**每特教育互联网微服务电商项目课程安排**

# 课程内容

1、搭建分布式配置中心携程阿波罗

2、微服务项目整合阿波罗平台

3、网关整合阿波罗配置中心

4、自定义swagger文档配置文件

5、动态刷新swagger文档配置

6、微服务中，配置文件如何区分？

# 相关说明

**该项目“基于SpringCloud2.x构建微服务电商项目”由每特教育|蚂蚁课堂版权所有，未经过允许的情况下，私自分享视频和源码属于违法行为。**

**该项目作者:97后互联网架构师-余胜军**

**相关网址:www.mayikt.com**

**联系方式:微信yushengjun644、QQ644064779**

**该想项目是采用目前比较流行的SpringBoot/SpringCloud构建微服务电商项目，实现一套串联的微服务电商项目，能完全掌握该知识，可以在一线城市拿到月薪25+k薪资。**

**完全符合一线城市微服务电商的需求，对中国程序猿学习微服务电商架构，有非常大的帮助，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SpringCloud各自组件、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、大数据统计分析等。**

**注意：该项目会在2019年4月份左右上线GitHub和码云等开源网站平台。**

**如需参加培训，可以加余老师QQ644064779**

## 学习时间

**学习周期大概四个月时间，每周 2 4 6 晚上20:30-22:30，一个半小时编码课、半个小时面试题辅导。**

**2019年01月03日本周四，开始干我们微服务电商结业项目啦！！希望同学们，坚持哦！**

# 需求讨论与技术架构选型

## 技术选型

**A. 项目采用SpringBoot2.x+SpringCloud2.x构建微服务电商项目**

**1.使用SpringCloudEureka作为注册中心，实现服务治理**

**2.使用Zuul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FeginHttp客户端调用工具**

**4.使用Hystrix服务保护框架(服务降级、隔离、熔断、限流)**

**5.使用消息总线Stream RabbitMQ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**

**3.分布式事务解决方案LCN**

**4.分布式锁解决方案Zookeeper、Redis**

**5.分布式配置中心携程阿波罗**

**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**

**8.分布式服务追踪与调用链ZipKin**

**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**

## 课程安排

**第一节 需求讨论与技术架构选型**

**第二节 微服务架构环境搭建Maven私服仓库、GitLiab代码仓库**

**第三节 构建微服务电商项目与数据库实战设计**

**第四节 分布式基础设施之分布式配置中心环境搭建**

**第五节 微服务电商项目-会员服务-联合登陆**

**第六节 微服务电商项目-会员服务-唯一登陆**

**第七节 微服务电商项目-会员服务-SSO联合登陆**

**第八节 微服务电商项目-微服务公众号-构建微信公众号**

**第九节 微服务电商项目-消息服务平台-消息服务平台**

**第十节 微服务电商项目-前端技术-VUE**

**第十一节 微服务电商项目-商品服务-数据设计**

**第十二节 微服务电商项目-商品服务-搜索服务**

**第十三节 微服务电商项目-商品服务-购物车设计**

**第十四节 微服务电商项目-订单服务-订单服务**

**第十五节 微服务电商项目-聚合支付-数据设计**

**第十六节 微服务电商项目-聚合支付-银联支付**

**第十七节 微服务电商项目-聚合支付-支付宝支付**

**第十八节 微服务电商项目-秒杀服务-秒杀抢购**

**第十九节 微服务电商项目-后台运营平台设计**

**第二十节 微服务电商项目ELK日志分析**

**第二十一节 微服务电商项目安全架构**

**第二十二节 微服务电商项目静态资源CDN加速**

**第二十三节 使用jenkins+docker+k8s实现自动部署**

**第二十四节 微服务电商项目上线部署环境**

**第二十五节 微服务电商项目大数据用户行为分析**

**最终以实际课程表格为准。**

## 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I5、32GB内存或者电脑采用集群化部署。**

**1.JDK统一要求:JDK1.8K**

**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IDE或者Eclipse**

## 系统架构

**每特教育独创电商项目完全采用微服务架构模式，如果对微服务架构概念还是不够理解的话，可以参考SpringCloud2.x课程。**

**详细参考架构文档:**

****

## 构建项目

meite-shop-parent-----公共Pranet接口

-----meite-shop-basics----分布式基础设施

---------meite-shop-basics-springcloud-eureka—注册中心 8080

---------meite-shop-basics-apollo-config-server—阿波罗分布式配置中心

---------meite-shop-basics-springcloud-zuul—统一请求入口 80

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

