基于 Charles 录制会话的 HTTP 接口自动化测试框架设计与实现

刘国庆 汪兴轩*
(复旦大学信息科学与工程学院 上海 200433)

摘 要 传统的 HTTP 接口测试步骤繁琐,工作量巨大,且现有的接口测试工具功能单一、可扩展性差。为了提高接口测试效率、弥补现有工具的不足,提出一种基于 Charles 录制会话的 HTTP 接口自动化测试框架。从客户端出发,录制 HTTP 会话过程,以此构建测试用例池;将测试用例通过特定的中间件服务嵌入至单元测试框架,持续集成平台持续调用测试框架并生成测试报告;框架集成报告发送功能,方便远程查看。通过实验验证,该框架可快速构建测试用例,持续测试接口,测试结果查看方便,测试效率提高。

关键词 Charles 接口 自动化测试 HTTP

中图分类号 TP3 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2019.06.002

DESIGN AND IMPLEMENTATION OF AUTOMATED TEST FRAMEWORK FOR HTTP INTERFACE BASED ON CHARLES RECORDING SESSION

Liu Guoqing Wang Xingxuan*
(*School of Information Science and Technology*, *Fudan University*, *Shanghai 200433*, *China*)

Abstract The traditional procedure of HTTP interface test is cumbersome, the workload is huge, and the existing interface test tools have a single function and poor scalability. In order to improve the efficiency of interface testing and make up for the deficiencies of existing tools, this paper proposed an HTTP interface automated testing framework based on Charles recording session. From the client side, we recorded HTTP session to build a test case pool; we embeded the test case through a specific middleware service into the unit test framework, and the continuous integration platform continuously called the test framework and generated test reports; the framework had a sending report function for remote viewing. Through practice verification, the framework can quickly build test cases, continuously test the interface, and it is convenient to view the test results and can improve test efficiency.

Keywords Charles Interface Automated test HTTP

0 引 言

接口测试是指 Web 系统组件间的测试,它主要用于检测组件与组件之间的交互点[1]。在移动软件开发领域,大多采用分层软件体系结构,服务提供者和消费者基于接口契约传递数据和上下文信息。应用软件的数据大多来源于服务端的接口返回,持续保障接口有效返回、及时发现异常接口是移动软件测试中的一个重要课题[2]。

在 Web 应用开发领域,客户端服务器模型是一种被广泛应用的网络架构,它把整套软件系统划分为客户端和服务端[3],即客户端请求数据,服务端响应内容。超文本传输协议是一个基于请求与响应的无状态 Web 传输协议,因其具有传输效率高、占用较低的网络带宽等特点而被广泛使用[4]。

Charles 是一款被广大 Web 开发者广泛使用的 HTTP/HTTPS 代理服务器、监视器、反向代理服务器软件,当 Web 应用程序通过 Charles 的代理访问互联网时,Charles 可以监控 HTTP/HTTPS 应用程序发送和接

收稿日期:2019-01-11。国家自然科学基金项目(61174041)。**刘国庆**,硕士生,主研领域:计算机网络,软件测试。**汪兴轩**,副教授。

收的所有数据。Charles 允许开发者查看所有连接互联网的 HTTP/HTTPS 通信,包括 Request、Response 以及 Headers 等,在 Web 开发中扮演着重要的角色。

1 概 述

传统的 HTTP 接口测试需要测试人员手工设计用例与构建参数,利用接口测试工具,如 postman、jmeter、fiddler 等[5],模拟客户端向服务端发送请求,服务端处理后返回请求结果,测试人员手工检查返回的结果是否符合预期。传统的测试工具虽然使用方便,易于上手,但是它们也有很多不足之处:

(1) *测试数据需手工输入* 接口测试本质上是对接口输入和输出数据对应关系的测试。测试人员手动调用接口,输入测试数据,然后验证接口返回数据的正确性。每次接口测试之前,测试人员必须手动向工具输入测试数据,耗时耗力。

(2) *无法测试加密接口* 接口测试工具无法自动执行加解密操作,例如 md5、base64、AES 等加密方式。对于加密接口,测试人员需使用加解密工具对参数处理之后才能进行测试,导致测试效率低下。

(3) *扩展能力不足* 接口测试工具虽然有它的便利性,但与此同时也伴随着局限性。例如,测试人员需要将测试结果生成 HTML 形式的测试报告后发到指定的邮箱、需要对接口做持续集成测试等,这些都是工具难以做到的。

