

1 学习目标

- 了解电商行业
- 了解商城项目结构
- 能独立搭建项目基本框架
- 能参考使用ES6的新语法

2 了解电商行业

学习电商项目，自然要先了解这个行业，所以我们首先来聊聊电商行业

2.1 项目分类

主要从需求方、盈利模式、技术侧重点这三个方面来看它们的不同

2.1.1 传统项目

各种企业里面用的管理系统（ERP、HR、OA、CRM、物流管理系统。。。。。。）

不涉及到大量的用户

- 需求方：公司、企业内部
- 盈利模式：项目本身卖钱
- 技术侧重点：业务功能

2.1.2 互联网项目

门户网站、电商网站：baidu.com、qq.com、taobao.com、jd.com

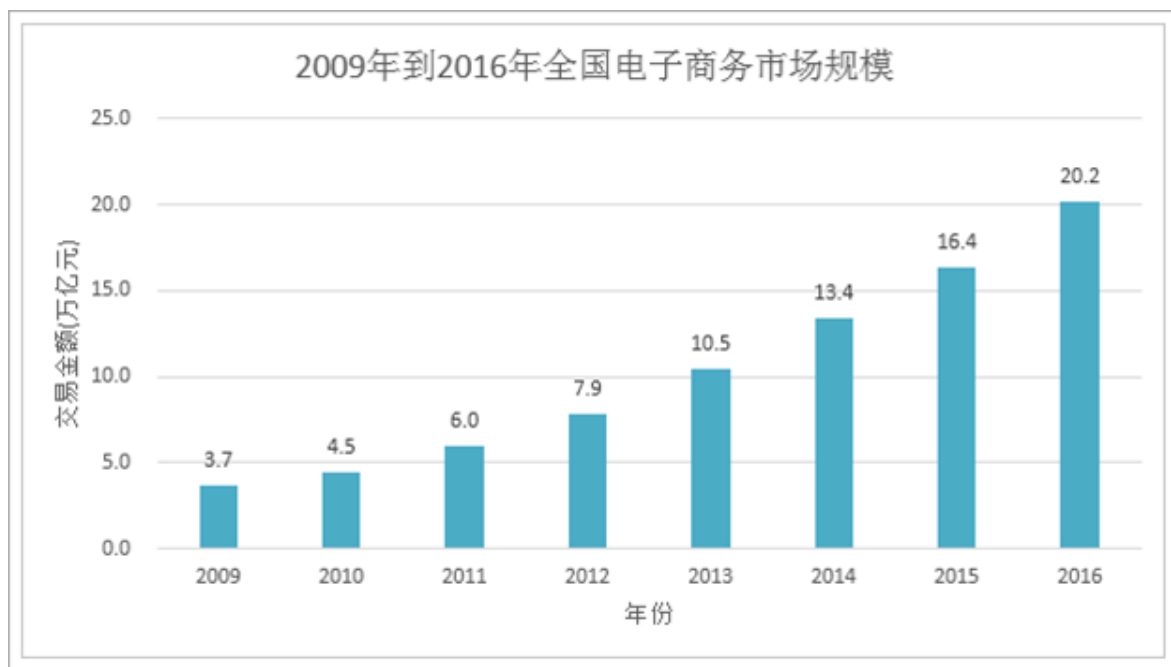
- 需求方：广大用户群体
- 盈利模式：虚拟币、增值服务、现金流、金融借贷、广告收益.....
- 技术侧重点：网站性能、业务功能

2.2 电商行业的发展


2.2.1 钱景

近年来，中国的电子商务快速发展，交易额连创新高，电子商务在各领域的应用不断拓展和深化、相关服务业蓬勃发展、支撑体系不断健全完善、创新的动力和能力不断增强。电子商务正在与实体经济深度融合，进入规模性发展阶段，对经济社会生活的影响不断增大，正成为我国经济发展的新引擎。

