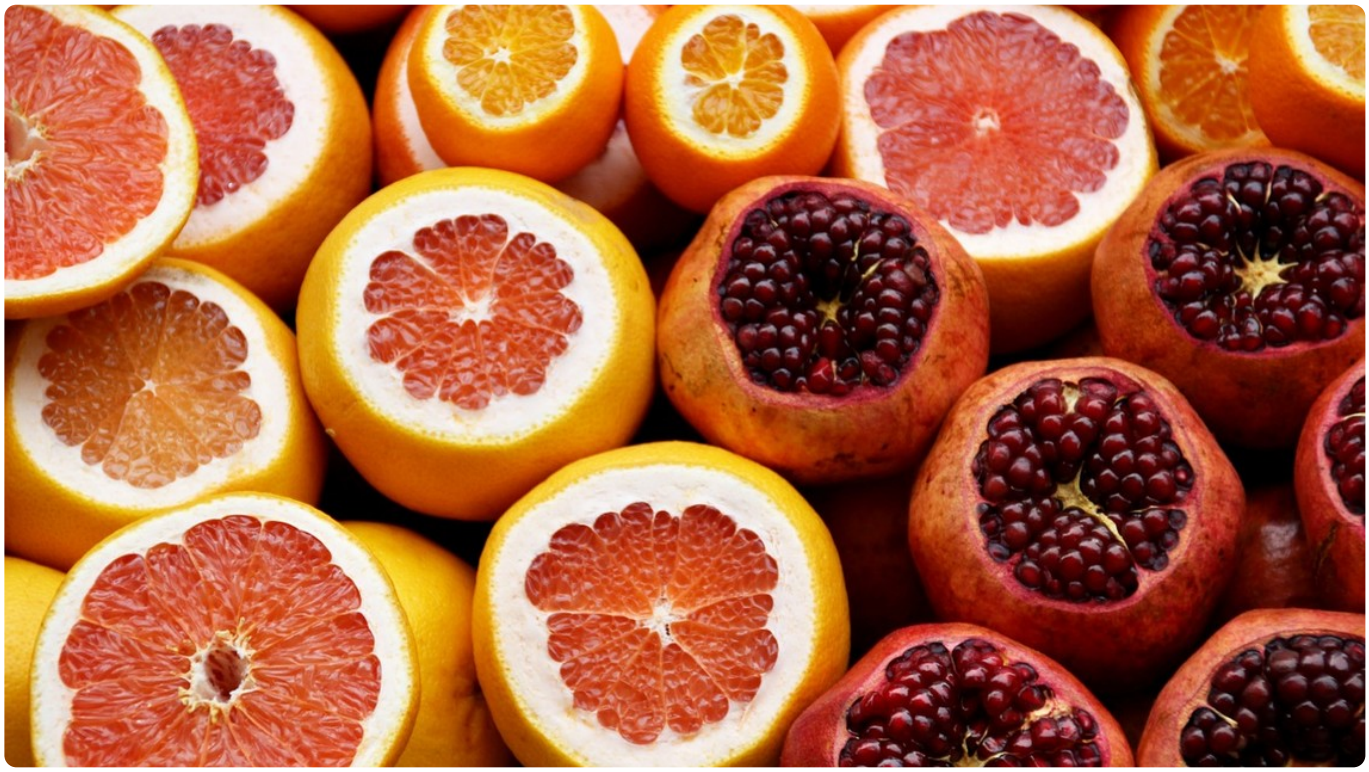
23 | 知其然知其所以然：聊聊API自动化测试框架的前世今生

2018-08-20 茹炳晟

软件测试52讲 进入课程 >



讲述：茹炳晟
时长 14:01 大小 6.42M

你好，我是茹炳晟，今天我和你分享的主题是“知其然知其所以然：聊聊 API 自动化测试框架的前世今生”。

在上一篇文章中，我以一个简单的 Restful API 为例，分别介绍了 cURL 和 Postman 的使用方法，相信你已经对 API 测试有个感性认识了。

但是，我们不能仅仅停留在感性认识的层面，还需要熟悉并掌握这些测试方法，完成相应的 API 测试工作。所以，也就有了我今天分享的主题，希望可以通过对 API 自动化测试框架发展的介绍，让你理解 API 测试是如何一步一步地发展成今天的样子，以“知其所以然”的方式加深你对 API 自动化测试的理解。

