05 | 你知道软件开发各阶段都有哪些自动化测试技术吗?

2018-07-09 茹炳晟

进入课程 >

讲述：茹炳晟

时长 14:36　大小 6.70M

在前面的文章中，我介绍了为什么要做自动化测试，以及什么样的项目适合做自动化测试，那么现在我来说说软件开发生命周期的各个阶段都有哪些类型的自动化测试技术。

说到自动化测试，你可能最为熟悉的就是 GUI 自动化测试了。比如，早年的 C/S 架构，通常就是用自动化测试脚本打开被测应用，然后在界面上以自动化的方式执行一系列的操作；再比如，现今的 Web 站点测试，也是用自动化测试脚本打开浏览器，然后输入要访问的网址，之后用自动化脚本识别定位页面元素，并进行相应的操作。

因此，说到自动化测试时，你的第一反应很可能就是 GUI 自动化测试。然而，**在软件研发生命周期的各个阶段都有自动化测试技术的存在，并且对提升测试效率有着至关重要的作用。**

今天这篇文章，我将会以不同的软件开发阶段涉及的自动化测试技术为主线，带你了解单元测试、代码级集成测试、Web Service 测试和 GUI 测试阶段的自动化技术，希望可以帮助你更深入地理解“自动化测试”的内涵以及外延。

单元测试的自动化技术

首先，你可能认为单元测试本身就是自动化的，因为它根据软件详细设计采用等价类划分和边界值分析方法设计测试用例，在测试代码实现后再以自动化的方式统一执行。

这个观点非常正确，但这仅仅是一部分，并没有完整地描述单元测试“自动化”的内涵。**从广义上讲，单元测试阶段的“自动化”内涵不仅仅指测试用例执行的自动化，还应该包含以下五个方面：**

1. 用例框架代码生成的自动化；
2. 部分测试输入数据的自动化生成；
3. 自动桩代码的生成；
4. 被测代码的自动化静态分析；
5. 测试覆盖率的自动统计与分析。

你可能感觉这些内容有些陌生，不过没关系，下面我就详细地跟你说说每一条的具体含义。

第一，用例框架代码生成的自动化

有些框架代码应该由自动化工具生成，而不是由开发者手工完成。这样一来，单元测试开发者可以把更多的精力放在测试逻辑的覆盖和测试数据的选择上，从而大幅提高单元测试用例的质量和开发效率。

```java
import org.testng.Assert;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

public class TestNGExample {

    @DataProvider(name = "addMethodDataProvider")
    public Object[][] dataProvider() {
        return new Object[][] { { 0, 0, 0 }, { 0, 0, 0 }, { 0, 0, 0 } };
    }

    @Test(dataProvider = "addMethodDataProvider")
    public void testAddMethod(int a, int b, int result) {
        Calculator calculator = new Calculator();
        Assert.assertEquals(calculator.add(a, b), result);
    }

}
```

TestNG 框架代码应该由自动化工具生成

第二，部分测试输入数据的自动化生成

这部分是指，自动化工具能够根据不同变量类型自动生成测试输入数据。自动化工具本身不可能明白代码逻辑，你可能很难理解它是如何根据需要测试的代码逻辑生成合适的输入数据，并且去判断预计的测试结果的。那我给你举个例子，你就很容易明白了。

比如，某个被测函数的原型是 void fun（int* p, short b），那么测试数据自动生成技术就会为输入参数 int* p 自动生成“空”和“非空”的两个指针 p，然后分别执行函数 void fun（int* p, short b），并观察函数的执行情况。

如果函数内部没有对空指针进行特殊处理，那么函数 fun 的调用必定会抛出异常，从而发现函数的设计缺陷。同样地，对于输入参数 short b 会自动生成超出 short 范围的 b，测试函数 fun 的行为。

第三，自动桩代码的生成

简单地说，桩代码（stub code）是用来代替真实代码的临时代码。 比如，某个函数 A 的内部实现中调用了一个尚未实现的函数 B，为了对函数 A 的逻辑进行测试，那么就需要模拟一个函数 B，这个模拟的函数 B 实现就是所谓的桩代码。

自动桩代码的生成是指自动化工具可以对被测试代码进行扫描分析，自动为被测函数内部调用的其他函数生成可编程的桩代码，并提供基于测试用例的桩代码管理机制。此时，**单元测试开发者只需重点关注桩代码内的具体逻辑实现，以及桩代码的返回值。**

