苍穹外卖day6

1. httpClient

HttpClient 是Apache Jakarta Common 下的子项目，可以用来提供高效的、最新的、功能丰富的支持 HTTP 协议的客户端编程工具包，并且它支持 HTTP 协议最新的版本和建议。

```
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.13</version>
</dependency>
```

核心API：

- HttpClient
- HttpClients
- CloseableHttpClient
- HttpGet
- HttpPost

发送请求步骤：

- 创建HttpClient对象
- 创建Http请求对象
- 调用HttpClient的execute方法发送请求

`aliyun-sdk-oss`底层使用了`httpclient`，所以这里不用额外导入坐标。

启动SkyApplication，然后启动测试类HttpClientTest.java

sky-server/src/test/java/com/sky/test/HttpClientTest.java

```
1  package com.sky.test;
2
3  import com.alibaba.fastjson.JSONObject;
4  import org.apache.http.HttpEntity;
5  import org.apache.http.client.methods.CloseableHttpResponse;
6  import org.apache.http.client.methods.HttpGet;
7  import org.apache.http.client.methods.HttpPost;
8  import org.apache.http.entity.StringEntity;
9  import org.apache.http.impl.client.CloseableHttpClient;
10 import org.apache.http.impl.client.HttpClients;
11 import org.apache.http.util.EntityUtils;
12 import org.junit.jupiter.api.Test;
13 import org.springframework.boot.test.context.SpringBootTest;
14
15 import java.io.IOException;
16
17 @SpringBootTest
18 public class HttpClientTest {
19
20     /**
21      * 测试通过httpclient发送GET方式的请求
```

```
     */
    @Test
    public void testGet() throws Exception{
        //创建httpClient对象
        CloseableHttpClient httpClient= HttpClients.createDefault();

        //创建请求对象
        HttpGet httpGet=new
HttpGet("http://localhost:8080/user/shop/status");//需要启动SkyApplication

        //发送请求
        CloseableHttpResponse response= httpClient.execute(httpGet);

        //获取服务端返回的状态码
        int statusCode=response.getStatusLine().getStatusCode();
        System.out.println("服务器端返回的状态码"+statusCode);

        HttpEntity httpEntity = response.getEntity();
        String body = EntityUtils.toString(httpEntity);
        System.out.println("服务器端返回的数据为"+body);

        //关闭资源
        response.close();
        httpClient.close();
    }

    /**
     * 测试通过httpClient发送post方式的请求
     */
    @Test
    public void testPost() throws Exception{//java编程需要调用大量第三方接口的时
候，底层一般用HttpClient
        //创建httpclient对象
        CloseableHttpClient httpClient = HttpClients.createDefault();
        //创建请求对象
        HttpPost httpPost = new
HttpPost("http://localhost:8080/admin/employee/login");
        JSONObject jsonObject = new JSONObject();
        jsonObject.put("username", "admin");
        jsonObject.put("password", "123456");
        StringEntity stringEntity = new
StringEntity(jsonObject.toString());//对象转json字符串
        //指定请求编码方式
        stringEntity.setContentEncoding("utf-8");
        //数据格式
        stringEntity.setContentType("application/json");
        httpPost.setEntity(stringEntity);
        //发送请求
        CloseableHttpResponse response= httpClient.execute(httpPost);
        //解析返回结果
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("post请求响应码："+statusCode);

        HttpEntity httpEntity = response.getEntity();
        String body = EntityUtils.toString(httpEntity);
        System.out.println("post请求响应数据："+body);
```

```
        //关闭资源
        response.close();
        httpClient.close();
    }
}
```

```
服务器端返回的状态码200
服务器端返回的数据为{"code":1,"msg":null,"data":1}
```

```
post请求响应码：200
post请求响应数据：{"code":1,"msg":null,"data":{"id":1,"userName":"admin","name":"管理员","token":"eyJhbGciOiJIUzI1NiJ9.eyJlbXBJZCI6MSwiZXhwIjox
```

苍穹外卖在`sky-common/src/main/java/com/sky/utils/HttpClientUtil.java`工具类封装了httpClient项目

2. 微信小程序开发-入门案例

1. 小程序开发环境配置

访问https://mp.weixin.qq.com/，注册一个微信小程序账号

下载微信小程序开发工具，我的版本是`1.06.2503300`。

```
ID:  wx1111xxxxxx
密钥：40xxxxxxxxx
```

用开发工具创一个小程序项目，不使用云服务与模板

勾上不校验合法域名

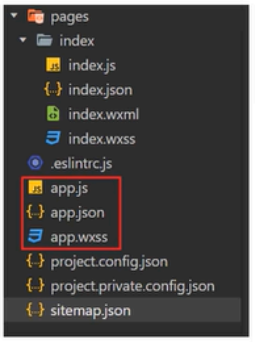

上传
版本管理
教育套件
详情
消息
基本信息
性能分析
本地设置
项目配置
调试基础库
3.8.5
灰度中
推送
该基础库支持微信客户端
iOS
8.0.57 及以上版本
Android
8.0.57 及以上版本
MacOS
暂不支持
Windows
暂不支持
将 JS 编译成 ES5
上传代码时样式自动补全
上传代码时自动压缩样式文件
上传代码时自动压缩脚本文件
上传代码时自动压缩wxml文件
上传时过滤无依赖文件
上传时进行代码保护
自动运行体验评分
不校验合法域名、web-view（业务域名）、TLS 版本以及 HTTPS 证书
预览及真机调试时主包、分包体积上限调整为4M
启用数据预拉取
启用代码自动热重载（不支持 json 文件）
懒注入占位组件调试
启用多核心编译