接下来，我将会遵循由简入繁的原则，为你介绍 API 测试框架，以发现问题然后解决问题的思路为主线，展开今天的分享。

早期的基于 Postman 的 API 测试

早期的 API 测试，往往都是通过类似 Postman 的工具完成的。但是，由于这类工具都是基于界面操作的，所以有以下两个问题亟待解决：

1. 当需要频繁执行大量的测试用例时，基于界面的 API 测试就显得有些笨拙；
2. 基于界面操作的测试难以与 CI/CD 流水线集成。

所以，我们迫切需要一套可以基于命令行执行的 API 测试方案。这样，API 测试可以直接通过命令行发起，与 CI/CD 流水线的整合也就方便得多了。

基于 Postman 和 Newman 的 API 测试

于是就出现了集成 Postman 和 Newman 的方案，然后再结合 Jenkins 就可以很方便地实现 API 测试与 CI/CDl 流水线的集成。Newman 其实就是一个命令行工具，可以直接执行 Postman 导出的测试用例。

用 Postman 开发调试测试用例，完成后通过 Newman 执行，这个方案看似很完美。但是在实际工程实践中，测试场景除了简单调用单个 API 以外，还存在连续调用多个 API 的情况。

此时，往往会涉及到多个 API 调用时的数据传递问题，即下一个 API 调用的参数可能是上一个 API 调用返回结果中的某个值。另外，还会经常遇到的情况是，API 调用前需要先执行一些特定的操作，比如准备测试数据等。

因此，对于需要连续调用多个 API 并且有参数传递的情况，Postman+Newman 似乎就不再是理想的测试方案了。

基于代码的 API 测试

为了解决这个问题，于是就出现了基于代码的 API 测试框架。比较典型的是，基于 Java 的 OkHttP 和 Unirest、基于 Python 的 http.client 和 Requests、基于 NodeJS 的 Native 和 Request 等。

小型的互联网企业，往往会根据自己的业务需求，选用这些成熟的 API 测试框架。

但是，对于中大型的互联网企业，一般都会自己开发更适合自身业务上下文的 API 测试框架，比如 eBay，我们为了实现代码化的 API 测试，开发了自己的 HttpClient，后期为了使 API 测试的代码更简洁易懂，就基于 Rest-Assured 封装了全新的 API 测试框架。

这种根据公司业务上下文开发实现的 API 测试框架，在使用上有很多优点，而且灵活性也很好，主要体现在以下几个方面：

1. 可以灵活支持多个 API 的顺序调用，方便数据在多个 API 之间传递，即上一个 API 调用返回结果中的某个字段值可以作为后续 API 调用的输入参数；
2. 方便在 API 调用之前或者之后执行额外的任意操作，可以在调用前执行数据准备操作，可以在调用后执行现场清理工作等；
3. 可以很方便地支持数据驱动测试，这里的数据驱动测试概念和 GUI 测试中的数据驱动测试完全相同，也就是可以将测试数据和测试代码分离解耦；
4. 由于直接采用了代码实现，所以可以更灵活地处理测试验证的断言（Assert）；
5. 原生支持命令行的测试执行方式，可以方便地和 CI/CD 工具做集成。

这里我给出了一段伪代码示例，用于展示如何用代码实现一个简单的 API 测试。


```
1  public class CreateUserAPI extends RestAPI{
2      public static String ENDPOINT = "https://xxxx/user/create/v3/{%userId%}";
3      public CreateUserAPI(){
4          super(Methiod.PUT, ENDPOINT);
5      }
6      public Request buildRequest(String userId, String password){
7          Request req = _buildRequest();
8          req.getEndpoint().addInlineParam("userId", userId);
9          req.getEndpoint().addParam("password", password);
10         return req;
11     }
12 }
13
14 public void testCreateUser(String userId, String password){
15     CreateUserAPI createUserAPI = new CreateUserAPI();
16     Request req = createUserAPI.buildRequest(userId, password);
17     Response response = req.request();
18     assert(response.statusCode == 200);
19 }
```

图 1 基于代码的 API 测试的伪代码示例

代码的第 1-12 行，创建了 CreateUserAPI 类，其中包含了 endpoint、操作方法 PUT、InlineParam 和 Param 的设置，并且构建了对应的 request 对象；

代码的第 14-19 行，是测试的主体函数。这段函数的逻辑是这样的：

首先，构建 CreateUserAPI 的对象；

然后，用 CreateUserAPI 对象的 buildRequest 方法结合输入参数构建 request 对象；

接着，通过 request 对象的 request() 方法发起了 API 调用；

最后，验证 response 中的状态码是不是 200。

在这段伪代码中，有以下几点需要你特别注意：

1. 代码中“CreateUserAPI 的父类 RestAPI”“_buildRequest() 方法”“request() 方法”“addInlineParam() 方法”等，都是由 API 测试框架提供的。
2. 为了简化代码，这里并没有引入数据驱动的 data provider。但在实际项目中，代码第 14 行的测试输入参数，往往来自于 data provider，即由数据驱动的方式提供测试输入数据。
3. 由于测试过程完全由代码实现，所以可以很方便的在测试执行前后增加任意的额外步骤。比如，需要在 CreateUser 前增加数据创建的步骤时，只需要在代码第 15 行前直接添加就可以了。
4. 这里的例子只有一个 API 调用，当需要多个 API 顺序调用时，直接扩展 testCreateUser 方法即可，两个 API 之间的数据传递可以通过上一个 API 返回的 response.XXXX 完成。