对应客户端服务端模型,在移动软件测试领域中,基本上采用"端分离测试"的工作模式,即各端保障各端质量[6]。实际上,客户端工作时往往频繁地调用服务端接口,获得了大量的测试数据,生成了丰富的测试用例。而"端分离测试"这种工作模式忽略了客户端测试人员在测试过程产生的大量的服务端测试用例,从而导致了冗余的测试工作。

传统的接口测试需要测试人员了解接口定义、设计测试用例、配置请求参数、观察接口响应等[7]。由于软件开发中接口设计场景化、请求参数复杂化、服务端数据动态化等趋势愈发明显,人工观察接口响应容易出错,上述步骤必然会导致测试效率不高。特别对于新开发的移动应用软件,必伴随着大量的服务端接口,测试人员逐个测试接口,造成了大量的重复工作。

为此,本文提出了一种应用于新开发移动应用场景下,伴随客户端功能测试,基于 Charles 录制会话的 HTTP 接口自动化测试框架。它依托于业界内已有的开源持续集成平台[8]和单元测试框架,从客户端功能测试出发,实现了自动生成测试用例、对测试数据自动处理、持续集成测试、生成测试报告等功能,测试人员在进行客户端功能测试的同时收集服务端测试数据与用例,添加接口响应校验规则后,便可持续检验接口是否正常工作。

相较于传统的接口测试工具与测试流程,本文提出的方案省去手工构建请求参数环节,实现自动构建请求与校验响应,增加集成测试与扩展功能,实现接口测试由人工到自动化的转变,提高了测试效率。

2 方案设计

2.1 架构设计

如图 1 所示,本文提出的接口自动化测试框架主要包含测试用例录制单元、中间件服务单元以及持续集成测试单元三部分,每个单元又由不同的子单元构成。

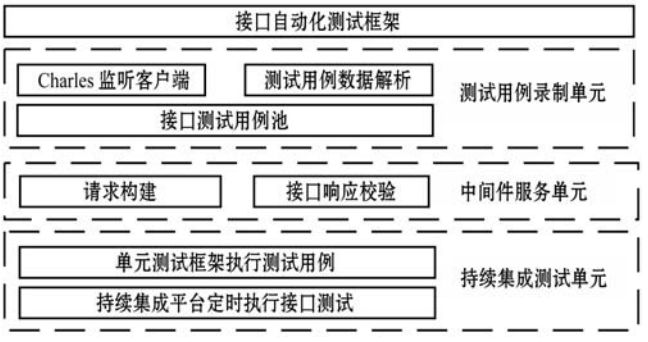

图 1 接口自动化测试框架系统架构

(1) 测试用例录制单元:客户端监听工具(Charles):记录 HTTP 会话,监听被调用接口传递的参数,录制接口测试用例,导出".har"文件;

".har"文件解析:提取会话数据,生成测试用例;

测试用例池:存放测试用例、接口请求参数与校验规则。

(2) 中间件服务单元:接口请求构建:调用测试用例池中的接口 Url、Method 等参数,构建并向服务端发送接口请求;

接口响应校验:校验接口响应信息,检查接口返回的数据结构与值是否符合预期。

(3) 持续集成测试单元:单元测试框架:调用中间件服务单元,执行测试用例,生成测试报告;

持续集成平台:定时调用单元测试框架,自动化执行接口测试,发送测试报告,如有异常则发送报警信息。

2.2 流程设计

如图 2 所示,当客户端开发人员完成应用开发并且转交给测试人员后,测试人员在客户端安装待测应用进行功能测试的同时,监听工具自动采集接口测试

用例,具体流程如下:启动 Charles,客户端连接代理服务器;测试人员进行客户端功能测试,Charles 录制客户端与服务端交互过程中所调用的接口以及传递的参数信息;通过调用 Python 脚本,提取 Charles 录制的接口请求数据,生成测试用例;测试人员完善测试用例并且添加接口响应校验规则后装入单元测试框架;持续集成平台以任务的形式定时运行单元测试框架,其分别完成向服务端发送接口请求数据、接收服务端响应并与预先填写的检验规则作对比、发送测试结果三项任务。