2. 小程序目录结构

- 了解小程序目录结构

小程序包含一个描述整体程序的 app 和多个描述各自页面的 page。
一个小程序主体部分由三个文件组成，必须放在项目的根目录，如下：

文件	必需	作用
app.js	是	小程序逻辑
app.json	是	小程序公共配置
app.wxss	否	小程序公共样式表



一个小程序页面由四个文件组成：

文件类型	必需	作用
js	是	页面逻辑
wxml	是	页面结构
json	否	页面配置
wxss	否	页面样式表

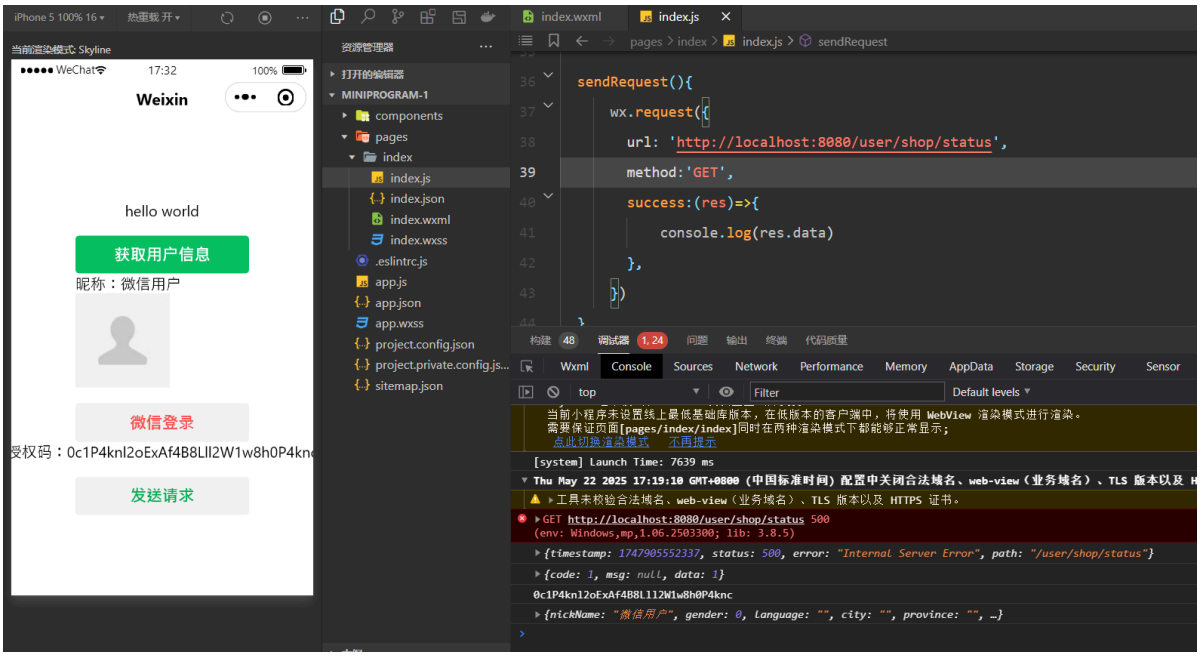
从基础库 2.21.2 开始支持

注意：**当小程序需要让用户完善个人资料时，可以通过微信提供的`头像昵称填写能力`完善，而不是`wx.getUserProfile`。**

这里留到后面前端部分解决。

启动skyapplication和redis。

`redis-server.exe redis.windows.conf`



index.wxml

```
<!--index.wxml-->
<navigation-bar title="Weixin" back="{{false}}" color="black" background="#FFF"></navigation-bar>
<scroll-view class="scrollarea" scroll-y type="list">
  <view class="container">
    <view>
      {{msg}}
    </view>
    <view>
        <button bind:tap="getUserInfo" type="primary">获取用户信息</button>
        昵称：{{nickName}}
        <image style="width:100px ;height: 100px;" src="{{url}}"></image>
    </view>
    <view>
        <button bind:tap="wxLogin" type="warn">微信登录</button>
        授权码：{{code}}
    </view>

    <view>
        <button bind:tap="sendRequest" type="default">发送请求</button>
    </view>
  </view>
</scroll-view>
```

index.js

```
// index.js
Page({
  data:{
    msg:'hello world',
    nickName:'',
```

```
    url:'',
    code:''
  },

  //获取微信用户的头像和昵称
  getUserInfo(){
    wx.getUserProfile({//自 2022 年 10 月 25 日 24 时后,这个方法不能获取用户头像昵称
      desc: '获取用户信息',
      success: (res)=>{
          console.log(res.userInfo);
          this.setData({
              nickName: res.userInfo.nickName,
              url:res.userInfo.avatarUrl
          })
      }
    })
  },

  //微信登录，获取微信用户的授权码
  wxLogin(){
      wx.login({
          success:(res)=>{
            console.log(res.code)
            this.setData({
                code:res.code
            })
          }
      })
  },

  sendRequest(){
      wx.request({
        url: 'http://localhost:8080/user/shop/status',
        method:'GET',
        success:(res)=>{
            console.log(res.data)
        },
      })
  }
})
```