通过这段伪代码，我们可以看到，虽然基于代码的 API 测试灵活性很好，也可以很方便地和 CI/CD 集成，但是也引入了一些新的问题，比如：

对于单个 API 测试的场景，工作量相比 Postman 要大得多；

对于单个 API 测试的场景，无法直接重用 Postman 里面已经积累的 Collection。

在实际工程中，这两个问题非常重要，而且必须要解决。因为公司管理层肯定无法接受相同工作的工作量直线上升，同时原本已经完成的部分无法继续使用，所以自动化生成 API 测试代码的技术也就应运而生了。

自动生成 API 测试代码

自动生成 API 测试代码是指，基于 Postman 的 Collection 生成基于代码的 API 测试用例。

其实，在上一篇文章《从 0 到 1：API 测试怎么做？常用 API 测试工具简介》最后的部分，我已经提到过 Postman 工具本身已经支持将 Collection 转化成测试代码，但如果直接使用这个功能的话，还有两个问题需要解决：

1. 测试中的断言（assert）部分不会生成代码，也就是说测试代码的生成只支持发起 request 的部分，而不会自动生成测试验证点的代码；
2. 很多中大型互联网企业都是使用自己开发的 API 测试框架，那么测试代码的实现就会和自研 API 测试框架绑定在一起，显然 Postman 并不支持这类代码的自动生成。

鉴于以上两点，理想的做法是自己实现一个代码生成工具，这个工具的输入是 Postman 中 Collection 的 JSON 文件，输出是基于自研 API 框架的测试代码，而且同时会把测试的断言一并转化为代码。

这个小工具实现起来并不复杂，其本质就是解析 Collection JSON 文件的各个部分，然后根据自研 API 框架的代码模板实现变量替换。 具体来讲，实现过程大致可以分为以下三步：

首先，根据自研 API 框架的代码结构建立一个带有变量占位符的模板文件；

然后，通过 JSON 解析程序，按照 Collection JSON 文件的格式定义去提取 header、method 等信息；

最后，用提取得到的具体值替换之前模板文件中的变量占位符，这样就得到了可执行的自研框架的 API 测试用例代码。

有了这个工具后，我建议你的工作模式（Working Model）可以转换成这样：

对于 Postman 中已经累积的 Collection，全部由这个工具统一转换成基于代码的 API 测试用例；

开发人员继续使用 Postman 执行基本的测试，并将所有测试用例保存成 Collection，后续统一由工具转换成基于代码的 API 测试用例；

对于复杂测试场景（比如，顺序调用多个 API 的测试），可以组装由工具转换得到的 API 测试用例代码，完成测试工作。

如图 2 所示，就是一个组装多个由工具转换得到的 API 测试用例代码的例子。其中，代码第 3 行的类“CreateUserAPI”和第 10 行的类“BindCreditCardAPI”的具体代码就可以通过工具转换得到。

```
1  public void testComplexScenario(String userId, String password, String creditCardId, String cvv){
2
3      CreateUserAPI createUserAPI = new CreateUserAPI();
4      Request createUserRequest = CreateUserAPI.buildRequest(userId, password);
5      Response createUserResponse = createUserRequest.request();
6      assert(createUserResponse.statusCode == 200);
7
8      // 可以在此处添加额外的步骤，比如准备测试数据，数据格式的转换等
9
10     BindCreditCardAPI bindCreditCardAPI = new BindCreditCardAPI();
11     Request bindCreditCardRequest = bindCreditCardAPI.buildRequest(createUserResponse.body.userId, creditCardId, cvv);
12     Response bindCreditCardResponse = bindCreditCardRequest.request();
13     assert(bindCreditCardResponse.statusCode == 200);
14     assert(...);
15
16 }
```

