微服务实战： 外卖订单系统

1、需求分析

主要由客户端和后台管理系统两部分构成。

> 客户端

- 普通用户即可访问
- 具体功能

  用户登录、用户退出、菜品订购、我的订单

> 后台管理系统

- 管理员方可访问
- 具体功能

  管理员登录、管理员退出、添加菜品、查询菜品、修改菜品、删除菜品、订单处理、添加用户、查询用户、删除用户

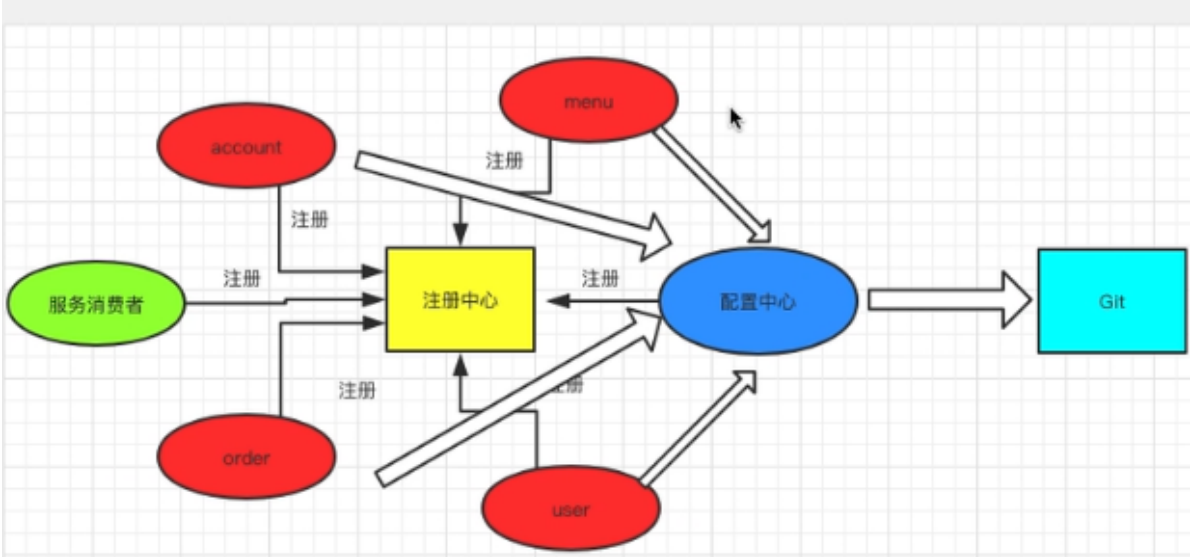
> 功能拆分

- account 微服务：提供账户服务，负责用户（普通用户和管理员）的登录和退出
- menu 微服务：提供菜品服务，负责菜品的增删改查
- order 微服务：提供订单服务，负责订单的增删改查和处理
- user 微服务：提供用户服务：负责用户的增删改查

以上为4个服务提供者，额外再分离出一个服务消费者负责调用以上四个服务提供者。服务消费者包含了客户端的前端页面和后台接口以及后台管理系统的前端页面和后台接口。

用户和管理员直接访问的资源都保存在服务消费者中，服务消费者根据具体的需求调用四个废物提供者的业务逻辑，通过 Feign 实现负载均衡。

四个服务提供者和一个服务消费者都需要在注册中心进行注册，同时可以使用配置重新来对配置文件进行统一集中管理。配置文件上传到Git进行远程管理。



2、工程搭建

1、父工程搭建

- 创建 Maven工程，pom.xml中添加一些各模块使用的相关依赖

```
<?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>org.example</groupId>
    <artifactId>food_book</artifactId>
    <version>1.0-SNAPSHOT</version>
    <!--Spring Boot 父包-->
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-
parent</artifactId>
        <version>2.3.4.RELEASE</version>
    </parent>
    <dependencies>
        <!-- web 组件相关依赖-->
        <dependency>

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

        <!-- Lombok -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>
    </dependencies>

    <!-- 添加对SpringCloud的依赖管理添加 -->
    <dependencyManagement>
        <dependencies>
            <dependency>

 <groupId>org.springframework.cloud</groupId>
```

```
                <artifactId>spring-cloud-
dependencies</artifactId>
                <version>Hoxton.SR8</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

</project>
```

2、注册中心子模块搭建

- 父工程新建子模块，pom.xml中导入相关依赖

```
<?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>food_book</artifactId>
        <groupId>org.example</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>eureka_server</artifactId>
    <!-- 注册中心所需相关依赖导入 -->
    <dependencies>
        <dependency>

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

```
</project>
```

- 创建配置文件 application.yml

```yaml
server:
  port: 8761
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
    register-with-eureka: false
    fetch-registry: false
```

- 创建启动类

```java
package com.gloryh;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootAp
plication;
import
org.springframework.cloud.netflix.eureka.server.Ena
bleEurekaServer;

/**
 * 注册中心启动类
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@SpringBootApplication
@EnableEurekaServer
public class EurekaServerApplication {
    public static void main(String[] args) {

 SpringApplication.run(EurekaServerApplication.clas
s,args);
    }
}
```

3.创建配置中心

- 创建配置中心 Config Server 子模块，并导入相关依赖

```
<?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>food_book</artifactId>
        <groupId>org.example</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>config_server</artifactId>
    <!--Config Server 相关依赖-->
    <dependencies>
        <dependency>

 <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-config-
server</artifactId>
        </dependency>
        <dependency>

 <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-
netflix-eureka-client</artifactId>
        </dependency>
    </dependencies>

</project>
```

- 创建配置文件 application.yml

```
server:
```

```
  port: 8762
spring:
  application:
    name: configserver
  profiles:
    active: native
  cloud:
    config:
      server:
        native:
          search-locations: classpath:/shared
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
```

我们就可以在 shared 文件夹下创建个歌微服务对应的配置文件进行统一管理

- 创建配置中心启动类

```
package com.gloryh;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootAp
plication;
import
org.springframework.cloud.config.server.EnableConfi
gServer;

/**
 * 配置中心 启动类
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@SpringBootApplication
@EnableConfigServer
public class ConfigServerApplication {
    public static void main(String[] args) {
```

```
 SpringApplication.run(ConfigServerApplication.clas
s,args);
    }
}
```

4、第一个服务提供者 —— order 服务搭建

- 创建子模块 order， pom.xml 导入相关依赖

```
<?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>food_book</artifactId>
        <groupId>org.example</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>order</artifactId>
    <!--添加相关依赖-->
    <dependencies>
        <!--注册为服务提供者-->
        <dependency>

 <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-
netflix-eureka-client</artifactId>
        </dependency>
        <!--从配置中心读取配置文件-->
        <dependency>

 <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-
config</artifactId>
```

```
        </dependency>
    </dependencies>
</project>
```

- 创建配置文件 bootstrap.yml 配置读取配置中心的对应配置文件

```yaml
spring:
  application:
    name: order
  profiles:
    active: dev
  cloud:
    config:
      uri: http://localhost:8762
      fail-fast: true
```

即从为8762端口的配置中心读取名为 order-dev的配置文件。

- 根据配置中心的配置，我们在配置中心的 resources/shared 文件夹下创建order-dev.yml

```yaml
server:
  port: 8010
```

- 创建配置文件

```
package com.gloryh;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootAp
plication;

/**
 * order 服务启动类
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@SpringBootApplication
public class OrderApplication {
    public static void main(String[] args) {
```

```
 SpringApplication.run(OrderApplication.class,args)
;
    }
}
```

- 创建前后端交互层

5、第二个服务提供者 —— menu 服务搭建

- 创建子模块，pom导入相关依赖

```
<?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>food_book</artifactId>
        <groupId>org.example</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>menu</artifactId>
    <dependencies>
        <!--Eureka Client 注册中心进行注册-->
        <dependency>

 <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-
netflix-eureka-client</artifactId>
        </dependency>
        <!-- Mybatis 负责和数据库进行交互 -->
        <dependency>

 <groupId>org.mybatis.spring.boot</groupId>
```

```
            <artifactId>mybatis-spring-boot-
starter</artifactId>
            <version>2.1.3</version>
        </dependency>
        <!-- MySQL 驱动  -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-
java</artifactId>
            <version>8.0.21</version>
        </dependency>
        <!--Config Client 从配置中心读取相关配置-->
        <dependency>

 <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-
config</artifactId>
        </dependency>
    </dependencies>

</project>
```

- 配置中心的 shared文件夹下创建 menu 的相关配置文件 menu-dev.yml

```
server:
  port: 8020
```

- 创建 配置读取文件 bootstrap.yml

```
spring:
  application:
    name: menu
  profiles:
    active: dev
  cloud:
    config:
      uri: http://localhost:8762
      fail-fast: true
```

- 创建启动类

```
package com.gloryh;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootAp
plication;

/**
 * menu 服务启动类
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@SpringBootApplication
public class MenuApplication {
    public static void main(String[] args) {

 SpringApplication.run(MenuApplication.class,args);
    }
}
```

- 创建相关实体类

```
package com.gloryh.entity;

import lombok.Data;

/**
 * 对应 menu 数据表的实体类
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@Data
public class Menu {
    private long id;
    private String name;
    private double price;
    private String flavor;
}
```

- 创建 数据库交互方法接口

```
package com.gloryh.repository;

import com.gloryh.entity.Menu;

import java.util.List;

/**
 * MyBatis 与数据库交互的调用接口类
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
public interface MenuRepository {
    /**
     * 查询菜单列表
     * @return 菜单列表List
     */
    public List<Menu> findAll();

    /**
     * 统计菜单内菜品数量
     * @return int
     */
    public int count();

    /**
     *  按id 查询菜品
     * @param id
     * @return 菜品
     */
    public Menu findById(long id);

    /**
     * 添加菜品信息
     * @param menu
     */
    public void save(Menu menu);

    /**
```

```
     * 修改菜品信息
     * @param menu
     */
    public void update(Menu menu);

    /**
     * 按 id 删除菜品信息
     * @param id
     */
    public void deleteById(long id);
}
```

- 创建 MyBatis 的相关mapper.xml接口 实现接口 (resources/mapping 文件夹下)

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.gloryh.repository.MenuRepository">
    <select id="findAll" resultType="Menu">
        SELECT * FROM t_menu LIMIT #{param1},#{param2}
    </select>
    <select id="count" resultType="int">
        SELECT COUNT(*) FROM t_menu
    </select>
    <select id="findById" resultType="Menu">
        SELECT * FROM t_menu WHERE id = #{id}
    </select>
    <insert id="save" parameterType="Menu">
        INSERT INTO t_menu(name,price,flavor,tid) VALUES (#{name},#{price},#{flavor},#{type.id})
    </insert>
    <update id="update" parameterType="com.gloryh.entity.Menu">
        UPDATE t_menu SET name = #{name},price=#{price},flavor=#{flavor},tid=#{type.id} WHERE id = #{id}
```

```
    </update>
    <delete id="deleteById" parameterType="long">
        DELETE FROM t_menu WHERE id =#{ids}
    </delete>

</mapper>
```

- 配置中心内对应配置文件添加Mapper文件读取

```
spring:
  datasource:
    url: jdbc:mysql://localhost:3305/food_book?
useUnicode=true&characterEncoding=UTF-
8&serverTimezone=UTC
    username: admin
    driver-class-name: com.mysql.cj.jdbc.Driver
    password: 123
mybatis:
  mapper-locations: classpath:/mapping/*.xml
  type-aliases-package: com.gloryh.entity
```