3. 微信登录

1. 导入项目和测试基本接口

微信小程序开发文档：https://developers.weixin.qq.com/miniprogram/dev/framework/

下载postman。

打开redis和skyapplication

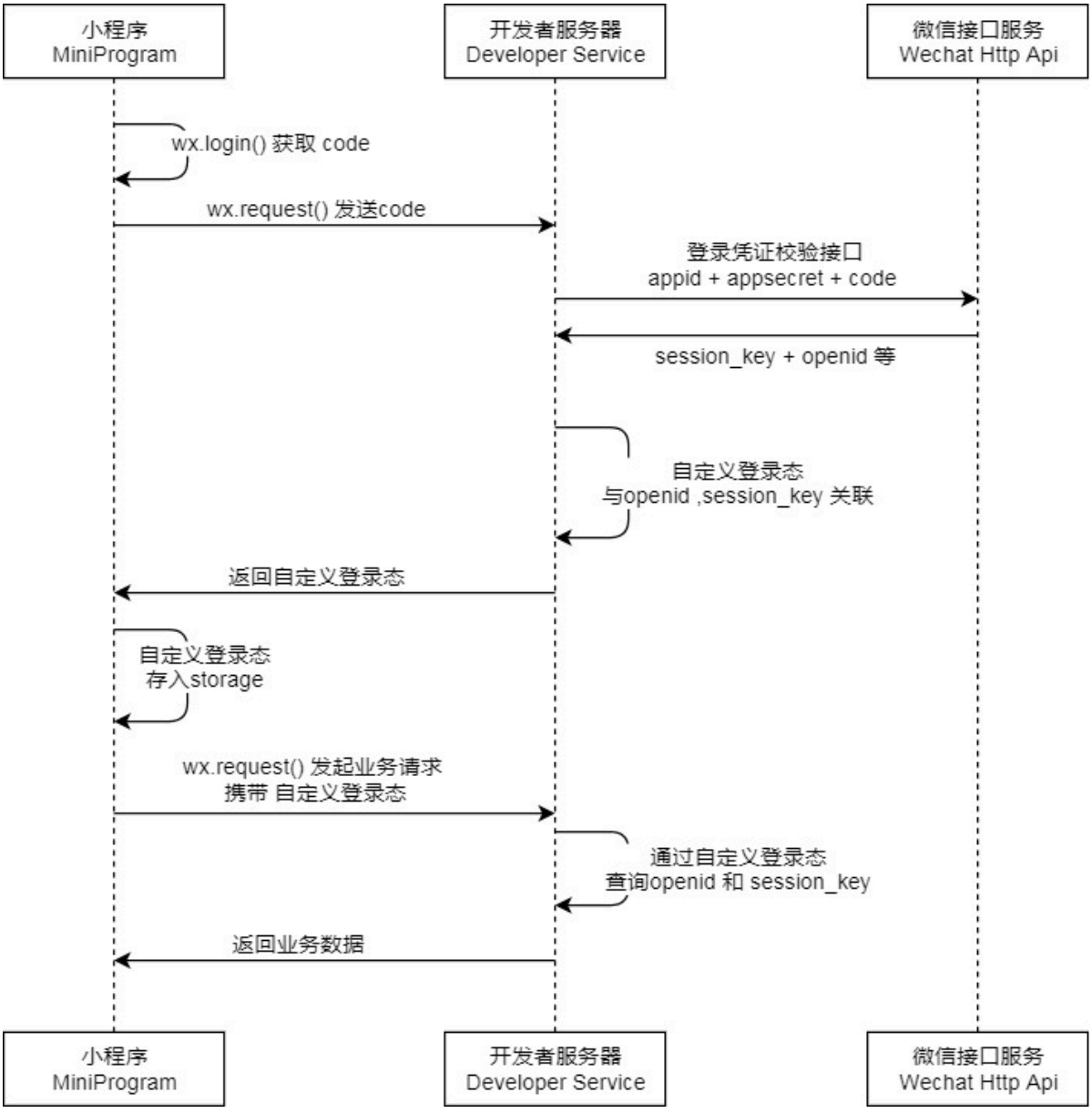
导入小程序代码，然后点击编译

C:\baidunetdiskdownload\资料\day06\微信小程序代码\mp-weixin

使用微信云开发，模式为小程序

\mp-weixin\common\vendor.js注意后端的端口号

```
var baseUrl = 'http://localhost:8080';//请求nginx，由nginx将请求转发到后端服务
```

