**自动化测试**

|  |
| --- |
| **分层的自动化测试**    　　这个概念最近曝光度比较高，传统的自动化测试更关注的产品UI层的自动化测试，而分层的自动化测试倡导产品的不同阶段（层次）都需要自动化测试。  http://images.cnitblog.com/i/311516/201404/091113466221239.jpg    　　相信测试同学对上面的金字塔并不陌生，这不就是对产品开发不同阶段所对应的测试么！我们需要规范的来做单元测试同样需要相应的单元测试框架，如java的Junit、testNG，C#的NUnit ，python 的unittest、pytest 等，几乎所有的主流语言，都会有其对应的单元测试框架。  　　集成、接口测试对于不少测试新手来说不太容易理解，单元测试关注代码的实现逻辑，例如一个if 分支或一个for循环的实现；那么集成、接口测试关注的一是个函数、类（方法）所提供的接口是否可靠。例如，我定义一个add()函数用于计算两个参数的结果并返回，那么我需要调用add()并传参，并比较返回值是否两个参数相加。当然，接口测试也可以是url的形式进行传递。例如，我们通过get方式向服务器发送请求，那么我们发送的内容做为URL的一部分传递到服务器端。但比如 Web service 技术对外提供的一个公共接口，需要通过soapUI 等工具对其进行测试。  　　UI层的自动化测试，这个大家应该再熟悉不过了，大部分测试人员的大部分工作都是对UI层的功能进行测试。例如，我们不断重复的对一个表单提交，结果查询等功能进行测试，我们可以通过相应的自动化测试工具来模拟这些操作，从而解放重复的劳动。UI层的自动化测试工具非常多，比较主流的是QTP，Robot Framework、watir、selenium 等。  　　为什么要画成一个金字塔形，则不是长方形 或倒三角形呢？ 这是为了表示不同阶段所投入自动化测试的比例。如果一个产品从没有做单元测试与接口测试，只做UI层的自动化测试是不科学的，从而很难从本质上保证产品的质量。如果你妄图实现全面的UI层的自动化测试，那更是一个劳民伤财的举动，投入了大量人力时间，最终获得的收益可能会远远低于所支付的成本。因为越往上层，其维护成本越高。尤其是UI层的元素会时常的发生改变。所以，我们应该把更多的自动化测试放在单元测试与接口测试阶段进行。  　　既然UI层的自动化测试这么劳民伤财，那我们只做单元测试与接口测试好了。NO! 因为不管什么样的产品，最终呈现给用户的是UI层。所以，测试人员应该更多的精力放在UI层。那么也正是因为测试人员在UI层投入大量的精力，所以，我们有必要通过自动化的方式帮助我们“部分解放”重复的劳动。  　　在自动化测试中最怕的是变化，因为变化的直接结果就是导致测试用例的运行失败，那么就需要对自动化脚本进行维护；如何控制失败，降低维护成本对自化的成败至关重要。反过来讲，一份永远都运行成功的自动化测试用例是没有价值。  　　至于在金字塔中三种测试的比例要根据实际的项目需求来划分。在《google 测试之道》一书，对于google产品，70%的投入为单元测试，20%为集成、接口测试，10% 为UI层的自动化测试。 |
| 首先考考虑产品是否适合做自动化测试。这方法比较普遍的共识是从三个方面进行权衡。  　　软件需求变动不频繁  　　测试脚本的稳定性决定了自动化测试的维护成本。如果软件需求变动过于频繁，测试人员需要根据变动的需求来更新测试用例以及相关的测试脚本，而脚本的维护本身就是一个代码开发的过程，需要修改、调试，必要的时候还要修改自动化测试的框架，如果所花费的成本不低于利用其节省的测试成本，那么自动化测试便是失败的。  　　项目中的某些模块相对稳定，而某些模块需求变动性很大。我们便可对相对稳定的模块进行自动化测试，而变动较大的仍是用手工测试。  　　项目周期较长  由于自动化测试需求的确定、自动化测试框架的设计、测试脚本的编写与调试均需要相当长的时间来完成。这样的过程本身就是一个测试软件的开发过程，需要较长的时间来完成。如果项目的周期比较短，没有足够的时间去支持这样一个过程，那么自动化测试便成为笑谈。  　　自动化测试脚本可重复使用  　　自动化测试脚本的重复使用要从三个方面来考量，一方面所测试的项目之间是否很大的差异性（如C/S系统和B/S系统的差异）；所选择的测试工具是否适应这种差异；最后，测试人员是否有能力开发出适应这种差异的自动化测试框架。 |
