v2版本

解决问题：主要支撑接口自动化，适合简单场景的接口自动化。

QA1：为什么有自动化框架V1和V2？

智能交互大部分核心用例使用接口框架或平台，不能实现；只有使用V1版本实现。

简单接口能实现的用例场景，使用V1版本，成本较高；使用V2版本比较合适。

用例	复杂度	自动化方式	使用版本	用例步骤	实现方式
呼叫中心手动外呼	混合场景	接口、UI、SIPP	V1	前置处理：（1）获取通话总数 （2）注册模拟手机	接口/sipp
				1.呼出坐席登陆	UI
				2.拨号盘拨打电话	UI/SIPP
				3.等待用户接听电话	UI
				4.等待电话挂断	UI
				5.业务记录页，选择业务小结，填写服务备注，点击“完结”"	UI
				6.检查通话报表	接口
外呼机器人-应答逻辑	复杂场景	接口	V1	前置处理：（1）获取流程id （2）初始化会话	接口
				1.读取数据源文件，根据question获取机器人答案	接口/逻辑处理
				2.比对预期和实际结果	逻辑处理
				3.获得excel输出对应的robotSessionDataList	逻辑处理
用户的增删改查	简单场景	接口	V2	1.新增用户	接口
				2.查询用户	接口

				3.修改用户	接口
				4.删除用户	接口

外呼机器人-应答逻辑

原始数据 robot_qa.xlsx

输出数据 robot_qa202310251410984.xlsx

QA2：自动化目标是什么？

A. 提高质量； -->人工不好实现或容易忽略的检测

B. 提高效率。 -->重复执行

警惕：为了自动化而自动化。

目标	分类	案例	结果数据体现
A.提高质量	人工不好实现	（1）批量外呼，覆盖外呼链路各分支 （2）线上打断巡检，每10分钟执行一次，保持输入数据一致	版本测试：自动化发现bug数据/人工提交bug数 线上巡检：巡检bug数/线上P1以上bug数据
	人工容易忽略	（1）呼叫中心外呼，检验拨号盘每一个字段	
B.提高效率	重复执行	（1）呼叫中心核心用例	节省人力/人天

