

v1版本（done）

解决问题：主要支撑主流程场景用例，适合复杂场景用例，包括ui、sipp、接口。

一.自动化框架



二.权限token问题

login()

读取配置文件账号登陆，此token作为所有接口的默认权限token。

```
#登陆账号
login.account=zh100@s3.com
login.pwd=123456
```

login(String account, String pwd)

入参账号，此token会作为后续接口请求的权限token。（一些接口需要从网关获取用户信息）

eg. `speech/api/v1/agent/outboundTask/manual/customer/list/{value}`, 此接口需要获取用户信息

```
aBasicApi.login("hc01@auto.com", "123456");
```

```
HttpResponse httpResponse_task_id = agentApi.getOutboundTaskCustomerList(task_id);
```

```
@Value("${login.account}")
private String account;

@Value("${login.pwd}")
private String pwd;

public void login(){
    if(GlobalVar.GLOBAL_DATA_MAP.get("Authorization") == null){
        HttpResponse httpResponse = basicApi.login(account, pwd);
        GLOBAL_DATA_MAP.put("Authorization", httpResponse.getHeader("key: Authorization"));
    }
}

public void login(String account, String pwd){
    HttpResponse httpResponse = basicApi.login(account, pwd);
    GLOBAL_DATA_MAP.put("Authorization", httpResponse.getHeader("key: Authorization"));
}
```

三.http请求实现

1.http封装

一个http请求包括：请求方式、请求路径、请求头部、请求数据

先看下feign的使用：注入到ioc容器-->jdk代理

[illegible]

postman上的一个请求

新增呼叫中心技能组

POST

http://smartrxmauss-zkj-stest3-8080.zkj.test/api/v1/ussgroup?...

Params

Send

Key	Value	Description
<input checked="" type="checkbox"/> tenantId	87462158132343186600590049	
New key	Value	Description

Authorization

Headers (1)

Body

Pre-request Script

Tests

form-data

x-www-form-urlencoded

raw

binary

JSON (application/json)

```
1 {
2   "groupName": "测试1000",
3   "onlineAbility": false,
4   "speechAbility": true,
5   "speechAcw": 15,
6   "speechDid": "15",
7   "speechOverflowGroupId": "",
8   "speechOverflowQueueSize": 0,
9   "speechQueueOverflowSwitch": false,
10  "speechRestLastTime": 30,
11  "speechRestNumber": 10,
12  "speechType": 1,
13  "videoAbility": false,
14  "workorderAbility": false
15 }
```

