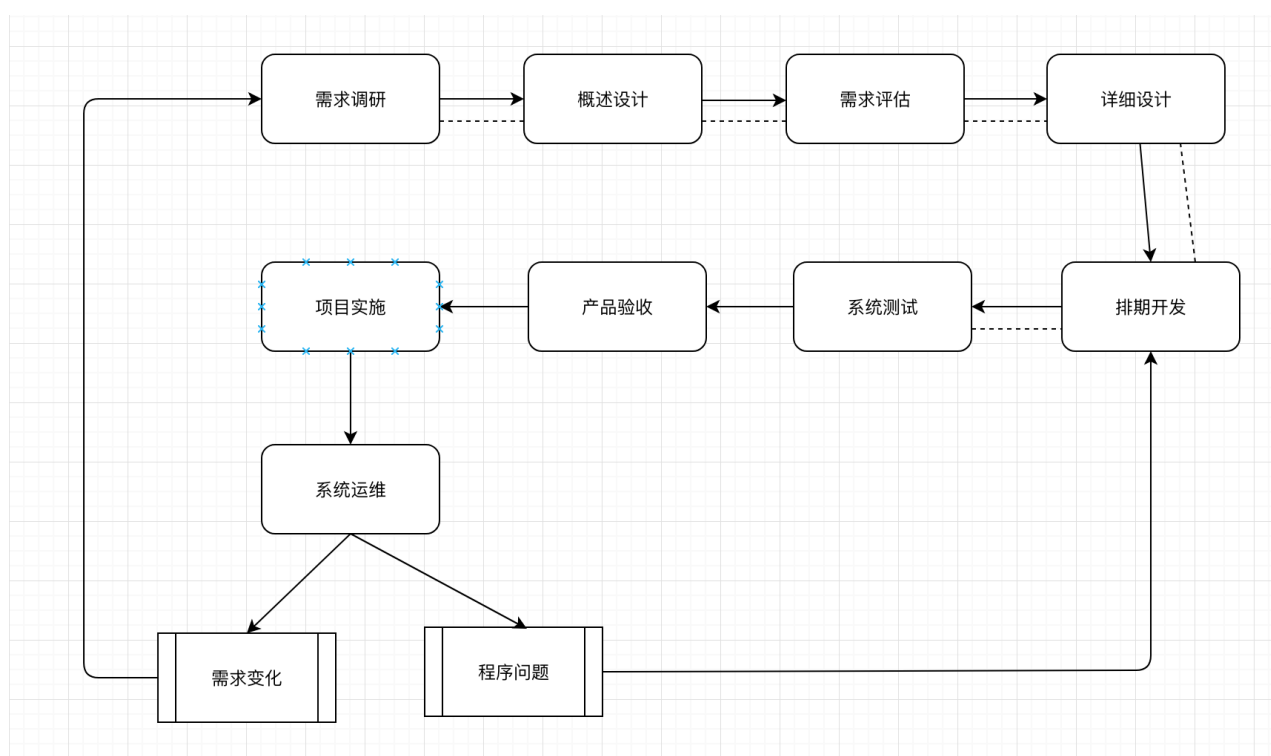


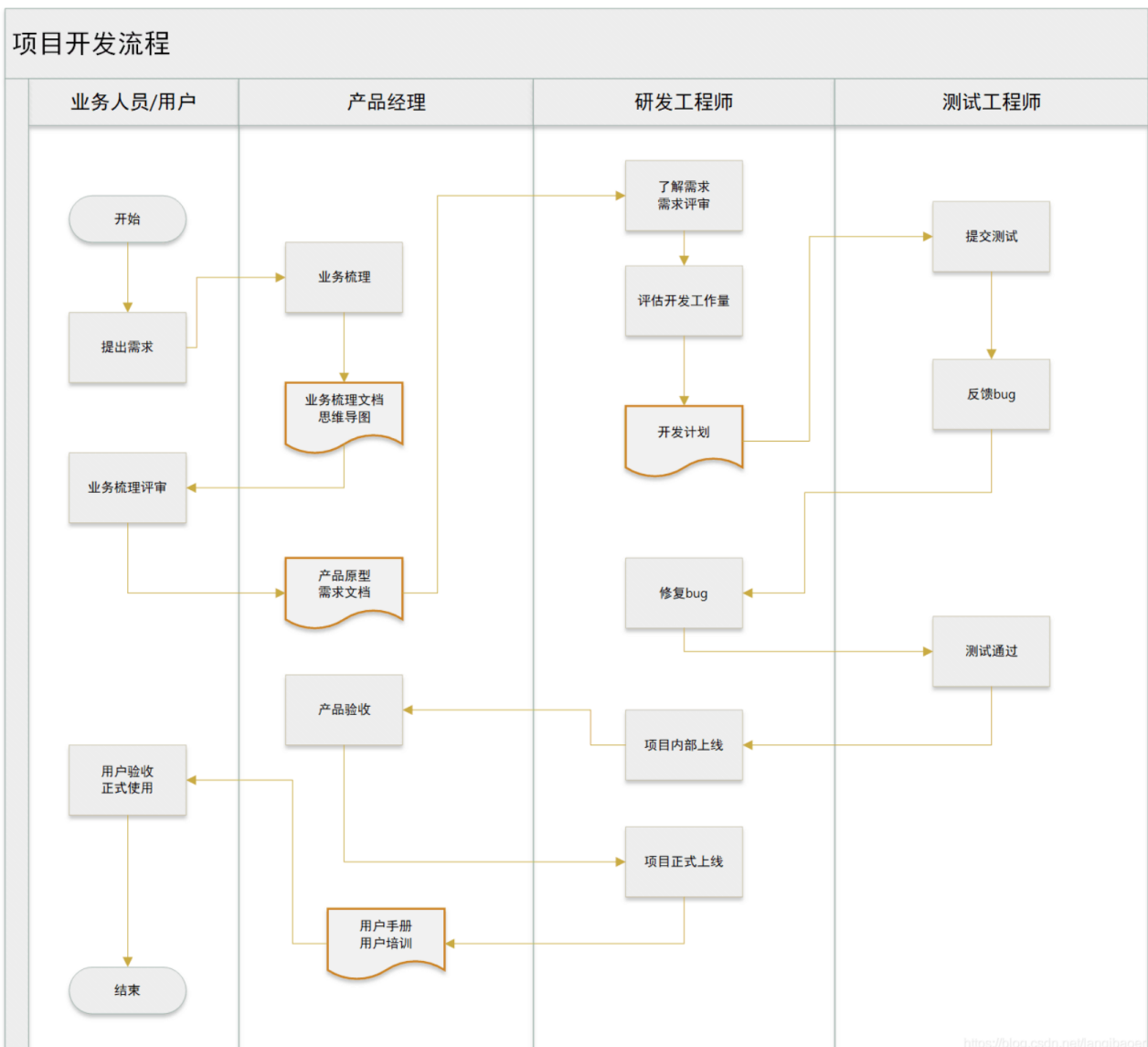
# 微服务项目开发

## 1、项目开发流程

### 1.1、整体流程



### 1.2、人员配置流程



## 2、网络订餐系统

### 2.1、单体应用项目需求

前台展示：

用户登录，用户退出，菜品展示，菜品订购，我的订单

后台管理系统：

管理员登录，管理员退出，菜品管理（增加、查询、删除、修改），订单处理，用户管理（管理员[增加、查询、删除、修改]、普通用户[增加、查询、删除、修改]）

## 2.2、微服务应用需求

微服务应用需求是通过业务来拆分需求

### 2.2.1、服务提供者

account 提供账户服务：用户和管理员的登录退出。

menu 提供菜品服务：添加菜品，删除菜品，修改菜品，查询菜品

order 提供订单服务：添加订单，查询订单，删除订单，处理订单

user 提供用户服务：添加用户、查询用户，删除用户

### 2.2.2、服务消费者

分离出一个服务消费者，调用以上4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、后台管理系统的前端页面和后台接口。

用户/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要调用四个服务提供者的业务逻辑，通过Feign实现负载均衡。