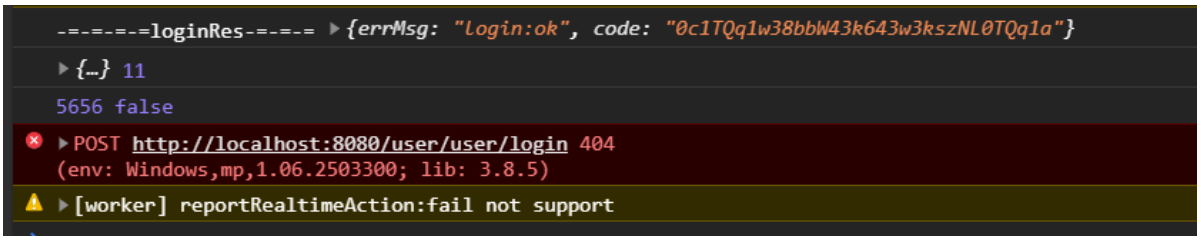


访问微信小程序开发文档

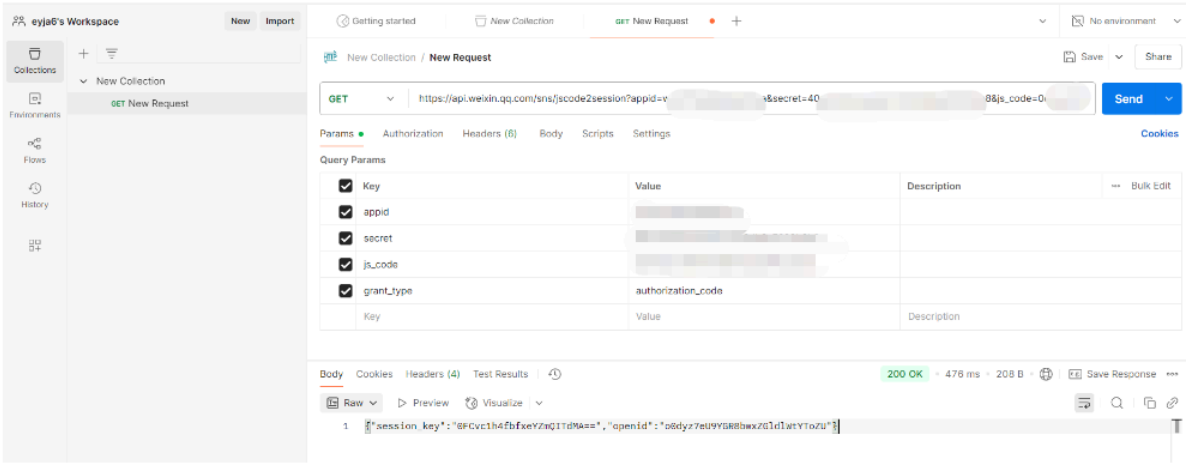
查看jscode2session调用的具体方法

```
GET https://api.weixin.qq.com/sns/jscode2session?
appid=APPID&secret=SECRET&js_code=JSCODE&grant_type=authorization_code
```

js_code填写微信开发工具的控制台输出的code，这个code过期时间为5分钟，并且只能用一次。



send一下，得到openid



2. 需求分析和设计

产品原型：

业务规则：

- 基于微信登录实现小程序的登录功能
- 如果是新用户需要自动完成注册

接口设计：

基本信息

Path: /user/user/login

Method: POST

接口描述:

请求参数

Headers

参数名称	参数值	是否必须	示例	备注
Content-Type	application/json	是		

Body

名称	类型	是否必须	默认值	备注	其他信息
code	string	必须		微信用户授权码	

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	integer	必须			format: int32
data	object	必须			
├─ id	integer	必须		用户id	format: int64
├─ openid	string	必须		微信openid	
├─ token	string	必须		jwt令牌	
msg	string	非必须			

数据库设计主要为user表。

3. 代码开发

application.yml

```yaml
sky:
  jwt:
    # 设置jwt签名加密时使用的密钥
    user-secret-key: itheima #为微信用户生成令牌的配置项
    # 设置jwt过期时间
    user-ttl: 7200000
    # 设置前端传递过来的令牌名称
    user-token-name: authentication
  wechat: #微信登录所需配置项
    appid: ${sky.wechat.appid}
    secret: ${sky.wechat.secret}
```

application-dev.yml

```yaml
sky:
  wechat:
    appid: wxxxxxx
    secret: 4xxxxxxxxxx
```

sky-server/src/main/java/com/sky/controller/user/UserController.java

```java
package com.sky.controller.user;

import com.sky.constant.JwtClaimsConstant;
import com.sky.dto.UserLoginDTO;
import com.sky.entity.User;
import com.sky.properties.JwtProperties;
import com.sky.result.Result;
import com.sky.service.UserService;
import com.sky.utils.JwtUtil;
import com.sky.vo.UserLoginVO;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.HashMap;
import java.util.Map;

@RestController
@RequestMapping("/user/user")
@Api(tags="C端用户相关接口")
@Slf4j
public class UserController {

    @Autowired
    private UserService userService;
    @Autowired
    private JwtProperties jwtProperties;

    /**
     * 微信登录
     * @return
     */
    @PostMapping("/login")
    @ApiOperation("微信登录")
    public Result<UserLoginVO> login(@RequestBody UserLoginDTO userLoginDTO)
{
        log.info("微信用户登录：{}",userLoginDTO.getCode());
```

```
        //调用微信登录
        User user=userService.wxLogin(userLoginDTO);
        //为微信用户生成jwt令牌
        Map<String, Object> claims=new HashMap<>();
        claims.put(JwtClaimsConstant.USER_ID,user.getId());
        String token=JwtUtil.createJWT(jwtProperties.getUserSecretKey(),
jwtProperties.getUserTtl(), claims);

        UserLoginVO userLoginVO=UserLoginVO.builder()
                .id(user.getId())
                .openid(user.getOpenid())
                .token(token)
                .build();

        return Result.success(userLoginVO);
    }
}
```

sky-server/src/main/java/com/sky/service/UserService.java

```
package com.sky.service;

import com.sky.dto.UserLoginDTO;
import com.sky.entity.User;

public interface UserService {

    /**
     * 微信登陆
     * @param userLoginDTO
     * @return
     */
    User wxLogin(UserLoginDTO userLoginDTO);
}

```

sky-server/src/main/java/com/sky/service/impl/UserServiceImpl.java

```
package com.sky.service.impl;

import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import com.sky.constant.MessageConstant;
import com.sky.dto.UserLoginDTO;
import com.sky.entity.User;
import com.sky.exception.LoginFailedException;
import com.sky.mapper.UserMapper;
import com.sky.properties.WeChatProperties;
import com.sky.service.UserService;
import com.sky.utils.HttpClientUtil;
import lombok.extern.slf4j.Slf4j;

```

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.time.LocalDateTime;
import java.util.HashMap;
import java.util.Map;

@Service
@Slf4j
public class UserServiceImpl implements UserService {

    @Autowired
    private WeChatProperties weChatProperties;//使用写好的配置属性类注入

    @Autowired
    private UserMapper userMapper;

    //微信服务接口地址
    public static final String
WX_LOGIN="https://api.weixin.qq.com/sns/jscode2session";

    public User wxLogin(UserLoginDTO userLoginDTO) {
        //调用封装方法来调用微信接口服务，获取当前微信用户的openid
        String openid=getOpenid(userLoginDTO.getCode());

