**自动化测试**

**什么是自动化测试？**

把人对软件的测试行为转化为由机器执行测试行为的一种实践，对于最常见的 GUI 自动化测试来讲，就是由自动化测试工具模拟之前需要人工在软件界面上的各种操作，并且自动验证其结果是否符合预期。

自动化测试用例的维护成本高于其节省的测试成本时，自动化测试就失去价值与意义，在这样的项目中推进自动化测试就会得不偿失.

**为什么需要自动化测试(优点)？**

可以替代大量的手工机械重复性操作，测试工程师可以把更多的时间花在更全面的用例设计和新功能的测试上；

自动化测试可以大幅提升回归测试的效率，非常适合敏捷开发过程；

可以更好地利用无人值守时间，去更频繁地执行测试，特别适合现在非工作时间执行测试，工作时间分析失败用例的工作模式；

可以高效实现某些手工测试无法完成或者代价巨大的测试类型，比如关键业务 7×24 小时持续运行的系统稳定性测试和高并发场景的压力测试等；

可以保证每次测试执行的操作以及验证的一致性和可重复性，避免人为的遗漏或疏忽。

**自动化测试有哪些劣势?**

自动化测试并不能取代手工测试，它只能替代手工测试中执行频率高、机械化的重复步骤。不要奢望所有的测试都自动化，否则一定会得不偿失。

自动测试远比手动测试脆弱，无法应对被测系统的变化，自动化测试用例的维护成本一直居高不下。对于执行过程中出现的明显错误和意外事件，自动化测试没有任何处理能力。

自动化测试用例的开发工作量远大于单次的手工测试.

手工测试发现的缺陷数量通常比自动化测试要更多，并且自动化测试仅仅能发现回归测试范围的缺陷。

测试的效率很大程度上依赖自动化测试用例的设计以及实现质量，不稳定的自动化测试用例实现比没有自动化更糟糕。

实行自动化测试的初期，用例开发效率通常都很低，大量初期开发的用例通常会在整个自动化测试体系成熟，和测试工程师全面掌握测试工具后，需要重构。

业务测试专家和自动化测试专家通常是两批人，前者懂业务不懂自动化技术，后者懂自动化技术但不懂业务，只有二者紧密合作，才能高效开展自动化测试。

自动化测试开发人员必须具备一定的编程能力，这对传统的手工测试工程师会是一个挑战。

**什么样的项目适合自动化测试？**

1. 需求稳定，不会频繁变更。需求不稳定，需求变更频繁, 界面变化，或者是业务流程变化频繁不适合.
2. 研发和维护周期长，需要频繁执行回归测试。对于短期的一次性项目，我觉得你应该选择手工探索式测试，以发现缺陷为第一要务。而对于一些中长期项目，我的建议是：对比较稳定的软件功能进行自动化测试，对变动较大或者需求暂时不明确的功能进行手工测试，最终目标是用 20% 的精力去覆盖 80% 的回归测试。
3. 需要在多种平台上重复运行相同测试的场景.
4. 测试项目通过手工测试无法实现或手工成本太高。所有的性能和压力测试，很难通过手工方式实现。
5. 被测软件的开发较为规范，能够保证系统的可测试性。
6. 测试人员已经具备一定的编程能力。

**软件开发各阶段都有哪些自动化测试技术?(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)**

**单元测试的自动化技术包括:**不仅仅指测试用例执行的自动化，还包含以下五个方面：

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，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。)；

部分测试输入数据的自动化生成；自动桩代码的生成；被测代码的自动化静态分析(常用的代码静态分析工具有 Sonar 和 Coverity )；测试覆盖率的自动统计与分析。

**代码级集成测试的自动化技术:**

指将已经开发完成的软件模块放在一起测试, 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。

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.

**Web Service 测试的自动化技术:**

指 SOAP API 和 REST API 这两类 API 测试，典型是采用 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。

**对于基于代码的 API 测试用例，通常包含三大步骤：**准备 API 调用时需要的测试数据；准备 API 的调用参数并发起 API 的调用；验证 API 调用的返回结果。

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.

