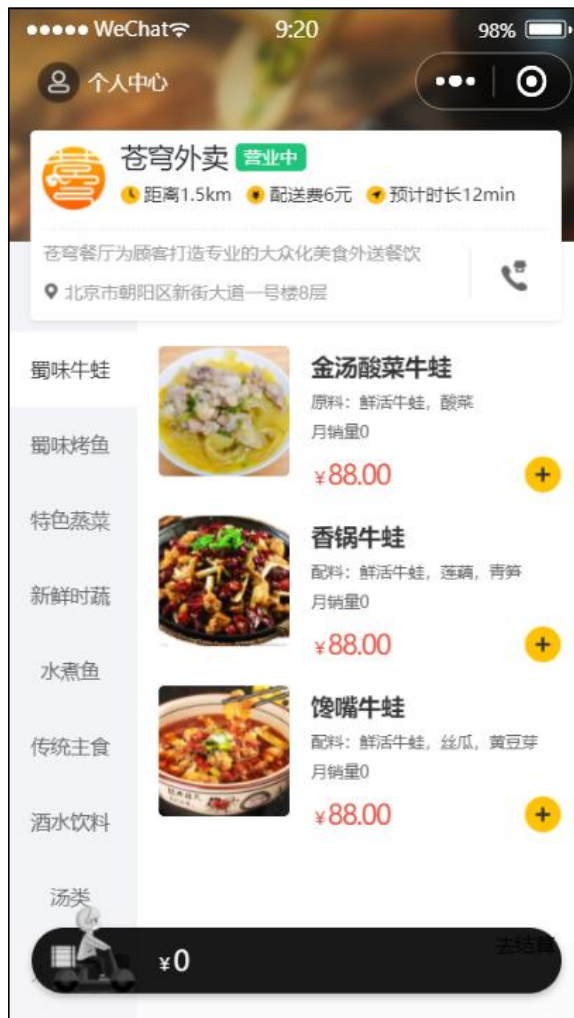


微信登录、商品浏览



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌





目录

Contents

- ◆ HttpClient
- ◆ 微信小程序开发
- ◆ 微信登录
- ◆ 导入商品浏览功能代码

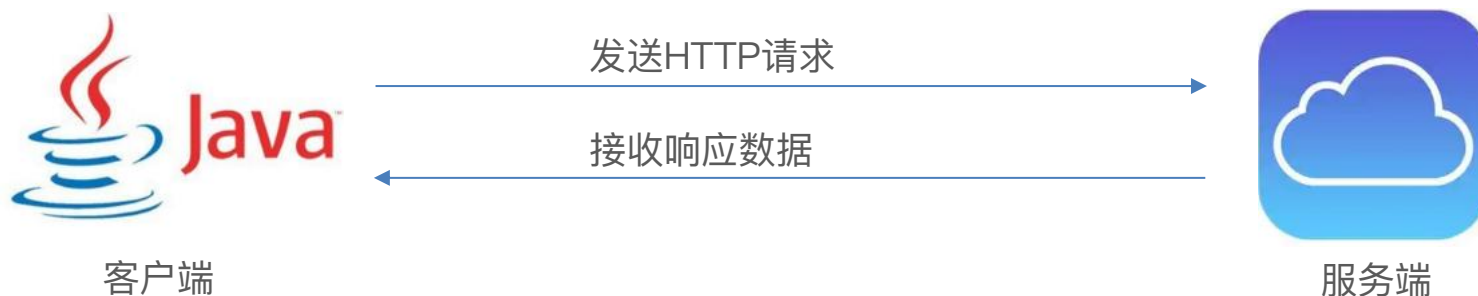


HttpClient

- 介绍
- 入门案例

HttpClient介绍 — 作用

HttpClient是Apache的一个子项目，是高效的、功能丰富的支持HTTP协议的客户端编程工具包。



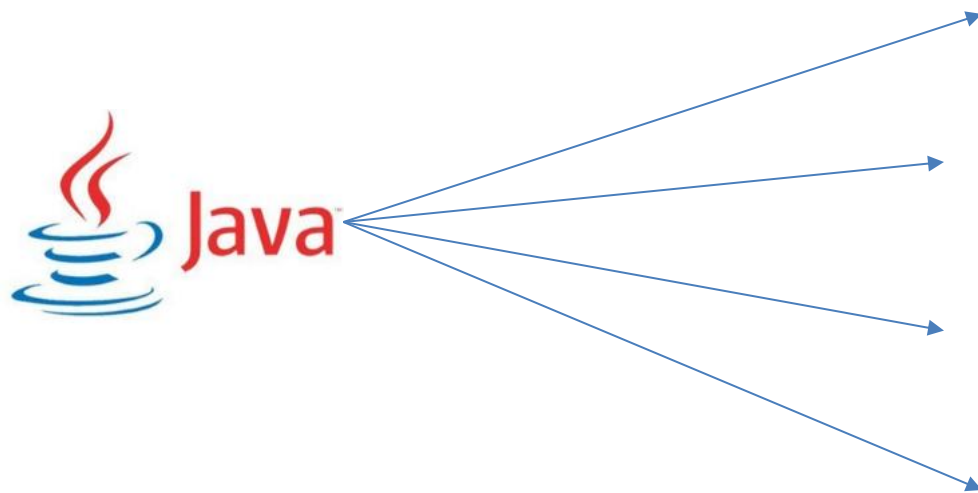
HttpClient作用：

- 发送HTTP请求
- 接收响应数据



为什么要在Java程序中发送Http请求?