QA3：自动化用例编写的建议？

我们应该思考：单次执行成功-->多次执行成功-->多条执行成功-->多环境执行成功

单次执行成功：输入固定参数

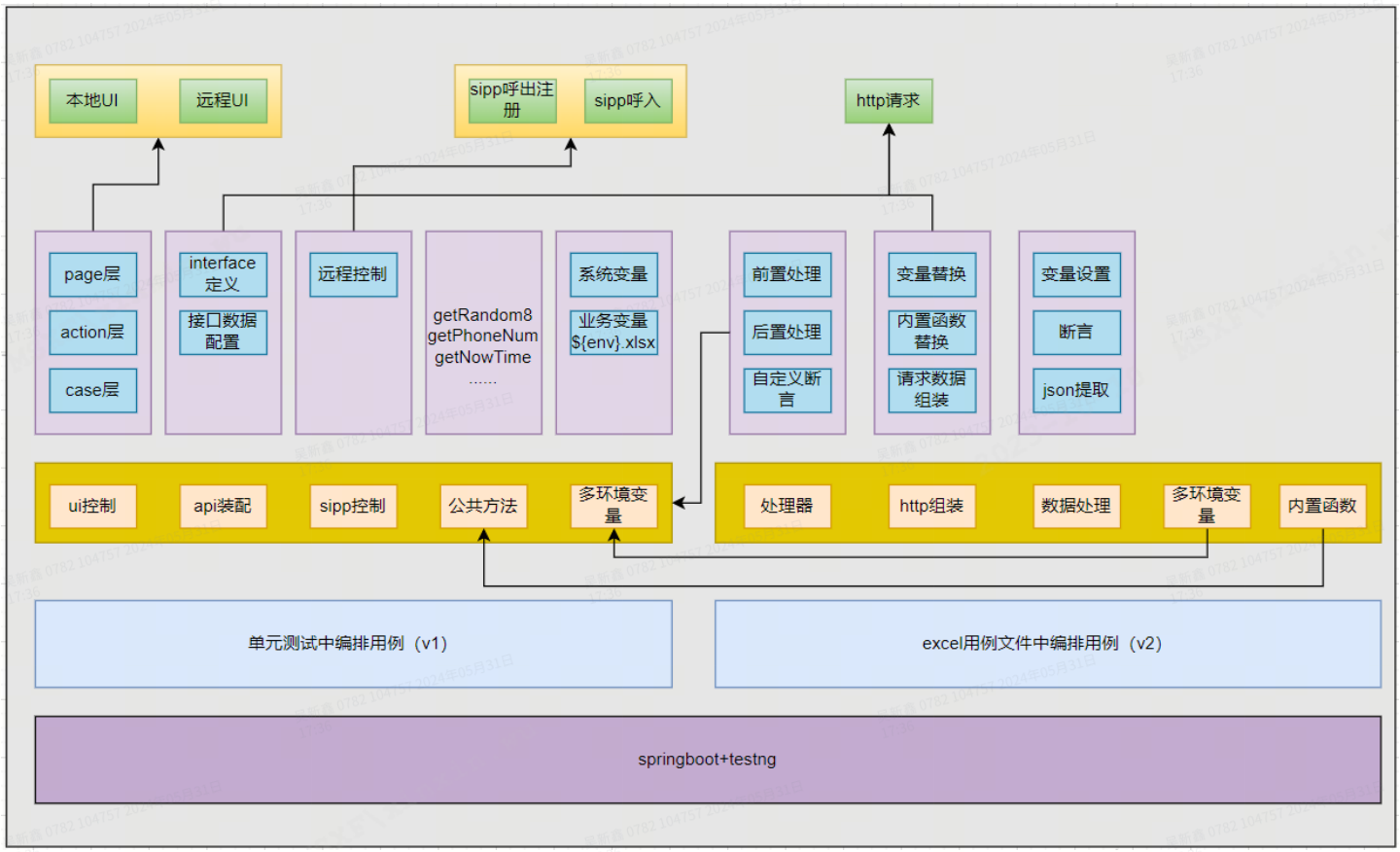
多次执行成功：排重参数设置为随机值

多条执行成功：除非是同测试套下用例，否则需要解耦用例

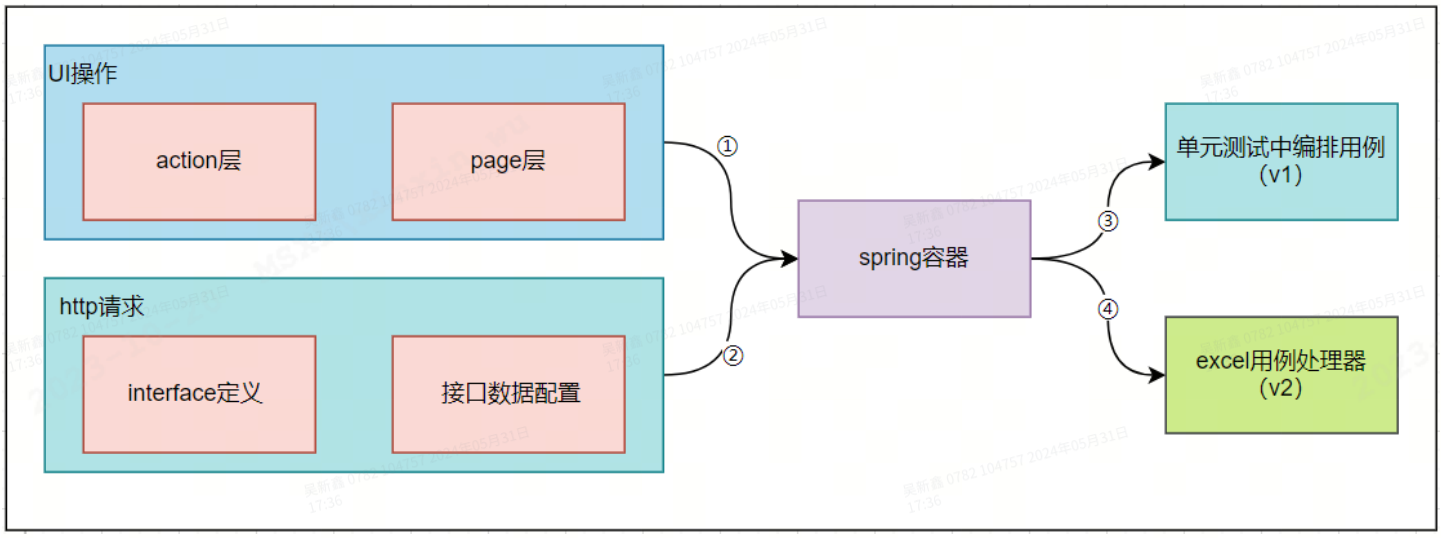
多环境执行成功：排除环境特定属性，如a环境有一个技能组A

一、自动化工程框架

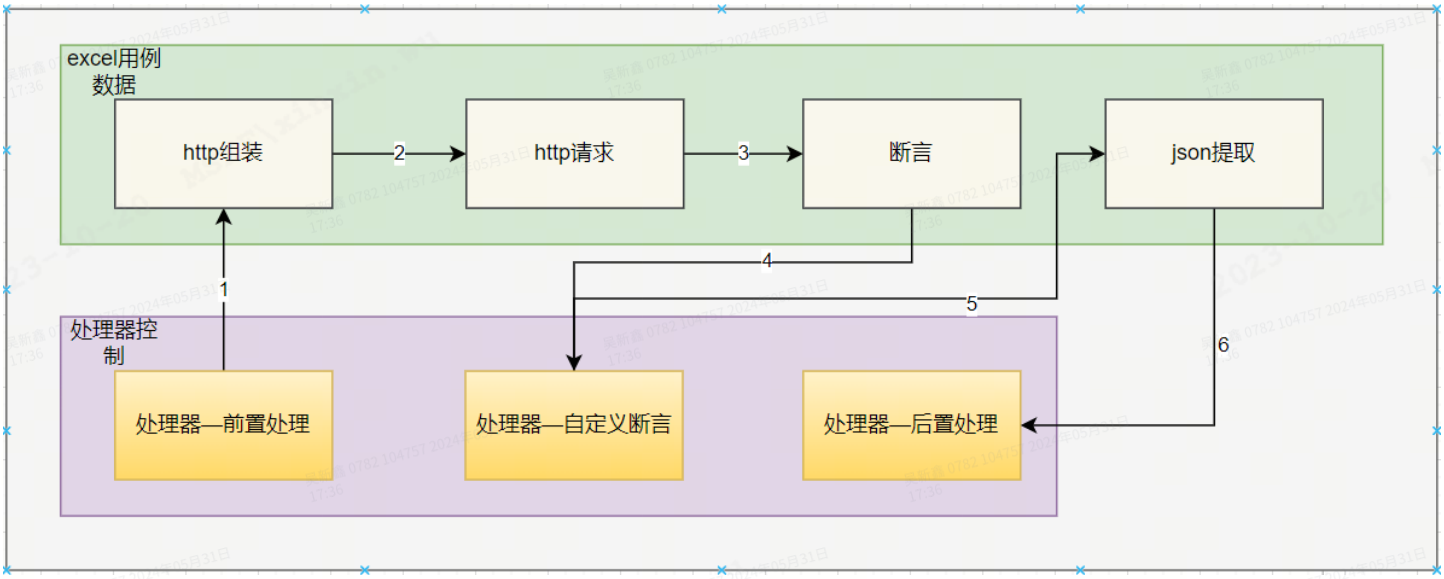
V1和V2整体框架



解耦业务能力和用例编排，注册、引用spring实例



v2版本excel单条用例执行顺序



二、V2版本使用

1.excel用例文件编排用例

编写模板 case-wxx-t01.xlsx

1.1 列名说明

用例名称	测试套	测试场景	处理器	变量设置	请
呼叫中心，新建技能组	呼叫中心技能组管理	呼叫中心，新建技能组	myCaseHandle	${__time(yyyyMMddHHMMSS)};mytest=helloword	post

列名	说明	是否必须	备注
用例名称	用例名称	必须	
步骤	用例执行步骤	非必须	
测试套	测试套说明	必须	
~~测试场景~~	~~测试场景说明~~	~~必须~~	
处理器	补充excel用例文件编写接口用例，可以在对应处理器beforeHandle、	非必须	

	afterHandle、assertion方法中用代码编写用例逻辑		
变量设置	设置用户变量，设置格式为”name1=value1;name2=value2“	非必须	
请求方式	post/get/put/delete	必须	
请求路径	请求路径	必须	
头信息	头信息设置，设置格式为”name1=value1;name2=value2“	非必须	
接口入参	（1）body体：{} （2）formdatas表单提交：name1=value1&name2=value2 （3）文件上传： [FilePath]:data/basic/员工导入模板.xlsx	非必须	
响应断言	响应断言，如$.code=10000	必须	
json提取	json提取，code=$.code	非必须	
需要运行	”Y/N“	必须	
优先级	”P0/P1/P2/P3“	必须	
添加人	添加人	必须	
备注	备注	非必须	

1.2 测试套内变量传递

用户变量只能在测试套内传递

（1）”变量设置“栏，设置的变量

如下，myname变量可以在”测试套demo“这个测试套中引用，但不能在”测试套test“中引用

测试套	变量设置	请求入参
测试套demo	myname=wxx	name=${myname}

测试套demo		name=${myname}
测试套test		

（2）"json提取"栏，提取的变量

测试套	接口入参	json提取
测试套demo		id=$.data.list[0].id
测试套demo	userId=${id}	

（3）处理器中，设置的用户变量

处理器myCaseHandler，前置处理中添加用户变量groupName=瓜瓜技能组

```
GlobalApiVar.userVars.put("groupName", "瓜瓜技能组");
```

excel用例文件中对应测试套，可以引用groupName用户变量

测试套	处理器	接口入参

测试套demo	myCaseHandler	{
		"name":"${groupName}",
		"speechAbility":true
		}

测试套demo		{
		"name1":"${groupName}",
		"speechAbility":true
		}

2.处理器

casehandle目录下，新建一个用例处理器子类，需要继承CaseHandler 父类

```
public class myCaseHandler extends CaseHandler {
......
```

```
}
```

我们在excel用例文件中"处理器"栏设置需要使用的处理器

用例名称	测试套	处理器	变量设置	请求方式	请求路径	头信息	
添加员工	用户管理	myCaseHandler	name=auto${__time(yyyyMMddHHMMSS)};email=auto${__time(yyyyMMddHHMMSS)}@msxf.com;phone=${__phone()};empno=${__random8()};id=;roleIds=${g.basic.roleId};departmentIds=	post	/basic/api/v1/user/sub	sign=test;tt=aa	name=${name}&=${email}&phorpno}&id=&roleIs=
搜索员工	用户管理			get	/basic/api/v1/user/ext/page/front?pageNo=1&pageSize=10&searchInfo=${email}&status=		

2.1 前置处理

在子类中，重写父类CaseHandler的beforeHandle方法，模板如下

```java
/**
 * @description: 前置处理，用于准备case数据等操作
 * @param
 * @return void
 */
@Override
public void beforeHandle() {
    super.beforeHandle();

    //前置处理逻辑
    System.out.println("myCaseHandler.beforeHandle");
    GlobalApiVar.userVars.put("test", "tttttt");
}
```

2.2 后置处理

在子类中，重写父类CaseHandler的afterHandle方法，模板如下

```java
/**
 * @description: 后置处理，用于case数据清理等操作
 * @param
 * @return void
 */
@Override
public void afterHandle() {
```

```
        super.afterHandle();

        //后置处理逻辑
        System.out.println("myCaseHandler.beforeHandle");
        //打印前置处理中定义的用户变量值
        logger.info("test is " + GlobalApiVar.userVars.get("test"));
    }
```

2.3 自定义断言

在子类中，重写父类CaseHandler的*assertion*方法，模板如下

```
/**
 * @description: 自定义断言
 * @param body  http请求返回body
 * @return void
 */
@Override
public void assertion(JSONObject body) {
    //自定义断言逻辑处理
    System.out.println("myCaseHandler.assertion");
    Assert.assertTrue(false);
}
```

