**微服务电商项目**

# 今日任务

**1.微服务电商项目介绍**

**2.微服务电商项目**

**3.微服务电商项目技术架构选型**

**4.微服务电商项目系统的架构模型**

**5.微服务电商项目模块的划分**

**6.构建项目Maven、编码相关说明**

**7.基于Nacos作为分布式注册中心**

**8.构建微服务项目之微信服务**

**9.构建微服务项目之会员服务**

**10.整合openfeign客户端实现rpc调用**

**11.定义微服务接口响应的规范**

# 相关说明

**项目是采用目前比较流行的SpringBoot2.0/SpringCloudAlibaba构建微服务电商项目，实现一套串联的微服务电商项目。**

**该项目涵盖从微服务电商需求讨论、数据库设计、技术选型、互联网安全架构、整合SpringCloudAlibaba核心组件Nacos/** **sentinel/** **Seata/** **canal等、分布式基础设施、使Docker+k8s+jenkins实现微服务自动化部署、项目上线域名配置、SEO优化、大数据统计分析、前后分离架构、微信开发等。**

# 需求讨论与技术架构选型

## 技术选型

**A. 项目采用SpringBoot2.0+SpringCloudAlibaba构建微服务电商项目**

**1.使用Nacos作为注册中心，实现服务治理**

**2.使用新一代Gateway网关框架管理服务请求入口**

**3.使用Ribbon实现本地负载均衡器和openFegin客户端调用工具**

**4.使用Sentinel服务保护框架(系统自适应限流、降级、热词限流等)**

**5.使用消息总线Stream RocketMQ和Kafka**

**6.微服务API接口安全控制与单点登陆系统CAS+JWT+Oauth2.0**

**B. 分布式基础设施环境构建**

**1.分布式任务调度平台XXL-Job**

**2.分布式日志采集系统ELK +kafka**

**3.分布式事务解决方案Seata和RocketMQ**

**4.分布式锁解决方案Zookeeper、Redis**

**5.分布式配置中心Nacos**

**6.高并发分布式全局ID生成雪花算法**

**7.分布式Session框架Spring-Session 禁止使用Session 使用token**

**8.分布式服务追踪与调用链ZipKin**

**9.基于canal解决MySQL与Redis一致性问题**

**10.基于网关统一解决微服务接口跨域问题**

**11.基于openresty+lua+Redis实现亿级商品详情页面**

**C.项目运营与部署环境**

**1.分布式设施环境，统一采用docker安装**

**2.使用jenkins+docker+k8s实现自动部署**

**3.微服务API管理ApiSwagger**

**4.使用GitLab代码管理**

**5.统一采用第三方云数据库**

**6.使用七牛云服务器对静态资源实现加速**

**7.构建企业级Maven私服**

**D.前端技术架构**

**1.全完采用Vue前后端分离架构模式**

**技术选型:**

**1 微服务电商项目需求讨论与技术架构选型**

**2 微服务架构环境搭建Maven私服仓库、GitLiab代码仓库**

**3 构建微服务微服务网关、api文档doc管理**

## 环境要求

**为了能够更好的学习互联网微服务架构，该项目对环境要求非常高，建议电脑配置CPU在I7 8700k以上处理、32GB内存或者电脑采用集群化部署。**

**1.JDK统一要求:JDK1.8以上**

**2.Maven 统一管理Jar**

**3.统一采用Docker安装软件**

**4.编码统一采用为UTF-8**

**5.开发工具完全采用idea**

## 构建项目

mt-shop-parent-----公共Pranet接口

-----mt-shop-basics----分布式基础设施

---------mt-shop-basics-alibaba-nacos—注册中心 8080

---------mt-shop-basics-alibaba-nacos—分布式配置中心 8080

---------mt-shop-basics-alibaba-seata 分布式事务解决方案 8730

---------mt-shop-basics-alibaba-canal mysql与redis一致性的问题