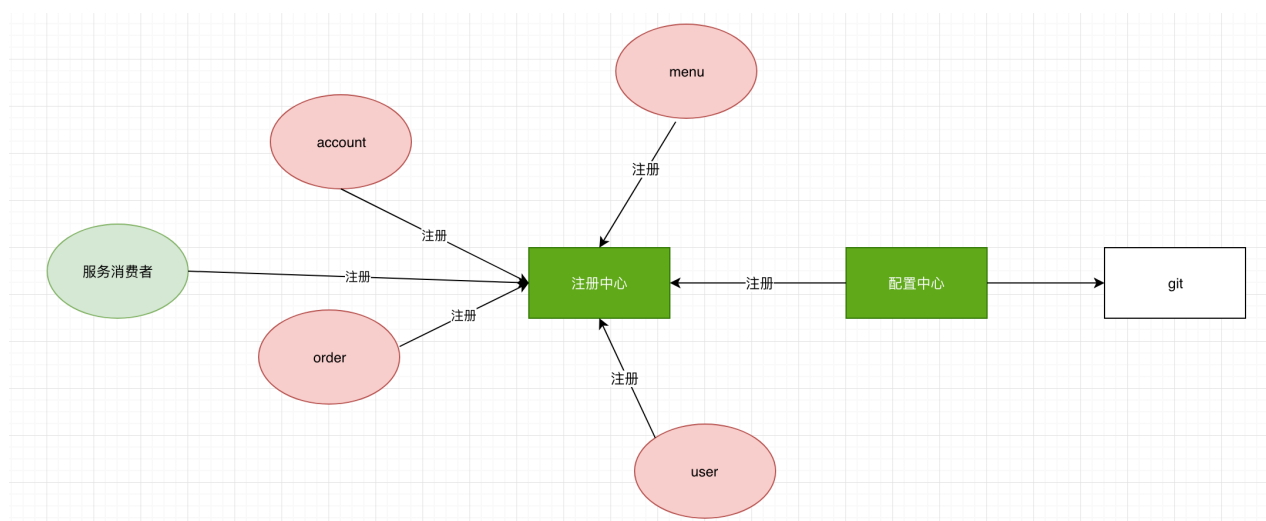
### 2.2.3、注册中心

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

### 2.2.4、配置中心

同时可以使用配置中心来对配置文件进行统一集中管理


## 2.3、网络订餐系统微服务流程



## 3、数据库设计

# 3.1、表设计

## 管理员表

名	类型	长度	小数点	不是 null	键
id	int	11	0	<input checked="" type="checkbox"/>	 1
username 用户名	varchar	255	0	<input type="checkbox"/>	
password 密码	varchar	255	0	<input type="checkbox"/>	

## 用户表

名	类型	长度	小数点	不是 null	键
id	int	11	0	<input checked="" type="checkbox"/>	 1
username 用户名	varchar	255	0	<input type="checkbox"/>	
password 密码	varchar	255	0	<input type="checkbox"/>	
nickname 真识姓名	varchar	255	0	<input type="checkbox"/>	
gender 性别	varchar	255	0	<input type="checkbox"/>	
telephone 电话	varchar	255	0	<input type="checkbox"/>	
registerdate 注册时间	date	0	0	<input type="checkbox"/>	
address 地址	varchar	255	0	<input type="checkbox"/>	

## 菜品分类表

名	类型	长度	小数点	不是 null	键
id	int	11	0	<input checked="" type="checkbox"/>	 1
name 菜品分类	varchar	255	0	<input type="checkbox"/>	

## 菜品表

名	类型	长度	小数点	不是 null	键
id	int	11	0	<input checked="" type="checkbox"/>	 1
name 菜名	varchar	255	0	<input type="checkbox"/>	
price 价格	decimal	10	0	<input type="checkbox"/>	
flavor 味道	varchar	255	0	<input type="checkbox"/>	
tid 菜品分类	int	11	0	<input type="checkbox"/>	