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：

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

**GUI 测试的自动化技术**

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。

**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。二者采用的具体技术差别很大，用例设计的思路类似。Web端有selenium,** **Micro Focus 的 UFT（前身是 HP 的 QTP）.移动端有Appium等.**

GUI 自动化测试的技术、原理和行业最佳实践。

 GUI 测试中两个非常重要的概念：测试脚本和数据的解耦，以及页面对象（Page Object）模型.

**数据驱动(Data-driven)测试:**

测试脚本只有一份，其中需要输入数据的地方会用变量来代替，然后把测试输入数据单独放在一个文件中。这个存放测试输入数据的文件，通常是表格的形式，也就是最常见的 CSV 文件。然后，在测试脚本中通过 data provider 去 CSV 文件中读取一行数据，赋值给相应的变量，执行测试用例。接着再去 CSV 文件中读取下一行数据，读取完所有的数据后，测试结束。CSV 文件中有几行数据，测试用例就会被执行几次。

本质是实现了数据驱动的测试，让操作相同但是数据不同的测试可以通过同一套自动化测试脚本来实现，只是在每次测试执行时提供不同的测试输入数据。

**数据驱动的好处:** 很好地解决了大量重复脚本的问题，实现了“测试脚本和数据的解耦”。数据驱动测试的数据文件中不仅可以包含测试输入数据，还可以包含测试验证结果数据，甚至可以包含测试逻辑分支的控制变量。数据驱动测试的思想不仅适用于 GUI 测试，还可以用于 API 测试、接口测试、单元测试等。

**页面对象模型(Page Object)**的核心理念是，以页面（Web Page 或者 Native App Page）为单位来封装页面上的控件以及控件的部分操作。而测试用例，更确切地说是操作函数，基于页面封装对象来完成具体的界面操作，最典型的模式是“XXXPage.YYYComponent.ZZZOperation”。

**如何把控操作函数的粒度？**

操作函数的粒度是指，一个操作函数到底应该包含多少操作步骤才是最合适的。以完成一个业务流程（business flow）为主线，抽象出其中的“高内聚低耦合”的操作步骤集合，操作函数就由这些操作步骤集合构成。

**如何衔接两个操作函数之间的页面？**

如果连续的两个操作函数之间无法用页面衔接，那就需要在两个操作函数之间加入额外的页面跳转代码，或者是在操作函数内部加入特定的页面跳转代码。

**GUI测试创建测试数据的方法:**从创建的技术手段上来讲，创建测试数据的方法主要分为三种：API 调用；数据库操作；综合运用 API 调用和数据库操作。

从创建的时机来讲，创建测试数据的方法主要分为两种：测试用例执行过程中，实时创建测试数据，我们通常称这种方式为 On-the-fly。测试用例执行前，事先创建好“开箱即用”的测试数据，通常称这种方式为 Out-of-box。

对于页面对象自动生成，商用测试软件已经实现。如果选择开源测试框架，就需要自己实现这个功能了。

GUI 测试数据自动生成，主要是基于测试输入数据的类型以及对应的自定义规则库实现的，并且对于多个测试输入数据，可以基于笛卡尔积来自动组合出完整的测试用例集合。

对于无头浏览器，你可以把它简单地想象成运行在内存中的浏览器，它拥有完整的浏览器内核。与普通浏览器最大的不同是，它在执行过程中看不到运行的界面。目前，Headless Chrome 结合 Puppeteer 是最先进的无头浏览器方案，如果感兴趣，你可以下载试用