2.4 回忆V1版本注册bean

ui操作、http请求、sipp操作都以单例模式注册到spring中，使用时直接引用即可。

（1）UI操作注册bean

`@Service` **注解标识ui操作（case层、page层）实现类， spring应用启动时会**自动创建对应的Bean实例并加入到 Spring 容器。

```
/*
* 登录页--登录操作
*/
@Service
public class ALoginPage {
    @Value("${portal.url}")
    private String url;

    @Autowired
    private LoginPage loginPage;

```

```
    @Autowired
    private UIDriver ui;

    //登陆
    public void login(String account, String pwd) {
        ui.open(url);
        loginPage.inputUserName(account);
        loginPage.inputPassword(pwd);
        loginPage.clickLogin();
    }
}
```

（2）http请求注册bean

以AgentApi.class为例，自动化框架启动后，会结合接口文件和请求数据文件组装一个bean实例注册到spring

请求数据文件AgentApi.json

```
{
  "getOutboundTaskCustomerList": {
    "param": {
      "pageNum": 1,
      "pageSize": 20,
      "query": "",
      "callStatus": "",
      "limitId": 0
    }
  }
}
```

接口文件AgentApi.class

```
@Mapping(path = "/speech", url = "")
public interface AgentApi {

    @Get(path = "/api/v1/agent/outboundTask/manual/customer/list/{value}", description = "坐席维度-获取外呼任务下关联的客户列表分页查询-手动外呼")
    HttpResponse getOutboundTaskCustomerList(@PathVariable("value") Long value);
```

```
6
7 }
```

（3）测试用例中引用bean

通过@Autowired注解，spring会自动注入

```
1  @SpringBootTest
2  @Feature("呼叫中心-坐席状态切换")
3  public class Smoke extends BaseOfTestCase {
4      //呼叫中心任务接口实例
5      @Autowired
6      private AgentApi agentApi;
7
8      //登录页操作实例
9      @Autowired
10     private ALoginPage aLoginPage;
11
12     @Test
13     public void agentStatusMonitor() {
14         //获取外呼任务下关联的客户列表分页查询
15         HttpResponse rsp = agentApi.getOutboundTaskCustomerList(100);
16         AssertUtil.assertBodyEquals(rsp, "$.code", "10000");
17     }
18     ......
19 }
```

2.5 获取spring管理的实例

在v2版本处理器中调用UI操作、http请求、sipp操作,我们通过BeanContextUtils.*getBean*(Class<T> clazz)手动获取bean实例。

```
1  public class DelUssGroupSpeechHandler extends CaseHandler {
2
3      //手动获取bean实例
4      private AgentApi agentApi = BeanContextUtils.getBean(AgentApi.class);
5
6       /**
7       * 前置处理，用于准备case数据等操作
8       */
```

```
    @Override
    public void beforeHandle() {
        //获取外呼任务下关联的客户列表分页查询
        HttpResponse rsp = agentApi.getOutboundTaskCustomerList(100);;
    }

    ......

}
```

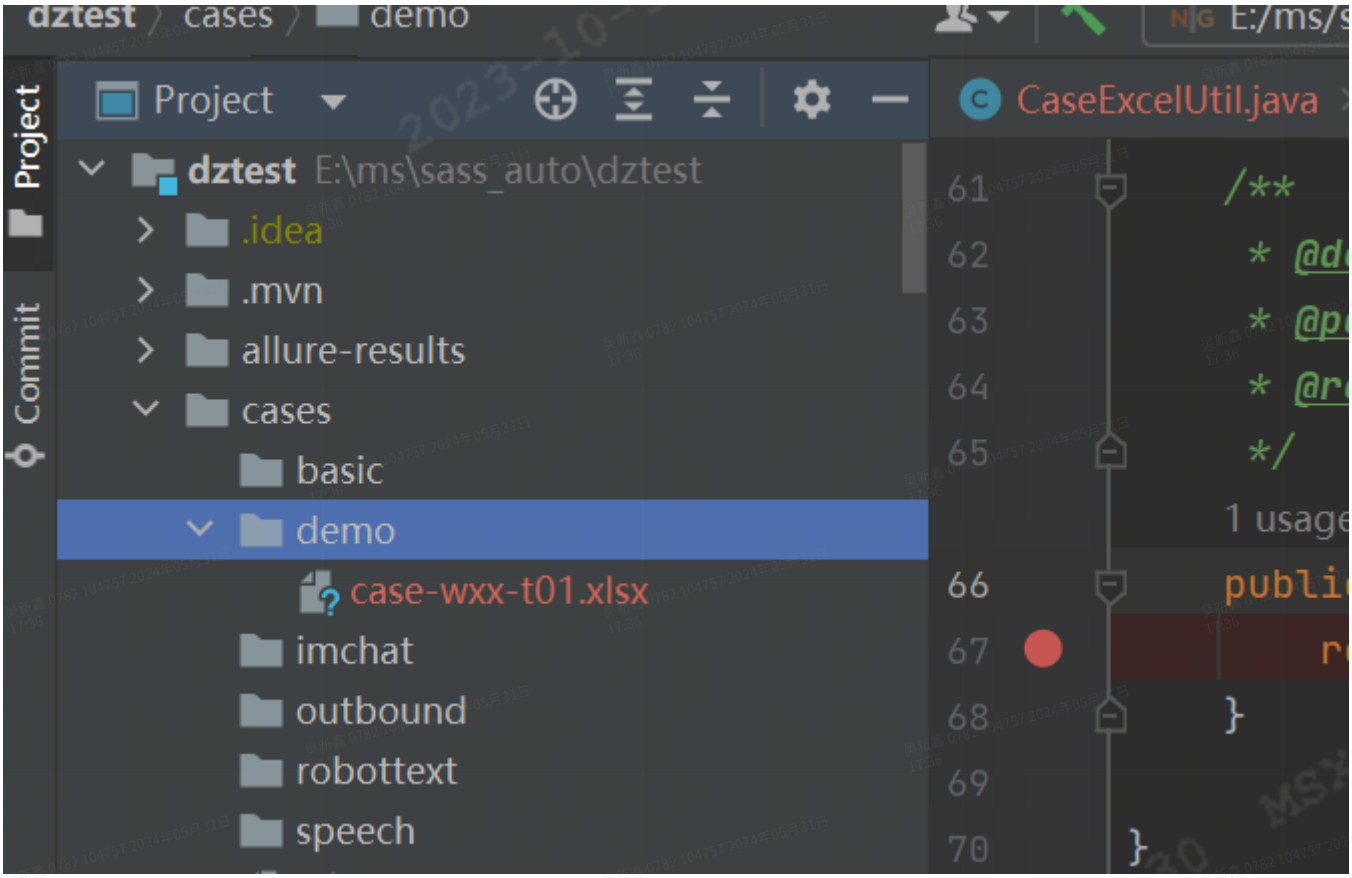
3.执行入口

以case-wxx-t01.xlsx为例

3.1 测试用例文件管理

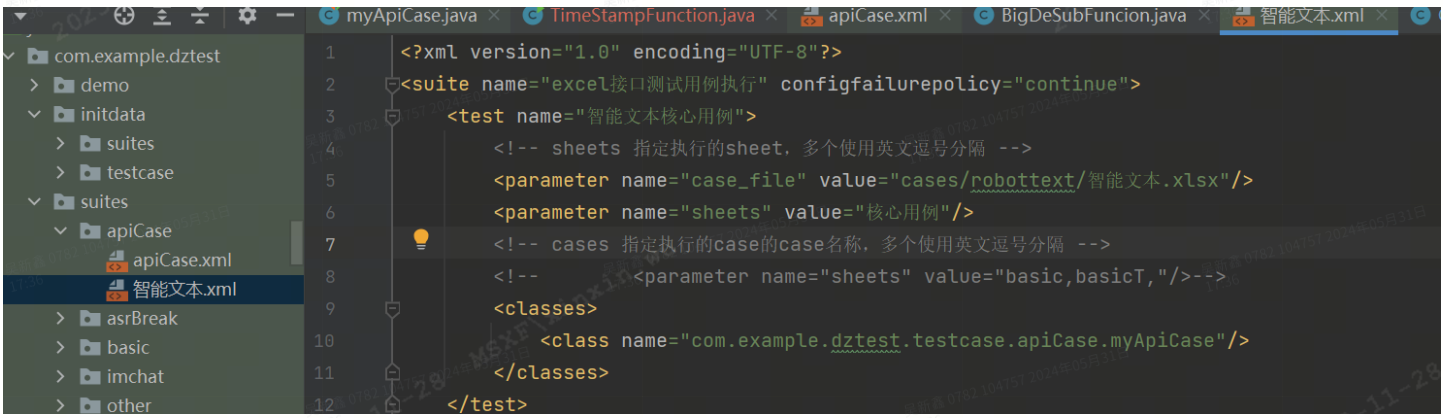
所以的测试用例文件放到cases目录

子目录	模块
demo	demo
basic	基础平台
imchat	在线客服
outbound	智能外呼
robottext	文本机器人
speech	呼叫中心