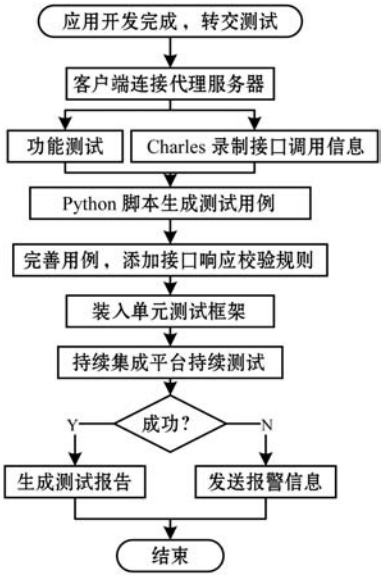

图 2　接口自动化测试框架工作流程

2.3　工程设计

如图 3 所示,在工程上,本文提出的接口自动化测试框架由 Charles + Python + Unittest + Git + Jenkins 构成,主要分为两部分:录制接口测试用例与接口自动化测试。

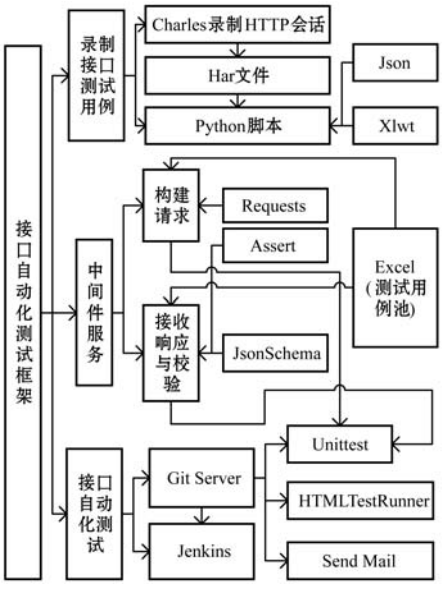

图 3　接口自动化测试框架工程设计

录制接口测试用例部分以 Charles 为基础,利用 Python 的 json 和 xlwt 库构建脚本解析 Charles 导出的“.har”文件,提取测试用例的参数,装入 Excel 测试用例池。接口自动化测试部分则以 Unittest 单元测试框架和持续集成平台 Jenkins 为基础,从测试用例池中提取测试用例,以单元测试的形式将其嵌入至 Unittest,通过中间件服务向服务端发送请求数据与校验接口响应,同时使用 HTMLTestRunner 生成测试报告。另外,构建了 SendMail 模块发送测试报告及相关报警信息。测试代码托管于 Git 服务器,Jenkins 平台从 Git 服务器拉取代码,执行持续集成测试。

3　录制接口测试用例

如图 4 所示,录制接口测试用例部分以 Charles 为核心,Python 脚本将 HTTP 会话过程转换成可用的测试用例。测试人员在进行功能测试的同时,Charles 录制客户端与服务端会话,测试结束后,导出“.har”格式文件。经过白名单与黑名单过滤后,接口测试用例存入测试用例池。

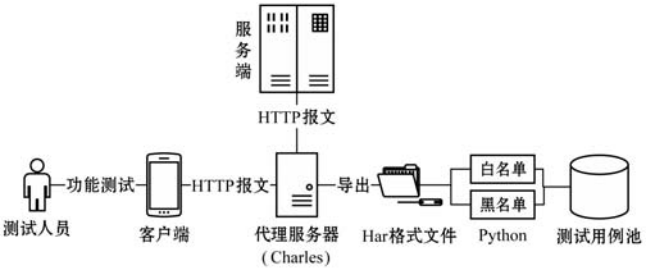

图 4　接口测试用例录制结构

3.1　录制接口测试用例

当客户端连接 Charles 后,客户端的请求参数通过 Charles 转发至服务端,服务端的响应又通过 Charles 下发至客户端。如图 5 所示,对于某款特定的移动应用,其接口一般归于一个或多个域名,而 Charles 将客户端与服务端会话过程中所有的数据按照域名进行归类,导出 HTTP Archive(Har) format 格式(“.har”)后,可查看会话详情。

