

第七章 Ego微商小程序接口测试篇

学习目标

1. 知道接口的测试流程
2. 知道接口文档包含的内容
3. 能够根据接口文档编写接口测试用例
4. 能够使用Postman实现接口测试
5. 能够通过python+requests+unittest实现接口测试

一、接口测试流程

1. 接口测试相关概念

1.1 接口

接口：是指系统或组件之间的交互点，通过这些交互点可以来进行数据之间的交互。（数据交互的通道）

1.2 接口测试

接口测试：是对系统或组件之间的接口进行测试，主要是校验数据的交换、传递和控制管理过程，以及相互逻辑依赖关系。

1.3 接口测试的特点

- 测试可以提前介入，提早发现Bug，符合质量控制前移的理念
- 可以发现一些页面操作发现不了的问题
- 接口测试低成本高效益
- 不同于传统的单元测试，接口测试是从用户的角度对系统进行检测

1.4 接口测试的实现方式

- 使用接口测试工具来实现（比如：JMeter、Postman）
- 通过编写代码来实现（比如：Python + Requests）

2. 接口测试流程

1. 需求分析

- 主要依据需求文档

2. 接口文档解析

- 一般是由开发人员编写接口文档（API文档）

3. 设计测试用例

4. 执行测试

- 使用接口测试工具实现
- 通过编写代码实现

5. 接口缺陷管理与跟踪
6. 接口自动化持续集成

二、接口文档解析

1. 接口文档介绍

1.1 什么是接口文档？

接口文档：又称为API文档，一般是由开发人员所编写的，用来描述系统所提供接口信息的文档。大家都根据这个接口文档进行开发，并需要一直维护和遵守。

1.2 为什么要写接口文档？

1. 能够让前端开发与后台开发人员更好的配合，提高工作效率。（有一个统一参考的文件）
2. 项目迭代或者项目人员更迭时，方便后期人员查看和维护
3. 方便测试人员进行接口测试

1.3 接口文档内容

一个规范的接口文档，要包含以下信息：

- 基本信息
 - 接口名称、请求方法、请求路径、接口描述
- 请求参数
 - 请求头
 - 请求体（包含具体的请求参数名称、参数类型、是否必须、示例、备注）
- 返回数据
 - 不同情况的响应状态码
 - 响应数据（包含具体的响应数据名称、类型、是否必须、默认值、示例、备注）

1.4 接口文档示例

用户认证（登录注册）

基本信息

Path: /app/v1_0/authorizations

Method: POST

接口描述:

1. 线上地址
http://ttapi.research.itcast.cn/app/v1_0/authorizations
2. 返回HTTP状态码
 1. 201 OK
 2. 400 请求参数错误(包括: 参数缺失、手机号格式不正确、验证码失效等)
 3. 507 服务器数据库异常
3. token说明
token用于访问需要身份认证的普通接口, 有效期2小时

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 |
|--------|--------|------|-----|-------|------|
| mobile | string | 必须 | | 手机号 | |
| code | string | 必须 | | 短信验证码 | |

返回数据

| 名称 | 类型 | 是否必须 | 默认值 | 备注 | 其他信息 |
|----------|--------|------|-----|-----------|------|
| message | string | 必须 | | 提示信息 | |
| data | object | 非必须 | | 数据 | |
| └─ token | string | 必须 | | 用户token令牌 | |

2. 接口文档解析

小程序项目接口文档接口列表:

1. 首页
 - 轮播图
 - 专题栏位

- 最近新品

2. 商品

- 获取商品分类
- 获取商品分类下的商品
- 获取商品信息

3. 订单

- 获取用户订单列表
- 创建订单
- 查看订单

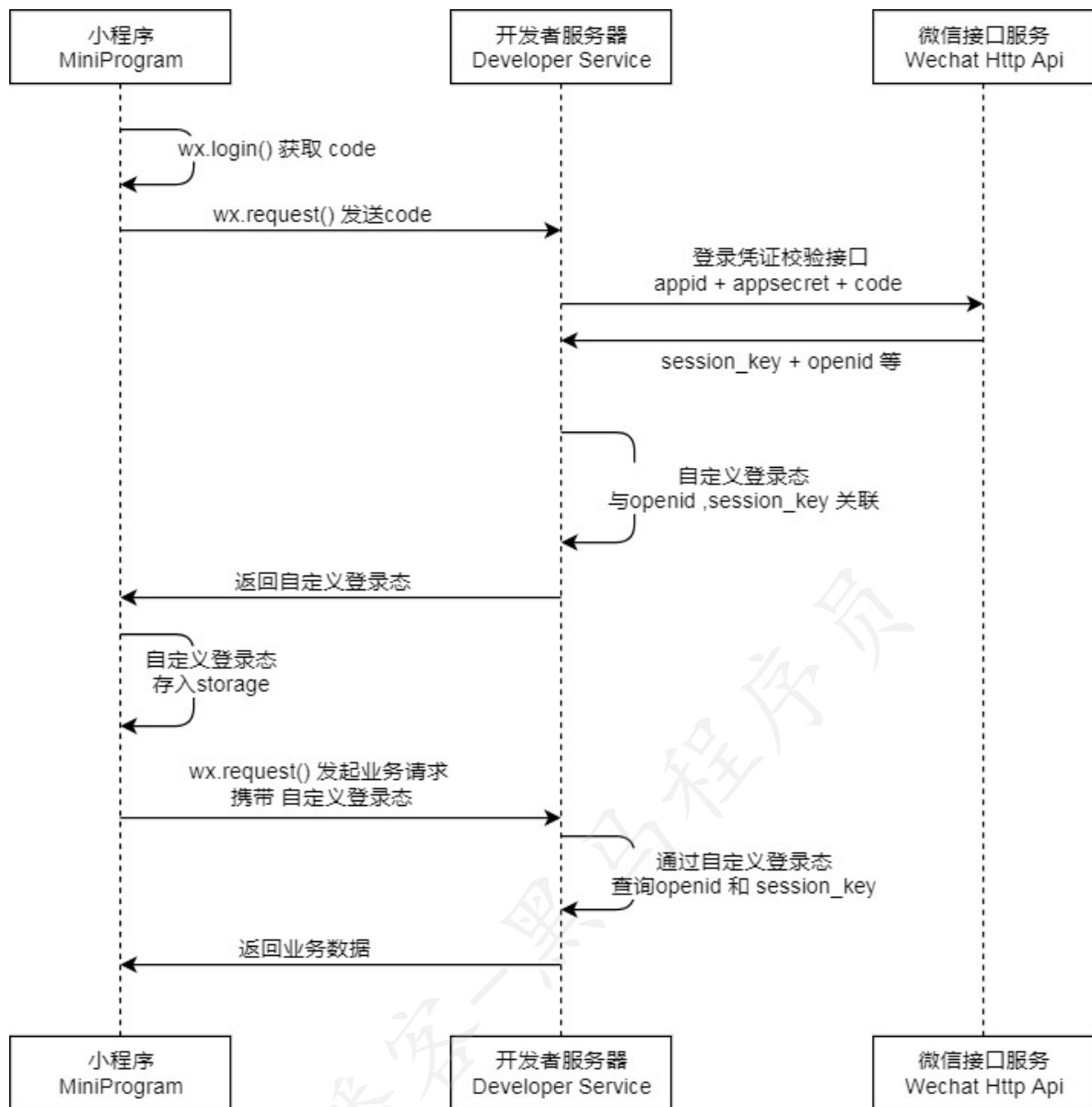
4. 用户权限

- 获取Token
- Token验证
- 获取地址信息

依据接口文档进行接口解析...

3. 小程序登录

3.1 小程序登录流程图



说明：

1. 调用 wx.login() 获取 临时登录凭证code，并回传到开发者服务器。
2. 调用 auth.code2Session 接口，换取 用户唯一标识 OpenID 和 会话密钥 session_key。
3. 之后开发者服务器可以根据用户标识来生成自定义登录态，用于后续业务逻辑中前后端交互时识别用户身份。

3.2 wx.login(Object object)

调用接口获取登录凭证（code），登录凭证有效期为五分钟，并且一个code只能使用一次。

示例代码：

```

wx.login({
  success (res) {
    if (res.code) {
      // 发起网络请求
      wx.request({
        url: 'https://test.com/onLogin',
        data: {
          code: res.code
        }
      })
    } else {