结合feign和postman实现我们的http请求：

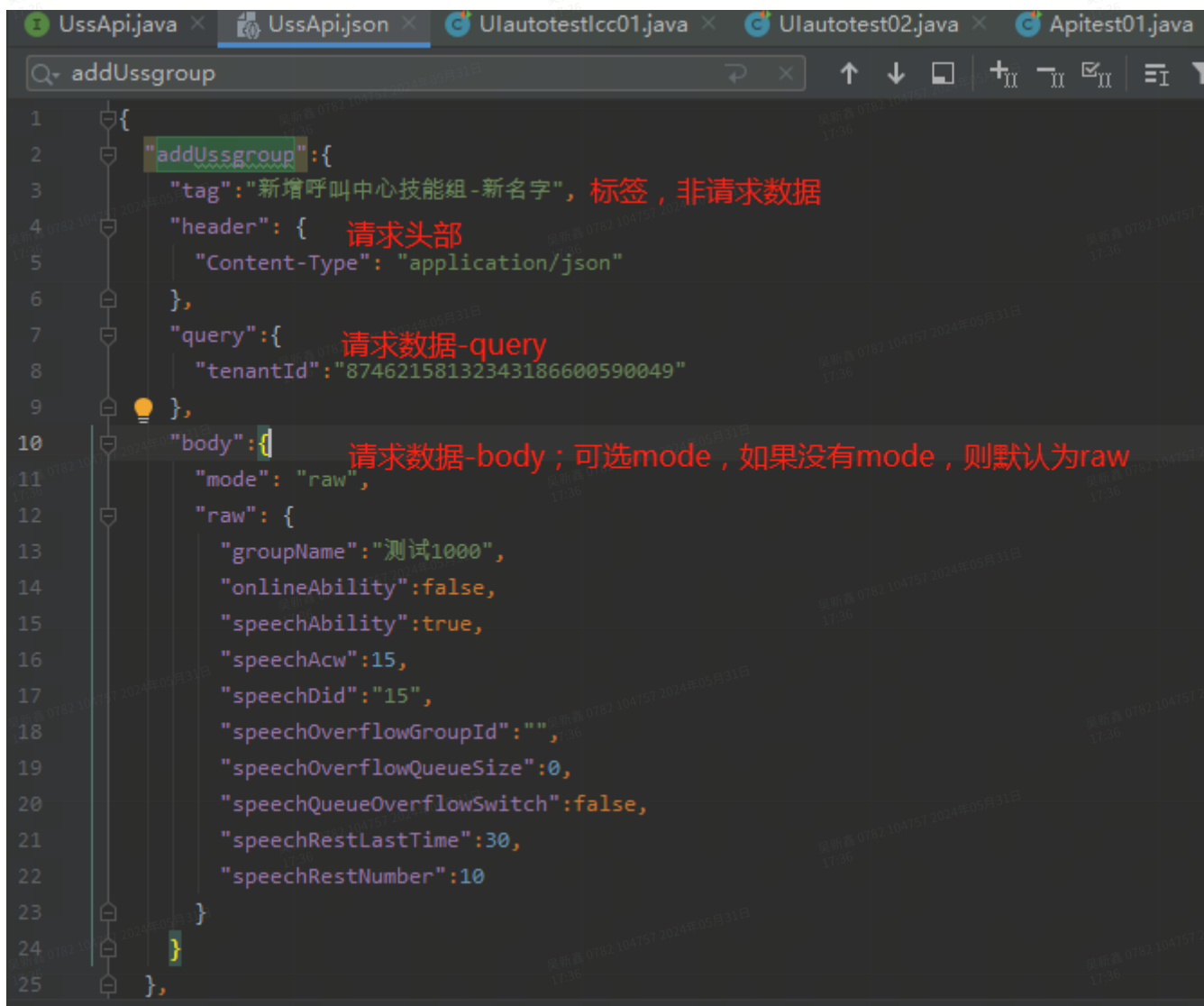
(1) 请求方式、请求路径 在接口中声明

在接口中还会声明入参，入参将替换请求数据中对应的值

UssApi.java Mapping.java Post.java

```
1 package com.example.uiautest.service.interfaces.uss;
2
3 import com.example.uiautest.common.annotations.*;
4 import org.apache.http.HttpResponse;
5
6 @Mapping(path = "/uss", url = "${url.uss}")
7 public interface UssApi {
8
9     @Post(path = "/api/v1/ussgroup", description = "新增呼叫中心技能组")
10    HttpResponse addUssgroup(@Param("groupName") String groupName);
11
12    @Put(path = "/api/v1/ussgroup", description = "更新呼叫中心技能组")
13    HttpResponse putUssgroup();
14
15    @Get(path = "/api/v1/ussgroup", description = "获取呼叫中心技能组")
16    HttpResponse getUssgroups();
17
18 }
```

(2) 请求头部、请求数据 (body、query) 在json数据文件中声明



```
1 {
2   "addUssgroup": {
3     "tag": "新增呼叫中心技能组-新名字", 请求数据
4     "header": { 请求头部
5       "Content-Type": "application/json"
6     },
7     "query": { 请求数据-query
8       "tenantId": "87462158132343186600590049"
9     },
10    "body": { 请求数据-body; 可选mode, 如果没有mode, 则默认为raw
11      "mode": "raw",
12      "raw": {
13        "groupName": "测试1000",
14        "onlineAbility": false,
15        "speechAbility": true,
16        "speechAcw": 15,
17        "speechDid": "15",
18        "speechOverflowGroupId": "",
19        "speechOverflowQueueSize": 0,
20        "speechQueueOverflowSwitch": false,
21        "speechRestLastTime": 30,
22        "speechRestNumber": 10
23      }
24    }
25  },
26 }
```

