

# 玩转小成本自动化设计

教你如何快速搭建中小测试团队的自动化框架平台

论坛ID : seagull1985



# 目录

CONTENTS

1

面临的困境

2

自动化模型设计

3

接口&UI自动化的实现封装

4

LuckyFrame演示





# PART 01

---

## 中小团队的自动化困境

# 我们的困境

## 用什么开发语言？

自动化主流开发语言，  
JAVA&PYTHON应该怎么  
选择，哪种更适合我们公司。

## 怎么选择框架？

目前市场测试平台&框架开  
源还有商业的都有很多，要  
怎么选择，要不要选择一款  
商业的测试平台。

## 选什么项目当试验田？

应该选一个什么样的项目当  
试验田，能更好的迈出自动  
化的第一步。

## 团队怎么分配工作？

忙不完的功能测试任务，根  
本没时间去开展自动化相关  
的工作。

## 自动化的目标是什么？

不知道自动化到底要做到什  
么程度，我们能做到什么程  
度，目标好模糊。

## 怎么得到领导认可与支持？

要投入人力资源.可能影响现  
有的工作量以及节奏，领导  
可能会不认可。

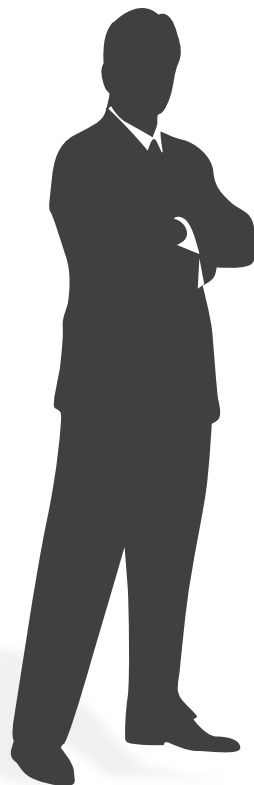
# 自动化瓶颈



- 01 代码能力  
( java/vbs/python )
- 02 技术方向 ( 如开源框架  
的掌握程度 )



- 01 回报周期长
- 02 人力成本高



- 01 团队规模小、任务重
- 02 缺少技术方向规划



- 01 项目流程不规范
- 02 项目变动大，周期短等

# 主流自动化开发语言

50%

从测试行业自动化的语言应用范围来说，python稍占优势

Python

自动化开发语言占比 %

40%

个人感受会更推荐使用java来做自动化

Java

自动化开发语言占比 %



## Python

支持主流自动化开源框架，如webdriver\appium等。

- ✓ 开发快速、简单易学，易学程度以及开发速度远远甩开JAVA三条街。
- ✓ 国内相关中文资源少，自学渠道匮乏，遇到问题难通过自行查找资料解决。



## Java

支持主流自动化开源框架，如webdriver\appium等。目前国内JAVA项目占比较高，更容易应用到接口测试。

- ✓ 运行速度快，可用资源相当丰富，扩展性好，JAVA相关的类库非常多。
- ✓ 遇到问题求教渠道非常多，也许您身边的开发用的都是JAVA，当面沟通问题比搜索引擎靠谱。

# 解决困境从这里入手（一）



# 解决困境从这里入手（二）

## 技术薄弱

利用现成的开源框架  
选择更容易掌握的开发语言



## 团队规划

Team Lead  
侧重管理以及技术研究工作



## 项目选择

接口自动化入手  
稳定长期的项目  
配合度较高  
能较快的体现自动化价值





# PART 02

## 自动化模型设计

# 测试分层属性

- ◆ excel
- ◆ 数据库
- ◆ Xml
- ◆ ...