**GUI测试的稳定性:**

要提高 GUI 测试稳定性，首先你需要知道到底是什么原因引起的不稳定。你必须找出尽可能多的不稳定因素，然后找到每一类不稳定因素对应的解决方案。

**五种造成 GUI 测试不稳定的因素**：**非预计的弹出对话框；页面控件属性的细微变化；被测系统的 A/B 测试,随机的页面延迟造成控件识别失败；测试数据问题。**

**非预计的弹出对话框可以:** (1) 当自动化脚本发现控件无法正常定位，或者无法操作时，GUI 自动化框架自动**进入“异常场景恢复模式**. 在“异常场景恢复模式”下，GUI 自动化框架依次检查各种可能出现的对话框，一旦确认了对话框的类型，立即执行预定义的操作（比如，单击“确定”按钮，关闭这个对话框），接着重试刚才失败的步骤。而对于新类型的对话框，只能通过自动化的方式尝试点击上面的按钮进行处理。每当发现一种潜在会弹出的对话框，我们就把它的详细信息（包括对象定位信息等）更新到“异常场景恢复”库中，下次再遇到相同类型的对话框时，系统就可自动关闭。

**页面控件属性的细微变化: 采用“组合属性”定位控件会更精准**，而且成功率会更高，如果能在**此基础上加入“模糊匹配”技术**，可以进一步**提高控件的识别率**。但是，开源的 GUI 自动化测试框架，目前还没有现成的框架直接支持模糊匹配，通常需要你进行二次开发，实现思路是：实现自己的对象识别控制层，也就是在原本的对象识别基础上额外封装一层，在这个额外封装的层中加上模糊匹配的实现逻辑。通常，不建议把模糊匹配逻辑以硬编码的方式写在代码里，而是引入规则引擎，将具体的规则通过配置文件的方式与代码逻辑解耦。

**被测系统的 A/B 测试:** 在测试脚本内部对不同的被测版本做分支处理，脚本需要能够区分 A 和 B 两个的不同版本，并做出相应的处理。

**随机的页面延迟造成控件识别失败:** **加入重试（retry）机制**。重试机制是指，当某一步 GUI 操作失败时，框架会自动发起重试，重试**可以是步骤级别的**，**也可以是页面级别的**，甚至是业务流程级别的。对于开源 GUI 测试框架，重试机制往往不是自带的功能，需要自己二次开发来实现。需要特别注意的是，对于那些会修改一次性使用数据的场景，切忌不要盲目启用页面级别和**业务流程级别**的重试.

**测试数据问题:略.**

**GUI 自动化测试报告**

**理想中的 GUI 测试报告**应该是由一系列按时间顺序排列的屏幕截图组成，并且这些截图上可以高亮显示所操作的元素，同时按照执行顺序配有相关操作步骤的详细描述。

**开源 GUI 测试框架的测试报告实现思路：**

利用 Selenium WebDriver 的 screenshot 函数在一些特定的时机（比如，页面发生跳转时，在页面上操作某个控件时，或者是测试失败时，等等）完成界面截图功能。具体到代码实现，**通常有两种方式：扩展 Selenium 原本的操作函数；在相关的 Hook 操作中调用 screenshot 函数。**

1. **扩展 Selenium 原本的操作函数实现截图以及高亮显示操作元素的功能：**

**自己实现的 click 函数被调用时：**首先，用 Javascript 代码高亮显示被操作的元素，高亮的实现方式就是利用 JavaScript 在对象的边框上渲染一个 5-8 个像素的边缘；然后，调用 screenshot 函数完成点击前的截图；最后，调用 Selenium 原生的 click 函数完成真正的点击操作。那么，以后凡是需要调用 click 函数时，都直接调用这个自己封装的 click 函数，直接得到高亮了被操作对象的界面截图。

1. **在相关的 Hook 操作中调用 screenshot 函数实现截图以及高亮显示操作元素的功能。**

**大型全球化电商网站的 GUI 自动化测试如何开展（如果你所在的企业或者项目正在大规模开展 GUI 测试，并且准备使用页面对象模型以及业务流程封装等最佳实践的话）：**

**GUI 测试通常只覆盖最核心且直接影响主营业务流程的 E2E 场景。**按照自底向上的顺序分层次介绍 GUI 自动化的测试策略。

首先，要从前端组件的级别来保证质量，也就是需要对那些自定义开发的组件进行完整全面的测试。公共组件库会被很多上层的前端模块依赖，它的质量将直接影响这些上层模块的质量，所以我们往往会对这些公共组件进行严格的单元测试。最常用的方案是：基于 Jest 开展单元测试，并考量 JavaScript 的代码覆盖率指标。