HttpClient介绍 — 应用场景



微信服务



地图服务



短信服务



天气预报服务



HttpClient介绍 — 效果展示

百度地图地理编码服务接口：

<https://api.map.baidu.com/geocoding/v3/?address=北京市海淀区上地十街10号&output=json&ak=UEBQm9c3KZ5LrsO2C2qsOAs1eSdLvlzM>



客户端

发送HTTP请求

接收响应数据



百度地图服务

通过浏览器请求服务的步骤：

1. 构造请求地址和参数
2. 发送请求
3. 接收数据

介绍

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
- HttpClient
- CloseableHttpClient
- HttpGet
- HttpPost

发送请求步骤:

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



HttpClient

- 介绍
- 入门案例

入门案例

GET方式请求：

```
//http客户端对象，可以发送http请求
CloseableHttpClient httpClient = HttpClients.createDefault();

//构造Get方式请求
HttpGet httpGet = new HttpGet("http://localhost:8080/user/shop/status");

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

//http响应码
int statusCode = response.getStatusLine().getStatusCode();

//http响应体
HttpEntity entity = response.getEntity();
//将响应体转为String字符串
String body = EntityUtils.toString(entity);
System.out.println(body);

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

入门案例

POST方式请求:

```
CloseableHttpClient httpClient = HttpClients.createDefault();
//Post方式请求
HttpPost httpPost = new HttpPost("http://localhost:8080/admin/employee/login");

//构造json数据
JSONObject jsonObject = new JSONObject();
jsonObject.put("username","admin");
jsonObject.put("password", "123456");

//构造请求体
StringEntity stringEntity = new StringEntity(jsonObject.toString());
//设置请求编码
stringEntity.setContentEncoding("utf-8");
//设置数据类型
stringEntity.setContentType("application/json");
//设置当前Post请求的请求体
httpPost.setEntity(stringEntity);

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

//http响应码
int statusCode = response.getStatusLine().getStatusCode();

//http响应体
HttpEntity entity = response.getEntity();
//将响应体转为String字符串
String body = EntityUtils.toString(entity);
System.out.println(body);

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



目录

Contents

- ◆ HttpClient
- ◆ 微信小程序开发
- ◆ 微信登录
- ◆ 导入商品浏览功能代码



微信小程序开发

- 介绍
- 准备工作
- 入门案例

介绍



小程序

一种新的开放能力，可以在微信内被便捷地获取和传播，同时具有出色的使用体验。

https://mp.weixin.qq.com/cgi-bin/wx?token=&lang=zh_CN

介绍

开放注册范围



个人



企业



政府



媒体



其他组织

介绍

开发支持

提供一系列工具帮助开发者快速接入并完成小程序开发。



开发文档



开发者工具



设计指南



小程序体验DEMO

介绍

接入流程

- 1 注册**
在微信公众平台注册小程序，完成注册后可以同步进行信息完善和开发。
⋮
- 2 小程序信息完善**
填写小程序基本信息，包括名称、头像、介绍及服务范围等。
⋮
- 3 开发小程序**
完成小程序开发者绑定、开发信息配置后，开发者可下载开发者工具、参考开发文档进行小程序的开发和调试。
⋮
- 4 提交审核和发布**
完成小程序开发后，提交代码至微信团队审核，审核通过后即可发布（公测期间不能发布）。



微信小程序开发

- 介绍
- 准备工作
- 入门案例

准备工作

开发微信小程序之前需要做如下准备工作：

- 注册小程序
- 完善小程序信息
- 下载开发者工具

准备工作 — 注册小程序

注册地址：

<https://mp.weixin.qq.com/wxopen/waregister?action=step1>

小程序注册

咨询客服

① 帐号信息 — ② 邮箱激活 — ③ 信息登记

每个邮箱仅能申请一个小程序

邮箱


作为登录帐号，请填写未被微信公众平台注册，未被微信开放平台注册，未被个人微信号绑定的邮箱

密码

字母、数字或者英文符号，最短8位，区分大小写

确认密码

请再次输入密码

验证码  换一张

☐ 你已阅读并同意《微信公众平台服务协议》及《微信小程序平台服务条款》

注册

已有微信小程序？立即登录

创建测试号，免注册快速体验小程序开发。立即申请

准备工作 — 完善小程序信息

登录小程序后台: <https://mp.weixin.qq.com/>

登录

邮箱/微信号

密码

☐ 记住帐号

找回帐号或密码

登录

登录

使用帐号登录



微信扫一扫，选择该微信下的
公众平台帐号登录

准备工作 — 完善小程序信息

完善小程序信息、小程序类目



The screenshot displays the '小程序发布流程' (Mini Program Release Process) in the WeChat developer console. The first step, '1 小程序信息' (1 Mini Program Information), is highlighted with a red box and marked as '已完成' (Completed). This step includes two sub-tasks: '小程序信息' (Mini Program Information) and '小程序类目' (Mini Program Category), both also marked as '已完成'. The second step, '2 小程序开发与管理' (2 Mini Program Development and Management), is currently active and divided into two columns: '自己开发' (Self-development) and '找服务商开发' (Find service provider for development). The '自己开发' column lists tasks like downloading tools, adding developers, configuring servers, and reading help documents. The '找服务商开发' column lists benefits such as rich services, time-saving, and reliability.

步骤	任务	状态
1 小程序信息	小程序信息	已完成
	小程序类目	已完成
2 小程序开发与管理	自己开发	进行中
	找服务商开发	进行中

自己开发

任务	描述
开发工具	下载开发者工具进行代码的开发和上传：普通小程序开发者工具、小游戏开发者工具。
添加开发者	添加开发者，进行代码上传。
配置服务器	在开发设置页面查看AppID和AppSecret，配置服务器域名，或使用微信云开发或微信云托管，免配置服务器。
帮助文档	可以阅读入门介绍（普通小程序 小游戏）、开发文档（普通小程序 小游戏）、设计规范和运营规范。

找服务商开发

优势	描述
服务丰富	提供多行业、多功能的小程序开发服务。
省时省心	从小程序认证到发布上线，全流程支持。
安全可靠	可免费试用后再支付、在线客服响应、官方售后保障。