        //判断openid是否为空，空则登录失败抛异常
        if (openid==null){
            throw new LoginFailedException(MessageConstant.LOGIN_FAILED);
        }

        //判断是否为新用户
        User user = userMapper.getByOpenid(openid);

        //新用户自动完成注册
        if (user==null){
            user = User.builder()
                    .openid(openid)
                    .createTime(LocalDateTime.now())
                    .build();
            userMapper.insert(user);
        }




        //返回用户对象
        return user;
    }

    /**
     * 调用微信接口服务，获取微信用户openid
     * @param code
     * @return
     */
    private String getOpenid(String code) {
        //调用微信接口服务，获取当前微信用户的openid
        Map<String, String> map=new HashMap<>();
```

```
        map.put("appid", weChatProperties.getAppid());
        map.put("secret", weChatProperties.getSecret());
        map.put("js_code",code);
        map.put("grant_type", "authorization_code");

        String json=HttpClientUtil.doGet(WX_LOGIN,map);
        JSONObject jsonObject = JSON.parseObject(json);
        String openid = jsonObject.getString("openid");
        return openid;
    }
}

```

sky-server/src/main/java/com/sky/mapper/UserMapper.java

```
package com.sky.mapper;

import com.sky.entity.User;
import org.apache.ibatis.annotations.Mapper;
import org.apache.ibatis.annotations.Select;

@Mapper
public interface UserMapper {

    /**
     * 根据openid查询用户
     * @param openid
     * @return
     */
    @Select("select * from user where openid = #{openid}")
    User getByOpenid(String openid);

    /**
     * 插入数据
     * @param user
     */
    void insert(User user);
}

```

sky-server/src/main/resources/mapper/UserMapper.xml

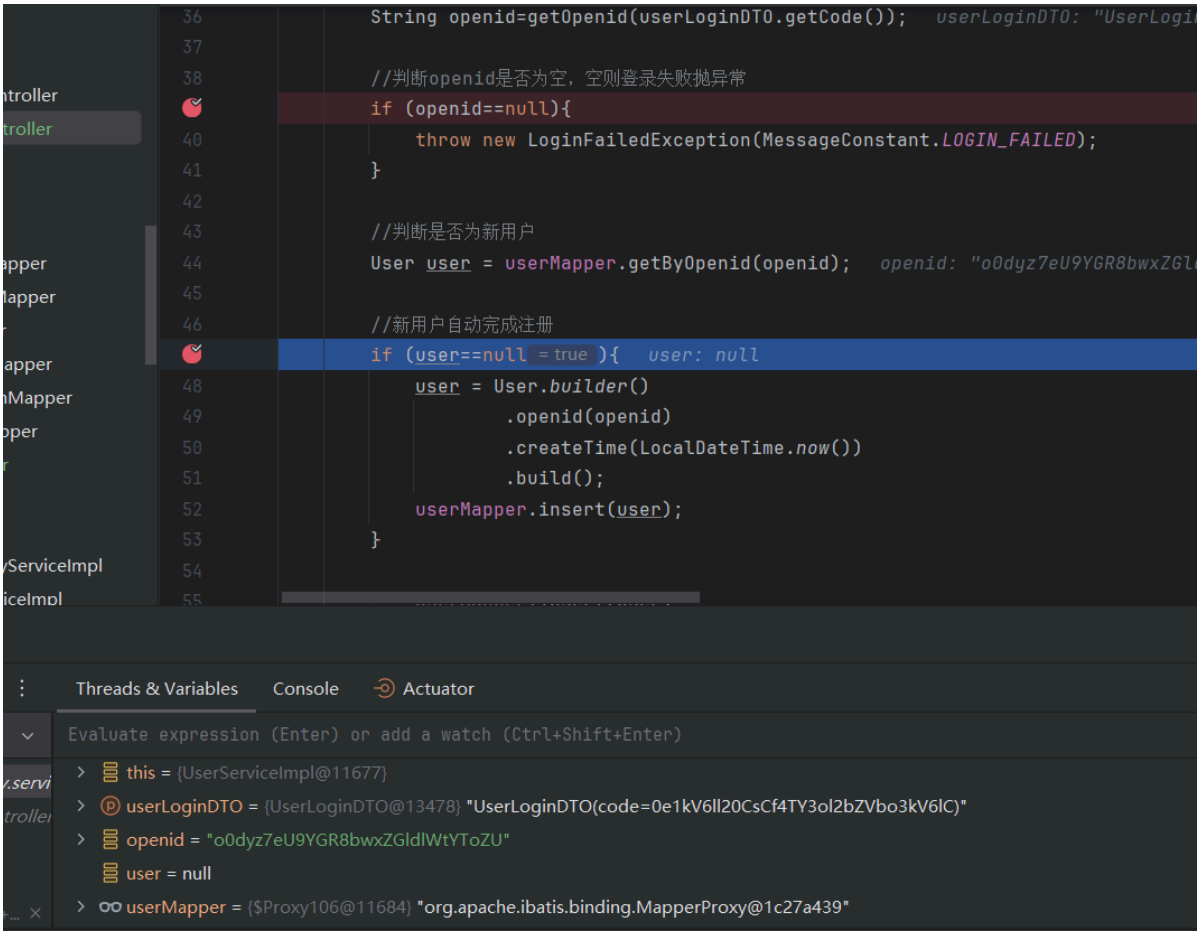
```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.sky.mapper.UserMapper">


