr">
      <div class="clear"></div>
    </div>
    <form name="flowForm" id="flowForm" method="post" action="">
      <table class="GoodList">
        <tbody>
        <tr>
          <th class="name">商品</th>
          <th class="price">原价</th>
          <th class="priceNew">价格</th>
        </tr>
        </tbody>
        <tbody>
        <!-- <tr>
          <td colspan="3" class="Title red f18 fb"><p>限时折扣</p></td>
        </tr> -->
        <tr>
          <td colspan="3" class="teacher">讲师：{{order.teacherName}}</td>
        </tr>
        <tr class="good">
          <td class="name First">
            <a target="_blank"
:href="'https://localhost:3000/course/'+order.courseId">
              <img :src="order.courseCover"></a>
```

```
            <div class="goodInfo">
              <input type="hidden" class="ids ids_14502" value="14502">
              <a target="_blank" :href="'https://localhost:3000/course/'+
order.courseId">{{order.courseTitle}}</a>
            </div>
          </td>
          <td class="price">
            <p>￥<strong>{{order.totalFee}}</strong></p>
            <!-- <span class="discName red">限时8折</span> -->
          </td>
          <td class="red priceNew Last">￥<strong>{{order.totalFee}}</strong></td>
        </tr>
        <tr>
          <td class="Billing tr" colspan="3">
            <div class="tr">
              <p>共 <strong class="red">1</strong> 件商品，合计<span
                class="red f20">￥<strong>{{order.totalFee}}</strong></span></p>
            </div>
          </td>
        </tr>
        </tbody>
      </table>
      <div class="Finish">
        <div class="fr" id="AgreeDiv">

          <label for="Agree"><p class="on"><input type="checkbox"
checked="checked">我已阅读并同意<a href="javascript:" target="_blank">《谷粒学院购买
协议》</a></p></label>
        </div>
        <div class="clear"></div>
        <div class="Main fl">
          <div class="fl">
            <a :href="'/course/'+order.courseId">返回课程详情页</a>
          </div>
          <div class="fr">
            <p>共 <strong class="red">1</strong> 件商品，合计<span class="red
f20">￥<strong
              id="AllPrice">{{order.totalFee}}</strong></span></p>
          </div>
        </div>
        <input name="score" value="0" type="hidden" id="usedScore">
        <button class="fr redb" type="button" id="submitPay" @click="toPay()">去支
付</button>
        <div class="clear"></div>
```

```
      </div>
    </form>
  </div>
</template>
```

**（2）调用部分**

```
<script>
import orderApi from '@/api/order'
export default {
  //根据订单id获取订单信息
  asyncData({params, error}) {
      return orderApi.getById(params.oid).then(response => {
        return {
            order: response.data.data.item
            }
      })
  },
  methods: {
    //点击去支付，跳转到支付页面
    toPay() {
      this.$router.push({path: '/pay/' + this.order.orderNo})
    }
  }
}
</script>
```

# 4、创建支付页面，生成二维码完成支付

（1）页面部分

```
<template>
  <div class="cart py-container">
    <!--主内容-->
    <div class="checkout py-container  pay">
      <div class="checkout-tit">
        <h4 class="fl tit-txt"><span class="success-icon"></span><span
class="success-info">订单提交成功，请您及时付款！订单
```

```
号: {{payObj.out_trade_no}}</span>
        </h4>
        <span class="fr"><em class="sui-lead">应付金额: </em><em class="orange
money">￥{{payObj.total_fee}}</em></span>
        <div class="clearfix"></div>
      </div>
      <div class="checkout-steps">
        <div class="fl weixin">微信支付</div>
        <div class="fl sao">
          <p class="red">请使用微信扫一扫。</p>
          <div class="fl code">
            <!-- <img id="qrious" src="~/assets/img/erweima.png" alt=""> -->
            <!-- <qriously value="weixin://wxpay/bizpayurl?pr=R7tnDpZ"
:size="338"/> -->
            <qriously :value="payObj.code_url" :size="338"/>
            <div class="saosao">
              <p>请使用微信扫一扫</p>
              <p>扫描二维码支付</p>
            </div>

          </div>

        </div>
        <div class="clearfix"></div>
        <!-- <p><a href="pay.html" target="_blank">> 其他支付方式</a></p> -->

      </div>
    </div>
  </div>
</template>
```

**(2) 调用部分**

```
<script>
  import orderApi from '@/api/course'

  export default {
    //根据订单id生成微信支付二维码
    asyncData({params, error}) {
      return orderApi.createNative(params.pid).then(response => {
```

```
        return {
          payObj: response.data.data
        }
      })
    },
    data() {
      return {
        timer: null,  // 定时器名称
        initQCode: '',
        timer1:''
      }
    },
    mounted() {
      //在页面渲染之后执行
      //每隔三秒，去查询一次支付状态
      this.timer1 = setInterval(() => {
        this.queryPayStatus(this.payObj.out_trade_no)
      }, 3000);
    },
    methods: {
      //查询支付状态的方法
      queryPayStatus(out_trade_no) {
        orderApi.queryPayStatus(out_trade_no).then(response => {
          if (response.data.success) {
              //如果支付成功，清除定时器
              clearInterval(this.timer1)
            this.$message({
              type: 'success',
              message: '支付成功!'
            })
            //跳转到课程详情页面观看视频
            this.$router.push({path: '/course/' + this.payObj.course_id})
          }
        })
      }
    }
  }
</script>
```

# 一、修改课程详情接口

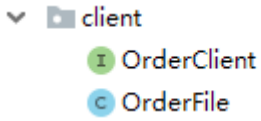
## 1、在service_order模块添加接口

根据用户id和课程id查询订单信息

```
@GetMapping("isBuyCourse/{memberid}/{id}")
public boolean isBuyCourse(@PathVariable String memberid,
                           @PathVariable String id) {
    //订单状态是1表示支付成功
    int count = orderService.count(new QueryWrapper<TOrder>().eq("member_id",
memberid).eq("course_id", id).eq("status", 1));
    if(count>0) {
        return true;
    } else {
        return false;
    }
}
```

## 2、在service_edu模块课程详情接口远程调用

（1）创建OrderClient接口




```
@Component
@FeignClient(value = "service-order", fallback = OrderFile.class)
public interface OrderClient {
    //查询订单信息
    @GetMapping("/orderservice/order/isBuyCourse/{memberid}/{id}")
    public boolean isBuyCourse(@PathVariable("memberid") String memberid,
@PathVariable("id") String id);
}
```

（2）在课程详情接口调用

```
//根据id查询课程详情信息
@GetMapping("getCourseInfo/{id}")
public R getCourseInfo(@PathVariable String id, HttpServletRequest request) {
    //课程查询课程基本信息
    CourseFrontInfo courseFrontInfo = courseService.getFrontCourseInfo(id);
    //查询课程里面大纲数据
    List<ChapterVo> chapterVideoList = chapterService.getChapterVideoById(id);

    //远程调用，判断课程是否被购买
    boolean buyCourse = orderClient.isBuyCourse(JwtUtils.getMemberIdByJwtToken(request), id);

    return R.ok().data("courseFrontInfo",courseFrontInfo).data("chapterVideoList",chapterVideoList).data("isbuy",buyCourse);
}
```

# 二、修改课程详情页面

## 1、页面内容修改

```
<section v-if="isbuy || Number(courseInfo.price)===0" class="c-attr-mt">
            <a href="#" title="立即观看" class="comm-btn c-btn-3" >立即观看</a>
</section>
<section v-else class="c-attr-mt">
    <a href="#" title="立即购买" class="comm-btn c-btn-3" @click="createOrder()">立即购买</a>
</section>
```

## 2、调用方法修改

```
<script>
import course from '@/api/course'
export default {
  // asyncData({ params, error }) {
  //   return course.getCourseInfo(params.id)
```

```
  //     .then(response => {
  //       return {
  //         courseInfo: response.data.data.courseFrontInfo,
  //         chapterVideoList: response.data.data.chapterVideoList,
  //         isbuy: response.data.data.isbuy,
  //         courseId:params.id
  //       }
  //     })
  // },
   //和页面异步开始的
  asyncData({ params, error }) {
    return {courseId: params.id}

  },
  data() {
    return {
      courseInfo: {},
          chapterVideoList: [],
          isbuy: false,
    }
  },
  created() {
    this.initCourseInfo()
  },
  methods:{
    initCourseInfo() {
      course.getCourseInfo(this.courseId)
        .then(response => {
          this.courseInfo=response.data.data.courseFrontInfo,
          this.chapterVideoList=response.data.data.chapterVideoList,
          this.isbuy=response.data.data.isbuy
        })
    },
    createOrder(){
      course.createOrder(this.courseId).then(response => {
        if(response.data.success){
            this.$router.push({ path: '/order/'+ response.data.data.orderId })
        }
      })
    }
  }
};
</script>
```

# 一、数据库设计

## 1、数据库

guli_statistics

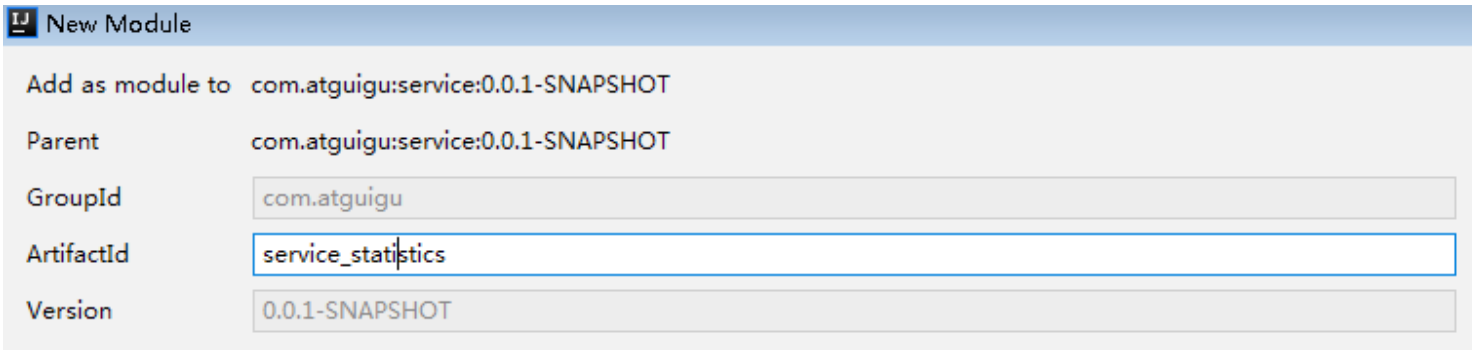
## 2、数据表

```
guli_statistics.sql
```

# 二、创建微服务

## 1、在service模块下创建子模块

service_statistics



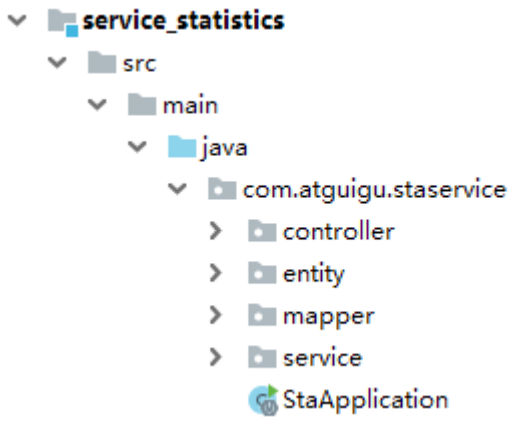

## 2、application.properties

resources目录下创建文件

```
# 服务端口
server.port=8008
# 服务名
spring.application.name=service-statistics

# mysql数据库连接
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

```
spring.datasource.url=jdbc:mysql://localhost:3306/guli?serverTimezone=GMT%2B8
spring.datasource.username=root
spring.datasource.password=root

#返回json的全局时间格式
spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
spring.jackson.time-zone=GMT+8

#配置mapper xml文件的路径
mybatis-plus.mapper-locations=classpath:com/atguigu/staservice/mapper/xml/*.xml

#mybatis日志
mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl

# nacos服务地址
spring.cloud.nacos.discovery.server-addr=127.0.0.1:8848

#开启熔断机制
feign.hystrix.enabled=true
# 设置hystrix超时时间，默认1000ms
hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds=3000
```

# 3、MP代码生成器生成代码




# 4、创建SpringBoot启动类

```
@SpringBootApplication
@MapperScan("com.atguigu.staservice.mapper")
```

```
@ComponentScan("com.atguigu")
@EnableDiscoveryClient
@EnableFeignClients
public class StaApplication {
    public static void main(String[] args) {
        SpringApplication.run(StaApplication.class, args);
    }
}
```

# 三、实现服务调用

## 1、在service_ucenter模块创建接口，统计某一天的注册人数

### controller

```
@GetMapping(value = "countregister/{day}")
public R registerCount(
        @PathVariable String day){
    Integer count = memberService.countRegisterByDay(day);
    return R.ok().data("countRegister", count);
}
```

### service

```
@Override
public Integer countRegisterByDay(String day) {
    return baseMapper.selectRegisterCount(day);
}
```

### mapper

```
<select id="selectRegisterCount" resultType="java.lang.Integer">
    SELECT COUNT(1)
    FROM ucenter_member
```

```
    WHERE DATE(gmt_create) = #{value}
</select>
```

# 2、在service_statistics模块创建远程调用接口

创建client包和UcenterClient接口

```
@Component
@FeignClient("service-ucenter")
public interface UcenterClient {

    @GetMapping(value = "/ucenterservice/member/countregister/{day}")
    public R registerCount(@PathVariable("day") String day);
}
```

# 3、在service_statistics模块调用微服务

## service

```
@Service
public class StatisticsDailyServiceImpl extends ServiceImpl<StatisticsDailyMapper,
StatisticsDaily> implements StatisticsDailyService {

    @Autowired
    private UcenterClient ucenterClient;

    @Override
    public void createStatisticsByDay(String day) {
        //删除已存在的统计对象
        QueryWrapper<StatisticsDaily> dayQueryWrapper = new QueryWrapper<>();
        dayQueryWrapper.eq("date_calculated", day);
        baseMapper.delete(dayQueryWrapper);


        //获取统计信息
        Integer registerNum = (Integer)
ucenterClient.registerCount(day).getData().get("countRegister");
        Integer loginNum = RandomUtils.nextInt(100, 200);//TODO
```

```
        Integer videoViewNum = RandomUtils.nextInt(100, 200);//TODO
        Integer courseNum = RandomUtils.nextInt(100, 200);//TODO

        //创建统计对象
        StatisticsDaily daily = new StatisticsDaily();
        daily.setRegisterNum(registerNum);
        daily.setLoginNum(loginNum);
        daily.setVideoViewNum(videoViewNum);
        daily.setCourseNum(courseNum);
        daily.setDateCalculated(day);

        baseMapper.insert(daily);
    }
}
```

### controller

```
@PostMapping("{day}")
public R createStatisticsByDate(@PathVariable String day) {
    dailyService.createStatisticsByDay(day);
    return R.ok();
}
```

# 四、添加定时任务

## 1、创建定时任务类，使用cron表达式

复制日期工具类

```
@Component
public class ScheduledTask {

    @Autowired
    private StatisticsDailyService dailyService;

    /**
     * 测试
```

```
     * 每天七点到二十三点每五秒执行一次
     */
    @Scheduled(cron = "0/5 * * * * ?")
    public void task1() {
        System.out.println("*********++++++++++++++*****执行了");
    }

    /**
     * 每天凌晨1点执行定时
     */
    @Scheduled(cron = "0 0 1 * * ?")
    public void task2() {
        //获取上一天的日期
        String day = DateUtil.formatDate(DateUtil.addDays(new Date(), -1));
        dailyService.createStatisticsByDay(day);

    }
}
```

## 2、在启动类上添加注解

```
@EnableScheduling
public class OrdersApplication {
```

## 3、在线生成cron表达式

http://cron.qqe2.com/

# 一、nginx配置

```
location ~ /staservice/ {
    proxy_pass http://localhost:8008;
}
```

# 二、前端页面实现

## 1、创建api

创建src/api/sta.js

```
import request from '@/utils/request'

const api_name = '/admin/statistics/daily'
export default {

  createStatistics(day) {
    return request({
      url: `${api_name}/${day}`,
      method: 'post'
    })
  }
}
```

## 2、增加路由

src/router/index.js

```
{
  path: '/statistics/daily',
  component: Layout,
  redirect: '/statistics/daily/create',
  name: 'Statistics',
```

```
  meta: { title: '统计分析', icon: 'chart' },
  children: [
    {
      path: 'create',
      name: 'StatisticsDailyCreate',
      component: () => import('@/views/statistics/daily/create'),
      meta: { title: '生成统计' }
    }
  ]
},
```

# 3、创建组件

src/views/statistics/daily/create.vue

模板部分

```
<template>
  <div class="app-container">
    <!--表单-->
    <el-form :inline="true" class="demo-form-inline">

      <el-form-item label="日期">
        <el-date-picker
          v-model="day"
          type="date"
          placeholder="选择要统计的日期"
          value-format="yyyy-MM-dd" />
      </el-form-item>

      <el-button
        :disabled="btnDisabled"
        type="primary"
        @click="create()">生成</el-button>
    </el-form>

  </div>
</template>
```

script部分

```
<script>
import daily from '@/api/sta'

export default {
  data() {
    return {
      day: '',
      btnDisabled: false
    }
  },

  methods: {
    create() {
      this.btnDisabled = true
      daily.createStatistics(this.day).then(response => {
        this.btnDisabled = false
        this.$message({
          type: 'success',
          message: '生成成功'
        })
      })
    }
  }
}
</script>
```

# 一、ECharts

## 1、简介

ECharts是百度的一个项目，后来百度把Echart捐给apache，用于图表展示，提供了常规的折线图、柱状图、散点图、饼图、K线图，用于统计的盒形图，用于地理数据可视化的地图、热力图、线图，用于关系数据可视化的关系图、treemap、旭日图，多维数据可视化的平行坐标，还有用于 BI 的漏斗图，仪表盘，并且支持图与图之间的混搭。

官方网站：https://echarts.baidu.com/

## 2、基本使用

入门参考：官网->文档->教程->5分钟上手ECharts

（1）创建html页面：柱图.html

（2）引入ECharts

```
<!-- 引入 ECharts 文件 -->
<script src="echarts.min.js"></script>
```

（3）定义图表区域

```
<!-- 为ECharts准备一个具备大小（宽高）的Dom -->
<div id="main" style="width: 600px;height:400px;"></div>
```

（4）渲染图表

```
<script type="text/javascript">
    // 基于准备好的dom，初始化echarts实例
    var myChart = echarts.init(document.getElementById('main'));

    // 指定图表的配置项和数据
    var option = {
        title: {
            text: 'ECharts 入门示例'
```

```
        },
        tooltip: {},
        legend: {
            data:['销量']
        },
        xAxis: {
            data: ["衬衫","羊毛衫","雪纺衫","裤子","高跟鞋","袜子"]
        },
        yAxis: {},
        series: [{
            name: '销量',
            type: 'bar',
            data: [5, 20, 36, 10, 10, 20]
        }]
    };

    // 使用刚指定的配置项和数据显示图表。
    myChart.setOption(option);
</script>
```

# 3、折线图

实例参考：官网->实例->官方实例 https://echarts.baidu.com/examples/

折线图.html

```
<script>
    var myChart = echarts.init(document.getElementById('main'));
    var option = {
        //x轴是类目轴（离散数据）,必须通过data设置类目数据
        xAxis: {
            type: 'category',
            data: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
        },
        //y轴是数据轴（连续数据）
        yAxis: {
            type: 'value'
        },
        //系列列表。每个系列通过 type 决定自己的图表类型
        series: [{
            //系列中的数据内容数组
```

```
            data: [820, 932, 901, 934, 1290, 1330, 1320],
            //折线图
            type: 'line'
        }]
    };
    myChart.setOption(option);

</script>
```

# 二、项目中集成ECharts

## 1、安装ECharts

```
npm install --save echarts@4.1.0
```

## 2、增加路由

src/router/index.js

在统计分析路由中增加子路由

```
{
    path: 'chart',
    name: 'StatisticsDayChart',
    component: () => import('@/views/statistics/daily/chart'),
    meta: { title: '统计图表' }
}
```

## 3、创建组件

src/views/statistics/daily/chart.vue

**模板**

```
<template>
  <div class="app-container">
```

```
    <!--表单-->
    <el-form :inline="true" class="demo-form-inline">

      <el-form-item>
        <el-select v-model="searchObj.type" clearable placeholder="请选择">
          <el-option label="学员登录数统计" value="login_num"/>
          <el-option label="学员注册数统计" value="register_num"/>
          <el-option label="课程播放数统计" value="video_view_num"/>
          <el-option label="每日课程数统计" value="course_num"/>
        </el-select>
      </el-form-item>

      <el-form-item>
        <el-date-picker
          v-model="searchObj.begin"
          type="date"
          placeholder="选择开始日期"
          value-format="yyyy-MM-dd" />
      </el-form-item>
      <el-form-item>
        <el-date-picker
          v-model="searchObj.end"
          type="date"
          placeholder="选择截止日期"
          value-format="yyyy-MM-dd" />
      </el-form-item>
      <el-button
        :disabled="btnDisabled"
        type="primary"
        icon="el-icon-search"
        @click="showChart()">查询</el-button>
    </el-form>

    <div class="chart-container">
      <div id="chart" class="chart" style="height:500px;width:100%" />
    </div>
  </div>
</template>
```

**js：暂时显示临时数据**

```
```

```vue
<script>
import echarts from 'echarts'

export default {
  data() {
    return {
      searchObj: {
        type: '',
        begin: '',
        end: ''
      },
      btnDisabled: false,
      chart: null,
      title: '',
      xData: [],
      yData: []
    }
  },
  methods: {
    showChart() {
      this.initChartData()

      this.setChart()
    },

    // 准备图表数据
    initChartData() {

    },

    // 设置图标参数
    setChart() {
      // 基于准备好的dom，初始化echarts实例
      this.chart = echarts.init(document.getElementById('chart'))
      // console.log(this.chart)

      // 指定图表的配置项和数据
      var option = {
        // x轴是类目轴（离散数据）,必须通过data设置类目数据
        xAxis: {
          type: 'category',
          data: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
        },
```

```
        // y轴是数据轴（连续数据）
        yAxis: {
          type: 'value'
        },
        // 系列列表。每个系列通过 type 决定自己的图表类型
        series: [{
          // 系列中的数据内容数组
          data: [820, 932, 901, 934, 1290, 1330, 1320],
          // 折线图
          type: 'line'
        }]
      }

      this.chart.setOption(option)
    }
  }
}
</script>
```

# 三、完成后端业务

## 1、controller

```
@GetMapping("show-chart/{begin}/{end}/{type}")
public R showChart(@PathVariable String begin,@PathVariable String
end,@PathVariable String type){
    Map<String, Object> map = dailyService.getChartData(begin, end, type);
    return R.ok().data(map);
}
```

## 2、service

### 接口

```
Map<String, Object> getChartData(String begin, String end, String type);
```

**实现**

```
@Override
public Map<String, Object> getChartData(String begin, String end, String type) {

    QueryWrapper<Daily> dayQueryWrapper = new QueryWrapper<>();
    dayQueryWrapper.select(type, "date_calculated");
    dayQueryWrapper.between("date_calculated", begin, end);

    List<Daily> dayList = baseMapper.selectList(dayQueryWrapper);

    Map<String, Object> map = new HashMap<>();
    List<Integer> dataList = new ArrayList<Integer>();
    List<String> dateList = new ArrayList<String>();
    map.put("dataList", dataList);
    map.put("dateList", dateList);


    for (int i = 0; i < dayList.size(); i++) {
        Daily daily = dayList.get(i);
        dateList.add(daily.getDateCalculated());
        switch (type) {
            case "register_num":
                dataList.add(daily.getRegisterNum());
                break;
            case "login_num":
                dataList.add(daily.getLoginNum());
                break;
            case "video_view_num":
                dataList.add(daily.getVideoViewNum());
                break;
            case "course_num":
                dataList.add(daily.getCourseNum());
                break;
            default:
                break;
        }
    }

    return map;
}
```

# 四、前后端整合

## 1、创建api

src/api/statistics/daily.js中添加方法

```
showChart(searchObj) {
    return request({
        url: `${api_name}/show-chart/${searchObj.begin}/${searchObj.end}/${searchObj.type}`,
        method: 'get'
    })
}
```

## 2、chart.vue中引入api模块

```
import daily from '@/api/statistics/daily'

......
```

## 3、修改initChartData方法

```
showChart() {
  this.initChartData()
  // this.setChart()//放在initChartData回调中执行
},

// 准备图表数据
initChartData() {
  daily.showChart(this.searchObj).then(response => {
    // 数据
    this.yData = response.data.dataList

    // 横轴时间
    this.xData = response.data.dateList
```

```
    // 当前统计类别
    switch (this.searchObj.type) {
      case 'register_num':
        this.title = '学员注册数统计'
        break
      case 'login_num':
        this.title = '学员登录数统计'
        break
      case 'video_view_num':
        this.title = '课程播放数统计'
        break
      case 'course_num':
        this.title = '每日课程数统计'
        break
    }

    this.setChart()
  })
},
```

## 4、修改options中的数据

```
xAxis: {
    type: 'category',
    data: this.xData//-------绑定数据
},
// y轴是数据轴（连续数据）
yAxis: {
    type: 'value'
},
// 系列列表。每个系列通过 type 决定自己的图表类型
series: [{
    // 系列中的数据内容数组
    data: this.yData,//-------绑定数据
    // 折线图
    type: 'line'
}],
```

# 五、样式调整

参考配置手册：https://echarts.baidu.com/option.html#title

## 1、显示标题

```
title: {
    text: this.title
},
```

## 2、x坐标轴触发提示

```
tooltip: {
    trigger: 'axis'
},
```

## 3、区域缩放

```
dataZoom: [{
  show: true,
  height: 30,
  xAxisIndex: [
    0
  ],
  bottom: 30,
  start: 10,
  end: 80,
  handleIcon: 'path://M306.1,413c0,2.2-1.8,4-4,4h-59.8c-2.2,0-4-1.8-4-4V200.8c0-2.2,1.8-4,4-4h59.8c2.2,0,4,1.8,4,4V413z',
  handleSize: '110%',
  handleStyle: {
    color: '#d3dee5'

  },
  textStyle: {
```

```
    color: '#fff'
  },
  borderColor: '#90979c'
},
{
  type: 'inside',
  show: true,
  height: 15,
  start: 1,
  end: 35
}]
```

# 一、Canal介绍

## 1、应用场景

在前面的统计分析功能中，我们采取了服务调用获取统计数据，这样耦合度高，效率相对较低，目前我采取另一种实现方式，通过实时同步数据库表的方式实现，例如我们要统计每天注册与登录人数，我们只需把会员表同步到统计库中，实现本地统计就可以了，这样效率更高，耦合度更低，Canal就是一个很好的数据库同步工具。canal是阿里巴巴旗下的一款开源项目，纯Java开发。基于数据库增量日志解析，提供增量数据订阅&消费，目前主要支持了MySQL。

## 2、Canal环境搭建

**canal的原理是基于mysql binlog技术，所以这里一定需要开启mysql的binlog写入功能**

**开启mysql服务：** service mysql start

**（1）检查binlog功能是否有开启**

```
mysql> show variables like 'log_bin';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | OFF   |
+---------------+-------+
1 row in set (0.00 sec)
```

**（2）如果显示状态为OFF表示该功能未开启，**开启binlog功能

```
1, 修改 mysql 的配置文件 my.cnf
vi /etc/my.cnf
追加内容:
log-bin=mysql-bin     #binlog文件名
binlog_format=ROW     #选择row模式
server_id=1           #mysql实例id,不能和canal的slaveId重复

2, 重启 mysql:
service mysql restart
```

```
3，登录 mysql 客户端，查看 log_bin 变量
mysql> show variables like 'log_bin';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | ON|
+---------------+-------+
1 row in set (0.00 sec)
————————————
如果显示状态为ON表示该功能已开启
```

**(3)** 在mysql里面添加以下的相关用户和权限

```
CREATE USER 'canal'@'%' IDENTIFIED BY 'canal';
GRANT SHOW VIEW, SELECT, REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO
'canal'@'%';
FLUSH PRIVILEGES;
```

# 3、下载安装Canal服务

下载地址：

https://github.com/alibaba/canal/releases

（**1**）下载之后，放到目录中，解压文件

cd `/usr/local/canal`

canal.deployer-1.1.4.tar.gz

`tar zxvf canal.deployer-1.1.4.tar.gz`

（**2**）修改配置文件

`vi conf/example/instance.properties`

```
#需要改成自己的数据库信息
canal.instance.master.address=192.168.44.132:3306

#需要改成自己的数据库用户名与密码
```

```
5
6  canal.instance.dbUsername=canal
7  canal.instance.dbPassword=canal
8
9  #需要改成同步的数据库表规则，例如只是同步一下表
10 #canal.instance.filter.regex=.*\\..*
11 canal.instance.filter.regex=guli_ucenter.ucenter_member
```

注:

mysql 数据解析关注的表，Perl正则表达式.