| 选择什么工具进行自动化测试  　　假如你已经确认了XX 项目适合做自动化测试，那么接下来你要做的就是选测试工具了。  　　首先要先确认你所测试的产品是桌面程序（C/S）还是web应用（B/S）。  　　桌面程序的工具有：QTP、 AutoRunner  　　web应用的工具有：QTP、AutoRunner、Robot Framework、watir、selenium  　　由于B/S架构的诸多优势，早几年前大量C/S架构的应用转为B/S结构。从而也推动了web开发与测试技术的发展。假如，被测试有产品是C/S架构的，那么推荐QTP ，QTP在UI自动化测试领域占到了一半的试用率。所以，足以说明QTP在自动化领域强大，易用性等。学习主流的工具也可以使你获得更多的机会。市面上关于QTP的书籍也非常丰富。当然，要想学好QTP ，你必须要掌握VBS脚本语言。  　　如果，被测产品是B/S 结构，那么推荐selenium ，为什么不是QTP 或其它工具？因为selenium 对B/S应用支持很好，更重要的一点，它支持多语言的开发，真正的试用selenium ，你所要掌握的不仅仅是一个工具而已，你还需要学习一门语言。我为什么要选择selenium？还要学一门语言，这无疑增加了我的学习成本。增加成本的同时，也增加的你的竞争力，而且，在这个过程中你不单单只是学会了一个自动化工具而已，你完全可以使用所学的语言去做更多的事情。  　　好吧！假如你决定试用selenium 了之后，你又面临了一个新的问题，选择一门语言。selenium 是支持java、python、ruby、php、C#、JavaScript 。  　　从语言易学性来讲，首选ruby ，python  　　从语言应用广度来讲，首选java、C#、php、  　　从语言相关测试技术成度（及 资料）来讲：ruby ,python ,java  　　或者你可以考虑整个技术团队主流用什么语言，然后选择相应的语言。 |
| selenium 用前须知  　　OK！经过上的过程，我相信你一定做出的相应的选择，如果你选择的是selenium 工具，那么接着往下阅读。  首选你在开始selenium之前，需要花一到两个月时间去学一门语言，这里是根据没有语言基础的同学而定的。我推荐ruby ,python ,java 任意一门语言来进行学习。  　　当然，已经如果有很好的语言基础略过这个环节，或者你的丰富的java编程能力，那么学习python 可能只需要几天时间或更短。  　　假如，你已经搞定了一门语言的基础，接下来你需要先了解selenium ，selenium 并不是单纯的一个工具，他是一组工具的集合，而且，他还有1.0与2.0之分，当然3.0也已经到来。  　　selenium 也不是简单一个工具，而是由几个工具组成，每个工具都有其特点和应用场景。  selenium IDE  　　selenium IDE 是嵌入到Firefox浏览器中的一个插件，实现简单的浏览器操作的录制与回放功能。那么什么情况下用到它呢？  　　快速的创建bug重现脚本，在测试人员的测试过程中，发现了bug之后可以通过IDE将重现的步骤录制下来，以帮助开发人员更容易的重现bug。  　　IDE录制的脚本可以可以转换成多种语言，从而帮助我们快速的开发脚本，关于这个功能后而用到时再详细介绍。    selenium Grid  　　Selenium Grid是一种自动化的测试辅助工具，Grid通过利用现有的计算机基础设施，能加快Web-app的功能测试。利用Grid，可以很方便地同时在多台机器上和异构环境中并行运行多个测试事例。其特点为：  · 并行执行  · 通过一个主机统一控制用例在不同环境、不同浏览器下运行。  · 灵活添加变动测试机    selenium RC  　　selenium RC 是selenium 家族的核心工具，selenium RC 支持多种不同的语言编写自动化测试脚本，通过selenium RC 的服务器作为代理服务器去访问应用从而达到测试的目的。  　　selenium RC 使用分Client Libraries和selenium Server，Client Libraries库主要主要用于编写测试脚本，用来控制selenium Server的库。  　　Selenium Server负责控制浏览器行为，总的来说，Selenium Server主要包括3个部分：Launcher、Http Proxy、Core。