必要的时候，自动化工具还需要实现 “抽桩”，以适应后续的代码级集成测试的需求。

那什么是“抽桩”呢？其实也很简单，在单元测试阶段，假如函数 A 内部调用的函数 B 是桩代码，那么在代码级集成测试阶段，我们希望函数 A 不再调用假的函数 B，而是调用真实的函数 B，这个用真实函数 B 代替原本桩代码函数 B 的操作，就称为“抽桩”。

第四，被测代码的自动化静态分析

静态分析主要指代码的静态扫描，目的是识别出违反编码规则或编码风格的代码行。通常这部分工作是结合项目具体的编码规则和编码风格，由自动化工具通过内建规则和用户自定义规则自动化完成的。目前比较常用的代码静态分析工具有 Sonar 和 Coverity 等。

严格意义上讲，静态分析不属于单元测试的范畴，但这部分工作一般是在单元测试阶段通过自动化工具完成的，所以我也把它归入到了单元测试自动化的范畴。

第五，测试覆盖率的自动统计与分析

单元测试用例执行结束后，自动化工具可以自动统计各种测试覆盖率，包括代码行覆盖率、分支覆盖率、MC/DC 覆盖率等。这些自动统计的指标，可以帮你衡量单元测试用例集合的充分性和完备性，并可以为你提供适当增补测试用例以提高测试覆盖率的依据。

代码级集成测试的自动化技术

通俗地讲，代码级集成测试是指将已经开发完成的软件模块放在一起测试。

从测试用例设计和测试代码结构来看，代码级集成测试和单元测试非常相似，它们都是对被测试函数以不同的输入参数组合进行调用并验证结果，只不过代码级集成测试的关注点，更多的是软件模块之间的接口调用和数据传递。

代码级集成测试与单元测试最大的区别只是，代码级集成测试中被测函数内部调用的其他函数必须是真实的，不允许使用桩代码代替，而单元测试中允许使用桩代码来模拟内部调用的其他函数。

以上的这些异同点就决定了代码级集成测试“自动化”的内涵与单元测试非常相似，尤其是在实际操作层面，比如测试用例的设计方法、测试用例的代码结构以及数据驱动思想的应用等等。

但是，代码级集成测试对测试框架的要求非常高，这个框架除了可以顺利装载自己的软件模块外，还必须能装载其他相互依赖的模块，做到被测软件模块可运行（Runnable）。

由于代码级集成测试主要应用在早期非互联网的传统软件企业，那时候的软件以“单体”应用居多，一个软件内部包含大量的功能，每一个软件功能都是通过不同的内部模块来实现的，那么这些内部模块在做集成的时候，就需要做代码级集成测试。

现在的开发理念追求的是系统复杂性的解耦，会去尽量避免“大单体”应用，采用 Web Service 或者 RPC 调用的方式来协作完成各个软件功能。所以现在的软件企业，尤其是互联网企业，基本不会去做代码级集成测试，我在这里也就不再进一步展开了。

Web Service 测试的自动化技术

Web Service 测试，主要是指 SOAP API 和 REST API 这两类 API 测试，最典型的是采用 SoapUI 或 Postman 等类似的工具。但这类测试工具基本都是界面操作手动发起 Request 并验证 Response，所以难以和 CI/CD 集成，于是就出现了 API 自动化测试框架。

如果采用 API 自动化测试框架来开发测试用例，那么这些测试用例的表现形式就是代码。为了让你更直观地理解基于代码的 API 测试用例是什么样子的，我给你举一个“创建用户”API 的例子，你只需要看代码的大致步骤就可以了，具体到每行代码的含义，我会在后续文章中详细讲解。

```
public void testCreateUser(data provide parameters) {
     CreateUserAPI createUserAPI = new CreateUserAPI();
     // build the request for the API
     Request req = createUserAPI.buildXXXRequest(String userId, String oldPassword);
     // call the API and get the reponse
     Response response = req.request();
     //Validate repsonse
     assert(response.statusCode == 200);
}

public class CreateUserAPI extends RestAPI {
    public static String ENDPOINT = "https://xxx.com/user/create/v2/{%userId%}/";
    public CreateUserAPI() {
        super(Method.PUT, ENDPOINT);
    }
    public Request buildRequest(String userId, String password) {
        Request req = _buildRequest();
        req.getEndpoint().addInlineParam("userId", userId);
        req.getEndpoint().addParam("oldPassword", oldPassword);
        return req;
    }
}
```