如下，没有mode，则默认mode为raw；默认请求头部Content-Type=application/json（大部分post请求都是此类）

ps：postman参数格式raw适用于json格式的数据进行传参，并且header里必须加上Content-Type的值为application/json，才能被后端接收到。（后端注解@RequestBody）

```
1 {
2   "addUssgroup": {
3     "tag": "新增呼叫中心技能组-新名字",
4     "query": {
5       "tenantId": "8746215813313186600598849"
6     },
7     "body": {
8       "groupName": "测试1000",
9       "onlineAbility": false,
10      "speechAbility": true,
11      "speechAcw": 15,
12      "speechDid": "15",
13      "speechOverflowGroupId": "",
14      "speechOverflowQueueSize": 0,
15      "speechQueueOverflowSwitch": false,
16      "speechRestLastTime": 30,
17      "speechRestNumber": 10
18    }
19  },
20
21  "addUssgroup": { "Content-Type": "application/json",
22    "tag": "新增呼叫中心技能组-老名字",
23    "header": {
24
25  }
```

目前只支持form-data、raw；默认为raw

mode为“form-data”类型时，默认请求头部Content-Type=application/x-www-form-urlencoded

```
{
  "addUser": {
    "body": {
      "mode": "form-data",
      "form-data": {
        "name": "test001",
        "nickname": "test001",
        "email": "test001@auto.com",
        "phone": "18190890001",
        "empno": "907001",
        "roleIds": "700233019",
        "id": "",
        "departmentIds": ""
      }
    }
  }
}
```

▶ 新增呼叫中心技能组 Copy

POST ▾

http://smartxmauss-zkj-stest3-8080.zkj.test/api/v1/ussgroup?tenantId=87462158132343186600590049

Authorization

Headers (1)

Body ●

Pre-request Script

Tests

☒ form-data

☐ x-www-form-urlencoded

☐ raw

☐ binary

Key	Value
<input checked="" type="checkbox"/> test	1133

New key

Text ▾

Value

(3) 请求域名组装

如果存在登陆（全局变量authorization不为空），则请求域名部分会走gateway转发请求（例如“gateway地址 + Mapping->path + Post->path”组装成请求url）

如果不存在登陆（全局变量authorization为空），则会直接使用Mapping->url作为请求域名部分；
（例如“Mapping->url + Post->path” 组装成请求url） 【目前未实现】

```
UssApi.java Mapping.java Post.java
1 package com.example.uiatest.service.interfaces.uss;
2
3 import com.example.uiatest.common.annotations.*;
4 import org.apache.http.HttpResponse;
5
6 @Mapping(path = "/uss", url = "${url.uss}")
7 public interface UssApi {
8
9     @Post(path = "/api/v1/ussgroup", description = "新增呼叫中心技能组")
10    HttpResponse addUssgroup(@Param("groupName") String groupName);
11
12    @Put(path = "/api/v1/ussgroup", description = "更新呼叫中心技能组")
13    HttpResponse putUssgroup();
14
15    @Get(path = "/api/v1/ussgroup", description = "获取呼叫中心技能组")
16    HttpResponse getUssgroups();
17
18 }
```

2.数据优先级

入参>json数据

json文件层：指定环境>default环境



json层：匹配tag>默认第一个


```
{
  "addUssgroup": [
    {
      "tag": "新增呼叫中心技能组-新名字",
      "param": {
        "tenantId": "87462158132343186600590049"
      },
      "body": {
        "groupName": "test1000",
        "speechAbility": true,
        "onlineAbility": false,
        "speechDid": "202025",
        "speechAcw": 15,
        "speechRestNumber": 10,
        "speechQueueOverflowSwitch": false,
        "speechOverflowQueueSize": 0,
        "speechOverflowGroupId": ""
      }
    }
  ]
}
```

可匹配的tag

3.A端接口请求

(1) 配置文件写入a端登录账号

```
application-sass.properties
login.pwd=Msxf123456
#A端登录账号
a.login.account=login-stu
a.login.pwd=TT554Q202300
```