## 订单表

名	类型	长度	小数点	不是 null	键
id	int	11	0	<input checked="" type="checkbox"/>	
uid 用户id	int	11	0	<input type="checkbox"/>	
mid 菜品id	int	11	0	<input type="checkbox"/>	
aid 管理员id	int	11	0	<input type="checkbox"/>	
date 下单日期	date	0	0	<input type="checkbox"/>	
state 下单状态	int	11	0	<input type="checkbox"/>	

## 3.2、创建表

/\*

Navicat MySQL Data Transfer

Source Server : localhost

Source Server Type : MySQL

Source Server Version : 80011

Source Host : localhost

Source Database : ordersystem

Target Server Type : MySQL

Target Server Version : 80011

File Encoding : utf-8

Date: 10/16/2020 21:57:49 PM

\*/

SET NAMES utf8;

```

SET FOREIGN_KEY_CHECKS = 0;

-- -----
--  Table structure for `t_admin`
-- -----

DROP TABLE IF EXISTS `t_admin`;
CREATE TABLE `t_admin` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(255) DEFAULT NULL,
  `password` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT
CHARSET=utf8;

-- -----
--  Records of `t_admin`
-- -----

BEGIN;
INSERT INTO `t_admin` VALUES ('1', 'admin',
'admin');
COMMIT;

-- -----
--  Table structure for `t_menu`
-- -----

DROP TABLE IF EXISTS `t_menu`;

```

```
CREATE TABLE `t_menu` (  
    `id` int(11) NOT NULL AUTO_INCREMENT,  
    `name` varchar(255) DEFAULT NULL,  
    `price` decimal(10,0) DEFAULT NULL,  
    `flavor` varchar(255) DEFAULT NULL,  
    `tid` int(11) DEFAULT NULL,  
    PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT  
CHARSET=utf8;
```

```
-- -----  
-- Records of `t_menu`  
-- -----
```

```
BEGIN;  
INSERT INTO `t_menu` VALUES ('1', '香酥鸡',  
'34', '香', '1'), ('2', '梅菜扣肉', '65',  
'香', '1'), ('3', '三杯鸡', '33', '香', '1'),  
( '4', '毛血旺', '68', '辣', '1'), ('5', '菠菜  
抖粉丝', '12', '辣', '2'), ('6', '凉拌豆腐皮',  
'34', '苦', '2'), ('7', '酱牛肉', '32', '微  
辣', '2'), ('8', '鱼头豆腐汤', '32', '香',  
'3'), ('9', '瘦肉鸡蛋白菜汤', '21', '甜',  
'3'), ('10', '西葫芦虾仁蒸饺', '10', '香',  
'4'), ('11', '蛋炒饭', '15', '咸', '4'),  
( '12', '椰蓉面包', '8', '香', '5');  
COMMIT;
```



```
-----  
-- Table structure for `t_order`  
-----  
  
DROP TABLE IF EXISTS `t_order`;  
CREATE TABLE `t_order` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `uid` int(11) DEFAULT NULL,  
  `mid` int(11) DEFAULT NULL,  
  `aid` int(11) DEFAULT NULL,  
  `date` date DEFAULT NULL,  
  `state` int(11) DEFAULT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT  
CHARSET=utf8;  
  
-----  
-- Records of `t_order`  
-----  
  
BEGIN;
```

```
INSERT INTO `t_order` VALUES ('1', '1',  
'7', '1', '2020-10-07', '1'), ('2', '1',  
'2', '1', '2020-10-07', '1'), ('3', '1',  
'5', '1', '2020-10-07', '1'), ('4', '1',  
'9', '1', '2020-10-07', '1'), ('5', '1',  
'10', '1', '2020-10-07', '1'), ('6', '1',  
'10', null, '2020-10-07', '0'), ('7', '1',  
'10', '1', '2020-10-07', '1'), ('8', '1',  
'6', '1', '2020-10-07', '1'), ('9', '1',  
'10', '1', '2020-10-07', '1'), ('10', '1',  
'7', '1', '2020-10-07', '1'), ('11', '2',  
'8', null, '2020-10-07', '0'), ('12', '2',  
'12', null, '2020-10-07', '0');
```

```
COMMIT;
```

```
-- -----  
-- Table structure for `t_type`  
-- -----
```

```
DROP TABLE IF EXISTS `t_type`;
```

```
CREATE TABLE `t_type` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `name` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT  
CHARSET=utf8;
```

```

-- -----
--  Records of `t_type`
-- -----

BEGIN;
INSERT INTO `t_type` VALUES ('1', '热菜'),
('2', '凉菜'), ('3', '烫羹'), ('4', '主食'),
('5', '烘焙');
COMMIT;

-- -----
--  Table structure for `t_user`
-- -----

DROP TABLE IF EXISTS `t_user`;
CREATE TABLE `t_user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(255) DEFAULT NULL,
  `password` varchar(255) DEFAULT NULL,
  `nickname` varchar(255) DEFAULT NULL,
  `gender` varchar(255) DEFAULT NULL,
  `telephone` varchar(255) DEFAULT NULL,
  `registerdate` date DEFAULT NULL,
  `address` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT
CHARSET=utf8;