数  
据



用  
例

- ◆ Testlink
- ◆ Excel
- ◆ 关键字管理

- ◆ API JAVA
- ◆ PYTHON
- ◆ Selenium
- ◆ Appium

脚  
本



框  
架

- ◆ 调用API封装
- ◆ 日志读写
- ◆ 流程组织
- ◆ 多线程



# 自动化的基本分层

## 自动化分层模型



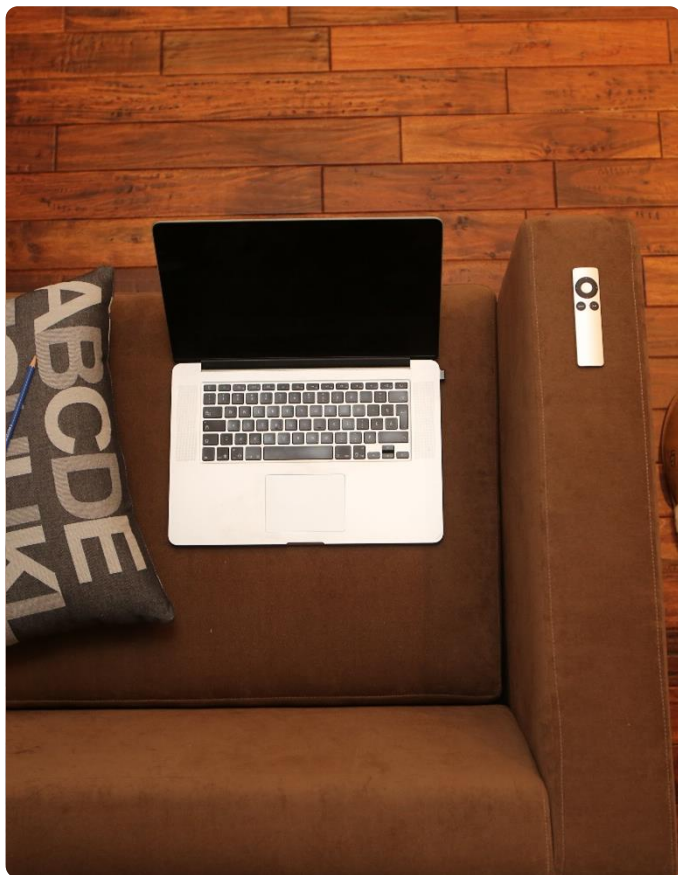
TestLink&API  
关键字驱动

JAVA API测试驱动桩  
封装Selenium\Applum

LuckyFrame



# 三步小框架



## JAVA反射机制

### method.invoke

JAVA反射机制是在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法;对于任意一个对象，都能够调用它的任意一个方法和属性;这种动态获取的信息以及动态调用对象的方法的功能称为java语言的反射机制。

## 封装开源测试框架

### Selenium/Appium

对开源测试框架中的API进行封装，这样可以通过关键字驱动UI测试，免去UI测试中需要自己编码脚本的困难，更简单的通过关键字完成自动化。

## 用例管理

### TestLink API

通过TestLink管理测试用例，并通过API对用例中的要素取值进行解析，拿到关键字以及驱动数据。管理更方便清晰科学。

# TestLink的关键字管理

Interface

范围内

您不能编辑这个测试用例

系统设置不允许

版本 1

摘要

前提

#	步骤动作
1	com.test.settle.ClearingBillTest#costRebatesAmt; 20130601 100

状态：草稿

测试方式：自动的

测试执行时间 (分钟)：

Web UI

#	步骤动作
1	Open(10.213.23.35:8080/ysuser_manager/)
2	id=username sendKeys(ysboss) 15*wait;
3	id=loginbutton click 3*wait;

创建步骤

重新排序步骤

状态：草稿

测试方式：自动的

测试执行时间 (分钟)：

保存

Interface

期望的结果

1	yspos.online.test.Publictest#executeSql; update URMTMINF t set t.spec_disc_tp = '04' where t.merc_id = '826440389318017	1
2	yspos.online.test.Publictest#GetUncashamount; 826440389318017 02	S=amt1
3	yspos.online.test.posgate.PosgateTest#sale; 4:000000000131	S=Response
4	yspos.online.test.posgate.PosgateTest#subString; @Response 11:" ",12 30*Wait;	S=Pos_no
5	yspos.online.test.Publictest#GetUncashamount; 826440389318017 02	S=amt2
6	yspos.online.test.posgate.PosgateTest#AbsoluteValue; @amt1 @amt2	0.0

Web UI

1	cssselector=a[ref=2014040948] exjsjob(arguments[0].scrollIntoView(true);)
2	cssselector=a[ref=2014040948] click
3	cssselector=iframe[src*=method=geturl&limitid=2014040948] gotoframe
4	cssselector=.title>label isdisplayed

创建步骤

重新排序步骤

TestLink

[admin]

TestLink

关键字管理示例

过滤器

测试用例标识

LJPOS-

测试用例标题

测试用例集

状态

{任意}

草稿

准备评审

评审中

测试方式

{任意}

应用过滤器 重置过滤器

展开树 折叠树

(9718)

自动化用例 (484)

2016年测试用例 (5715)