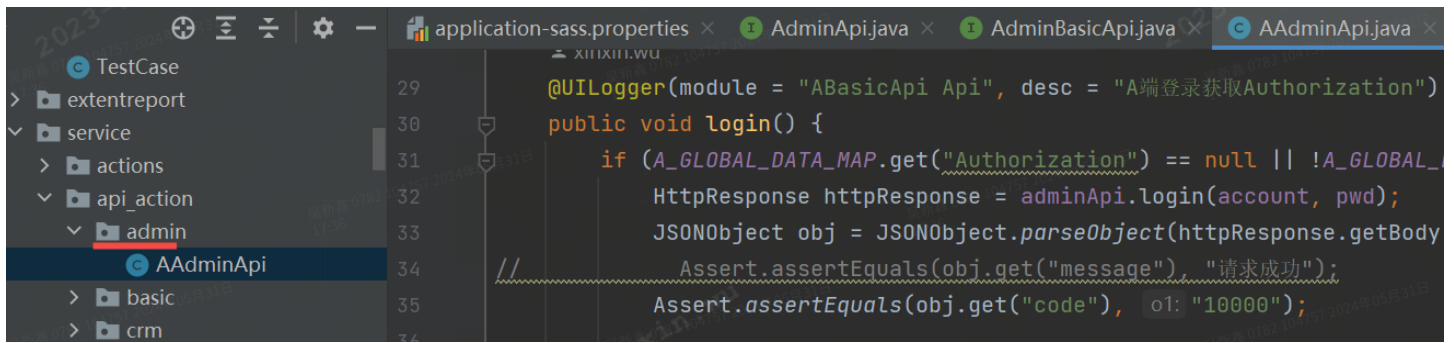
(2) a端接口请求

a端接口声明类放到service.interfaces.admin目录下

接口声明类注解需标识 url="admin" (底层请求会根据url="admin"判断取A端登录token)

```
application-sass.properties x AdminApi.java x AdminBasicApi.java x
2 usages xinxin.wu
@Mapping(path = "/basic", url = "admin")
public interface AdminBasicApi {
1 usage xinxin.wu
@GetMapping(path = "/front/config/a", description = "配置信息")
HttpResponse front();
}
```