完成单元测试后，往往还会基于被测控件构建专用的测试页面，在页面层面再次验证控件相关的功能和状态。这部分测试工作也需要采用自动化的形式实现，具体的做法是：

1. 先构建一个空页面，并加入被测控件，由此可以构建出一个包含被测控件的测试页面，这个页面往往被称为 Dummy Page；
2. 从黑盒的角度出发，在这个测试页面上通过手工和自动化的方式操作被测控件，并验证其功能的正确性。

对于自动化的部分，需要基于 GUI 自动化测试框架开发对应的测试用例。这些测试用例，往往采用和 GUI E2E 一样的测试框架，也是从黑盒的角度来对被测控件做功能验证。

**其次，每一个前端模块，都会构建自己的页面对象库，并且在此基础上封装开发自己的业务流程脚本。这些业务流程的脚本，可以组装成每个前端模块的测试用例。最后，组合各个前端模块，并站在终端用户的视角，以黑盒的方式使用网站的端到端（E2E）测试。**

掌握一门编程语言Python或Java，一定的前端知识。Web自动化测试的selenium环境的搭建，webdriver元素定位，app自动化测试的Appium环境搭建，webdrive元素定位，常见的自动化测试框架，对于接口自动化测试，python实现的话需要掌握requests库和unittest框架。

**框架( Framwork )**：是构成一类特定软件可复用设计的一组相互协作的类。框架规定了你的应用的体系结构。它定义了整体结构，类和对象的分割，各部分的主要责任，类和对象怎么协作，以及控制流程。框架预定义了这些设计参数，以便于应用设计者或实现者能集中精力于应用本身的特定细节。

**自动化测试框架**：是一个工具脚本集合，用于组织和管理自动化基础功能（打开浏览器、鼠标点击、登录等）自动化测试用例、提供测试结果统计报告。自动化测试框架能够提供可靠的测试服务，对于新需求能够快速扩展，能够让测试工程师简单高效的完成自动化测试任务。国内的阿里Macaca,腾讯开源微信小程序自动化测试工具,百度内部的Café框架.测试平台包括阿里MQC,腾讯WeTest,百度MTC,Testin

**测试框架的优点：**代码复用 ，将基础的测试代码封装，从而降低代码的复杂性。提高维护效率，有效组织和管理测试脚本。快速实现项目的自动化测试，不用从0开始，一般测试框架完成后，其他类似程序也可以快速复用。输出各种美观易懂的测试报告。

**自动化测试的目的：**缩短测试周期，节省成本。避免人为出错，提高准确性和可靠性。获取需求覆盖率，代码覆盖率，提供衡量软件质量的指标。

**自动化测试的意义：**手工测试完成后。项目周期长，需求稳定。频繁的回归测试。

**合理的使用UI自动化:**

**单元测试:使用app分层测试策略,增加单测规模,控制UI自动化测试规模.**

**移动app自动化测试:核心集成与验收测试用例交给自动化测试.**

**自动化遍历测试:大部分的基础回归测试交给自动遍历.**

**人工测试:测试团队主负责,整个产品团队参与.**

**典型的自动化测试框架目录：**

config(配置文件目录)：存放所有使用的配置文件，实现配置与代码分离。

common(公共函数)：公共函数、方法以及通用操作的管理。

data(测试数据)：将所有的用例参数化使用的文件放到这里，一般可采用xlsx、csv、xml等格式。实现数据与代码分离。

drivers（驱动目录）：一般存放浏览器驱动，如如Chromedriver等。

logs（日志目录）：日志记录和管理功能，存放运行时日志和错误日志error log等。

testcase（测试用例）：测试用例管理功能，可以分模块编写，建相应的目录。

report（测试报告）：管理和存放程序运行后生成的测试报告，一般可有html报告、excel报告等。

**python主流测试框架：**

**unittest框架：**

python内置的单元测试框架，unittest的设计灵感最初来源于Junit以及其他语言中具有共同特征的单元框架。它能编写组织用例、执行用例，还可输出测试报告等。