2015年测试用例 (448)

2015年前测试用例 (1537)

(115)

2017年测试用例 (1055)

UI自动化用例 (364)

系统 (240)

系统登录退出 (2)

LJPOS-7701-登录系统

LJPOS-7702-退出系统

管理 (108)

# 框架的扩展



UI界面

**测试平台**

**测试过程可视化操作**

解决测试任务调度、任务&用例  
执行情况查询等可视化操作问题，  
让自动化过程更简单自如

问题分析

**测试日志**

**更快速的定位问题**

测试日志可分为2部分，一类直接  
使用JAVA的LOG4J框架日志，  
更详细精确的定位问题。一类应用  
日志可直接插入到数据库中，  
更方便在UI做查询展示。

数据库操作

**数据库操作封装**

**测试数据验证&测试过程记录**

解决测试过程中数据验证以及  
数据更改的问题，以及框架对  
测试过程的记录(如日志)

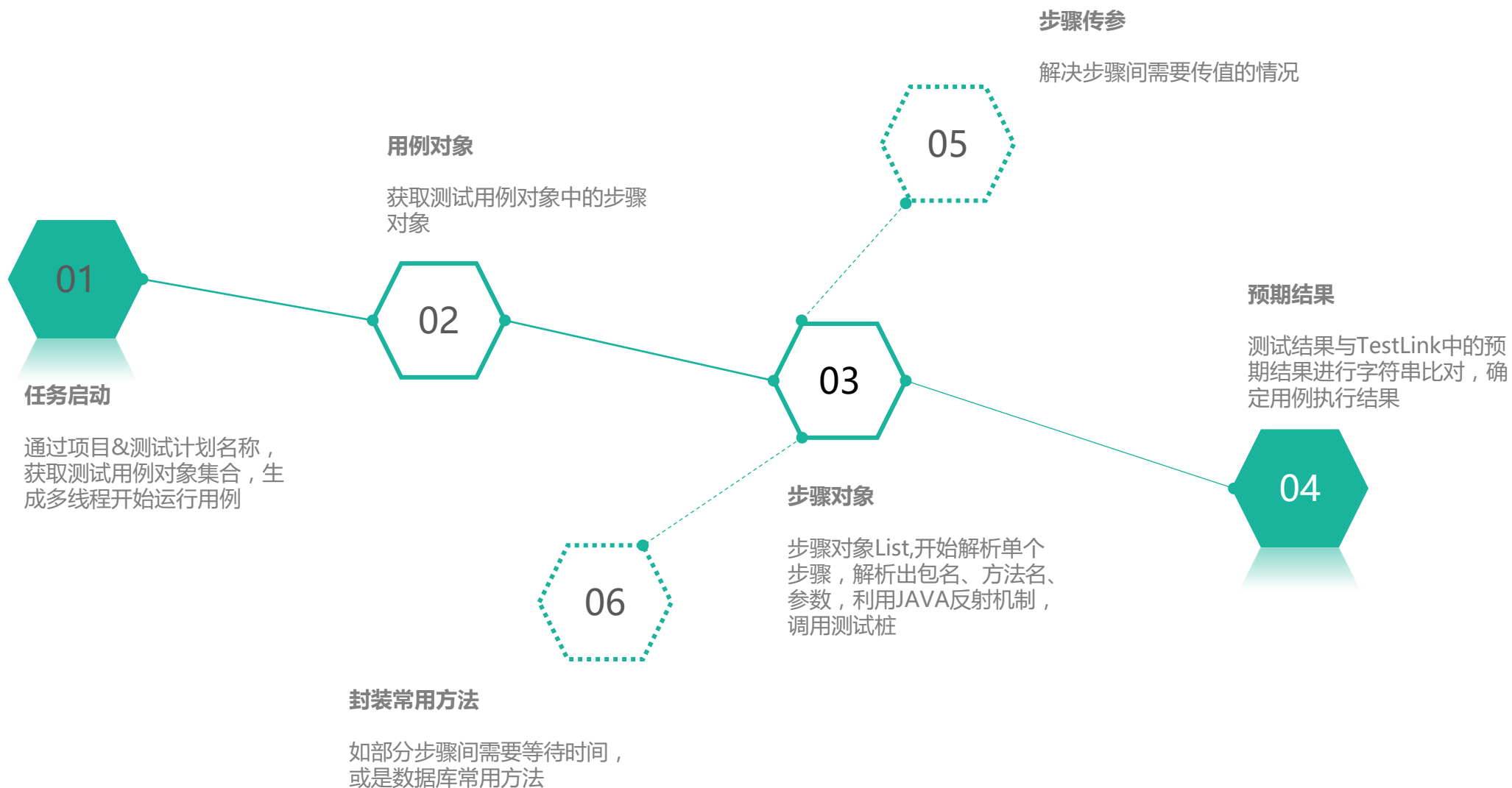




PART 03

# 接口&UI自动化的实现

# TestLink API的使用



# TestLink API 初始化

TestLink API初始化

## TestLink API

- ◆ *TESTLINK\_URL*  
*xmlrpc的地址*
- ◆ *TESTLINK\_DEVKEY*  
*密钥, 请前往testlink  
个人设置界面查看*

### 注意：

- ◆ Testlink在工程的根目录必须有一个  
testlinkjavaapi.properties的  
基本配置文件
- ◆ API 低版本有BUG, 目录发现  
低版本部分API无法使用,现在  
最新的为1.9.16, 暂时没有发现BUG
- ◆ Testlink版本1.9.15在某些情  
况下, *xmlrpc.php*中发现无  
法使用的BUG, 建议使用最  
新的1.9.16.

```
//testlink的IP以及密钥
private final static String TESTLINK_URL =
    "http://127.0.0.1:80/testlink/lib/api/xmlrpc/v1/xmlrpc.php";
protected final static String TESTLINK_DEVKEY =
    "85dbef4ace58e2313418555176a6f2fb";
protected final static Integer PLATFORMID = 0;
protected final static String PLATFORMNAME = null;
protected static TestLinkAPI api= iniTestlinkApi();

private static TestLinkAPI iniTestlinkApi() {
    URL testlinkURL = null;
    try {
        testlinkURL = new URL(TESTLINK_URL);
    } catch ( MalformedURLException mue ) {
        mue.printStackTrace( System.err );
        System.exit(-1);
    }
    return new TestLinkAPI(testlinkURL, TESTLINK_DEVKEY);
}
```