a端封装方法放在service.api_action.admin目录下

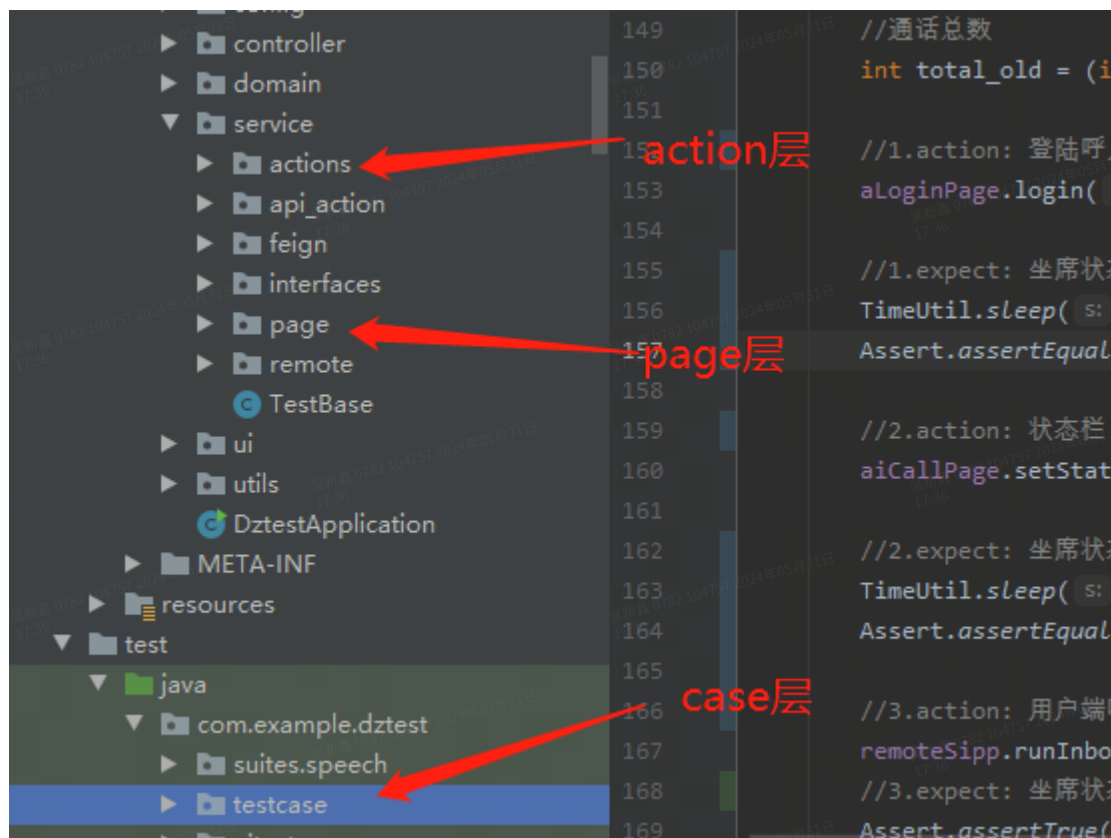


四. 页面控制

使用selenium完成页面控制，非不得已的情况下不使用页面控制。

1.page模式

- (1) page页：主要包括各个页面的元素，以及元素的基础操作；被case层、action层调用
- 支持的元素定位方式：id、name、linkText、partialLinkText、xpath、className、cssSelector
- 点击操作：driver.findElement(By.id(id)).click();
- 输入：driver.findElement(By.id(id)).sendKeys("输入");
- 获取文本：driver.findElement(By.id(id)).getText();
- 获取对象属性值：driver.findElement(By.id(id)).getAttribute("属性");
- (2) action： 页面元素的操作组合，它可以被Case层重复利用
- (3) case：业务逻辑的测试用例