- 启动类中添加扫描注解-=`@MapperScan`

```
@MapperScan("com.gloryh.repository")
```

- 创建前后端交互层

```
package com.gloryh.controller;

import com.gloryh.entity.Menu;
import com.gloryh.repository.MenuRepository;
import
org.springframework.beans.factory.annotation.Autowi
red;
import
org.springframework.beans.factory.annotation.Value;
import
org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PathVariabl
e;
```

```
import
org.springframework.web.bind.annotation.RequestMapp
ing;
import
org.springframework.web.bind.annotation.RestControl
ler;

import java.util.List;

/**
 * menu controller
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@RestController
@RequestMapping("/menu")
public class MenuHandler {

    @Autowired
    private MenuRepository menuRepository;

    @GetMapping("/findAll/{index}/{limit}")
    public List<Menu>
findAll(@PathVariable("index") int
index,@PathVariable("limit") int limit){
        return menuRepository.findAll(index,limit);
    }
}
```

6、服务消费者搭建

- 创建服务消费者模块，pom导入相关依赖

```
<?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>food_book</artifactId>
        <groupId>org.example</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>client</artifactId>
    <!-- 相关依赖 -->
    <dependencies>
        <!-- Eureka Client -->
        <dependency>

 <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-
netflix-eureka-client</artifactId>
        </dependency>
        <!--Feign-->
        <dependency>

 <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-
openfeign</artifactId>
        </dependency>
        <!-- thymeleaf -->
        <dependency>

 <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-
thymeleaf</artifactId>
        </dependency>
        <!-- 配置中心 -->
        <dependency>

 <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-
config</artifactId>
        </dependency>
```

```
    </dependencies>

</project>
```

- 配置中心中shared的文件夹下创建对应配置文件

```
server:
  port: 8030
spring:
  thymeleaf:
    prefix: classpath:/static/
    suffix: .html
```

- 创建 bootstrap.yml 从配置中心读取对应的配置文件 client-dev

```
spring:
  application:
    name: client
  profiles:
    active: dev
  cloud:
    config:
      uri: http://localhost:8762
      fail-fast: true
```

- 创建启动类

```
package com.gloryh;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootAp
plication;

/**
 * 服务消费者启动类
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@SpringBootApplication
```

```
public class ClientApplication {
    public static void main(String[] args) {

 SpringApplication.run(ClientApplication.class,args
);
    }
}
```

7、服务消费者整合 Menu

- 创建对应的 Menu 实体类

```
package com.gloryh.entity;

import lombok.Data;

/**
 * 对应 menu 数据表的实体类
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@Data
public class Menu {
    private long id;
    private String name;
    private double price;
    private String flavor;
}
```

- 由于使用 Feign 声明式接口调用，首先创建 feign 调用接口类

```
package com.gloryh.feign;

import com.gloryh.entity.Menu;
import
org.springframework.cloud.openfeign.FeignClient;
import
org.springframework.web.bind.annotation.GetMapping;
```

```
import
org.springframework.web.bind.annotation.PathVariabl
e;

import java.util.List;

/**
 * Menu 的 Feign 声明式接口
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@FeignClient(value = "menu")
public interface MenuFeign {
    /**
     * 调用 menu 微服务的相关方法
     * @param index
     * @param limit
     * @return List<Menu>
     */
    @GetMapping("/menu/findAll/{index}/{limit}")
    public List<Menu> findAll(@PathVariable int
index,@PathVariable int limit);
}
```

- 启动类中加入Feign声明式接口调用注解——`@EnableFeignClients`
- 创建对应的前后端交互方法(由于 LayUI 的原因，数据不再使用Restful风格)

```
package com.gloryh.controller;

import com.gloryh.entity.Menu;
import com.gloryh.feign.MenuFeign;
import
org.springframework.beans.factory.annotation.Autowi
red;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

import java.util.List;
```

```
/**
 * 服务消费者 controller
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@Controller
@RequestMapping("/client")
public class ClientHandler {

    @Autowired
    private MenuFeign menuFeign;

    @GetMapping("/findAll")
    @ResponseBody
    public List<Menu> findAll(@RequestParam("page")
int page, @RequestParam("limit") int limit){
        int index=(page-1)*limit;
        return menuFeign.findAll(index,limit);
    }

    @GetMapping("/redirect/{location}")
    public String redirect(@PathVariable String
location ){
        return location;
    }
}
```

- 由于 LayUI 的原因，需要在menu 服务中对返回的数据进行二次封装。官网参考：https://www.layui.com/doc/

  LayUI 需要的类型：

```
{
  "code": 0,
  "msg": "",
  "count": 1000,
  "data": [{}, {}]
}
```

  menu 模块中创建VO进行转换：

```
package com.gloryh.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;

/**
 * 将返回的实体类类型进行二次封装，已达到前端和服务消费者所需要的格式
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@Data
@NoArgsConstructor
@AllArgsConstructor
public class MenuVO {
    private int code;
    private String msg;
    private int count;
    private List<Menu> data;
}
```

- 对 menu 的前后端交互方法进行修改

```
package com.gloryh.controller;

import com.gloryh.entity.Menu;
import com.gloryh.entity.MenuVO;
import com.gloryh.repository.MenuRepository;
import
org.springframework.beans.factory.annotation.Autowi
red;
import
org.springframework.beans.factory.annotation.Value;
import
org.springframework.web.bind.annotation.GetMapping;
```

```
import
org.springframework.web.bind.annotation.PathVariabl
e;
import
org.springframework.web.bind.annotation.RequestMapp
ing;
import
org.springframework.web.bind.annotation.RestControl
ler;

import java.util.List;

/**
 * menu controller
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@RestController
@RequestMapping("/menu")
public class MenuHandler {

    @Autowired
    private MenuRepository menuRepository;

    @GetMapping("/findAll/{index}/{limit}")
    public MenuVO findAll(@PathVariable("index")
int index, @PathVariable("limit") int limit) {
        // 获取所有列表
        List<Menu> menus =
menuRepository.findAll(index, limit);
        //获取总数
        int count = menuRepository.count();
        return new MenuVO(0, "", count, menus);
    }
}
```

- 服务消费者添加MenuVO

```
package com.gloryh.entity;
```

```
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;

/**
 * 将返回的实体类类型进行二次封装，已达到前端和服务消费者所需要的格式
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@Data
@NoArgsConstructor
@AllArgsConstructor
public class MenuVO {
    private int code;
    private String msg;
    private int count;
    private List<Menu> data;
}
```

- 服务消费者的Feign进行修改

```
package com.gloryh.feign;

import com.gloryh.entity.MenuVO;
import
org.springframework.cloud.openfeign.FeignClient;
import org.springframework.stereotype.Repository;
import
org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PathVariable;

import java.util.List;

/**
 * Menu 的 Feign 声明式接口
```

```
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@Repository
@FeignClient(value = "menu")
public interface MenuFeign {
    /**
     * 调用 menu 微服务的相关方法
     * @param index
     * @param limit
     * @return List<Menu>
     */
    @GetMapping("/menu/findAll/{index}/{limit}")
    public MenuVO findAll(@PathVariable int index,
@PathVariable int limit);
}
```

- 服务消费者前后端交互方法进行相应修改

```
package com.gloryh.controller;

import com.gloryh.entity.MenuVO;
import com.gloryh.feign.MenuFeign;
import
org.springframework.beans.factory.annotation.Autowi
red;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

/**
 * 服务消费者 controller
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@Controller
@RequestMapping("/client")
public class ClientHandler {

    @Autowired
```

```
    private MenuFeign menuFeign;

    @GetMapping("/findAll")
    @ResponseBody
    public MenuVO findAll(@RequestParam("page") int
page, @RequestParam("limit") int limit){
        int index=(page-1)*limit;
        return menuFeign.findAll(index,limit);
    }

    @GetMapping("/redirect/{location}")
    public String redirect(@PathVariable String
location ){
        return location;
    }
}
```

- 前端页面修改（index.html）

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet"
th:href="@{/layui/css/layui.css}" media="all">
</head>
<body>
<div class="layui-container" style="width:
700px;height: 600px;margin-top: 0px;padding-top:
60px;">

   <!-- <div style="margin-left: 460px; width:
200px;">
        欢迎回来！<a href="/account/redirect/order"
th:text="${session.user.nickname}"></a><a
href="/account/logout">   <button
class="layui-btn layui-btn-warm layui-btn-radius">退
出</button></a>
    </div>-->
```

```
    <table class="layui-hide" id="test" lay-
filter="test"></table>
    <script type="text/html" id="barDemo">
        <a class="layui-btn layui-btn-xs" lay-
event="order">订购</a>
    </script>
    <script th:src="@{/layui/layui.js}"
charset="utf-8"></script>
    <script>
        layui.use('table', function(){
            var table = layui.table;

            table.render({
                elem: '#test'
                ,url:'/client/findAll'
                ,title: '菜单列表'
                ,cols: [
                    [
                        {field:'id', width:100,
title: '编号', sort: true}
                        ,{field:'name', width:200,
title: '菜品'}
                        ,{field:'price', width:100,
title: '单价'}
                        ,{field:'flavor',
width:100, title: '口味'}
                        ,{field:'tid',width:100,
 title: '分类',templet:function(data){
                                /*return
data.type.name*/
                            return "暂无分类"
                            }
                        }
                        ,{fixed: 'right', title:'操
作', toolbar: '#barDemo', width:70}
                    ]
                ]
                ,page: true
            });
```

```
            //监听行工具事件
            table.on('tool(test)', function(obj){
                var data = obj.data;
                if(obj.event === 'order'){

 window.location.href="/order/save/"+data.id;
                }
            });
        });
    </script>

</div>
<script>
    //二级菜单联动
    layui.use('element', function(){
        var element = layui.element;

    });
</script>
</body>
</html>
```

- 运行 访问 http://localhost:8030/client/redirect/index

编号	菜品	单价	口味	分类	操作
1	香酥鸡	39	五香	暂无分类	订购
2	烧椒扣肉	46	微辣	暂无分类	订购
3	栗子三杯鸡	56	五香	暂无分类	订购
4	毛血旺	50	麻辣	暂无分类	订购
5	菠菜拌粉丝	22	五香	暂无分类	订购
6	凉拌豆腐皮	19	微辣	暂无分类	订购
7	酱牛肉	36	麻辣	暂无分类	订购
8	鱼头豆腐汤	32	五香	暂无分类	订购
9	瘦肉鸡蛋白菜汤	30	五香	暂无分类	订购
10	西葫芦虾仁蒸饺	26	五香	暂无分类	订购

1 2 到第 1 页 确定 共 12 条 10 条/页

- 使用多表关联查询获取分类，首先创建分类实体类Type

```
package com.gloryh.entity;
```

```
import lombok.Data;

/**
 * 菜品分类实体类
 *
 * @author 黄光辉
 * @since 2020/10/28
 **/
@Data
public class Type {
    private long id;
    private String name;
}
```

- 将 Type 加入Menu

```
package com.gloryh.entity;

import lombok.Data;

/**
 * 对应 menu 数据表的实体类
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@Data
public class Menu {
    private long id;
    private String name;
    private double price;
    private String flavor;
    private Type type;
}
```

- 需要的语句：