中国电子商务研究中心数据显示，截止到 2012 年底，中国电子商务市场交易规模达 7.85 万亿人民币，同比增长 30.83%。其中，B2B 电子商务交易额达 6.25 万亿，同比增长 27%。而 2011 年全年，中国电子商务市场交易额达 6 万亿人民币，同比增长 33%，占 GDP 比重上升到 13%；2012 年，电子商务占 GDP 的比重已经高达 15%。



2.2.2 数据



订单：**278545339**

2014年双11**支付宝**交易峰值**285万笔/分钟**，相比去年双11期间79万笔/分钟的交易峰值，系统支撑能力达到了去年**3倍**以上。

2016双11开场30分钟，创造**每秒交易峰值17.5万笔**，**每秒支付峰值12万笔**的新纪录。菜鸟单日物流订单量超过**4.67亿**，创历史新高。

2017：25.6万笔/秒的支付峰值再创世界新纪录 总交易额1600y

2018年：截止02分05秒，100亿。04分01秒，200亿。09分05秒，300亿。15分00秒，400亿。26分03秒，500亿。28分41秒，520亿元，“我爱你”截止2018年11月12日0时0分0秒，2018天猫双11总成交额超2100亿元

2019:天猫双11交易峰值创下新纪录，达到54.4万笔/秒。是2009年第一次双11的1360倍

2.2.3 技术特点

从上面的数据我们不仅要看到钱，更要看到背后的技术实力。正是得益于电商行业的高强度并发压力，促使了BAT等巨头们的技术进步。电商行业有些什么特点呢？

- 技术范围广
- 技术新
- 高并发（分布式、静态化技术、cdn加速，缓存技术、异步并发、搜索、队列）--> 在指定时间段内的并发量(QPS)
- 高可用（集群、负载均衡、限流、降级、熔断）
- 数据量大
- 业务复杂
- 数据安全

2.3 常见电商模式

电商行业的一些常见模式：

- B2C: 商家对个人, 如: 小米官网、苹果官网等
- C2C平台: 个人对个人, 如: 闲鱼、ebay
- B2B平台: 商家对商家, 如: 阿里巴巴、阿里云等
- O2O: 线上和线下结合, 如: 美团, 滴滴等
- P2P: 在线金融, 贷款, 如: 拍拍贷、蚂蚁金融等。
- B2C平台: 天猫、京东、一号店等

2.4 一些专业术语

- SaaS: 软件即服务 --> 云计算的一块
- SOA: 面向服务 --> web service --> httpclient
- RPC: 远程过程调用
- RMI: 远程方法调用
- PV: (page view), 即页面浏览量;
用户每1次对网站中的每个网页访问均被记录1次。用户对同一页面的多次访问, 访问量累计
- UV: (unique visitor), 独立访客
指访问某个站点或点击某条新闻的不同IP地址的人数。在同一天内, uv只记录第一次进入网站的具有独立IP的访问者, 在同一天内再次访问该网站则不计数。
- PV与带宽:
 - 计算带宽大小需要关注两个指标: 峰值流量和页面的平均大小。
 - 计算公式是: 网站带宽= (PV * 平均页面大小 (单位MB) * 8)/统计时间 (换算到秒)
 - 为什么要乘以8?
 - 网站大小为单位是字节(Byte), 而计算带宽的单位是bit, 1Byte=8bit
 - 这个计算的是平均带宽, 高峰期还需要扩大一定倍数
- PV、QPS、并发
 - QPS: 每秒处理的请求数量。8000/s
 - 比如你的程序处理一个请求平均需要0.1s, 那么1秒就可以处理10个请求。QPS自然就是10, 多线程情况下, 这个数字可能就会有所增加。
 - 由PV和QPS如何需要部署的服务器数量?
 - 根据二八原则, 80%的请求集中在20%的时间来计算峰值压力:
 - $(\text{每日PV} * 80\%) / (3600s * 24 * 20\%) * \text{每个页面的请求数} = \text{每个页面每秒的请求数量}$
 - 然后除以服务器的QPS值, 即可计算得出需要部署的服务器数量