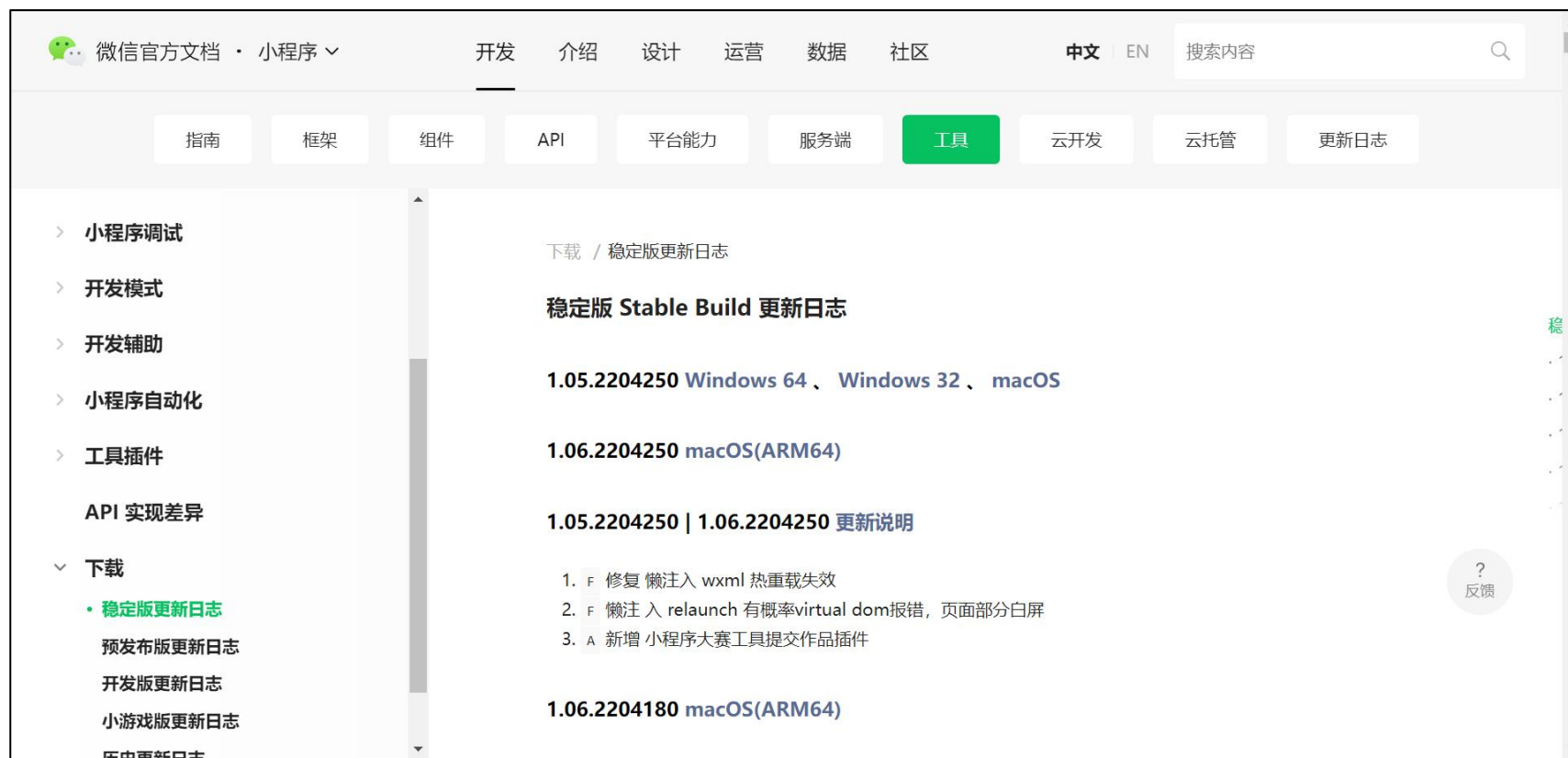
准备工作 — 完善小程序信息

查看小程序的 AppID



准备工作 — 下载开发者工具

下载地址: <https://developers.weixin.qq.com/miniprogram/dev/devtools/stable.html>



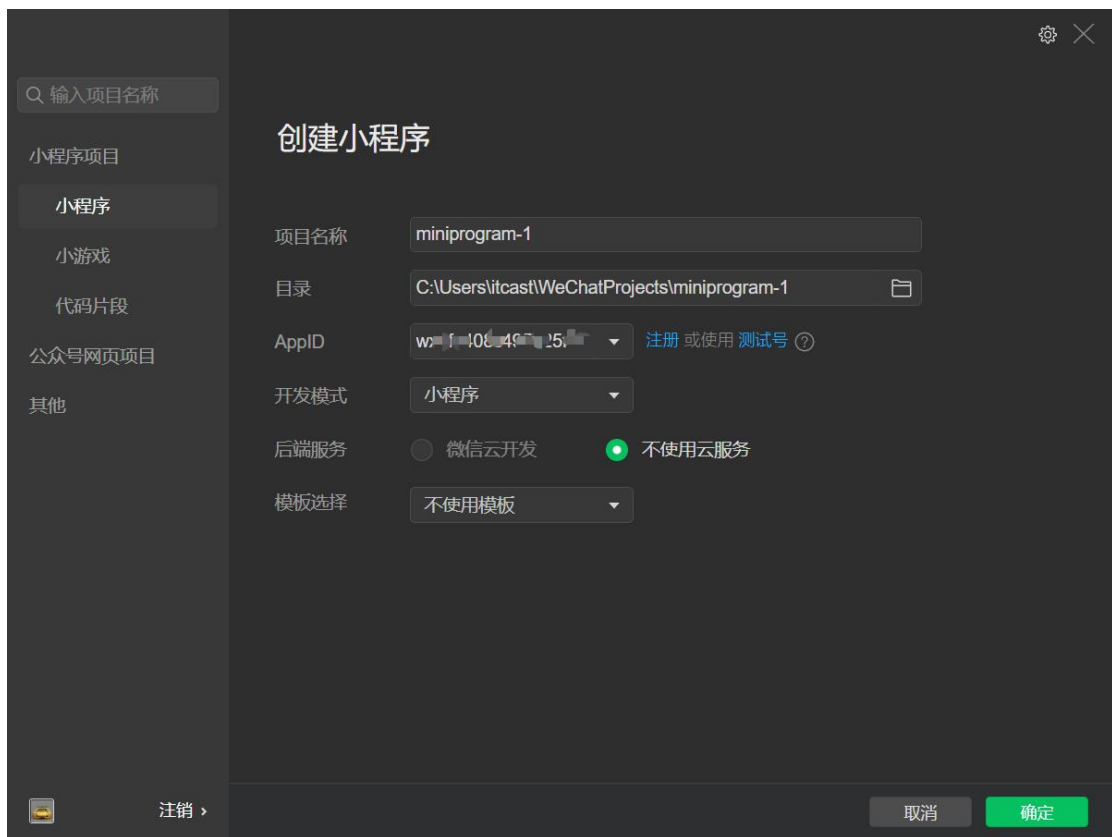
准备工作 — 下载开发者工具