```
SELECT m.id,m.name ,m.price,m.flavor,t.name AS
tname FROM t_menu m,t_type t WHERE  m.tid = t.id
```

- 对mapper文件对应的方法进行修改

```
<resultMap id="menuMap"
type="com.gloryh.entity.Menu">
    <id column="id" property="id"></id>
    <result column="name" property="name"></result>
    <result column="price" property="price">
</result>
    <result column="flavor" property="flavor">
</result>
    <collection property="type"
ofType="com.gloryh.entity.Type">
        <id column="tid" property="id"></id>
        <result column="tname" property="name">
</result>
    </collection>
</resultMap>
<select id="findAll" resultMap="menuMap">
    SELECT m.id,m.name
,m.price,m.flavor,m.tid,t.name AS tname FROM t_menu
m,t_type t WHERE  m.tid = t.id LIMIT #{param1},#
{param2}
</select>
```

- 修改Client Server的实体类为对应实体类

  加入实体类Type

```
package com.gloryh.entity;

import lombok.Data;

/**
 * 菜品分类实体类
 *
 * @author 黄光辉
 * @since 2020/10/28
 **/
@Data
public class Type {
    private long id;
    private String name;
```

```
}
```

为Menu添加成员变量属性 type

```
package com.gloryh.entity;

import lombok.Data;

/**
 * 对应 menu 数据表的实体类
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@Data
public class Menu {
    private long id;
    private String name;
    private double price;
    private String flavor;
    private Type type;
}
```

- 修改前端代码

```
table.render({
    elem: '#test'
    ,url:'/client/findAll'
    ,title: '菜单列表'
    ,cols: [
        [
            {field:'id', width:100, title: '编号',
sort: true}
            ,{field:'name', width:200, title: '菜
品'}
            ,{field:'price', width:100, title: '单
价'}
            ,{field:'flavor', width:100, title: '口
味'}
            ,{field:'tid',width:100,  title: '分
类',templet:function(data){
```

```
                    return data.type.name
                }
            }
            ,{fixed: 'right', title:'操作', toolbar:
'#barDemo', width:70}
        ]
    ]
    ,page: true
});
```

- 测试运行：

编号	菜品	单价	口味	分类	操作
1	香酥鸡	39	五香	热菜	订购
2	烧椒扣肉	46	微辣	热菜	订购
3	栗子三杯鸡	56	五香	热菜	订购
4	毛血旺	50	麻辣	热菜	订购
5	菠菜拌粉丝	22	五香	凉菜	订购
6	凉拌豆腐皮	19	微辣	凉菜	订购
7	酱牛肉	36	麻辣	凉菜	订购
8	鱼头豆腐汤	32	五香	汤羹	订购
9	瘦肉鸡蛋白菜汤	30	五香	汤羹	订购
10	西葫芦虾仁蒸饺	26	五香	主食	订购

1 2 到第 1 页 确定 共 12 条 10 条/页

- 完善CRUD功能，修改前端代码，添加修改和删除按钮，并处理响应的点击事件

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet"
th:href="@{/layui/css/layui.css}" media="all">
</head>
<body>
```

```
<div class="layui-container" style="width: 700px;height: 600px;margin-top: 0px;padding-top: 60px;">

   <!-- <div style="margin-left: 460px; width: 200px;">
        欢迎回来！<a href="/account/redirect/order" th:text="${session.user.nickname}"></a><a href="/account/logout">   <button class="layui-btn layui-btn-warm layui-btn-radius">退出</button></a>
    </div>-->

    <table class="layui-hide" id="test" lay-filter="test"></table>
    <script type="text/html" id="barDemo">
        <a class="layui-btn layui-btn-xs" lay-event="update">修改</a>
        <a class="layui-btn layui-btn-danger layui-btn-xs" lay-event="del">删除</a>
    </script>
    <script th:src="@{/layui/layui.js}" charset="utf-8"></script>
    <script>
        layui.use('table', function(){
            var table = layui.table;

            table.render({
                elem: '#test'
                ,url:'/client/findAll'
                ,title: '菜单列表'
                ,cols: [
                    [
                        {field:'id', width:100, title: '编号', sort: true}
                        ,{field:'name', width:150, title: '菜品'}
                        ,{field:'price', width:80, title: '单价'}
                        ,{field:'flavor', width:100, title: '口味'}
```

```
                        ,{field:'tid',width:100,
 title: '分类',templet:function(data){
                                return
data.type.name
                            }
                        }
                        ,{fixed: 'right', title:'操
作', toolbar: '#barDemo', width:140}
                    ]
                ]
                ,page: true
            });

            //监听行工具事件
            table.on('tool(test)', function(obj){
                var data = obj.data;
                if(obj.event === 'update'){

 window.location.href="/client/findById/"+data.id;
                }
                if(obj.event === 'del'){
                   layer.confirm('确定要删除
吗？',function (index){

window.location.href="/client/deleteById/"+data.id;
                       layer.close(index);
                   });
                }
            });
        });
    </script>

</div>
<script>
    //二级菜单联动
    layui.use('element', function(){
        var element = layui.element;

    });
</script>
</body>
```

```
</html>
```

- 处理删除操作

  menu添加对应的方法处理`window.location.href="/client/deleteById/"+data.id;`。调用deleteById接口方法。

```
@DeleteMapping("/deleteById/{id}")
public void deleteById(@PathVariable("id") long id)
{
    menuRepository.deleteById(id);
}
```

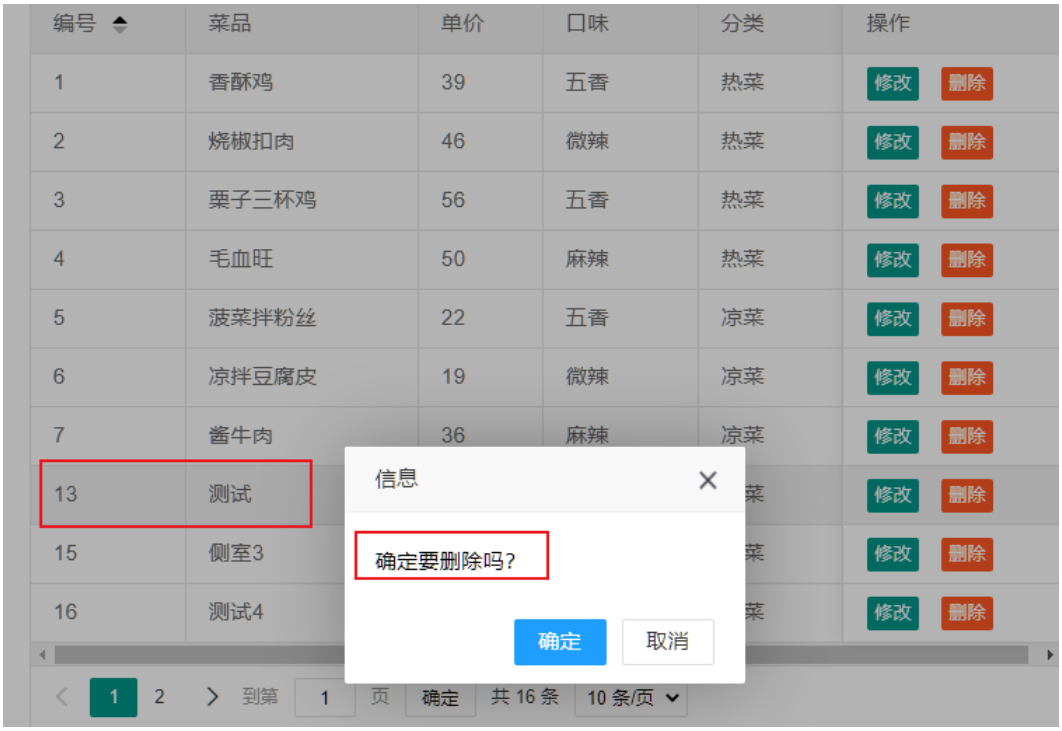
  客户端MenuFeign接口调用 menu 提供的方法

```
@DeleteMapping("/menu/deleteById/{id}")
public void deleteById(@PathVariable long id);
```

  客户端前后端交互层调用 MenuFeign 接口，删除后刷新页面

```
@GetMapping("/deleteById/{id}")
public String deleteById(@PathVariable long id){
    menuFeign.deleteById(id);
    return "redirect:/client/redirect/index";
}
```

- 运行测试



成功

- 菜品分类数据返回，获取菜品的分类信息以供菜单的修改和删除操作使用

  首先，创建菜品分类的相关调用接口

```
package com.gloryh.repository;

import com.gloryh.entity.Type;
import org.springframework.stereotype.Repository;
```

```java
import java.util.List;

/**
 * 菜品分类相关操作接口类
 *
 * @author 黄光辉
 * @since 2020/10/28
 **/
@Repository
public interface TypeRepository {
    /**
     * 查询所有菜品分类信息
     * @return List
     */
    public List<Type> findAll();
}
```

相关 mapper 文件

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.gloryh.repository.TypeRepository">
    <select id="findAll" resultType="com.gloryh.entity.Type">
       SELECT * FROM t_type
    </select>
</mapper>
```

MenuHandler 调用相关接口

```java
@Autowired
private TypeRepository typeRepository;

@GetMapping("/findTypes")
public List<Type> findTypes(){
    return typeRepository.findAll();
}
```

客户端 MenuFeign 调用其方法

```
@GetMapping("/menu/findTypes")
public List<Type> findTypes();
```

客户端前后端交互方法调用 MenuFeign 对应接口

```
@GetMapping("/findTypes")
public ModelAndView findTypes(){
    ModelAndView modelAndView =new ModelAndView();
    modelAndView.setViewName("menu_add");

 modelAndView.addObject("list",menuFeign.findTypes(
));
    return modelAndView;
}
```

- 接下来完成菜品添加功能

  添加前端页面menu_add:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet"
th:href="@{/layui/css/layui.css}" media="all">
</head>
<body>
<div class="layui-container" style="margin-top:
50px;">

    <div class="layui-container" style="width:
500px;height: 330px;padding-top: 60px;">
        <form class="layui-form"
action="/menu/save" method="post">
            <div class="layui-form-item">
                <label class="layui-form-label">菜
品: </label>
                <div class="layui-inline">
```

```
                    <input type="text" name="name"
lay-verify="required" autocomplete="off"
placeholder="请输入用户名" class="layui-input">
                </div>
            </div>
            <div class="layui-form-item">
                <label class="layui-form-label">单
价：</label>
                <div class="layui-inline">
                    <input type="text" name="price"
lay-verify="required" placeholder="请输入密码"
autocomplete="off" class="layui-input">
                </div>
            </div>
            <div class="layui-form-item">
                <label class="layui-form-label">口
味：</label>
                <div class="layui-inline">
                    <input type="text"
name="flavor" lay-verify="required" placeholder="请
输入年龄" autocomplete="off" class="layui-input">
                </div>
            </div>
            <div class="layui-form-item">
                <label class="layui-form-label">分
类：</label>
                <div class="layui-input-inline">
                    <select name="type.id">
                        <option
th:each="type:${list}" th:text="${type.name}"
th:value="${type.id}"></option>
                    </select>
                </div>
            </div>
            <div class="layui-form-item">
                <button class="layui-btn" lay-
submit="" lay-filter="demo2" style="margin-left:
160px;">提交</button>
            </div>
        </form>
    </div>
```

```
</div>
<script th:src="@{/layui/layui.js}" charset="utf-8"></script>
<script>
    layui.use(['form', 'element'], function(){
        var form = layui.form;
        var element = layui.element;

        //自定义验证规则
        form.verify({
            price: [/(^[1-9]([0-9]+)?(\.[0-9]{1,2})?$)|(^(0){1}$)|(^[0-9]\.[0-9]([0-9])?$)/, "请输入正确的价格"]
        });

    });
</script>
</body>
</html>
```