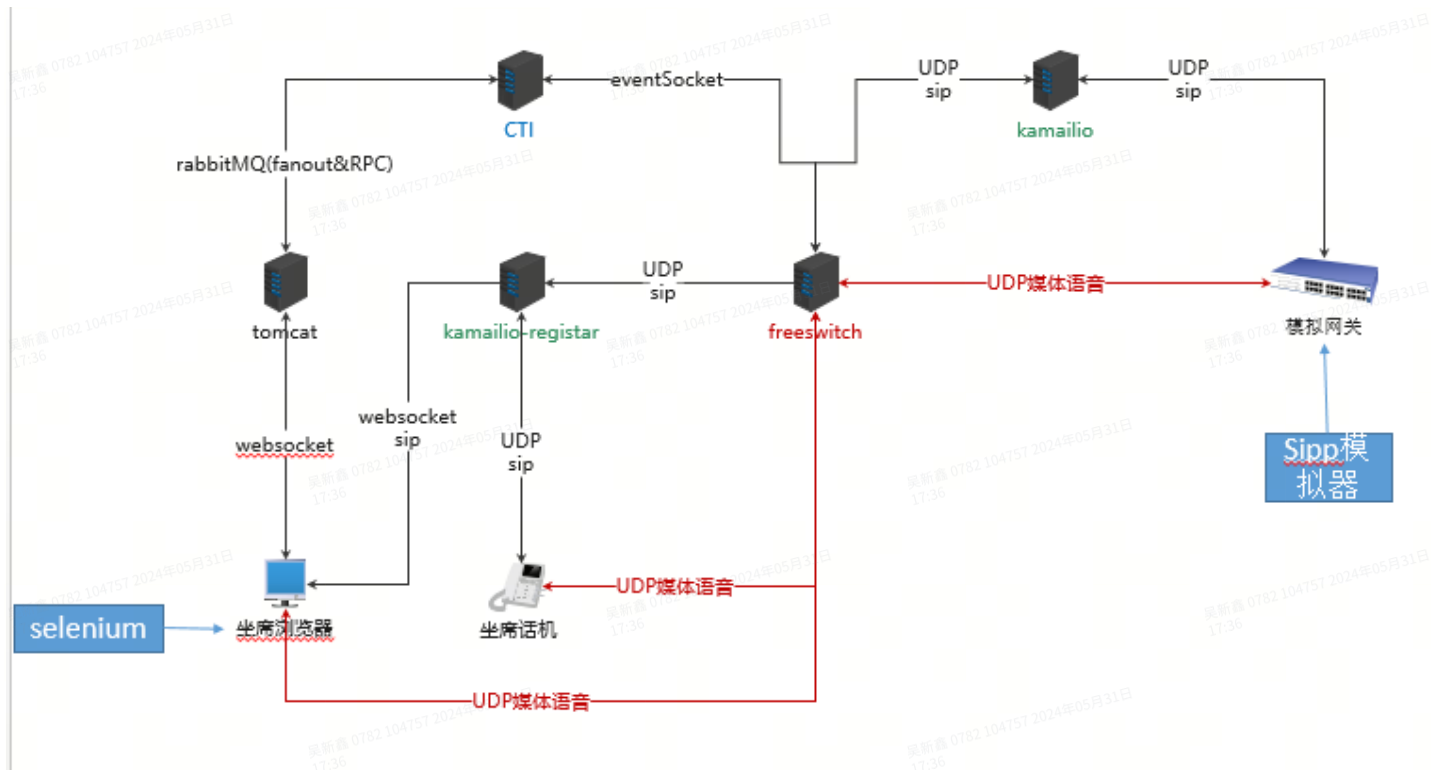


2.chrome-linux服务

linux服务安装chrome参考 <http://wiki.msxf.com/pages/viewpage.action?pageId=12686952>

五. 电话模拟

1.呼叫中心拨打控制网络逻辑结构



2.模拟呼出

登陆测试虚拟机 10.193.198.22 执行下面命令注册号码

```
cd /home/icc;sh outbound.sh
```

3.模拟呼入

登陆测试虚拟机 10.193.198.22 执行下面命令即可模拟多个号码拨打:

```
/home/icc/sipp/sipp-3.3/sipp 10.193.198.159:38911 -r 20 -rp 1000 -i 10.193.198.22 -p 50021 -s 33330002 -sf /home/icc/sipp/sipp-3.3/dezhutest/inbound_G711a.xml -inf /home/icc/sipp/sipp-3.3/dezhutest/CS2.csv -m 2 -trace_msg -trace_screen -trace_err -aa
```

这是命令，参数你只需要修改下面几个：

-i 10.193.198.166 (sipp本机ip)

-p 50021 (端口)

-s 222 (拨打的号码)

-m 99 (最大呼叫个数)

-r 20 -rp 1000 每1000毫秒送号20个

4.ssh远程调用

模拟呼出调用远程执行方法，需等待结果

executeWithResult(Connection connection, String command)

模拟呼入调用远程执行方法，只执行不需要等待结果

executeWithoutResult(Connection connection, String command)