# TestLink 常用 API

TestLink 常用API

## 自动化常用API

- ◆ 自动化过程基本最常用的API就可以满足了，我们需要的是借用testlink有一套科学的用例管理方法。

### 注意：

- ◆ API建议大家使用MAVEN进行构建更新，方便使用。

```
//通过计划名称&项目名称获取测试计划对象
api.getTestPlanByName(testplan, projectname);

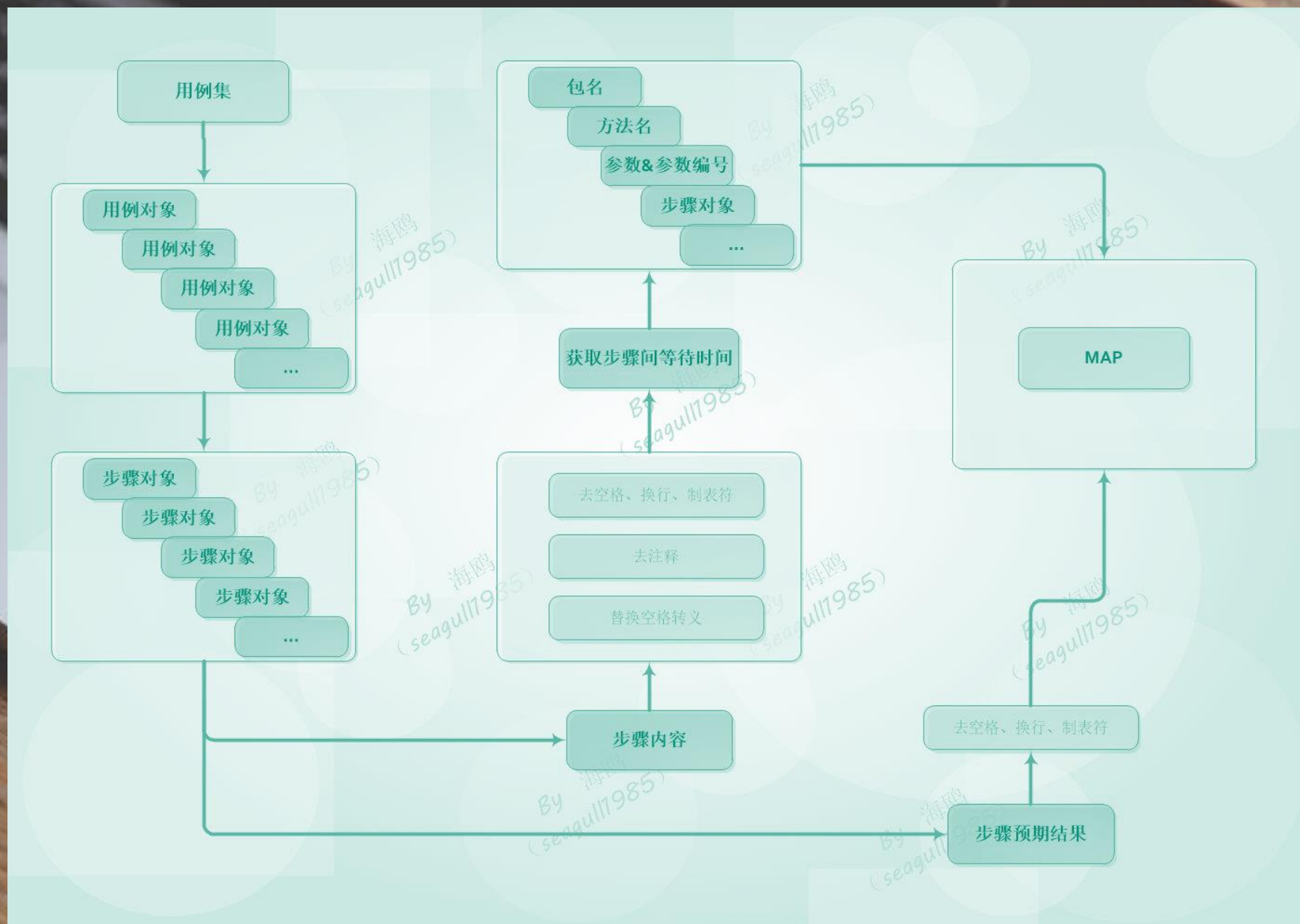
//通过计划ID获取测试用例对象集合
api.getTestCasesForTestPlan(
    planid, null, null, null, null, null, null, null,
    ExecutionType.AUTOMATED, true, TestCaseDetails.FULL);

//通过用例编号&版本号获取用例对象
api.getTestCaseByExternalId(TestCaseExternalId, version);

//获取步骤对象集合
TestCase testcaseob = api.getTestCaseByExternalId(testCaseExternalId,
version);
List<TestCaseStep> testcasesteps = testcaseob.getSteps();
//获取步骤动作内容
testcasesteps.get(ordersteps-1).getActions();
//获取步骤预期结果
testcasesteps.get(ordersteps-1).getExpectedResults();
```

## 流程：

- ◆获取用例集
- ◆生成线程池
- ◆读取单个用例对象的步骤对象
- ◆获取步骤内容
- ◆解析处理步骤内容
- ◆将内容暂放至MAP中，等待调用。



# JAVA Invoke



## 注意：

- ◆ 所有的测试桩使用String方法传参，有需要方法内类型转换处理。
- ◆ 所有的测试桩异常处理，建议抛出至框架层处理，方便定位问题。
- ◆ 框架层不处理、不封装业务方面的东西。

```
public static String CallCase(String packagename, String functionname, Object[]
getParameterValues) {
    try {
        Object server = Class.forName(packagename).newInstance(); // 调用非静态方法用到
        Class[] getParameterTypes = null;
        if (getParameterValues != null) {
            int paramscount = getParameterValues.length;
            // 赋值数组，定义类型
            getParameterTypes = new Class[paramscount];
            for (int i = 0; i < paramscount; i++) {
                getParameterTypes[i] = String.class;
            }
            Method method = getMethod(server.getClass().getMethods(), functionname,
                getParameterTypes);
            if (method == null) {
                throw new Exception( "调用异常，请查看错误日志！" );
            }
            Object str = method.invoke(server, getParameterValues);
            if (str == null) {
                return "返回结果是null";
            } else {
                return str.toString();
            }
        } catch (Throwable e) {
            return "调用异常，请查看错误日志！";
        }
    }
}
```