3 商城介绍

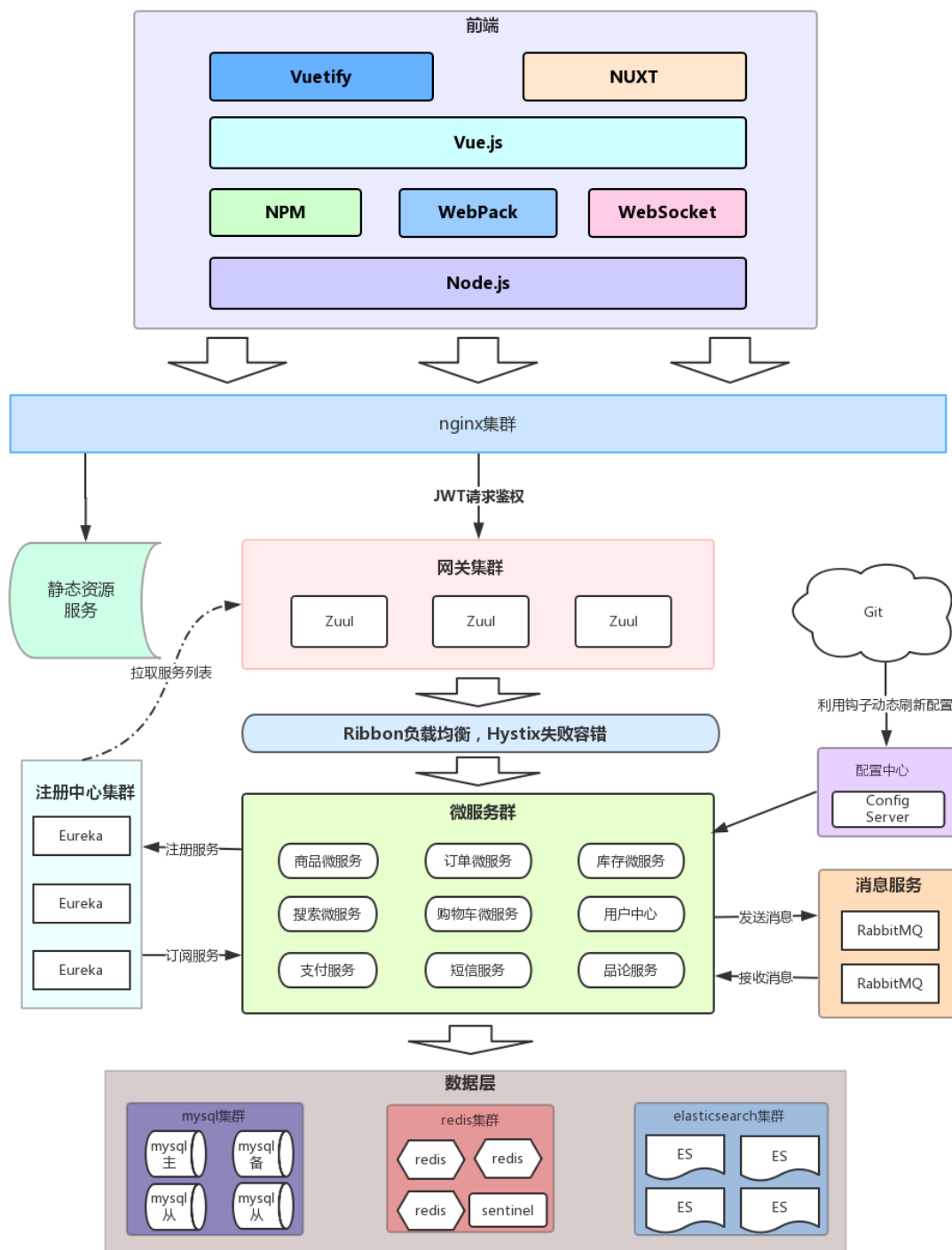
3.1 项目介绍

- 商城是一个全品类的电商购物网站 (B2C, 踩坑的几率会稍微小一点), 没有第三方商家入驻
- 用户可以在线购买商品、加入购物车、下单、秒杀商品
- 可以评论已购买商品
- 管理员可以在后台管理商品的上下架、促销活动
- 管理员可以监控商品销售状况
- 客服可以在后台处理退款操作
- 希望未来3到5年可以支持千万用户的使用