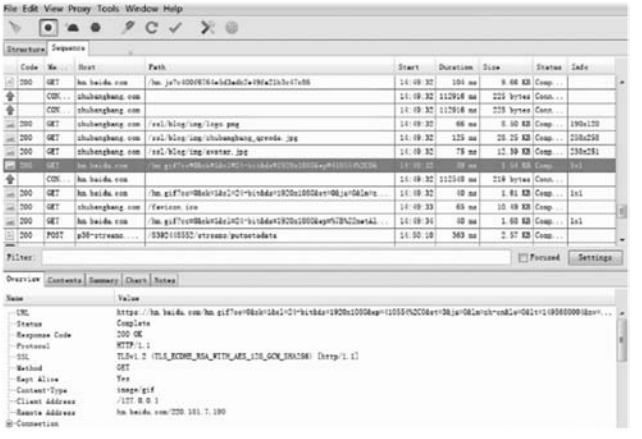

图 5　Charles 代理服务器软件

".har"格式文件在本质上是"utf-8"格式的 json 字符串文件,其中的"entries"字段为列表对象,保存了 HTTP 会话的详情,一个元素则记录了一条 HTTP 会话,"entries"的数据结构如表 1 所示。

表 1 "entries"数据结构

字段名	数据类型	描述
pageref	string	页面 id
startedDateTime	string	请求开始时间
time	number	请求消耗时间(单位 ms)
request	object	请求的消息信息
response	object	响应的详细信息
cache	object	缓存使用情况
timings	object	请求/响应过程时间信息
serverIPAddress	string	服务器 IP 地址
connection	string	TCP/IP 连接标识
comment	string	注释

3.2 构建接口测试用例池

从上一节得知,"entries"中的一个元素即可视为一条测试用例,而其中的"request"和"response"对象则保存了完整的请求与响应信息。本文构建了 Python 脚本将请求参数与响应数据映射至测试用例池。测试用例根据接口的 url 字段归档,采用"一对多"的模式组织数据结构,即一个 url 对应多组参数,从而实现同一个接口对应多个测试用例的组织模式。构建测试用例池流程如图 6 所示。

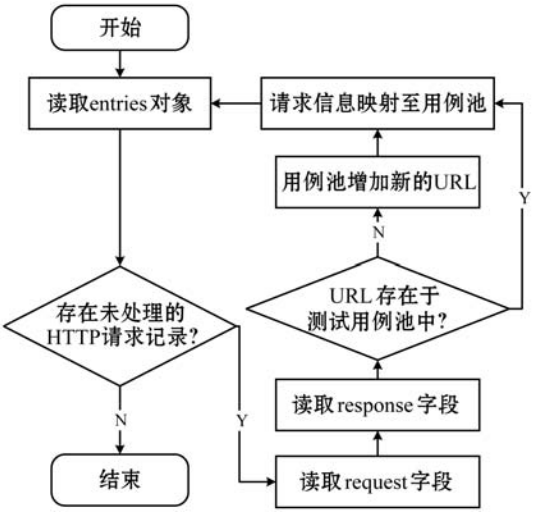

图 6 构建测试用例流程图

除了完整的请求与响应信息以外,赋予测试用例名称与编号,方便后期的维护与扩展。测试用例池的每条用例结构如表 2 所示,其中 headers、queryString、cookies、postData 均为 json 对象。

表 2 测试用例组织结构

	请求/响应	字段	描述
用例编号 用例名称	Request	method	请求方法
		headers	请求头
		url	请求连接
		cookies	本地终端数据
		queryString	请求参数
		postData	post 传输数据
	Response	status	响应状态码
		headers	请求头
		content	响应内容

Python 的 xlrd 和 xlwt 库可以很方便地实现对 Excel 文件的写入与读取,而 Excel 表格可以直观地显示接口的信息以及请求的详细数据。本文使用 Excel 文件作为测试用例池的存储容器,一行为一条测试用例,而测试用例之间可能存在相同的请求对象,后期维护时可以较快扩展用例。

4 中间件服务

中间件服务用于连接用例池与服务端。它分为两个部分,一部分是构建单元,另一部分是接收单元。构建单元负责将测试用例池中的用例参数组装成 HTTP 请求并向服务端发送,待服务端解析请求后,接收单元接收接口响应并将响应报文转换成预期的格式,与预先配置的校验规则做对比。

4.1 构建单元

构建单元以 Python 的 requests 库为基础,将测试用例中参数组装成一个完整的 HTTP 请求。requests 库支持 HTTP 协议中的所有请求方法,并且其能携带所有的 HTTP 请求信息。

服务端解析构建单元的请求之后,会将响应以 Response 对象的形式返回。而在移动应用软件开发中,大多数移动应用基本采用 json 格式耦合客户端与服务端。借助于 Python 中的 json 库,将 HTTP 响应报文转换成 json 格式,方便校验其响应内容。

