前后端接口

- 说明：
  - 交互方法参考文档：前端和后端的数据交互(jquery ajax+python flask+mysql)_写代码的柯长的博客-CSDN博客
  - 建议参照源码中的web/static/js/common.js设置好对应的方法库，便于具体的js功能调用。
  - 使用JQuery所封装的$.ajax方式进行前端和服务器之间的异步通讯。
    - jQuery - AJAX 简介 (w3school.com.cn)
    - 其结构如下：
      - 语法：$.ajax({name:value, name:value, ... })
      - 参数：
        -

名称	值类型	描述
async	Boolean	表示请求是否异步处理。默认是 true。
beforeSend(xhr)	Function	发送请求前运行的函数。
cache	Boolean	表示浏览器是否缓存被请求页面。默认是 true。
complete(xhr,status)	Function	请求完成时运行的函数（在请求成功或失败之后均调用，即在 success 和 error 函数之后）。
contentType	String	发送数据到服务器时所使用的内容类型。默认是："application/x-www-form-urlencoded"。
context	String	为所有 AJAX 相关的回调函数规定 "this" 值。
data	Object	规定要发送到服务器的数据。
dataFilter(data,type)	Function	用于处理 XMLHttpRequest 原始响应数据的函数
dataType	String	预期的服务器响应的数据类型。
error(xhr,status,error)	Function	如果请求失败要运行的函数。
global	Boolean	规定是否为请求触发全局 AJAX 事件处理程序。默认是 true。
ifModified	Boolean	规定是否仅在最后一次请求以来响应发生改变时才请求成功。默认是 false。
jsonp	String	在一个 jsonp 中重写回调函数的字符串。
jsonpCallback	Function	在一个 jsonp 中规定回调函数的名称。
password	String	规定在 HTTP 访问认证请求中使用的密码。
processData	Boolean	规定通过请求发送的数据是否转换为查询字符串。默认是 true。
scriptCharset	String	规定请求的字符集。
success(result,status,xhr)	Function	当请求成功时运行的函数。
timeout	Number	设置本地的请求超时时间（以毫秒计）。
traditional	Boolean	规定是否使用参数序列化的传统样式。
type	String	规定请求的类型（GET 或 POST）。
url	String	规定发送请求的 URL。默认是当前页面。
username	String	规定在 HTTP 访问认证请求中使用的用户名。
xhr	Function	用于创建 XMLHttpRequest 对象的函数。

实例

下例演示了使用 AJAX 请求改变 元素的文本：

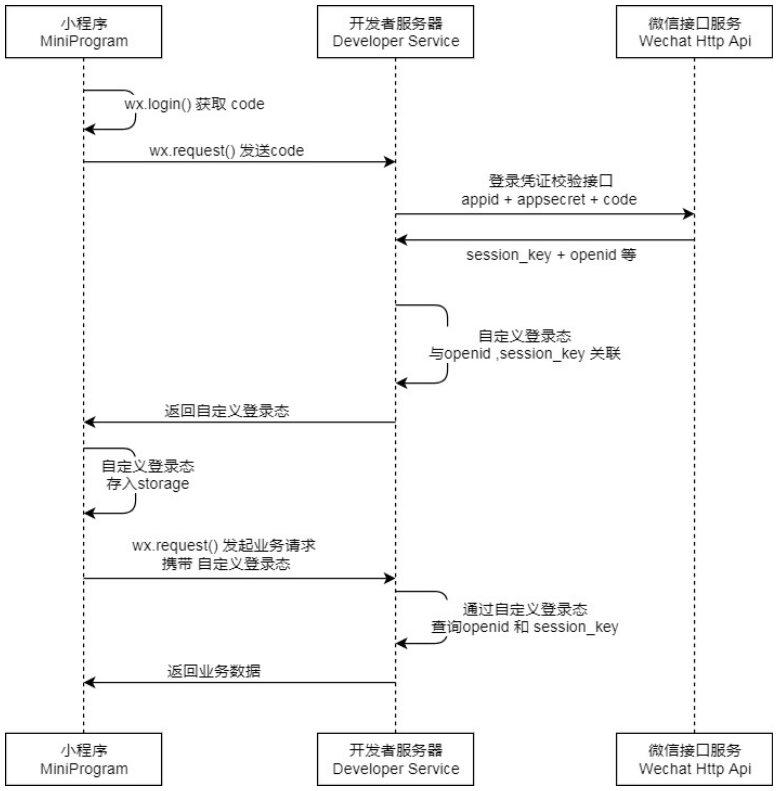
- 分页：仅用于web管理后台，管理用户、管理菜品、管理订单三个页面具有分页功能。
  - 每页15条记录
- 登录态验证：
  - 后端根据前端的request.cookies，将对应字段的信息作为token存在数据库中，对cookies进行设置后发回前端。
  - 前端每次向后端发送请求时会自动随着request把cookies发到后端，由后端对其解析并从数据库验证有效性。
  - 这种形式避免了前端每次都往后端发送token。