基于 API 自动化测试框架的测试用例示例（测试 CreateUser API）

对于基于代码的 API 测试用例，通常包含三大步骤：

1. 准备 API 调用时需要的测试数据；
2. 准备 API 的调用参数并发起 API 的调用；
3. 验证 API 调用的返回结果。

目前最流行的 API 自动测试框架是 REST Assured，它可以方便地发起 Restful API 调用并验证返回结果。关于 REST Assured 的用法和优点，我会在后续文章中详细介绍。

同样地，Web Service 测试“自动化”的内涵不仅仅包括 API 测试用例执行的自动化，还包括以下四个方面：

1. 测试脚手架代码的自动化生成；
2. 部分测试输入数据的自动生成；
3. Response 验证的自动化；
4. 基于 SoapUI 或者 Postman 的自动化脚本生成。

接下来，我会依次为你解释这 4 个方面代表什么含义。

第一，测试脚手架代码的自动化生成
和单元测试阶段的用例框架代码自动生成一个道理，你在开发 API 测试的过程中更关心的是，如何设计测试用例的输入参数以及组合，以及在不同参数组合情况下 Response 的验证，而你不希望将精力浪费在代码层面如何组织测试用例、测试数据驱动如何实现等非测试业务上。

这时，测试脚手架代码的自动生成技术就派上用场了。它生成的测试脚手架代码，通常包含了被测试 API 的调用、测试数据与脚本的分离，以及 Response 验证的空实现。

第二，部分测试输入数据的自动生成

这一点和单元测试的测试输入数据的自动化生成也很类似，唯一不同的是，单元测试针对的参数是函数输入参数和函数内部输入，而 API 测试对应的是 API 的参数以及 API 调用的

Payload。数据生成的原则同样遵循边界值原则。

第三，Response 验证的自动化

对于 API 调用返回结果的验证，通常关注的点是返回状态码（status code）、Scheme 结构以及具体的字段值。如果你写过这种类型的测试用例，那你就会知道字段值的验证相当麻烦，只有那些你明确写了 assert 的字段才会被验证，但是通常你不可能针对所有的字段都写 assert，这时就需要 Response 验证的自动化技术了。

Response 验证自动化的核心思想是自动比较两次相同 API 调用的返回结果，并自动识别出有差异的字段值，比较过程可以通过规则配置去掉诸如时间戳、会话 ID（Session ID）等动态值。 这部分内容，我会在后续文章中详细讲解。

第四，基于 SoapUI 或者 Postman 的自动化脚本生成

你在使用 SoapUI 或者 Postman 等工具进行 Web Service 测试时，已经在这些工具里面积累了很多测试用例。那么，在引入了基于代码实现的 API 测试框架之后，就意味着需要把这些测试用例都用代码的方式重写一遍，而这额外的工作量是很难被接受的。

我的建议是，开发一个自动化代码转换生成工具。这个工具的输入是 SoapUI 或者 Postman 的测试用例元数据（即测试用例的 JSON 元文件），输出是符合 API 测试框架规范的基于代码实现的测试用例。这样一来，原本的测试用例积累可以直接转换成在 CI/CD 上可以直接接入的自动化测试用例。

对于新的测试用例，还可以继续用 SoapUI 或者 Postman 做初步的测试验证，初步验证没有问题后，直接转换成符合 API 测试框架规范的测试用例。对于复杂的测试用例，也可以直接基于代码来实现，而且灵活性会更好。

GUI 测试的自动化技术

GUI 测试的自动化技术可能是你最熟悉的，也是发展时间最长、应用最广的自动化测试技术。它的核心思想是，基于页面元素识别技术，对页面元素进行自动化操作，以模拟实际终端用户的行为并验证软件功能的正确性。

目前，GUI 自动化测试主要分为两大方向，传统 Web 浏览器和移动端原生应用（Native App）的 GUI 自动化。虽然二者采用的具体技术差别很大，但是用例设计的思路类似。

对于传统 Web 浏览器的 GUI 自动化测试，业内主流的开源方案采用 Selenium，商业方案采用 Micro Focus 的 UFT（前身是 HP 的 QTP）；