本文将 request 库和 json 库进一步封装,构成基础 http 请求包,以类方法的形式对外开放,用于传递 HTTP 数据。

类方法介绍如表 3 所示,Get_main 和 Post_main 分

别对应 HTTP 协议中的 Get 和 Post 方法,Headers 和 Data 若为空,则调用默认值,返回对象为统一的 json 格式的字符串。

表 3　封装 HTTP 请求方法

类方法名称	Get_main/Post_main		
参数	url	Headers	Data
参数名称	接口 url	请求头	传输数据
参数类型	String	Dict	Dict
是否为空	N	Y	Y
返回类型	json 对象		

4.2　接收单元

对于接口响应内容的校验,本文选取了两种校验方式。对于无代码能力的测试人员,使用 Json Schema;另一种方式是使用断言(Assert),而这也对测试人员提出了更高的要求。Json Schema 指的是数据交换中的一种虚拟“合同”,用于 Json 数据的一致性检验。它可以验证值的数据类型是否正确,是否包含所需的数据,数据的组成结构是否正确,值的范围是否符合预期等。Python 的 jsonschema 库提供了丰富的 json 数据验证功能,用户仅仅需要将数据导入库即可进行可靠的 json 验证。在 unittest 框架中,TestCase 类提供了较多的方法用于对接口响应结果的校验,表 4 列出了几个常用的断言方法。

表 4　TestCase 类常用断言方法

方法	检查
assertEqual(a, b)	a == b
assertNotEqual(a, b)	a! = b
assertTrue(x)	bool(x) is True
assertFalse(x)	bool(x) is False
assertIs(a, b)	a is b

5　持续测试

持续测试单元由 unittest 单元测试框架、HTMLTestRunnner、Sendmail 以及 Jenkins 构成。其中 unittest 完成测试用例池中的用例组织与管理,HTMLTestRunner 用于生成测试报告,Sendmail 发送测试报告,而 Jenkins 则是整个持续测试的核心,它定期执行测试任务,实现接口测试框架的持续运行。

使用 unittest 单元测试框架组织管理测试用例池中的测试用例,执行所有用例结束后利用 HTMLTestRunner 库生成 html 格式的测试报告,同时构建了 SendMail 模块发送测试信息,便于远程测试人员查看测试详情。

5.1　unittest 与 HTMLTestRunnner

unittest 是一个 Python 单元测试框架,支持自动化测试,全局配置测试的开启与关闭,测试用例以集合的形式聚合到框架中。unittest 以类的形式组织测试用例,每一个测试类起始由“setup()”构成,结束由“teardown()”构成,中间包括各个测试用例“test_case()”。HTMLTestRunner 配合 unittest 用于将测试结果保存至 html 文件中,提高可读性。

在调用 unittest 单元测试框架时,测试用例必须以“test_”开头。通过中间件服务单元将测试用例池中的用例以方法的形式实例化并嵌入至 unittest 单元测试框架。另外,unittest 提供了完整的断言方法,如果某个测试用例断言失败,框架则抛出“AssertionError”,并标识该用例测试失败;否则,则标识该用例为测试成功状态。例如 assertEqual 方法用于验证两个参数值是否相等,下述嵌入示例中的 assertEqual 验证响应结果的状态码是否为 200,若不是则输出“测试失败”,并标记该用例为失败状态;否则,测试用例通过。一个类中可定义多个测试用例函数,按类分组批量执行测试用例。嵌入示例及代码结构如下所示:

嵌入示例:利用 assertEqual 检查接口的响应状态码

```
类名←继承 unittest 中的 TestCase 类
    起始方法 setUp
    测试用例函数 func←传入用例参数(来源于测试用例池)
        url←用例池中的 Request. url
        header←用例池中的 Request. headers
        data←用例池中的 Request. data
        res←Get_main/Post_main(url, header, data)
        assertEqual(res['status'], '200', '测试失败')
    结束方法 teardown
```

测试用例嵌入至 unittest 后,给定 html 文件的保存路径以及测试报告名称,完整的接口测试框架遍搭建完毕。unitest 使用 TestSuite 管理测试用例,一个 TestSuite 是多个测试用例的集合,该集合内的所有测试用例将被一起执行。利用 TestSuite 可以将不同的测试用例组成测试逻辑组,即接口测试用例可根本被测软件的逻辑功能进行模块化组织,然后设置为统一的测试套件,对外仅暴露一个命令即可实现批量执行测试用例。