3.2 测试套入口

apiCase目录下新建测试套执行入口文件，如 智能文本.xml



根据自己的需求，组合测试套

```xml
<?xml version="1.0" encoding="UTF-8"?>
<suite name="excel接口测试用例执行" configfailurepolicy="continue">
    <test name="智能文本核心用例">
        <!-- sheets 指定执行的sheet，多个使用英文逗号分隔 -->
        <parameter name="case_file" value="cases/robottext/智能文本.xlsx"/>
        <parameter name="sheets" value="核心用例"/>
        <!-- cases 指定执行的case的case名称，多个使用英文逗号分隔 -->
        <!--        <parameter name="sheets" value="basic,basicT,"/>-->
```

```
        <classes>
            <class name="com.example.dztest.testcase.apiCase.myApiCase"/>
        </classes>
    </test>

<!--    <test name="号码管理">-->
<!--        &lt;!&ndash; sheets 指定执行的sheet，多个使用英文逗号分隔 &ndash;&gt;-->
<!--        <parameter name="case_file" value="cases/case-wxx-t01.xlsx"/>-->
<!--        <parameter name="sheets" value="basicT"/>-->
<!--        &lt;!&ndash; cases 指定执行的case的case名称，多个使用英文逗号分隔 &ndash;&gt;-->
<!--        &lt;!&ndash;        <parameter name="cases" value="submitCashApply_AO,getCashApplyResult_AO,"/>&ndash;&gt;-->
<!--        <classes>-->
<!--            <class name="com.example.dztest.testcase.apiCase.myApiCase"/>-->
<!--        </classes>-->
<!--    </test>-->

    <listeners>
        <listener class-name="com.example.dztest.apicase.ReporterListener.ExtendTestNGIReporterListenerNew"></listener>
    </listeners>
</suite>
```

指定执行文件的是下面2行

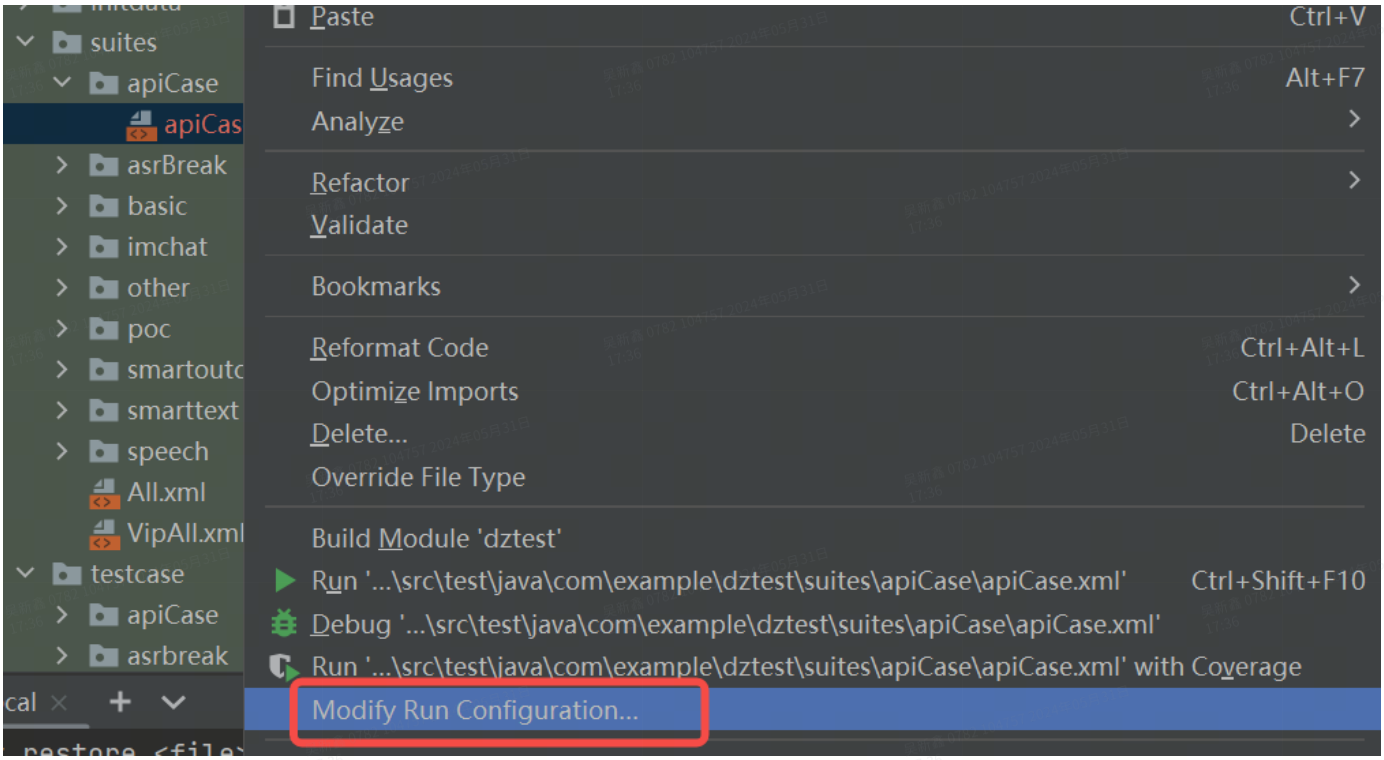
<parameter name="case_file" value="cases/robottext/智能文本.xlsx"/>
<parameter name="sheets" value="核心用例"/>