六.报表生成

allure-results为allure生成的文件夹

(1) 启动生成报表，并启动服务

#allure serve allure-results

(2) 将生成的数据生成html文件并指定位置

#allure generate allure-results -o ./tmp/allure

(3) 手动打开allure-report

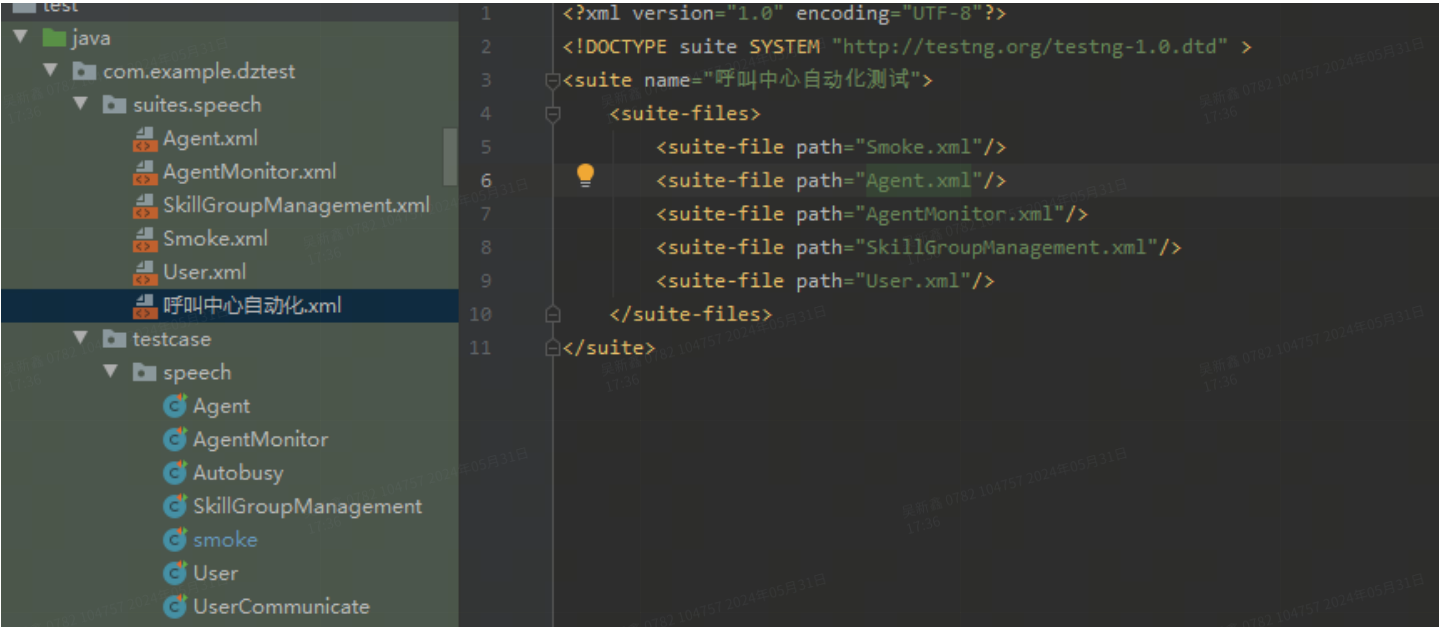
allure open [path of allure report]