**优点**：使用断言判断返回布尔值来判断期望值和实际值的差异。可以构建共同的初始化变量或实例。框架结构可以组织用例批量运行。

**Unittest的几个概念：**test fixture：一个test fixture 表示执行一个或多个测试前的准备工作，以及执行完成后清理工作。例如：创建临时或代理数据库或目录，或者是一个启动服务器进程。

test case 一个test case是一个独立的测试单元，它检查特定输入是否响应特定的输出，unittest提供了一个基本类——TestCase，这个类用于创建一个或多个test case。

test suite：一个test suite是test cases、或test suites或两者的一个集合，用于把想执行tests放在一起。

test runner： 一个test runner 是由两部分成分：合理安排tests的执行、提供给user输出结果。runner可以用一个图形界面、文本、或者是一个特殊的值，代表tests执行的结果。

**Pytest：**

很好的强大的单元测试框架，实际上广泛使用在自动化单元、接口、功能等测试。如果你的项目比较小、复杂度比较低，Pytest是最适合的自动化测试平台。非常多的Python开发者都喜欢它，他们大部分用它来进行单元测试。Pytest最好的特性之一是，它提供了测试用例的详细失败信息，使得开发者可以快速准确地改正问题。它兼容最新版本的Python。它还兼容unittest、doctest和nose，开箱即用。它还有包含更多功能的插件和多样化的现有测试技术和测试用例。官方文档：<https://pytest.org/en/latest/>

**Robot Framework：**

通用的python测试框架，提供一个图形用户界面，方便管理和编辑测试用例，易于上手，生成的报告比较好看，支持关键字和数据等驱动。Robot Framework是独立于操作系统和应用程序，支持跨平台，例如Windows、MacOS和Linux。这个产品是由世界上一些著名的测试人员创建的，拥有关键词驱动方案。它有丰富的工具和库，使得这个框架非常先进和健壮。

**自动化测试**：是把以人为驱动的测试行为转化为机器执行的一种过程。通常，在设计了测试用例并通过评审之后，由测试人员根据测试用例中描述的规程一步步执行测试，得到实际结果与期望结果的比较。自动化测试的主要目标是提高测试效率，降低项目成本，而不是完全取消手工测试。

**自动化测试的意义：**提升效率，减少重复工作：勿容置疑，最大的意义就是提高测试效率，手工测试的最大问题在于，面对快速迭代，无法快速完整的执行冒烟用例。节省人力成本：执行测试脚本可以实现无人值守，不限时间的测试。从而让测试人员可以做更多有意义事情，比如探索性测试等。保证每次测试的完整性和一致性。

发现更多隐藏问题：手工测试无法验证系统的稳定性、可靠性等，需要通过工具等自动化手段，对系统进行压力测试、稳定性测试等。

**Selenium**

Web测试框架，丰富的API，支持多种语言编写测试脚本且可在多种浏览器执行测试脚本。

对 Selenium 而言，V1.0 和 V2.0 版本的技术方案是截然不同的，V1.0 的核心是 Selenium RC，而 V2.0 的核心是 WebDriver, V3.0 相比 V2.0 并没有本质上的变化，主要是增加了对 MacOS 的 Safari 和 Windows 的 Edge 的支持，并彻底删除了对 Selenium RC 的支持。

**webdriver：**server -client设计模式设计的。server端即浏览器。当我们的脚本启动浏览器后，等待client发送请求并做出相应；client端即测试代码.

**搭建环境**:本地安装Python,在官网下载,安装时勾选path,安装完成后,在cmd中输入python,查看是否进入.

安装selenium,在cmd中输入pip install selenium

安装Chrome Webdriver(<https://sites.google.com/a/chromium.org/chromedriver/downloads>).