5.2 SendMail

接口自动化测试框架托管于远程服务器中，为了方便查看测试报告，本文利用 smtplib 库构建了 SendMail 模块集成至框架中，以邮件的形式发送测试报告。基本流程为：

1）读取测试报告；

2）添加邮件内容及附件；

3）连接邮件服务器；

4）发送邮件；

5）退出。

框架定义 list 对象存储邮件接收者信息，每个元素为接收者的邮箱地址，用于接收测试报告。测试报告以“MIMEText”格式集成至框架中，用户设置本地邮箱和密码后，建立 SMTP 邮件服务器，启动邮件发送流程，如发送异常，框架打印异常信息。

5.3 Jenkins 持续集成测试

Jenkins 平台安装 Git 插件后，将托管在 Git 服务器的代码更新至测试环境后做持续集成测试。在配置 Jenkins 任务时选择 Git 源码管理，通过“Poll SCM”设定测试时间间隔，便可持续进行接口测试。值得注意的是，设定测试时间间隔忌太短，以免对服务端造成额外的压力。每次测试结束后，Jenkins 控制台输出测试信息，同时，测试报告以 html 形式存放于测试环境中的指定路径。

6 实施验证

在已部署接口自动化测试框架的测试机系统上，本文对一个 Android 开发应用实施接口自动化测试。具体环境、测试过程、结果如下所述。

6.1 测试环境

搭建基于 CentOS 6.7 操作系统的的 Linux 测试环境，部署本文提出的接口自动化测试框架。框架使用 Python 3.6.12 版本编写，依托于 Jenkins 2.150.2 版本持续集成运行。测试环境安装 Git 命令行工具，Jenkins 配置 Git 服务器地址并生成密钥后与 Git 服务器建立连接。

6.2 测试过程

如表 5 所示，本文对国内某主流动漫社区安卓应用进行冒烟测试，收集其一级按钮下后端接口数据。首先使装有待测应用的客户端连接 Charles 代理服务器，然后对移动应用进行指定测试范围内的冒烟测试。测试结束后，Charles 导出接口“. har”文件，经过 Python 解析后，共获取 124 个接口测试用例，部分测试用例数据如图 7、图 8 所示。

表 5 测试对象、方法、范围

测试对象	国内某主流动漫社区安卓应用
测试方法	冒烟测试
测试范围	一级按钮下所有后端接口

	A	B	C	D	E	F	G
1	ID	caseName	urlPath	method	queryString	headers	cookies
2	1	emoji_list	/apiv2/setting/emoji/l	POST	{'device_id': '5'	{'X-BCY-Ve	{}
3	2	start_start	/api/start/start	POST	{'device_id': '5'	{'X-BCY-Ve	{}
4	3	app_log_	/service/2/app_log/	POST	{'iid': '5466723	{'method': 'P(	{}
5	4	channel_list	/apiv2/channel/list	POST	{'device_id': '5'	{'Accept-Enc	{}
6	5	log_settings_	/service/2/log_setting	POST	{'iid': '5466723	{'method': 'P(	{}
7	6	login_	/passport/auth/login/	GET	{'code': '0210m	{'method': 'G	{}
8	7	token_doLogin	/api/token/doLogin	POST	{'iid': '5466723	{'X-BCY-Ve	{}
9	8	token_relevance	/api/token/relevance	POST	{'iid': '5466723	{'Accept-Enc	{}
10	9	banner_pop	/apiv2/bangumi/banne	POST	{'iid': '5466723	{'X-BCY-Ve	{}
11	10	timeline_friendfeed_hasmore	/apiv2/timeline/friend	POST	{'iid': '5466723	{'X-BCY-Ve	{}
12	11	banner_stream	/apiv2/bangumi/banne	POST	{'iid': '5466723	{'X-BCY-Ve	{}
13	12	user_userAccessList	/api/user/userAccessL	POST	{'iid': '5466723	{'X-BCY-Ve	{}
14	13	message_noticecount	/apiv2/message/notice	POST	{'iid': '5466723	{'X-BCY-Ve	{}
15	14	item_detail	/apiv2/item/detail	POST	{'iid': '5466723	{'Accept-Enc	{}
16	15	item_relative	/apiv2/item/relative	POST	{'iid': '5466723	{'X-BCY-Ve	{}
17	16	reply_getreply	/apiv2/item/reply/getr	POST	{'iid': '5466723	{'X-BCY-Ve	{}
18	17	channel_banner	/apiv2/channel/banne	POST	{'iid': '5466723	{'Accept-Enc	{}
19	18	toplist_rankbrief	/apiv2/item/toplist/ra	POST	{'iid': '5466723	{'Accept-Enc	{}
20	19	banner_gethotbanner	/apiv2/circle/banner/g	POST	{'iid': '5466723	{'X-BCY-Ve	{}
21	20	banner_getslidesbanner	/apiv2/circle/banner/g	POST	{'iid': '5466723	{'X-BCY-Ve	{}