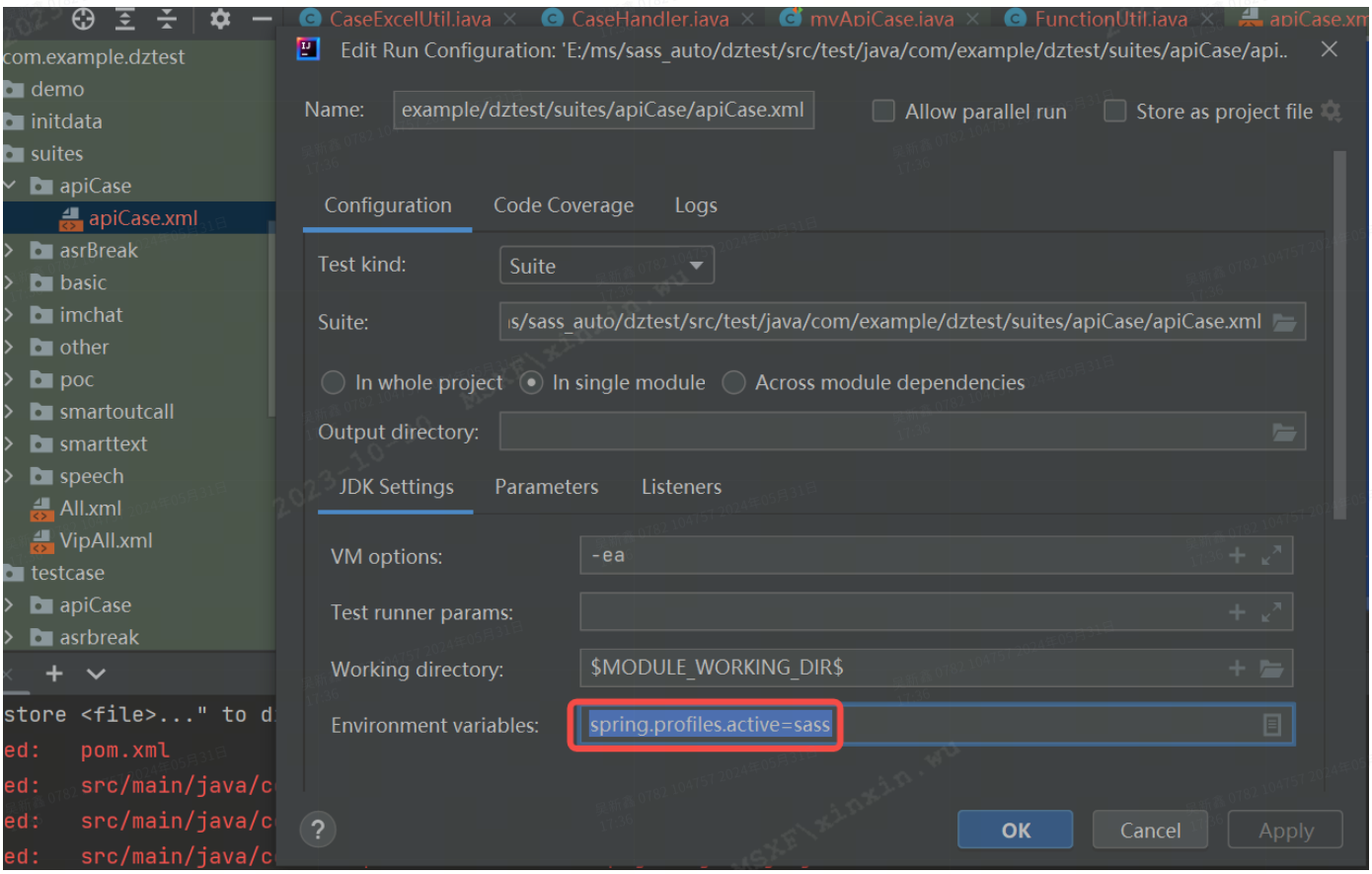
case_file 保存用例的excel文件

Sheets 用例文件中的sheet页

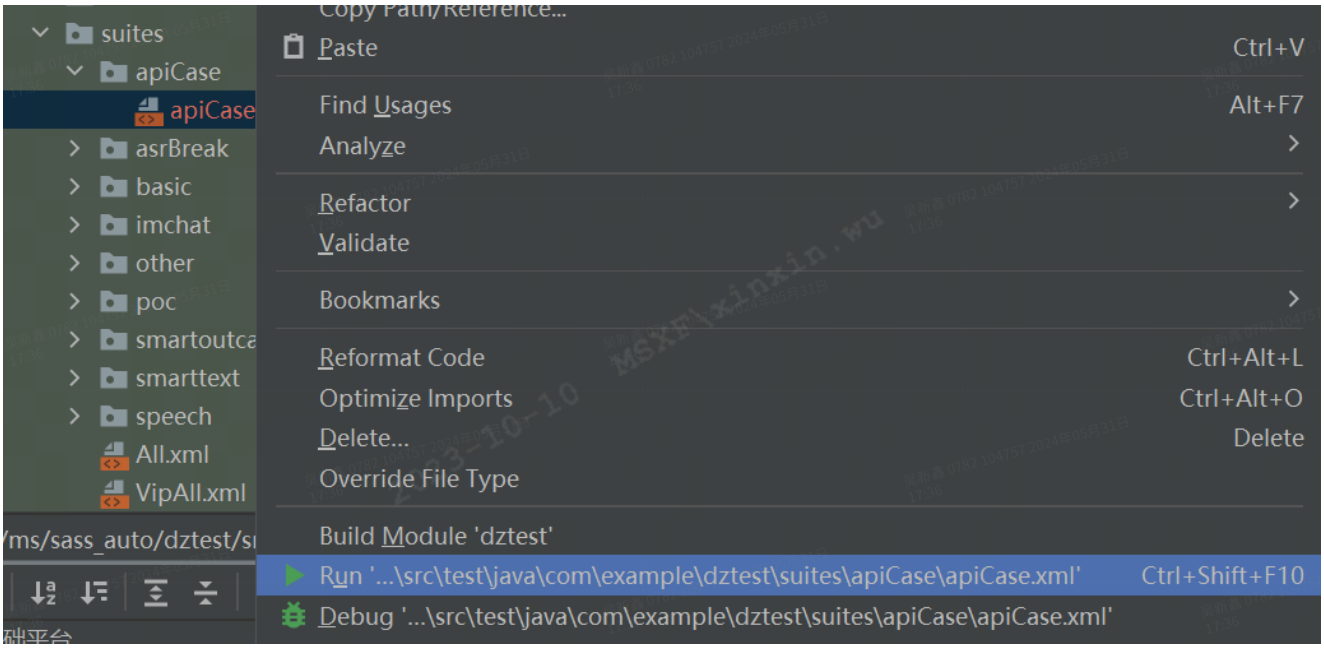
3.3 用例执行

设置执行环境


suites
apiCase
apiCas
asrBreak
basic
imchat
other
poc
smartoutc
smarttext
speech
All.xml
VipAll.xml
testcase
apiCase
asrbreak
Paste Ctrl+V
Find Usages Alt+F7
Analyze
Refactor
Validate
Bookmarks
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Override File Type
Build Module 'dztest'
Run '...\src\test\java\com\example\dztest\suites\apiCase\apiCase.xml' Ctrl+Shift+F10
Debug '...\src\test\java\com\example\dztest\suites\apiCase\apiCase.xml'
Run '...\src\test\java\com\example\dztest\suites\apiCase\apiCase.xml' with Coverage
Modify Run Configuration...

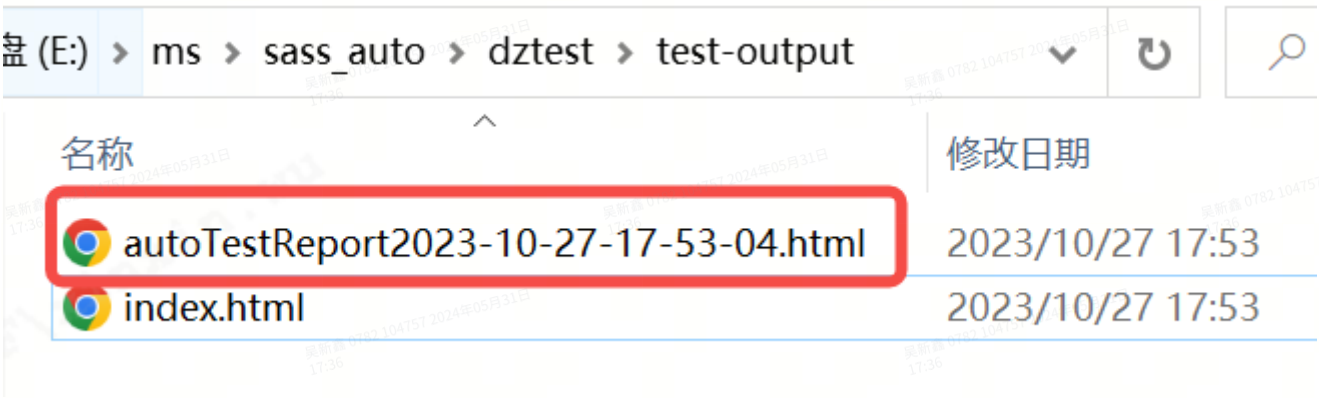

Edit Run Configuration: 'E:/ms/sass_auto/dztest/src/test/java/com/example/dztest/suites/apiCase/api..
Name: example/dztest/suites/apiCase/apiCase.xml
Allow parallel run
Store as project file
Configuration
Code Coverage
Logs
Test kind: Suite
Suite: s/sass_auto/dztest/src/test/java/com/example/dztest/suites/apiCase/apiCase.xml
In whole project
In single module
Across module dependencies
Output directory:
JDK Settings
Parameters
Listeners
VM options: -ea
Test runner params:
Working directory: $MODULE_WORKING_DIR$
Environment variables: spring.profiles.active=sass
OK
Cancel
Apply