menu 前后端交互层完善处理客户端的请求的方法：

```
package com.gloryh.controller;

import com.gloryh.entity.Menu;
import com.gloryh.entity.MenuVO;
import com.gloryh.entity.Type;
import com.gloryh.repository.MenuRepository;
import com.gloryh.repository.TypeRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.*;

import java.util.List;

/**
 * menu controller
 *
```

```
 * @author 黄光辉
 * @since 2020/10/27
 **/
@RestController
@RequestMapping("/menu")
public class MenuHandler {

    @Autowired
    private MenuRepository menuRepository;

    @GetMapping("/findAll/{index}/{limit}")
    public MenuVO findAll(@PathVariable("index")
int index, @PathVariable("limit") int limit) {
        // 获取所有列表
        List<Menu> menus =
menuRepository.findAll(index, limit);
        //获取总数
        int count = menuRepository.count();
        return new MenuVO(0, "", count, menus);
    }

    @DeleteMapping("/deleteById/{id}")
    public void deleteById(@PathVariable("id") long
id) {
        menuRepository.deleteById(id);
    }

    @Autowired
    private TypeRepository typeRepository;

    @GetMapping("/findTypes")
    public List<Type> findTypes() {
        return typeRepository.findAll();
    }

    @PostMapping("/save")
    public void save(@RequestBody Menu menu){
        menuRepository.save(menu);
    }

    @GetMapping("/findById/{id}")
```

```
    public Menu findById(@PathVariable("id") long
id){
        return menuRepository.findById(id);
    }
    @PutMapping("/update")
    public void update(@RequestBody Menu menu){
        menuRepository.update(menu);
    }

}
```

客户端完善 MenuFeign 接口 对menu的接口的调用

```
package com.gloryh.feign;

import com.gloryh.entity.Menu;
import com.gloryh.entity.MenuVO;
import com.gloryh.entity.Type;
import
org.springframework.cloud.openfeign.FeignClient;
import org.springframework.stereotype.Repository;
import org.springframework.web.bind.annotation.*;

import java.util.List;

/**
 * Menu 的 Feign 声明式接口
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@Repository
@FeignClient(value = "menu")
public interface MenuFeign {
    /**
     * 调用 menu 微服务的相关方法
     *
     * @param index
     * @param limit
     * @return List<Menu>
     */
```

```
    @GetMapping("/menu/findAll/{index}/{limit}")
    public MenuVO findAll(@PathVariable int index,
@PathVariable int limit);

    /**
     * 按 id 删除菜品信息
     *
     * @param id
     */
    @DeleteMapping("/menu/deleteById/{id}")
    public void deleteById(@PathVariable long id);

    /**
     * 获取菜品分类列表
     *
     * @return List<Type>
     */
    @GetMapping("/menu/findTypes")
    public List<Type> findTypes();

    /**
     * 添加菜品信息
     * @param menu
     */
    @PostMapping("/menu/save")
    public void save(Menu menu);

    /**
     * 修改菜品信息
     * @param menu
     */
    @PutMapping("/menu/update")
    public void update(Menu menu);

    /**
     * 按 id 查询 菜品信息
     * @param id
     * @return Menu
     */
    @GetMapping("/menu/findById/{id}")
    public Menu findById(@PathVariable long id);
```

```
}
```

客户端 前后端交互层 处理前端页面传过来的菜品信息并添加到数据库：

```
@PostMapping("/save")
public String save(Menu menu) {
    System.out.println(menu);
    menuFeign.save(menu);
    return "redirect:/menu/redirect/index";
}
```

- 测试

- 实现菜品编辑操作

  前端使用发送 findById 请求

```
if(obj.event === 'update'){
    window.location.href="/menu/findById/"+data.id;
}
```

  client 后端调用 MenuFeign 的 findById 方法查询到对应的菜品信息和分类信息，然后跳转至menu_update 页面

```
@GetMapping("/findById/{id}")
public ModelAndView edit(@PathVariable long id){
    ModelAndView modelAndView =new ModelAndView();
    Menu menu =menuFeign.findById(id);
    modelAndView.addObject("menu",menu);
    modelAndView.addObject("list",
menuFeign.findTypes());
    modelAndView.setViewName("menu_update");
    return modelAndView;
}
```

  menu_update 页面源码：

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet"
th:href="@{/layui/css/layui.css}" media="all">
</head>
<body>
<div class="layui-container" style="margin-top:
50px;">

    <div class="layui-container" style="width:
500px;height: 330px;padding-top: 60px;">
        <form class="layui-form"
action="/menu/update" method="post">
            <div class="layui-form-item">
                <label class="layui-form-label">编
号：</label>
                <div class="layui-inline">
                    <input type="text" name="id"
th:value="${menu.id}" lay-verify="required"
readonly autocomplete="off" placeholder="请输入用户
名" class="layui-input">
                </div>
            </div>
            <div class="layui-form-item">
                <label class="layui-form-label">菜
品：</label>
                <div class="layui-inline">
                    <input type="text" name="name"
th:value="${menu.name}" lay-verify="required"
autocomplete="off" placeholder="请输入用户名"
class="layui-input">
                </div>
            </div>
            <div class="layui-form-item">
                <label class="layui-form-label">单
价：</label>
                <div class="layui-inline">
```

```
                    <input type="text" name="price"
th:value="${menu.price}" lay-verify="required"
placeholder="请输入密码" autocomplete="off"
class="layui-input">
                </div>
            </div>
            <div class="layui-form-item">
                <label class="layui-form-label">口
味：</label>
                <div class="layui-inline">
                    <input type="text"
name="flavor" th:value="${menu.flavor}" lay-
verify="required" placeholder="请输入年龄"
autocomplete="off" class="layui-input">
                </div>
            </div>
            <div class="layui-form-item">
                <label class="layui-form-label">分
类：</label>
                <div class="layui-input-inline">
                    <select name="type.id">
                        <option
th:each="type:${list}" th:text="${type.name}"
th:value="${type.id}" th:if="${menu.type.id ==
type.id}" selected></option>
                        <option
th:each="type:${list}" th:text="${type.name}"
th:value="${type.id}" th:if="${menu.type.id !=
type.id}" ></option>
                    </select>
                </div>
            </div>
            <div class="layui-form-item">
                <button class="layui-btn" lay-
submit="" lay-filter="demo2" style="margin-left:
160px;">提交</button>
            </div>
        </form>
    </div>
</div>
```

```
<script th:src="@{/layui/layui.js}" charset="utf-8"></script>
<script>
    layui.use(['form', 'element'], function(){
        var form = layui.form;
        var element = layui.element;

        //自定义验证规则
        form.verify({
            price: [/(^[1-9]([0-9]+)?(\.[0-9]{1,2})?$)|(^(0){1}$)|(^[0-9]\.[0-9]([0-9])?$)/, "请输入正确的价格"]
        });

    });
</script>
</body>
</html>
```

修改完成后点击提交，完成修改，并返回菜单页面：

```
@PostMapping("/update")
public String update(Menu menu){
    menuFeign.update(menu);
    return "redirect:/menu/redirect/index";
}
```

测试：

编号	菜品	单价	口味	分类	操作
1	香酥鸡	40	五香	热菜	修改 删除
2	烧椒扣肉	46	微辣	热菜	修改 删除
3	栗子三杯鸡	56	五香	热菜	修改 删除
4	毛血旺	50	麻辣	热菜	修改 删除
16	测试4	4	等等	热菜	修改 删除
17	haha	12	五香	热菜	修改 删除
19	麻婆豆腐	12	麻辣	热菜	修改 删除
21	鱼香肉丝	24	五香	热菜	修改 删除
5	菠菜拌粉丝	22	五香	凉菜	修改 删除
6	凉拌豆腐皮	19	微辣	凉菜	修改 删除

编号	菜品	单价	口味	分类	操作
1	香酥鸡	50	五香	热菜	修改 删除
2	烧椒扣肉	46	微辣	热菜	修改 删除
3	栗子三杯鸡	56	五香	热菜	修改 删除
4	毛血旺	50	麻辣	热菜	修改 删除
16	测试4	4	等等	热菜	修改 删除
17	haha	12	五香	热菜	修改 删除
19	麻婆豆腐	12	麻辣	热菜	修改 删除
21	鱼香肉丝	24	五香	热菜	修改 删除
5	菠菜拌粉丝	22	五香	凉菜	修改 删除
6	凉拌豆腐皮	19	微辣	凉菜	修改 删除

8、第三个服务提供者——user 服务搭建

- 新建模块，pom导入相关依赖

```
<?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>food_book</artifactId>
        <groupId>org.example</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>user</artifactId>
    <dependencies>
        <!--Eureka Client 注册中心进行注册-->
        <dependency>

 <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-
netflix-eureka-client</artifactId>
        </dependency>
        <!-- Mybatis 负责和数据库进行交互 -->
        <dependency>

 <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-
starter</artifactId>
            <version>2.1.3</version>
        </dependency>
        <!-- MySQL 驱动  -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-
java</artifactId>
```

```
            <version>8.0.21</version>
        </dependency>
        <!--Config Client 从配置中心读取相关配置-->
        <dependency>

 <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-
config</artifactId>
        </dependency>
    </dependencies>


</project>
```

- 配置中心的 shared文件夹下创建 user 的相关配置文件 user-dev.yml

```
server:
  port: 8040
spring:
  datasource:
    url: jdbc:mysql://localhost:3305/food_book?
useUnicode=true&characterEncoding=UTF-
8&serverTimezone=UTC
    username: admin
    driver-class-name: com.mysql.cj.jdbc.Driver
    password: 123
mybatis:
  mapper-locations: classpath:/mapping/*.xml
  type-aliases-package: com.gloryh.entity
```

- 创建 配置读取文件 bootstrap.yml

```yaml
spring:
  application:
    name: user
  profiles:
    active: dev
  cloud:
    config:
      uri: http://localhost:8762
      fail-fast: true
```

- 创建启动类

```
package com.gloryh;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootAp
plication;

/**
 * user 服务启动类
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@SpringBootApplication
public class UserApplication {
    public static void main(String[] args) {

 SpringApplication.run(UserApplication.class,args);
    }
}
```

- 创建相关实体类

```
package com.gloryh.entity;

import lombok.Data;

import java.util.Date;
```

```java
/**
 * user 用户实体类
 *
 * @author 黄光辉
 * @since 2020/10/30
 **/
@Data
public class User {
    private long id;
    private String username;
    private String password;
    private String nickname;
    private String gender;
    private String telephone;
    private Date registerdate;
    private String address;
}
```

- 创建 数据库交互方法接口

```java
package com.gloryh.repository;

import com.gloryh.entity.User;

import java.util.List;

/**
 *  User MyBatis 与数据库交互的调用接口类
 *
 * @author 黄光辉
 * @since 2020/10/30
 **/
public interface UserRepository {
    /**
     * 查询所有用户信息
     * @return List<User>
     */
    public List<User> findAll();