3.2 系统架构

3.2.1 架构图

商城架构缩略图，大图请参考课前资料：



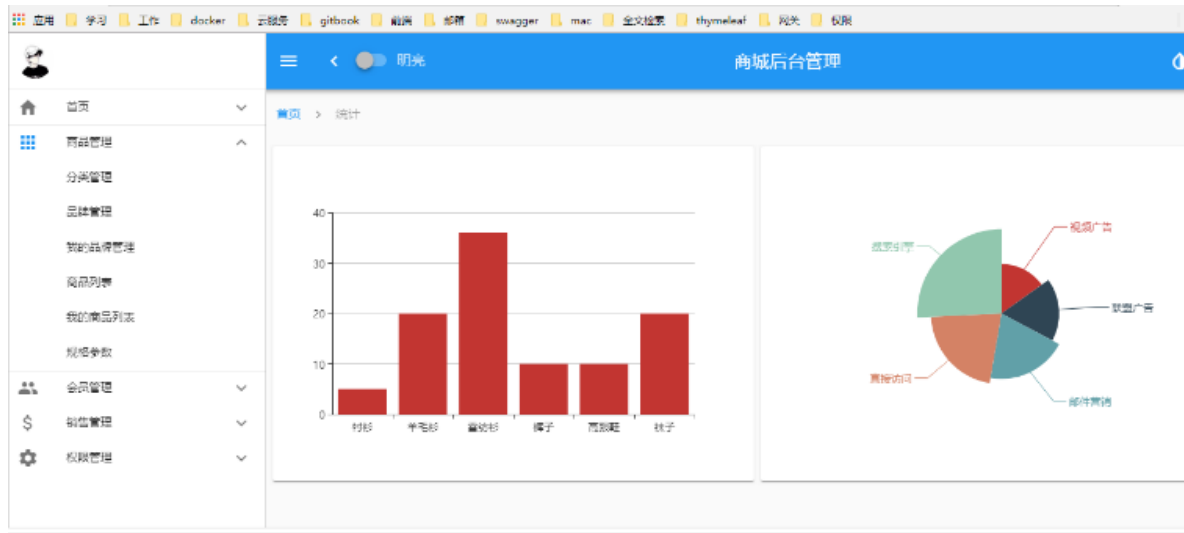
3.2.2 系统架构解读

整个商城可以分为两部分：后台管理系统、前台门户系统。

3.2.2.1 后台管理：

- 后台系统主要包含以下功能：
 - 商品管理，包括商品分类、品牌、商品规格等信息的管理

- 销售管理，包括订单统计、订单退款处理、促销活动生成等
 - 用户管理，包括用户控制、冻结、解锁等
 - 权限管理，整个网站的权限控制，采用JWT鉴权方案，对用户及API进行权限控制
 - 统计，各种数据的统计分析展示
- 后台系统会采用前后端分离开发，而且整个后台管理系统会使用Vue.js框架搭建出单页应用（SPA）。
- 预览图：



3.2.2.2 前台门户

- 前台门户面向的是客户，包含与客户交互的一切功能。例如：
 - 搜索商品
 - 加入购物车
 - 下单
 - 评价商品等等
- 前台系统我们会使用Nuxt结合Vue完成页面开发。出于SEO优化的考虑，我们将不采用单页应用。

无论是前台还是后台系统，都共享相同的微服务集群，包括：

- 商品微服务：商品及商品分类、品牌、库存等的服务
- 搜索微服务：实现搜索功能
- 订单微服务：实现订单相关
- 购物车微服务：实现购物车相关功能
- 用户中心：用户的登录注册等功能
- Eureka注册中心
- Zuul网关服务
- Spring Cloud Config配置中心
- ...