**不同浏览器的driver:**

browser = webdriver.Chrom()

browser = webdriver.Firefox() browser = webdriver.Safari() browser = webdriver.Ie()

**基本模板:**

from selenium import webdriver

browser = webdriver.Chrome(executable\_path="..\..\..\zc\chromedrive\chromedriver.exe")

# executable\_path来指定chromedirver路径

browser.get('https://www.baidu.com')

print(browser.title)

browser.quit()

**八种定位方法:**

**id, name, class name,tag name,link text,partial link text,xpath,css selector**

定位方法find\_element\_by\_xx()依次类推.如: find\_element\_by\_xpath()

**浏览器操作:**

set\_window\_size()方法来设置浏览器的大小 , maximize\_window() 全屏显示.如:browser.set\_window\_size(480,800)

back()和forward()方法来模拟后退和前进按钮, refresh()方法模拟F5刷新页面.

**元素操作:**

Clear()方法,清除文本. Send\_keys(‘value’)方法,模拟键盘输入. Click()方法,单击元素.

submit()方法用于提交表单, (submit()可与 click()方法互换来使用， submit()同样可以提交一个按钮但 submit()的应用范围远不及 click()广泛)

size： 返回元素的尺寸. text： 获取元素的文本. get\_attribute(name)： 获得属性值。is\_displayed()： 设置该元素是否用户可见。

**鼠标事件:**在 WebDriver中鼠标操作的方法封装在 ActionChains 类提供。常用方法如下:

perform()： 执行所有 ActionChains 中存储的行为；move\_to\_element()： 鼠标悬停。

context\_click()： 右击； double\_click()： 双击； drag\_and\_drop()： 拖动；

**如:**

# 引入 ActionChains 类

from selenium.webdriver.common.action\_chains import ActionChains

# 定位到要悬停的元素

cc = browser.find\_element\_by\_xpath('//\*[@id="content"]/div/div[1]/div[2]/ul/li[4]/a')

# 对定位到的元素执行鼠标悬停操作

ActionChains(browser).double\_click(cc).perform()

**键盘事件**: Keys()类提供了键盘上几乎所有按键的方法.

# 引入 Keys 模块

from selenium.webdriver.common.keys import Keys

**#常用键盘方法:**

send\_keys(Keys.BACK\_SPACE) 删除键（BackSpace）

send\_keys(Keys.SPACE) 空格键(Space)

send\_keys(Keys.TAB) 制表键(Tab)

send\_keys(Keys.ESCAPE) 回退键（Esc）

send\_keys(Keys.ENTER) 回车键（Enter）

send\_keys(Keys.CONTROL,'a') 全选（Ctrl+A）

send\_keys(Keys.CONTROL,'c') 复制（Ctrl+C）

send\_keys(Keys.CONTROL,'x') 剪切（Ctrl+X）

send\_keys(Keys.CONTROL,'v') 粘贴（Ctrl+V）

send\_keys(Keys.F1) 键盘 F1

send\_keys(Keys.F12) 键盘 F12

**断言与等待**:

title：用于获得当前页面的标题。 current\_url：用户获得当前页面的URL。 text：获取搜索条目的文本信息。

WebDriver提供两种类型的等待：显式等待和隐式等待。

显式等待:使WebdDriver等待某个条件成立时继续执行，否则在达到最大时长时抛出超时异常（TimeoutException）。

隐式等待:WebDriver提供了implicitly\_wait()方法来实现隐式等待，默认设置为0。

driver ：浏览器驱动。timeout ：最长超时时间，默认以秒为单位。

poll\_frequency ：检测的间隔（步长）时间，默认为0.5S。

ignored\_exceptions ：超时后的异常信息，默认情况下抛NoSuchElementException异常。

WebDriverWait()一般由until()或until\_not()方法配合使用，下面是until()和until\_not()方法的说明。

until(method, message=‘’)

调用该方法提供的驱动程序作为一个参数，直到返回值为True。

until\_not(method, message=‘’)

调用该方法提供的驱动程序作为一个参数，直到返回值为False。

**定位一组元素**:WebDriver还提供8种用于定位一组元素的方法。

find\_elements\_by\_id()

find\_elements\_by\_name()

find\_elements\_by\_class\_name()

find\_elements\_by\_tag\_name()

find\_elements\_by\_link\_text()

find\_elements\_by\_partial\_link\_text()

find\_elements\_by\_xpath()

find\_elements\_by\_css\_selector()