    /**
     * 按 id 查询用户信息
```

```
     * @param id
     * @return User
     */
    public User findById(long id);

    /**
     * 查询用户总数
     * @return int
     */
    public int count();

    /**
     * 添加新用户
     * @param user
     */
    public void save(User user);

    /**
     * 修改用户信息
     * @param user
     */
    public void update(User user);

    /**
     * 按 id 删除用户信息
     * @param id
     */
    public void deleteById(long id);
}
```

- 创建 MyBatis 的相关mapper.xml接口 实现接口 (resources/mapping 文件夹下)

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper
3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-
mapper.dtd">
<mapper
namespace="com.gloryh.repository.UserRepository">
```

```
    <select id="findAll"
resultType="com.gloryh.entity.User">
        SELECT * FROM t_user limit #{param1},#
{param2}
    </select>
    <select id="count" resultType="int">
        SELECT COUNT(*) FROM t_user
    </select>
    <select id="findById"
resultType="com.gloryh.entity.User">
        SELECT * FROM t_user WHERE id = #{id}
    </select>
    <insert id="save"
parameterType="com.gloryh.entity.User">
        INSERT INTO
t_user(username,password,nickname,gender,telephone,
registerdate,address) VALUES (#{username},#
{password},#{nickname},#{gender},#{telephone},#
{registerdate},#{address})
    </insert>
    <update id="update"
parameterType="com.gloryh.entity.User">
        UPDATE t_menu SET username = #
{username},password=#{password},nickname=#
{nickname},gender=#{gender},telephone=#
{telephone},registerdate=#{registerdate},address=#
{address} WHERE id = #{id}
    </update>
    <delete id="deleteById" parameterType="long">
        DELETE FROM t_user WHERE id =#{id}
    </delete>
</mapper>
```

- 配置中心内对应配置文件添加Mapper文件读取

```
spring:
  datasource:
    url: jdbc:mysql://localhost:3305/food_book?
useUnicode=true&characterEncoding=UTF-
8&serverTimezone=UTC
    username: admin
    driver-class-name: com.mysql.cj.jdbc.Driver
    password: 123
mybatis:
  mapper-locations: classpath:/mapping/*.xml
  type-aliases-package: com.gloryh.entity
```

- 启动类中添加扫描注解-=`@MapperScan`

```
@MapperScan("com.gloryh.repository")
```

- 创建UserVO

```
package com.gloryh.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;

/**
 * 将返回的实体类类型进行二次封装，已达到前端和服务消费者所需
要的格式
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@Data
@NoArgsConstructor
@AllArgsConstructor
public class UserVO {
    private int code;
    private String msg;
    private int count;
    private List<User> data;
```

```
}
```

- 创建前后端交互层，调用相关接口

```
package com.gloryh.controller;

import com.gloryh.entity.User;
import com.gloryh.entity.UserVO;
import com.gloryh.repository.UserRepository;
import
org.springframework.beans.factory.annotation.Autowi
red;
import org.springframework.web.bind.annotation.*;

import java.util.List;

/**
 * user controller
 *
 * @author 黄光辉
 * @since 2020/10/30
 **/
@RestController
@RequestMapping("/user")
public class UserHandler {
    @Autowired
    private UserRepository userRepository;

    @GetMapping("/findAll/{index}/{limit}")
    public UserVO findAll(@PathVariable("index")
int index, @PathVariable("limit") int limit) {
        // 获取所有列表
        List<User> users =
userRepository.findAll(index, limit);
        //获取总数
        int count = userRepository.count();
        return new UserVO(0, "", count, users);
    }

    @DeleteMapping("/deleteById/{id}")
```

```
    public void deleteById(@PathVariable("id") long
id) {
        userRepository.deleteById(id);
    }

    @PostMapping("/save")
    public void save(@RequestBody User user){
        userRepository.save(user);
    }

    @GetMapping("/findById/{id}")
    public User findById(@PathVariable("id") long
id){
        return userRepository.findById(id);
    }
    @PutMapping("/update")
    public void update(@RequestBody User user){
        userRepository.update(user);
    }
}
```

9、服务消费者整合 User

- 创建对应的 User 和 UserVO 实体类

```
package com.gloryh.entity;

import lombok.Data;

import java.util.Date;

/**
 * user 用户实体类
 *
 * @author 黄光辉
 * @since 2020/10/30
 **/
@Data
public class User {
    private long id;
    private String username;
```

```
    private String password;
    private String nickname;
    private String gender;
    private String telephone;
    private Date registerdate;
    private String address;
}
```

```
package com.gloryh.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;

/**
 * 将返回的实体类类型进行二次封装，已达到前端和服务消费者所需要的格式
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@Data
@NoArgsConstructor
@AllArgsConstructor
public class UserVO {
    private int code;
    private String msg;
    private int count;
    private List<User> data;
}
```

- 创建 UserFeign 接口，调用user 服务提供者相关的接口

```
package com.gloryh.feign;

import com.gloryh.entity.User;
import com.gloryh.entity.UserVO;
import
org.springframework.cloud.openfeign.FeignClient;
```

```java
import org.springframework.stereotype.Repository;
import org.springframework.web.bind.annotation.*;

/**
 * User 的 Feign 声明式接口
 *
 * @author 黄光辉
 * @since 2020/10/30
 **/
@Repository
@FeignClient(value = "user")
public interface UserFeign {
    /**
     * 调用 user 微服务的相关方法
     *
     * @param index
     * @param limit
     * @return List<User>
     */
    @GetMapping("/user/findAll/{index}/{limit}")
    public UserVO findAll(@PathVariable int index,
@PathVariable int limit);

    /**
     * 按 id 删除用户信息
     *
     * @param id
     */
    @DeleteMapping("/user/deleteById/{id}")
    public void deleteById(@PathVariable long id);


    /**
     * 添加用户信息
     * @param user
     */
    @PostMapping("/user/save")
    public void save(User user);

    /**
     * 修改用户信息
```

```
     * @param user
     */
    @PutMapping("/user/update")
    public void update(User user);

    /**
     * 按 id 查询 用户信息
     * @param id
     * @return User
     */
    @GetMapping("/user/findById/{id}")
    public User findById(@PathVariable long id);
}
```

- 启动类中加入Feign声明式接口调用注解——`@EnableFeignClients`
- 创建对应的前后端交互方法(由于 LayUI 的原因，数据不再使用Restful风格)

```
package com.gloryh.controller;

import com.gloryh.entity.User;
import com.gloryh.entity.UserVO;
import com.gloryh.feign.UserFeign;
import
org.springframework.beans.factory.annotation.Autowi
red;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

/**
 * 服务消费者 user controller
 *
 * @author 黄光辉
 * @since 2020/10/30
 **/
@Controller
@RequestMapping("/user")
public class UserHandler {
    @Autowired
    private UserFeign userFeign;
```

```
    @GetMapping("/redirect/{location}")
    public String redirect(@PathVariable String
location) {
        return location;
    }

}
```

- 实现 查询操作

  前端页面源码user_manage:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet"
th:href="@{/layui/css/layui.css}" media="all">
</head>
<body class="layui-layout-body">
<div class="layui-container" style="margin-top:
50px;width: 1000px;">

    <table class="layui-hide" id="test"
style="width: 300px" lay-filter="test"></table>
    <script type="text/html" id="barDemo">
        <a class="layui-btn layui-btn-danger layui-
btn-xs" lay-event="del">删除</a>
    </script>
    <script th:src="@{/layui/layui.js}"
charset="utf-8"></script>
    <script>
        layui.use('table', function(){
            var table = layui.table;

            table.render({
                elem: '#test'
                ,url:'/user/findAll'
                ,title: '用户列表'
```

```
                ,cols: [
                    [
                        {field:'id', width:70,
title: '编号', sort: true}
                        ,{field:'username',
width:100, title: '用户名'}
                        ,{field:'nickname',
width:100, title: '昵称'}
                        ,{field:'gender', width:70,
title: '性别'}
                        ,{field:'telephone',
width:160, title: '联系电话'}
                        ,{field:'registerdate',
width:220, title: '注册日期'}
                        ,{field:'address',
width:200, title: '地址'}
                        ,{fixed: 'right', title:'操
作', toolbar: '#barDemo', width:80}
                    ]
                    ]
                    ,page: true
            });

            //监听行工具事件
            table.on('tool(test)', function(obj){
                var data = obj.data;
                if(obj.event === 'del'){
                    layer.confirm('确定要删除吗？',
function(index){

 window.location.href="/user/deleteById/"+data.id;
                        layer.close(index);
                    });
                }
            });
        });
    </script>

</div>
<script>
    //二级菜单联动
```

```
    layui.use('element', function(){
        var element = layui.element;

    });
</script>
</body>
</html>
```

客户端 user前后端交互层添加findAll方法调用UserFeign对应接口

```
@GetMapping("/findAll")
@ResponseBody
public UserVO findAll(@RequestParam("page") int
page,@RequestParam("limit")int limit){
    int index=limit*(page-1);
    return userFeign.findAll(index,limit);
}
```

测试，访问http://localhost:8030/user/redirect/user_manage

- 实现用户添加操作

```
@PostMapping("/save")
public String save(User user) {
    user.setRegisterdate(new Date());
    userFeign.save(user);
    return "redirect:/user/redirect/user_manage";
}
```

- 实现用户删除操作

```
@GetMapping("/deleteById/{id}")
public String deleteById(@PathVariable long id){
    userFeign.deleteById(id);
    return "redirect:/user/redirect/user_manage";
}
```

10、第四个服务提供者——account 服务搭建

- 新建模块，pom导入相关依赖

```
<?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>food_book</artifactId>
        <groupId>org.example</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>account</artifactId>
    <dependencies>
        <!--Eureka Client 注册中心进行注册-->
        <dependency>

 <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-
netflix-eureka-client</artifactId>
        </dependency>
        <!-- Mybatis 负责和数据库进行交互 -->
        <dependency>

 <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-
starter</artifactId>
            <version>2.1.3</version>
        </dependency>
        <!-- MySQL 驱动  -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-
java</artifactId>
```

```
            <version>8.0.21</version>
        </dependency>
        <!--Config Client 从配置中心读取相关配置-->
        <dependency>

 <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-
config</artifactId>
        </dependency>
    </dependencies>

</project>
```

- 配置中心的 shared文件夹下创建 account 的相关配置文件 account-dev.yml

```
server:
  port: 8050
spring:
  datasource:
    url: jdbc:mysql://localhost:3305/food_book?
useUnicode=true&characterEncoding=UTF-
8&serverTimezone=UTC
    username: admin
    driver-class-name: com.mysql.cj.jdbc.Driver
    password: 123
mybatis:
  mapper-locations: classpath:/mapping/*.xml
  type-aliases-package: com.gloryh.entity
```

- 创建 配置读取文件 bootstrap.yml

```
spring:
  application:
    name: account
  profiles:
    active: dev
  cloud:
    config:
      uri: http://localhost:8762
      fail-fast: true
```

- 创建启动类

```
package com.gloryh;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootAp
plication;

/**
 * account 服务启动类
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@SpringBootApplication
public class AccountApplication {
    public static void main(String[] args) {

 SpringApplication.run(AccountApplication.class,arg
s);
    }
}
```

- 创建相关实体类,因为涉及到User和Admin两张表，所以需要使用多态实现

  User实体类

```
package com.gloryh.entity;

import lombok.Data;

import java.util.Date;