右键执行

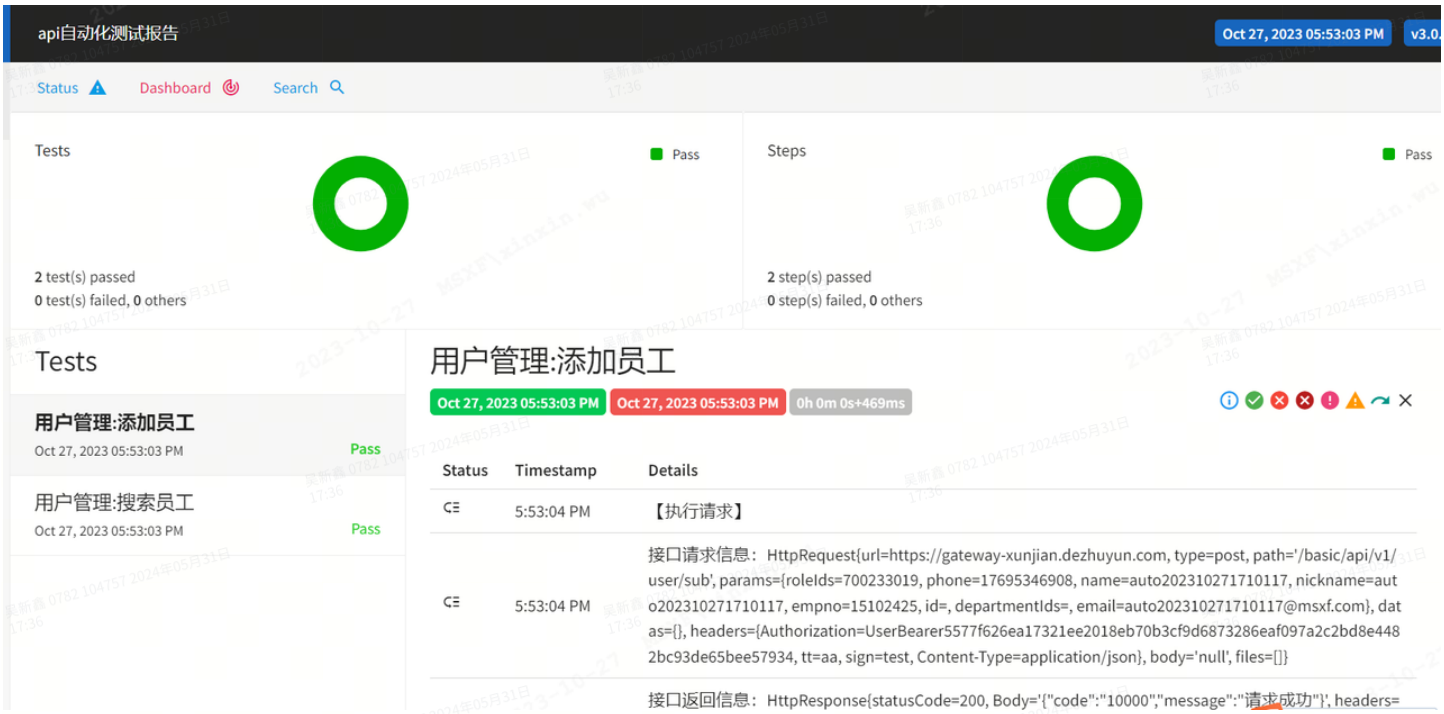


4.测试报告

在test-output目录下生产测试报告文件autoTestReport**.html



打开测试报告文件如下



三、V2版本核心实现

1.变量

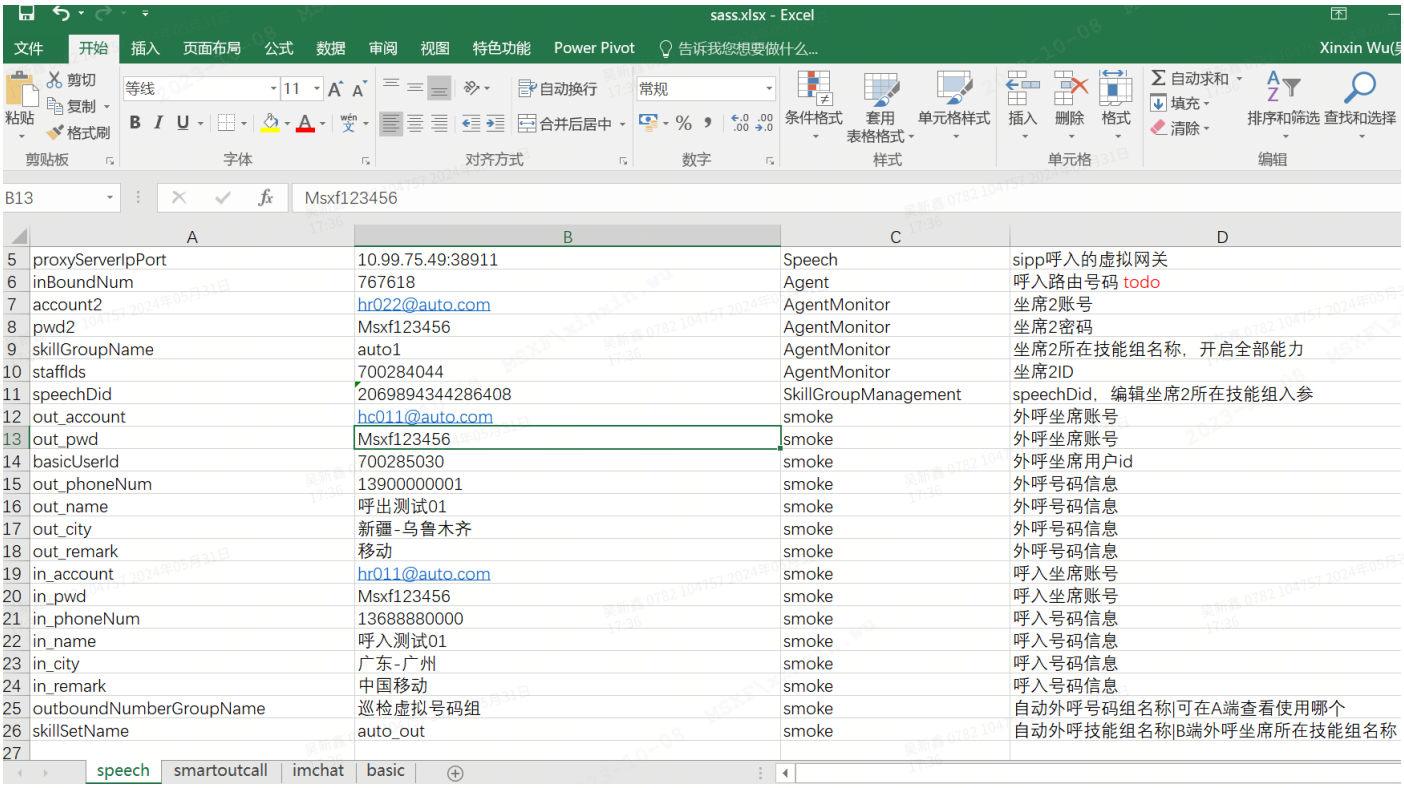
1.1 全局变量

生命周期：整个用例执行过程

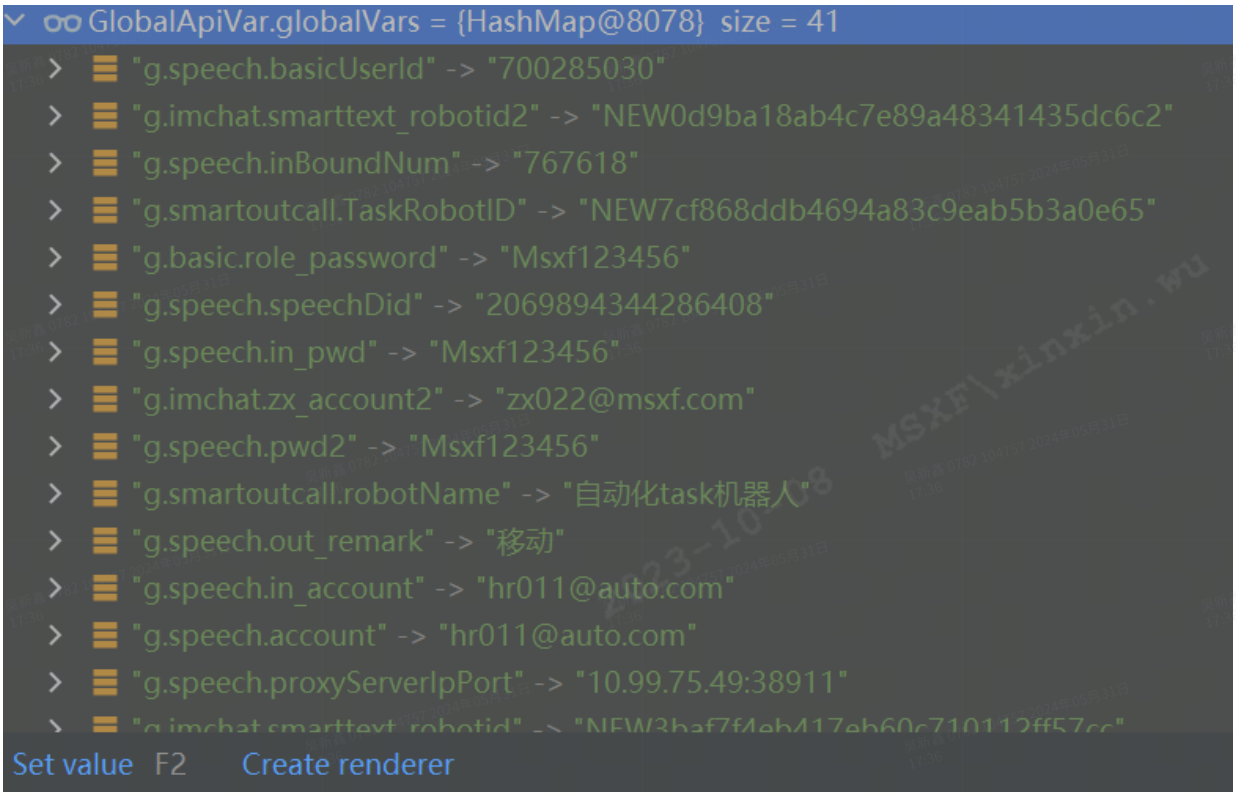
（1）excel变量文件，定义全局变量

从对应环境Excel变量文件读取，写入全局变量池GlobalApiVar.*globalVars*

（使用v1版本的excel业务变量文件）



	A	B	C	D
5	proxyServerIpPort	10.99.75.49:38911	Speech	sipp呼入的虚拟网关
6	inBoundNum	767618	Agent	呼入路由号码 todo
7	account2	hr022@auto.com	AgentMonitor	坐席2账号
8	pwd2	Msxf123456	AgentMonitor	坐席2密码
9	skillGroupName	auto1	AgentMonitor	坐席2所在技能组名称，开启全部能力
10	staffIds	700284044	AgentMonitor	坐席2ID
11	speechDid	2069894344286408	SkillGroupManagement	speechDid，编辑坐席2所在技能组入参
12	out_account	hc011@auto.com	smoke	外呼坐席账号
13	out_pwd	Msxf123456	smoke	外呼坐席账号
14	basicUserId	700285030	smoke	外呼坐席用户id
15	out_phoneNum	13900000001	smoke	外呼号码信息
16	out_name	呼出测试01	smoke	外呼号码信息
17	out_city	新疆-乌鲁木齐	smoke	外呼号码信息
18	out_remark	移动	smoke	外呼号码信息
19	in_account	hr011@auto.com	smoke	呼入坐席账号
20	in_pwd	Msxf123456	smoke	呼入坐席账号
21	in_phoneNum	13688880000	smoke	呼入号码信息
22	in_name	呼入测试01	smoke	呼入号码信息
23	in_city	广东-广州	smoke	呼入号码信息
24	in_remark	中国移动	smoke	呼入号码信息
25	outboundNumberGroupName	巡检虚拟号码组	smoke	自动外呼号码组名称|可在A端查看使用哪个
26	skillSetName	auto_out	smoke	自动外呼技能组名称|B端外呼坐席所在技能组名称
27				



（2）代码，定义全局变量

```
GlobalApiVar.globalVars.put("g.jvm.speech.groupId", "myValue");
```

如上，变量名为g.jvm.speech.groupId，变量值为myValue；

我们约定代码中定义全局变量，变量名规范满足“g.jvm.XX”；g.jvm代表在代码中定义的全局变量

（3）excel用例文件，引用全局变量

引用excel变量文件中的全局变量

${g.speech.speechDid}

引用代码中定义全局变量

${g.jvm.speech.groupId}

1.2 用户变量

生命周期：单个测试套

（1）excel用例文件，定义用户变量

a.变量设置 栏

groupName=技能组${__time(yyyyMMddHHMMSS)};mytest=helloword

如上，设置groupName、mytest 2个用户变量

b.json提取 栏

id=$.data.list[0].id

如上设置id用户变量

（2）代码，定义用户变量

```
GlobalApiVar.userVars.put("jvm.groupId", "myValue");
```

如上，变量名为jvm.groupId，变量值为myValue

我们约定代码中定义用户变量，变量名规范满足“jvm.XX”；jvm.代表在代码中定义的用户变量

（3）excel用例文件，引用用户变量

引用excel变量文件中的用户变量

${groupName}

引用代码中定义用户变量

${jvm.groupId}

2.内置函数

由专项人员编写，编写用例时直接引用

2.1 内置函数定义

excel用例文件编写过程中，有时需要函数生成参数值，如获取当前时间戳、获取一个随机手机号

函数生成的请求参数格式参考了Jmeter的函数格式，***__funName(arg1,arg2,…)***

首先要在程序里定义好内置函数接口

```java
/**
 * @author xinxin.wu
 * @description: 内置函数接口