# 自动化表结构

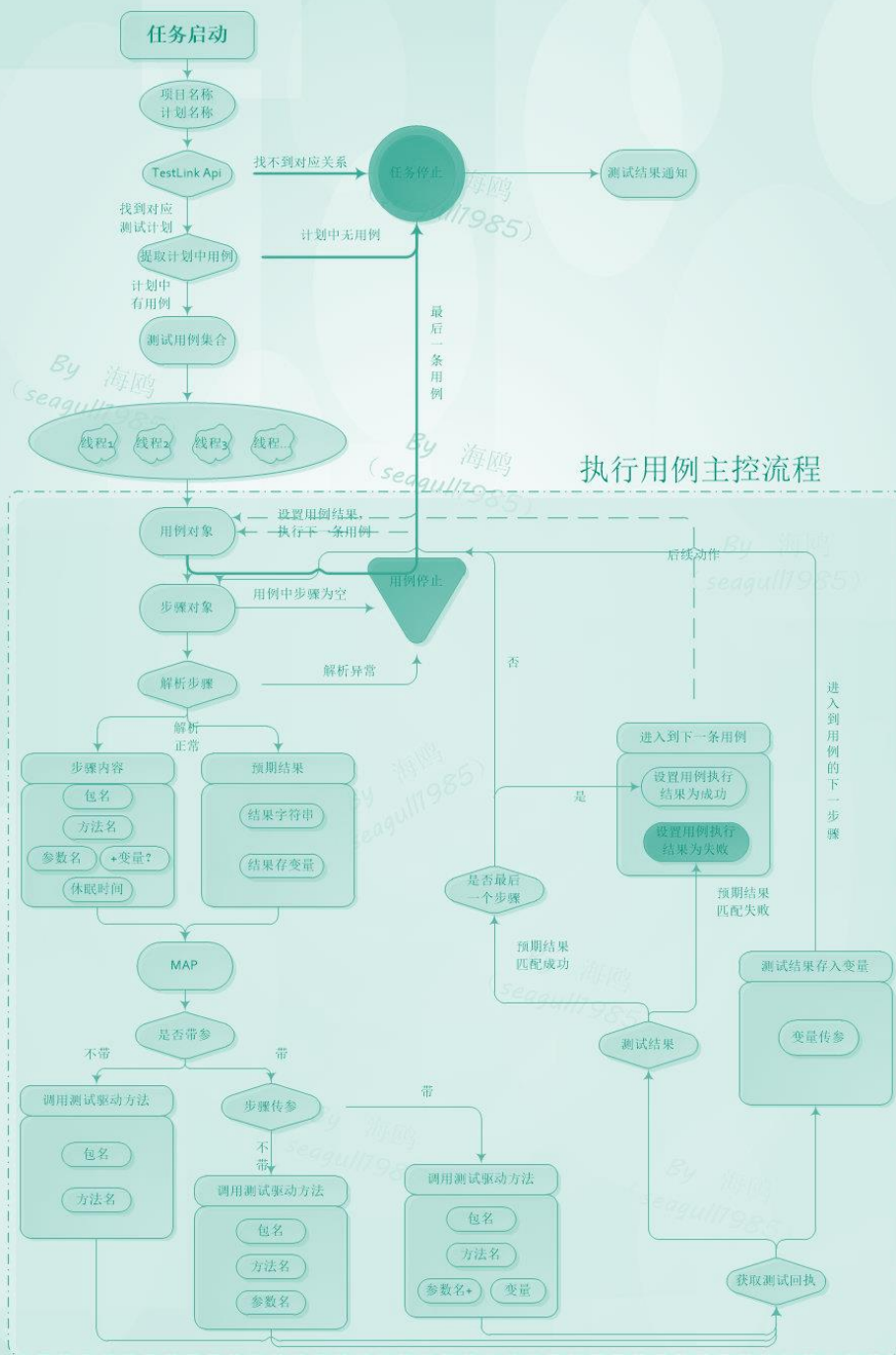


## 简单三张表

三张表搞定自动化表设计，如果你想要有任务调度之类的功能，那么可以再设计一张任务调度表，用来存放测试任务的基本配置信息。

## 流程：

- ◆获取用例集
- ◆生成线程池
- ◆读取单个用例对象的步骤对象(判断是否为空)
- ◆获取步骤内容&预期结果，解析处理步骤内容(判断是否解析异常)
- ◆判断预期结果是否传参。
- ◆判断测试桩方法是否带参(传入参)
- ◆获取执行结果
- ◆判断是否存在传参
- ◆匹配预期结果，判断执行是否成功
- ◆判断是否最后一个步骤





# PART 04

---

## LuckyFrame 演示





THANK YOU

---