扫描登录开发者工具



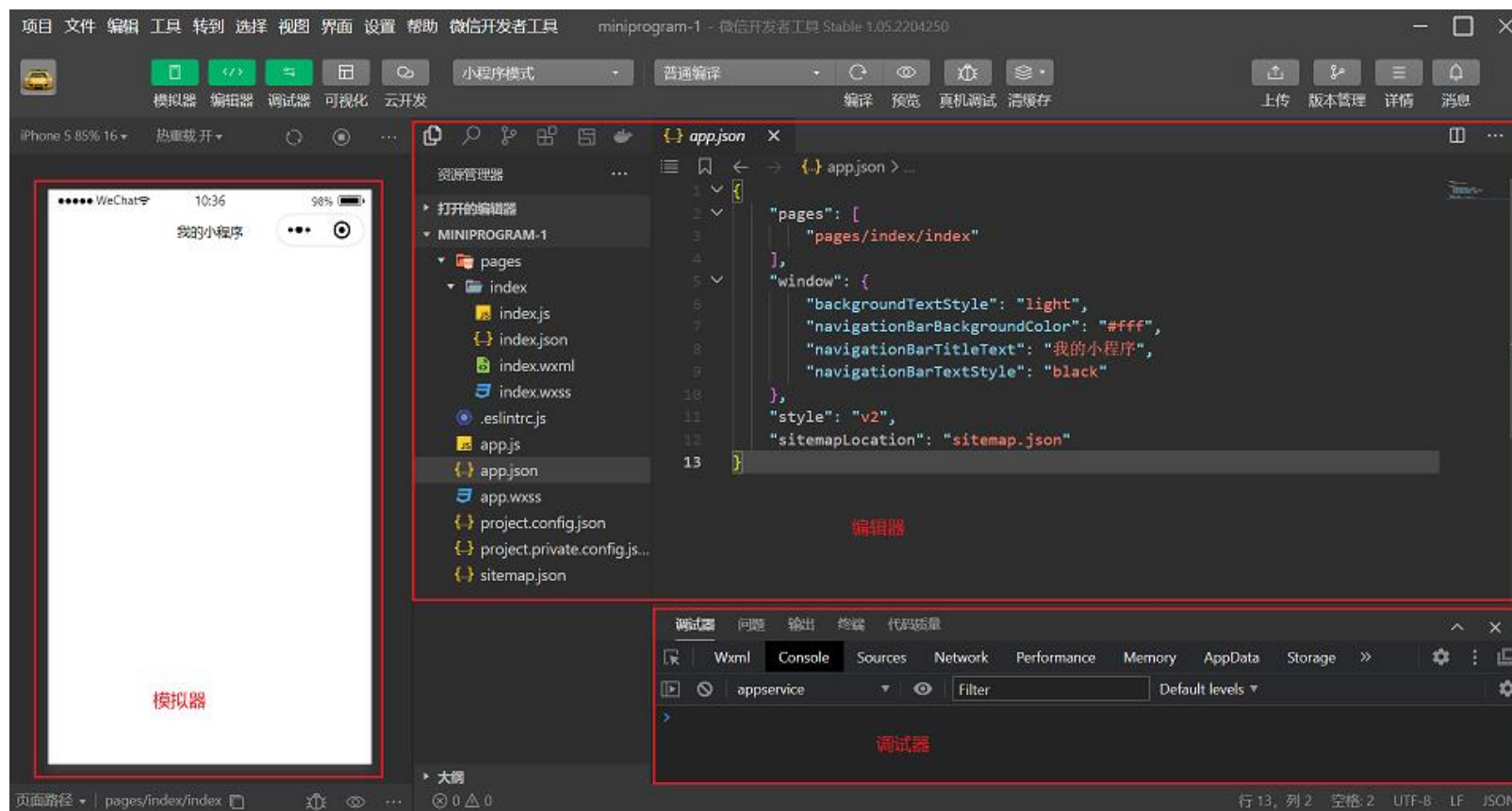
准备工作 — 下载开发者工具

创建小程序项目



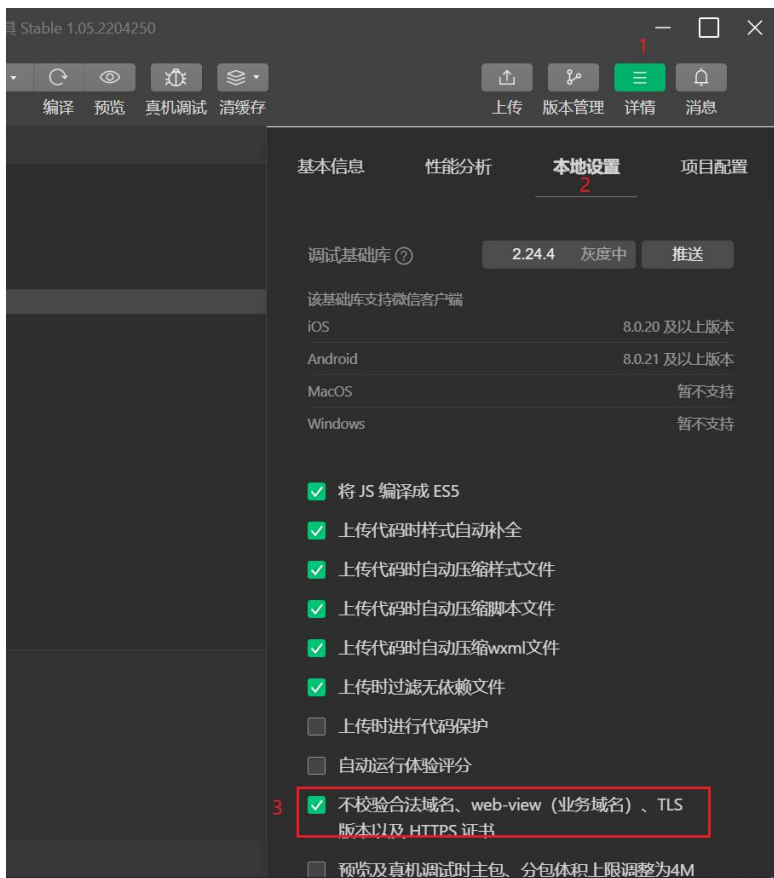
准备工作 — 下载开发者工具

熟悉开发者工具布局



准备工作 — 下载开发者工具

设置不校验合法域名





微信小程序开发

- 介绍
- 准备工作
- 入门案例

入门案例

操作步骤:

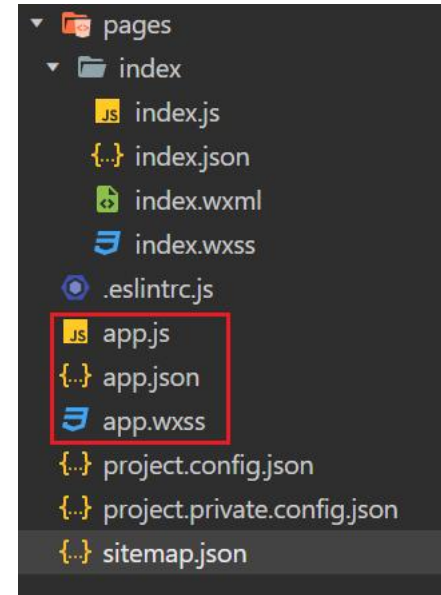
- 了解小程序目录结构
- 编写小程序代码
- 编译小程序

入门案例

- 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

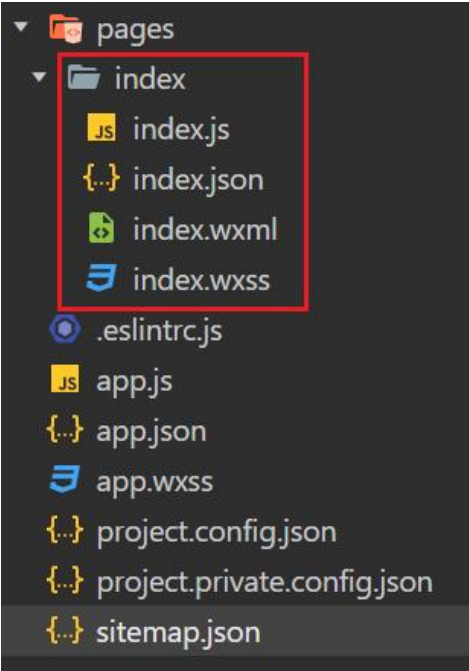


入门案例

- 了解小程序目录结构

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



入门案例

- 编写小程序代码

```
<view class="container">
  <view>{{msg}}</view>

  <view>
    <button type="default" bindtap="getUserInfo">获取用户信息</button>
    <image style="width: 100px;height: 100px;" src="{{avatarUrl}}"></image>
    {{nickName}}
  </view>

  <view>
    <button type="primary" bindtap="wxlogin">微信登录</button>
    授权码: {{code}}
  </view>

  <view>
    <button type="warn" bindtap="sendRequest">发送请求</button>
    响应结果: {{result}}
  </view>
</view>
```

index.wxml

入门案例

- 编写小程序代码

```
Page({
  data: {
    msg: 'hello world',
    avatarUrl: '',
    nickName: '',
    code: '',
    result: ''
  },

  getUserInfo: function() {
    wx.getUserProfile({
      desc: '获取用户信息',
      success: (res) => {
        console.log(res)
        this.setData({
          avatarUrl: res.userInfo.avatarUrl,
          nickName: res.userInfo.nickName
        })
      }
    })
  }
})
```

index.js

入门案例

- 编写小程序代码

```
wxlogin:function(){
  wx.login({
    success: (res) => {
      console.log("授权码: "+res.code)
      this.setData({
        code:res.code
      })
    }
  })
},

sendRequest:function(){
  wx.request({
    url: 'http://localhost:8080/user/shop/status',
    method:'GET',
    success:(res) => {
      console.log("响应结果: " + res.data.data)
      this.setData({
        result:res.data.data
      })
    }
  })
}
```

index.js

入门案例

- 编译小程序





目录

Contents

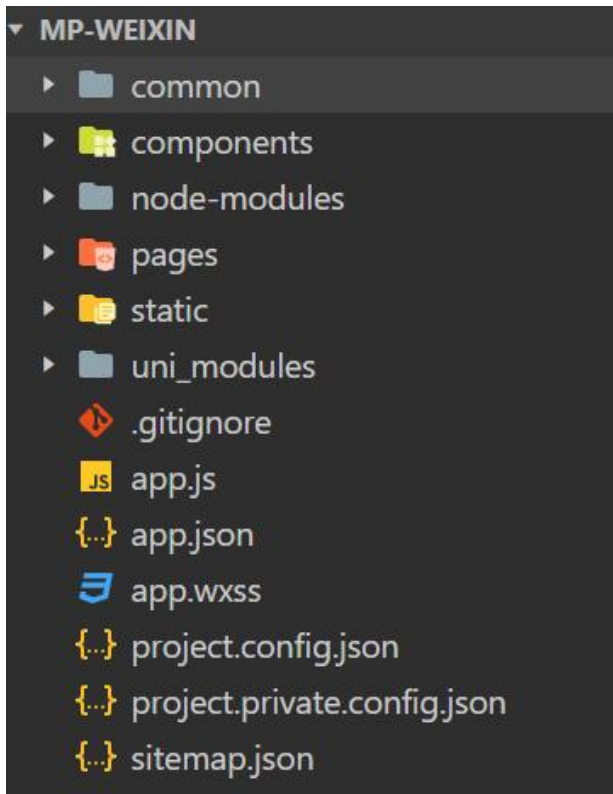
- ◆ HttpClient
- ◆ 微信小程序开发
- ◆ 微信登录
- ◆ 导入商品浏览功能代码