    <insert id="insert" useGeneratedKeys="true" keyProperty="id">
        insert into user
(openid,name,phone,sex,id_number,avatar,create_time)
        values (#{openid},#{name},#{phone},#{sex},#{idNumber},#{avatar},#
{createTime})
    </insert>
</mapper>

```

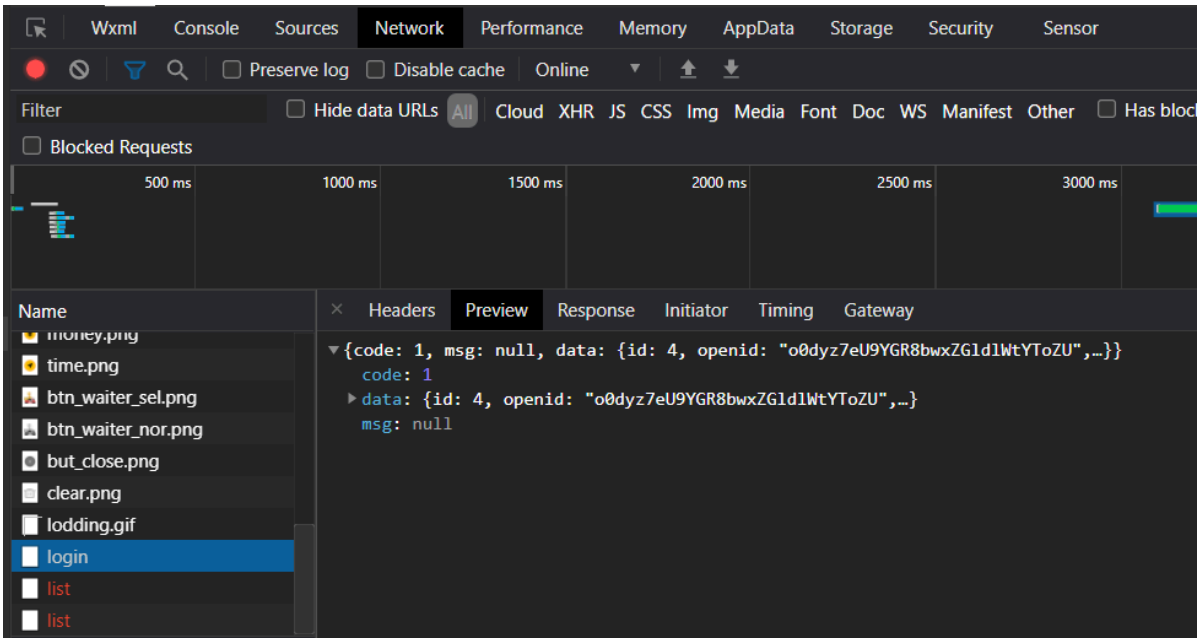
4. 功能测试

mvn clean

mvn compile

启动skyapplication，重新进入小程序





新建拦截器sky-server/src/main/java/com/sky/interceptor/JwtTokenUserInterceptor.java

```java
package com.sky.interceptor;

import com.sky.constant.JwtClaimsConstant;
import com.sky.context.BaseContext;
import com.sky.properties.JwtProperties;
import com.sky.utils.JwtUtil;
import io.jsonwebtoken.Claims;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.springframework.web.method.HandlerMethod;
import org.springframework.web.servlet.HandlerInterceptor;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * jwt令牌校验的拦截器
 */
@Component
@Slf4j
public class JwtTokenUserInterceptor implements HandlerInterceptor {

    @Autowired
    private JwtProperties jwtProperties;

    /**
     * 校验jwt
     *
     * @param request
     * @param response
     * @param handler
     * @return
     * @throws Exception
```

```
     */
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {

        System.out.println("当前线程的id值：" + Thread.currentThread().getId());

        //判断当前拦截到的是Controller的方法还是其他资源
        if (!(handler instanceof HandlerMethod)) {
            //当前拦截到的不是动态方法，直接放行
            return true;
        }

        //1、从请求头中获取令牌
        String token = request.getHeader(jwtProperties.getUserTokenName());

        //2、校验令牌
        try {
            log.info("jwt校验:{}", token);
            Claims claims = JwtUtil.parseJWT(jwtProperties.getUserSecretKey(), token);
            Long userId = Long.valueOf(claims.get(JwtClaimsConstant.USER_ID).toString());
            log.info("当前用户id：{}", userId);
            BaseContext.setCurrentId(userId);//利用了ThreadLocal
            //3、通过，放行
            return true;
        } catch (Exception ex) {
            //4、不通过，响应401状态码
            response.setStatus(401);
            return false;
        }
    }
}
```

在WebMvcConfiguration.java注册拦截器，修改添加以下部分

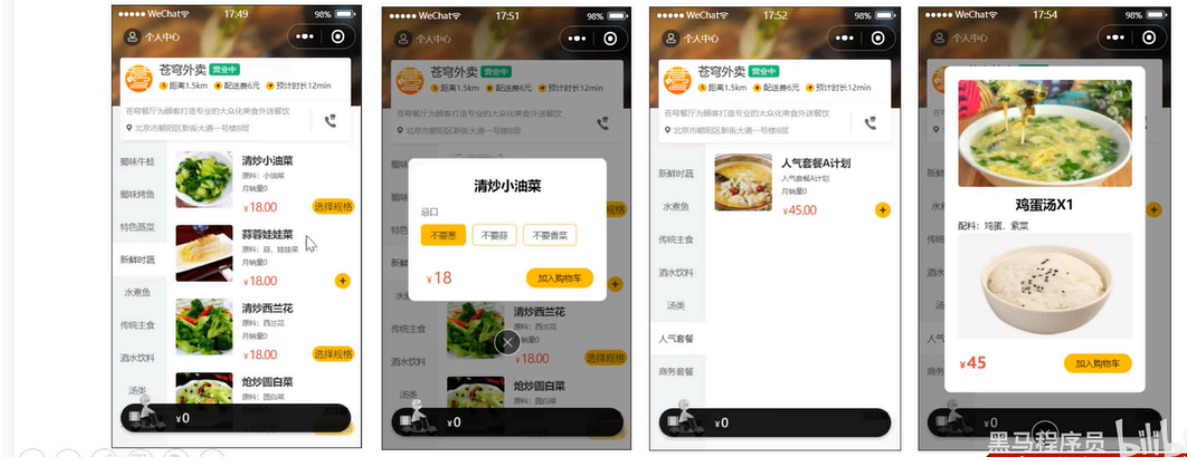
```
    @Autowired
    private JwtTokenUserInterceptor jwtTokenUserInterceptor;