```

```
-----  
-- Records of `t_user`  
-----  
  
BEGIN;  
INSERT INTO `t_user` VALUES ('1',  
'zhangsan', 'zhangsan', '张三', '男',  
'17890980967', '2020-09-09', '北京'), ('2',  
'lisi', 'lisi', '李四', '女', '18867851543',  
'2020-10-10', '天津');  
COMMIT;  
  
SET FOREIGN_KEY_CHECKS = 1;
```

## 4、微服务架构搭建

### 4.1、创建父工程

ordering

### 4.2、导入依赖

```
<!-- 引入Spring Boot依赖 -->  
<parent>  
  
<groupId>org.springframework.boot</groupId>
```

```
<artifactId>spring-boot-starter-  
parent</artifactId>  
    <version>2.0.7.RELEASE</version>  
</parent>  
  
<dependencies>  
    <dependency>  
  
<groupId>org.springframework.boot</groupId>  
        <artifactId>spring-boot-starter-  
web</artifactId>  
    </dependency>  
    <!-- 解决JDK 9以上版本没有JAXB API的问题 -->  
    <dependency>  
        <groupId>javax.xml.bind</groupId>  
        <artifactId>jaxb-api</artifactId>  
        <version>2.3.0</version>  
    </dependency>  
    <dependency>  
        <groupId>com.sun.xml.bind</groupId>  
        <artifactId>jaxb-impl</artifactId>  
        <version>2.3.0</version>  
    </dependency>  
    <dependency>  
        <groupId>com.sun.xml.bind</groupId>  
        <artifactId>jaxb-core</artifactId>
```

```

        <version>2.3.0</version>
    </dependency>
<!--通过可升级模块的概念，仅以模块化形式支持独立
API 。使用它们，可以在任何阶段使用该模块的版本，即
在编译时，构建时或运行时。-->
    <dependency>
        <groupId>javax.activation</groupId>
        <artifactId>activation</artifactId>
        <version>1.1.1</version>
    </dependency>
<!--以注解形式来简化java代码-->
<dependency>

<groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
</dependencies>
<!-- 引入Spring Cloud依赖，管理Spring Cloud生
态各个组件 -->
<dependencyManagement>
    <dependencies>
        <dependency>

        <groupId>org.springframework.cloud</groupI
d>

```

```
                <artifactId>spring-cloud-
dependencies</artifactId>

        <version>Finchley.SR2</version>
                <type>pom</type>
                <scope>import</scope>
        </dependency>
</dependencies>
</dependencyManagement>
<!--指定JDK的编译版本-->
<build>
        <plugins>
                <plugin>

        <groupId>org.apache.maven.plugins</groupId>
        >
                <artifactId>maven-compiler-
plugin</artifactId>
                <version>3.8.0</version>
                <configuration>
                        <source>1.8</source>
<!--language level-->
                        <target>1.8</target>
<!--java compiler version-->
                </configuration>
        </plugin>
```

```
        </plugins>
    </build>
```

## 4.3、创建注册中心

### 4.3.1、创建工程

ordering-eureka

### 4.3.2、导入依赖

pom.xml

```
<dependencies>
    <dependency>

    <groupId>org.springframework.cloud</groupId>
    <
        <artifactId>spring-cloud-starter-
netflix-eureka-server</artifactId>
        <version>2.0.2.RELEASE</version>
    </dependency>
</dependencies>
```