- **管理员后台页面**
  - 登录页面：
    - type: POST
    - data:
      - {
        - login_name
        - login_pwd
      - };
    - dataType: json
    - success: function(res){}
      - res是后端返回前端的结果，不同情况下取值如下(后面简写为res)：
        - 用户名或密码错误：
          - code=0
          - msg='用户名或密码错误'
        - 登陆成功：
          - code=1
          - msg='登录成功'
  - 餐品管理页面
    - type: POST
    - data: {
      - page(空： 全部；1-n：对应页面),
      - pageSize(空：10；1-n：页面条数)
      - }
    - dataType: json
    - success: function(res){}
      - res的结构：
        - {{
          - dish_id
          - dish_name
          - dish_class
        - },{etc. }};
  - 餐品编辑[餐品的增删修改]
    - type：POST
    - data：
      - {
        - dish_name,

        - dish_class,
        - dish_price,
        - dish_description
      - };
    - dataType: json
    - success: function(res){}
      - 编辑成功
        - code=1
        - msg='编辑成功'
      - 已存在同名餐品
        - code=0
        - msg='添加失败，请检查商品名称'
- 财务管理页面
  - type：POST
  - data:
    - {
      - page(-1： 全部；1-n：对应页面,
      - pageSize(空：10；1-n：页面条数）
    - }
  - dataType: json
  - success: function(res){}
    - res结构
    - {{
      - order_id
      - order_dish
      - order_price
      - order_status
    - },{etc. }};
- 订单状态修改
  - type: POST
  - data:
    - {
      - order_id,
      - order_status
    - };
  - dataType: json

    - success: function(res){}
        - 修改成功：
            - code=0
            - msg='订单状态修改成功'
        - 修改错误：
            - code=1
            - msg='订单状态修改失败，请联系用户支持'
- 统计管理页面：单品销售量统计
    - type:POST
    - data:
        - {
            - page(-1： 全部；1-n：对应页面，
            - pageSize(空：10；1-n：页面条数）
        - }
    - dataType: json
    - success: function(res){}
        - res结构
            - {{
                - dish_id
                - dish_name
                - dish_number
                - dish_score
            - },{etc. }};
- 统计管理页面：售卖总额【已成交金额】
    - type: POST
    - data:
        - {
        - }
    - dataType: json
    - success: function(res){}
        - res结构：
            - {
                - sell_quantity
            - };
- 统计管理页面：订单详情
    - type: POST

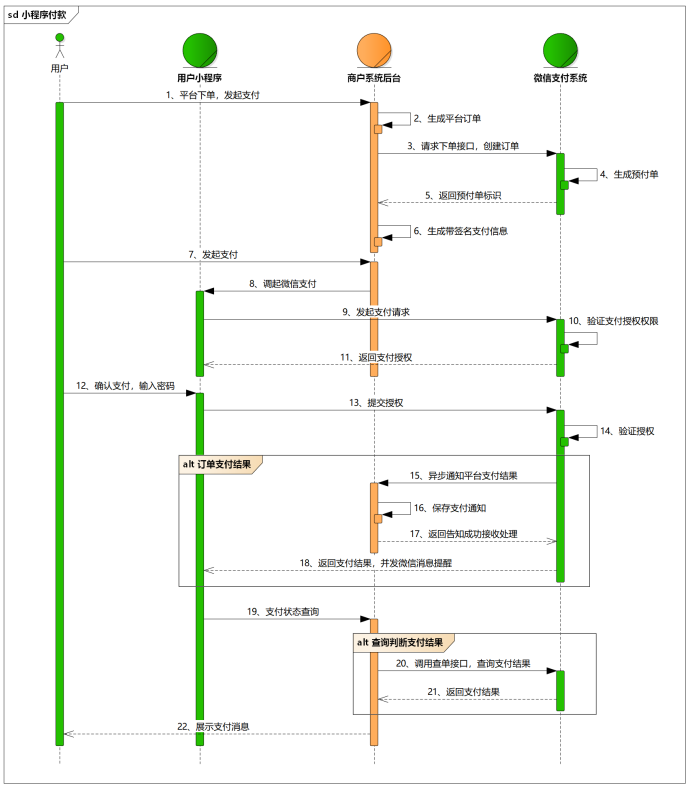
        - data:
            - {
                - order_status,
                - page(-1： 全部；1-n：对应页面,
                - pageSize(空：10；1-n：页面条数）
            - }
        - dataType: json
        - success: function(res){}
            - res结构：
                - {{
                    - order_id
                    - order_price
                    - user_id
                    - order_address
                    - order_time
                - },{etc. }};
- **微信小程序：**
    - 参考资料：
        - **目录结构 | 微信开放文档 (qq.com)**
        - RequestTask | 微信开放文档 (qq.com)
        - 小程序登录 | 微信开放文档 (qq.com)
    - request结构：
        - wx.request({
            - url: 'example.php', //仅为示例，并非真实的接口地址
            - data: {
                - x: '',
                - y: ''
                - },
            - header: { 'content-type': 'application/json' // 默认值 },
            - success (res) {
                - console.log(res.data)
            - }
        - })
    - 微信登录：小程序登录 | 微信开放文档 (qq.com)
        - 流程：