微信登录

- 导入小程序代码
- 微信登录流程
- 需求分析和设计
- 代码开发
- 功能测试

导入小程序代码



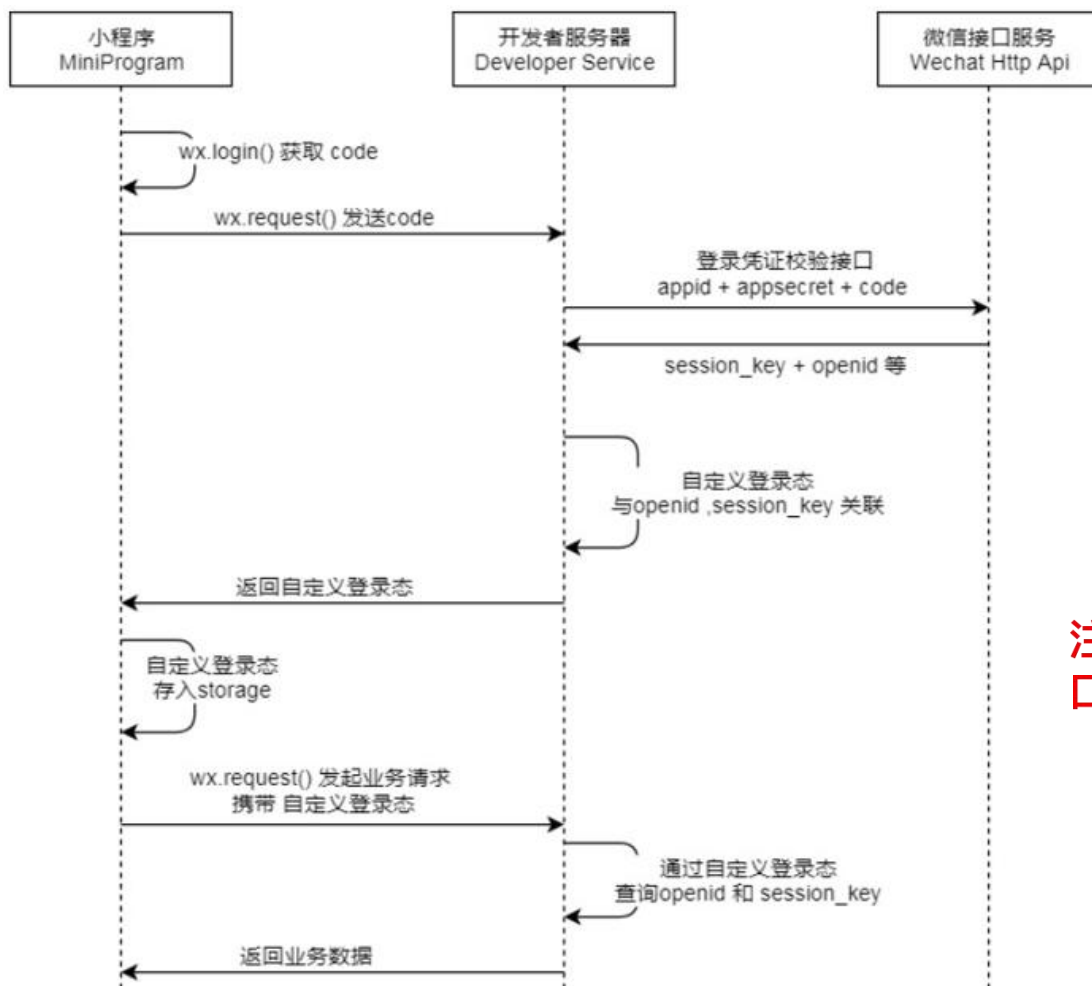


微信登录

- 导入小程序代码
- 微信登录流程
- 需求分析和设计
- 代码开发
- 功能测试

微信登录流程

微信登录: <https://developers.weixin.qq.com/miniprogram/dev/framework/open-ability/login.html>



注意: 可以使用postman测试登录凭证校验接口



微信登录

- 导入小程序代码
- 微信登录流程
- 需求分析和设计
- 代码开发
- 功能测试

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

需求分析和设计

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			

需求分析和设计

数据库设计（user表）：

字段名	数据类型	说明	备注
id	bigint	主键	自增
openid	varchar(45)	微信用户的唯一标识	
name	varchar(32)	用户姓名	
phone	varchar(11)	手机号	
sex	varchar(2)	性别	
id_number	varchar(18)	身份证号	
avatar	varchar(500)	微信用户头像路径	
create_time	datetime	注册时间	