---------meite-shop-basics-codingapi-lcn—分布式事务解决框架

---------meite-shop-basics-codingapi- ZipKin —分布式调用链系统

-----meite-shop-service-api提供公共接口

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

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

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

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

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

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

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

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

服务接口中包含内存内容: 实体类层、接口层

-----meite-shop-service-impl公共接口的实现

------------ meite-shop-service-weixin 微信服务接口实现

------------ meite-shop-service-member会员服务接口实现

------------ meite-shop-service-api-sso sso服务接口实现

------------ meite-shop-service-tem商品服务接口实现

------------ meite-shop-service-search 搜索服务接口实现

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

------------ meite-shop-service-order订单服务接口实现

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

-----meite-shop-common 工具类

---------meite-shop-common-core—核心工具类

-----meite-shop-portal 门户平台

--------meite-shop-portal-web 门户网站

--------meite-shop-portal-sso 单点登陆系统

--------meite-shop-portal-search 搜索系统

--------meite-shop-portal-spike 秒杀系统

--------meite-shop-portal-cms 系统

**后期有新的功能在加。**

**预计40个项目**

**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**

## 版本说明

**SpringCloud2.x以上版本**

## 注意事项

1. [统一采用@GetMapping](mailto:1.统一采用@GetMapping)**或者@PostMapping实现接口映射**

# 环境构建

步骤①创建以下项目，maven类型选择为Pom类型

1. 创建meite-shop-parent
2. 创建meite-shop-basics
3. 创建meite-shop-portal
4. 创建meite-shop-service
5. 创建meite-shop-service-api

步骤②分布式基础设施eureka

1. 创建meite-shop-basics-springcloud-eureka

步骤③创建微信、会员服务接口

1. 创建meite-shop-service-api-member
2. 创建meite-shop-service-api-weixin

步骤④创建微信、会员服务接口实现maven类型选择为Jar类型

1.创建meite-shop-service-member

1.创建meite-shop-service-weixin

## 分布式基础设施环境环境准备

### 基于Docker部署GitLab环境搭建

**建议虚拟机内存2G以上**

**1.下载镜像文件**

|  |
| --- |
| docker pull beginor/gitlab-ce:11.0.1-ce.0 |

**注意：一定要配置阿里云的加速镜像**

1. **创建GitLab 的配置 (etc) 、 日志 (log) 、数据 (data) 放到容器之外， 便于日后升级， 因此请先准备这三个目录。**

|  |
| --- |
| **mkdir -p /mnt/gitlab/etc**  **mkdir -p /mnt/gitlab/log**  **mkdir -p /mnt/gitlab/data** |

1. **运行GitLab容器**

|  |
| --- |
| **docker run \**  **--detach \**  **--publish 8443:443 \**  **--publish 8090:80 \**  **--name gitlab \**  **--restart unless-stopped \**  **-v /mnt/gitlab/etc:/etc/gitlab \**  **-v /mnt/gitlab/log:/var/log/gitlab \**  **-v /mnt/gitlab/data:/var/opt/gitlab \**  **beginor/gitlab-ce:11.0.1-ce.0** |