其中Selenium Core是被Selenium Server嵌入到浏览器页面中的。其实Selenium Core就是一堆JS函数的集合，就是通过这些JS函数，我们才可以实现用程序对浏览器进行操作。Launcher用于启动浏览器，把selnium Core加载到浏览器页面当中，并把浏览器的代理设置为Selenium Server 的Http Proxy。    selenium 2.0  　　搞清了selenium 1.0 的家族关系，selenium 2.0 是把WebDriver 加入到了这个家族中；简单用公式表示为：  　　selenium 2.0 = selenium 1.0 + WebDriver  　　需要强调的是，在selenium 2.0 中主推的是WebDriver ，WebDriver 是selenium RC 的替代品，因为 selenium 为了向下兼容性，所以selenium RC 并没有彻底抛弃，如果你使用selenium开发一个新自动化测试项目，强列推荐使用WebDriver 。那么selenium RC 与webdriver 主要有什么区别呢？  　　selenium RC 在浏览器中运行JavaScript应用，使用浏览器内置的JavaScript 翻译器来翻译和执行selenese命令（selenese 是selenium命令集合）。  　　WebDriver通过原生浏览器支持或者浏览器扩展直接控制浏览器。WebDriver针对各个浏览器而开发，取代了嵌入到被测Web应用中的JavaScript。与浏览器的紧密集成支持创建更高级的测试，避免了JavaScript安全模型导致的限制。除了来自浏览器厂商的支持，WebDriver还利用操作系统级的调用模拟用户输入。  　　如果是新项目直接学习webdriver 就OK了，RC是过时技术。 |
| selenium学习路线  　　配置你的测试环境，真对你所学习语言，来配置你相应的selenium 测试环境。selenium 好比定义的语义---“问好”，假如你使用的是中文，为了表术问好，你的写法是“你好”，假如你使用的是英语，你的写法是“hello”。 所以，同样有语义在不同的语言下会有不同的写法（语法）。  　　接着你需要熟悉webdriver API ，API就是selenium 所定义一方法，用于定位，操作页面上的各种元素。  　　先学习元素的定位，selenium 提供了id、name、class name、 tag name、link text、partial link text、 xpath、css、等定位方法。xpath和css 功能强大语法稍微复杂，在这其间你可能还需要了解更多的前端知识。xml ,javascript 等。  　　定位元素的目的是为了操作元素，接就要学习各种元素有操作，输入框，下拉框，按钮点击，文件上传、下载，分页，对话框，警告框...等等。  　　经过一段时间的学习，你可以游刃有余的模拟手工测试来操作页面上的各种元素了。接着你需要做的就是把这些“用例”组织起来，统一来跑。  　　那么你需要做的就是学习并使用单元测试框架，单元测试框架本身就解决了用例的组织与运行。  　　当你写了一些“测试用例” 之后，你会发现用例中有大量重复的操作，能不能写到一个单独的文件中，需要的时候调用这些操作？当然可以，运用你的编程能力来实现这一点将非常简单。然后，你又发现每个用例中都有一些数据，这些数据也是一样的，但如果变化了修改起来非常麻烦，你也可以把他写到一个单独的文件中进行读取。  　　接着你又遇到了新的疑问，我写的脚本（用例）都是流水式的，我怎么知道用例运行失败还是成功。那么就需要在脚本中加一些验证与断言。  　　接着你又有了更多的想法，单元测试框架的log太简陋了，能不能生成一张漂亮的测试报告出来。我能不能定时的来跑这个脚本。能不能把每一次跑脚本的测试结果直接发到我的邮箱。能不能......  　　为解决这些问题，你不得不学习更多的编程技术，然后你的“测试结构”会功能越来越强大，越来越灵活。产生了一定的通用性和移植性。一个有模有样的自动化测试框架诞生了。  　　假如，有一天你不再做UI的自动化测试了，你会发现你去做单元测试 或接口测试基本没什么难度。开发个测试工具之类的也不在话下，感谢selenium 吧！顺便也感谢一下我吧！ |