3.2.2.3 项目结构

- mingrui-shop-parent
 - mingrui-shop-basics
 - mingrui-shop-config-server
 - mingrui-shop-eureka-server 8761
 - mingrui-shop-zuul-server 8088 --> 80

- mingrui-shop-upload-server 80001
-
- mingrui-shop-commons
 - mingrui-shop-common-core
 -
- mingrui-shop-services
 - mingrui-shop-service-category 8100
 - mingrui-shop-service-brand 8200
 - mingrui-shop-service-spec 8300
- mingrui-shop-services-api
 - mingrui-shop-service-api-category
 - mingrui-shop-service-api-brand
 - mingrui-shop-service-api-spec

4 项目搭建

4.1 技术选型

4.1.1 前端技术：

- 基础的HTML、CSS、JavaScript（基于ES6标准）
- JQuery
- Vue.js 2.0以及基于Vue的UI框架：Vuetify
- 前端构建工具：WebPack
- 前端安装包工具：NPM
- Vue脚手架：Vue-cli
- Vue路由：vue-router
- ajax框架：axios
- 基于Vue的富文本框架：quill-editor

4.1.2 后端技术：

- 基础的SpringMVC、Spring 5.0和MyBatis3
- Spring Boot 2.2.2版本
- Spring Cloud 版本 Hoxton.SR1
- Redis-4.0
- RabbitMQ-3.8
- Elasticsearch-5.6.8
- nginx-1.10.2
- FastDFS - 5.0.8
- MyCat
- Thymeleaf
- JWT

4.2 开发环境

为了保证开发环境的统一，希望每个人都按照我的环境来配置：

- IDE：我们使用Idea
- JDK：统一使用JDK1.8

- 项目构建：maven3.3.9以上版本即可
- 版本控制工具：git

4.3 域名

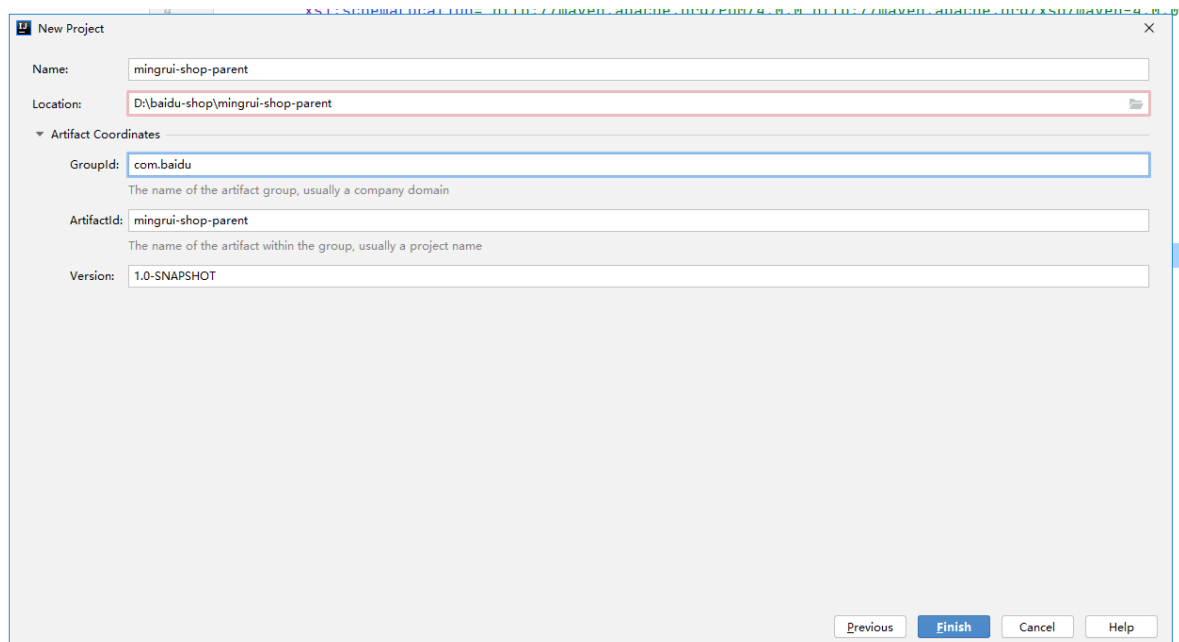
我们在开发的过程中，为了保证以后的生产、测试环境统一。尽量都采用域名来访问项目。