停止docker容器，并且删除

Docker stop 容器id

Docker rm 容器id

systemctl stop firewalld

4.修改/mnt/gitlab/etc/gitlab.rb

**把external\_url改成部署机器的域名或者IP地址**

|  |
| --- |
| **vi /mnt/gitlab/etc/gitlab.rb** |

**external\_url 'http://192.168.212.227'**

1. **修改/mnt/gitlab/data/gitlab-rails/etc/gitlab.yml**

**vi /mnt/gitlab/data/gitlab-rails/etc/gitlab.yml**

**找到关键字 \* ## Web server settings \***

**将host的值改成映射的外部主机ip地址和端口，这里会显示在gitlab克隆地址**



到此为止，gitlab的web管理页面就可以正常访问



**修改默认root账户密码**

**Yushengjun644**

#### GitLab创建项目





#### Eclipse提交项目

1. 项目初始化



选择Git

创建存储库





提交代码





#### 新增用户权限

1. 注册用户mayikt 密码yushengjun644 授权Git代码地址
2. 设置中找到成员，添加项目权限即可

### SwaggerApi管理

#### meite-shop-service-api

##### 引入Maven依赖

|  |
| --- |
| <!-- swagger-spring-boot -->  <dependency>  <groupId>com.spring4all</groupId>  <artifactId>swagger-spring-boot-starter</artifactId>  <version>1.7.0.RELEASE</version>  </dependency> |

#### meite-shop-service-member

配置文件新增

|  |
| --- |
| ####swagger相关配置  swagger:  base-package: com.mayikt.member.service  title: SpringCloud2.x构建微服务电商项目-微信服务接口  description: 该项目“基于SpringCloud2.x构建微服务电商项目”由每特教育|蚂蚁课堂版权所有，未经过允许的情况下，私自分享视频和源码属于违法行为。  version: 1.1  terms-of-service-url: www.mayikt.com  contact:  name: 97后互联网架构师-余胜军  email: 644064779@qq.com |

@EnableSwagger2Doc 开启Swagger生成文件

#### meite-shop-service-api-weixin

配置文件新增

|  |
| --- |
| ####swagger相关配置  swagger:  base-package: com.mayikt.weixin.service  title: SpringCloud2.x构建微服务电商项目-微信服务接口  description: 该项目“基于SpringCloud2.x构建微服务电商项目”由每特教育|蚂蚁课堂版权所有，未经过允许的情况下，私自分享视频和源码属于违法行为。  version: 1.1  terms-of-service-url: www.mayikt.com  contact:  name: 97后互联网架构师-余胜军  email: 644064779@qq.com |

@EnableSwagger2Doc 开启Swagger生成文件

http://127.0.0.1:8300/swagger-ui.html

http://127.0.0.1:8200/swagger-ui.html

#### 使用微服务网关统一管理Api

创建项目meite-shop-basics-springcloud-zuul

##### Maven依赖

|  |
| --- |
| <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-zuul</artifactId>  </dependency>  <!-- swagger-spring-boot -->  <dependency>  <groupId>com.spring4all</groupId>  <artifactId>swagger-spring-boot-starter</artifactId>  <version>1.7.0.RELEASE</version>  </dependency>  </dependencies> |

##### 统一管理API代码

|  |
| --- |
| @SpringBootApplication  @EnableEurekaClient  @EnableZuulProxy  @EnableSwagger2Doc  **public** **class** AppGateWay {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(AppGateWay.**class**, args);  }  // 添加文档来源  @Component  @Primary  **class** DocumentationConfig **implements** SwaggerResourcesProvider {  @Override  **public** List<SwaggerResource> get() {  List resources = **new** ArrayList<>();  // app-itmayiedu-order  resources.add(swaggerResource("app-mayikt-member", "/app-mayikt-member/v2/api-docs", "2.0"));  resources.add(swaggerResource("app-mayikt-weixin", "/app-mayikt-weixin/v2/api-docs", "2.0"));  **return** resources;  }  **private** SwaggerResource swaggerResource(String name, String location, String version) {  SwaggerResource swaggerResource = **new** SwaggerResource();  swaggerResource.setName(name);  swaggerResource.setLocation(location);  swaggerResource.setSwaggerVersion(version);  **return** swaggerResource;  }  }  } |

##### 相关配置

|  |
| --- |
| ###服务启动端口号  server:  port: 80  ###服务名称(服务注册到eureka名称)  spring:  application:  name: app-mayikt-zuul  ###服务注册到eureka地址  eureka:  client:  service-url:  defaultZone: http://localhost:8100/eureka  ### 配置网关反向代理  zuul:  routes:  api-a:  ### 以 /api-weixin/访问转发到会员服务  path: /api-weixin/\*\*  serviceId: app-mayikt-weixin  api-b:  ### 以 /api-member/访问转发到订单服务  path: /api-member/\*\*  serviceId: app-mayikt-member |