图 2 多个 API 顺序调用的测试用例代码

至此，基于代码的 API 测试发展得算是比较成熟了，但在实际应用过程中还有一个痛点一直未被解决，那就是测试验证中的断言，也是我接下来要和你一起讨论的话题。

Response 结果发生变化时的自动识别

在实际的工程项目中，开发了大量的基于代码的 API 测试用例后，你会发现一个让人很纠结的问题：到底应该验证 API 返回结果中的哪些字段？

因为你不可能对返回结果中的每一个字段都写 assert，通常情况下，你只会针对关注的几个字段写 assert，而那些没写 assert 的字段也就无法被关注了。

但对 API 测试来说，有一个很重要的概念是后向兼容性（backward compatibility）。API 的后向兼容性是指，发布的新 API 版本应该能够兼容老版本的 API。

后向兼容性除了要求 API 的调用参数不能发生变化外，还要求不能删减或者修改返回的 response 中的字段。因为这些返回的 response 会被下游的代码使用，如果字段被删减、改名或者字段值发生了非预期的变化，那么下游的代码就可能因为无法找到原本的字段，或者因为字段值的变化而发生问题，从而破坏 API 的后向兼容性。

所以，我们迫切需要找到一个方法，既可以不对所有的 response 字段都去写 assert，又可以监测到 response 的结构以及没有写 assert 的字段值的变化。

在这样的背景下，诞生了“Response 结果变化时的自动识别”技术。也就是说，即使我们没有针对每个 response 字段都去写 assert，我们仍然可以识别出哪些 response 字段发生了变化。

具体实现的思路是，在 API 测试框架里引入一个内建数据库，推荐采用非关系型数据库（比如 MongoDB），然后用这个数据库记录每次调用的 request 和 response 的组合，当下次发送相同 request 时，API 测试框架就会自动和上次的 response 做差异检测，对于有变化的字段给出告警。

你可能会说这种做法也有问题，因为有些字段的值每次 API 调用都是不同的，比如 token 值、session ID、时间戳等，这样每次的调用就都会有告警。

但是这个问题很好解决，现在的解决办法是通过规则配置设立一个“白名单列表”，把那些动态值的字段排除在外。

总结

为了让你可以更好地理解今天的 API 测试框架，我从其发展历程的角度进行了分析：

早期的基于 Postman 的 API 测试在面临频繁执行大量测试用例，以及与 CI/CD 流水线整合的问题时，显得心有余而力不足。为此，基于命令行的 API 测试实践，也就是 Postman+Newman，具有很好的灵活性，解决了这两个问题。

但是，Postman+Newman 的测试方案，只能适用于单个 API 调用的简单测试场景，对于连续调用多个 API 并涉及到参数传递问题时，这个方案就变得不那么理想和完美了。随后，API 测试就过渡到了基于代码的 API 测试阶段。

一些小型企业，则往往会选择适合自己业务的成熟 API 测试框架。中大型的互联网企业，一般都会根据自己的业务上下文，在成熟 API 测试框架的基础上封装自己的 API 测试框架，提升测试效率和灵活性。

但是，不管是采用现成的还是自己去开发 API 测试框架，都会遇到测试用例开发效率低下，以及无法直接重用 Postman 中积累的 Collection 的问题，为此我分享了两个比较好用的方法，也就是：自动生成 API 测试代码和 Response 结果变化的自动识别，并给出了这两个方法的实现思路。

希望我分享的这些内容，可以帮你解决在实际测试项目中遇到的问题。

思考题

目前，基于代码的 API 测试框架已经比较成熟了，所以在此基础上又出现了基于配置文件的 API 测试框架，比如典型的 HttpRunner，在此类 API 测试框架的支持下，测试用例本身往往就是纯粹的配置文件了。你是否有接触过这类 API 测试框架，对此又有什么看法呢？

欢迎你给我留言。

上一篇　22 | 从0到1：API测试怎么做？常用API测试工具简介

下一篇　24 | 紧跟时代步伐：微服务模式下API测试要怎么做？

精选留言 (26)

写留言

Martin 龚...

2018-09-07

20

用postman转python或者java测试脚本还是太慢了，而且需要一定编程技能，感觉已经是上一代了，我现在根据httprunner的yml的脚本规则，加上一些开源的组件，做了一个web页面可以进行代理抓包，测试人员无论从web页面还是app操作只要设置代理过来，就可以看到自己的所有请求，然后选择想自动化的请求，后台自动转成测试脚本，再在管理界面上通过拖拽等性质组装成自动化测试集，并可以执行调试、定时任务等。这样的...

展开

作者回复: 非常好的实践，其实这个我们也有在做，基于httprunner基本没有代码工作量，全部基于yaml或json的配置，而且就像你说的是通过har直接转换json或者yaml，结合locust还可以直接做性能测试。

Cynthia♦...