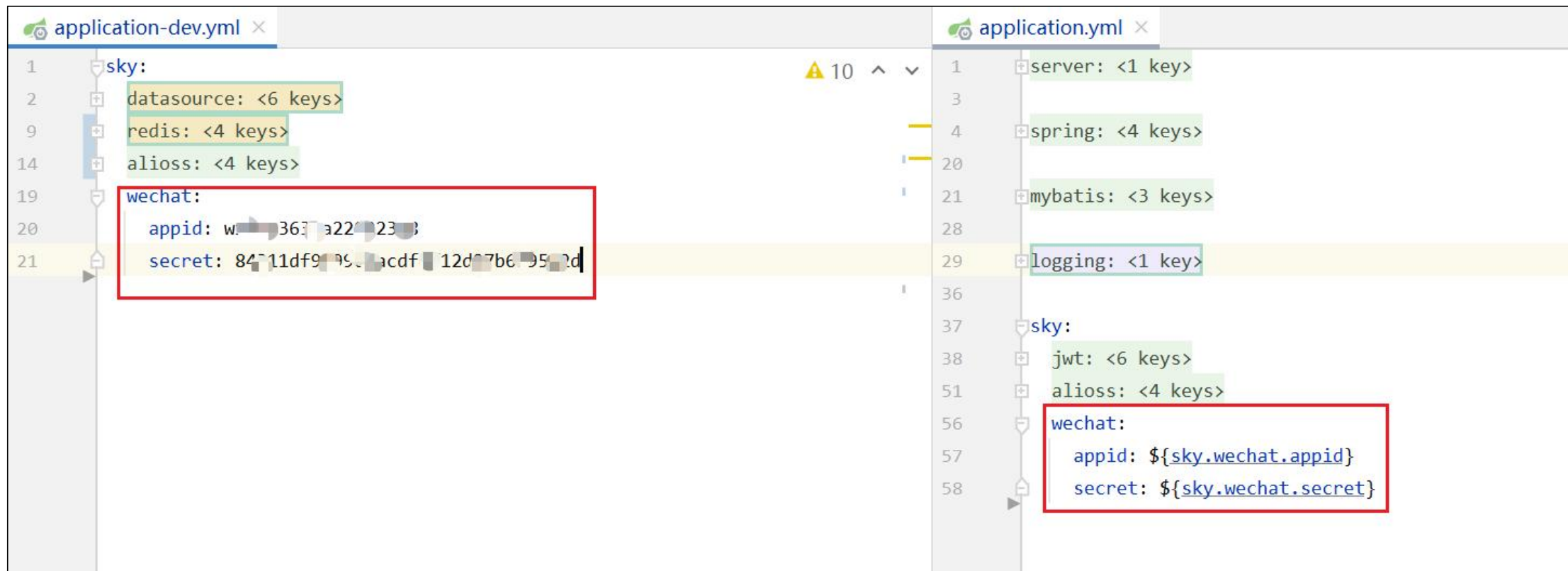


微信登录

- 导入小程序代码
- 微信登录流程
- 需求分析和设计
- 代码开发
- 功能测试

代码开发

配置微信登录所需配置项:



```
application-dev.yml
1 sky:
2   datasource: <6 keys>
9   redis: <4 keys>
14  alioss: <4 keys>
19  wechat:
20    appid: w...363...a22...23...
21    secret: 84...11df9...acdf...12d...7b6...95...2d

application.yml
1 server: <1 key>
3
4 spring: <4 keys>
20
21 mybatis: <3 keys>
28
29 logging: <1 key>
36
37 sky:
38   jwt: <6 keys>
51   alioss: <4 keys>
56   wechat:
57     appid: ${sky.wechat.appid}
58     secret: ${sky.wechat.secret}
```

代码开发

配置为微信用户生成jwt令牌时使用的配置项：

```
application.yml x
1  server: <1 key>
3
4  spring: <4 keys>
20
21  mybatis: <3 keys>
28
29  logging: <1 key>
36
37  sky:
38    jwt:
39      # 设置jwt签名加密时使用的密钥
40      admin-secret-key: itcast
41      # 设置jwt过期时间
42      admin-ttl: 7200000
43      # 设置前端传递过来的令牌名称
44      admin-token-name: token
45      # 设置jwt签名加密时使用的密钥
46      user-secret-key: itheima
47      # 设置jwt过期时间
48      user-ttl: 7200000
49      # 设置前端传递过来的令牌名称
50      user-token-name: authentication
51  alioss: <4 keys>
56  wechat: <9 keys>
```


代码开发

DTO设计:

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	



```
/**
 * C端用户登录
 */
@Data
public class UserLoginDTO implements Serializable {

    //微信用户授权码
    private String code;

}
```

代码开发

VO设计:

名称	类型	是否必须	默认值	备注
code	integer	必须		
data	object	必须		
└ id	integer	必须		用户id
└ openid	string	必须		微信openid
└ token	string	必须		jwt令牌
msg	string	非必须		



```
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class UserLoginVO implements Serializable {

    private Long id;
    private String openid;
    private String token;

}
```

代码开发

根据接口定义创建UserController的login方法：

```
@RestController
@RequestMapping("/user/user")
@Slf4j
@Api(tags = "C端-用户接口")
public class UserController {

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

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

UserController

代码开发

完善UserController的login方法:

```
/**
 * C端用户登录--微信登录
 * @param userLoginDTO
 * @return
 */
@PostMapping("/login")
@ApiOperation("登录")
public Result<UserLoginVO> login(@RequestBody UserLoginDTO userLoginDTO) {
    log.info("微信用户登录, 授权码为: {}", userLoginDTO.getCode());
    User user = userService.wxLogin(userLoginDTO);

    Map 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);
}
```

UserController

代码开发

创建UserService接口：

```
public interface UserService {  
  
    /**  
     * 根据微信授权码实现微信登录  
     * @param userLoginDTO  
     * @return  
     */  
    User wxLogin(UserLoginDTO userLoginDTO);  
}
```

代码开发

创建UserServiceImpl实现类：

```
@Service
@Slf4j
public class UserServiceImpl implements UserService {

    public static final String WX_LOGIN = "https://api.weixin.qq.com/sns/jscode2session";

    @Autowired
    private UserMapper userMapper;

    @Autowired
    private WeChatProperties weChatProperties;

    /**
     * 根据微信授权码实现微信登录
     * @param userLoginDTO
     * @return
     */
    public User wxLogin(UserLoginDTO userLoginDTO) {
        return null;
    }
}
```

代码开发

在UserServiceImpl中创建私有方法getOpenid:

```
/**
 * 获取微信用户的openid
 * @param code
 * @return
 */
private String getOpenid(String code){
    //请求参数封装
    Map map = new HashMap();
    map.put("appid",weChatProperties.getAppid());
    map.put("secret",weChatProperties.getSecret());
    map.put("js_code",code);
    map.put("grant_type","authorization_code");

    //调用工具类，向微信接口服务发送请求
    String json = HttpClientUtil.doGet(WX_LOGIN, map);
    log.info("微信登录返回结果: {}", json);

    //解析json字符串
    JSONObject jsonObject = JSON.parseObject(json);
    String openid = jsonObject.getString("openid");
    log.info("微信用户的openid为: {}", openid);

    return openid;
}
```

UserServiceImpl

代码开发

完善UserServiceImpl的wxLogin方法:

```
public User wxLogin(UserLoginDTO userLoginDTO) {  
    //授权码  
    String code = userLoginDTO.getCode();  
    String openid = getOpenid(code);  
  
    if(openid == null){  
        throw new LoginFailedException(MessageConstant.LOGIN_FAILED);  
    }  
  
    //根据openid查询用户信息  
    User user = userMapper.getByOpenid(openid);  
  
    if(user == null){  
        //当前是一个新用户, 自动完成注册  
        user = new User();  
        user.setOpenid(openid);  
        user.setCreateTime(LocalDateTime.now());  
        userMapper.insert(user);  
    }  
    return user;  
}
```

UserServiceImpl

代码开发

创建UserMapper接口：