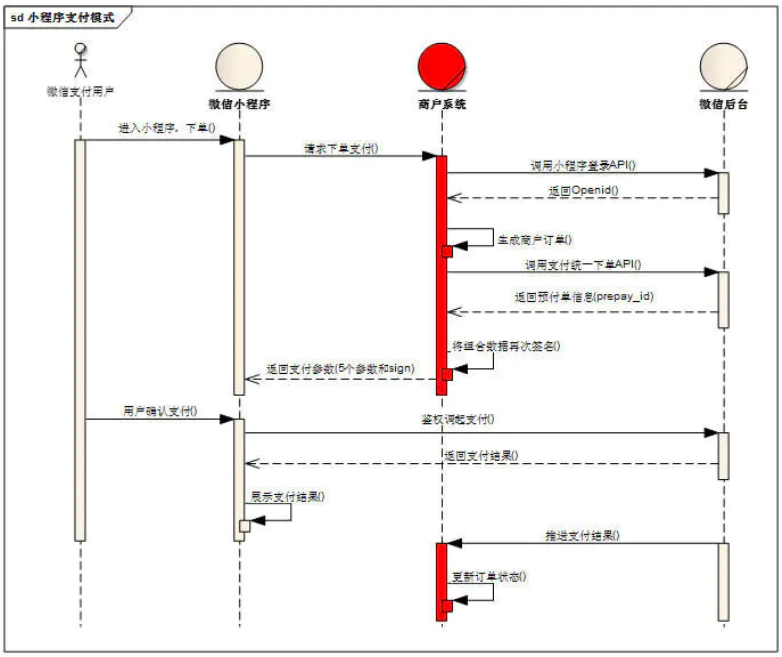


- wx.login()内调用wx.request发送临时登录凭证code：
  - data:{
    - code: res.code //res是login函数在本地获取的用户登录凭证
  - }
  - success (res) {}
    - res结构：
    - {
      - status：登录状态（0：登录成功，1：登录失败），
      - user_id：用户账号,
    - }
- 餐品列表：
  - data:{
    - }
  - success(res){}
    - res结构：
    - {{
      - dish_id,
      - dish_name,
      - dish_type(菜的大类,
      - dish_tag(菜的小类,
      - dish_thumbnail
    - },{etc.}}

- 特定餐品详情:
  - data:{
    - dish_id
    - }
  - success(res){}
    - res结构
    - {
      - dish_name,
      - dish_thumbnail,
      - dish_price,
      - dish_sold,
      - dish_score
    - }
- 提交订单【平台下单；后端需要同步完成微信支付接口的预付单生成】:
  - data:{{
    - dish_id,
    - num_dish,
    - },{etc. }}
  - success(res){}
    - res结构:
    - {
      - order_status,
      - order_id,
      - order_time,
      - order_price,
      - order_address
    - }
- 查询各种状态的订单:
  - data:{
    - status_req
    - }
  - success(res){}
    - res结构：{
      - {
        - order_status,
        - order_id,

            - order_time,
            - order_address
            - },
        - {etc.}
    - }
- 【替代方案】伪付款：每个用户无限钱，后端只接受付款数据并返回支付情况。
    - data：{
        - order_id,
        - mony_amount
    - };
    - success(res){}
        - res结构：{
            - pay_status(0:失败； 1:成功
        - }
- 【正经的】订单付款：微信支付开发指引，微信小程序支付API - 简书 (jianshu.com)，让你的微信小程序具有在线支付功能 - 简书 (jianshu.com)
    - 前提：主体为非个人账户，且绑定了商户
    - 流程图





- 流程与要用的接口【未完善】：
  - 发起支付并调起微信支付
    - data: {
    -
    - res结构：
  - 用户确认支付

- 取消订单：
  - data:{
    - order_id
    - }
  - success(res){}
    - res结构：
    - {
      - cancel_status(0 取消成功，1 取消失败)
      - msg(0 : '取消成功'，1: '取消失败')
    - }
- 确认收货
  - data:{
    - order_id
    - }
  - success(res){}
    - res结构：{
      - receive_status(0 收货成功，1 收货失败)
      - msg(0 : '收货成功'，1: '收货失败')

        - }
- 评价订单【对同一订单下所有商品做相同评价】
    - data:{
        - order_id,
        - score
        - }
    - success(res){}
        - res结构：{
            - rate_status(0 评价成功，1 评价失败)
            - msg(0 : '评价成功'，1: '评价失败')
        - }
- 查询收货地址
    - data:{
        - }
    - success(res){}
        - res结构：{
            - user_address【用/分割，便于和添加、修改功能一起实现；前端对地址解码并组织显示】
        - }
- 更新收货地址【修改添加删除共用统一接口】
    - data:{
        - user_address_new
        - }
    - success(res){}
        - res结构：{
            - rate_status(0 修改/添加/删除 成功，1 修改/添加/删除 失败)
            - msg(0 : '修改/添加/删除 成功'，1: '修改/添加/删除 失败')
        - }