图 7 测试用例池记录(1)

	H	I	J
1	postData	response	
2	{'mimeType': 'a	{'status': 1, 'message': '', 'data': [{'id': 1, 'name': '小呼呼', 'description': '小呼呼表情包上线啦~', 'cover': 'https://sf1-tt	
3	{'mimeType': 'a	{'status': 1, 'data': {'version': -1, 'start_cover': ''}}	
4	{'mimeType': 'a	{'magic_tag': 'ss_app_log', 'message': 'success', 'server_time': 1545017191}	
5	{'mimeType': 'a	{'status': 1, 'message': '', 'data': [{'cid': 6629287743077744899, 'name': '视频', 'type': 3}, {'cid': 66188006505054210	
6	{'mimeType': 'a	{'config': {'session_interval': 30, 'fingerprint_codes': [1, 2, 3, 4, 5, 6], 'allow_push_service': 0, 'image_error_report':	
7		{'data': {'user_id': 102942719846, 'user_id_str': '102942719846', 'name': '用户*102942719846', 'screen_name': '用户*	
8	{'mimeType': 'a	{'status': 1, 'data': {'uid': '102942719846', 'uname': '霜雪江然君', 'sex': '0', 'location': None, 'self_intro': '', 'utags': [], '	
9	{'mimeType': 'a	{'status': 1, 'data': {'weibo': {'bind': False, 'id': '', 'name': ''}, 'mobile': {'bind': False, 'id': '', 'name': ''}, 'qq': {'bind': Fals	
10	{'mimeType': 'a	{'status': 1, 'message': '', 'data': {'link': '', 'pic': '', 'show': False, 'md5': ''}}	
11	{'mimeType': 'a	{'status': 1, 'message': '', 'data': {'fresh_num': 0}}	
12	{'mimeType': 'a	{'status': 1, 'message': '', 'data': {'pic': '', 'show': False, 'md5': ''}}	
13	{'mimeType': 'a	{'status': 1, 'data': [{'access': 'reply_comment', 'value': False}]}	
14	{'mimeType': 'a	{'status': 1, 'message': '', 'data': {'notify': 0, 'remind': 0, 'item_set_update': 0, 'announce': 0, 'message': 0, 'group': 0, 'ask	
15	{'mimeType': 'a	{'status': 1, 'message': '', 'data': {'item_id': 6603886385575231748, 'uid': 2487814, 'odin_uid': '61307642798', 'ctime'	
16	{'mimeType': 'a	{'status': 1, 'message': '', 'data': {'recommend_rela': [{'type': 'note', 'item_id': '6556922141911023885', 'uid': '3508410	
17	{'mimeType': 'a	{'status': 1, 'message': '', 'data': [{'id': '1613820445347902', 'uid': '59947068832', 'uname': '映日茜荷', 'avatar': 'https:/	
18	{'mimeType': 'a	{'status': 1, 'message': '', 'data': []}	
19	{'mimeType': 'a	{'status': 1, 'message': '', 'data': {'rank_brief': [{'title': '7分钟日本催泪短片《入殓师》男孩忍着伤心，为恋人最	
20	{'mimeType': 'a	{'status': 1, 'message': '', 'data': [{'cover': 'https://img-bcy-qn.pstatp.com/editor/flag/c0juaq/6ea95810ee0111e8954915	
21	{'mimeType': 'a	{'status': 1, 'message': '', 'data': [{'cover': 'https://img-bcy-qn.pstatp.com/editor/flag/17973/fbc77880fa0e11e88ee6ff6	

图 8 测试用例池记录(2)