 * @date 2023/10/18
 * @version: 1.0
 */
public interface Function {

    /**
     * @param args 函数的入参
     * @return String   函数执行结果
     * @description: 函数的具体执行
     */
    String execute(String[] args);

    /**
     * @return String   函数的名字
     * @description: 获取函数名字
     */
    String getReferenceKey();
}
```

所有的内置函数都要去实现Function接口

如下方，定义获取当前时间戳的内置函数；实现execute方法和getReferenceKey方法，execute方法是内置函数具体执行逻辑，getReferenceKey方法返回内置函数名。

定义time内置函数后，我们可以在excel用例文件中引用，如${__time(yyyyMMddHHMMSS)}

```java
/**
 * @author xinxin.wu
 * @description: 内置函数，获取当前时间戳
 * @date 2023/10/19
 * @version: 1.0
 */
public class TimeFunction implements Function {
    @Override
    public String execute(String[] args) {
        String dateStyple = args[0];
        return TimeUtil.getNowTime(dateStyple);
    }

    @Override
    public String getReferenceKey() {
        return "time";
    }

}
```

2.2 内置函数使用

定义好内置函数之后，我们规定在excel用例文件中用__funName(arg1,arg2,…)这种特殊格式的字符串去标识对应的内置函数。

groupName=${__time(yyyyMMddHHMMSS)}

3.http组装

3.1 变量替换

excel用例文件中，“变量设置、请求路径、头信息、接口入参、响应断言”栏都可能引用变量。

excel用例文件中，引用变量参考Jmeter引用变量的格式，如下

请求路径

```
/basic/api/v1/user/ext/page/front?
pageNo=1&pageSize=10&searchInfo=${email}&status=${status}
```

数据处理时，替换引用变量：匹配到变量名之后，以变量名作key去公共参数池中取值并替换掉${xx}。

```java
/**
 * 截取变量正则表达式：${xx}
 */
protected static Pattern replaceParamPattern = Pattern.compile("\\$\\{([^_][^_].*?)\\}");

/**
 * @param param 要替换变量的字符串
 * @return String   替换${xxx}后的字符串
 * @description: 将参数param中的${xxx}替换为公共参数池map中以${xxx}中的xxx作key的value,并返回处理过的参数
 */
public static String getCommonParam(String param) {
    //如果参数为空，返回空字符串
    if (null == param || "".equals(param)) {
        return "";
    }
    //这个正则表达式模式匹配的是${}
    Matcher m = replaceParamPattern.matcher(param);// 取公共参数正则
    while (m.find()) {
        //取匹配到到的所有子字符串中的第一个分组，其实这个正则表达式就只有一个分组，( ${}  )
        String replaceKey = m.group(1);
        // 从公共参数池中获取值，以${xxx}中的xxx作key,取key对应的value
        String value = getSaveData(replaceKey);
        // 如果公共参数池中未能找到对应的值，该用例失败。
        Assert.assertNotNull(value, String.format("格式化参数失败，公共参数中找不到%s。", replaceKey));
        //将参数中的${xxx}替换为公共参数池中的值
        param = param.replace(m.group(), value);
    }
    return param;
}
```