---------mt-shop-basics-springcloud-gateway—统一请求入口 80

---------mt-shop-basics-xuxueli-xxljob—分布式任务调度平台

---------mt-shop-basics-codingapi-zipKin —分布式调用链系统

-----mt-shop-service-api提供公共接口

------------ mt-shop-service-api-weixin 微信服务接口

------------ mt-shop-service-api-member会员服务接口

------------ mt-shop-service-api-sso sso服务接口

------------ mt-shop-service-api-item商品服务接口

------------ mt-shop-service-api-search 搜索服务接口

------------ mt-shop-service-api-pay聚合支付平台

------------ mt-shop-service-api-order订单服务接口

------------ mt-shop-service-api-spike 秒杀服务接口

------------ mt-shop-service-api-sms 消息服务平台

服务接口中包含内存内容: 实体类层、接口层

-----mt-shop-service-impl公共接口的实现

------------ mt-shop-service-weixin 微信服务接口实现 9090

------------ mt-shop-service-member会员服务接口实现 7070

------------ mt-shop-service-api-sso sso服务接口实现 6060

------------ mt-shop-service-tem商品服务接口实现 5050

------------ mt-shop-service-search 搜索服务接口实现 3030

------------ mt-shop-service-pay聚合支付平台接口实现 2020

------------ mt-shop-service-order订单服务接口实现 1010

------------ mt-shop-service-spike 秒杀服务接口 4040

------------ mt-shop-service-sms 消息服务平台 9810

-----mt-shop-common 工具类

---------mt-shop-common-core—核心工具类

----mt-shop-job---任务调度工程

----mt-shop-pay- reconcile --- 支付对账任务

----mt-shop-pay- sms --- 消息服务平台

-----mt-shop-portal 门户平台 vue工程

--------mt-shop-portal-web 门户网站 支持PC端和H5端访问

--------mt-shop-portal-sso 单点登陆系统

--------mt-shop-portal-search 搜索系统

--------mt-shop-portal-spike 秒杀系统

--------mt-shop-portal-cms 系统

**如果项目需要被其他项目继承的话，那么该项目选择maven类型为pom类型**

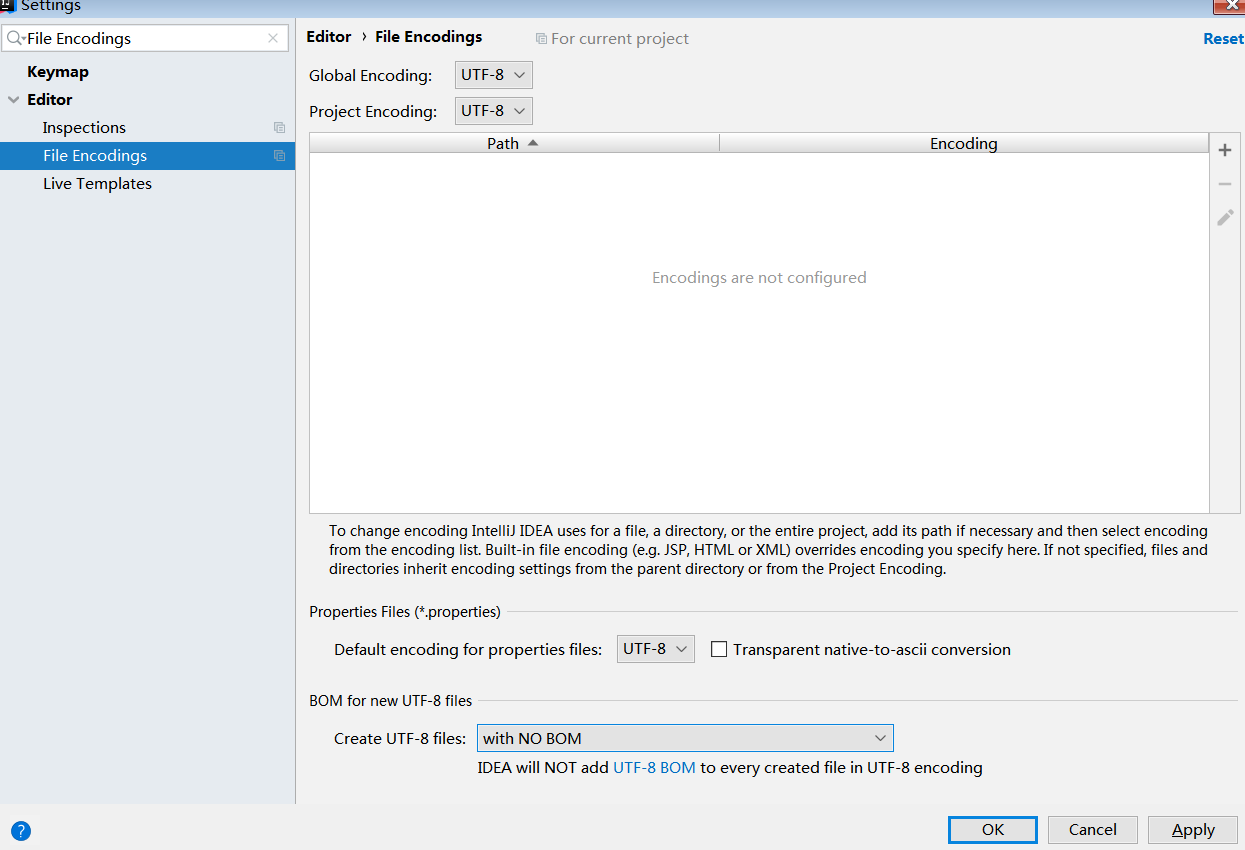
# 创建初始化的项目注意事项

## Maven阿里云加速镜像

|  |
| --- |
| <mirror>  <id>nexus-aliyun</id>  <mirrorOf>central</mirrorOf>  <name>Nexus aliyun</name>  <url>http://maven.aliyun.com/nexus/content/groups/public</url>  </mirror> |

apache-maven-3.3.9\conf\settings文件

## 构建项目编码为UTF-8



Settings 中找到File Encodings

## 构建项目注释模版信息

file->setting->Editor->Filr and Code Templates->Includes->File Header

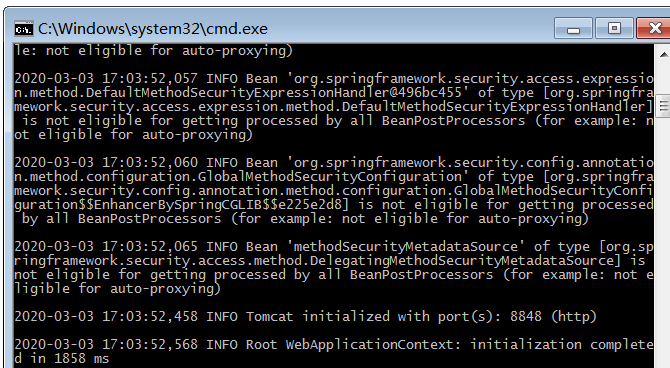
图片包含 屏幕截图

描述已自动生成