多个正则之间以逗号(,)分隔，转义符需要双斜杠(\\)

常见例子:

1. 所有表: .* or .*\\..*
2. canal schema下所有表: canal\\..*
3. canal下的以canal打头的表: canal\\.canal.*
4. canal schema下的一张表: canal.test1
5. 多个规则组合使用: canal\\..*,mysql.test1,mysql.test2 (逗号分隔)

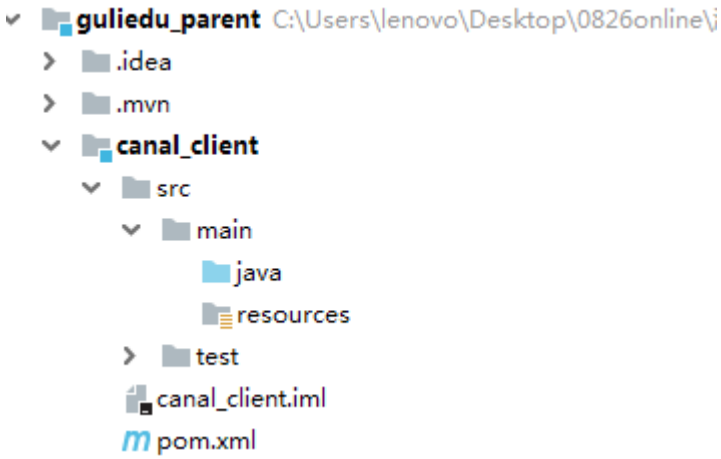
注意: 此过滤条件只针对row模式的数据有效(ps. mixed/statement因为不解析sql，所以无法准确提取tableName进行过滤)

（**3**）进入**bin**目录下启动

**sh bin/startup.sh**

# 二、创建canal_client模块

## 1、在guliedu_parent下创建canal_client模块




## 2、引入相关依赖

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!--mysql-->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>

    <dependency>
        <groupId>commons-dbutils</groupId>
        <artifactId>commons-dbutils</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jdbc</artifactId>
    </dependency>

    <dependency>
        <groupId>com.alibaba.otter</groupId>
        <artifactId>canal.client</artifactId>
    </dependency>
</dependencies>
```

## 3、创建application.properties配置文件

```
# 服务端口
server.port=10000
# 服务名
spring.application.name=canal-client

# 环境设置：dev、test、prod
spring.profiles.active=dev

# mysql数据库连接
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/guli?serverTimezone=GMT%2B8
spring.datasource.username=root
spring.datasource.password=root
```

## 4、编写canal客户端类

```
import com.alibaba.otter.canal.client.CanalConnector;
import com.alibaba.otter.canal.client.CanalConnectors;
import com.alibaba.otter.canal.protocol.CanalEntry.*;
import com.alibaba.otter.canal.protocol.Message;
import com.google.protobuf.InvalidProtocolBufferException;
import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;
import javax.sql.DataSource;
import java.net.InetSocketAddress;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.Iterator;
import java.util.List;
import java.util.Queue;
import java.util.concurrent.ConcurrentLinkedQueue;

@Component
```

```java
public class CanalClient {

    //sql队列
    private Queue<String> SQL_QUEUE = new ConcurrentLinkedQueue<>();

    @Resource
    private DataSource dataSource;

    /**
     * canal入库方法
     */
    public void run() {

        CanalConnector connector = CanalConnectors.newSingleConnector(new
InetSocketAddress("192.168.44.132",
                11111), "example", "", "");
        int batchSize = 1000;
        try {
            connector.connect();
            connector.subscribe(".*\\..*");
            connector.rollback();
            try {
                while (true) {
                    //尝试从master那边拉去数据batchSize条记录，有多少取多少
                    Message message = connector.getWithoutAck(batchSize);
                    long batchId = message.getId();
                    int size = message.getEntries().size();
                    if (batchId == -1 || size == 0) {
                        Thread.sleep(1000);
                    } else {
                        dataHandle(message.getEntries());
                    }
                    connector.ack(batchId);

                    //当队列里面堆积的sql大于一定数值的时候就模拟执行
                    if (SQL_QUEUE.size() >= 1) {
                        executeQueueSql();
                    }
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            } catch (InvalidProtocolBufferException e) {
                e.printStackTrace();
```

```
            }
        } finally {
            connector.disconnect();
        }
    }

    /**
     * 模拟执行队列里面的sql语句
     */
    public void executeQueueSql() {
        int size = SQL_QUEUE.size();
        for (int i = 0; i < size; i++) {
            String sql = SQL_QUEUE.poll();
            System.out.println("[sql]----> " + sql);

            this.execute(sql.toString());
        }
    }

    /**
     * 数据处理
     *
     * @param entrys
     */
    private void dataHandle(List<Entry> entrys) throws
InvalidProtocolBufferException {
        for (Entry entry : entrys) {
            if (EntryType.ROWDATA == entry.getEntryType()) {
                RowChange rowChange = RowChange.parseFrom(entry.getStoreValue());
                EventType eventType = rowChange.getEventType();
                if (eventType == EventType.DELETE) {
                    saveDeleteSql(entry);
                } else if (eventType == EventType.UPDATE) {
                    saveUpdateSql(entry);
                } else if (eventType == EventType.INSERT) {
                    saveInsertSql(entry);
                }
            }
        }
    }

    /**
     * 保存更新语句
```

```
     *
     * @param entry
     */
    private void saveUpdateSql(Entry entry) {
        try {
            RowChange rowChange = RowChange.parseFrom(entry.getStoreValue());
            List<RowData> rowDatasList = rowChange.getRowDatasList();
            for (RowData rowData : rowDatasList) {
                List<Column> newColumnList = rowData.getAfterColumnsList();
                StringBuffer sql = new StringBuffer("update " +
entry.getHeader().getTableName() + " set ");
                for (int i = 0; i < newColumnList.size(); i++) {
                    sql.append(" " + newColumnList.get(i).getName()
                            + " = '" + newColumnList.get(i).getValue() + "'");
                    if (i != newColumnList.size() - 1) {
                        sql.append(",");
                    }
                }
                sql.append(" where ");
                List<Column> oldColumnList = rowData.getBeforeColumnsList();
                for (Column column : oldColumnList) {
                    if (column.getIsKey()) {
                        //暂时只支持单一主键
                        sql.append(column.getName() + "=" + column.getValue());
                        break;
                    }
                }
                SQL_QUEUE.add(sql.toString());
            }
        } catch (InvalidProtocolBufferException e) {
            e.printStackTrace();
        }
    }

    /**
     * 保存删除语句
     *
     * @param entry
     */
    private void saveDeleteSql(Entry entry) {
        try {
            RowChange rowChange = RowChange.parseFrom(entry.getStoreValue());
            List<RowData> rowDatasList = rowChange.getRowDatasList();
```

```
            for (RowData rowData : rowDatasList) {
                List<Column> columnList = rowData.getBeforeColumnsList();
                StringBuffer sql = new StringBuffer("delete from " +
entry.getHeader().getTableName() + " where ");
                for (Column column : columnList) {
                    if (column.getIsKey()) {
                        //暂时只支持单一主键
                        sql.append(column.getName() + "=" + column.getValue());
                        break;
                    }
                }
                SQL_QUEUE.add(sql.toString());
            }
        } catch (InvalidProtocolBufferException e) {
            e.printStackTrace();
        }
    }

    /**
     * 保存插入语句
     *
     * @param entry
     */
    private void saveInsertSql(Entry entry) {
        try {
            RowChange rowChange = RowChange.parseFrom(entry.getStoreValue());
            List<RowData> rowDatasList = rowChange.getRowDatasList();
            for (RowData rowData : rowDatasList) {
                List<Column> columnList = rowData.getAfterColumnsList();
                StringBuffer sql = new StringBuffer("insert into " +
entry.getHeader().getTableName() + " (");
                for (int i = 0; i < columnList.size(); i++) {
                    sql.append(columnList.get(i).getName());
                    if (i != columnList.size() - 1) {
                        sql.append(",");
                    }
                }
                sql.append(") VALUES (");
                for (int i = 0; i < columnList.size(); i++) {
                    sql.append("'" + columnList.get(i).getValue() + "'");
                    if (i != columnList.size() - 1) {
                        sql.append(",");
                    }
```

```
                }
                sql.append(")");
                SQL_QUEUE.add(sql.toString());
            }
        } catch (InvalidProtocolBufferException e) {
            e.printStackTrace();
        }
    }

    /**
     * 入库
     * @param sql
     */
    public void execute(String sql) {
        Connection con = null;
        try {
            if(null == sql) return;
            con = dataSource.getConnection();
            QueryRunner qr = new QueryRunner();
            int row = qr.execute(con, sql);
            System.out.println("update: "+ row);
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            DbUtils.closeQuietly(con);
        }
    }
}
```

## 5、创建启动类

```
@SpringBootApplication
public class CanalApplication implements CommandLineRunner {
    @Resource
    private CanalClient canalClient;

    public static void main(String[] args) {
        SpringApplication.run(CanalApplication.class, args);
    }

```

```
    @Override
    public void run(String... strings) throws Exception {
        //项目启动，执行canal客户端监听
        canalClient.run();
    }
}
```

# 一、Canal介绍

## 1、应用场景

在前面的统计分析功能中，我们采取了服务调用获取统计数据，这样耦合度高，效率相对较低，目前我采取另一种实现方式，通过实时同步数据库表的方式实现，例如我们要统计每天注册与登录人数，我们只需把会员表同步到统计库中，实现本地统计就可以了，这样效率更高，耦合度更低，Canal就是一个很好的数据库同步工具。canal是阿里巴巴旗下的一款开源项目，纯Java开发。基于数据库增量日志解析，提供增量数据订阅&消费，目前主要支持了MySQL。

## 2、Canal环境搭建

**canal的原理是基于mysql binlog技术，所以这里一定需要开启mysql的binlog写入功能**

**开启mysql服务：** service mysql start **(或者 systemctl start mysqld.service)**

**（1）检查binlog功能是否有开启**

```
mysql> show variables like 'log_bin';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | OFF   |
+---------------+-------+
1 row in set (0.00 sec)
```

**（2）如果显示状态为OFF表示该功能未开启，**开启binlog功能

```
1, 修改 mysql 的配置文件 my.cnf
vi /etc/my.cnf
追加内容:
log-bin=mysql-bin     #binlog文件名
binlog_format=ROW     #选择row模式
server_id=1           #mysql实例id,不能和canal的slaveId重复

2, 重启 mysql:
service mysql restart
```

```
3，登录 mysql 客户端，查看 log_bin 变量
mysql> show variables like 'log_bin';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | ON|
+---------------+-------+
1 row in set (0.00 sec)
————————————
如果显示状态为ON表示该功能已开启
```

**(3)** 在mysql里面添加以下的相关用户和权限

```
CREATE USER 'canal'@'%' IDENTIFIED BY 'canal';
GRANT SHOW VIEW, SELECT, REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO
'canal'@'%';
FLUSH PRIVILEGES;
```

# 3、下载安装Canal服务

下载地址：

https://github.com/alibaba/canal/releases

（**1**）下载之后，放到目录中，解压文件

cd `/usr/local/canal`

canal.deployer-1.1.4.tar.gz

`tar zxvf canal.deployer-1.1.4.tar.gz`

（**2**）修改配置文件

`vi conf/example/instance.properties`

```
#需要改成自己的数据库信息
canal.instance.master.address=192.168.44.132:3306

#需要改成自己的数据库用户名与密码
```

```
5
6  canal.instance.dbUsername=canal
7  canal.instance.dbPassword=canal
8
9  #需要改成同步的数据库表规则，例如只是同步一下表
10 #canal.instance.filter.regex=.*\\..*
11 canal.instance.filter.regex=guli_ucenter.ucenter_member
```

注:

mysql 数据解析关注的表，Perl正则表达式.

多个正则之间以逗号(,)分隔，转义符需要双斜杠(\\)

常见例子:

1. 所有表: .* or .*\\..*
2. canal schema下所有表: canal\\..*
3. canal下的以canal打头的表: canal\\.canal.*
4. canal schema下的一张表: canal.test1
5. 多个规则组合使用: canal\\..*,mysql.test1,mysql.test2 (逗号分隔)

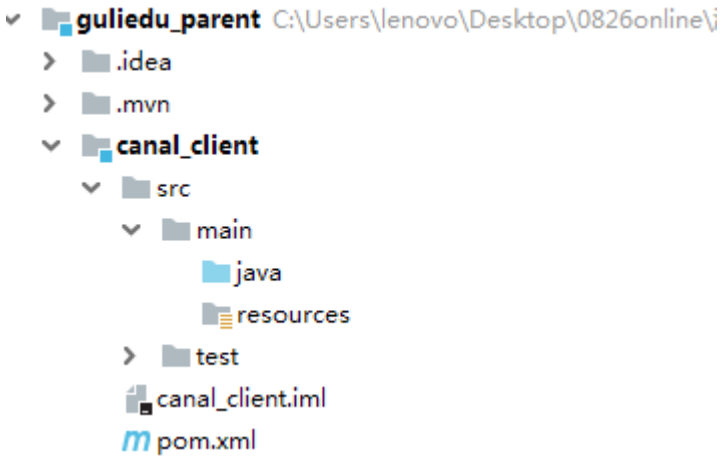
注意: 此过滤条件只针对row模式的数据有效(ps. mixed/statement因为不解析sql，所以无法准确提取tableName进行过滤)

（**3**）进入**bin**目录下启动

**sh bin/startup.sh**

# 二、创建canal_client模块

## 1、在guliedu_parent下创建canal_client模块




## 2、引入相关依赖

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!--mysql-->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>

    <dependency>
        <groupId>commons-dbutils</groupId>
        <artifactId>commons-dbutils</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jdbc</artifactId>
    </dependency>

    <dependency>
        <groupId>com.alibaba.otter</groupId>
        <artifactId>canal.client</artifactId>
    </dependency>
</dependencies>
```

## 3、创建application.properties配置文件

```
# 服务端口
server.port=10000
# 服务名
spring.application.name=canal-client

# 环境设置：dev、test、prod
spring.profiles.active=dev

# mysql数据库连接
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/guli?serverTimezone=GMT%2B8
spring.datasource.username=root
spring.datasource.password=root
```

## 4、编写canal客户端类

```
import com.alibaba.otter.canal.client.CanalConnector;
import com.alibaba.otter.canal.client.CanalConnectors;
import com.alibaba.otter.canal.protocol.CanalEntry.*;
import com.alibaba.otter.canal.protocol.Message;
import com.google.protobuf.InvalidProtocolBufferException;
import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;
import javax.sql.DataSource;
import java.net.InetSocketAddress;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.Iterator;
import java.util.List;
import java.util.Queue;
import java.util.concurrent.ConcurrentLinkedQueue;

@Component
```

```java
public class CanalClient {

    //sql队列
    private Queue<String> SQL_QUEUE = new ConcurrentLinkedQueue<>();

    @Resource
    private DataSource dataSource;

    /**
     * canal入库方法
     */
    public void run() {

        CanalConnector connector = CanalConnectors.newSingleConnector(new
InetSocketAddress("192.168.44.132",
                11111), "example", "", "");
        int batchSize = 1000;
        try {
            connector.connect();
            connector.subscribe(".*\\..*");
            connector.rollback();
            try {
                while (true) {
                    //尝试从master那边拉去数据batchSize条记录，有多少取多少
                    Message message = connector.getWithoutAck(batchSize);
                    long batchId = message.getId();
                    int size = message.getEntries().size();
                    if (batchId == -1 || size == 0) {
                        Thread.sleep(1000);
                    } else {
                        dataHandle(message.getEntries());
                    }
                    connector.ack(batchId);

                    //当队列里面堆积的sql大于一定数值的时候就模拟执行
                    if (SQL_QUEUE.size() >= 1) {
                        executeQueueSql();
                    }
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            } catch (InvalidProtocolBufferException e) {
                e.printStackTrace();
```

```
            }
        } finally {
            connector.disconnect();
        }
    }

    /**
     * 模拟执行队列里面的sql语句
     */
    public void executeQueueSql() {
        int size = SQL_QUEUE.size();
        for (int i = 0; i < size; i++) {
            String sql = SQL_QUEUE.poll();
            System.out.println("[sql]----> " + sql);

            this.execute(sql.toString());
        }
    }

    /**
     * 数据处理
     *
     * @param entrys
     */
    private void dataHandle(List<Entry> entrys) throws
InvalidProtocolBufferException {
        for (Entry entry : entrys) {
            if (EntryType.ROWDATA == entry.getEntryType()) {
                RowChange rowChange = RowChange.parseFrom(entry.getStoreValue());
                EventType eventType = rowChange.getEventType();
                if (eventType == EventType.DELETE) {
                    saveDeleteSql(entry);
                } else if (eventType == EventType.UPDATE) {
                    saveUpdateSql(entry);
                } else if (eventType == EventType.INSERT) {
                    saveInsertSql(entry);
                }
            }
        }
    }

    /**
     * 保存更新语句
```

```
     *
     * @param entry
     */
    private void saveUpdateSql(Entry entry) {
        try {
            RowChange rowChange = RowChange.parseFrom(entry.getStoreValue());
            List<RowData> rowDatasList = rowChange.getRowDatasList();
            for (RowData rowData : rowDatasList) {
                List<Column> newColumnList = rowData.getAfterColumnsList();
                StringBuffer sql = new StringBuffer("update " +
entry.getHeader().getTableName() + " set ");
                for (int i = 0; i < newColumnList.size(); i++) {
                    sql.append(" " + newColumnList.get(i).getName()
                            + " = '" + newColumnList.get(i).getValue() + "'");
                    if (i != newColumnList.size() - 1) {
                        sql.append(",");
                    }
                }
                sql.append(" where ");
                List<Column> oldColumnList = rowData.getBeforeColumnsList();
                for (Column column : oldColumnList) {
                    if (column.getIsKey()) {
                        //暂时只支持单一主键
                        sql.append(column.getName() + "=" + column.getValue());
                        break;
                    }
                }
                SQL_QUEUE.add(sql.toString());
            }
        } catch (InvalidProtocolBufferException e) {
            e.printStackTrace();
        }
    }

    /**
     * 保存删除语句
     *
     * @param entry
     */
    private void saveDeleteSql(Entry entry) {
        try {
            RowChange rowChange = RowChange.parseFrom(entry.getStoreValue());
            List<RowData> rowDatasList = rowChange.getRowDatasList();
```

```
            for (RowData rowData : rowDatasList) {
                List<Column> columnList = rowData.getBeforeColumnsList();
                StringBuffer sql = new StringBuffer("delete from " +
entry.getHeader().getTableName() + " where ");
                for (Column column : columnList) {
                    if (column.getIsKey()) {
                        //暂时只支持单一主键
                        sql.append(column.getName() + "=" + column.getValue());
                        break;
                    }
                }
                SQL_QUEUE.add(sql.toString());
            }
        } catch (InvalidProtocolBufferException e) {
            e.printStackTrace();
        }
    }

    /**
     * 保存插入语句
     *
     * @param entry
     */
    private void saveInsertSql(Entry entry) {
        try {
            RowChange rowChange = RowChange.parseFrom(entry.getStoreValue());
            List<RowData> rowDatasList = rowChange.getRowDatasList();
            for (RowData rowData : rowDatasList) {
                List<Column> columnList = rowData.getAfterColumnsList();
                StringBuffer sql = new StringBuffer("insert into " +
entry.getHeader().getTableName() + " (");
                for (int i = 0; i < columnList.size(); i++) {
                    sql.append(columnList.get(i).getName());
                    if (i != columnList.size() - 1) {
                        sql.append(",");
                    }
                }
                sql.append(") VALUES (");
                for (int i = 0; i < columnList.size(); i++) {
                    sql.append("'" + columnList.get(i).getValue() + "'");
                    if (i != columnList.size() - 1) {
                        sql.append(",");
                    }
```

```
                }
                sql.append(")");
                SQL_QUEUE.add(sql.toString());
            }
        } catch (InvalidProtocolBufferException e) {
            e.printStackTrace();
        }
    }

    /**
     * 入库
     * @param sql
     */
    public void execute(String sql) {
        Connection con = null;
        try {
            if(null == sql) return;
            con = dataSource.getConnection();
            QueryRunner qr = new QueryRunner();
            int row = qr.execute(con, sql);
            System.out.println("update: "+ row);
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            DbUtils.closeQuietly(con);
        }
    }
}
```

## 5、创建启动类

```
@SpringBootApplication
public class CanalApplication implements CommandLineRunner {
    @Resource
    private CanalClient canalClient;

    public static void main(String[] args) {
        SpringApplication.run(CanalApplication.class, args);
    }

```

```
    @Override
    public void run(String... strings) throws Exception {
        //项目启动，执行canal客户端监听
        canalClient.run();
    }
}
```

# 一、网关基本概念

## 1、API网关介绍

API 网关出现的原因是微服务架构的出现，不同的微服务一般会有不同的网络地址，而外部客户端可能需要调用多个服务的接口才能完成一个业务需求，如果让客户端直接与各个微服务通信，会有以下的问题：

（1）客户端会多次请求不同的微服务，增加了客户端的复杂性。

（2）存在跨域请求，在一定场景下处理相对复杂。

（3）认证复杂，每个服务都需要独立认证。

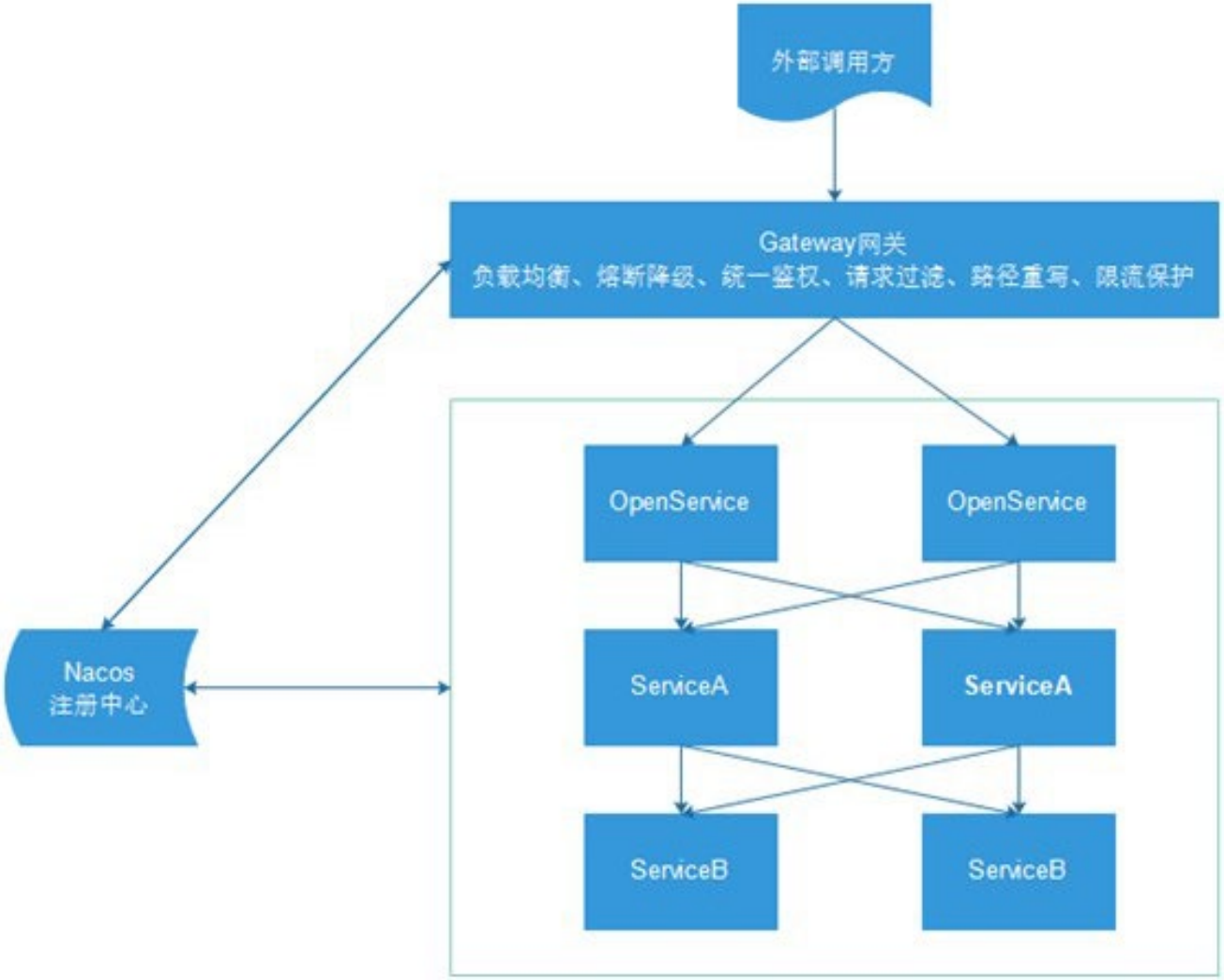
（4）难以重构，随着项目的迭代，可能需要重新划分微服务。例如，可能将多个服务合并成一个或者将一个服务拆分成多个。如果客户端直接与微服务通信，那么重构将会很难实施。

（5）某些微服务可能使用了防火墙 / 浏览器不友好的协议，直接访问会有一定的困难。

以上这些问题可以借助 API 网关解决。API 网关是介于客户端和服务器端之间的中间层，所有的外部请求都会先经过 API 网关这一层。也就是说，API 的实现方面更多的考虑业务逻辑，而安全、性能、监控可以交由 API 网关来做，这样既提高业务灵活性又不缺安全性

## 2、Spring Cloud Gateway

Spring cloud gateway是spring官方基于Spring 5.0、Spring Boot2.0和Project Reactor等技术开发的网关，Spring Cloud Gateway旨在为微服务架构提供简单、有效和统一的API路由管理方式，Spring Cloud Gateway作为Spring Cloud生态系统中的网关，目标是替代Netflix Zuul，其不仅提供统一的路由方式，并且还基于Filer链的方式提供了网关基本的功能，例如：安全、监控/埋点、限流等。



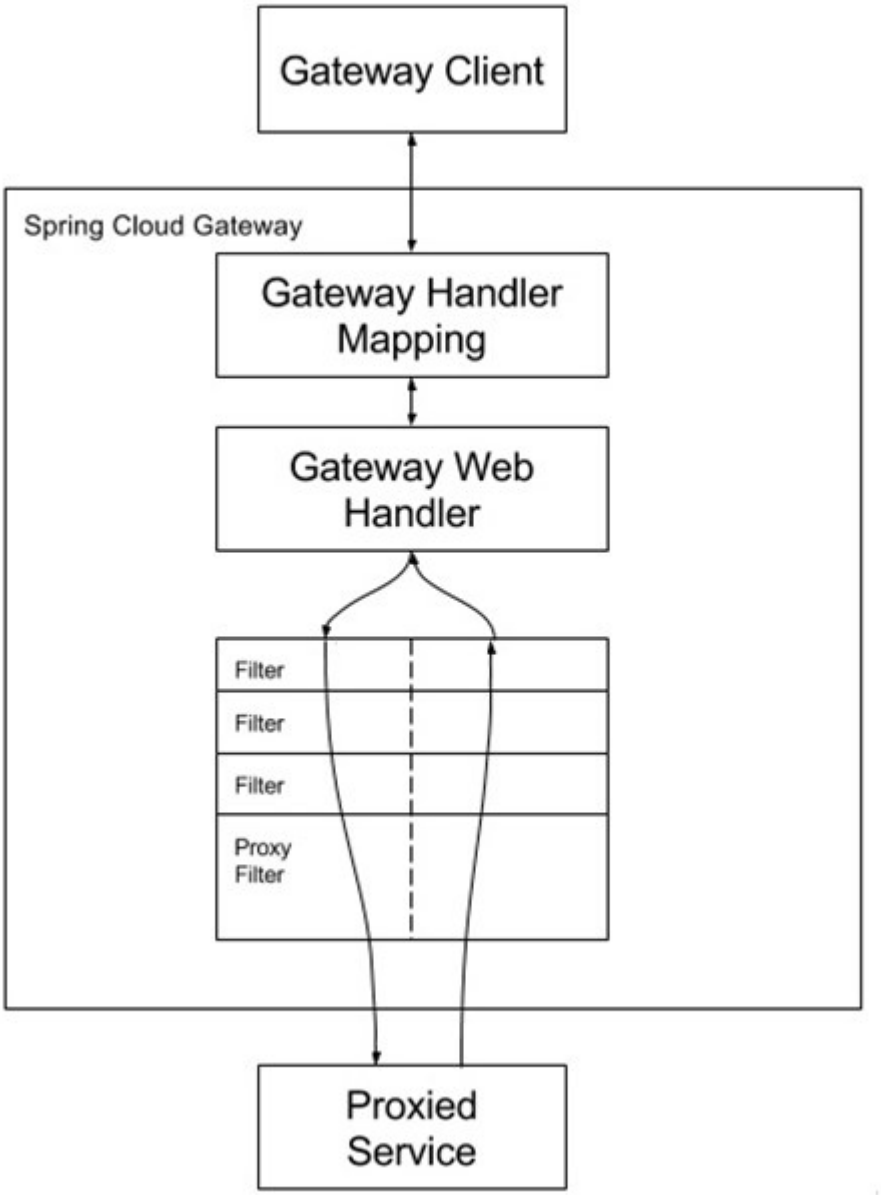

## 3、Spring Cloud Gateway核心概念

网关提供API全托管服务，丰富的API管理功能，辅助企业管理大规模的API，以降低管理成本和安全风险，包括协议适配、协议转发、安全策略、防刷、流量、监控日志等贡呢。一般来说网关对外暴露的URL或者接口信息，我们统称为路由信息。如果研发过网关中间件或者使用过Zuul的人，会知道网关的核心是Filter以及Filter Chain（Filter责任链）。Sprig Cloud Gateway也具有路由和Filter的概念。下面介绍一下Spring Cloud Gateway中几个重要的概念。