    protected void addInterceptors(InterceptorRegistry registry) {
        log.info("开始注册自定义拦截器...");
        registry.addInterceptor(jwtTokenAdminInterceptor)
                .addPathPatterns("/admin/**")
                .excludePathPatterns("/admin/employee/login");
        registry.addInterceptor(jwtTokenUserInterceptor)
                .addPathPatterns("/user/**")
                .excludePathPatterns("/user/user/login")
                .excludePathPatterns("/user/shop/status");//登录之前不用校验令牌
    }
```

4. 导入商品浏览代码

1. 需求分析和接口设计

需求分析和设计

产品原型：



接口设计：

- 查询分类
- 根据分类id查询菜品
- 根据分类id查询套餐
- 根据套餐id查询包含的菜品

需求分析和设计

- 查询分类 接口

基本信息

Path: /user/category/list

Method: GET

接口描述:

请求参数

Query

参数名称	是否必须	示例	备注
type	否	1	分类类型：1 菜品分类 2 套餐分类

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	integer	必须			format: int32
data	object []	必须			item 类型: object
├─ createTime	string	非必须			format: date-time
├─ createUser	integer	非必须			format: int64
├─ id	integer	必须			format: int64
├─ name	string	必须			
├─ sort	integer	非必须			format: int32
├─ status	integer	非必须			format: int32
├─ type	integer	非必须			format: int32
├─ updateTime	string	非必须			format: date-time
├─ updateUser	integer	非必须			format: int64
msg	string	非必须			

需求分析和设计

- 根据分类id查询菜品 接口

基本信息

Path: /user/dish/list

Method: GET

接口描述:

请求参数

Query

参数名称	是否必须	示例	备注
categoryId	是		分类id

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	integer	必须			format: int32
data	object []	必须			item 类型: object
├─ categoryId	integer	必须			format: int64
├─ categoryName	string	非必须			
├─ description	string	非必须			
├─ flavors	object []	非必须			item 类型: object
├─ dishId	integer	非必须			format: int64
├─ id	integer	非必须			format: int64
├─ name	string	非必须			
├─ value	string	非必须			
├─ id	integer	必须			format: int64
├─ image	string	必须			
├─ name	string	必须			
├─ price	number	必须			
├─ status	integer	非必须			format: int32
├─ updateTime	string	非必须			format: date-time
msg	string	非必须			

需求分析和设计

- 根据分类id查询套餐 接口

基本信息

Path: /user/setmeal/list

Method: GET

接口描述:

请求参数

Query

参数名称	是否必须	示例	备注
categoryId	是		分类id

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	integer	必须			format: int32
data	object []	必须			item 类型: object
├─ categoryId	integer	必须			format: int64
├─ createTime	string	非必须			format: date-time
├─ createUser	integer	非必须			format: int64
├─ description	string	非必须			
├─ id	integer	必须			format: int64
├─ image	string	必须			
├─ name	string	必须			
├─ price	number	必须			
├─ status	integer	非必须			format: int32
├─ updateTime	string	非必须			format: date-time
├─ updateUser	integer	非必须			format: int64
msg	string	非必须			

2. 导入代码

打开`C:\baidunetdiskdownload\资料\day06\代码导入\商品浏览`文件夹

CategoryController.java拷贝到`sky-server/src/main/java/com/sky/controller/user/CategoryController.java`

DishController.java拷贝到`sky-server/src/main/java/com/sky/controller/user/DishController.java`

SetmealController.java拷贝到`sky-server/src/main/java/com/sky/controller/user/SetmealController.java`

idea项目的DishService.java添加

```
    /**
     * 条件查询菜品和口味
     * @param dish
     * @return
     */
    List<DishVO> listWithFlavor(Dish dish);
```

DishServiceImpl.java添加

```java
    /**
     * 条件查询菜品和口味
     * @param dish
     * @return
     */
    public List<DishVO> listWithFlavor(Dish dish) {
        List<Dish> dishList = dishMapper.list(dish);

        List<DishVO> dishVOList = new ArrayList<>();

        for (Dish d : dishList) {
            DishVO dishVO = new DishVO();
            BeanUtils.copyProperties(d,dishVO);

            //根据菜品id查询对应的口味
            List<DishFlavor> flavors =
dishFlavorMapper.getByDishId(d.getId());

            dishVO.setFlavors(flavors);
            dishVOList.add(dishVO);
        }

        return dishVOList;
    }
```

SetmealService.java

```java
    /**
     * 条件查询
     * @param setmeal
     * @return
     */
    List<Setmeal> list(Setmeal setmeal);

    /**
     * 根据id查询菜品选项
     * @param id
     * @return
     */
    List<DishItemVO> getDishItemById(Long id);
```

SetmealServiceImpl.java