### 4.3.3、编写配置文件

application.yml

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

### 4.3.4、编写启动类

EurekaOrderingApplication.java

```
package com.wv;  
  
import  
org.springframework.boot.SpringApplication;  
import  
org.springframework.boot.autoconfigure.Spr  
ingBootApplication;  
import  
org.springframework.cloud.netflix.eureka.se  
rver.EnableEurekaServer;  
  
@SpringBootApplication  
@EnableEurekaServer  
public class EurekaOrderingApplication {  
    public static void main(String[] args)  
{  
  
    SpringApplication.run(EurekaOrderingApplic  
ation.class,args);  
    }  
}
```

## 4.4、创建配置中心

## 4.4.1、创建工程

ordering-config

## 4.4.2、导入依赖

pom.xml

```
<dependencies>
    <dependency>

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

        <version>2.0.2.RELEASE</version>
    </dependency>
    <dependency>

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

### 4.4.3、编写配置文件

applicaton.yml

```
server:
  port: 8762
spring:
  application:
    name: ordering-config
  profiles:
    active: native
  cloud:
    config:
      server:
        native:
          search-locations:
classpath:/shared #在shared路径下创建各个微服务
对应的配置文件
eureka:
  client:
    service-url:
      defaultZone:
http://localhost:8761/eureka/
```

在resources目录下创建shared文件夹

## 4.4.4、编写启动类

ConfigOrderingApplication.java

```
package com.wy;

import
org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.Spr
ingBootApplication;
import
org.springframework.cloud.config.server.Ena
bleConfigServer;

@SpringBootApplication
@EnableConfigServer
public class ConfigOrderingApplication {
    public static void main(String[] args)
    {

        SpringApplication.run(ConfigOrderingApplic
ation.class,args);
    }
}
```

# 5、菜品服务

---

## 5.1、创建工程

ordering-menu

## 5.2、导入依赖

pom.xml

```
<dependencies>
    <dependency>

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

        <version>2.0.2.RELEASE</version>
    </dependency>
    <dependency>

        <groupId>org.mybatis.spring.boot</groupId>
        <artifactId>mybatis-spring-
boot-starter</artifactId>
        <version>1.3.1</version>
    </dependency>
</dependencies>
```

```
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-
java</artifactId>
            <version>8.0.11</version>
        </dependency>
        <dependency>

            <groupId>org.springframework.cloud</groupI
d>

            <artifactId>spring-cloud-
starter-config</artifactId>

            <version>2.0.2.RELEASE</version>
        </dependency>
    </dependencies>
```

## 5.3、编写配置文件

bootstrap.yml

```
spring:
  application:
    name: menu
  profiles:
    active: dev
  cloud:
    config:
      uri: http://localhost:8762
      fail-fast: true #开启配置文件失败快速响应
```

menu-dev.xml

resources->shared->menu-dev.xml

```
server:
  port: 8020
spring:
  application:
    name: menu
  datasource:
    name: ordersystem
    url:
jdbc:mysql://localhost:3306/ordersystem
    username: root
    password: Root123456
eureka:
  client:
```



```
service-url:
    defaultZone:
http://localhost:8761/eureka/
```

## 5.4、编写启动类

MenuApplication.java

```
package com.wv;

import
org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.Spr
ingBootApplication;

@SpringBootApplication
public class MenuApplication {
    public static void main(String[] args)
    {

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

## 5.5、菜品服务操作

### 5.5.1、配置实体类

Menu.java

```
package com.ww.entity;

import lombok.Data;

@Data
public class Menu {
    private long id;
    private String name;
    private double price;
    private String flavor;
}
```

### 5.5.2、编写操作接口

MenuRepository.java

```
package com.ww.repository;

import com.ww.entity.Menu;

import java.util.List;
```

```
public interface MenuRepository {  
    public List<Menu> findAll(int index,  
int limit);  
    public int count();  
    public void save(Menu menu);  
    public void update(Menu menu);  
    public void delete(long id);  
    public Menu findById(long id);  
}
```