### Maven私服环境搭建

1.一些无法从外部仓库下载的构件，例如内部的项目还能部署到私服上，以便供其他依赖项目使用。

2. 为了节省带宽和时间，在局域网内架设一个私有的仓库服务器，用其代理所有外部的远程仓库。当本地Maven项目需要下载构件时，先去私服请求，如果私服没有，则再去远程仓库请求，从远程仓库下载构件后，把构件缓存在私服上。这样，及时暂时没有Internet链接，由于私服已经缓存了大量构件，整个项目还是可以正常使用的。同时，也降低了中央仓库的负荷。



#### 什么场景用Maven私服呢？

在实际开发中，项目中可能会用到第三方的jar、内部通讯的服务接口都会打入到公司的私服中。

#### 基于Docker搭建Maven私服

|  |
| --- |
| 1. 下载一个nexus3的镜像   docker pull sonatype/nexus3   1. 将容器内部/var/nexus-data挂载到主机/root/nexus-data目录。   docker run -d -p 8081:8081 --name nexus -v /root/nexus-data:/var/nexus-data --restart=always sonatype/nexus3 |

关闭防火墙,访问<http://ip:8081>

Maven私服启动容器稍微比较慢，等待1分钟即可。

默认登陆账号 admin admin123

#### 创建Maven私服仓库

##### 创建私服仓库

创建仓库，点击Create repository,然后选择maven2(hosted)然后输入仓库名称（test-release）。在version policy中选择这个仓库的类型，这里选择release,在Deployment policy中选择Allow redeploy（这个很重要）.

##### 创建私服账号

点击左侧菜单栏的Users菜单，然后点击Create local user.我这里创建了一个用户，账号密码都是：mayikt

Mayikt-release

##### 本地settings.xml

|  |
| --- |
| <servers>  <server>  <id>mayikt</id>  <username>mayikt</username>  <password>mayikt</password>  </server>    </servers> |

##### 创建一个Maven工程

创建一个maven工程，并且打包到maven私服。

相关配置

|  |
| --- |
| <!--注意限定版本一定为RELEASE,因为上传的对应仓库的存储类型为RELEASE -->  <!--指定仓库地址 -->  <distributionManagement>  <repository>  <!--此名称要和.m2/settings.xml中设置的ID一致 -->  <id>mayikt</id>  <url>http://192.168.212.230:8081/repository/mayikt-release/</url>  </repository>  </distributionManagement>  <build>  <plugins>  <!--发布代码Jar插件 -->  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-deploy-plugin</artifactId>  <version>2.7</version>  </plugin>  <!--发布源码插件 -->  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-source-plugin</artifactId>  <version>2.2.1</version>  <executions>  <execution>  <phase>package</phase>  <goals>  <goal>jar</goal>  </goals>  </execution>  </executions>  </plugin>  </plugins>  </build> |

mvn deploy

##### 测试依赖信息

|  |
| --- |
| <dependencies>  <dependency>  <groupId>com.mayikt</groupId>  <artifactId>mayikt\_springboot</artifactId>  <version>0.0.1-RELEASE</version>  </dependency>  </dependencies>  <repositories>  <repository>  <id>mayikt</id>  <url>http://192.168.212.233:8081/repository/mayikt-release/</url>  </repository>  </repositories> |

#### 如何判断文件是否发生改变

如何知道一个文件是否改变了呢？当然是用比较文件hash值的方法，文件hash又叫文件签名，文件中哪怕一个bit位被改变了，文件hash就会不同。比较常用的文件hash算法有MD5和SHA-1。

### 构建分布式配置中心阿波罗

#### 为什么要使用配置文件中心？

为什么要使用分布式配置中心？ 统一管理微服务配置文件，可以实现动态化刷新配置文件。

阿波罗 SpringCloudConfig

为什么我们要使用阿波罗 不使用SpringCloudConfig

阿波罗配置文件存放在数据库中，SpringCloudConfig存放在Git里面。

#### 搭建分布式配置中心阿波罗

1. 下载aploll配置中心 https://github.com/nobodyiam/apollo-build-scripts
2. 上传apollo-build-scripts-master文件到服务器中
3. unzip apollo-build-scripts-master.zip 解压配置文件