（1）路由。路由是网关最基础的部分，路由信息有一个ID、一个目的URL、一组断言和一组Filter组成。如果断言路由为真，则说明请求的URL和配置匹配

（2）断言。Java8中的断言函数。Spring Cloud Gateway中的断言函数输入类型是Spring5.0框架中的ServerWebExchange。Spring Cloud Gateway中的断言函数允许开发者去定义匹配来自于http request中的任何信息，比如请求头和参数等。

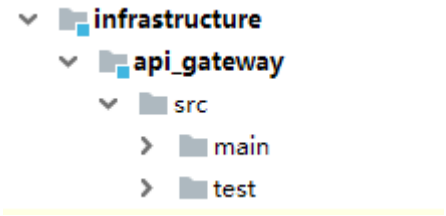
（3）过滤器。一个标准的Spring webFilter。Spring cloud gateway中的filter分为两种类型的Filter，分别是Gateway Filter和Global Filter。过滤器Filter将会对请求和响应进行修改处理




如上图所示，Spring cloud Gateway发出请求。然后再由Gateway Handler Mapping中找到与请求相匹配的路由，将其发送到Gateway web handler。Handler再通过指定的过滤器链将请求发送到我们实际的服务执行业务逻辑，然后返回。

## 二、创建**api-gateway**模块（网关服务）

1、在infrastructure模块下创建api_gateway模块




2、在pom.xml引入依赖

```xml
<dependencies>
    <dependency>
        <groupId>com.atguigu</groupId>
        <artifactId>common_utils</artifactId>
        <version>0.0.1-SNAPSHOT</version>
    </dependency>

    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-gateway</artifactId>
    </dependency>

    <!--gson-->
    <dependency>
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
    </dependency>

    <!--服务调用-->
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-openfeign</artifactId>
    </dependency>
</dependencies>
```

3、编写application.properties配置文件

```
# 服务端口
server.port=8222
# 服务名
spring.application.name=service-gateway

# nacos服务地址
spring.cloud.nacos.discovery.server-addr=127.0.0.1:8848

#使用服务发现路由
spring.cloud.gateway.discovery.locator.enabled=true
#服务路由名小写
#spring.cloud.gateway.discovery.locator.lower-case-service-id=true

#设置路由id
spring.cloud.gateway.routes[0].id=service-acl
#设置路由的uri
spring.cloud.gateway.routes[0].uri=lb://service-acl
#设置路由断言,代理servicerId为auth-service的/auth/路径
spring.cloud.gateway.routes[0].predicates= Path=/*/acl/**

#配置service-edu服务
spring.cloud.gateway.routes[1].id=service-edu
spring.cloud.gateway.routes[1].uri=lb://service-edu
spring.cloud.gateway.routes[1].predicates= Path=/eduservice/**

#配置service-ucenter服务
spring.cloud.gateway.routes[2].id=service-ucenter
spring.cloud.gateway.routes[2].uri=lb://service-ucenter
spring.cloud.gateway.routes[2].predicates= Path=/ucenterservice/**

#配置service-ucenter服务
spring.cloud.gateway.routes[3].id=service-cms
spring.cloud.gateway.routes[3].uri=lb://service-cms
spring.cloud.gateway.routes[3].predicates= Path=/cmsservice/**

spring.cloud.gateway.routes[4].id=service-msm
spring.cloud.gateway.routes[4].uri=lb://service-msm
spring.cloud.gateway.routes[4].predicates= Path=/edumsm/**

```

```
spring.cloud.gateway.routes[5].id=service-order
spring.cloud.gateway.routes[5].uri=lb://service-order
spring.cloud.gateway.routes[5].predicates= Path=/orderservice/**

spring.cloud.gateway.routes[6].id=service-order
spring.cloud.gateway.routes[6].uri=lb://service-order
spring.cloud.gateway.routes[6].predicates= Path=/orderservice/**

spring.cloud.gateway.routes[7].id=service-oss
spring.cloud.gateway.routes[7].uri=lb://service-oss
spring.cloud.gateway.routes[7].predicates= Path=/eduoss/**

spring.cloud.gateway.routes[8].id=service-statistic
spring.cloud.gateway.routes[8].uri=lb://service-statistic
spring.cloud.gateway.routes[8].predicates= Path=/staservice/**

spring.cloud.gateway.routes[9].id=service-vod
spring.cloud.gateway.routes[9].uri=lb://service-vod
spring.cloud.gateway.routes[9].predicates= Path=/eduvod/**

spring.cloud.gateway.routes[10].id=service-edu
spring.cloud.gateway.routes[10].uri=lb://service-edu
spring.cloud.gateway.routes[10].predicates= Path=/eduuser/**
```

**yml**文件:

```
server:
  port: 8222

spring:
  application:
  cloud:
    gateway:
      discovery:
        locator:
          enabled: true
      routes:
      - id: SERVICE-ACL
        uri: lb://SERVICE-ACL
        predicates:
```

```
        - Path=/*/acl/** # 路径匹配
      - id: SERVICE-EDU
        uri: lb://SERVICE-EDU
        predicates:
        - Path=/eduservice/** # 路径匹配
      - id: SERVICE-UCENTER
        uri: lb://SERVICE-UCENTER
        predicates:
        - Path=/ucenter/** # 路径匹配
    nacos:
      discovery:
        server-addr: 127.0.0.1:8848
```

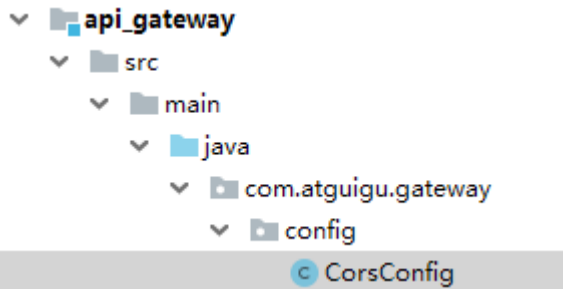
### 4、编写启动类

```
@SpringBootApplication
public class ApiGatewayApplication {
    public static void main(String[] args) {
        SpringApplication.run(ApiGatewayApplication.class, args);
    }
}
```

## 二、网关相关配置

### 1、网关解决跨域问题

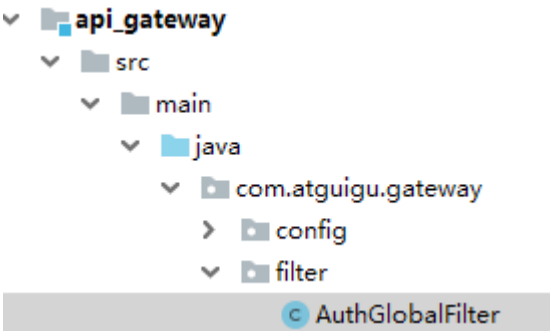
（1）创建配置类




```
@Configuration
public class CorsConfig {
```

```
    @Bean
    public CorsWebFilter corsFilter() {
        CorsConfiguration config = new CorsConfiguration();
        config.addAllowedMethod("*");
        config.addAllowedOrigin("*");
        config.addAllowedHeader("*");

        UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource(new PathPatternParser());
        source.registerCorsConfiguration("/**", config);

        return new CorsWebFilter(source);
    }
}
```

2、全局Filter，统一处理会员登录与外部不允许访问的服务



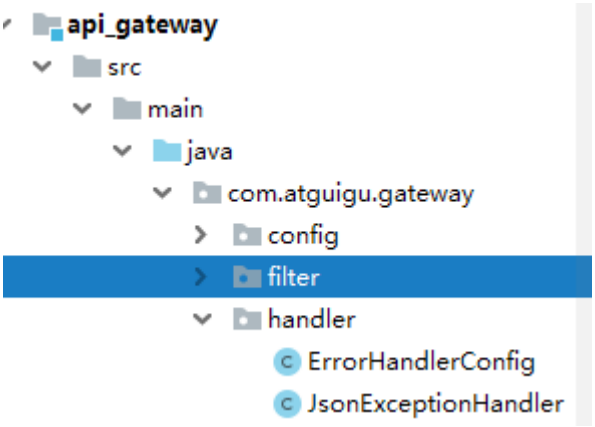

```
import com.google.gson.JsonObject;
import org.springframework.cloud.gateway.filter.GatewayFilterChain;
import org.springframework.cloud.gateway.filter.GlobalFilter;
import org.springframework.core.Ordered;
import org.springframework.core.io.buffer.DataBuffer;
import org.springframework.http.server.reactive.ServerHttpRequest;
import org.springframework.http.server.reactive.ServerHttpResponse;
import org.springframework.stereotype.Component;
import org.springframework.util.AntPathMatcher;
import org.springframework.web.server.ServerWebExchange;
import reactor.core.publisher.Mono;

import java.nio.charset.StandardCharsets;
import java.util.List;

```

```java
/**
 * <p>
 * 全局Filter，统一处理会员登录与外部不允许访问的服务
 * </p>
 */
@Component
public class AuthGlobalFilter implements GlobalFilter, Ordered {

    private AntPathMatcher antPathMatcher = new AntPathMatcher();

    @Override
    public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain)
{
        ServerHttpRequest request = exchange.getRequest();
        String path = request.getURI().getPath();
        //谷粒学院api接口，校验用户必须登录
        if(antPathMatcher.match("/api/**/auth/**", path)) {
            List<String> tokenList = request.getHeaders().get("token");
            if(null == tokenList) {
                ServerHttpResponse response = exchange.getResponse();
                return out(response);
            } else {
//                Boolean isCheck = JwtUtils.checkToken(tokenList.get(0));
//                if(!isCheck) {
                  ServerHttpResponse response = exchange.getResponse();
                  return out(response);
//                }
            }
        }
        //内部服务接口，不允许外部访问
        if(antPathMatcher.match("/**/inner/**", path)) {
            ServerHttpResponse response = exchange.getResponse();
            return out(response);
        }
        return chain.filter(exchange);
    }

    @Override
    public int getOrder() {
        return 0;
    }

    private Mono<Void> out(ServerHttpResponse response) {
```

```
        JsonObject message = new JsonObject();
        message.addProperty("success", false);
        message.addProperty("code", 28004);
        message.addProperty("data", "鉴权失败");
        byte[] bits = message.toString().getBytes(StandardCharsets.UTF_8);
        DataBuffer buffer = response.bufferFactory().wrap(bits);
        //response.setStatusCode(HttpStatus.UNAUTHORIZED);
        //指定编码，否则在浏览器中会中文乱码
        response.getHeaders().add("Content-Type", "application/json;charset=UTF-8");
        return response.writeWith(Mono.just(buffer));
    }
}

```

## 3、自定义异常处理

服务网关调用服务时可能会有一些异常或服务不可用，它返回错误信息不友好，需要我们覆盖处理




ErrorHandlerConfig：

```
import org.springframework.beans.factory.ObjectProvider;
import org.springframework.boot.autoconfigure.web.ResourceProperties;
import org.springframework.boot.autoconfigure.web.ServerProperties;
import org.springframework.boot.context.properties.EnableConfigurationProperties;
import org.springframework.boot.web.reactive.error.ErrorAttributes;
import org.springframework.boot.web.reactive.error.ErrorWebExceptionHandler;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
```

```java
import org.springframework.context.annotation.Configuration;
import org.springframework.core.Ordered;
import org.springframework.core.annotation.Order;
import org.springframework.http.codec.ServerCodecConfigurer;
import org.springframework.web.reactive.result.view.ViewResolver;

import java.util.Collections;
import java.util.List;

/**
 * 覆盖默认的异常处理
 *
 */
@Configuration
@EnableConfigurationProperties({ServerProperties.class, ResourceProperties.class})
public class ErrorHandlerConfig {

    private final ServerProperties serverProperties;

    private final ApplicationContext applicationContext;

    private final ResourceProperties resourceProperties;

    private final List<ViewResolver> viewResolvers;

    private final ServerCodecConfigurer serverCodecConfigurer;

    public ErrorHandlerConfig(ServerProperties serverProperties,
                              ResourceProperties resourceProperties,
                              ObjectProvider<List<ViewResolver>>
viewResolversProvider,
                                 ServerCodecConfigurer
serverCodecConfigurer,
                              ApplicationContext applicationContext) {
        this.serverProperties = serverProperties;
        this.applicationContext = applicationContext;
        this.resourceProperties = resourceProperties;
        this.viewResolvers =
viewResolversProvider.getIfAvailable(Collections::emptyList);
        this.serverCodecConfigurer = serverCodecConfigurer;
    }

    @Bean
```

```
    @Order(Ordered.HIGHEST_PRECEDENCE)
    public ErrorWebExceptionHandler errorWebExceptionHandler(ErrorAttributes errorAttributes) {
        JsonExceptionHandler exceptionHandler = new JsonExceptionHandler(
                errorAttributes,
                this.resourceProperties,
                this.serverProperties.getError(),
                this.applicationContext);
        exceptionHandler.setViewResolvers(this.viewResolvers);
        exceptionHandler.setMessageWriters(this.serverCodecConfigurer.getWriters());
        exceptionHandler.setMessageReaders(this.serverCodecConfigurer.getReaders());
        return exceptionHandler;
    }
}
```

JsonExceptionHandler:

```
import org.springframework.boot.autoconfigure.web.ErrorProperties;
import org.springframework.boot.autoconfigure.web.ResourceProperties;
import org.springframework.boot.autoconfigure.web.reactive.error.DefaultErrorWebExceptionHandler;
import org.springframework.boot.web.reactive.error.ErrorAttributes;
import org.springframework.context.ApplicationContext;
import org.springframework.http.HttpStatus;
import org.springframework.web.reactive.function.server.*;

import java.util.HashMap;
import java.util.Map;

/**
 * 自定义异常处理
 *
 * <p>异常时用JSON代替HTML异常信息<p>
 *
 */
public class JsonExceptionHandler extends DefaultErrorWebExceptionHandler {

    public JsonExceptionHandler(ErrorAttributes errorAttributes,
```

```
ResourceProperties resourceProperties,
                                  ErrorProperties errorProperties,
ApplicationContext applicationContext) {
        super(errorAttributes, resourceProperties, errorProperties,
applicationContext);
    }

    /**
     * 获取异常属性
     */
    @Override
    protected Map<String, Object> getErrorAttributes(ServerRequest request,
boolean includeStackTrace) {
        Map<String, Object> map = new HashMap<>();
        map.put("success", false);
        map.put("code", 20005);
        map.put("message", "网关失败");
        map.put("data", null);
        return map;
    }

    /**
     * 指定响应处理方法为JSON处理的方法
     * @param errorAttributes
     */
    @Override
    protected RouterFunction<ServerResponse> getRoutingFunction(ErrorAttributes
errorAttributes) {
        return RouterFunctions.route(RequestPredicates.all(),
this::renderErrorResponse);
    }

    /**
     * 根据code获取对应的HttpStatus
     * @param errorAttributes
     */
    @Override
    protected HttpStatus getHttpStatus(Map<String, Object> errorAttributes) {
        return HttpStatus.OK;
    }
}
```

# 一、权限管理需求描述

不同角色的用户登录后台管理系统拥有不同的菜单权限与功能权限，权限管理包含三个功能模块：菜单管理、角色管理和用户管理

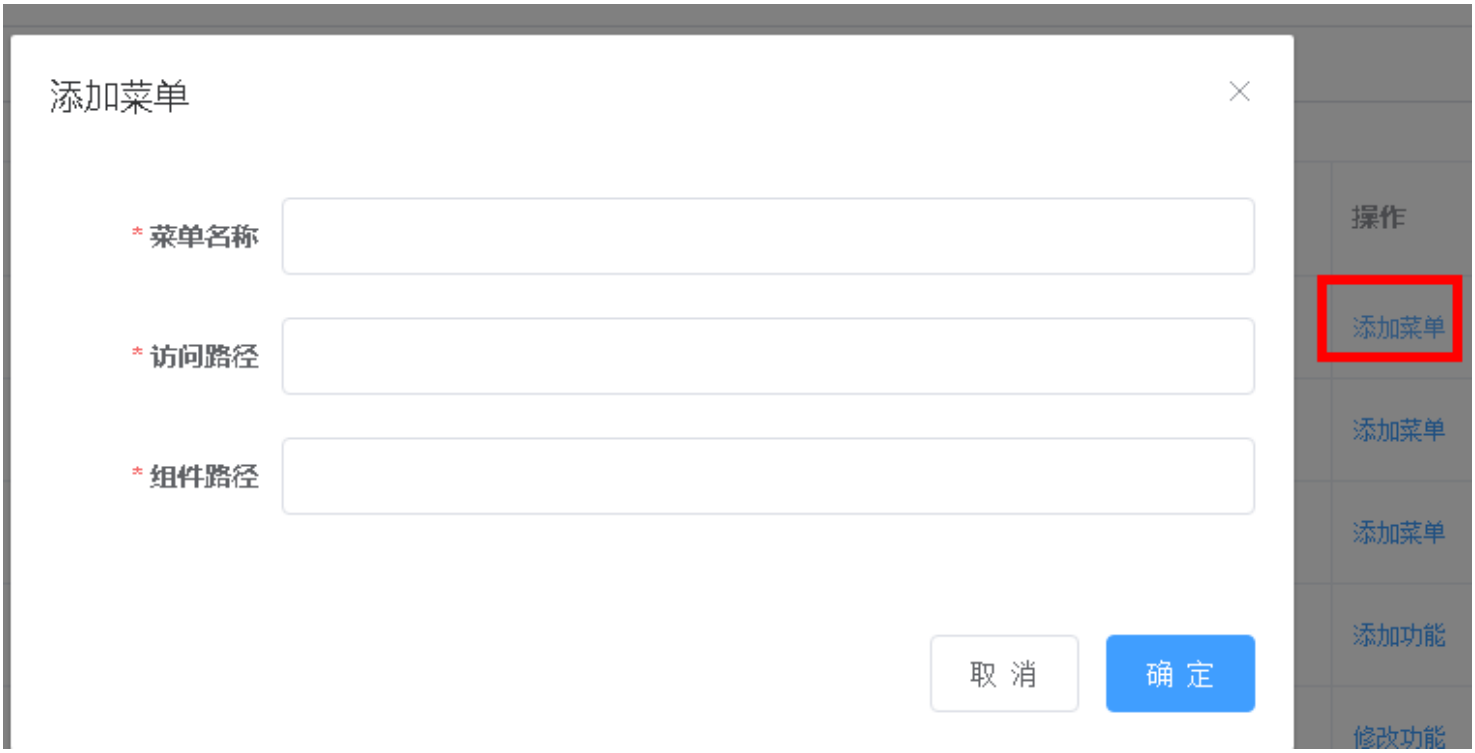

## 1、菜单管理

（1）菜单列表：使用树形结构显示菜单列表

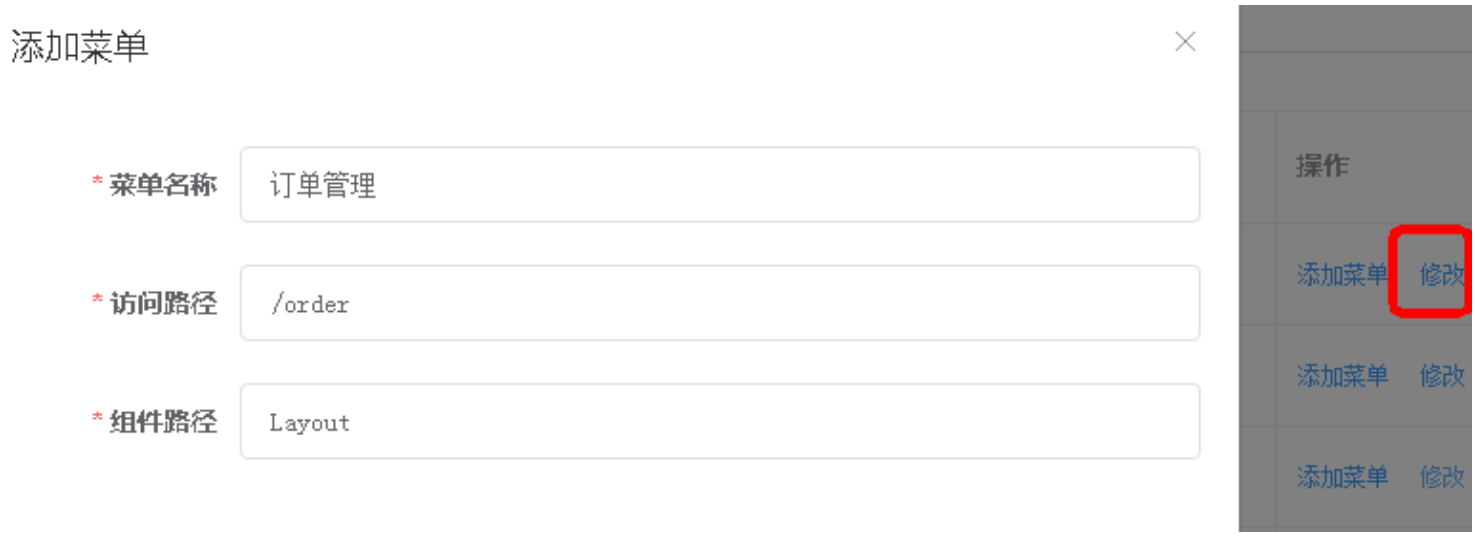
Filter keyword

| 名称 | 访问路径 | 组件路径 | 权限值 | 操作 |
| --- | --- | --- | --- | --- |
| 全部数据 | | | | 添加菜单 修改 删除 |
| 1 | 1 | 1 | | 添加菜单 修改 删除 |
| 订单管理 | /order | Layout | | 添加菜单 修改 删除 |