## 构建分布式注册中心nacos

解压nacos-server-1.1.4

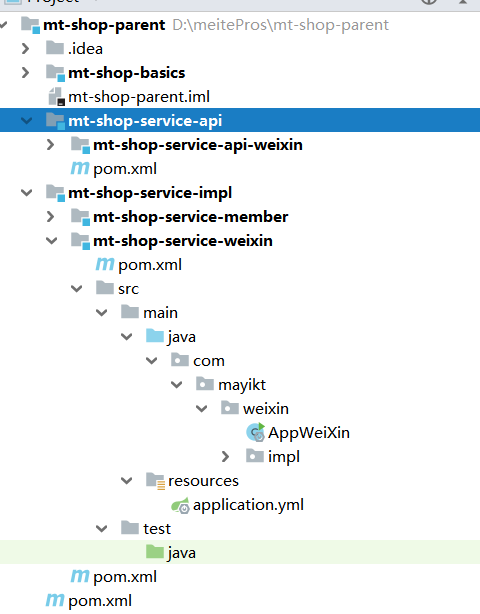
F:\path\alibaba\_nacos\nacos-server-1.1.4\nacos\bin 双击startup即可



Nacos端口为8848

<http://127.0.0.1:8848/nacos/#/login> 账号密码nacos naocs

## 构建微信服务接口项目



### mt-shop-parent

#### Maven依赖

|  |
| --- |
| <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.1.11.RELEASE</**version**>  <**relativePath**/> *<!-- lookup parent from repository -->* </**parent**>  <**properties**>  <**boot-version**>2.1.11.RELEASE</**boot-version**>  <**mybatis.starter.version**>2.0.0</**mybatis.starter.version**>  <**mapper.starter.version**>2.1.5</**mapper.starter.version**> </**properties**> <**dependencies**>  <**dependency**>  <**groupId**>org.projectlombok</**groupId**>  <**artifactId**>lombok</**artifactId**>  <**optional**>true</**optional**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-openfeign</**artifactId**>  </**dependency**>  *<!-- springboot 整合web组件-->* <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-web</**artifactId**>  </**dependency**>    <**dependency**>  <**groupId**>com.alibaba.cloud</**groupId**>  <**artifactId**>spring-cloud-alibaba-nacos-discovery</**artifactId**>  <**version**>2.1.1.RELEASE</**version**>  </**dependency**>  </**dependencies**>  <**dependencyManagement**>  <**dependencies**>  <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-dependencies</**artifactId**>  <**version**>Greenwich.RELEASE</**version**>  <**type**>pom</**type**>  <**scope**>import</**scope**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-dependencies</**artifactId**>  <**version**>${boot-version}</**version**>  <**type**>pom</**type**>  <**scope**>import</**scope**>  </**dependency**>  <**dependency**>  <**groupId**>com.alibaba.cloud</**groupId**>  <**artifactId**>spring-cloud-alibaba-dependencies</**artifactId**>  <**version**>2.1.1.RELEASE</**version**>  <**type**>pom</**type**>  <**scope**>import</**scope**>  </**dependency**>   </**dependencies**> </**dependencyManagement**> |