/**
 * user 用户实体类
 *
 * @author 黄光辉
 * @since 2020/10/30
 **/
@Data
```

```
public class User {
    private long id;
    private String username;
    private String password;
    private String nickname;
    private String gender;
    private String telephone;
    private Date registerdate;
    private String address;
}
```

Admin实体类

```
package com.gloryh.entity;

import lombok.Data;

/**
 * 管理员实体类
 *
 * @author 黄光辉
 * @since 2020/10/31
 **/
@Data
public class Admin {
    private long id;
    private String username;
    private String password;
}
```

- 创建 数据库交互方法接口

  Admin实体类相关接口类

```
package com.gloryh.repository;

import com.gloryh.entity.Admin;

/**
 * Admin MyBatis 与数据库交互的调用接口类
 *
 * @author 黄光辉
```

```
 * @since 2020/10/31
 **/
public interface AdminRepository {
    /**
     * 查询数据库中对应的Admin信息
     * @param username 用户名
     * @param password 密码
     * @return Admin
     */
    public Admin login(String username,String
password);
}
```

User实体类相关接口类

```
package com.gloryh.repository;

import com.gloryh.entity.User;

/**
 * User MyBatis 与数据库交互的调用接口类
 *
 * @author 黄光辉
 * @since 2020/10/31
 **/
public interface UserRepository {
    /**
     * 查询数据库中对应的User信息
     * @param username 用户名
     * @param password 密码
     * @return User
     */
    public User login(String username, String
password);
}
```

- 创建 MyBatis 的相关mapper.xml接口 实现接口 (resources/mapping 文件夹下)

  UserRepository 的对应 mapper

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper
3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-
mapper.dtd">
<mapper
namespace="com.gloryh.repository.UserRepository">
    <select id="login"
resultType="com.gloryh.entity.User">
        SELECT * FROM t_user WHERE username = #
{param1} AND password = #{param2}
    </select>
</mapper>
```

AdminRepository 的对应 mapper

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper
3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-
mapper.dtd">
<mapper
namespace="com.gloryh.repository.AdminRepository">
    <select id="login"
resultType="com.gloryh.entity.Admin">
        SELECT * FROM t_admin WHERE username = #
{param1} AND password = #{param2}
    </select>
</mapper>
```

- 配置中心内对应配置文件添加Mapper文件读取

```
spring:
  datasource:
    url: jdbc:mysql://localhost:3305/food_book?
useUnicode=true&characterEncoding=UTF-
8&serverTimezone=UTC
    username: admin
    driver-class-name: com.mysql.cj.jdbc.Driver
    password: 123
mybatis:
  mapper-locations: classpath:/mapping/*.xml
  type-aliases-package: com.gloryh.entity
```

- 启动类中添加扫描注解-=`@MapperScan`

```
@MapperScan("com.gloryh.repository")
```

- 创建前后端交互层，调用相关接口

```
package com.gloryh.controller;

import com.gloryh.repository.AdminRepository;
import com.gloryh.repository.UserRepository;
import
org.springframework.beans.factory.annotation.Autowi
red;
import
org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PathVariabl
e;
import
org.springframework.web.bind.annotation.RequestMapp
ing;
import
org.springframework.web.bind.annotation.RestControl
ler;

/**
 * account controller
 *
 * @author 黄光辉
```

```
 * @since 2020/10/31
 **/
@RestController
@RequestMapping("/account")
public class AccountHandler {
    @Autowired
    private UserRepository userRepository;

    @Autowired
    private AdminRepository adminRepository;


 @GetMapping("/login/{username}/{password}/{type}")
    private Object login(@PathVariable String
username,@PathVariable String
password,@PathVariable String type){
        //由于不确定验证的是用户还是管理员，使用多台对返回值
进行处理
        Object object =null;
        switch (type){
            case "user":
                object =
userRepository.login(username,password);
                break;
            case "admin":

 object=adminRepository.login(username,password);
                break;
        }
        return object;
    }
}
```

11、服务消费者整合 Account

- 创建对应的 User 和 Admin 实体类

```
package com.gloryh.entity;

import lombok.Data;
```

```java
import java.util.Date;

/**
 * user 用户实体类
 *
 * @author 黄光辉
 * @since 2020/10/30
 **/
@Data
public class User {
    private long id;
    private String username;
    private String password;
    private String nickname;
    private String gender;
    private String telephone;
    private Date registerdate;
    private String address;
}
```

```java
package com.gloryh.entity;

import lombok.Data;

/**
 * 管理员实体类
 *
 * @author 黄光辉
 * @since 2020/10/31
 **/
@Data
public class Admin {
    private long id;
    private String username;
    private String password;
}
```

- 创建 UserFeign 接口，调用user 服务提供者相关的接口

```java
package com.gloryh.feign;
```

```
import
org.springframework.cloud.openfeign.FeignClient;
import org.springframework.stereotype.Repository;
import
org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PathVariabl
e;

/**
 * Account 的 Feign 声明式接口
 *
 * @author 黄光辉
 * @since 2020/10/31
 **/
@Repository
@FeignClient(value = "account")
public interface AccountFeign {

    /**
     * 登录选项 返回一个 Object 对象，即多态处理 登录人身份
     * @param username 用户名
     * @param password 密码
     * @param type 角色
     * @return Object
     */

 @GetMapping("/account/login/{username}/{password}/
{type}")
    public Object login(@PathVariable String
username,@PathVariable String
password,@PathVariable String type);
}
```

- 启动类中加入Feign声明式接口调用注解——`@EnableFeignClients`
- 创建对应的前后端交互方法(由于 LayUI 的原因，数据不再使用Restful风格)

```
package com.gloryh.controller;
```

```
import com.gloryh.entity.Admin;
import com.gloryh.entity.User;
import com.gloryh.feign.AccountFeign;
import
org.springframework.beans.factory.annotation.Autowi
red;
import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PostMapping
;
import
org.springframework.web.bind.annotation.RequestMapp
ing;
import
org.springframework.web.bind.annotation.RequestPara
m;
import
org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpSession;
import java.util.Date;
import java.util.LinkedHashMap;

/**
 * 服务消费者 account controller
 *
 * @author 黄光辉
 * @since 2020/10/31
 **/
@Controller
@RequestMapping("/account")
public class AccountHandler {
    @Autowired
    private AccountFeign accountFeign;

    @PostMapping("/login")
```

```
    public String login(@RequestParam("username")
String username, @RequestParam("password") String
password, @RequestParam("type") String type,
HttpSession session) {
        //这里由于 object 为LinkedHashMap 类型，强转会出
现错误
        Object object =
accountFeign.login(username, password, type);
        //为处理错误将object先强转为 LinkedHashMap
        LinkedHashMap<String, Object> hashMap =
(LinkedHashMap) object;
        String result=null;
        if (object != null){
            switch (type){
                case "user":
                    //单个赋值处理强转错误
                    User user = new User();

 user.setId(Long.parseLong(hashMap.get("id").toStri
ng()));

 user.setNickname((String)hashMap.get("nickname"));

 session.setAttribute("user",user);
                    result="index";
                    break;
                case "admin":
                    //单个赋值处理强转错误
                    Admin admin = new Admin();

 admin.setId(Long.parseLong(hashMap.get("id").toStr
ing()));

 admin.setUsername((String)hashMap.get("username"))
;

 session.setAttribute("admin",admin);
                    result="main";
                    break;
            }
        }else {
```

```
            result="login";
        }
        return result;
    }
}
```

- 实现登陆跳转不同页面操作

  前端页面源码index：

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet"
th:href="@{/layui/css/layui.css}" media="all">
</head>
<body>
<div class="layui-container" style="width:
700px;height: 600px;margin-top: 0px;padding-top:
60px;">

    <div style="margin-left: 460px; width: 200px;">
        欢迎回来！<a href="/account/redirect/order"
th:text="${session.user.nickname}"></a><a
href="/account/logout">   <button
class="layui-btn layui-btn-warm layui-btn-radius">退
出</button></a>
    </div>

    <table class="layui-hide" id="test" lay-
filter="test"></table>
    <script type="text/html" id="barDemo">
        <a class="layui-btn layui-btn-xs" lay-
event="order">订购</a>
    </script>
    <script th:src="@{/layui/layui.js}"
charset="utf-8"></script>
    <script>
        layui.use('table', function(){
```

```
            var table = layui.table;

            table.render({
                elem: '#test'
                ,url:'/menu/findAll'
                ,title: '菜单列表'
                ,cols: [
                    [
                        {field:'id', width:100,
title: '编号', sort: true}
                        ,{field:'name', width:200,
title: '菜品'}
                        ,{field:'price', width:100,
title: '单价'}
                        ,{field:'flavor',
width:100, title: '口味'}
                        ,{field:'tid',width:100,
 title: '分类',templet:function(data){
                                return
data.type.name
                            }
                        }
                        ,{fixed: 'right', title:'操
作', toolbar: '#barDemo', width:70}
                    ]
                ]
                ,page: true
            });

            //监听行工具事件
            table.on('tool(test)', function(obj){
                var data = obj.data;
                if(obj.event === 'order'){

 window.location.href="/order/save/"+data.id;
                }
            });
        });
    </script>

</div>
```

```
<script>
    //二级菜单联动
    layui.use('element', function(){
        var element = layui.element;

    });
</script>
</body>
</html>
```

前端页面源码 menu_manage：

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet"
th:href="@{/layui/css/layui.css}" media="all">
</head>
<body class="layui-layout-body">
<div class="layui-container" style="margin-top:
50px;width: 750px;">

    <table class="layui-hide" id="test"
style="width: 300px" lay-filter="test"></table>
    <script type="text/html" id="barDemo">
        <a class="layui-btn layui-btn-xs" lay-
event="edit">编辑</a>
        <a class="layui-btn layui-btn-danger layui-
btn-xs" lay-event="del">删除</a>
    </script>
    <script th:src="@{/layui/layui.js}"
charset="utf-8"></script>
    <script>
        layui.use('table', function(){
            var table = layui.table;

            table.render({
                elem: '#test'
```

```
                ,url:'/menu/findAll'
                ,title: '菜单列表'
                ,cols: [
                    [
                        {field:'id', width:100,
title: '编号', sort: true}
                        ,{field:'name', width:200,
title: '菜品'}
                        ,{field:'price', width:100,
title: '单价'}
                        ,{field:'flavor',
width:100, title: '口味'}
                        ,{field:'tid',width:100,
 title: '分类',templet:function(data){
                            return data.type.name
                        }
                    }
                    ,{fixed: 'right', title:'操作',
toolbar: '#barDemo', width:120}
                ]
                ]
                ,page: true
            });

            //监听行工具事件
            table.on('tool(test)', function(obj){
                var data = obj.data;
                if(obj.event === 'del'){
                    layer.confirm('确定要删除吗？',
function(index){

 window.location.href="/menu/deleteById/"+data.id;
                        layer.close(index);
                    });
                } else if(obj.event === 'update'){

 window.location.href="/menu/findById/"+data.id;
                }
            });
        });
    </script>
```

```
</div>
<script>
    //二级菜单联动
    layui.use('element', function(){
        var element = layui.element;

    });
</script>
</body>
</html>
```

- 退出登录实现

```
@GetMapping("/logout")
public String logout(HttpSession session){
    session.invalidate();
    return "login";
}
```

- 测试，访问http://localhost:8030/login.html

12、订单微服务 order 完善

- 完善依赖文件