一级域名：www.mrshop.com

二级域名：manage.mrshop.com , api.mrshop.com

4.4 创建项目

4.4.1 创建父工程



4.4.1.1 删除src文件夹

父工程不需要写任何代码

4.4.1.2 pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.baidu</groupId>
  <artifactId>mingrui-shop-parent</artifactId>
  <version>1.0-SNAPSHOT</version>

  <!-- 父级项目不需要打包所有packaging的类型为pom-->
  <packaging>pom</packaging>

  <properties>
    <!-- 项目构建编码-->
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
```

```

        <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
        <!-- 声明JDK版本-->
        <java.version>1.8</java.version>
        <!--spring cloud 版本.注意此版本是建立在boot2.2.2版本上的-->
        <mr.spring.cloud.version>Hoxton.SR1</mr.spring.cloud.version>
    </properties>

    <!--boot 版本-->
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.1.RELEASE</version>
        <!--始终从仓库中获取,不从本地路径获取-->
        <relativePath />
    </parent>

    <dependencies>
        <!-- 集成commons工具类 -->
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
        </dependency>

        <!-- 集成lombok 框架 -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>

        <!--junit测试-->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
        </dependency>

        <!-- SpringBoot整合eureka客户端 -->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>

        <!--boot 测试模块-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

    </dependencies>

    <!-- 项目依赖,子级模块可以继承依赖-->
    <dependencyManagement>

        <dependencies>

            <!--cloud 依赖-->

```



```

        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>${mr.spring.cloud.version}</version>
            <type>pom</type>
            <!--解决maven单继承的问题-->
            <scope>import</scope>
        </dependency>

    </dependencies>

</dependencyManagement>

<!-- 注意： 这里必须要添加， 否者各种依赖有问题 -->
<repositories>
    <repository>
        <id>spring-milestones</id>
        <name>Spring Milestones</name>
        <url>https://repo.spring.io/libs-milestone</url>
        <snapshots>
            <enabled>false</enabled>
        </snapshots>
    </repository>
</repositories>

</project>

```

4.4.2 创建基础服务父工程

1. 在mingrui-shop-parent 工程名上右键-->new-->module
2. 项目名为: mingrui-shop-basics

4.4.2.1 删除src文件夹

父工程不需要写任何代码

4.4.2.2 pom.xml

只需要把打包方式设置为pom即可,暂时先不要引入其他依赖

```

<!-- 父级项目不需要打包所有packging的类型为pom-->
<packaging>pom</packaging>

```

4.4.3 创建公共(工具)工程

1. 在mingrui-shop-parent 工程名上右键-->new-->module
2. 项目名为: mingrui-shop-commoms

4.4.3.1 删除src文件夹

父工程不需要写任何代码

4.4.3.2 pom.xml

```

<!-- 父级项目不需要打包所有packging的类型为pom-->
<packaging>pom</packaging>

```

4.4.4 创建服务实现工程

1. 在mingrui-shop-parent 工程名上右键-->new-->module
2. 项目名为: mingrui-shop-service

4.4.4.1 删除src文件夹

父工程不需要写任何代码

4.4.4.2 pom.xml

```
<!-- 父级项目不需要打包所有packaging的类型为pom-->
<packaging>pom</packaging>

<dependencies>

    <!-- SpringBoot-整合web组件 -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- springcloud feign组件 -->
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-openfeign</artifactId>
    </dependency>

</dependencies>
```

4.4.5 创建服务接口工程

1. 在mingrui-shop-parent 工程名上右键-->new-->module
2. 项目名为: mingrui-shop-service-api

4.4.5.1 删除src文件夹

父工程不需要写任何代码

4.4.5.2 pom.xml