（2）添加菜单：点击添加菜单，弹框进行添加




（**3**）修改菜单

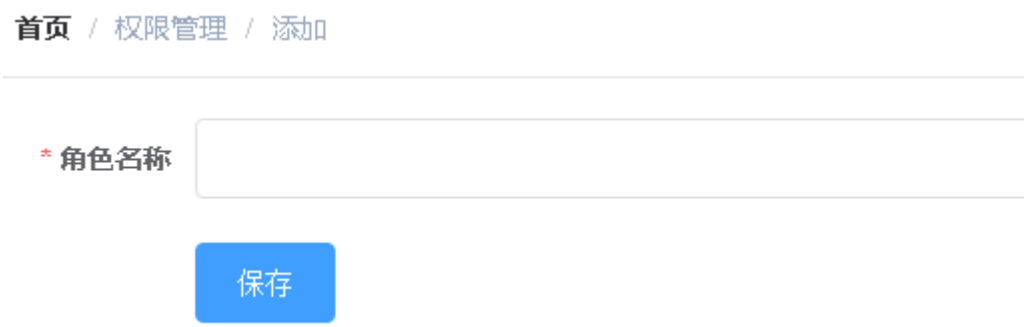



（**4**）删除菜单


## **2**、角色管理

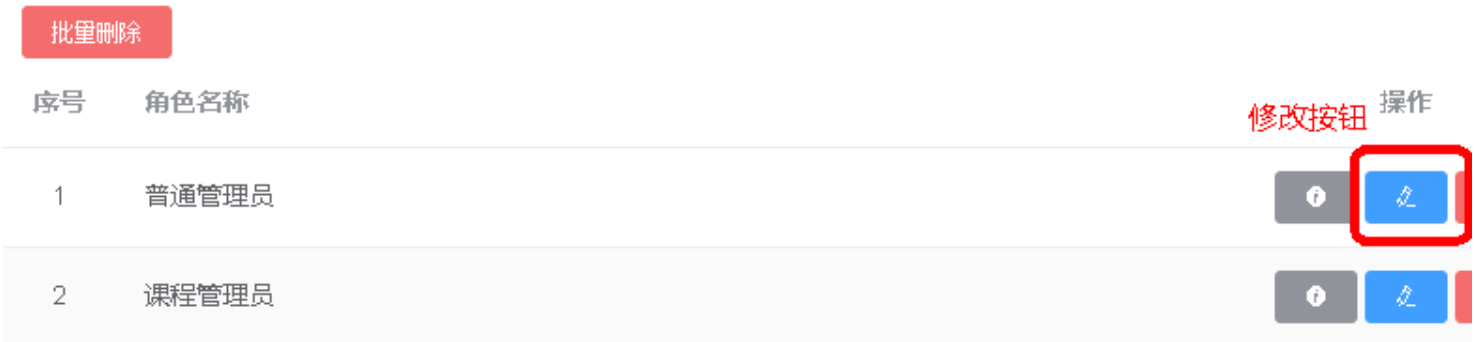
（**1**）角色列表：实现角色的条件查询带分页功能


## （2）角色添加




## （3）角色修改

点击修改按钮




数据回显，进行修改

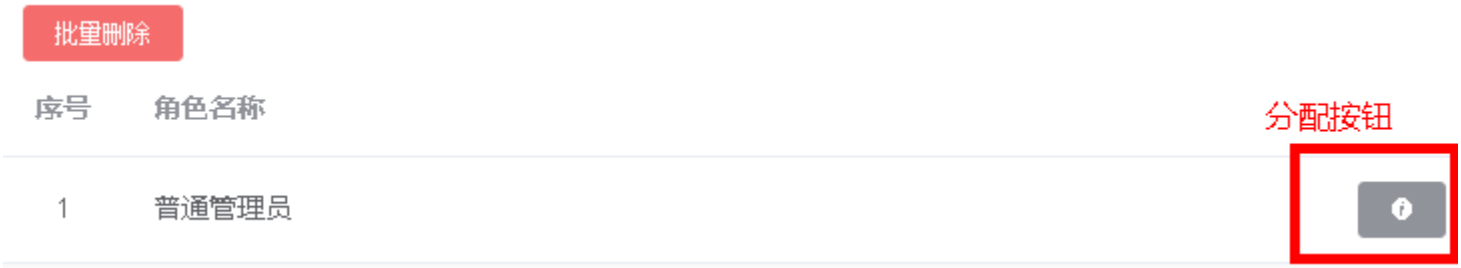

## （4）角色删除

普通删除

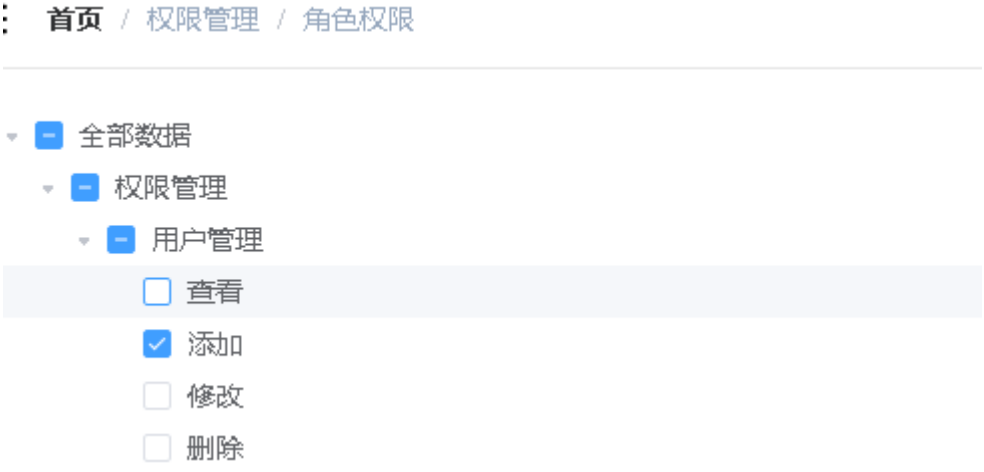

批量删除


### （5）角色分配菜单

点击分配按钮



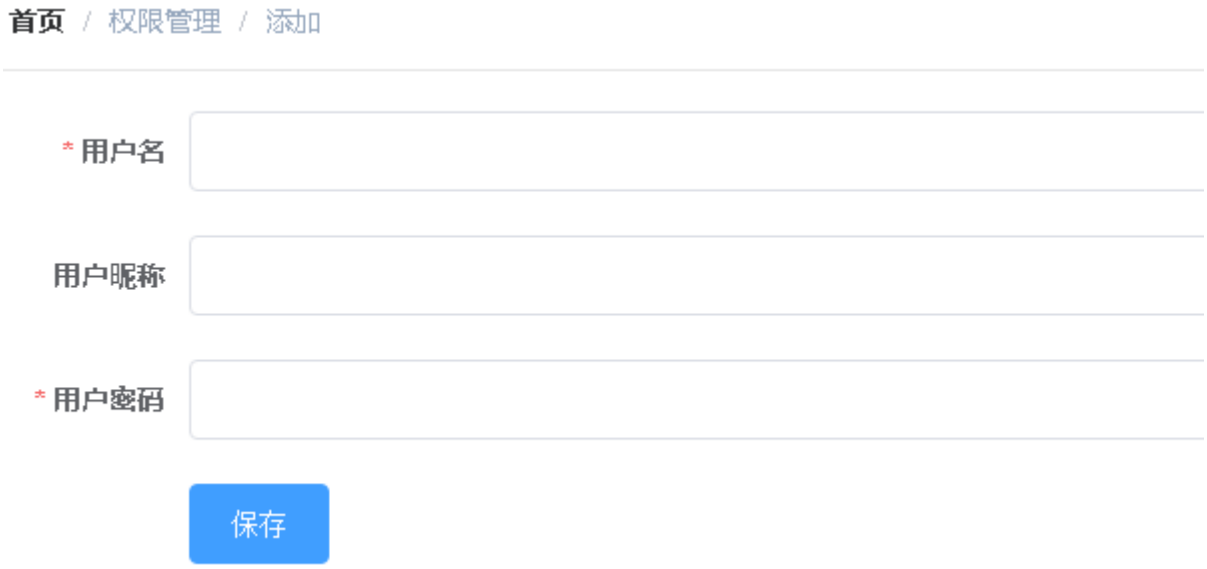

给角色分配菜单




## 3、用户管理

（1）用户列表


（2）用户添加




（3）用户修改

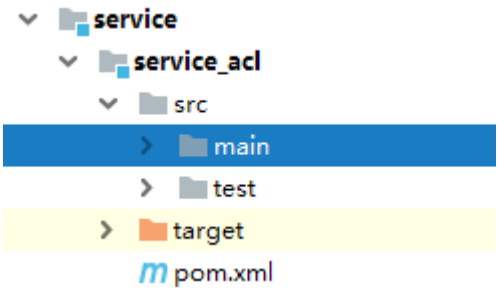

## （4）用户删除

普通删除和批量删除


## （5）用户分配角色

# 一、创建权限管理服务

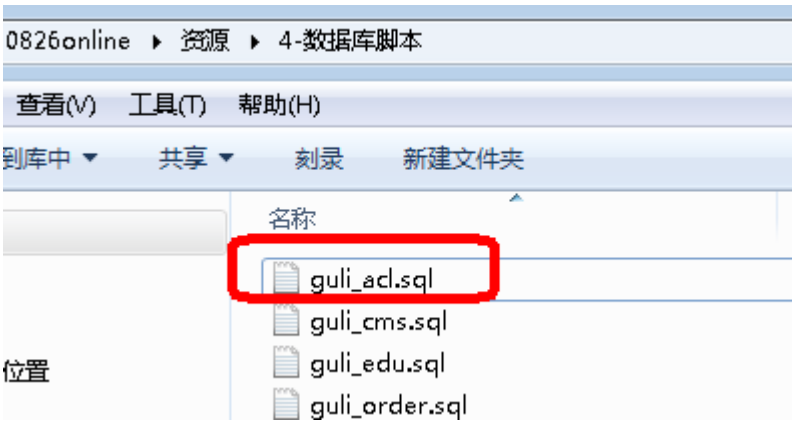
## 1、在service模块下创建子模块service-acl

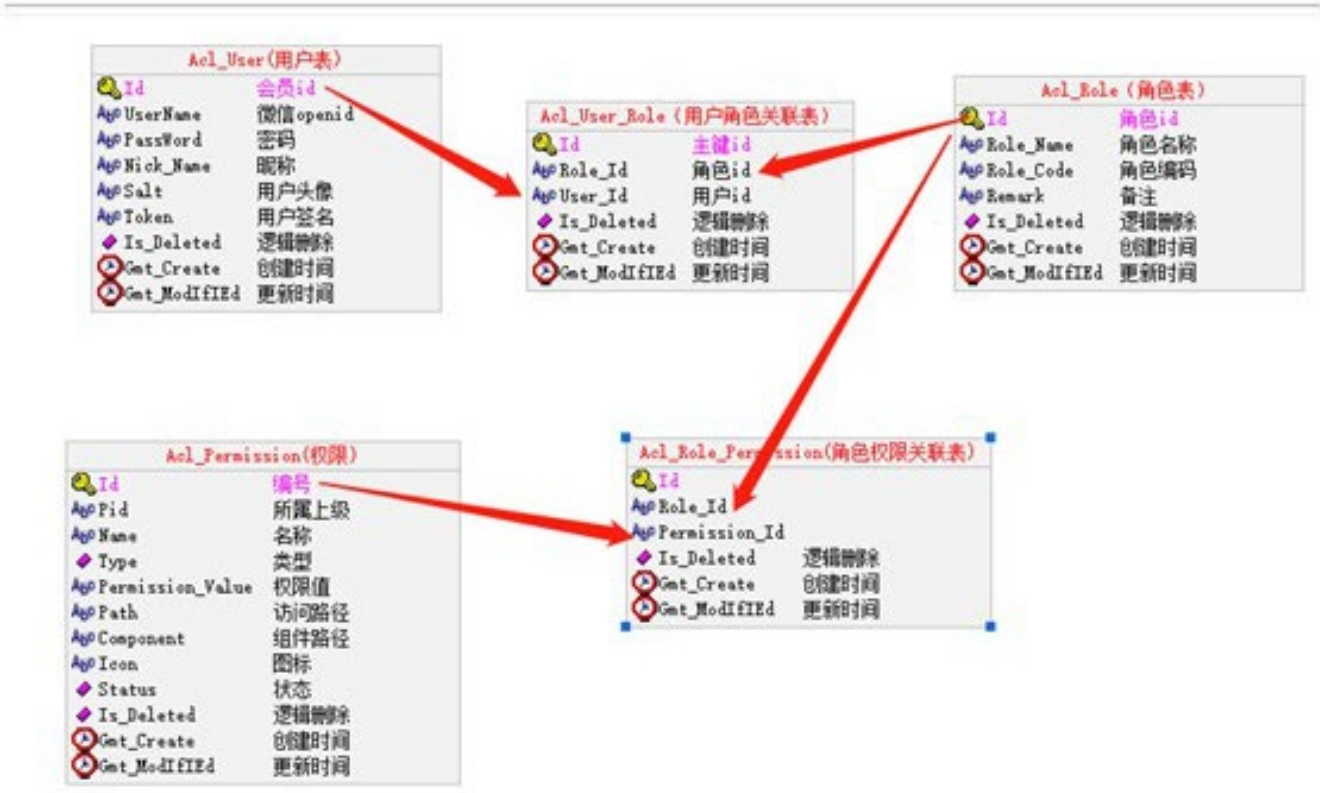



## 2、在service_acl模块中引入依赖

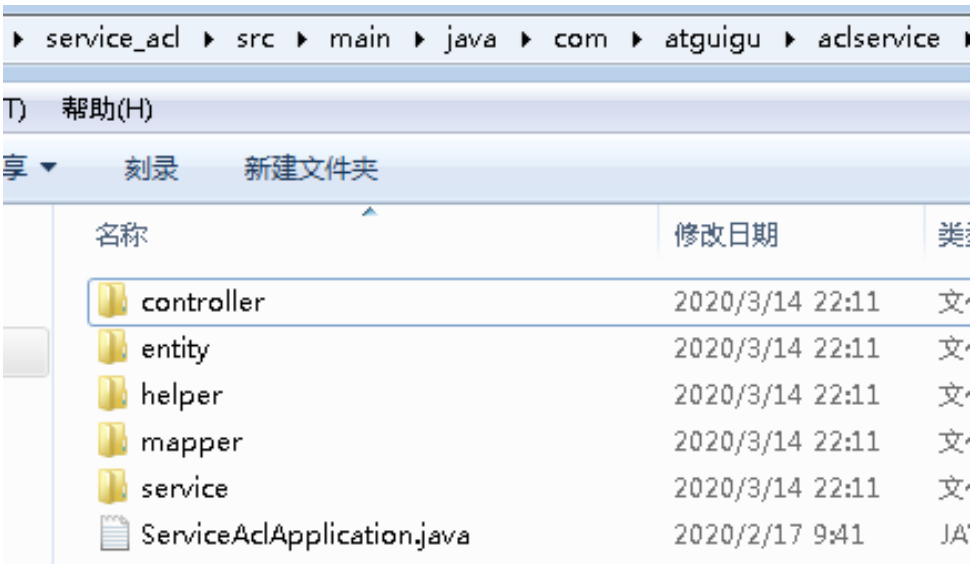
```
<dependencies>
    <dependency>
        <groupId>com.atguigu</groupId>
        <artifactId>spring_security</artifactId>
        <version>0.0.1-SNAPSHOT</version>
    </dependency>
    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>fastjson</artifactId>
    </dependency>
</dependencies>
```

## 3、创建权限管理相关的表



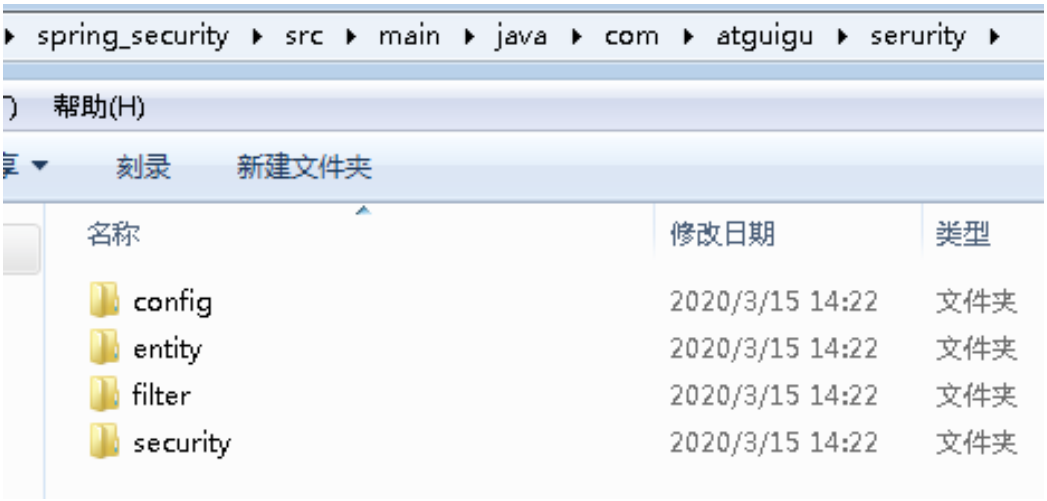



**4**、复制权限管理接口代码




**5**、复制整合**Spring Security**代码

（**1**）在**common**模块下创建子模块**spring_security**




## 6、编写application.properties配置文件

```
# 服务端口
server.port=8009
# 服务名
spring.application.name=service-acl

# mysql数据库连接
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/guli?serverTimezone=GMT%2B8
spring.datasource.username=root
spring.datasource.password=root

#返回json的全局时间格式
spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
spring.jackson.time-zone=GMT+8

spring.redis.host=192.168.44.132
spring.redis.port=6379
spring.redis.database= 0
spring.redis.timeout=1800000

spring.redis.lettuce.pool.max-active=20
spring.redis.lettuce.pool.max-wait=-1
#最大阻塞等待时间(负数表示没限制)
spring.redis.lettuce.pool.max-idle=5
spring.redis.lettuce.pool.min-idle=0
#最小空闲
```

```
#配置mapper xml文件的路径
mybatis-plus.mapper-locations=classpath:com/atguigu/aclservice/mapper/xml/*.xml

#指定注册中心地址
eureka.client.service-url.defaultZone=http://127.0.0.1:8761/eureka/
#eureka服务器上获取的是服务器的ip地址，否则是主机名
eureka.instance.prefer-ip-address=true

#mybatis日志
mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl
```

# 二、开发权限管理接口

## 1、获取所有菜单

### （1）controller

```
@RestController
@RequestMapping("/admin/acl/permission")
@CrossOrigin
public class PermissionController {

    @Autowired
    private PermissionService permissionService;

    //获取全部菜单
    @GetMapping
    public R indexAllPermission() {
        List<Permission> list =  permissionService.queryAllMenu();
        return R.ok().data("children",list);
    }
}
```

### (2) service

```
//获取全部菜单
@Override
```

```
public List<Permission> queryAllMenu() {

    QueryWrapper<Permission> wrapper = new QueryWrapper<>();
    wrapper.orderByDesc("id");
    List<Permission> permissionList = baseMapper.selectList(wrapper);

    List<Permission> result = bulid(permissionList);
    return result;
}
```

（3）在**Permission**实体类添加属性

```
@ApiModelProperty(value = "层级")
@TableField(exist = false)
private Integer level;

@ApiModelProperty(value = "下级")
@TableField(exist = false)
private List<Permission> children;

@ApiModelProperty(value = "是否选中")
@TableField(exist = false)
private boolean isSelect;
```

（4）编写工具类，根据菜单构建数据

```
package com.atguigu.aclservice.helper;

import com.atguigu.aclservice.entity.Permission;
import java.util.ArrayList;
import java.util.List;
/**
 * <p>
 * 根据权限数据构建菜单数据
 * </p>
 */
public class PermissionHelper {
```

```
12
13      /**
14       * 使用递归方法建菜单
15       * @param treeNodes
16       * @return
17       */
18      public static List<Permission> bulid(List<Permission> treeNodes) {
19          List<Permission> trees = new ArrayList<>();
20          for (Permission treeNode : treeNodes) {
21              if ("0".equals(treeNode.getPid())) {
22                  treeNode.setLevel(1);
23                  trees.add(findChildren(treeNode,treeNodes));
24              }
25          }
26          return trees;
27      }
28
29      /**
30       * 递归查找子节点
31       * @param treeNodes
32       * @return
33       */
34      public static Permission findChildren(Permission treeNode,List<Permission>
treeNodes) {
35          treeNode.setChildren(new ArrayList<Permission>());
36
37          for (Permission it : treeNodes) {
38              if(treeNode.getId().equals(it.getPid())) {
39                  int level = treeNode.getLevel() + 1;
40                  it.setLevel(level);
41                  if (treeNode.getChildren() == null) {
42                      treeNode.setChildren(new ArrayList<>());
43                  }
44                  treeNode.getChildren().add(findChildren(it,treeNodes));
45              }
46          }
47          return treeNode;
48      }
49  }
```

## 2、递归删除菜单

### （1）controller

```
@ApiOperation(value = "递归删除菜单")
@DeleteMapping("remove/{id}")
public R remove(@PathVariable String id) {
    permissionService.removeChildById(id);
    return R.ok();
}
```

### (2) service

```
//递归删除菜单
@Override
public void removeChildById(String id) {
    List<String> idList = new ArrayList<>();
    this.selectChildListById(id, idList);
    //把根据节点id放到list中
    idList.add(id);
    baseMapper.deleteBatchIds(idList);
}

/**
     *  递归获取子节点
     * @param id
     * @param idList
     */
private void selectChildListById(String id, List<String> idList) {
    List<Permission> childList = baseMapper.selectList(new
QueryWrapper<Permission>().eq("pid", id).select("id"));
    childList.stream().forEach(item -> {
        idList.add(item.getId());
        this.selectChildListById(item.getId(), idList);
    });
}
```

## 3、给角色分配权限

### （1）controller

```
@ApiOperation(value = "给角色分配权限")
@PostMapping("/doAssign")
public R doAssign(String roleId,String[] permissionId) {
    permissionService.saveRolePermissionRealtionShip(roleId,permissionId);
    return R.ok();
}
```

### （2）service

```
//给角色分配权限
@Override
public void saveRolePermissionRealtionShip(String roleId, String[] permissionIds)
{

    rolePermissionService.remove(new QueryWrapper<RolePermission>().eq("role_id",
roleId));

    List<RolePermission> rolePermissionList = new ArrayList<>();
    for(String permissionId : permissionIds) {
        if(StringUtils.isEmpty(permissionId)) continue;
        RolePermission rolePermission = new RolePermission();
        rolePermission.setRoleId(roleId);
        rolePermission.setPermissionId(permissionId);
        rolePermissionList.add(rolePermission);
    }
    rolePermissionService.saveBatch(rolePermissionList);
}
```

# 一、Spring Security介绍

## 1、框架介绍

Spring 是一个非常流行和成功的 Java 应用开发框架。Spring Security 基于 Spring 框架，提供了一套 Web 应用安全性的完整解决方案。一般来说，Web 应用的安全性包括用户认证（**Authentication**）和用户授权（**Authorization**）两个部分。

（1）用户认证指的是：验证某个用户是否为系统中的合法主体，也就是说用户能否访问该系统。用户认证一般要求用户提供用户名和密码。系统通过校验用户名和密码来完成认证过程。

（2）用户授权指的是验证某个用户是否有权限执行某个操作。在一个系统中，不同用户所具有的权限是不同的。比如对一个文件来说，有的用户只能进行读取，而有的用户可以进行修改。一般来说，系统会为不同的用户分配不同的角色，而每个角色则对应一系列的权限。

**Spring Security**其实就是用**filter**，多请求的路径进行过滤。

（1）如果是基于Session，那么Spring-security会对cookie里的sessionid进行解析，找到服务器存储的sesion信息，然后判断当前用户是否符合请求的要求。

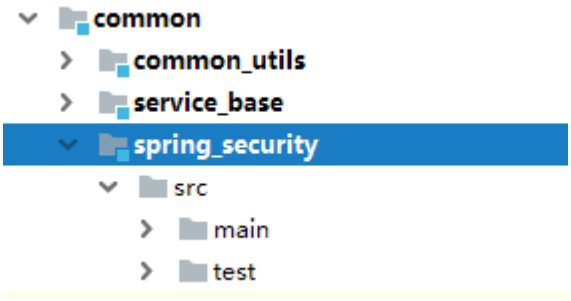
（2）如果是token，则是解析出token，然后将当前请求加入到Spring-security管理的权限信息中去

## 2、认证与授权实现思路

如果系统的模块众多，每个模块都需要就行授权与认证，所以我们选择基于token的形式进行授权与认证，用户根据用户名密码认证成功，然后获取当前用户角色的一系列权限值，并以用户名为key，权限列表为value的形式存入redis缓存中，根据用户名相关信息生成token返回，浏览器将token记录到cookie中，每次调用api接口都默认将token携带到header请求头中，Spring-security解析header头获取token信息，解析token获取当前用户名，根据用户名就可以从redis中获取权限列表，这样Spring-security就能够判断当前请求是否有权限访问

# 二、整合Spring Security

## 1、在common下创建spring_security模块




## 2、在spring_security引入相关依赖

```
<dependencies>
    <dependency>
        <groupId>com.atguigu</groupId>
        <artifactId>common_utils</artifactId>
        <version>0.0.1-SNAPSHOT</version>
    </dependency>

    <!-- Spring Security依赖 -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>

    <dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt</artifactId>
    </dependency>
</dependencies>
```

## 3、在service_acl引入spring_security依赖

```
<dependency>
    <groupId>com.atguigu</groupId>
    <artifactId>spring_security</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</dependency>
```

## 0、代码结构说明：


## 4、创建**spring security**核心配置类


Spring Security的核心配置就是继承WebSecurityConfigurerAdapter并注解@EnableWebSecurity的配置。

这个配置指明了用户名密码的处理方式、请求路径的开合、登录登出控制等和安全相关的配置

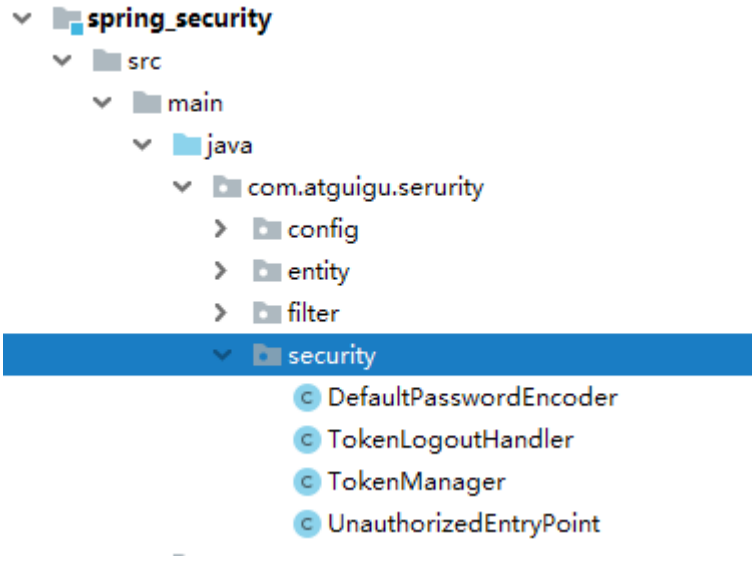
```
import com.atguigu.serurity.filter.TokenAuthenticationFilter;
import com.atguigu.serurity.filter.TokenLoginFilter;
import com.atguigu.serurity.security.DefaultPasswordEncoder;
import com.atguigu.serurity.security.TokenLogoutHandler;
import com.atguigu.serurity.security.TokenManager;
```

```
import com.atguigu.serurity.security.UnauthorizedEntryPoint;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.core.RedisTemplate;
import
org.springframework.security.config.annotation.authentication.builders.Authenticat
ionManagerBuilder;
import
org.springframework.security.config.annotation.method.configuration.EnableGlobalMe
thodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity
;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfig
urerAdapter;
import org.springframework.security.core.userdetails.UserDetailsService;

/**
 * <p>
 * Security配置类
 * </p>
 */
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class TokenWebSecurityConfig extends WebSecurityConfigurerAdapter {

    private UserDetailsService userDetailsService;
    private TokenManager tokenManager;
    private DefaultPasswordEncoder defaultPasswordEncoder;
    private RedisTemplate redisTemplate;

    @Autowired
    public TokenWebSecurityConfig(UserDetailsService userDetailsService,
DefaultPasswordEncoder defaultPasswordEncoder,
                                  TokenManager tokenManager, RedisTemplate
redisTemplate) {
        this.userDetailsService = userDetailsService;
        this.defaultPasswordEncoder = defaultPasswordEncoder;
        this.tokenManager = tokenManager;
        this.redisTemplate = redisTemplate;
```

```
    }

    /**
     * 配置设置
     * @param http
     * @throws Exception
     */
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.exceptionHandling()
                .authenticationEntryPoint(new UnauthorizedEntryPoint())
                .and().csrf().disable()
                .authorizeRequests()
                .anyRequest().authenticated()
                .and().logout().logoutUrl("/admin/acl/index/logout")
                .addLogoutHandler(new
TokenLogoutHandler(tokenManager,redisTemplate)).and()
                .addFilter(new TokenLoginFilter(authenticationManager(),
tokenManager, redisTemplate))
                .addFilter(new TokenAuthenticationFilter(authenticationManager(),
tokenManager, redisTemplate)).httpBasic();
    }

    /**
     * 密码处理
     * @param auth
     * @throws Exception
     */
    @Override
    public void configure(AuthenticationManagerBuilder auth) throws Exception {

 auth.userDetailsService(userDetailsService).passwordEncoder(defaultPasswordEncode
r);
    }

    /**
     * 配置哪些请求不拦截
     * @param web
     * @throws Exception
     */
    @Override
    public void configure(WebSecurity web) throws Exception {
        web.ignoring().antMatchers("/api/**",
                "/swagger-resources/**", "/webjars/**", "/v2/**", "/swagger-
```

```
ui.html/**"
                );
    }
}
```

## 5、创建认证授权相关的工具类




（1）DefaultPasswordEncoder：密码处理的方法

```
package com.atguigu.serurity.security;

import com.atguigu.commonutils.utils.MD5;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Component;

/**
 * <p>
 * 密码的处理方法类型
 * </p>
 */
@Component
public class DefaultPasswordEncoder implements PasswordEncoder {

    public DefaultPasswordEncoder() {
        this(-1);
    }

    /**
     * @param strength
```

```
     *            the log rounds to use, between 4 and 31
     */
    public DefaultPasswordEncoder(int strength) {

    }

    public String encode(CharSequence rawPassword) {
        return MD5.encrypt(rawPassword.toString());
    }

    public boolean matches(CharSequence rawPassword, String encodedPassword) {
        return encodedPassword.equals(MD5.encrypt(rawPassword.toString()));
    }
}
```

（2）TokenManager：token操作的工具类

```
import io.jsonwebtoken.CompressionCodecs;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.stereotype.Component;

import java.util.Date;

/**
 * <p>
 * token管理
 * </p>
 */
@Component
public class TokenManager {

    private long tokenExpiration = 24*60*60*1000;
    private String tokenSignKey = "123456";
```

```
18
19      public String createToken(String username) {
20          String token = Jwts.builder().setSubject(username)
21                  .setExpiration(new Date(System.currentTimeMillis() +
tokenExpiration))
22                  .signWith(SignatureAlgorithm.HS512,
tokenSignKey).compressWith(CompressionCodecs.GZIP).compact();
23          return token;
24      }
25
26      public String getUserFromToken(String token) {
27          String user =
Jwts.parser().setSigningKey(tokenSignKey).parseClaimsJws(token).getBody().getSubje
ct();
28          return user;
29      }
30
31      public void removeToken(String token) {
32          //jwttoken无需删除，客户端扔掉即可。
33      }
34  }
```

### （3）TokenLogoutHandler：退出实现

```
1  import com.atguigu.commonutils.R;
2  import com.atguigu.commonutils.utils.ResponseUtil;
3  import org.springframework.data.redis.core.RedisTemplate;
4  import org.springframework.security.core.Authentication;
5  import org.springframework.security.web.authentication.logout.LogoutHandler;
6
7  import javax.servlet.http.HttpServletRequest;
8  import javax.servlet.http.HttpServletResponse;
9
10 /**
11  * <p>
12  * 登出业务逻辑类
13  * </p>
14  */
15 public class TokenLogoutHandler implements LogoutHandler {
16
```

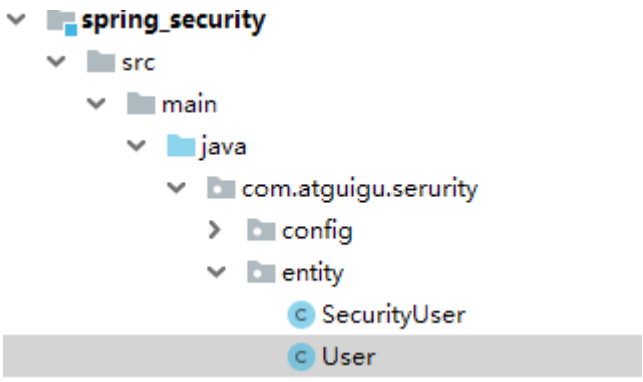
```
    private TokenManager tokenManager;
    private RedisTemplate redisTemplate;

    public TokenLogoutHandler(TokenManager tokenManager, RedisTemplate
redisTemplate) {
        this.tokenManager = tokenManager;
        this.redisTemplate = redisTemplate;
    }

    @Override
    public void logout(HttpServletRequest request, HttpServletResponse response,
Authentication authentication) {
        String token = request.getHeader("token");
        if (token != null) {
            tokenManager.removeToken(token);

            //清空当前用户缓存中的权限数据
            String userName = tokenManager.getUserFromToken(token);
            redisTemplate.delete(userName);
        }
        ResponseUtil.out(response, R.ok());
    }
}
```

(4) UnauthorizedEntryPoint：未授权统一处理

```
import com.atguigu.commonutils.R;
import com.atguigu.commonutils.utils.ResponseUtil;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.AuthenticationEntryPoint;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
/**
 * <p>
 * 未授权的统一处理方式
 * </p>
 */
public class UnauthorizedEntryPoint implements AuthenticationEntryPoint {
```

```
15
16      @Override
17      public void commence(HttpServletRequest request, HttpServletResponse response,
18                           AuthenticationException authException) throws
IOException, ServletException {
19
20          ResponseUtil.out(response, R.error());
21      }
22  }
```

## 6、创建认证授权实体类




### （1）SecutityUser

```
1  import lombok.Data;
2  import lombok.extern.slf4j.Slf4j;
3  import org.springframework.security.core.GrantedAuthority;
4  import org.springframework.security.core.authority.SimpleGrantedAuthority;
5  import org.springframework.security.core.userdetails.UserDetails;
6  import org.springframework.util.StringUtils;
7
8  import java.util.ArrayList;
9  import java.util.Collection;
10 import java.util.List;
11
12 /**
13  * <p>
14  * 安全认证用户详情信息
15  * </p>
16  */
```

```java
@Data
@Slf4j
public class SecurityUser implements UserDetails {

    //当前登录用户
    private transient User currentUserInfo;

    //当前权限
    private List<String> permissionValueList;

    public SecurityUser() {
    }

    public SecurityUser(User user) {
        if (user != null) {
            this.currentUserInfo = user;
        }
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        Collection<GrantedAuthority> authorities = new ArrayList<>();
        for(String permissionValue : permissionValueList) {
            if(StringUtils.isEmpty(permissionValue)) continue;
            SimpleGrantedAuthority authority = new
SimpleGrantedAuthority(permissionValue);
            authorities.add(authority);
        }

        return authorities;
    }

    @Override
    public String getPassword() {
        return currentUserInfo.getPassword();
    }

    @Override
    public String getUsername() {
        return currentUserInfo.getUsername();
    }

    @Override
```