对于移动端原生应用，通常采用主流的 Appium，它对 iOS 环境集成了 XCUITest，对 Android 环境集成了 UIAutomator 和 Espresso。

这部分内容，我会在后续的文章中详细展开。

总结

我给你梳理了软件研发生命周期各个阶段的自动化测试技术，包括单元测试、代码级集成测试、Web Service 测试和 GUI 测试的自动化技术，并给你归纳了每一类技术的核心方法和应用场景。

我希望你通过这篇文章，可以先对自动化测试的全局有一个比较清晰的认识，然后在后续的文章中我还会针对这些技术展开讨论，并给你分享一些相应的实际案例。

思考题

你现在所在的公司，是否实行代码级测试，用到了哪些自动化测试技术？

欢迎你给我留言。

上一篇　04 | 为什么要做自动化测试？什么样的项目适合做自动化测试？

下一篇　06 | 你真的懂测试覆盖率吗？

精选留言 (53)

写留言

康美之心 ...
2018-07-09

62

Rest-assured是一个非常适合，非常好用的API开源测试框架，我已经在我们项目里基于Spring-boot，Rest-Assured，Ccucumber开发了一个标准通用的微服务API自动化测试框架，我们项目里的所有微服务API都可以基于框架快速高效的在各个测试环境无缝切换进行自动化测试，并且完全集成到了公司的CICD Jenkins平台，以前从开发到单元测试，FVT测试，UAT测试，回归测试到MTP需要3天左右时间的Size的Story，现在相同工作量...

展开

作者回复: 厉害，非常棒的实践，我们也在做类似的事情，但是我们暂时没用cucumber。高手👍

fekgih
2018-07-10

15

Postman也可以集成到cicd，通过Postman+newman+jenkins。在实际项目中使用过，非常方便。平常积累的测试数据，在Postman导出一个json文件，在另一个服务器部署newman(命令行纸执行Postman导出的json文件），然后直接在服务器是用newman工具一行命令就能执行测试并生成测试报告，这样很方便集成的cicd里。

展开

作者回复: 非常棒的分享，我们以前也用过和你说的完全一样的方案，但是最终鉴于用例需要更多的灵活性，比如数据准备，更方便通用的assert等，所以我们最终还是选择了全代码的实现。👍

西海
2018-07-09

9

老师，可以谈谈Katalon Studio这个软件吗？国内使用的人很少，不知道国外使用的人多

吗?

作者回复: Katalon Studio的确是个好东西，它结合了selenium和appium，而且提供了很简单易用的IDE环境，关键还支持录制和object spy，有点像uft的感觉，无论是资深的测试开发还是初级的自动化工程师，都能比较好的上手，但是这个工具比较新，目前国内的确好像没看到有人用，中文的资料非常少，再加上它天生不支持中文。但是我觉得好东西一定会发光的，我相信很快就会慢慢普及起开。另外一个就是这个版本更新现在非常快，基本每个月都会有版本发布，我个人还是看好的

Cynthia🔥...
2018-07-09

9

对于测试来说，并没有接触到代码级测试。经历过的公司，有的即使做了，也是由开发自行做单元测试或者代码评审之类。
其他的自动化技术，有做过gui和api的自动化测试。但是gui自动化测试呢，在产品更迭迅速的互联网公司，经常页面，产品逻辑有微小变化，大多数用例就要推倒重来，维护成本太高。

展开

作者回复: 你说的太对了，相信很多人都会有你这种感觉，后面我讲gui测试的时候，会重点来看怎么缓解你说的问题

才子
2018-08-14

7

我们日常工作更多的是关注测试用例的自动执行，对于其他方面，如测试脚手架、相应判断和测试数据准备的自动化生成都没有关注。还是对于自动化的理解太片面了。

现在项目，由于组内没有统一的测试脚手架，每次新项目来时，很多精力用来制作脚手架。由于人员水平的不同，导致每个项目的测试脚手架设计水平不一，可读性差，重复...