### 5.5.3、修改menu-dev.yml配置

ordering-config->shared->menu-dev.yml

```
mybatis:  
  mapper-locations:  
    classpath:/mapping/*.xml  
  type-aliases-package: com.ww.entity #配置  
文件中可省略前缀
```

### 5.5.4、配置操作映射文件

resources->mapping->MenuRepository.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.ww.repository.MenuRepository
">
    <!-- 配置查询所有操作 -->
    <select id="findAll"
resultType="com.ww.entity.Menu">
        select * from t_menu limit #
{param1},#{param2}
    </select>
    <select id="count" resultType="int">
        select count(id) from t_menu
    </select>
    <select id="findById"
parameterType="long"
resultType="com.ww.entity.Menu">
        select * from t_menu where id=#{id}
    </select>
    <insert id="save"
parameterType="com.ww.entity.Menu">
        insert into
t_menu(mame,price,flavor)values(#{name},#
{price},#{flavor})
```

```
        </insert>

        <update id="update"
parameterType="com.ww.entity.Menu">
            update t_menu set name=#
{name},price=#{price},flavor=#{flavor}
where id=#{id}
        </update>

        <delete id="delete"
parameterType="long">
            delete from t_menu where id=#{id}
        </delete>
    </mapper>
```

## 5.5.5、创建控制器

MenuHandler.java

```
package com.ww.controller;

import com.ww.entity.Menu;
import com.ww.repository.MenuRepository;
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.web.bind.annotation.GetMapping;
```

```
import
org.springframework.web.bind.annotation.Pat
hVariable;
import
org.springframework.web.bind.annotation.Req
uestMapping;
import
org.springframework.web.bind.annotation.Res
tController;

import java.util.List;

@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);
    }
}
```

```
}
```

## 5.5.6、修改启动类

```
//.....  
@MapperScan("com.ww.repository")  
public class MenuApplication {  
    //.....  
}
```

## 5.5.7、启动服务测试

<http://localhost:8020/menu/findAll/0/10>

# 5.6、菜品服务调用

## 5.6.1、创建工程

ordering-client

## 5.6.2、导入依赖

pom.xml

```
<dependencies>  
    <dependency>
```

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

```
    <artifactId>spring-cloud-  
starter-netflix-eureka-client</artifactId>
```

```
<version>2.0.2.RELEASE</version>
```

```
</dependency>
```

```
<dependency>
```

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

```
    <artifactId>spring-cloud-  
starter-openfeign</artifactId>
```

```
<version>2.0.2.RELEASE</version>
```

```
</dependency>
```

```
<dependency>
```

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

```
    <artifactId>spring-boot-  
starter-thymeleaf</artifactId>
```

```
</dependency>
```

```
<dependency>
```



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

    <artifactId>spring-cloud-
starter-config</artifactId>

    <version>2.0.2.RELEASE</version>
    </dependency>
</dependencies>
```

**注意：**

使用模板thymeleaf

Spring Boot默认存放模板页面的路径在  
src/main/resources/templates

Thymeleaf默认的页面文件后缀是.html

### 5.6.3、编写配置文件

bootstrap.yml

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

client-dev.yml

resources->shared->client-dev.xml

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

## 5.6.4、编写启动类

ClientOrderingApplication.java

```
package com.ww;

import
org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.Spr
ingBootApplication;
import
org.springframework.cloud.openfeign.EnableF
eignClients;

@SpringBootApplication
@EnableFeignClients
public class ClientOrderingApplication {
    public static void main(String[] args)
    {

        SpringApplication.run(ClientOrderingApplic
ation.class,args);
    }
}
```

## 5.6.5、配置实体类

将Menu.java复制到ordering-client->com.ww.entity

## 5.6.6、创建Feign调用

MenuFeign.java

```
package com.ww.feign;

import com.ww.entity.Menu;
import
org.springframework.cloud.openfeign.FeignCl
ient;
import
org.springframework.web.bind.annotation.Get
Mapping;
import
org.springframework.web.bind.annotation.Pat
hVariable;

import java.util.List;

@FeignClient(value="menu")
public interface MenuFeign {

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