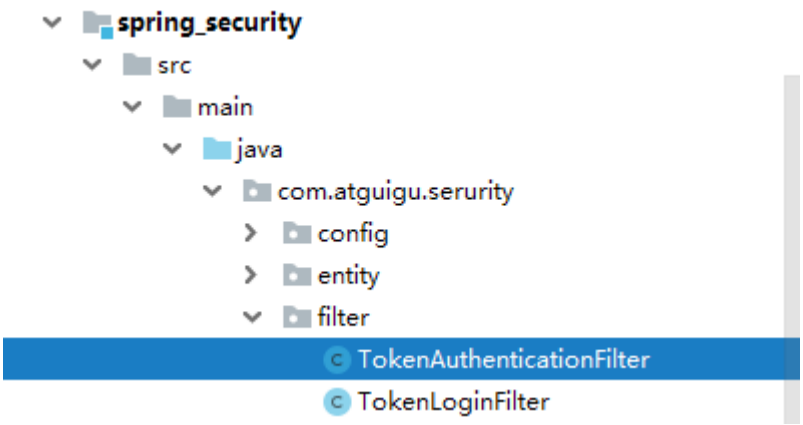
```
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }
}
```

## （2）User

```
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;
import java.io.Serializable;

/**
 * <p>
 * 用户实体类
 * </p>
 */
@Data
```

```java
@ApiModel(description = "用户实体类")
public class User implements Serializable {

    private static final long serialVersionUID = 1L;

    @ApiModelProperty(value = "微信openid")
    private String username;

    @ApiModelProperty(value = "密码")
    private String password;

    @ApiModelProperty(value = "昵称")
    private String nickName;

    @ApiModelProperty(value = "用户头像")
    private String salt;

    @ApiModelProperty(value = "用户签名")
    private String token;
}
```

## 7、创建认证和授权的**filter**




### （1）**TokenLoginFilter**：认证的**filter**

```java
import com.atguigu.commonutils.R;
import com.atguigu.commonutils.utils.ResponseUtil;
import com.atguigu.serurity.entity.SecurityUser;
import com.atguigu.serurity.entity.User;
```

```
import com.atguigu.serurity.security.TokenManager;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilt
er;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.ArrayList;

/**
 * <p>
 * 登录过滤器，继承UsernamePasswordAuthenticationFilter，对用户名密码进行登录校验
 * </p>
 */
public class TokenLoginFilter extends UsernamePasswordAuthenticationFilter {

    private AuthenticationManager authenticationManager;
    private TokenManager tokenManager;
    private RedisTemplate redisTemplate;

    public TokenLoginFilter(AuthenticationManager authenticationManager,
TokenManager tokenManager, RedisTemplate redisTemplate) {
        this.authenticationManager = authenticationManager;
        this.tokenManager = tokenManager;
        this.redisTemplate = redisTemplate;
        this.setPostOnly(false);
        this.setRequiresAuthenticationRequestMatcher(new
AntPathRequestMatcher("/admin/acl/login","POST"));
    }

    @Override
    public Authentication attemptAuthentication(HttpServletRequest req,
HttpServletResponse res)
```

```
            throws AuthenticationException {
        try {
            User user = new ObjectMapper().readValue(req.getInputStream(),
User.class);

            return authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(user.getUsername(), user.getPassword(), new
ArrayList<>()));
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

    }

    /**
     * 登录成功
     * @param req
     * @param res
     * @param chain
     * @param auth
     * @throws IOException
     * @throws ServletException
     */
    @Override
    protected void successfulAuthentication(HttpServletRequest req,
HttpServletResponse res, FilterChain chain,
                                            Authentication auth) throws
IOException, ServletException {
        SecurityUser user = (SecurityUser) auth.getPrincipal();
        String token =
tokenManager.createToken(user.getCurrentUserInfo().getUsername());
        redisTemplate.opsForValue().set(user.getCurrentUserInfo().getUsername(),
user.getPermissionValueList());

        ResponseUtil.out(res, R.ok().data("token", token));
    }

    /**
     * 登录失败
     * @param request
     * @param response
     * @param e
     * @throws IOException
     * @throws ServletException
```

```
     */
    @Override
    protected void unsuccessfulAuthentication(HttpServletRequest request,
HttpServletResponse response,
                                              AuthenticationException e) throws
IOException, ServletException {
        ResponseUtil.out(response, R.error());
    }
}
```

（2）TokenAuthenticationFilter：

授权filter

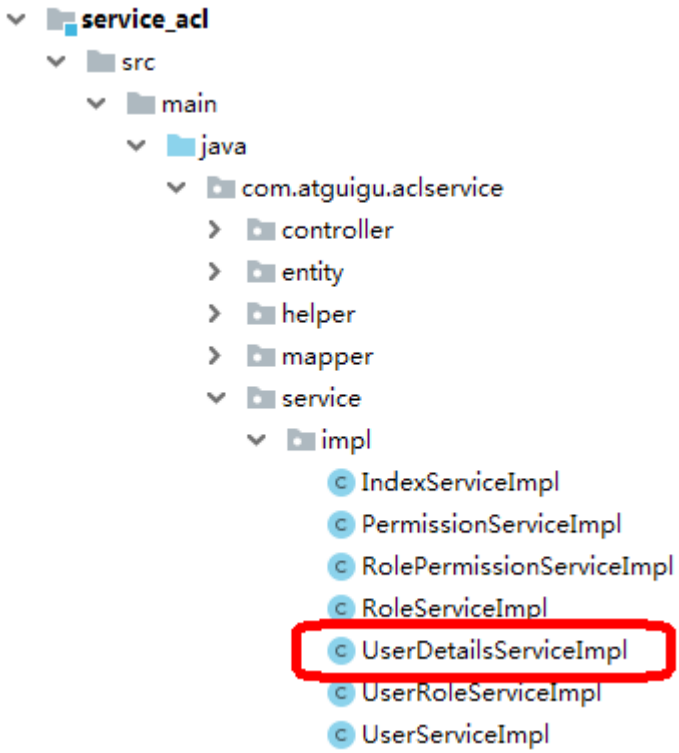
```
package com.atguigu.serurity.filter;

import com.atguigu.commonutils.R;
import com.atguigu.commonutils.utils.ResponseUtil;
import com.atguigu.serurity.security.TokenManager;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.context.SecurityContextHolder;
import
org.springframework.security.web.authentication.www.BasicAuthenticationFilter;
import org.springframework.util.StringUtils;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

/**
```

```
 * <p>
 * 访问过滤器
 * </p>
 */
public class TokenAuthenticationFilter extends BasicAuthenticationFilter {
    private TokenManager tokenManager;
    private RedisTemplate redisTemplate;

    public TokenAuthenticationFilter(AuthenticationManager authManager,
TokenManager tokenManager,RedisTemplate redisTemplate) {
        super(authManager);
        this.tokenManager = tokenManager;
        this.redisTemplate = redisTemplate;
    }

    @Override
    protected void doFilterInternal(HttpServletRequest req, HttpServletResponse
res, FilterChain chain)
            throws IOException, ServletException {
        logger.info("=================="+req.getRequestURI());
        if(req.getRequestURI().indexOf("admin") == -1) {
            chain.doFilter(req, res);
            return;
        }

        UsernamePasswordAuthenticationToken authentication = null;
        try {
            authentication = getAuthentication(req);
        } catch (Exception e) {
            ResponseUtil.out(res, R.error());
        }

        if (authentication != null) {
            SecurityContextHolder.getContext().setAuthentication(authentication);
        } else {
            ResponseUtil.out(res, R.error());
        }
        chain.doFilter(req, res);
    }

    private UsernamePasswordAuthenticationToken
getAuthentication(HttpServletRequest request) {
        // token置于header里
```

```
        String token = request.getHeader("token");
        if (token != null && !"".equals(token.trim())) {
            String userName = tokenManager.getUserFromToken(token);

            List<String> permissionValueList = (List<String>)
redisTemplate.opsForValue().get(userName);
            Collection<GrantedAuthority> authorities = new ArrayList<>();
            for(String permissionValue : permissionValueList) {
                if(StringUtils.isEmpty(permissionValue)) continue;
                SimpleGrantedAuthority authority = new
SimpleGrantedAuthority(permissionValue);
                authorities.add(authority);
            }

            if (!StringUtils.isEmpty(userName)) {
                return new UsernamePasswordAuthenticationToken(userName, token,
authorities);
            }
            return null;
        }
        return null;
    }
}
```

# 一、创建自定义查询用户类

（1）在**service_acl**模块创建，因为其他模板不会用到

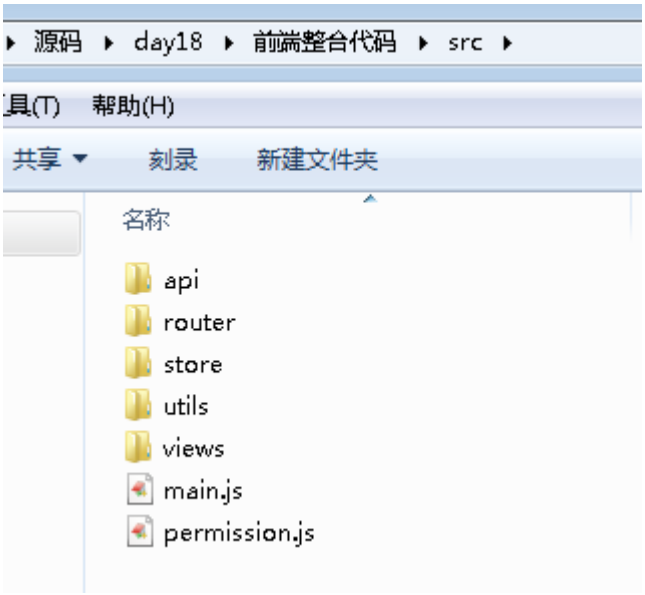



# 二、后端接口和前端页面对接

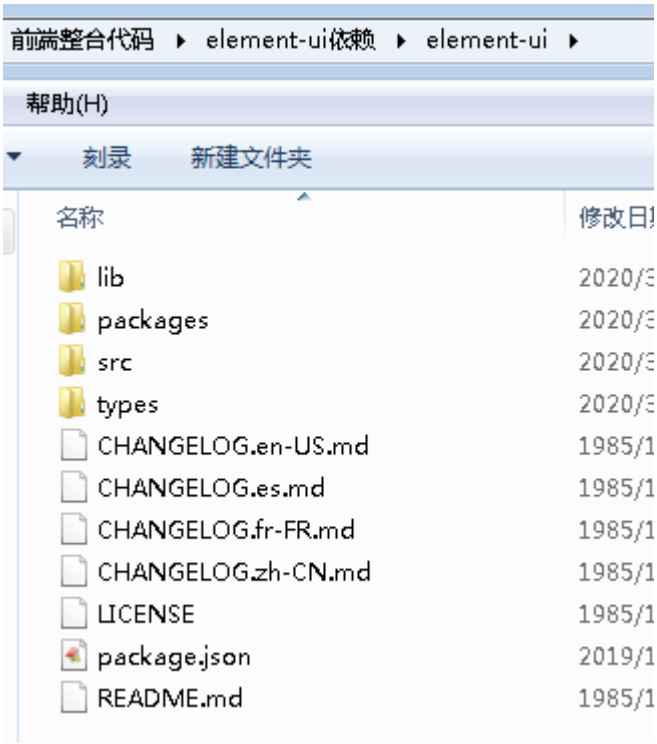
1、在前端项目中下载依赖

```
npm install --save vuex-persistedstate
```

2、替换相关文件




3、在node_modules文件夹中替换element-ui依赖



# 一、配置中心介绍

## 1、Spring Cloud Config

Spring Cloud Config 为分布式系统的外部配置提供了服务端和客户端的支持方案。在配置的服务端您可以在所有环境中为应用程序管理外部属性的中心位置。客户端和服务端概念上的Spring Environment 和 PropertySource 抽象保持同步，它们非常适合Spring应用程序，但是可以与任何语言中运行的应用程序一起使用。当应用程序在部署管道中从一个开发到测试直至进入生产时，您可以管理这些环境之间的配置，并确保应用程序在迁移时具有它们需要运行的所有内容。服务器存储后端的默认实现使用git，因此它很容易支持标记版本的配置环境，并且能够被管理内容的各种工具访问。很容易添加替代的实现，并用Spring配置将它们插入。

Spring Cloud Config 包含了Client和Server两个部分，server提供配置文件的存储、以接口的形式将配置文件的内容提供出去，client通过接口获取数据、并依据此数据初始化自己的应用。Spring cloud使用git或svn存放配置文件，默认情况下使用git。

## 2、Nacos替换Config

Nacos 可以与 Spring, Spring Boot, Spring Cloud 集成，并能代替 Spring Cloud Eureka, Spring Cloud Config。通过 Nacos Server 和 spring-cloud-starter-alibaba-nacos-config 实现配置的动态变更。

### （1）应用场景

在系统开发过程中，开发者通常会将一些需要变更的参数、变量等从代码中分离出来独立管理，以独立的配置文件的形式存在。目的是让静态的系统工件或者交付物（如 WAR，JAR 包等）更好地和实际的物理运行环境进行适配。配置管理一般包含在系统部署的过程中，由系统管理员或者运维人员完成。配置变更是调整系统运行时的行为的有效手段。

如果微服务架构中没有使用统一配置中心时，所存在的问题:

- 配置文件分散在各个项目里，不方便维护
- 配置内容安全与权限
- 更新配置后，项目需要重启

nacos配置中心: 系统配置的集中管理（编辑、存储、分发）、动态更新不重启、回滚配置（变更管理、历史版本管理、变更审计）等所有与配置相关的活动。

# Nacos

# 二、读取　　配置中心的配置文件

## 1、在Nacos创建统一配置文件

（1）点击创建按钮


（2）输入配置信息


a）Data ID 的完整规则格式如下

${prefix}-${spring.profile.active}.${file-extension}

- prefix 默认为所属工程配置spring.application.name 的值（即：nacos-provider），也可以通过配置项 spring.cloud.nacos.config.prefix来配置。

- spring.profiles.active=dev 即为当前环境对应的 profile。 注意：当 spring.profiles.active 为空时，对应的连接符 - 也将不存在，dataId 的拼接格式变成 ${prefix}.${file-extension}

- file-exetension 为配置内容的数据格式，可以通过配置项 spring.cloud.nacos.config.file-extension 来配置。目前只支持 properties 和 yaml 类型。

## 2、以service-statistics模块为例

### （1）在service中引入依赖

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId>
</dependency>
```

### （2）创建bootstrap.properties配置文件

```
#配置中心地址
spring.cloud.nacos.config.server-addr=127.0.0.1:8848

#spring.profiles.active=dev

# 该配置影响统一配置中心中的dataId
spring.application.name=service-statistics
```

### （3）把项目之前的application.properties内容注释，启动项目查看效果

## 3、补充：springboot配置文件加载顺序

其实yml和properties文件是一样的原理，且一个项目上要么yml或者properties，二选一的存在。推荐使用yml，更简洁。

bootstrap与application

（**1**）加载顺序

这里主要是说明application和bootstrap的加载顺序。

bootstrap.yml（bootstrap.properties）先加载
application.yml（application.properties）后加载
bootstrap.yml 用于应用程序上下文的引导阶段。

bootstrap.yml 由父Spring ApplicationContext加载。

父ApplicationContext 被加载到使用 application.yml 的之前。

（**2**）配置区别

bootstrap.yml 和application.yml 都可以用来配置参数。

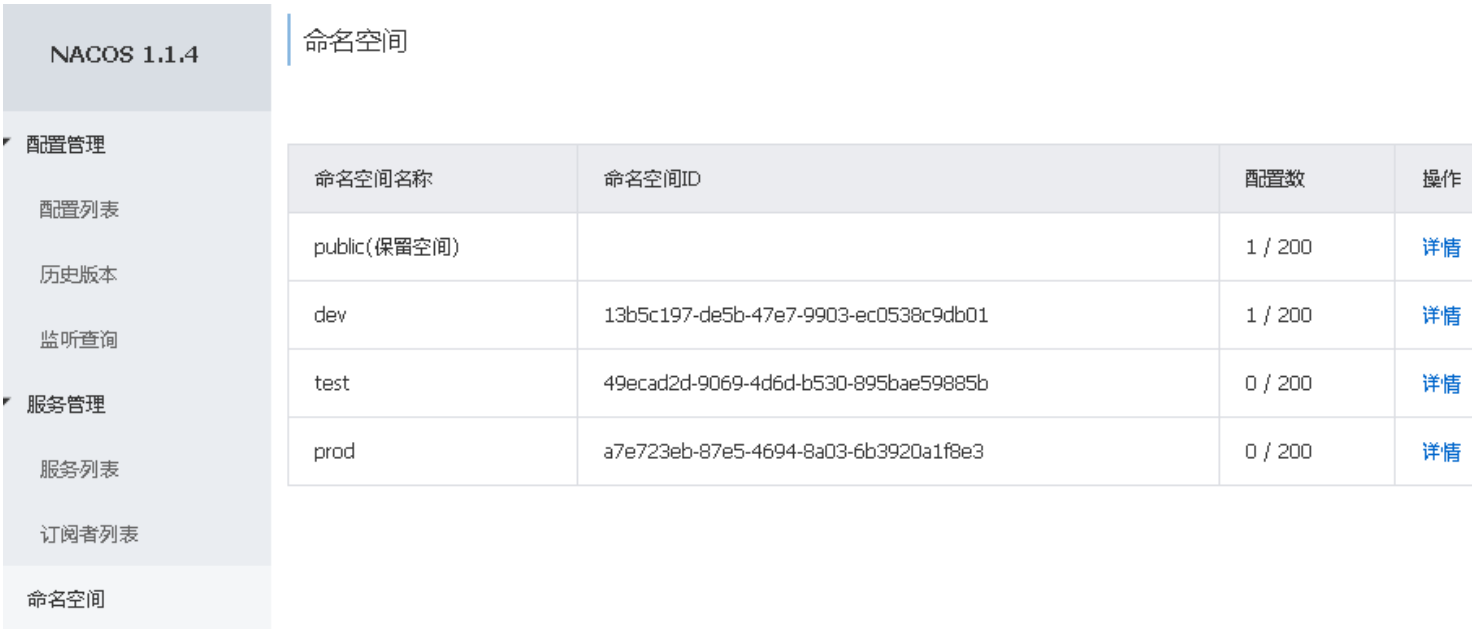
bootstrap.yml 可以理解成系统级别的一些参数配置，这些参数一般是不会变动的。
application.yml 可以用来定义应用级别的。

# 三、名称空间切换环境

在实际开发中，通常有多套不同的环境（默认只有public），那么这个时候可以根据指定的环境来创建不同的 namespce，例如，开发、测试和生产三个不同的环境，那么使用一套 nacos 集群可以分别建以下三个不同的 namespace。以此来实现多环境的隔离。

## 1、创建命名空间


**默认只有public**，新建了**dev**、**test**和**prod**命名空间




## 2、克隆配置

（1）切换到配置列表：


可以发现有四个名称空间：**public**（默认）以及我们自己添加的**3**个名称空间（**prod**、**dev**、**test**），可以点击查看每个名称空间下的配置文件，当然现在只有**public**下有一个配置。

默认情况下，项目会到**public**下找 服务名**.properties**文件。

接下来，在**dev**名称空间中也添加一个**nacos-provider.properties**配置。这时有两种方式：

第一，切换到**dev**名称空间，添加一个新的配置文件。缺点：每个环境都要重复配置类似的项目

第二，直接通过**clone**方式添加配置，并修改即可。推荐


点击编辑：修改配置内容，端口号改为**8013**以作区分

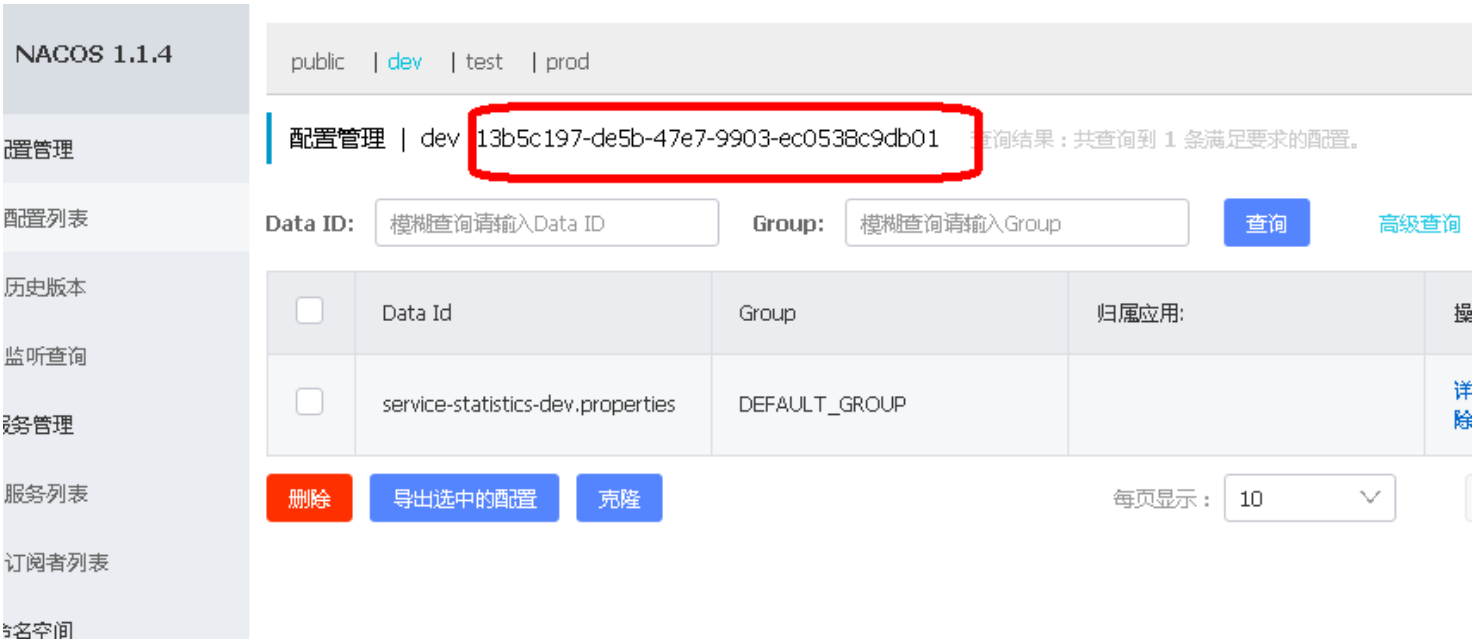
配置内容：

```
# 服务端口
server.port=8013
# 服务名
spring.application.name=service-statistics

# mysql数据库连接
```

在项目模块中，修改**bootstrap.properties**添加如下配置

```
spring.cloud.nacos.config.server-addr=127.0.0.1:8848

spring.profiles.active=dev

# 该配置影响统一配置中心中的dataId，之前已经配置过
spring.application.name=service-statistics

spring.cloud.nacos.config.namespace=13b5c197-de5b-47e7-9903-ec0538c9db01
```

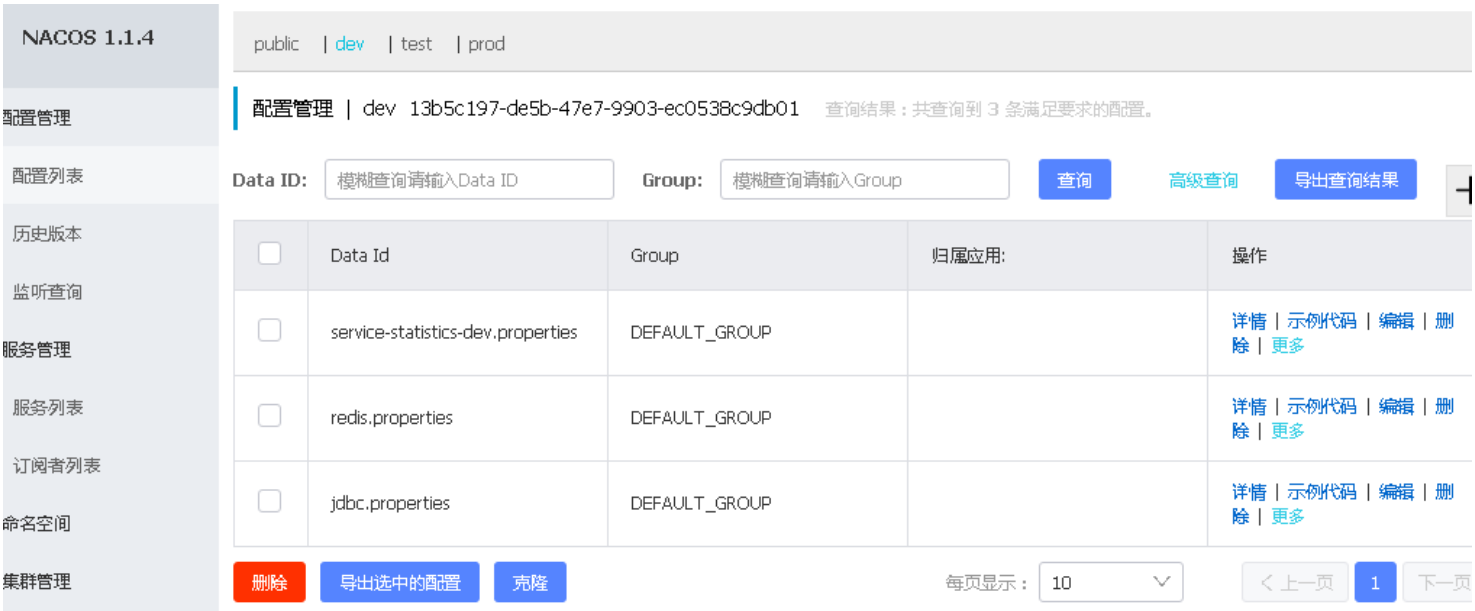
**namespace**的值为：




重启服务提供方服务，测试修改之后是否生效

# 四、多配置文件加载

在一些情况下需要加载多个配置文件。假如现在dev名称空间下有三个配置文件：service-statistics.properties、redis.properties、jdbc.properties




添加配置，加载多个配置文件