6.3 添加校验规则与配置 Jenkins

根据接口响应数据格式以及接口数据结构的不同，本文选取 jsonschema 与断言两种校验方式交叉验证接口数据是否正常返回。为上一小节测试用例池中的测试用例逐条添加校验规则装入 unittest 后，提交至 Git 服务器。

如表 6 的配置说明所述，在 Jenkins 平台，配置 Git 服务器地址，以 Python 脚本的方式每 5 分钟循环遍历用例池中的用例，每条测试用例通过 unittest 向服务端发出请求，检查响应是否符合预期设定的校验规则。单次遍历接口用例结束后，生成一份 html 格式的测试报告，包括执行用例时间、用例状态、成功或失败详情等，如图 9 所示。如图 10 所示，在 Jenkins 中配置发送者与接收者邮箱后，测试结束后便可向指定邮箱发送测试报告。

表 6　Jenkins 配置说明

触发方式	Python 脚本
触发时机	每 5 分钟
其它配置值	管理员邮箱、git 服务器地址


测试报告

Start Time: 2018-12-17 15:35:53
Duration: 0:00:05.762601
Status: Pass 124

接口自动化测试报告详情:

Show Summary Failed All

Test Group/Test case	Count	Pass	Fail	Error	View
My_TestCase	124	124	0	0	Detail
emoji_list			pass		
start_start			pass		
app_log_			pass		
channel_list			pass		
log_settings_			pass		
login_			pass		
token_doLogin			pass		
token_relevance			pass		
banner_pop			pass		
timeline_friendfeed_hasmore			pass		
banner_stream			pass		
user_userAccessList			pass		
message_noticecount			pass		
item_detail			pass		
item_relative			pass		

图 9　测试报告

图 10　发送测试报告

6.4　框架运行分析

本文将提出的接口自动化测试框架在测试环境中运行 24 小时。邮箱以 5 分钟为间隔收集测试报告,每份报告提供测试详情。因接口返回数据动态变化,框架运行前期会出现因校验规则设计不完整与接口响应数据不匹配的现象,需及时修改校验规则以兼容所有可能的接口并正确返回;运行后期,框架便可实现无人值守的接口自动化测试。

另外,本文对于某些接口设计了相应的负向情况以检验框架的稳定性,当框架运行异常时,如脚本错误、服务端响应超时、测试任务异常等,Jenkins 平台可按照向预先设置的管理员邮箱发送报警邮件,并提供出错的堆栈输出。

7　结　语

本文提出了一种基于 Charles 录制测试用例的 HTTP 接口自动化测试框架,它融合了现有接口测试工具的便利性,与此同时提高了测试效率。以功能测试为出发点,收集功能测试过程中的数据,以此为基础快速完善与补充测试用例,配合 unittest 框架和 Jenkins 平台实现完整的接口自动化测试。通过实践证明,工具在执行自动化接口测试的同时,起到了一定程度的监控作用,持续保证接口下发数据正常。

下一步的研究方向是增加参数列表,实现工具的自动构建测试用例,进而提高接口测试效率以及保证接口测试的全面性。

参　考　文　献

[1] 胡春美. Web 项目接口测试[J]. 电子技术与软件工程,2018(20):45.

[2] 卓欣欣,白晓颖,许静,等. 服务接口测试自动化工具的研究[J]. 计算机研究与发展,2018,55(2):358 - 376.

[3] Tu J, Guo R. The application reseach of mixed program structure based on client-server, browser-server and web service[C]//2011 International Conference on Business Management and Electronic Information. IEEE,2011: 193 - 195.

[4] 祝瑞,车敏. 基于 HTTP 协议的服务器程序分析[J]. 现代电子技术,2012,35(4):117 - 119,122.

[5] 王大伟. 基于 Python 的 Web API 自动化测试方法研究[J]. 电子科学技术,2015,2(5):573 - 581.

[6] 喻晓, 袁谦, 吴广, 等. 移动应用测试, 重点与关键技术, 移动应用软件测试要点综述[J]. 信息化建设, 2018, 239(8):56 - 59.

[7] 蒋灵仙. 基于 Testng 的 Web 接口测试的自动化框架设计与实现[D]. 杭州:浙江工业大学,2016.

[8] 林新党, 穆加艳. 基于 Jenkins 的持续集成系统研究[J]. 雷达与对抗, 2014(1):58 - 61.