```
<?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>food_book</artifactId>
        <groupId>org.example</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
```

```
    <artifactId>order</artifactId>
    <!--添加相关依赖-->
    <dependencies>
        <!--Eureka Client 注册中心进行注册-->
        <dependency>

 <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-
netflix-eureka-client</artifactId>
        </dependency>
        <!-- Mybatis 负责和数据库进行交互 -->
        <dependency>

 <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-
starter</artifactId>
            <version>2.1.3</version>
        </dependency>
        <!-- MySQL 驱动  -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-
java</artifactId>
            <version>8.0.21</version>
        </dependency>
        <!--Config Client 从配置中心读取相关配置-->
        <dependency>

 <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-
config</artifactId>
        </dependency>
    </dependencies>

</project>
```

- 创建相关实体类

  Order 实体类

```
package com.gloryh.entity;

```

```java
import lombok.Data;

import java.util.Date;

/**
 * 订单实体类 order
 *
 * @author 黄光辉
 * @since 2020/11/1
 **/
@Data
public class Order {
    private long id;
    private Admin admin;
    private User user;
    private Menu menu;
    private Date date;
    private int state;
}
```

Menu 实体类

```java
package com.gloryh.entity;

import lombok.Data;

/**
 * 对应 menu 数据表的实体类
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@Data
public class Menu {
    private long id;
    private String name;
    private double price;
    private String flavor;
    private Type type;
}
```

User 实体类

```
package com.gloryh.entity;

import lombok.Data;

import java.util.Date;

/**
 * user 用户实体类
 *
 * @author 黄光辉
 * @since 2020/10/30
 **/
@Data
public class User {
    private long id;
    private String username;
    private String password;
    private String nickname;
    private String gender;
    private String telephone;
    private Date registerdate;
    private String address;
}
```

Type实体类

```
package com.gloryh.entity;

import lombok.Data;

/**
 * 菜品分类实体类
 *
 * @author 黄光辉
 * @since 2020/10/28
 **/
@Data
public class Type {
    private long id;
```

```
    private String name;
}
```

OrderVO 实体类

```
package com.gloryh.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;

/**
 * 将返回的实体类类型进行二次封装，已达到前端和服务消费者所需要的格式
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@Data
@NoArgsConstructor
@AllArgsConstructor
public class OrderVO {
    private int code;
    private String msg;
    private int count;
    private List<Order> data;
}
```

Admin实体类

```
package com.gloryh.entity;

import lombok.Data;

/**
 * 管理员实体类
 *
 * @author 黄光辉
 * @since 2020/10/31
```

```
 **/
@Data
public class Admin {
    private long id;
    private String username;
    private String password;
}
```

- 创建 数据库交互方法接口

```
package com.gloryh.repository;

import com.gloryh.entity.Menu;
import com.gloryh.entity.Order;
import com.gloryh.entity.OrderVO;
import org.apache.ibatis.annotations.Param;
import org.springframework.stereotype.Repository;

import java.util.List;

/**
 * Order MyBatis 与数据库交互的调用接口类
 *
 * @author 黄光辉
 * @since 2020/11/1
 **/
@Repository
public interface OrderRepository {
    /**
     * 查询订单列表列表
     *
     * @param index 起始下标
     * @param limit 长度
     * @return  订单列表List
     */
    public List<Order> findAll(int index, int
limit);

    /**
     * 统计订单数量
     * @return int
```

```
     */
    public int count();

    /**
     *  按id 查询订单
     * @param id
     * @return Order
     */
    public Order findById(long id);

    /**
     * 添加订单信息
     * @param order
     */
    public void save(Order order);

    /**
     * 管理员审核订单
     * @param order
     */
    public void update(Order order);

    /**
     * 根据用户 id 查询订单信息
     *
     * @param uid 用户id
     * @param index 起始下标
     * @param limit 长度
     * @return List<order>
     */
    public List<Order> findAllByUid(@Param("uid")
long uid, @Param("index") int index,
@Param("limit") int limit);

    /**
     * 按 用户id 查询 对应订单总数
     * @param uid 用户 id
     * @return int
     */
    public int countByUid(long uid);
```

```
    /**
     * 根据订单状态查询订单信息
     * @param state 状态
     * @param index 起始下标
     * @param limit 长度
     * @return List<order>
     */
    public List<Order>
findAllByState(@Param("state") int state,
@Param("index") int index, @Param("limit") int
limit);

    /**
     * 根据订单状态查询订单数量
     * @param state 状态
     * @return int
     */
    public int countByState(int state);
}
```

- 创建 MyBatis 的相关mapper.xml接口 实现接口 (resources/mapping 文件夹下)

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper
3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-
mapper.dtd">
<mapper
namespace="com.gloryh.repository.OrderRepository">
    <resultMap id="orderMap"
type="com.gloryh.entity.Order">
        <id column="id" property="id"></id>
        <result column="date" property="date">
</result>
        <result column="state" property="state">
</result>
        <collection property="user"
ofType="com.gloryh.entity.User">
            <id column="uid" property="id"></id>
```

```
            <result column="uusername"
property="username"></result>
            <result column="unickname"
property="nickname"></result>
            <result column="telephone"
property="telephone"></result>
            <result column="address"
property="address"></result>
        </collection>
        <collection property="admin"
ofType="com.gloryh.entity.Admin">
            <id column="aid" property="id"></id>
            <result column="ausername"
property="username"></result>
        </collection>
        <collection property="menu"
ofType="com.gloryh.entity.Menu">
            <id column="mid" property="id"></id>
            <result property="name" column="mname">
</result>
            <result property="price"
column="mprice"></result>
            <result property="flavor"
column="mflavor"></result>
        </collection>
    </resultMap>
    <select id="findAll" resultMap="orderMap">
        SELECT o.*,u.nickname AS unickname
,u.username AS uusername ,a.username AS
ausername,m.name AS mname FROM t_order AS o,t_menu
AS m,t_user AS u,t_admin AS a WHERE o.aid =a.id AND
o.mid =m.id AND o.uid =u.id  LIMIT #{param1},#
{param2}
    </select>
    <select id="count" resultType="int">
        SELECT COUNT(*) FROM t_order
    </select>
    <select id="findById" resultMap="orderMap">
```

```
        SELECT o.*,u.nickname AS unickname
,u.username AS uusername ,a.username AS
ausername,m.name AS mname,m.price AS
mprice,m.flavor AS mflavor FROM t_order AS o,t_menu
AS m,t_user AS u,t_admin AS a WHERE  o.mid =m.id
AND o.uid =u.id AND o.id =#{id}
    </select>
    <insert id="save"
parameterType="com.gloryh.entity.Order">
        INSERT INTO t_order(uid,mid,date,state)
VALUES (#{user.id},#{menu.id},#{date},0)
    </insert>
    <update id="update"
parameterType="com.gloryh.entity.Order">
        UPDATE t_order SET aid =#{admin.id},state=1
WHERE id = #{id}
    </update>
    <select id="findAllByUid" resultMap="orderMap">
        SELECT o.*,u.nickname AS unickname
,u.username AS uusername ,a.username AS
ausername,m.name AS mname,m.price AS
mprice,m.flavor AS mflavor FROM t_order AS o,t_menu
AS m,t_user AS u,t_admin AS a WHERE  o.mid =m.id
AND o.uid =u.id AND o.uid = #{uid}   LIMIT #
{index},#{limit}
    </select>
    <select id="countByUid" resultType="int">
        SELECT COUNT(*) FROM t_order WHERE uid =#
{uid}
    </select>
    <select id="findAllByState"
resultMap="orderMap">
        SELECT o.*,u.nickname AS unickname
,u.username AS uusername ,u.telephone
,u.address,a.username AS ausername,m.name AS
mname,m.price AS mprice,m.flavor AS mflavor FROM
t_order AS o,t_menu AS m,t_user AS u,t_admin AS a
WHERE  o.mid =m.id AND o.uid =u.id AND o.state =#
{state}  LIMIT #{index},#{limit}
    </select>
    <select id="countByState" resultType="int">
```

```
        SELECT COUNT(*) FROM t_order WHERE state =#
{state}
    </select>
</mapper>
```

- 配置中心内对应配置文件添加Mapper文件读取

```
spring:
  datasource:
    url: jdbc:mysql://localhost:3305/food_book?
useUnicode=true&characterEncoding=UTF-
8&serverTimezone=UTC
    username: admin
    driver-class-name: com.mysql.cj.jdbc.Driver
    password: 123
mybatis:
  mapper-locations: classpath:/mapping/*.xml
  type-aliases-package: com.gloryh.entity
```

- 启动类中添加扫描注解-=`@MapperScan`

```
@MapperScan("com.gloryh.repository")
```

- 创建前后端交互层

```
package com.gloryh.controller;

import com.gloryh.entity.Order;
import com.gloryh.entity.OrderVO;
import com.gloryh.repository.OrderRepository;
import
org.springframework.beans.factory.annotation.Autowi
red;
import
org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.*;

import java.util.List;

/**
 * order controller
```

```
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@RestController
@RequestMapping("/order")
public class OrderHandler {

    @Autowired
    private OrderRepository orderRepository;

    @PostMapping("/save")
    public void save(@RequestBody Order order){
        orderRepository.save(order);
    }

    @GetMapping("/findAll/{index}/{limit}")
    public OrderVO findAll(@PathVariable int index,
@PathVariable int limit){
        List<Order> orders =
orderRepository.findAll(index,limit);
        int count =orderRepository.count();
        return new OrderVO(0,"",count,orders);
    }

    @GetMapping("/findById/{id}")
    public Order findById(@PathVariable long id){
        return orderRepository.findById(id);
    }

    @PutMapping("/update")
    public void update(@RequestBody Order order){
        orderRepository.update(order);
    }


 @GetMapping("/findAllByUid/{uid}/{index}/{limit}")
    public OrderVO findAllByUid(@PathVariable long
uid,@PathVariable int index,@PathVariable int
limit){
```

```
        List<Order> orders=
orderRepository.findAllByUid(uid, index, limit);
        int count =orderRepository.countByUid(uid);
        return new OrderVO(0,"",count,orders);
    }


 @GetMapping("/findAllByState/{state}/{index}/{limi
t}")
    public OrderVO findAllByState(@PathVariable int
state ,@PathVariable int index,@PathVariable int
limit){
        List<Order> orders=
orderRepository.findAllByState(state, index,
limit);
        int count
=orderRepository.countByState(state);
        return new OrderVO(0,"",count,orders);
    }

}
```

13、服务消费者整合 Order

- 创建对应的 实体类

```
package com.gloryh.entity;

import lombok.Data;

import java.util.Date;

/**
 * 订单实体类 order
 *
 * @author 黄光辉
 * @since 2020/11/1
 **/
@Data
public class Order {
    private long id;
```

```
    private Admin admin;
    private User user;
    private Menu menu;
    private Date date;
    private int state;
}
```

```
package com.gloryh.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;

/**
 * 将返回的实体类类型进行二次封装，已达到前端和服务消费者所需要的格式
 *
 * @author 黄光辉
 * @since 2020/10/27
 **/
@Data
@NoArgsConstructor
@AllArgsConstructor
public class OrderVO {
    private int code;
    private String msg;
    private int count;
    private List<Order> data;
}
```