```
spring.cloud.nacos.config.server-addr=127.0.0.1:8848
```

```
spring.profiles.active=dev

# 该配置影响统一配置中心中的dataId，之前已经配置过
spring.application.name=service-statistics

spring.cloud.nacos.config.namespace=13b5c197-de5b-47e7-9903-ec0538c9db01

spring.cloud.nacos.config.ext-config[0].data-id=redis.properties
# 开启动态刷新配置，否则配置文件修改，工程无法感知
spring.cloud.nacos.config.ext-config[0].refresh=true
spring.cloud.nacos.config.ext-config[1].data-id=jdbc.properties
spring.cloud.nacos.config.ext-config[1].refresh=true
```

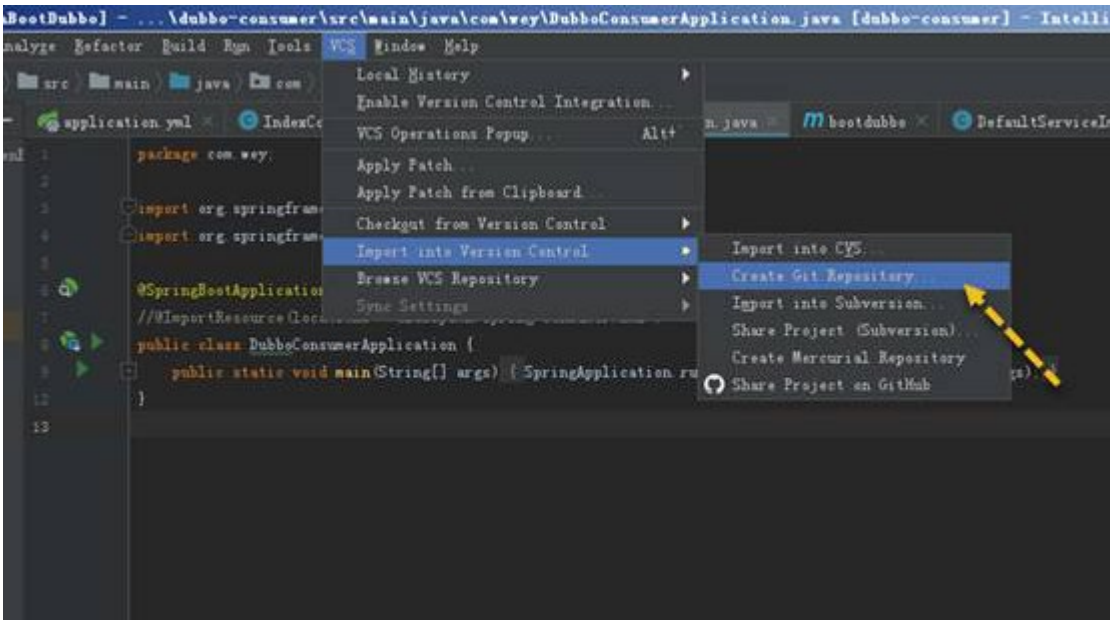
# 1. 准备 Git 仓库

码云：https://gitee.com

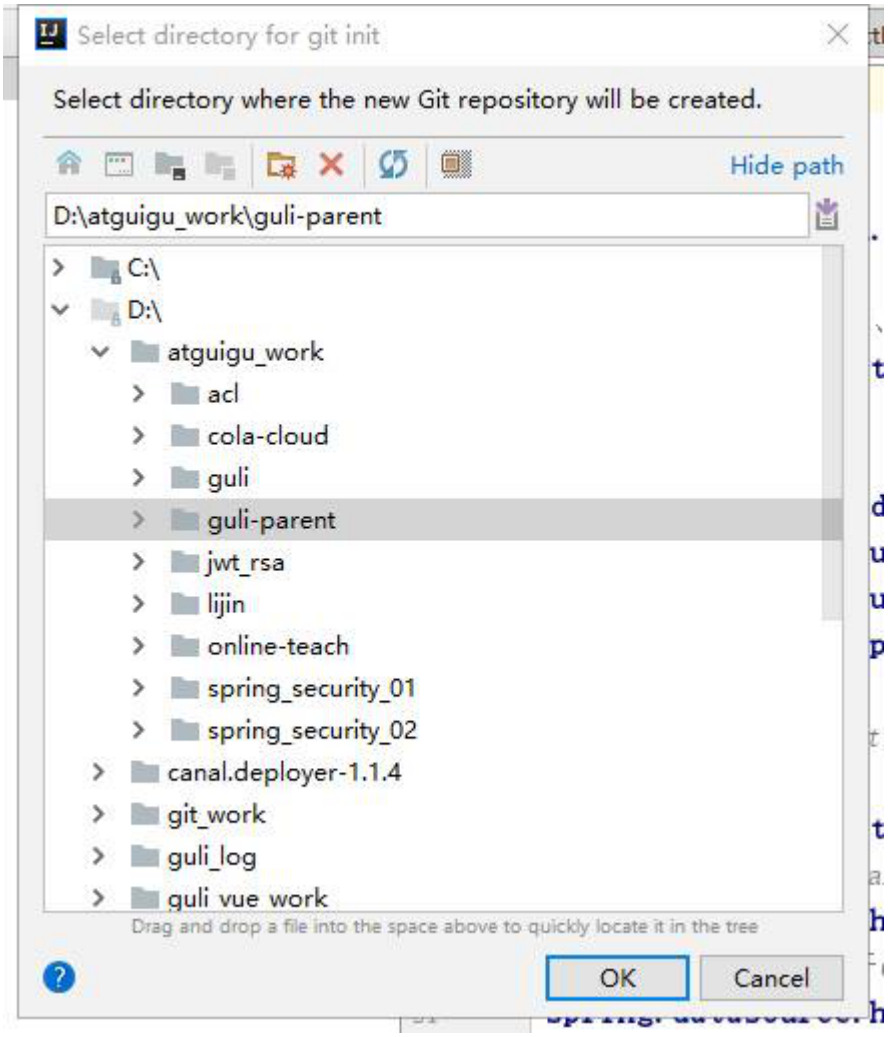
## 1.1. 通过网站右上角的「+」号，选择「新建仓库」，进入新建仓库页面


## 1.2. 新建仓库

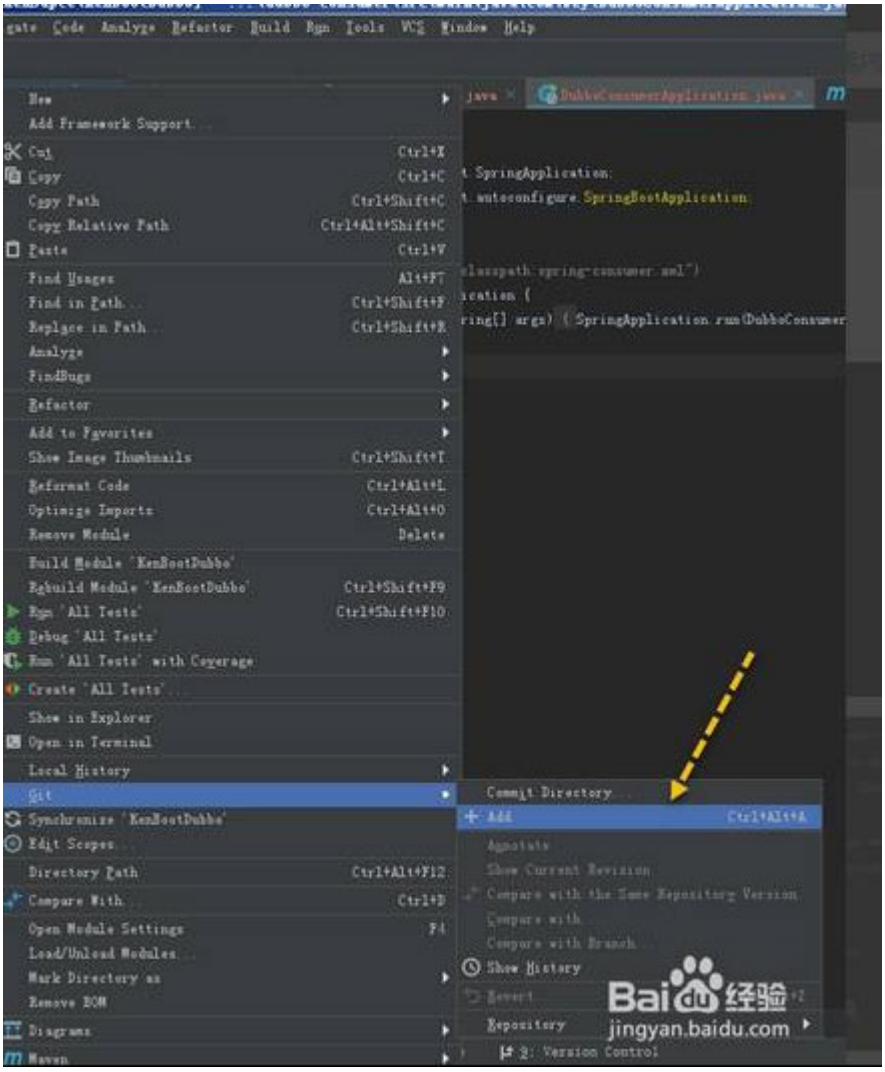
**1.3.** 打开项目并点击菜单栏上的**【CVS】--》【Import into version control】--》【Create Git Repository】**创建本地仓库



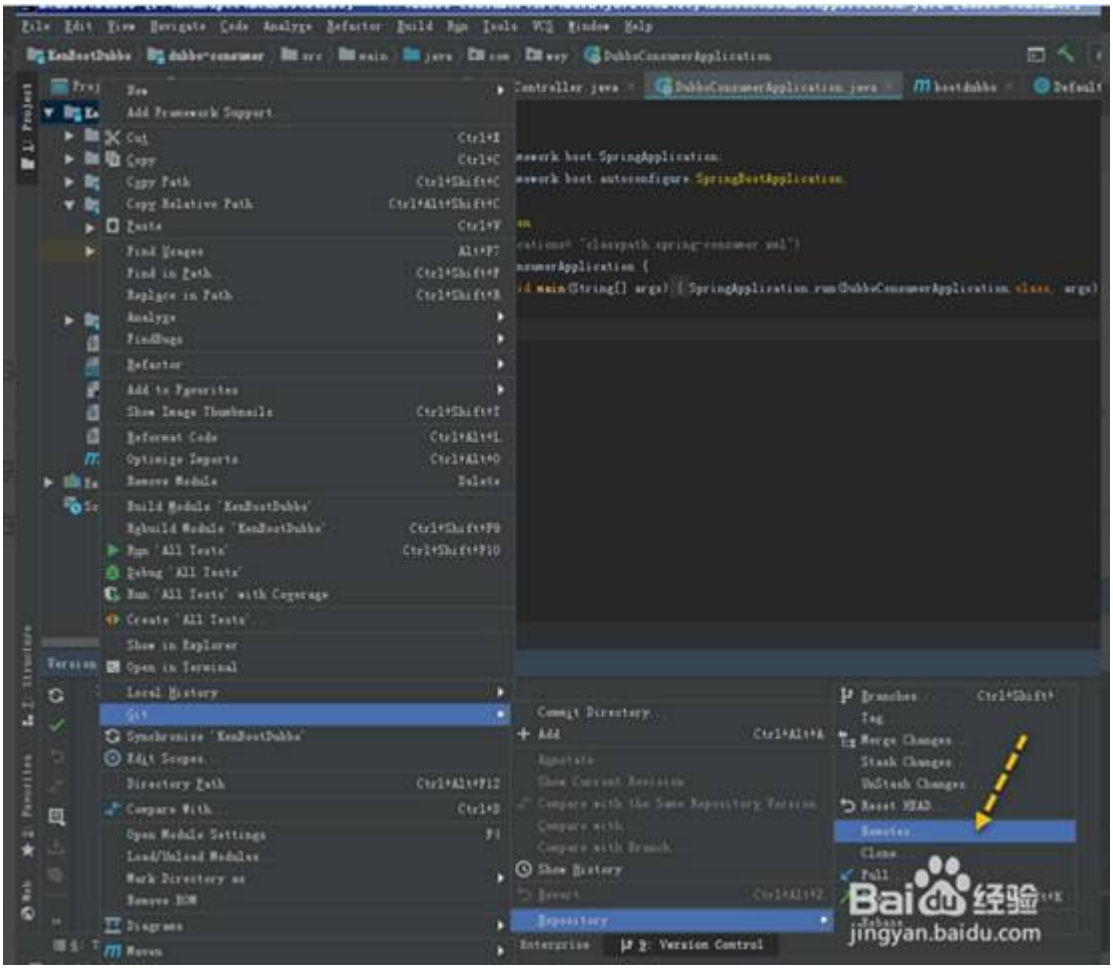

**1.4.** 在打开的**【Create Git Repository】**对话框内选择本地仓库的位置，这里我选择项目的根目录。

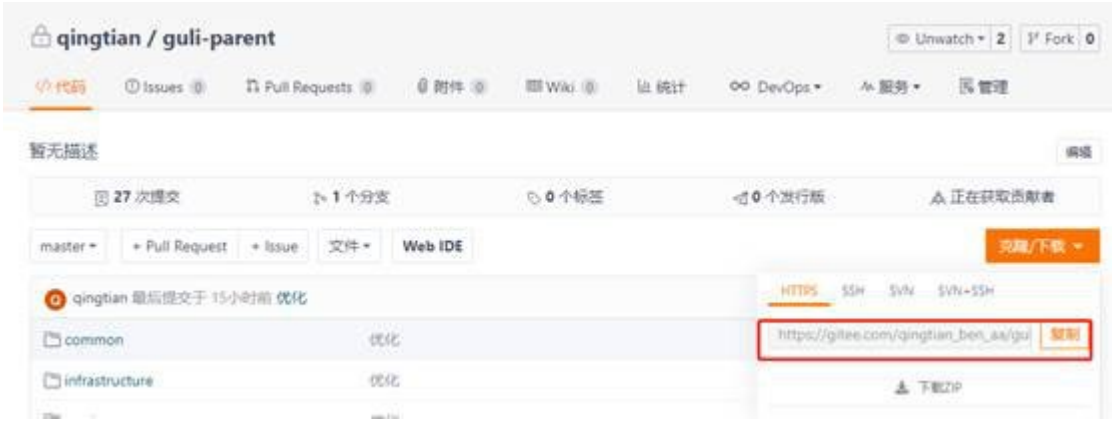


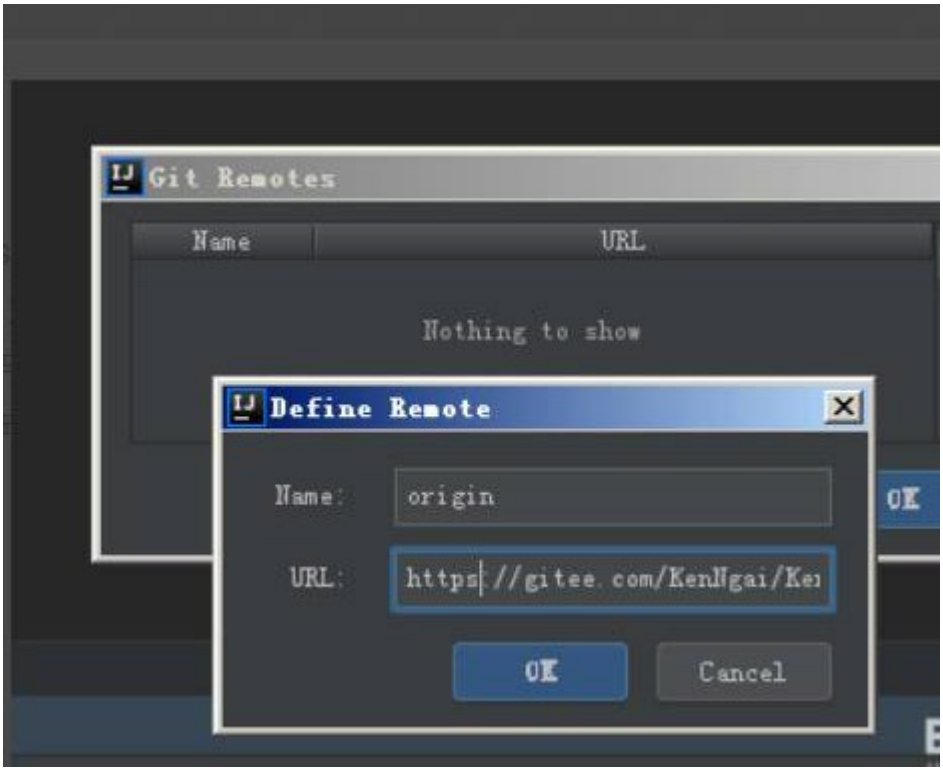

**1.5.** 右击项目点击**【Git】--》【Add】**，接着点击**【Git】--》【Commit Directory】**在打开的窗口中选择要上传到本地仓库的代码并添加注释后提交到本地仓库内。



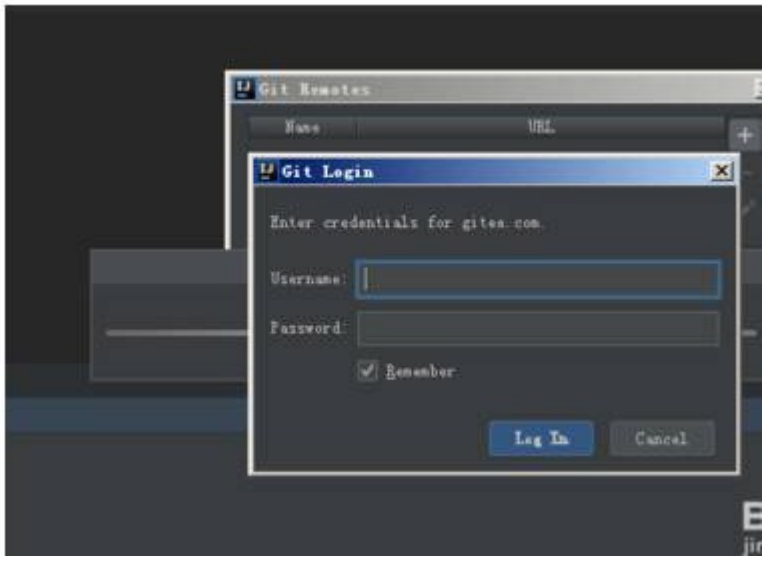

**1.6.** 右击项目点击【Git】--》【Repository】--》【Remotes...】。在打开的【Git Remotes】窗口中添加码云的远程仓库。码云的远程仓库地址可以在码云仓库内找到。


Add Framework Support
Cut
Copy
Copy Path
Copy Relative Path
Paste
Find Usages
Find in Path
Replace in Path
Analyze
Refactor
Add to Favorites
Show Image Thumbnails
Reformat Code
Optimize Imports
Remove Module
Build Module 'KenBootDubbo'
Rebuild Module 'KenBootDubbo'
Run 'All Tests'
Debug 'All Tests'
Run 'All Tests' with Coverage
Create 'All Tests'
Show in Explorer
Open in Terminal
Local History
Git
Synchronize 'KenBootDubbo'
Edit Scopes
Directory Path
Compare With
Open Module Settings
Load/Unload Modules
Mark Directory as
Remove BOM
Diagrams
Maven
Commit Directory...
Add
Show History
Repository
Branches
Merge Changes
Stash Changes
UnStash Changes
Reset HEAD...
Remote
Clone...
Pull
Enterprise
9: Version Control
Baidu 经验
jingyan.baidu.com


qingtian / guli-parent
Unwatch 2
Fork 0
代码
Issues 0
Pull Requests 0
附件 0
Wiki 0
统计
DevOps
服务
管理
暂无描述
编辑
27 次提交
1 个分支
0 个标签
0 个发行版
正在获取贡献者
master
+ Pull Request
+ Issue
文件
Web IDE
克隆/下载
HTTPS
SSH
SVN
SVN+SSH
https://gitee.com/qingtian_ben_aa/gu
复制
下载ZIP
qingtian 最后提交于 15小时前 优化
common
优化
infrastructure
优化




**1.7.** 点击【OK】后接着输入码云的账号密码。

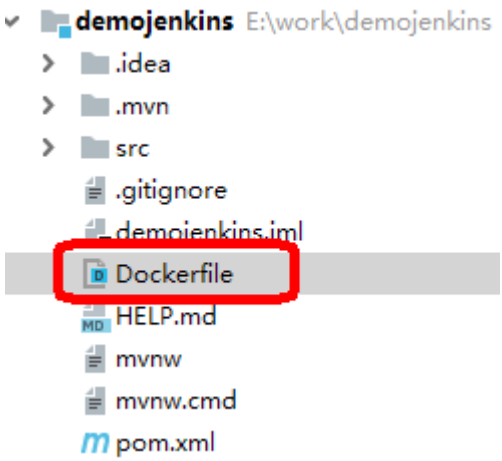



**1.8.** 上传代码到码云，右击项目点击【Git】--》【Repository】--》【Push...】在打开的【Push commits】内可以看到已提交到本地仓库的提交信息。点击【Push】按钮将本地仓库的代码上传到码云上，上传成功后就可以在码云上看到啦。

# 1. 准备代码，提交到码云Git库

代码中需要包含以下几部分内容：

（1）代码中需要包含Dockerfile文件




文件内容：

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
COPY ./target/demojenkins.jar demojenkins.jar
ENTRYPOINT ["java","-jar","/demojenkins.jar", "&"]
```

（2）在项目pom文件中指定打包类型，包含build部分内容

```
<groupId>com.atguigu</groupId>
<artifactId>demojenkins</artifactId>
<packaging>jar</packaging>
<version>0.0.1-SNAPSHOT</version>
<name>demojenkins</name>
<description>Demo project for Spring Boot</description>
```

```
<build>
    <finalName>demojenkins</finalName>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

## 2. 安装JAVA 运行环境

第一步：上传或下载安装包

cd/usr/local

jdk-8u121-linux-x64.tar.gz

第二步：解压安装包

tar -zxvf jdk-8u121-linux-x64.tar.gz

第三步：建立软连接

ln -s /usr/local/jdk1.8.0_121/ /usr/local/jdk

第四步：修改环境变量

vim /etc/profile

export JAVA_HOME=/usr/local/jdk

export JRE_HOME=$JAVA_HOME/jre

export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib

export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin

通过命令source /etc/profile让profile文件立即生效

source /etc/profile

第五步、测试是否安装成功

②、使用java -version，出现版本

# 3. 安装maven

第一步：上传或下载安装包

cd/usr/local

apache-maven-3.6.1-bin.tar.gz

第二步：解压安装包

tar -zxvf apache-maven-3.6.1-bin.tar.gz

第三步：建立软连接

ln -s /usr/local/apache-maven-3.6.1/ /usr/local/maven

第四步：修改环境变量

vim /etc/profile

export MAVEN_HOME=/usr/local/maven

export PATH=$PATH:$MAVEN_HOME/bin

通过命令source /etc/profile让profile文件立即生效

source /etc/profile

第五步、测试是否安装成功

mvn -v

# 4. 安装git

yum -y install git

# 5. 安装docker

参考文档：

https://help.aliyun.com/document_detail/60742.html?spm=a2c4g.11174283.6.548.24c14541ssYFIZ

第一步：安装必要的一些系统工具

yum install -y yum-utils device-mapper-persistent-data lvm2

第二步：添加软件源信息

yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

第三步：更新并安装Docker-CE

yum makecache fast

yum -y install docker-ce

第四步：开启Docker服务

service docker start

第五步、测试是否安装成功

docker -v

# 6. 安装Jenkins

第一步：上传或下载安装包

cd/usr/local/jenkins

jenkins.war

第二步：启动

nohup java -jar  /usr/local/jenkins/jenkins.war >/usr/local/jenkins/jenkins.out &

第二步：访问

http://ip:8080

# 7. 初始化 Jenkins 插件和管理员用户

## 7.1访问jenkins

http://ip:8080

## 7.2解锁jenkins

获取管理员密码