#### mt-shop-service-api

##### mt-shop-service-api-weixin

|  |
| --- |
| **public interface** WeiXinService {   */\*\*  \* 微信应用信息  \*  \** ***@return*** *\*/* @GetMapping(**"/getAppInfo"**)  String getAppInfo(@RequestParam(**"userId"**) Long userId); } |

#### mt-shop-service-impl

##### mt-shop-service-weixin

|  |
| --- |
| @RestController **public class** WeiXinServiceImpl **implements** WeiXinService {   **public** String getAppInfo(Long userId) {  **return "微信服务接口 userId:"** + userId;  } } |
|  |

### application.yml配置文件

|  |
| --- |
| **server**:  **port**: 9090 **spring**:  **application**:  **name**: mayikt-weixin  **cloud**:  **nacos**:  **discovery**:  *##nacos服务注册地址* **server-addr**: 127.0.0.1:8848 |

### 启动微信服务接口

|  |
| --- |
| @SpringBootApplication **public class** AppWeiXin {  **public static void** main(String[] args) {  SpringApplication.*run*(AppWeiXin.**class**);  } } |

## 构建会员服务接口项目

### mt-shop-service-member

|  |
| --- |
| @RestController **public class** MemberServiceImpl **extends** BaseApiService {  @Autowired  **private** WeiXinServiceFeign **weiXinServiceFeign**;   @GetMapping(**"/menberToWeiXin"**)  **public** String menberToWeiXin(Long userId) {  **return weiXinServiceFeign**.getAppInfo(userId);  }   @GetMapping(**"/getMember"**)  **public** BaseResponse<String> getMember() {  **return** setResultSuccess(**"123456"**);  } } |

### 重写feign客户端接口

|  |
| --- |
| @FeignClient(name = **"mayikt-weixin"**) **public interface** WeiXinServiceFeign **extends** WeiXinService {  *// /\*\* // \* 微信应用信息 // \* // \* @return // \*/ // @GetMapping("/getAppInfo") // String getAppInfo(@RequestParam("userId") Long userId);* } |

### application.yml

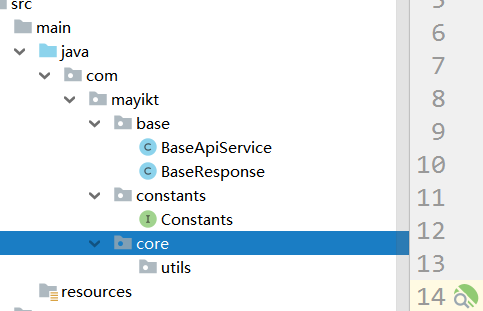
|  |
| --- |
| **server**:  **port**: 7070 **spring**:  **application**:  **name**: mayikt-member  **cloud**:  **nacos**:  **discovery**:  **server-addr**: 127.0.0.1:8848 |

### 启动会员服务项目

|  |
| --- |
| @SpringBootApplication @EnableFeignClients **public class** AppMember {  **public static void** main(String[] args) {  SpringApplication.*run*(AppMember.**class**);  } } |

## 定义接口响应的规范

{"code":200,"msg":"success","data":"123456"}



用法

|  |
| --- |
| @RestController **public class** MemberServiceImpl **extends** BaseApiService {  @Autowired  **private** WeiXinServiceFeign **weiXinServiceFeign**;   @GetMapping(**"/menberToWeiXin"**)  **public** String menberToWeiXin(Long userId) {  **return weiXinServiceFeign**.getAppInfo(userId);  }   @GetMapping(**"/getMember"**)  **public** BaseResponse<String> getMember() {  **return** setResultSuccess(**"123456"**);  } } |

## 常见报错问题

status 405 reading WeiXinServiceFeign#getAppInfo(Long)

传递参数加上 该注解就可以 @RequestParam("userId")