```java
    /**
     * 条件查询
     * @param setmeal
     * @return
     */
    public List<Setmeal> list(Setmeal setmeal) {
        List<Setmeal> list = setmealMapper.list(setmeal);
        return list;
```

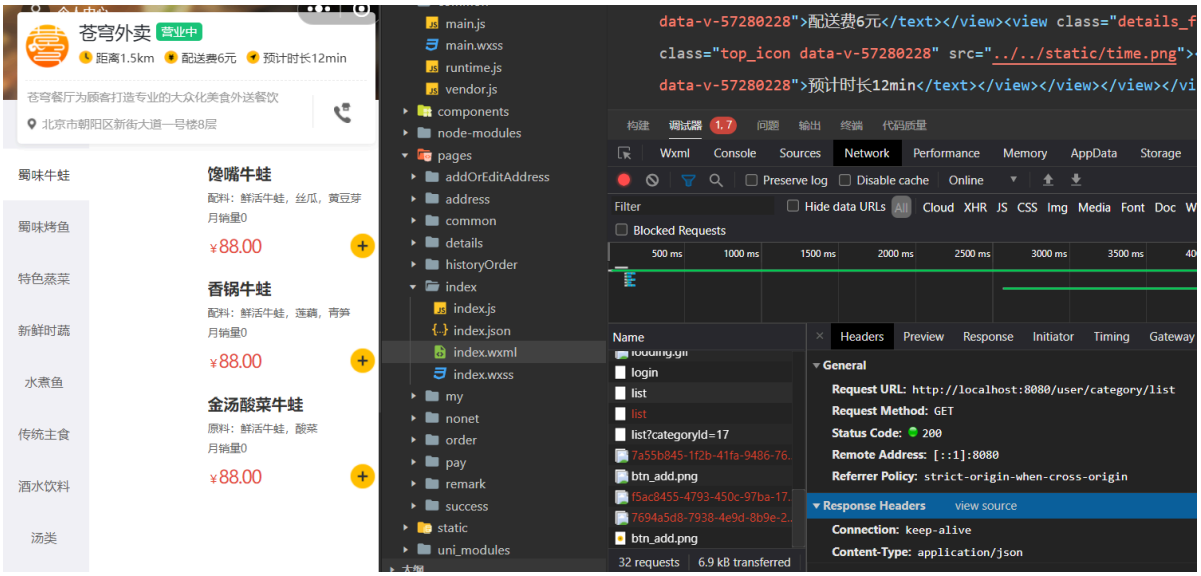
```
    }

    /**
     * 根据id查询菜品选项
     * @param id
     * @return
     */
    public List<DishItemVO> getDishItemById(Long id) {
        return setmealMapper.getDishItemBySetmealId(id);
    }
```

SetmealMapper.java

```
    /**
     * 动态条件查询套餐
     * @param setmeal
     * @return
     */
    List<Setmeal> list(Setmeal setmeal);

    /**
     * 根据套餐id查询菜品选项
     * @param setmealId
     * @return
     */
    @Select("select sd.name, sd.copies, d.image, d.description " +
            "from setmeal_dish sd left join dish d on sd.dish_id = d.id " +
            "where sd.setmeal_id = #{setmealId}")
    List<DishItemVO> getDishItemBySetmealId(Long setmealId);
```

SetmealMapper.xml

```
    <select id="list" parameterType="Setmeal" resultType="Setmeal">
        select * from setmeal
        <where>
            <if test="name != null">
                and name like concat('%',#{name},'%')
            </if>
            <if test="categoryId != null">
                and category_id = #{categoryId}
            </if>
            <if test="status != null">
                and status = #{status}
            </if>
        </where>
    </select>
```


苍穹外卖
营业中
距离1.5km
配送费6元
预计时长12min
苍穹餐厅为顾客打造专业的大众化美食外送餐饮
北京市朝阳区新街大道一号楼8层
蜀味牛蛙
蜀味烤鱼
特色蒸菜
新鲜时蔬
水煮鱼
传统主食
酒水饮料
汤类
馋嘴牛蛙
配料：鲜活牛蛙，丝瓜，黄豆芽
月销量0
¥88.00
香锅牛蛙
配料：鲜活牛蛙，莲藕，青笋
月销量0
¥88.00
金汤酸菜牛蛙
原料：鲜活牛蛙，酸菜
月销量0
¥88.00
main.js
main.wxss
runtime.js
vendor.js
components
node-modules
pages
addOrEditAddress
address
common
details
historyOrder
index
index.js
index.json
index.wxml
index.wxss
my
nonet
order
pay
remark
success
static
uni_modules
data-v-57280228">配送费6元</text></view><view class="details_f
class="top_icon data-v-57280228" src="../../static/time.png">
data-v-57280228">预计时长12min</text></view></view></view></vi
构建
调试器 1, 7
问题
输出
终端
代码质量
Wxml
Console
Sources
Network
Performance
Memory
AppData
Storage
Preserve log
Disable cache
Online
Filter
Hide data URLs
All
Cloud
XHR
JS
CSS
Img
Media
Font
Doc
Blocked Requests
500 ms
1000 ms
1500 ms
2000 ms
2500 ms
3000 ms
3500 ms
Name
login
list
list
list?categoryId=17
btn_add.png
btn_add.png
32 requests
6.9 kB transferred
Headers
Preview
Response
Initiator
Timing
Gateway
General
Request URL: http://localhost:8080/user/category/list
Request Method: GET
Status Code: 200
Remote Address: [::1]:8080
Referrer Policy: strict-origin-when-cross-origin
Response Headers
view source
Connection: keep-alive
Content-Type: application/json