```
[root@atonlinetest jenkins]# cat /root/.jenkins/secrets/initialAdminPassword
cf78ba0647e042d7a2b4671883953e99
[root@atonlinetest jenkins]#
```


## 注意：配置国内的镜像

官方下载插件慢 更新下载地址

cd {你的Jenkins工作目录}/updates #进入更新配置位置

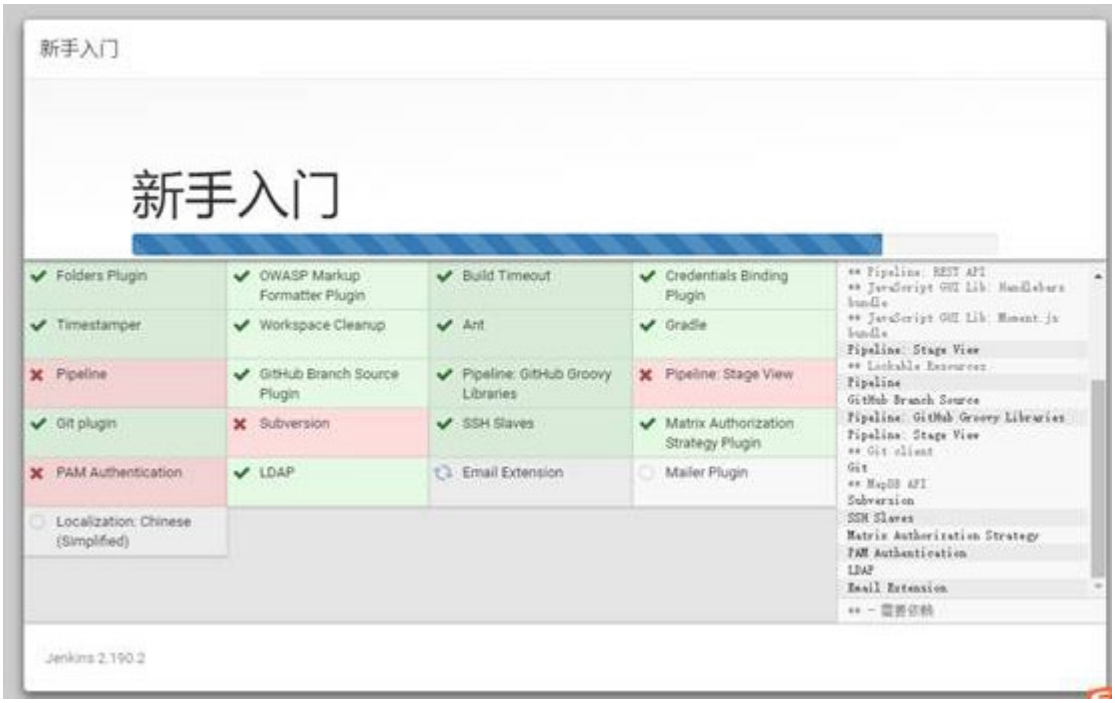
sed -i 's/http:\/\/updates.jenkins-ci.org\/download/https:\/\/mirrors.tuna.tsinghua.edu.cn\/jenkins/g' default.json && sed -i 's/http:\/\/www.google.com/https:\/\/www.baidu.com/g' default.json

这是直接修改的配置文件，如果前边Jenkins用sudo启动的话，那么这里的两个sed前均需要加上sudo

重启Jenkins，安装插件

## 7.3选择“继续”

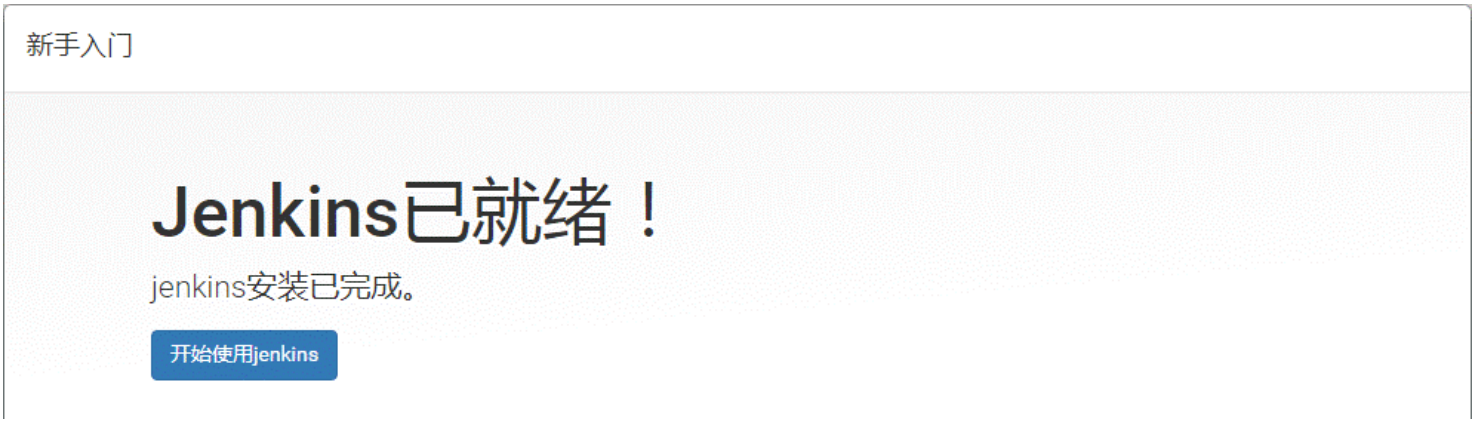

**7.4**选择“安装推荐插件”




**7.5**插件安装完成，创建管理员用户


## 7.6保存并完成


## 7.7进入完成页面




# 8. 配置 Jenkins 构建工具


## 8.1全局工具配置


### 8.1.1配置jdk

JAVA_HOME：/usr/local/jdk


### 8.1.2配置maven

MAVEN_HOME：/usr/local/maven


### 8.1.2配置git

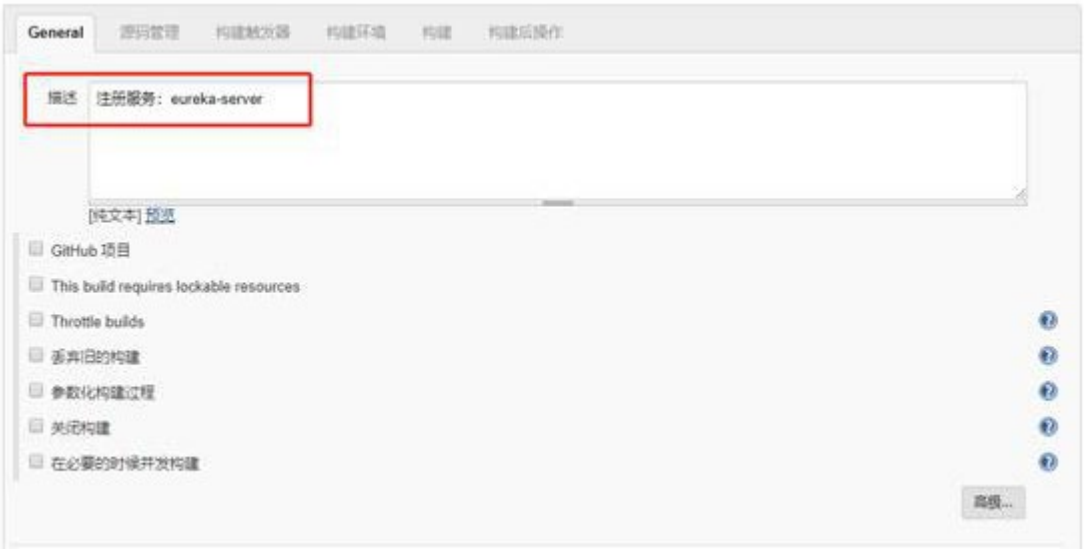
查看git安装路径：which git


## 9. 构建作业

### 9.1点击创建一个新任务，进入创建项目类型选择页面


填好信息点击“确认”

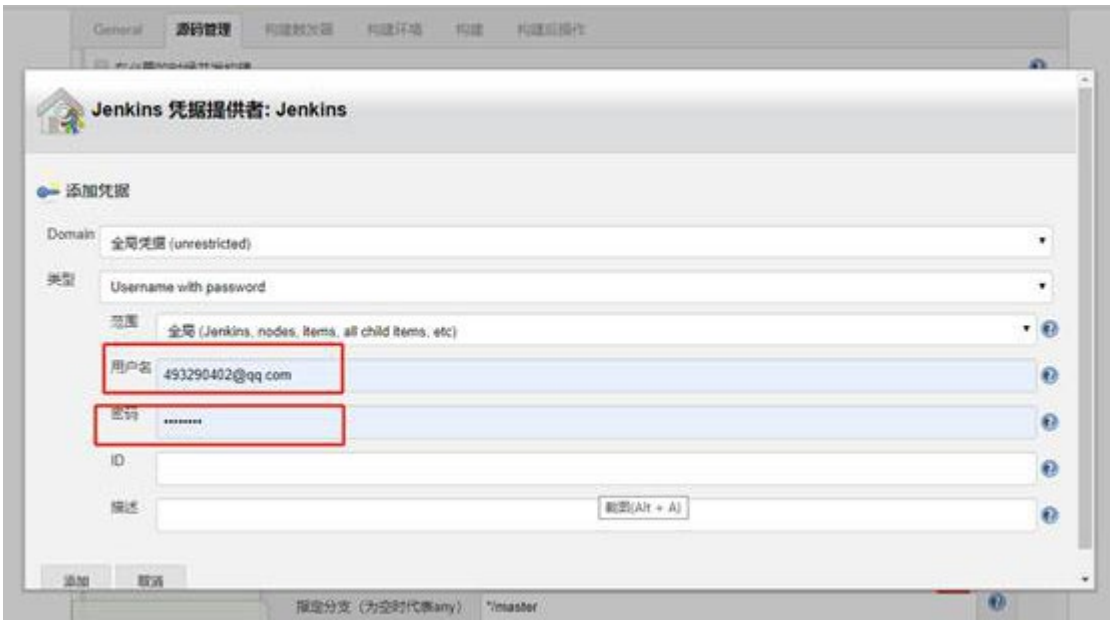
## 9.2配置“General”



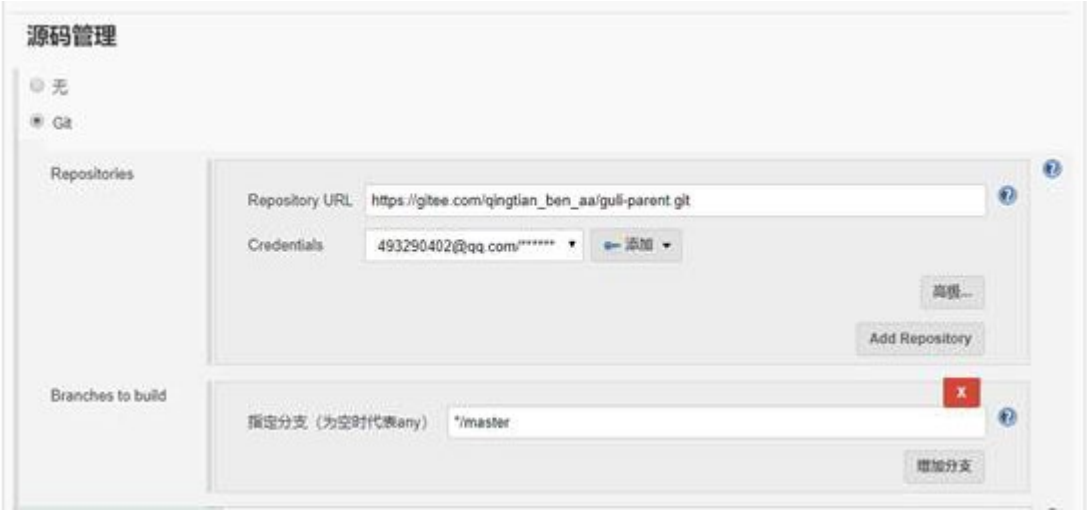

## 9.3配置“源码管理”

填写源码的git地址


添加git用户，git的用户名与密码



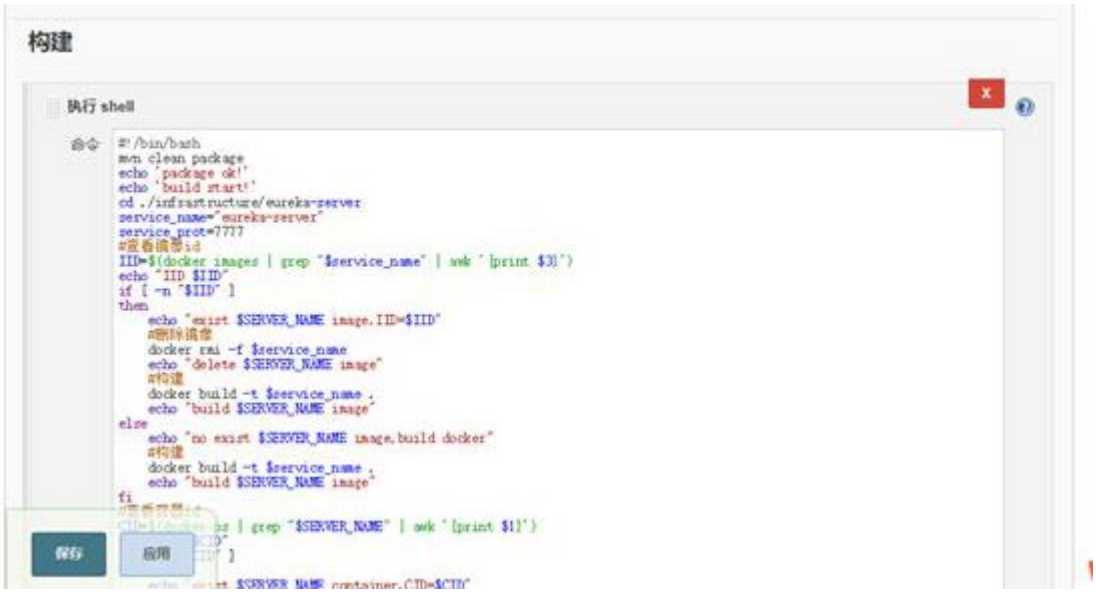

选择添加的用户，上面的红色提示信息消失，说明连接成功，如下图




# 9.4构建作业

到源码中找到docker脚本

选择“执行shell”




保存上面的构建作业

## 9.5构建

构建作业之后，就可以执行构建过程了。

### 9.5.1执行构建过程


## 9.5.2构建结构

第一列是 "上次构建状态显示"，是一个圆形图标，一般分为四种：


上一个构建成功

蓝色：构建成功；


上一个构建成功，但不够稳定。此图标主要用来表示某些测试失败。

黄色：不确定，可能构建成功，但包含错误；


上一个构建彻底失败

红色：构建失败；


项目从未构建，或者被禁用

灰色：项目从未构建过，或者被禁用；

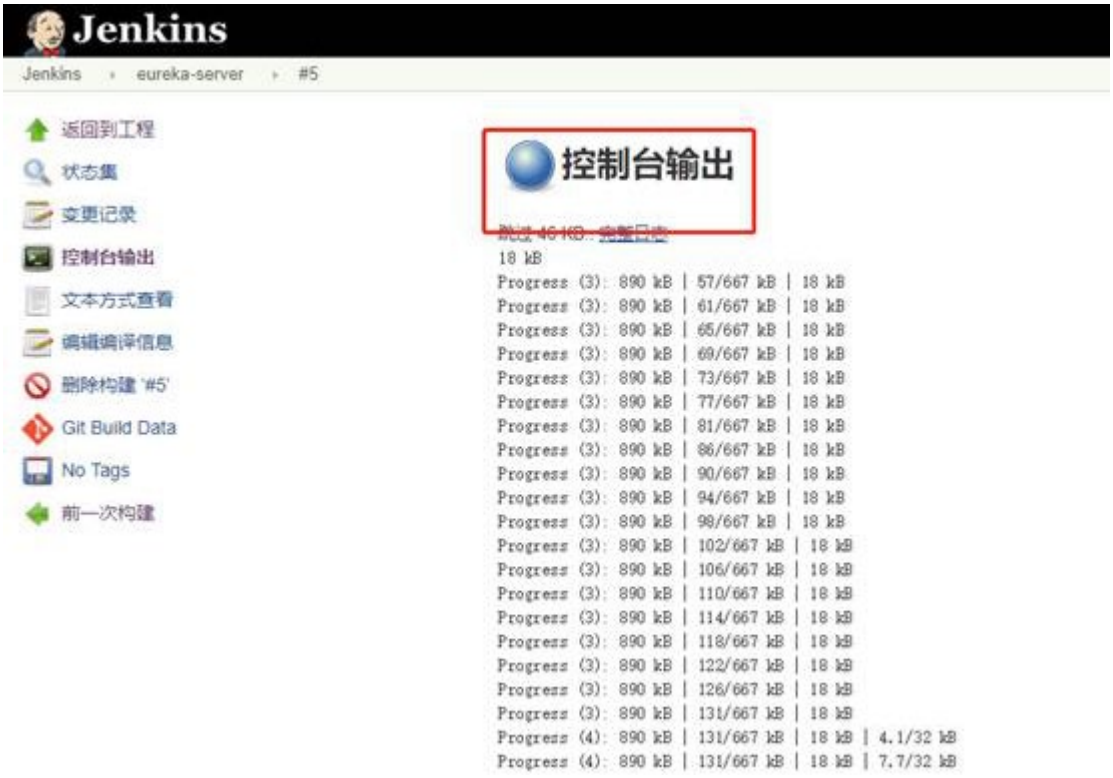
如上显示蓝色，表示构建成功。

注意：手动触发构建的时间与自动定时构建的时间互不影响。

## 9.5.3查看控制台输出


Jenkins
Jenkins › eureka-server ›
返回面板
状态
修改记录
工作空间
立即构建
删除 工程
配置
重命名
Build History
构建历史
Snd
2019-11-25 下午6:20
2019-11-25 下午6:17
工程 eureka-server
eureka-server
工作区
最新修改记录
相关链接
最近一次构建(#5) 1 分 59 秒之前
最近稳定构建(#5) 1 分 59 秒之前
最近成功的构建(#5) 1 分 59 秒之前
最近失败的构建(#4) 4 分 45 秒之前
最近未成功的构建(#4) 4 分 45 秒之前
最近完成的构建(#5) 1 分 59 秒之前

日志内容：


Jenkins
Jenkins › eureka-server › #5
返回到工程
状态集
变更记录
控制台输出
文本方式查看
编辑编译信息
删除构建 '#5'
Git Build Data
No Tags
前一次构建
控制台输出
18 kB
Progress (3): 890 kB | 57/667 kB | 18 kB
Progress (3): 890 kB | 61/667 kB | 18 kB
Progress (3): 890 kB | 65/667 kB | 18 kB
Progress (3): 890 kB | 69/667 kB | 18 kB
Progress (3): 890 kB | 73/667 kB | 18 kB
Progress (3): 890 kB | 77/667 kB | 18 kB
Progress (3): 890 kB | 81/667 kB | 18 kB
Progress (3): 890 kB | 86/667 kB | 18 kB
Progress (3): 890 kB | 90/667 kB | 18 kB
Progress (3): 890 kB | 94/667 kB | 18 kB
Progress (3): 890 kB | 98/667 kB | 18 kB
Progress (3): 890 kB | 102/667 kB | 18 kB
Progress (3): 890 kB | 106/667 kB | 18 kB
Progress (3): 890 kB | 110/667 kB | 18 kB
Progress (3): 890 kB | 114/667 kB | 18 kB
Progress (3): 890 kB | 118/667 kB | 18 kB
Progress (3): 890 kB | 122/667 kB | 18 kB
Progress (3): 890 kB | 126/667 kB | 18 kB
Progress (3): 890 kB | 131/667 kB | 18 kB
Progress (4): 890 kB | 131/667 kB | 18 kB | 4.1/32 kB
Progress (4): 890 kB | 131/667 kB | 18 kB | 7.7/32 kB

```
[INFO] ------------------------------------------------------------------------
package ok!
build start!
IID
no exist  image.build docker
Sending build context to Docker daemon  43.84MB

Step 1/4 : FROM openjdk:8-jdk-alpine
 ---> a3562aa0b991
Step 2/4 : VOLUME /tmp
 ---> Using cache
 ---> 1a6af07524a8
Step 3/4 : COPY ./target/eureka-server.jar eureka-server.jar
 ---> 0cbe77000c39
Step 4/4 : ENTRYPOINT ["java","-jar","/eureka-server.jar", "&"]
 ---> Running in 56fc13533774
Removing intermediate container 56fc13533774
 ---> 800da1cb0cda
Successfully built 800da1cb0cda
Successfully tagged eureka-server:latest
build  image
CID CONTAINER
e6c1f1b39bb8
exist  container.CID=CONTAINER
e6c1f1b39bb8
Error response from daemon: No such container: eureka-server
Error: No such container: eureka-server
[39000e0fe9004dd0345691cf865907383d562e080e02ebff134010aba3fe0
Finished: SUCCESS
```

## 1、项目描述

（1）

在线教育系统，分为前台网站系统和后台运营平台，B2C模式。

前台用户系统包括课程、讲师、问答、文章几大大部分，使用了微服务技术架构，前后端分离开发。

后端的主要技术架构是：SpringBoot + SpringCloud + MyBatis-Plus + HttpClient + MySQL + Maven+EasyExcel+ nginx

前端的架构是：Node.js + Vue.js +element-ui+NUXT+ECharts

其他涉及到的中间件包括Redis、阿里云OSS、阿里云视频点播

业务中使用了ECharts做图表展示，使用EasyExcel完成分类批量添加、注册分布式单点登录使用了JWT

（2）

项目前后端分离开发，后端采用SpringCloud微服务架构，持久层用的是MyBatis-Plus，微服务分库设计，使用Swagger生成接口文档

接入了阿里云视频点播、阿里云OSS。

系统分为前台用户系统和后台管理系统两部分。

前台用户系统包括：首页、课程、名师、问答、文章。

后台管理系统包括：讲师管理、课程分类管理、课程管理、统计分析、Banner管理、订单管理、权限管理等功能。

在线教育计费案例：

小A是一名杭州的创业者，带领团队研发了一个在线教育平台。他希望把视频托管在阿里云上，存量视频大约1000个，占用存储空间近1T，每月预计新增视频100个，并新增存储约100G，课程视频的时长集中在20-40分钟，并且按照不同课程进行分类管理。为了保障各端的观看效果，计划为用户提供"标清480P"和"高清720P"两种清晰度。目前已有用户400人左右，每日平均视频观看次数1000次，在移动端和PC端观看次数比例大致为3:1。

## 2、这是一个项目还是一个产品

这是一个产品

1.0版本是单体应用：SSM

2.0版本加入了SpringCloud，将一些关键业务和访问量比较大的部分分离了出去

目前独立出来的服务有教学服务、视频点播服务、用户服务、统计分析服务、网关服务

## 3、测试要求

首页和视频详情页qps单机qps要求 2000+

经常用每秒查询率来衡量域名系统服务器的机器的性能，其即为QPS

QPS = 并发量 / 平均响应时间

## 4、企业中的项目（产品）开发流程

一个中大型项目的开发流程

1、需求调研（产品经理）

2、需求评审（产品/设计/前端/后端/测试/运营）

3、立项（项目经理、品管）

4、UI设计

5、开发

- 架构、数据库设计、API文档、MOCK数据、开发、单元测试
- 前端
- 后端

6、前端后端联调

7、项目提测：黑盒白盒、压力测试（qps） loadrunner

8、bug修改

9、回归测试

10、运维和部署上线

11、灰度发布

12、全量发布

13、维护和运营

## 5、系统中都有那些角色？数据库是怎么设计的？

前台：会员（学员）

后台：系统管理员、运营人员

后台分库，每个微服务一个独立的数据库，使用了分布式id生成器

## 6、视频点播是怎么实现的（流媒体你们是怎么实现的）

我们直接接入了阿里云的云视频点播。云平台上的功能包括视频上传、转码、加密、智能审核、监控统计等。

还包括视频播放功能，阿里云还提供了一个视频播放器。

## 7、前后端联调经常遇到的问题：

1、请求方式post、get

2、json、x-wwww-form-urlencoded混乱的错误

3、后台必要的参数，前端省略了

4、数据类型不匹配

5、空指针异常

6、分布式系统中分布式id生成器生成的id 长度过大（19个字符长度的整数），js无法解析（js智能解析16个长度：2的53次幂）

id策略改成 ID_WORKER_STR

## 8、前后端分离项目中的跨域问题是如何解决的

后端服务器配置：我们的项目中是通过Spring注解解决跨域的 @CrossOrigin

也可以使用nginx反向代理、httpClient、网关

## 9、说说你做了哪个部分、遇到了什么问题、怎么解决的

问题1:

分布式id生成器在前端无法处理，总是在后三位进行四舍五入。

分布式id生成器生成的id是19个字符的长度，前端javascript脚本对整数的处理能力只有2的53次方，也就是最多只能处理16个字符

解决的方案是把id在程序中设置成了字符串的性质

问题2：

项目迁移到Spring-Cloud的时候，经过网关时，前端传递的cookie后端一只获取不了，看了cloud中zuul的源码，发现向下游传递数据的时候，zull默认过滤了敏感信息，将cookie过滤掉了

解决的方案是在配置文件中将请求头的过滤清除掉，使cookie可以向下游传递

问题3.....

## 10、分布式系统的id生成策略

https://www.cnblogs.com/haoxinyue/p/5208136.html

## 11、项目组有多少人，人员如何组成?

## 12、分布式系统的CAP原理

**CAP定理：**

指的是在一个分布式系统中，Consistency（一致性）、 Availability（可用性）、Partition tolerance（分区容错性），三者不可同时获得。

一致性（C）：在分布式系统中的所有数据备份，在同一时刻是否同样的值。（所有节点在同一时间的数据完全一致，越多节点，数据同步越耗时）

可用性（A）：负载过大后，集群整体是否还能响应客户端的读写请求。（服务一直可用，而且是正常响应时间）

分区容错性（P）：分区容错性，就是高可用性，一个节点崩了，并不影响其它的节点（100个节点，挂了几个，不影响服务，越多机器越好）

**CA 满足的情况下，P不能满足的原因：**

数据同步(C)需要时间，也要正常的时间内响应(A)，那么机器数量就要少，所以P就不满足

**CP 满足的情况下，A不能满足的原因：**

数据同步(C)需要时间, 机器数量也多(P)，但是同步数据需要时间，所以不能再正常时间内响应，所以A就不满足

**AP 满足的情况下，C不能满足的原因：**

机器数量也多(P)，正常的时间内响应(A)，那么数据就不能及时同步到其他节点，所以C不满足

**注册中心选择的原则：**

Zookeeper：CP设计，保证了一致性，集群搭建的时候，某个节点失效，则会进行选举行的leader，或者半数以上节点不可用，则无法提供服务，因此可用性没法满足

Eureka：AP原则，无主从节点，一个节点挂了，自动切换其他节点可以使用，去中心化

**结论：**

分布式系统中P,肯定要满足，所以我们只能在一致性和可用性之间进行权衡

如果要求一致性，则选择zookeeper，如金融行业

如果要求可用性，则Eureka，如教育、电商系统

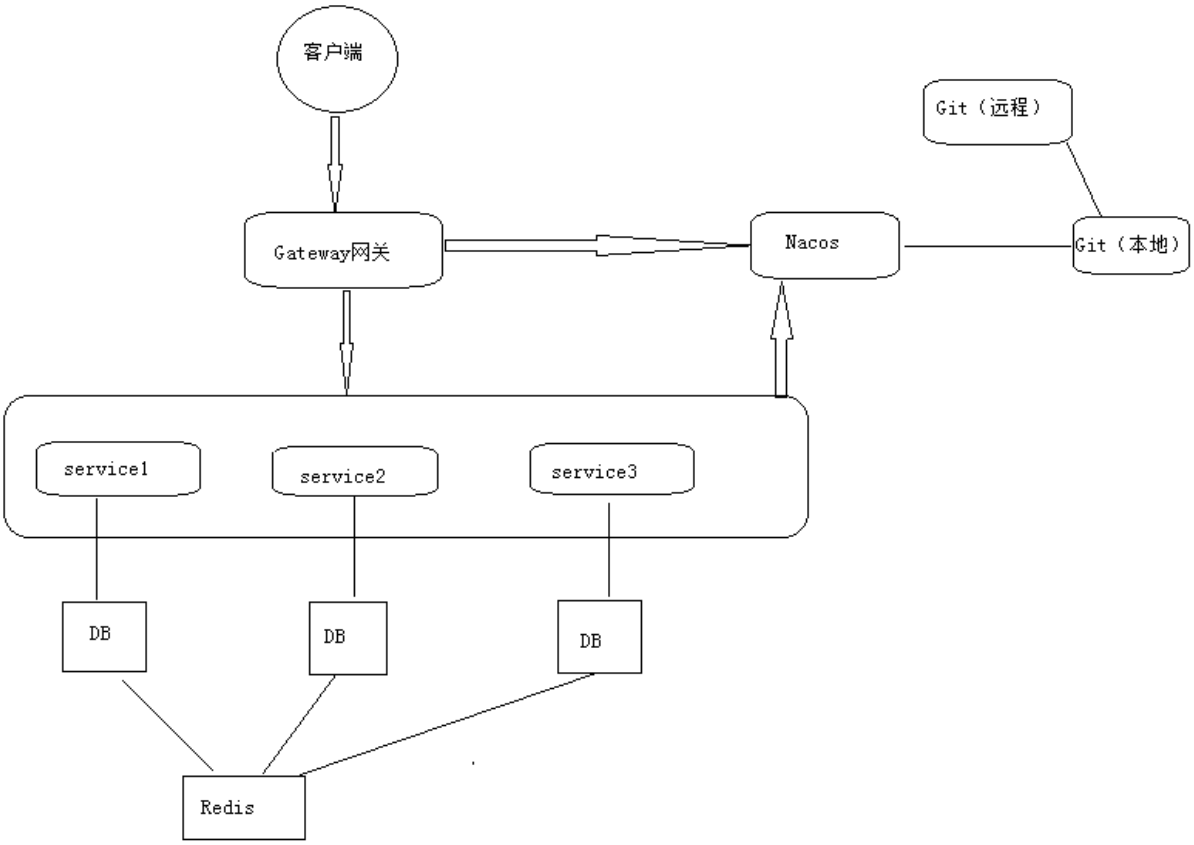
没有最好的选择，最好的选择是根据业务场景来进行架构设计

## 13、前端渲染和后端渲染有什么区别

前端渲染是返回json给前端，通过javascript将数据绑定到页面上

后端渲染是在服务器端将页面生成直接发送给服务器，有利于SEO的优化

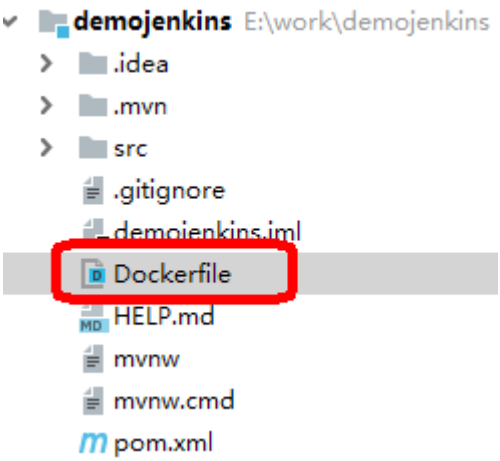
## 14、能画一下系统架构图吗


客户端
Git（远程）
Gateway网关
Nacos
Git（本地）
service1
service2
service3
DB
DB
DB
Redis

# 1. 准备代码，提交到码云Git库

代码中需要包含以下几部分内容：

（1）代码中需要包含Dockerfile文件




文件内容：

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
COPY ./target/demojenkins.jar demojenkins.jar
ENTRYPOINT ["java","-jar","/demojenkins.jar", "&"]
```

（2）在项目pom文件中指定打包类型，包含build部分内容

```
<groupId>com.atguigu</groupId>
<artifactId>demojenkins</artifactId>
<packaging>jar</packaging>
<version>0.0.1-SNAPSHOT</version>
<name>demojenkins</name>
<description>Demo project for Spring Boot</description>
```

```
<build>
    <finalName>demojenkins</finalName>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

## 2. 安装JAVA 运行环境

第一步：上传或下载安装包

cd/usr/local

jdk-8u121-linux-x64.tar.gz

第二步：解压安装包

tar -zxvf jdk-8u121-linux-x64.tar.gz

第三步：建立软连接

ln -s /usr/local/jdk1.8.0_121/ /usr/local/jdk

第四步：修改环境变量

vim /etc/profile

export JAVA_HOME=/usr/local/jdk

export JRE_HOME=$JAVA_HOME/jre

export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib

export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin

通过命令source /etc/profile让profile文件立即生效

source /etc/profile

第五步、测试是否安装成功

②、使用java -version，出现版本

# 3. 安装maven

第一步：上传或下载安装包

cd/usr/local

apache-maven-3.6.1-bin.tar.gz

第二步：解压安装包

tar -zxvf apache-maven-3.6.1-bin.tar.gz

第三步：建立软连接

ln -s /usr/local/apache-maven-3.6.1/ /usr/local/maven

第四步：修改环境变量

vim /etc/profile

export MAVEN_HOME=/usr/local/maven

export PATH=$PATH:$MAVEN_HOME/bin

通过命令source /etc/profile让profile文件立即生效

source /etc/profile

第五步、测试是否安装成功

mvn -v

# 4. 安装git

yum -y install git

# 5. 安装docker

参考文档：

https://help.aliyun.com/document_detail/60742.html?spm=a2c4g.11174283.6.548.24c14541ssYFIZ

第一步：安装必要的一些系统工具

yum install -y yum-utils device-mapper-persistent-data lvm2

第二步：添加软件源信息

yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

第三步：更新并安装Docker-CE

yum makecache fast

yum -y install docker-ce

第四步：开启Docker服务

service docker start

第五步、测试是否安装成功

docker -v

# 6. 安装Jenkins

第一步：上传或下载安装包

cd/usr/local/jenkins

jenkins.war

第二步：启动

nohup java -jar  /usr/local/jenkins/jenkins.war >/usr/local/jenkins/jenkins.out &

第二步：访问

http://ip:8080

# 7. 初始化 Jenkins 插件和管理员用户

## 7.1访问jenkins

http://ip:8080

## 7.2解锁jenkins

获取管理员密码