```
}
```

## 5.6.7、创建控制器

ClientMenuHandler.java

```
package com.ww.controller;

import com.ww.entity.Menu;
import com.ww.feign.MenuFeign;
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RestController;

import java.util.List;
```

```
@RestController
@RequestMapping("/client/menu")
public class ClientMenuHandler {
    @Autowired
    private MenuFeign menuFeign;
    @GetMapping("/findAll/{index}/{limit}")
    public List<Menu>
findAll(@PathVariable("index") int index,
@PathVariable("limit") int limit){
        return
menuFeign.findAll(index, limit);
    }
}
```

## 5.6.8、启动服务测试

<http://localhost:8030/client/menu/findAll/0/10>

## 5.7、菜品服务页面

### 5.7.1、前端UI框架

创建static文件夹和index.html页面

Layui

下载: <https://www.layui.com/>

解压后将框架放入到static.layui文件夹 (client->resources->static)

## Jquery

下载: <https://jquery.com/download/> 或<http://www.aib173.com/jquery/>

解压后将框架放入到js文件夹 (client->resources->static->js)

## 5.7.2、index.html

**thymeleaf:** Thymeleaf 最为显著的特征是增强属性, 任何属性都可以通过th:xx 来完成交互, 例如th:value最终会覆盖value属性。

教程: <https://www.jianshu.com/p/908b48b10702>

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
```

[illegible]



```
<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 ''
            }
        }
    )
}
```

```

        }
    }
    ,{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.i
d;

    }
    });
});
</script>

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

```

```
        var element = layui.element;

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

访问:

<http://localhost:8030/index.html>

问题:

不能直接访问html文件，不然无法加载th

修改: ClientMenuHandler.java

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

启动服务测试

<http://localhost:8030/client/redirect/index>

欢迎回来!

退出

编号 ▾	菜品	单价	口味	分类	操作
数据接口请求异常：错误					

## 传参方式：

► GET <http://localhost:8030/client/findAll/0/10?page=1&limit=10> 500

## 修改：ClientMenuHandler.java

```
//.....
@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);
}
//.....
```

## 5.7.3、返回数据格式不对

欢迎回来!

退出

编号 ▾	菜品	单价	口味	分类	操作
返回的数据不符合规范，正确的成功状态码应为：“code”：0					

### 数据正确正确返回格式：

内置模块

弹出层 layer

日期与时间选择 laydate

即时通讯 layim

分页 laypage

模板引擎 laytpl

数据表格 table

表单 form

文件上传 upload

headers

接口的请求头。如：headers: {token: 'sasasas'}

数据格式解析的回调函数，用于将返回的任意数据格式解析成 table 组件规定的格式。

table 组件默认规定的格式为（参考：</demo/table/user/>）：

默认规定的格式

layui.code

```
01. {
02.   "code": 0,
03.   "msg": "",
04.   "count": 1000,
05.   "data": [{}, {}]
06. }
07.
```

很多时候，您接口返回的数据格式并不一定都符合 table 默认规定的格式，比如：

## 创建MenuVO模块

ordering-menu->com.ww.entity->MenuVO.java

```
package com.ww.entity;

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

import java.util.List;

@Data
```

```

@AllArgsConstructor
@NoArgsConstructor
public class MenuVO {
    private int code;
    private String msg;
    private int count;
    private List<Menu> data;
}

```

## 修改MenuHandler.java

ordering-menu->com.ww.controller-  
>MenuHandler.java

```

//.....
@GetMapping("/findAll/{index}/{limit}")
public MenuVO
findAll(@PathVariable("index") int index,
@PathVariable("limit") int limit){
    return new
MenuVO(0,"",1000,menuRepository.findAll(ind
ex,limit));
}
//.....

```

## 拷贝：MenuVO.java

MenuVO.java拷贝到ordering-client->com.ww.entity

## 修改：MenuFeign.java

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

## 修改：ClientMenuHandler.java

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

## 5.7.4、启动服务测试

<http://localhost:8030/client/redirect/index>