展开

红娟
2018-07-10

4

回答问题，我所在的部门没有代码级别的测试，但是也是在努力过程中，现在是统一的环境编译，有静态代码扫描，推广自动化测试。对员工也是各种培训，有代码风格培训，有

如何编写单元测试代码。
关于自动化测试，还是要回到目的上来，自动化测试是为了提高测试效率，节约成本。解放重复劳动。所以在工作的任意阶段，只要符合这些要求，都可以用自动化来做。

展开

作者回复: 你说得非常对，一切都是经济利益导向的

huahuan b...
2018-07-09

4

我们团队在Response的自动化上花了很多功夫，但是目前来看还是有很多人工去输入要检验的字段值，检验这块如果不全面和不够自动化和易维护，结果就不可信，而且写测试用例起来也麻烦，导致很难全面取代手动测试的。

展开

齐涛-道长
2018-07-22

3

我们采用robot framework+requestslibrary的方式来做API自动化测试。个人觉得比较适合团队编程能力不是太强的团队

阿甘
2018-07-09

3

目前公司用的是jmeter接口自动化初级阶段，想要更进阶的，跟着进步学习中

展开

作者回复: 嗯嗯，我们也用jmeter

yiluo
2018-07-09

3

希望看到更多的讲解，关于用例框架代码和自动生成数据。谢谢

展开

Geek_84a77...

2018-07-09

3

文章中提到了很多自动化生成，不管是单元测试还是GUI测试，都涉及了自动化生成，这个是代码实现的自动化工具么？后期会和我们分享吗？

作者回复: 这个并不是一个现成的工具，而是方法，很多项目都会自己开发这种工具，比如前段时间我就在做把postman的测试用例，就是json文件转换成代码

yunfeng

2018-07-12

2

系统测试这一块，对着代码的某个接口测试，实际测试过程中仅仅是根据开发编写的逻辑来验证是否流程ok，结果是否符合预期就可以了，实际很难全面去测试，因为成本很大，而且每次测试模块都会变，接口的业务功能复杂在一次测试过程中很难了解全面，很想了解下系统测试这一块该如何形成一个比较全面的测试框架呢？

展开

作者回复: 往往接口测试是在单元测试完成后，当然有些也有不做单元测试而直接做接口测试的，理论上讲，接口测试关心的问题是接口的功能需求，而不用过多关注接口内部的覆盖率，但有时候也会具体代码覆盖率来指导用例设计。两者的边界不是太严格，所以我们往往往往会把这种测试称为灰盒测试，介于白盒和黑盒之间

一笑而泯

2018-07-10

2

自动生成response期望的想法不谋而合，我们也在做这方面的尝试，期待后面的课程详细介绍。

作者回复: 现在已经有工具直接支持了，不过好像不是开源的

family

2018-07-09

2

很期待后续的具体讲解自动化

展开

作者回复: 后面最先会讲的是GUI自动化

口水窝
2019-03-04

1

由于没有亲自实践过单元测试和代码级集成测试，对于单元测试，还停留在函数测试的程度上，而代码级集成测试就是接口测试，Postman的界面就是输入接口参数，来获得验证码是否在正确的过程，不知道理解的是否正确？

但是不明白后面说的，输入json格式的参数，输出符合API测试框架的测试用例，具体的怎样和CI/CD集成，麻烦老师给予指点一二？

展开

sylan215
2019-02-12

1

1.单元测试我感觉很少公司能做到吧，目前国内还是偏功能为主，速度迭代，甚至允许带 bug 上线，这个单元测试的要求相悖，单元测试的发展趋势应该是和 Google 一样，开发成了质量保证的第一环；
2.代码级集成测试，这个怎么说，开发代码联调其实算这个类型，但是他们联调的时间点和关注点，好像仅仅是和功能相关，和质量无关；...

展开

幸福花开李...
2018-09-05

1

最近公司项目需求，需要在移动端通过不同浏览器打开一些网址，不知道采用什么技术实现自动化。请大神指导，多谢

作者回复: appium+selenium

阿东
2018-08-28

1

老师，请教几个问题。1，测试脚手架代码的自动化生成。这句可不可以理解成主要是测试框架生成的测试模板，比如Python的unittest的测试模板2.部分测试输入数据的自动化生成。这一点是测试人员自己写脚本生成，还是借助测试框架？能否举几个例子？谢谢

图•美克尔

2018-07-12

1

这里用java的多还是用python的多?

展开

作者回复: 这里不限开发语言，有代码的地方也仅仅是为了说明思路，所以会尽量采用伪代码来描述问题。目前的自动化java和python基本四六开的样子。

涅槃Ls

2018-07-10

1

打卡05

展开

作者回复: 欢迎👏