```

```
console.log('登录失败!' + res.errMsg)
    }
  }
});
```

三、设计接口用例

1. 接口测试的测试点



2. 接口用例设计的方法与思路

本项目主要关注接口的功能测试

功能测试：验证接口功能是否按照接口文档实现（输入+处理+输出）

- 单接口测试
 - 正向功能：(通过性测试)
 - 仅必填参数
 - 全部参数
 - 参数组合
 - 反向测试：(异常测试)
 - 参数异常：无参、少参、多参、错误参数
 - 数据异常：数据为空、长度不符、类型不符、错误数据
 - 业务数据异常：结合业务功能考虑输出的各种异常返回情况
- 多接口测试：业务场景功能测试（站在用户角度考虑常用的使用场景）
 - 接口之间数据依赖

3. 接口测试用例

| ID | 模块 | 用例名称 | 前置条件 | 请求URL | 请求类型 | 请求头 | 请求参数类型 | 请求参数 | 预期结果 |
|-----|------|------------|---------|---|------|-------------------------------------|--------|-----------|--|
| 001 | 首页 | 获取轮播图 | | http://e.cn/api/v1/banner/1 | GET | | 路径参数 | 1 | 成功, 状态码: 200, 返回数据: {"id":1,"name":"首页置顶","items":[{"key_word":"6","type":1,"img":{"url":"http://e.cn/images/banner-4a.png","update_time":"1970-01-01 08:00:00"}}]} |
| 002 | 首页 | 获取专题栏位 | | http://e.cn/api/v1/theme?ids=1,2,3 | GET | | 查询参数 | ids=1,2,3 | 成功, 状态码: 200, 返回数据: [{"id":1,"name":"专题栏位一","description":"美味水果世界","topic_img":{"url":"http://e.cn/images/1@theme.png"},"head_img":{"url":"http://e.cn/images/1@theme-head.png"}}] |
| 003 | 首页 | 获取最近新品 | | http://e.cn/api/v1/product/recent | GET | | | | 成功, 状态码: 200, 返回数据: [{"id":1,"name":"芹菜 半斤","price":"0.01","stock":998,"main_img_url":"http://e.cn/images/product-vg@1.png","img_id":13}] |
| 004 | 商品 | 获取商品分类 | | http://e.cn/api/v1/category/all | GET | | | | 成功, 状态码: 200, 返回数据: [{"id":2,"name":"果味","topic_img_id":6,"description":null,"img":{"url":"http://e.cn/images/category-dryfruit.png"}}] |
| 005 | 商品 | 获取商品分类下的商品 | | http://e.cn/api/v1/product/by_category?id=2 | GET | | 查询参数 | id=2 | 成功, 状态码: 200, 返回数据: [{"id":1,"name":"芹菜 半斤","price":"0.01","stock":998,"main_img_url":"http://e.cn/images/product-vg@1.png","img_id":13}] |
| 006 | 商品 | 获取商品信息 | | http://e.cn/api/v1/product/2 | GET | | 路径参数 | 2 | 成功, 状态码: 200, 返回数据: {"id":2,"name":"梨花带雨 3个","price":"0.01","stock":984,"main_img_url":"http://e.cn/images/product-dryfruit@1.png","summary":null,"img_id":10,"imgs":[{"properties":{"name":"保质期","detail":"10天"}}]} |
| 007 | 用户权限 | 获取Token | 获取到code | http://e.cn/api/v1/token/user | POST | {"Content-Type":"application/json"} | JSON | | 成功, 状态码: 200, 返回数据: {"token":"8741c34e360da3851cc3c6e484619b42"} |
| 008 | 用户权限 | Token验证 | 登录成功 | http://e.cn/api/v1/token/verify | POST | {"Content-Type":"application/json"} | JSON | | 成功, 状态码: 200, 返回数据: {"isValid":true} |
| 009 | 订单 | 获取用户订单列表 | 登录成功 | http://e.cn/api/v1/order/by_user?page=1 | GET | {"token":"xxx"} | 查询参数 | page=1 | 成功, 状态码: 200, 返回数据: {"current_page":1,"data":[{"id":46,"order_no":"CB15891253844855","create_time":"2019-11-15 11:38:45","total_price":"0.02","status":1,"snap_img":"http://e.cn/images/product-dryfruit@1.png","snap_name":"梨花带雨","total_count":2,"prepay_id":null}]} |
| 010 | 订单 | 创建订单 | 登录成功 | http://e.cn/api/v1/order | POST | {"Content-Type":"application/json"} | JSON | | 成功, 状态码: 200, 返回数据: {"order_no":"CB15913583627625","order_id":"47","create_time":"2019-11-15 12:15:58","pass":true} |
| 011 | 订单 | 查看订单 | 登录成功 | http://e.cn/api/v1/order/50 | GET | {"token":"xxx"} | 路径参数 | 50 | 成功, 状态码: 200, 返回数据: {"id":46,"order_no":"CB15891253844855","total_price":"0.02","status":1,"snap_name":"梨花带雨等","total_count":1,"snap_items":[{"id":2,"name":"梨花带雨","count":1,"totalPrice":"0.01","price":"0.01","counts":1},"snap_address":{"name":"李白","mobile":"13012345678"}]} |
| 012 | 用户 | 获取地址信息 | 登录成功 | http://e.cn/api/v1/address | GET | {"token":"xxx"} | | | 成功, 状态码: 200, 返回数据: {"name":"李白","mobile":"13012345678","province":"北京市","city":"北京市","country":"朝阳区","detail":"望京001号"} |