#allure open ./tmp/allure

参考 <https://www.jianshu.com/p/aa9fbd2cdef4>

七.测试套管理

使用testng文件编排用例

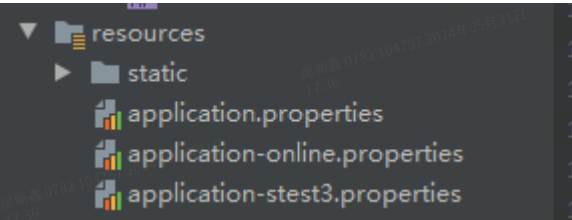


八.多环境支持

1.环境信息

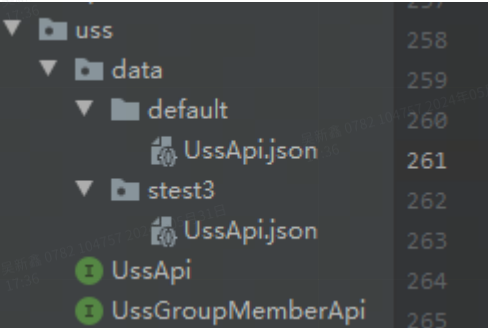
-Dspring.profiles.active=stest3

项目properties配置文件存放不同环境的配置数据



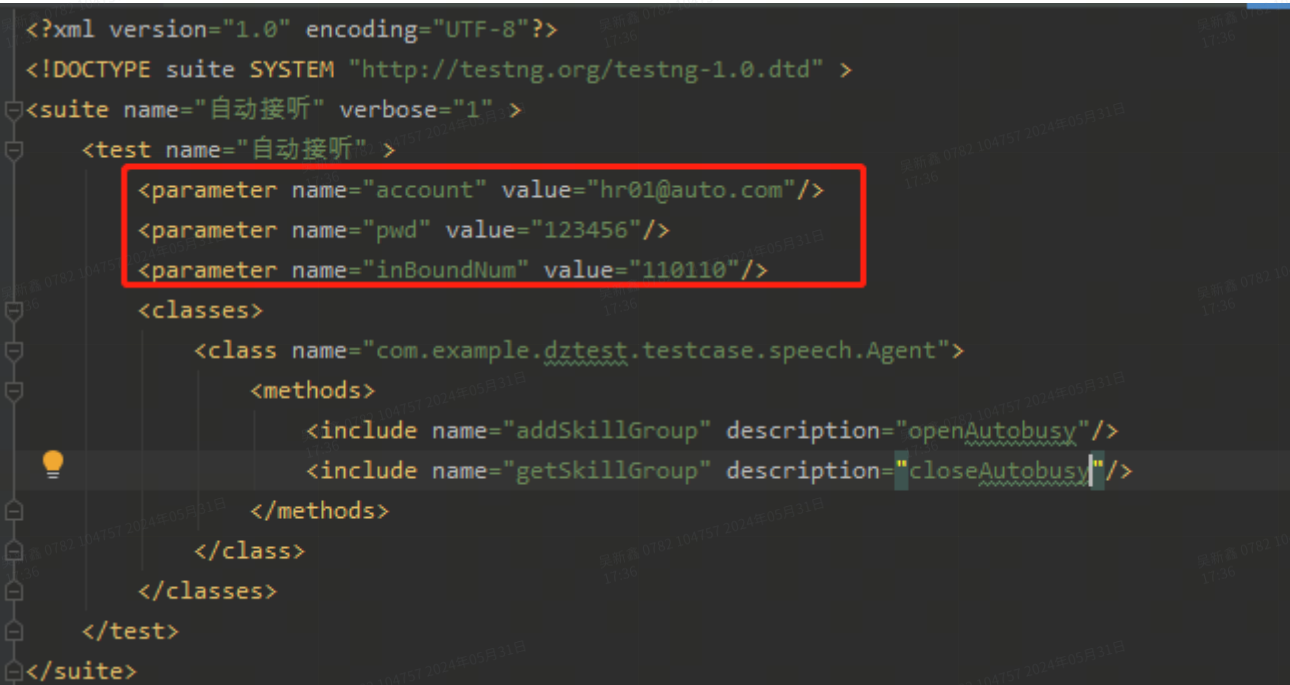
2.接口请求组装数据

json数据文件存放不同环境的接口请求组装数据

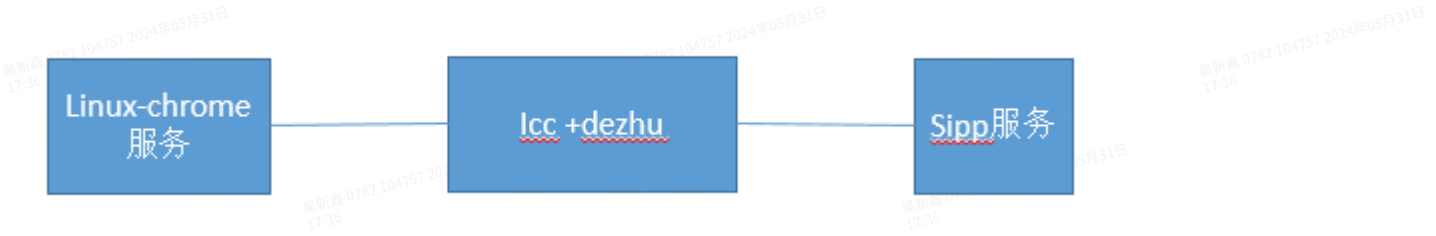


3.用例输入数据

testng测试套文件存放不同环境的用例输入数据



九.远程chrome驱动



通过修改项目properties配置文件决定使用本地chrome还是远程chrome

ui.mode = chrome_local #本地chrome

ui.mode = chrome_remote #远程chrome

```
public void init() {
    if ("chrome_remote".equals(mode)) {
        Log.info("mode:chrome_remote");
        this.driver = chromeDriverInit.initRemoteChromeDriver();
    } else {
        Log.info("" + chromeDriverInit);
        this.driver = chromeDriverInit.initLocalChromeDriver();
    }
    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
}
```