```
<!-- 父级项目不需要打包所有packaging的类型为pom-->
<packaging>pom</packaging>

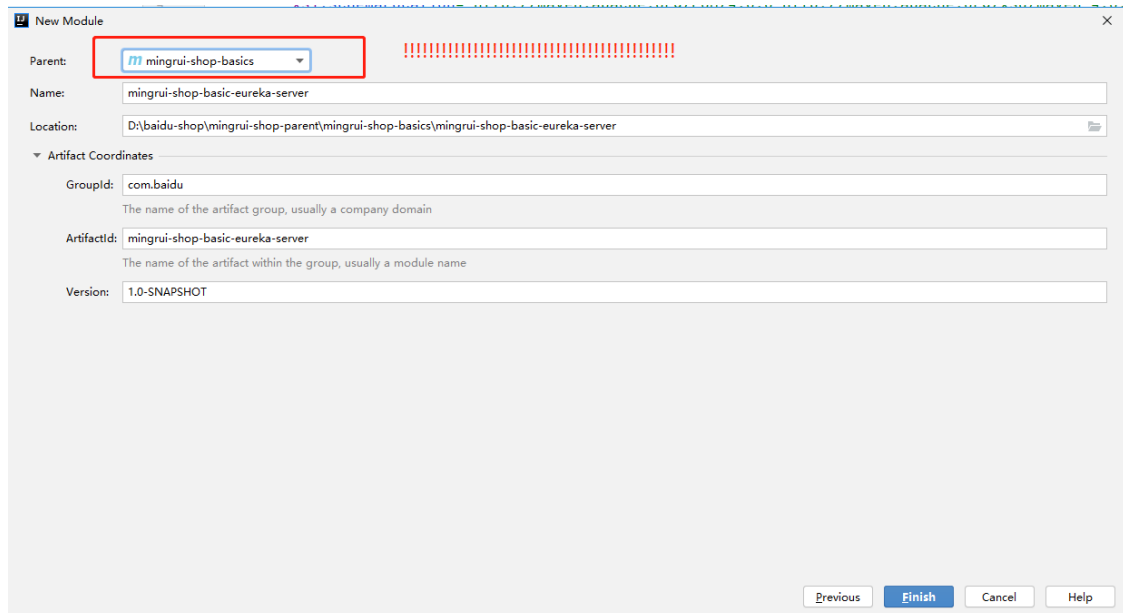
<dependencies>

    <!-- SpringBoot-整合web组件 -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

</dependencies>
```

4.4.6 创建eureka服务

在mingrui-shop-basics工程名上右键-->new-->module



注意:在点击finish之前一定要确认一遍当前创建工程的父工程是mingrui-shop-basics,接下来的项目创建的时候也是一样的..

4.4.6.1 pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <parent>
    <artifactId>mingrui-shop-basics</artifactId>
    <groupId>com.baidu</groupId>
    <version>1.0-SNAPSHOT</version>
  </parent>
  <modelVersion>4.0.0</modelVersion>

  <artifactId>mingrui-shop-basic-eureka-server</artifactId>

  <dependencies>
    <!--eureka 服务依赖-->
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
    </dependency>
  </dependencies>

</project>
```

4.4.6.2 application.yml

```
server:
  port: 8761

spring:
  application:
    name: eureka-server
```

```
eureka:
  client:
    # eureka服务url,值为map集合默认key为defaultZone
    service-url:
      defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka
    # 当前服务是否同时注册
    register-with-eureka: false
    # 去注册中心获取其他服务的地址
    fetch-registry: false
  instance:
    hostname: localhost
    # 定义服务续约任务（心跳）的调用间隔，单位：秒 默认30
    lease-renewal-interval-in-seconds: 1
    # 定义服务失效的时间，单位：秒 默认90
    lease-expiration-duration-in-seconds: 2
  server:
    # 测试时关闭自我保护机制，保证不可用服务及时踢出
    enable-self-preservation: false
```

4.4.6.3 启动类

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

/**
 * @ClassName RunEurekaServerApplication
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/14
 * @Version V1.0
 */
@SpringBootApplication
@EnableEurekaServer
public class RunEurekaServerApplication {

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