应用系统体系架构 — 作业5

# 学号： 522031910213 姓名： 朱涵 得分：

## 编写微服务代码并实现指定的任务

## 思路：

## 用nacos作为服务发现注册中心 用spring cloud gateway作为网关

## 具体实现：

## 拆分微服务 将原本的项目转移到子模块里 命名为main-service；新建search-service模块来实现新的功能；另外新建shared模块来存放各个服务需要共享的一些类 比如entity类和DTO类；新建gateway模块实现gateway的路由转发逻辑

## 在建立子模块后 适当调整pom.xml中的依赖结构 把共用的保留在父模块 把特殊的依赖放入子模块中（比如redis和kafka）

## 2.编写新功能：同样建立控制层、服务层等软件包 然后简单的编写一个新接口 映射到"/api/search/author" 上 这个接口会根据输入的标题返回找到的书的作者名字

## 下图是调用接口的结果：

## 

## 3.配置服务注册：使用nacos作为服务注册中心 先在yml配置里配置好服务名称、服务注册中心的url等 然后下载新版的nacos 使用终端启动 最后启动各个微服务即可 访问nacos页面 可以发现三个服务都已注册 如下图

## 

## 4.配置gateway：gateway作为一个网关 会处理所有请求并转发给对应的服务 同时还可以配置负载均衡 在这里我们配置gateway的端口为8000 另外两个服务的端口分别为8080和8081

## gateway的路由配置可以在application.yml里静态配置 也支持在nacos的配置管理中进行动态配置 我选择先静态配置 下图是配置内容：

## 即把/api/search/\*\*的请求转发给search-service 剩下所有其他请求转发给main-service

## 接下来可以向8000端口发送请求 如下图：

## 

## 左边是向search服务发送请求 右边是向main-service发送一个登录请求 可以看到gateway正确的进行了转发 未来添加的所有服务都可以通过gateway来路由 后续用nginx把8000端口的请求反向代理为http的端口80后 用户就可以不指定端口来访问项目前端了

## 注：其中鉴权机制仍可以正常运行 因为spring cloud gateway会把请求带有的cookie一起发送给服务 并且响应中的set-cookies也会带上

## 另外 如果要涉及到微服务框架中的鉴权 有两种方案：1 使用jwt 因为是无状态的鉴权 微服务不需要做什么修改 只需要在各自服务内部进行鉴权即可 2 使用session 因为是有状态的鉴权 要么把所有需要鉴权的请求集中发送给某一个服务 要么所有服务各自进行鉴权 但是需要使用redis等方式进行session的共享 保证不同服务可以共享用户的状态 在此就不赘述了

## 下图是一个简易的前端页面 来实现作者搜索功能：

## 

## 解释Gateway和Service Registry在微服务架构中都起到了什么作用：

## 服务注册中心会把注册的服务存储为类似于<服务名，ip，端口>的map（实际可能更复杂） 当有客户端想向某一个服务发送请求时 比如main-service 先向注册中心查询这个服务有没有健康的实例 如果有 返回对应的ip和端口 客户端就可以向对应的url发送请求了 注册中心在这里起的作用类似于dns的解析 也就是让微服务可以**注册**并**发现**其他微服务

## 除此之外 服务注册中心可能还可以提供负载均衡 健康检查 动态配置（比如nacos就支持）等功能

## 网关则是整个微服务框架的入口点 负责接受所有请求并根据路由规则转发给对应的微服务 然后返回响应给客户端 可以简单认为网关是一个代理 把用户的请求代理到对应的微服务 把微服务的响应再代理回给用户

## 在我的代码中可以看到 把api/search/\*\*的请求转发给了lb://search-service 其中lb://{服务名称}的格式是由spring-load-balance提供的 其会把服务名称根据注册中心解析为对应的ip和端口 然后带有负载均衡的对某一个实例进行请求

## 除此之外 网关也可以做安全管理（筛选请求）负载均衡等功能

## 总体而言 网关是一个代理入口 服务注册中心是一个管理组件 维护各个服务的名称、ip和端口等信息 在其他中间件的帮助下（比如openfeign或是spring-loadbalance等） 用户的请求可以被正确的解析和转发给对应的微服务

## 编写函数式服务代码并实现指定的任务：

## 实现：

## 1. 创建新模块名为price-service 导入springcloud-funciton的依赖 端口为8082

## 2. 为方便 在启动类编写一个计算总价的函数Bean 接受一个Flux<String> 其格式是json数组 内容是orderitem的列表 遍历其中每一个item 累加其数量与其价格之积 最终得到总价 代码如下图：

## 3. 在gateway中加入新路由 将/computePrice的请求转发给部署在8082端口的price-service 从而可以向网关访问此服务

## 4. 前端需要计算总价时调用这个接口即可 下图是调用接口的示例：

## 

## 正确回答1.A.ii中指定的问题：

## 1.函数式服务的无状态指的是什么：函数即接受一个输入产生唯一的输出 也就是说服务的调用与之前的调用状态完全无关 只取决于输入的参数 如这个计算总价的函数式服务 返回的结果只和输入的orderItems有关 而和用户的状态等无关

## 2. 函数式服务为什么容易扩展：既然函数式服务无状态 就不用和其他微服务一样考虑状态的共享 比如需要鉴权的有状态服务在添加实例或者扩展服务时需要考虑如何在实例和服务之间共享session状态 而计算总价的函数式服务如果要扩展 只需要简单的多部署几个实例即可 配合docker或k8s 这种扩展将变得十分简单 从而支持轻松的负载均衡