四、Postman实现接口测试

1. 初始化工作

1.1 创建测试用例结构

1. 创建测试集: 小程序
2. 创建目录
 - 首页
 - 商品
 - 用户权限
 - 订单



1.2 设置环境变量

把项目的公共配置信息可以添加到环境变量中，在请求中直接引用

- 测试环境
 - base_url=<http://e.cn/api/v1>
 - code=xxx

2. 实现测试用例

根据编写的测试用例文档，使用Postman实现测试用例

2.1 首页-获取轮播图

操作步骤

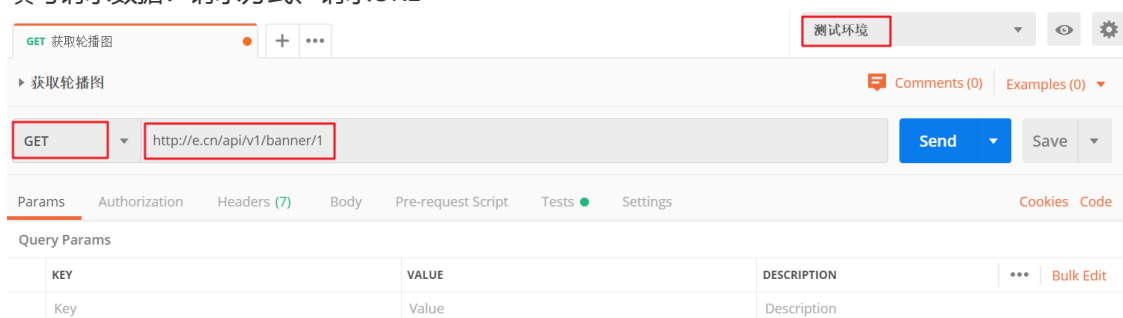
1. 在'首页'目录下，添加'获取轮播图'的请求
2. 填写请求数据：请求方式、请求URL、请求头、请求体
3. 在'Tests'标签页中，编写测试脚本：断言、业务数据处理
4. 发送请求，调试脚本