```
@Mapper
public interface UserMapper {

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

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

UserMapper

代码开发

创建UserMapper.xml映射文件：

```
<?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">

    <!--插入-->
    <!--
    useGeneratedKeys设置为true：在执行插入记录之后可以获取到数据库自动生成的主键值
    keyProperty：指定Java对象的属性名
    -->
    <insert id="insert" parameterType="User" 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>
```

UserMapper.xml

代码开发

编写拦截器JwtTokenUserInterceptor，统一拦截用户端发送的请求并进行jwt校验：

//1、从请求头中获取令牌

```
String token = request.getHeader(jwtProperties.getUserTokenName());
```

//2、校验令牌

```
try{  
    Claims claims = JwtUtil.parseJWT(jwtProperties.getUserSecretKey(), token);  
    Long userId = Long.valueOf(claims.get(JwtClaimsConstant.USER_ID).toString());  
    BaseContext.setCurrentId(userId);  
    //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;  
}
```

代码开发

在WebMvcConfiguration配置类中注册拦截器：

```
@Configuration
@Slf4j
public class WebMvcConfiguration extends WebMvcConfigurationSupport {

    @Autowired
    private JwtTokenAdminInterceptor jwtTokenAdminInterceptor;

    @Autowired
    private JwtTokenUserInterceptor jwtTokenUserInterceptor;

    /**
     * 注册自定义拦截器
     * @param registry
     */
    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");
    }
}
```



微信登录

- 导入小程序代码
- 微信登录流程
- 需求分析和设计
- 代码开发
- 功能测试

功能测试

可以通过接口文档进行测试，最后完成前后端联调测试即可



目录

Contents

- ◆ HttpClient
- ◆ 微信小程序开发
- ◆ 微信登录
- ◆ 导入商品浏览功能代码

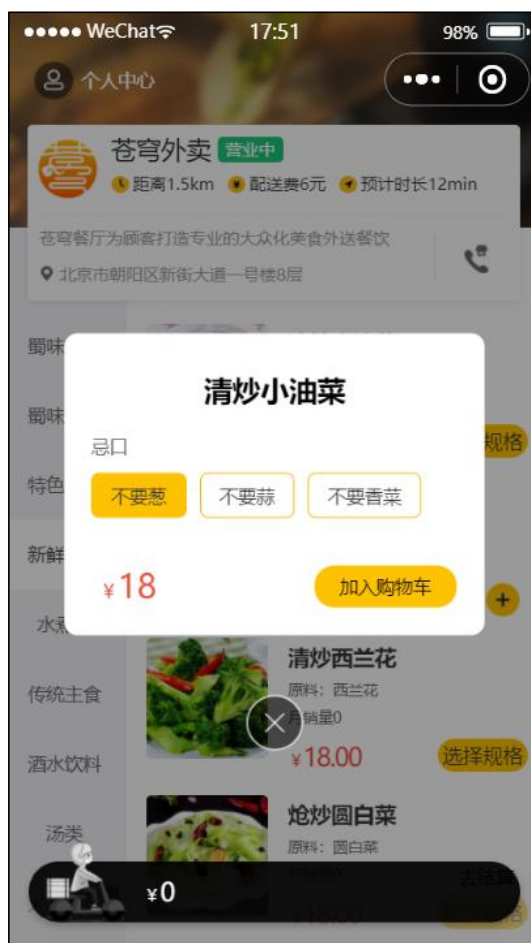
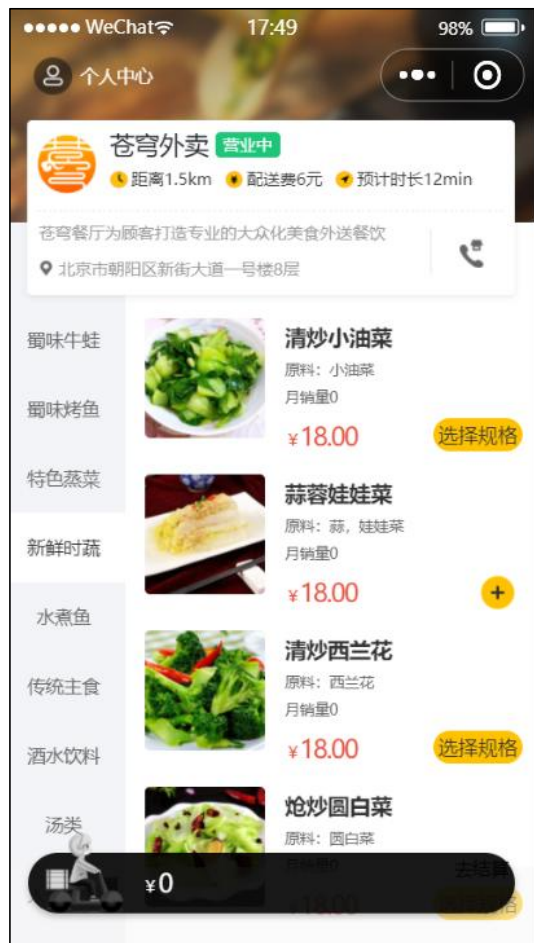


导入商品浏览功能代码

- 需求分析和设计
- 代码导入
- 功能测试

需求分析和设计

产品原型:



需求分析和设计

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			

需求分析和设计

- 根据套餐id查询包含的菜品 接口

基本信息

Path: /user/setmeal/dish/{id}

Method: GET

接口描述:

请求参数

路径参数

参数名称	示例	备注
id		套餐id

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	number	必须			
data	object []	非必须			item 类型: object
└─ copies	number	必须		份数	
└─ description	string	必须		菜品描述	
└─ image	string	必须		菜品图片	
└─ name	string	必须		菜品名称	
msg	string	非必须			



导入商品浏览功能代码

- 需求分析和设计
- 代码导入
- 功能测试

代码导入

› 授课资料 › day06-微信登录、商品浏览 › 资料 › 代码导入 › 商品浏览

名称

CategoryController.java
DishController.java
DishService.java
DishServiceImpl.java
SetmealController.java
SetmealMapper.java
SetmealMapper.xml
SetmealService.java
SetmealServiceImpl.java



导入商品浏览功能代码

- 需求分析和设计
- 代码导入
- 功能测试

功能测试

通过Swagger接口文档进行测试，通过后再前后端联调测试即可



传智教育旗下高端IT教育品牌