- 创建 OrderFeign 接口，调用order 服务提供者相关的接口

```
package com.gloryh.feign;

import com.gloryh.entity.Order;
import com.gloryh.entity.OrderVO;
import
org.springframework.cloud.openfeign.FeignClient;
```

```java
import org.springframework.stereotype.Repository;
import
org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PathVariabl
e;
import
org.springframework.web.bind.annotation.PostMapping
;
import
org.springframework.web.bind.annotation.PutMapping;

import java.util.List;

/**
 * Order 的 Feign 声明式接口
 *
 * @author 黄光辉
 * @since 2020/11/1
 **/
@Repository
@FeignClient(value = "order")
public interface OrderFeign {
    /**
     * 查询订单列表列表
     *
     * @param index 起始下标
     * @param limit 长度
     * @return  订单列表OrderVO
     */
    @GetMapping("/order/findAll/{index}/{limit}")
    public OrderVO findAll(@PathVariable int
index,@PathVariable int limit);

    /**
     * 统计订单数量
     * @return int
     */
    @GetMapping("/order/count")
    public int count();
```

```
    /**
     *  按id 查询订单
     * @param id
     * @return Order
     */
    @GetMapping("/order/findById/{id}")
    public Order findById(@PathVariable long id);

    /**
     * 添加订单信息
     * @param order
     */
    @PostMapping("/order/save")
    public void save(Order order);

    /**
     * 管理员审核订单
     * @param order
     */
    @PutMapping("/order/update")
    public void update(Order order);

    /**
     * 按用户 id 查询订单列表
     * @param uid 用户 id
     * @param index 起始下标
     * @param limit 数据长度
     * @return OrderVO
     */

 @GetMapping("/order/findAllByUid/{uid}/{index}/{li
mit}")
    public OrderVO findAllByUid(@PathVariable long
uid,@PathVariable int index,@PathVariable int
limit);

    /**
     * 按订单状态查询订单信息
     * @param state 状态
     * @param index 起始下标
     * @param limit 数据长度
```

```
     * @return OrderVO
     */

 @GetMapping("/order/findAllByState/{state}/{index}
/{limit}")
    public OrderVO findAllByState(@PathVariable int
state,@PathVariable int index,@PathVariable int
limit);
}
```

- 启动类中加入Feign声明式接口调用注解——`@EnableFeignClients`
- 创建对应的前后端交互方法(由于 LayUI 的原因，数据不再使用Restful风格)

```
package com.gloryh.controller;

import com.gloryh.entity.*;
import com.gloryh.feign.OrderFeign;
import
org.springframework.beans.factory.annotation.Autowi
red;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import
org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpSession;
import java.util.Date;

/**
 * 服务消费者 order controller
 *
 * @author 黄光辉
 * @since 2020/11/1
 **/
@Controller
@RequestMapping("/order")
public class OrderHandler {

    @Autowired
```

```
    private OrderFeign orderFeign;

    @GetMapping("/findAll")
    @ResponseBody
    public OrderVO findAll(@RequestParam("page")
int page ,@RequestParam("limit") int limit){
        int index =(page-1)*limit;
        return orderFeign.findAll(index,limit);
    }

    @GetMapping("/save/{mid}")
    public String save(@PathVariable long mid,
HttpSession session){
        //mid通过传值获取，uid通过session获取，后台获取当
前时间date
        User user =(User)
session.getAttribute("user");
        Menu menu =new Menu();
        menu.setId(mid);
        Order order = new Order();
        order.setDate(new Date());
        order.setMenu(menu);
        order.setUser(user);
        orderFeign.save(order);
        return "order";
    }

    @GetMapping("/findAllByUid")
    @ResponseBody
    public OrderVO
findAllByUid(@RequestParam("page") int
page,@RequestParam("limit") int limit,HttpSession
session){
        User user = (User)
session.getAttribute("user");
        int index =(page-1)*limit;
        return
orderFeign.findAllByUid(user.getId(),index,limit);
    }

    @GetMapping("/findAllByState")
```

```
    @ResponseBody
    public OrderVO
findAllByState(@RequestParam("page") int
page,@RequestParam("limit") int limit,HttpSession
session){
        User user = (User)
session.getAttribute("user");
        int index =(page-1)*limit;
        return
orderFeign.findAllByState(0,index,limit);
    }

    @GetMapping("/updateState/{id}")
    public String updateState(@PathVariable long
id,HttpSession session){
        Order order =new Order();
        order.setId(id);
        Admin admin=
(Admin)session.getAttribute("admin");
        order.setAdmin(admin);
        orderFeign.update(order);
        return "order_handler";
    }
}
```

- 实现登陆跳转不同页面操作

  前端页面源码order:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet"
th:href="@{/layui/css/layui.css}" media="all">
</head>
<body>
<div class="layui-container" style="width:
960px;height: 600px;margin-top: 0px;padding-top:
60px;">
```

```
    <div style="margin-left: 460px; width: 300px;">
        <a href="/menu/redirect/index">首页
</a>  |  欢迎回来！<a
href="/account/redirect/order"
th:text="${session.user.nickname}"></a><a
href="/account/logout">   <button
class="layui-btn layui-btn-warm layui-btn-radius">退
出</button></a>
    </div>

    <table class="layui-hide" id="test" lay-
filter="test"></table>
    <script th:src="@{/layui/layui.js}"
charset="utf-8"></script>
    <script>
        layui.use('table', function(){
            var table = layui.table;

            table.render({
                elem: '#test'
                ,url:'/order/findAllByUid'
                ,title: '订单列表'
                ,cols: [
                    [
                        {field:'id', width:100,
title: '编号', sort: true}
                        , {
                        field: 'name', width: 200,
title: '菜品', templet: function (data) {
                            return
data.menu.name
                        }
                        }
                        , {
                        field: 'price', width: 100,
title: '单价', templet: function (data) {
                            return
data.menu.price
                        }
                        }
```

```
                        ,{field:'flavor',
width:100, title: '口味',templet:function(data){
                                return
data.menu.flavor
                            }
                        }
                        ,{field:'date',width:300,
 title: '下单时间'}
                        ,{field:'state', width:160,
title: '订单状态',templet:function(data){
                                var result = "";
                                switch (data.state)
{
                                    case 0:
                                        result = "未
派送";
                                        break;
                                    case 1:
                                        result = "已
派送";
                                        break;
                                }
                                return result
                            }
                        }
                    ]
                ]
                ,page: true
            });

            //监听行工具事件
            table.on('tool(test)', function(obj){
                var data = obj.data;
                if(obj.event === 'order'){

 window.location.href="/order/save/"+data.id;
                }
            });
        });
    </script>
```

```
</div>
<script>
    //二级菜单联动
    layui.use('element', function(){
        var element = layui.element;

    });
</script>
</body>
</html>
```

14、添加过滤器

- 未登录访问index页面自动跳转至登录页面

  在 client 微服务中添加过滤器：

```
package com.gloryh.filter;

import com.gloryh.entity.User;
import org.springframework.stereotype.Component;

import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

/**
 * 用户登录过滤器未登录则跳转回login
 *
 * @author 黄光辉
 * @since 2020/11/2
 **/
@WebFilter(urlPatterns = {"/index.html"},filterName
= "userFilter")
public class UserFilter implements Filter {
    @Override
    public void init(FilterConfig filterConfig)
throws ServletException {
```

```
    }

    @Override
    public void doFilter(ServletRequest
servletRequest, ServletResponse servletResponse,
FilterChain filterChain) throws IOException,
ServletException {
        //判断session中是否有对应的用户信息，没有则返回登录
页面
        HttpServletRequest request=
(HttpServletRequest) servletRequest;
        HttpServletResponse response=
(HttpServletResponse) servletResponse;
        HttpSession session = request.getSession();
        User user =
(User)session.getAttribute("user");
        if(user ==null){
            //用户信息为null，重定向到登录页面
            response.sendRedirect("login.html");
        }else{
            //user不为null，放行

 filterChain.doFilter(servletRequest,servletRespons
e);
        }
    }

    @Override
    public void destroy() {

    }
}
```

在 启动类中添加`@ServletComponentScan`注解

15、完善后台管理页面——管理员相关操作

main页面源码：

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet"
th:href="@{/layui/css/layui.css}" media="all">
    <script type="text/javascript"
th:src="@{/layui/layui.js}"></script>
    <script type="text/javascript" th:src="@{/js/jquery-
3.3.1.min.js}"></script>
    <script type="text/javascript">
        $(function () {
            $("#menu_manage").click(function(){

 $("iframe").attr("src","/menu/redirect/menu_manage");
            });
            $("#menu_add").click(function(){

 $("iframe").attr("src","/menu/redirect/menu_add");
            });
            $("#order_handler").click(function(){

 $("iframe").attr("src","/menu/redirect/order_handler");
            });
            $("#user_manage").click(function(){

 $("iframe").attr("src","/menu/redirect/user_manage");
            });
            $("#user_add").click(function(){

 $("iframe").attr("src","/menu/redirect/user_add");
            });
        })
    </script>
</head>

<div class="layui-layout layui-layout-admin">
    <body class="layui-layout-body">
    <!-- 顶部菜单开始 -->
    <div class="layui-header">
        <div class="layui-logo">外卖点餐管理系统</div>
```

```
        <ul class="layui-nav layui-layout-right">
            <li class="layui-nav-item">
                <a href="javascript:;">
                    <img th:src="@{/images/img.jpeg}"
class="layui-nav-img">
                    <span
th:text="${session.admin.username}"></span>
                </a>
                <dl class="layui-nav-child">
                    <dd><a href="/account/logout">退出系统
</a></dd>
                </dl>
            </li>
        </ul>
    </div>

    <!-- 顶部菜单结束 -->

    <!-- 左侧菜单开始 -->
    <div class="layui-side layui-bg-black">
        <div class="layui-side-scroll">
            <!-- 左侧导航区域（可配合layui已有的垂直导航） -->
            <ul class="layui-nav layui-nav-tree"  lay-
filter="test">
                <li class="layui-nav-item layui-nav-
itemed">
                    <a class="" href="javascript:;">菜品管
理</a>
                    <dl class="layui-nav-child">
                        <dd class="layui-this"><a
id="menu_manage" href="javascript:void(0)">查询菜品</a>
</dd>
                        <dd><a id="menu_add"
href="javascript:void(0)">添加菜品</a></dd>
                        <dd><a id="order_handler"
href="javascript:void(0)">处理订单</a></dd>
                    </dl>
                </li>
                <li class="layui-nav-item">
                    <a href="javascript:;">用户管理</a>
                    <dl class="layui-nav-child">
```

```
                        <dd><a id="user_manage"
href="javascript:void(0)">查询用户</a></dd>
                        <dd><a id="user_add"
href="javascript:void(0)">添加用户</a></dd>
                    </dl>
                </li>
            </ul>
        </div>
    </div>
    <!-- 左侧菜单结束 -->

    <!-- 主体开始 -->
    <div class="layui-body">

        <iframe src="/menu/redirect/menu_manage"
style="width: 100%;height: 100%;border: 0px"></iframe>

    </div>
    <!-- 主体结束 -->

    <!-- 底部开始 -->
    <div class="layui-footer">
        <!-- 底部固定区域 -->
        © 黄光辉
    </div>
    <!-- 底部结束 -->
</div>
<script>
    //二级菜单联动
    layui.use('element', function(){
        var element = layui.element;

    });
</script>
</body>
</html>
```