如果没有unzip命令的话，安装zip插件 yum -y install zip unzip

1. 配置数据策略

修改demo.sh账号:cdb-8y8qmojr.gz.tencentcdb.com:10010 root Ww861642669+

1. 启动阿波罗 ./demo.sh start

<https://github.com/ctripcorp/apollo/wiki/Apollo%E9%85%8D%E7%BD%AE%E4%B8%AD%E5%BF%83%E4%BB%8B%E7%BB%8D>

systemctl stop firewalld.service

默认账号密码 Apollo admin

#### 服务客户端集成配置文件

1.将本地配置存入到阿波罗平台中。

转换工具

<http://www.toyaml.com/index.html>

2.引入Maven依赖

|  |
| --- |
| <dependency>  <groupId>com.ctrip.framework.apollo</groupId>  <artifactId>apollo-client</artifactId>  <version>1.0.0</version>  </dependency>  <dependency>  <groupId>com.ctrip.framework.apollo</groupId>  <artifactId>apollo-core</artifactId>  <version>1.0.0</version>  </dependency> |

3.创建 application.properties

|  |
| --- |
| app.id=mayikt644064779  apollo.meta=http://192.168.212.236:8080 |

4.项目启动开启阿波罗配置文件

@EnableApolloConfig

5.修改/opt/settings/server.properties（Mac/Linux）或C:\opt\settings\server.properties（Windows）文件，设置env为DEV：

env=DEV

#### 网关服务集成阿波罗

|  |
| --- |
| @SpringBootApplication  @EnableEurekaClient  @EnableZuulProxy  @EnableSwagger2Doc  @EnableApolloConfig  **public** **class** AppGateWay {  // 获取ApolloConfig  @ApolloConfig  **private** Config appConfig;  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(AppGateWay.**class**, args);  }  // 添加文档来源  @Component  @Primary  **class** DocumentationConfig **implements** SwaggerResourcesProvider {  @Override  **public** List<SwaggerResource> get() {  // 开启监听，配置文件发生改变需要更改  appConfig.addChangeListener(**new** ConfigChangeListener() {  @Override  **public** **void** onChange(ConfigChangeEvent changeEvent) {  get();  }  });  **return** resources();  }  /\*\*  \* 从阿波罗服务器中获取resources  \*  \* **@return**  \*/  **private** List<SwaggerResource> resources() {  List resources = **new** ArrayList<>();  // app-itmayiedu-order  // 网关使用服务别名获取远程服务的SwaggerApi  String swaggerDocJson = swaggerDocument();  JSONArray jsonArray = JSONArray.*parseArray*(swaggerDocJson);  **for** (Object object : jsonArray) {  JSONObject jsonObject = (JSONObject) object;  String name = jsonObject.getString("name");  String location = jsonObject.getString("location");  String version = jsonObject.getString("version");  resources.add(swaggerResource(name, location, version));  }  **return** resources;  }  /\*\*  \* 获取swaggerDocument配置  \*  \* **@return**  \*/  **private** String swaggerDocument() {  String property = appConfig.getProperty("mayikt.zuul.swaggerDocument", "");  **return** property;  }  **private** SwaggerResource swaggerResource(String name, String location, String version) {  SwaggerResource swaggerResource = **new** SwaggerResource();  swaggerResource.setName(name);  swaggerResource.setLocation(location);  swaggerResource.setSwaggerVersion(version);  **return** swaggerResource;  }  }  } |

自定义Swagger文档配置mayikt.zuul.swaggerDocument

[

{

"name": "app-mayikt-member",

"location": "/app-mayikt-member/v2/api-docs",

"version": "2.0"

},

{

"name": "app-mayikt-weixin",

"location": "/app-mayikt-weixin/v2/api-docs",

"version": "2.0"

}

]

思考：在微服务项目中，阿波罗分布式配置中心平台如何区分不同项目的配置文件？

每一个团队有自己独立appid，区分不同团队的配置文件。

注意：不是所有的配置文件都会在阿波罗平台修改后，就会立马生效，因为没有采用监听刷新配置文件。

Name

Age