3.2 内置函数替换

excel用例文件中，“变量设置、请求路径、头信息、接口入参、响应断言”栏都可能引用内置函数。

excel用例文件中，引用内置函数参考Jmeter引用内置函数的格式，如下

变量设置
name=auto${__time(yyyyMMddHHMMSS)};email=auto${__time(yyyyMMddHHMMSS)}@msxf.com;phone=${__phone()}

数据处理时，替换引用内置函数：

a.借助反射机制，根据函数名和参数值获取到函数执行结果

b.以函数返回值替换${__xx}

```
/**
 * 截取自定义方法正则表达式: ${__xxx(param1,param2,...)}
 */
protected static Pattern funPattern = Pattern.compile("\\$\\{__(\\w*?)\\((([\\w\\\\\\/:\\.\\$]*,?)*)\\)\\}");

/**
 * @param param 要替换内置函数及变量的字符串
 * @return String   替换内置函数和变量后的字符串
 * @description: 处理内置函数${__fucn(param1,param2,...)}以及变量${xxxx}
 */
public static String buildParam(String param) {
    // 处理${xxx}
    param = getCommonParam(param);

    //这个funPattern是指${__fun(xxx,...)}的正则匹配模式
    Matcher m = funPattern.matcher(param);
    while (m.find()) {
        //匹配到的子字符串的第一个分组是(\w*?)，也就是函数名
        String funcName = m.group(1);
        //匹配到的子字符串的第二个分组是(([\w\\\/:\$\.]*,?)*)，也就是param1,param2...
        String args = m.group(2);
        String value;
        // bodyfile属于特殊情况，不进行匹配，在post请求的时候进行处理
        if (FunctionUtil.isFunction(funcName)) {
            // 属于函数助手，调用那个函数助手获取。
            value = FunctionUtil.getValue(funcName, args.split(","));
            // 解析对应的函数失败
            Assert.assertNotNull(value, String.format("解析函数失败：%s。", funcName));
            param = StringUtils.replace(param, m.group(), value);
        }
```

```
    }
    return param;
}
```

3.3 请求数据组装

根据接口用例数据组装http请求

```
/**
 * @description: 根据单条用例数据实体，设置http请求数据
 * @param
 * @return void
 */
public void initHttpRequest() {
    this.httpRequest = new HttpRequest();
    this.softAssert = new SoftAssert();

    //设置请求方式
    switch (this.apiCaseModel.getHttpMethod()) {
        case "post":
            this.httpRequest.setType(HttpType.POST);
            break;
        case "get":
            this.httpRequest.setType(HttpType.GET);
            break;
        case "put":
            this.httpRequest.setType(HttpType.PUT);
            break;
        case "delete":
            this.httpRequest.setType(HttpType.DELETE);
            break;
        default:
            logger.error("不能匹配请求方式post/get/delete/put");
    }

    //获取原始请求入参
    String param = this.apiCaseModel.getParams();
    //替换变量和函数
    String param_p = CaseExcelUtil.buildParam(param);
    //设置body 或 param
    CaseExcelUtil.setParams(param_p, this.httpRequest);

    //设置path和param
    String path_p = CaseExcelUtil.buildParam(this.apiCaseModel.getPath());
    CaseExcelUtil.setPath(path_p, this.httpRequest);
```

```
38
39      //设置server
40      httpRequest.setServer_url(ProxyUtils.server_url);
41
42      //设置headers
43      CaseExcelUtil.setHeaders(this.apiCaseModel.getHeaders(), httpRequest);
44
45  }
```

4.数据处理

4.1 变量设置

变量设置
phone=${__phone()};empno=${__random8()};num=${u_num}

如上，数据处理后，我们能获取到的用户变量有：

phone=1390000001;empno=24567800;num=223 (value是假设值)

替换引用的变量和内置函数，并将(key, value)添加到用户变量池

```
1  /**
2   * @param varsSet 变量设置栏数据，如groupName=技能组
   ${__time(yyyyMMddHHMMSS)};mytest1=${id};mytest=helloword
3   * @return void
4   * @description: 处理”变量设置栏数据“: 1.替换变量和函数 2.根据设置添加用户变量
5   */
6  public static void setUserVars(String varsSet) {
7      if (null == varsSet || "".equals(varsSet)) {
8          return;
9      }
10
11     //替换变量和函数
12     String varsSet_p = buildParam(varsSet);
13
14     Pattern pattern = Pattern.compile("([^;=]*)=([^;]*)");
15     Matcher m = pattern.matcher(varsSet_p.trim());
16
17     //phone=18191992233;empno=990033&id=
18     //利用;将字符串分隔开来
```

```
        String[] vars = varsSet_p.split(";");

        //遍历save  msg_id=$.jsonPath.xx;
        for (String var : vars) {
            // 正则表达式 由0-n个非; =组成的字符串 =  0-n个非;组成的字符串
            //对msg_id=ssss做正则匹配
            Matcher matcher = pattern.matcher(var.trim());
            while (matcher.find()) {
                String key = matcher.group(1);
                String value = matcher.group(2);

                //将key value加到用户变量map
                GlobalApiVar.userVars.put(key, value);
            }
        }
    }
```

4.2 断言

断言
$.code=10000;$.data.list[0].phone=${phone}

通过jsonpath获取接口返回body体对应字段值，这里是$.code和$.data.list[0].phone；然后与value值判断是否相等。

```
    /**
     * @description: case断言
     * @param
     * @return void
     */
    public final void assertion() {
        //excel中响应断言
        Map<String, String> assertMap = CaseExcelUtil.getAssertMap(this.apiCaseModel.getAssertion());
        assertMap.forEach((key,value)->{
            String actual_value = JsonPath.read(this.bodyRespone, key);
            String expect_value = value;
            Reporter.log(key + " 期望值:" + expect_value + "; 实际值: " + actual_value, true);
            Assert.assertEquals(expect_value, actual_value);
        });

```

```
    //casehandler中自定义断言
    assertion(this.bodyRespone);

    this.softAssert.assertAll();
}
```

4.3 json提取

json提取
id=$.data.list[0].id

通过jsonpath获取接口返回body体对应字段值，这里是$.data.list[0].id；然后将(key, value)添加到用户变量池

```
/**
 * @description: 提取json串中的值添加为用户变量
 * @param body  json响应
 * @param allSave   json提取字符串，如msg_id=$.jsonPath.xx
 * @return void
 */
public static void saveResult(JSONObject body, String allSave) {
    if (null == body || null == allSave
            || "".equals(allSave)) {
        return;
    }

    //first_currency=$.result[0].currency;first_name=$.result[0].name
    //将excel里save  msg_id=$.jsonPath.xx; 字符串分隔开来
    String[] saves = allSave.split(";");

    //遍历save  msg_id=$.jsonPath.xx;
    for (String save : saves) {
        // 正则表达式 由0-n个非; =组成的字符串 =  0-n个非;组成的字符串
        //对msg_id=$.jsonPath.xx做正则匹配
        Pattern pattern = Pattern.compile("([^;=]*)=([^;]*)");
        Matcher m = pattern.matcher(save.trim());
        while (m.find()) {
            //将excel save字段里等式左边的key作为key
            //msg_id是匹配到到第一个分组group(1)
            //对左边调用getBuildValue函数可能只起到一个trim的作用
            String key = m.group(1);
            //根据excel里save字段的jsonPath提取响应里对应着jsonPath的值作为value
```

```
29                  //$.jsonPath.xx是匹配到到第二个分组group(2)
30                  String jsonPath = m.group(2);
31                  Object value = JsonPath.read(body, jsonPath);
32
33  //                  ReportUtil.log(String.format("存储公共参数    %s值为: %s.", key,
    value));
34                  GlobalApiVar.userVars.put(key, value);
35              }
36          }
37      }
```