实现

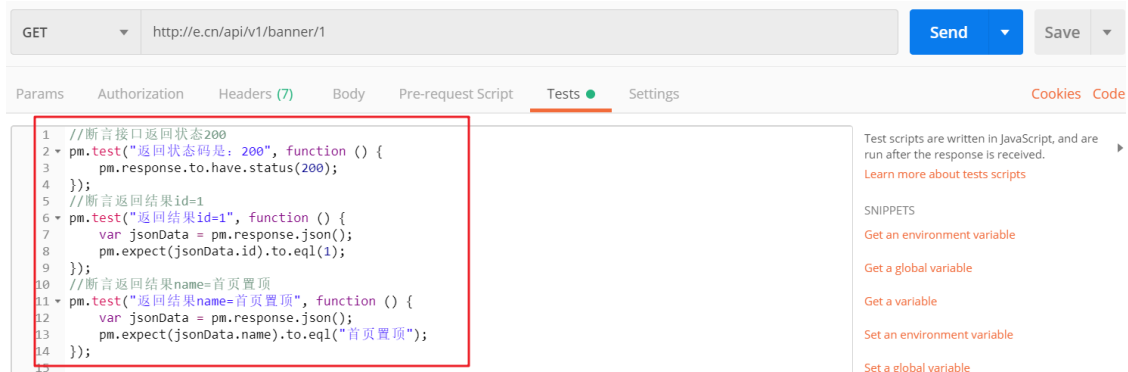
1. 添加'获取轮播图'的请求



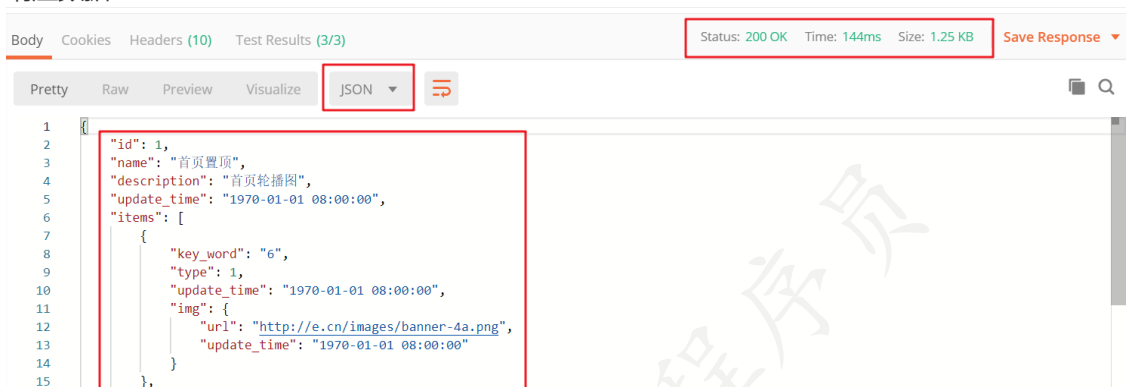
2. 填写请求数据：请求方式、请求URL



3. 编写测试脚本



4. 响应数据



5. 断言结果



3. 生成测试报告

```
newman run 测试脚本文件 -e 环境变量文件 -r html --reporter-html-export report.html
```

操作步骤

1. 导出测试集数据
2. 导出环境变量数据
3. 执行命令: `newman run mini_program.postman_collection.json -e test.postman_environment.json -r html --reporter-html-export report.html`

五、Requests实现接口测试

1. 项目搭建

1.1 新建项目

项目名称: apiTestMiniProgram

1.2 创建目录结构

```
apiTestMiniProgram
├─ api          # 封装接口类
├─ script       # 定义测试脚本
├─ report       # 存放测试报告
├─ tools        # 存放第三方文件
├─ log          # 存放日志文件
├─ app.py       # 定义项目配置信息
├─ utils.py     # 封装工具类
└─ run_suite.py # 封装测试套件
```

1.3 安装依赖包

- 安装 requests 包
- 添加 HTMLTestRunner

2. 初始化日志配置

使用Python中的 logging 日志模块来收集日志，把日志信息输出到控制台和日志文件中

```
# app.py
# 初始化日志配置
def init_log_config():
    # 创建日志器
    logger = logging.getLogger()
    logger.setLevel(logging.INFO)

    # 创建控制台处理器
    sh = logging.StreamHandler()

    # 创建文件处理器
    log_path = BASE_DIR + "/log/mini_program.log"
    fh = logging.handlers.TimedRotatingFileHandler(log_path, when="midnight",
    interval=1,
    backupCount=7, encoding="UTF-8")

    # 创建格式化器
    f = '%(asctime)s %(levelname)s [(name)s] [(filename)s%(funcName)s:%(lineno)d] - %(message)s'
    formatter = logging.Formatter(f)

    # 把格式化器添加到处理器中
    sh.setFormatter(formatter)
    fh.setFormatter(formatter)

    # 把处理器添加到日志器中
    logger.addHandler(sh)
    logger.addHandler(fh)
```

3. 编写代码

3.1 封装接口类

根据用例文档分析待测功能，按功能模块定义接口类

```
首页模块: index.py
商品模块: product.py
用户模块: user.py
订单模块: order.py
```

3.2 编写测试脚本

1. 定义测试脚本文件

```
首页模块: test_index.py
商品模块: test_product.py
用户模块: test_user.py
订单模块: test_order.py
```

2. 使用UnitTest管理测试脚本

3.3 执行测试脚本

1. 使用UnitTest执行测试脚本
2. 调试代码

4. 生成测试报告

使用 `HTMLTestRunner` 生成测试报告，并分析测试结果

```
report_file = "./report/report{}.html".format(time.strftime("%Y%m%d-%H%M%S"))
with open(report_file, "wb") as f:
    runner = HTMLTestRunner(f, title="小程序接口自动化测试报告", description="v1.0")
    runner.run(suite)
```