2018-08-21

6

感觉本篇对我最有用的，就是Response结果发生变化时的自动识别这块啊！自己在项目中遇到过类似的问题，虽然采取了一定的措施，但是没有想到作者这个解决方案。准备顺着作者的思路捋一捋，看看把他具体实现一下。

展开

作者回复: 哈哈，这的确是个很好的方法，非常值得落地

sylan215

2018-08-20

5

确实，Postman 作为自测工具或者开发之间的接口对接，还是很方便的，但是涉及批量每日回归执行，或者同一个接口的多参数验证等，稍微有点不方便，这应该也是那么的代码级 API 测试框架出现的原因吧。
相对于自动生成 API 测试代码，我更倾向于配置文件的方式，毕竟通过工具把 JSON 转换为自己需要的格式，说到底也是配置文件而已，只是还要增加一个转换的过程。...

展开

楚耳

2019-03-22

2

我觉得老师对api的测试都是基于单个api的测试，我是做P2P这块的，公司的api执行后都会去操作数据库，操作各种表，api调用后，需要验证的是各表的字段是否修改正确，这个

才是我们验证的重点，我想如果业务复杂的api基本都是跟数据库关联很大，Response的返回值验证都是次要的了。所以我觉得老师这个api测试的讲解不够深入

展开

不错的葱

2018-12-19

2

老师请教两个问题：
1.做api测试时，是否有必要对数据库做验证？比如测试添加接口，添加成功后是否需要逐一验证数据库字段是否正确；再比如，测试获取详情接口，是否需要把接口的返回和数据库中数据做对比验证。
2.引入数据驱动后，对于不同的输入，验证的步骤相差很大。比如：可能有的输入直接验...

展开

颜瑞

2018-10-24

1

在框架的选择上，茹老师也提了一个开放性问题，Rest-Assured和HttpRunner哪个更合适。其实我也一直在犹豫，自己封装接口，对各个产品线上测试人员有编码的能力要求，另外框架的开发还是需要一定工作量的，HttpRunner少了这些编码成本。HttpRunner还没开始用，疑惑点在于数据准备和环境清理能否在这个框架里面实现，考虑到一些数据准备需要插表，是不是从扩展性和兼容性上讲，企业还是自己开发一套API测试框架的好？...

展开

楚耳

2018-09-13

1

老师 能具体讲下在框架中如何实现 可以灵活支持多个 API 的顺序调用，方便数据在多个 API 之间传递

，即上一个 API 调用返回结果中的某个字段值可以作为后续 API 调用的输入参数 这个需求吗 讲下设计思路

展开

静静张

2018-08-25

1

老师你好，基于配置文件的api测试，和将数据外化到文件是一回事吗？

展开

作者回复: 不是一回事，基于配置的api测试，是不需要写任何代码的，测试用例本身就是配置文件，api框架会去解析配置文件并发起调用，典型的框架是httprunner

柠檬
2019-05-23

我们公司现在正在用httprunner做接口测试，有个重要的接口需要取得第三方授权的access token，但是好像都绕不过手动登录第三方网站的步骤。不知道老师是否有解决方法可以提供一下参考？谢谢。

展开

口水窝
2019-04-12

以前有用过locust，但不知道什么原理，就知道能压测，今天查了下可以和httprunner完美组合，先记下来，后面实践。

作者回复: 值得一试，学习成本是很低的

口水窝
2019-04-11

有了理论支持，后续每个阶段的工具都实践起来！

展开

yudi5158
2019-03-30

response结果变化的自动识别，我目前项目使用了大量的json schema验证 效果还不错

2019-02-14

使用httprunner做API自动化测试时，数据准备这块需要额外开发吗？

展开

johnny
2019-01-17

老师将的逻辑结构严谨、清晰，能够由简入繁，既有实操，又有解决思路，赞。

zhangliqu...
2018-12-26

老师，接口自动化测试我看过robotframeworker接口自动化框架，老师RF框架和您讲的有差异呢?

作者回复: 不同的工具间都有差异，但是本质的东西是一样的

Geek_9db07...
2018-12-01

有了api自动化集成测试平台，还要代码生成工具有啥用，restassured其实也是在httpclient上封装的，他不仅能发请求，还能断言

农夫山泉
2018-10-29

问下老师，现在用python3写接口自动化，用哪个框架最好

展开

小老鼠
2018-10-27

在中小型公司有无必要引入了response 变化工具吗

展开

Robert小七
2018-10-25

现在项目都是手工测试，如何将手工测试和接口测试相结合？也就是测试同学既要做手工也要做自动化如何开展！

胖虫子 2019-10-17

做过个rf的接口测试，但有时想想觉得接口的测试没啥大意义，现在接口正常不正常都能监控