```
[root@atonlinetest jenkins]# cat /root/.jenkins/secrets/initialAdminPassword
cf78ba0647e042d7a2b4671883953e99
[root@atonlinetest jenkins]#
```


## 注意：配置国内的镜像

官方下载插件慢 更新下载地址

cd {你的Jenkins工作目录}/updates #进入更新配置位置

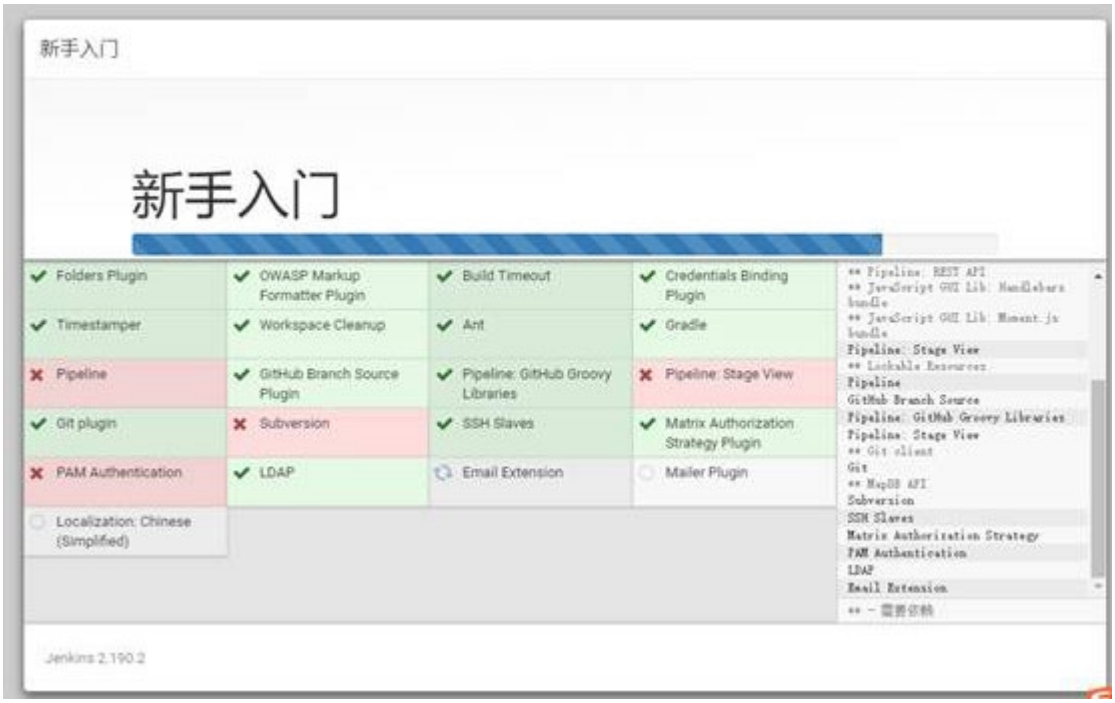
sed -i 's/http:\/\/updates.jenkins-ci.org\/download/https:\/\/mirrors.tuna.tsinghua.edu.cn\/jenkins/g' default.json && sed -i 's/http:\/\/www.google.com/https:\/\/www.baidu.com/g' default.json

这是直接修改的配置文件，如果前边Jenkins用sudo启动的话，那么这里的两个sed前均需要加上sudo

重启Jenkins，安装插件

## 7.3选择“继续”

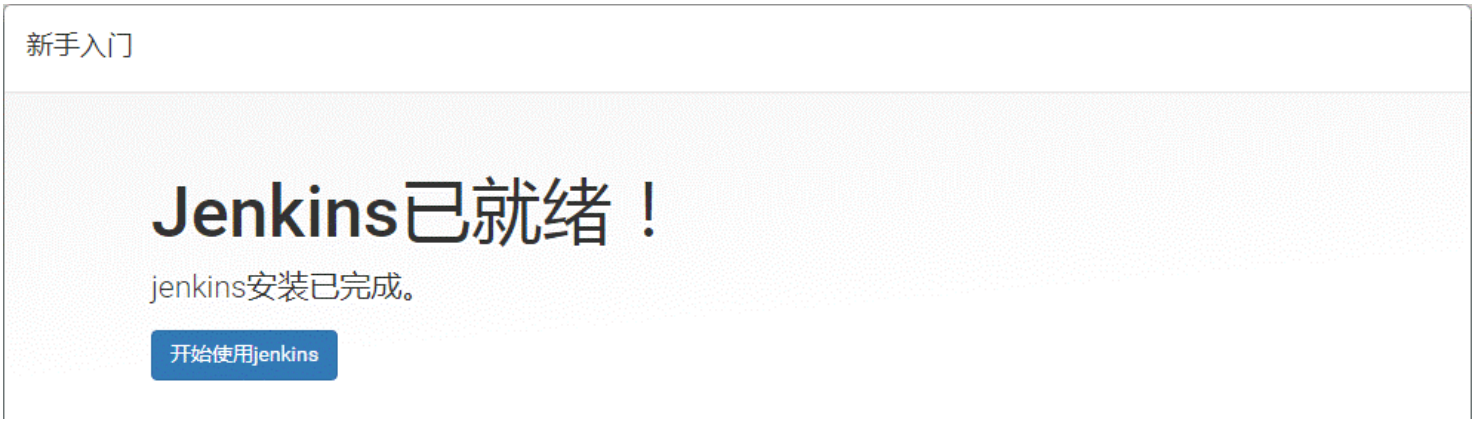

## 7.4选择“安装推荐插件”




## 7.5插件安装完成，创建管理员用户


## 7.6保存并完成


## 7.7进入完成页面




# 8. 配置 Jenkins 构建工具


## 8.1全局工具配置


### 8.1.1配置jdk

JAVA_HOME：/usr/local/jdk


## 8.1.2配置maven

MAVEN_HOME：/usr/local/maven


## 8.1.2配置git

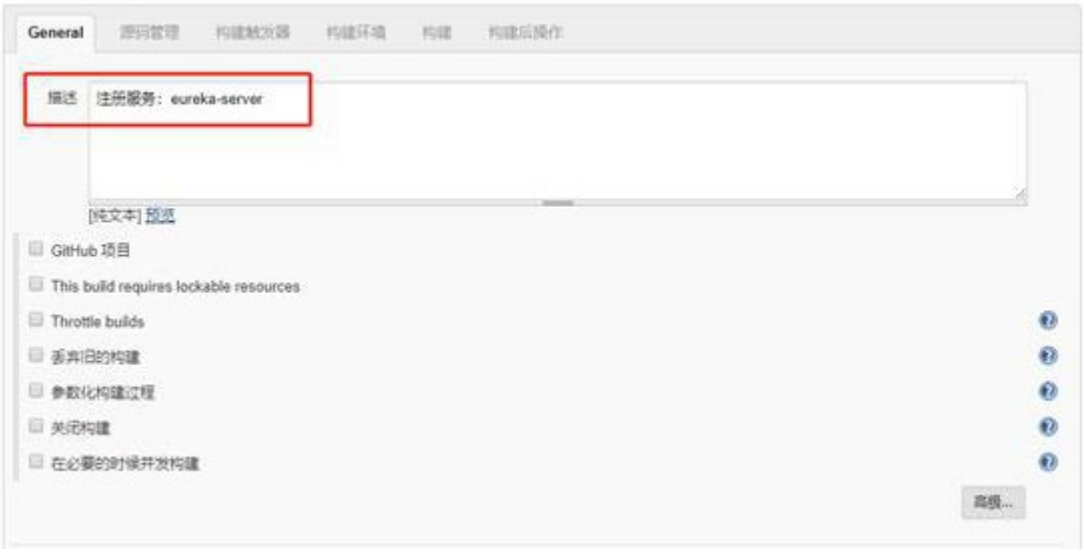
查看git安装路径：which git


# 9. 构建作业

## 9.1点击创建一个新任务，进入创建项目类型选择页面


填好信息点击“确认”

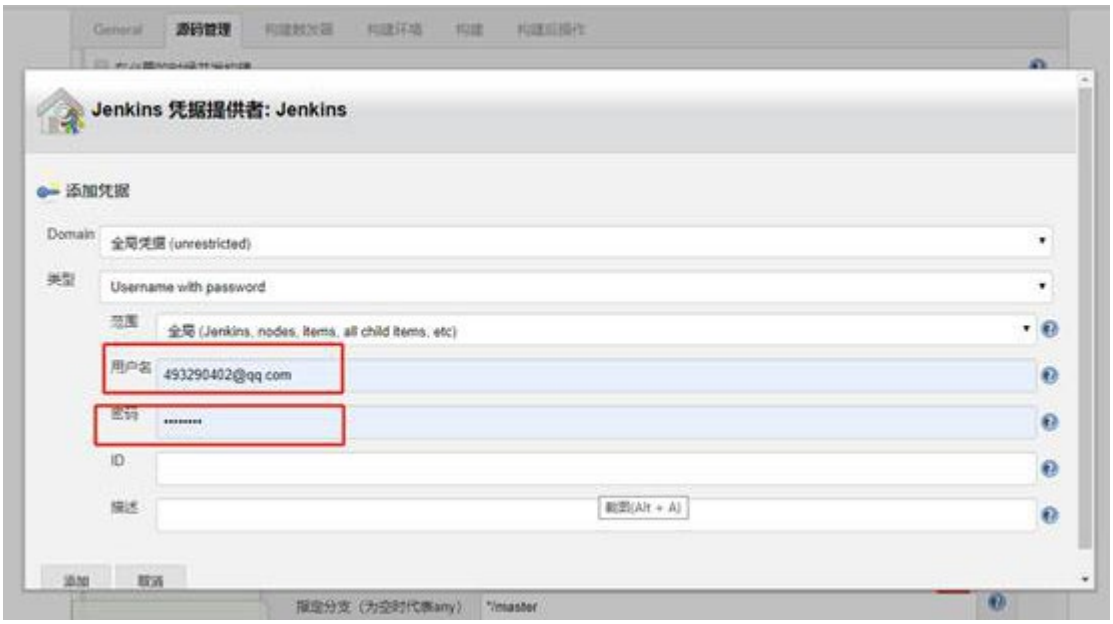
## 9.2配置“General”



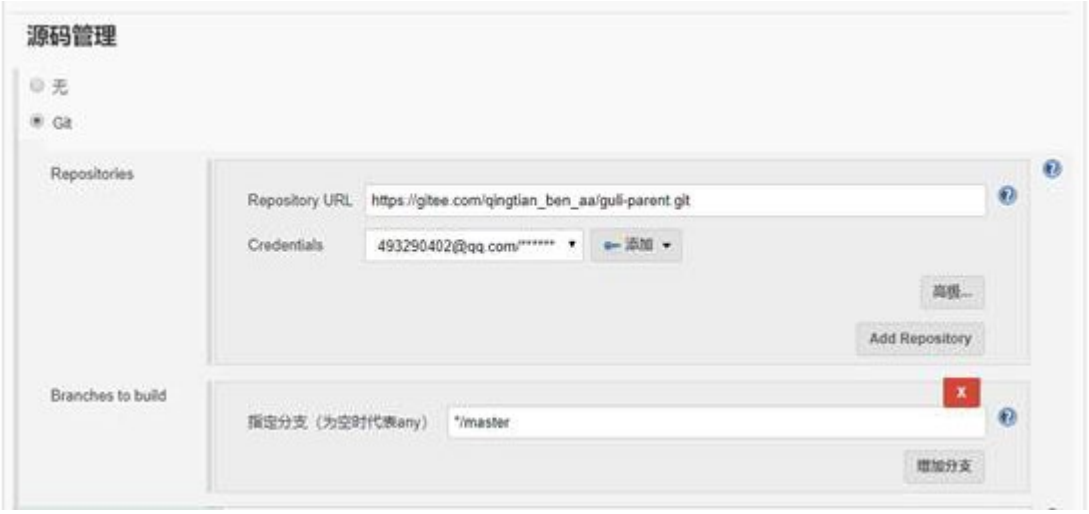

## 9.3配置“源码管理”

填写源码的git地址


添加git用户，git的用户名与密码



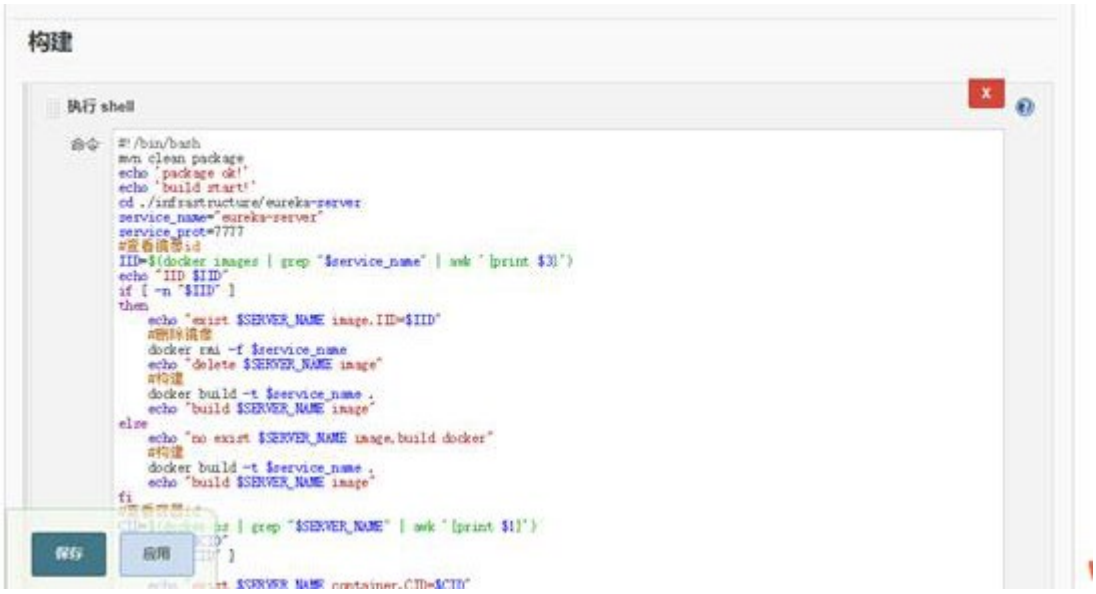

选择添加的用户，上面的红色提示信息消失，说明连接成功，如下图




# 9.4构建作业

到源码中找到docker脚本

选择“执行shell”




保存上面的构建作业

# 9.5构建

构建作业之后，就可以执行构建过程了。

## 9.5.1执行构建过程


## 9.5.2构建结构

第一列是 "上次构建状态显示"，是一个圆形图标，一般分为四种：


蓝色：构建成功；


黄色：不确定，可能构建成功，但包含错误；


红色：构建失败；


灰色：项目从未构建过，或者被禁用；

如上显示蓝色，表示构建成功。

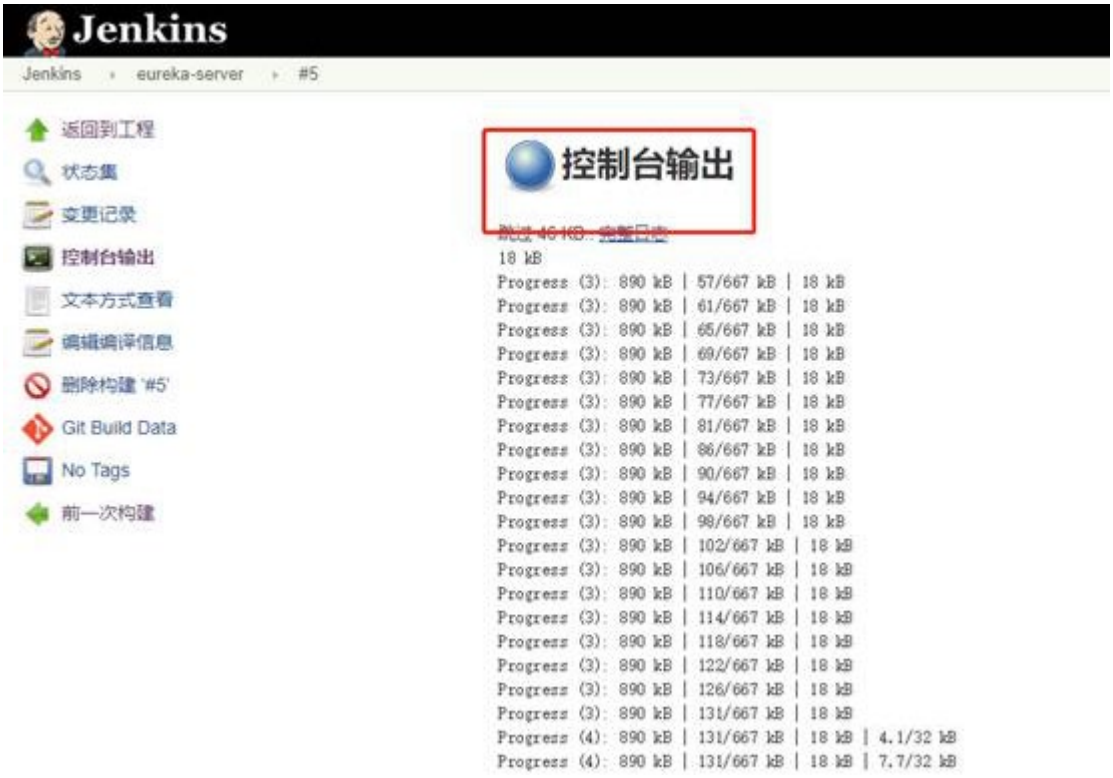
注意：手动触发构建的时间与自动定时构建的时间互不影响。

## 9.5.3查看控制台输出

01-项目部署


Jenkins
Jenkins › eureka-server ›
返回面板
状态
修改记录
工作空间
立即构建
删除 工程
配置
重命名
Build History
构建历史
2019-11-25 下午6:20
2019-11-25 下午6:17
工程 eureka-server
eureka-server
工作区
最新修改记录
相关链接
最近一次构建(#5) 1 分 59 秒之前
最近稳定构建(#5) 1 分 59 秒之前
最近成功的构建(#5) 1 分 59 秒之前
最近失败的构建(#4) 4 分 45 秒之前
最近未成功的构建(#4) 4 分 45 秒之前
最近完成的构建(#5) 1 分 59 秒之前

日志内容：


Jenkins
Jenkins › eureka-server › #5
返回到工程
状态集
变更记录
控制台输出
文本方式查看
编辑编译信息
删除构建 '#5'
Git Build Data
No Tags
前一次构建
控制台输出
18 kB
Progress (3): 890 kB | 57/667 kB | 18 kB
Progress (3): 890 kB | 61/667 kB | 18 kB
Progress (3): 890 kB | 65/667 kB | 18 kB
Progress (3): 890 kB | 69/667 kB | 18 kB
Progress (3): 890 kB | 73/667 kB | 18 kB
Progress (3): 890 kB | 77/667 kB | 18 kB
Progress (3): 890 kB | 81/667 kB | 18 kB
Progress (3): 890 kB | 86/667 kB | 18 kB
Progress (3): 890 kB | 90/667 kB | 18 kB
Progress (3): 890 kB | 94/667 kB | 18 kB
Progress (3): 890 kB | 98/667 kB | 18 kB
Progress (3): 890 kB | 102/667 kB | 18 kB
Progress (3): 890 kB | 106/667 kB | 18 kB
Progress (3): 890 kB | 110/667 kB | 18 kB
Progress (3): 890 kB | 114/667 kB | 18 kB
Progress (3): 890 kB | 118/667 kB | 18 kB
Progress (3): 890 kB | 122/667 kB | 18 kB
Progress (3): 890 kB | 126/667 kB | 18 kB
Progress (3): 890 kB | 131/667 kB | 18 kB
Progress (4): 890 kB | 131/667 kB | 18 kB | 4.1/32 kB
Progress (4): 890 kB | 131/667 kB | 18 kB | 7.7/32 kB

01-项目部署

```
[INFO] ------------------------------------------------------------------------
package ok!
build start!
IID
no exist  image.build docker
Sending build context to Docker daemon  43.84MB

Step 1/4 : FROM openjdk:8-jdk-alpine
 ---> a3562aa0b991
Step 2/4 : VOLUME /tmp
 ---> Using cache
 ---> 1a6af07524a8
Step 3/4 : COPY ./target/eureka-server.jar eureka-server.jar
 ---> 0cbe77000c39
Step 4/4 : ENTRYPOINT ["java","-jar","/eureka-server.jar", "&"]
 ---> Running in 56fc13533774
Removing intermediate container 56fc13533774
 ---> 800da1cb0cda
Successfully built 800da1cb0cda
Successfully tagged eureka-server:latest
build  image
CID CONTAINER
e6c1f1b39bb8
exist  container.CID=CONTAINER
e6c1f1b39bb8
Error response from daemon: No such container: eureka-server
Error: No such container: eureka-server
[39000e0fe9004dd03145691cf865907383d562e080e02ebff134010aba3fe0
Finished: SUCCESS
```

# 1、项目描述

（1）

在线教育系统，分为前台网站系统和后台运营平台，B2C模式。

前台用户系统包括课程、讲师、问答、文章几大大部分，使用了微服务技术架构，前后端分离开发。

后端的主要技术架构是：SpringBoot + SpringCloud + MyBatis-Plus + HttpClient + MySQL + Maven+EasyExcel+ nginx

前端的架构是：Node.js + Vue.js +element-ui+NUXT+ECharts

其他涉及到的中间件包括Redis、阿里云OSS、阿里云视频点播

业务中使用了ECharts做图表展示，使用EasyExcel完成分类批量添加、注册分布式单点登录使用了JWT

（2）

项目前后端分离开发，后端采用SpringCloud微服务架构，持久层用的是MyBatis-Plus，微服务分库设计，使用Swagger生成接口文档

接入了阿里云视频点播、阿里云OSS。

系统分为前台用户系统和后台管理系统两部分。

前台用户系统包括：首页、课程、名师、问答、文章。

后台管理系统包括：讲师管理、课程分类管理、课程管理、统计分析、Banner管理、订单管理、权限管理等功能。

在线教育计费案例：

小A是一名杭州的创业者，带领团队研发了一个在线教育平台。他希望把视频托管在阿里云上，存量视频大约1000个，占用存储空间近1T，每月预计新增视频100个，并新增存储约100G，课程视频的时长集中在20-40分钟，并且按照不同课程进行分类管理。为了保障各端的观看效果，计划为用户提供"标清480P"和"高清720P"两种清晰度。目前已有用户400人左右，每日平均视频观看次数1000次，在移动端和PC端观看次数比例大致为3:1。

# 2、这是一个项目还是一个产品

这是一个产品

1.0版本是单体应用：SSM

2.0版本加入了SpringCloud，将一些关键业务和访问量比较大的部分分离了出去

目前独立出来的服务有教学服务、视频点播服务、用户服务、统计分析服务、网关服务

## 3、测试要求

首页和视频详情页qps单机qps要求 2000+

经常用每秒查询率来衡量域名系统服务器的机器的性能，其即为QPS

QPS = 并发量 / 平均响应时间

## 4、企业中的项目（产品）开发流程

一个中大型项目的开发流程

1、需求调研（产品经理）

2、需求评审（产品/设计/前端/后端/测试/运营）

3、立项（项目经理、品管）

4、UI设计

5、开发

- 架构、数据库设计、API文档、MOCK数据、开发、单元测试
- 前端
- 后端

6、前端后端联调

7、项目提测：黑盒白盒、压力测试（qps） loadrunner

8、bug修改

9、回归测试

10、运维和部署上线

11、灰度发布

12、全量发布

13、维护和运营

## 5、系统中都有那些角色？数据库是怎么设计的？

前台：会员（学员）

后台：系统管理员、运营人员

后台分库，每个微服务一个独立的数据库，使用了分布式id生成器

## 6、视频点播是怎么实现的（流媒体你们是怎么实现的）

我们直接接入了阿里云的云视频点播。云平台上的功能包括视频上传、转码、加密、智能审核、监控统计等。

还包括视频播放功能，阿里云还提供了一个视频播放器。

## 7、前后端联调经常遇到的问题：

1、请求方式post、get

2、json、x-wwww-form-urlencoded混乱的错误

3、后台必要的参数，前端省略了

4、数据类型不匹配

5、空指针异常

6、分布式系统中分布式id生成器生成的id 长度过大（19个字符长度的整数），js无法解析（js智能解析16个长度：2的53次幂）

id策略改成 ID_WORKER_STR

## 8、前后端分离项目中的跨域问题是如何解决的

后端服务器配置：我们的项目中是通过Spring注解解决跨域的 @CrossOrigin

也可以使用nginx反向代理、httpClient、网关

## 9、说说你做了哪个部分、遇到了什么问题、怎么解决的

问题1:

分布式id生成器在前端无法处理，总是在后三位进行四舍五入。

分布式id生成器生成的id是19个字符的长度，前端javascript脚本对整数的处理能力只有2的53次方，也就是最多只能处理16个字符

解决的方案是把id在程序中设置成了字符串的性质

问题2：

项目迁移到Spring-Cloud的时候，经过网关时，前端传递的cookie后端一只获取不了，看了cloud中zuul的源码，发现向下游传递数据的时候，zull默认过滤了敏感信息，将cookie过滤掉了

解决的方案是在配置文件中将请求头的过滤清除掉，使cookie可以向下游传递

问题3.....

## 10、分布式系统的id生成策略

https://www.cnblogs.com/haoxinyue/p/5208136.html

## 11、项目组有多少人，人员如何组成?

## 12、分布式系统的CAP原理

**CAP定理：**

指的是在一个分布式系统中，Consistency（一致性）、 Availability（可用性）、Partition tolerance（分区容错性），三者不可同时获得。

一致性（C）：在分布式系统中的所有数据备份，在同一时刻是否同样的值。（所有节点在同一时间的数据完全一致，越多节点，数据同步越耗时）

可用性（A）：负载过大后，集群整体是否还能响应客户端的读写请求。（服务一直可用，而且是正常响应时间）

分区容错性（P）：分区容错性，就是高可用性，一个节点崩了，并不影响其它的节点（100个节点，挂了几个，不影响服务，越多机器越好）

**CA 满足的情况下，P不能满足的原因：**

数据同步(C)需要时间，也要正常的时间内响应(A)，那么机器数量就要少，所以P就不满足

**CP 满足的情况下，A不能满足的原因：**

数据同步(C)需要时间, 机器数量也多(P)，但是同步数据需要时间，所以不能再正常时间内响应，所以A就不满足

**AP 满足的情况下，C不能满足的原因：**

机器数量也多(P)，正常的时间内响应(A)，那么数据就不能及时同步到其他节点，所以C不满足

**注册中心选择的原则：**

Zookeeper：CP设计，保证了一致性，集群搭建的时候，某个节点失效，则会进行选举行的leader，或者半数以上节点不可用，则无法提供服务，因此可用性没法满足

Eureka：AP原则，无主从节点，一个节点挂了，自动切换其他节点可以使用，去中心化

**结论：**

分布式系统中P,肯定要满足，所以我们只能在一致性和可用性之间进行权衡

如果要求一致性，则选择zookeeper，如金融行业

如果要求可用性，则Eureka，如教育、电商系统

没有最好的选择，最好的选择是根据业务场景来进行架构设计

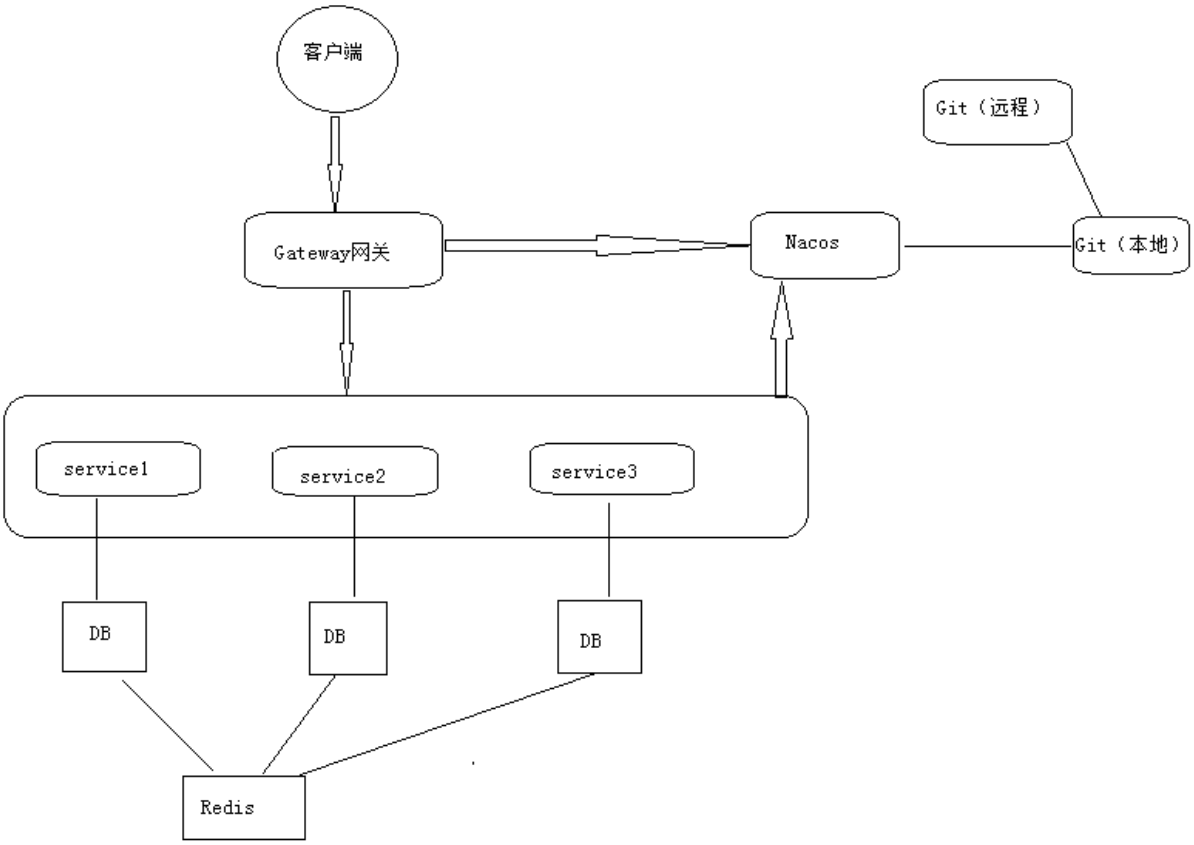
## 13、前端渲染和后端渲染有什么区别

前端渲染是返回json给前端，通过javascript将数据绑定到页面上

后端渲染是在服务器端将页面生成直接发送给服务器，有利于SEO的优化

## 14、能画一下系统架构图吗

面试总结


客户端
Git（远程）
Gateway网关
Nacos
Git（本地）
service1
service2
service3
DB
DB
DB
Redis