**多表单切换与多窗口切换,警告框处理:**

WebDriver只能在一个页面上对元素识别与定位，对于frame/iframe表单内嵌页面上的元素无法直接定位。需要通过switch\_to.frame()方法将当前定位的主体切换为frame/iframe表单的内嵌页面中。

如:browser.switch\_to.frame(‘xx’)

switch\_to.frame() 默认可以直接取表单的id 或name属性。如果iframe没有可用的id和name属性，则可以通过下面的方式进行定位。

#先通过xpth定位到iframe

xf = driver.find\_element\_by\_xpath('//\*[@id="x-URS-iframe"]')

#再将定位对象传给switch\_to.frame()方法

driver.switch\_to.frame(xf)

……

driver.switch\_to.parent\_frame()

除此之外，在进入多级表单的情况下，还可以通过switch\_to.default\_content()跳回最外层的页面。

WebDriver提供了switch\_to.window()方法，可以实现在不同的窗口之间切换。

current\_window\_handle：获得当前窗口句柄。

window\_handles：返回所有窗口的句柄到当前会话。

switch\_to.window()：用于切换到相应的窗口，与上一节的switch\_to.frame()类似，前者用于不同窗口的切换，后者用于不同表单之间的切换。

WebDriver中处理JavaScript所生成的alert、confirm以及prompt十分简单，具体做法是使用 switch\_to.alert 方法定位到 alert/confirm/prompt，然后使用text/accept/dismiss/ send\_keys等方法进行操作。

text：返回 alert/confirm/prompt 中的文字信息。 accept()：接受现有警告框。

dismiss()：解散现有警告框。 send\_keys(keysToSend)：发送文本至警告框。keysToSend：将文本发送至警告框。

**下拉框选择:**

WebDriver提供了Select类来处理下拉框, Select类用于定位select标签。

select\_by\_value() 方法用于定位下接选项中的value值。

send\_keys()方法来实现文件上传

**cookie操作:**

get\_cookies()： 获得所有cookie信息。

get\_cookie(name)： 返回字典的key为“name”的cookie信息。

add\_cookie(cookie\_dict) ： 添加cookie。“cookie\_dict”指字典对象，必须有name 和value 值。

delete\_cookie(name,optionsString)：删除cookie信息。“name”是要删除的cookie的名称，“optionsString”是该cookie的选项，目前支持的选项包括“路径”，“域”。

delete\_all\_cookies()： 删除所有cookie信息。

WebDriver提供了execute\_script()方法来执行JavaScript代码。

WebDriver提供了截图函数get\_screenshot\_as\_file()来截取当前窗口。

**关闭浏览器:**

close() 关闭单个窗口

quit() 关闭所有窗口

**Appium**

**移动自动化测试基础:**

**自动化测试的价值,常见误区,技术体系,UI分层测试体系,Android app的常见概念,adb Android调试桥,测试常用命令**

**Appium框架知识点,技术架构与原理,如何封装定制与改进,Android,ios,手机浏览器,微信小程序.**

**PageObject模式,App的实战,自动遍历测试,兼容性测试在内的验收测试体系,开源的多设备调度,远程交互调试的自动化测试平台,持续集成.**

**待测产品app解析,自动化测试框架,框架原理解析,二次二次封装定制,Android自动化测试,ios自动化测试.**

**微信小程序自动化测试,手机浏览器自动化测试,Page Object设计模式,实战演练环节.**

**App结构分析,自动化测试框架,**

一个自动化测试开源工具，支持 iOS 平台和 Android 平台上的原生应用，web应用和混合应用。

“移动原生应用”是指那些用iOS或者 Android SDK 写的应用（Application简称app）。

“移动web应用”是指使用移动浏览器访问的应用（appium支持iOS上的Safari和Android上的 Chrome）。

“混合应用”是指原生代码封装网页视图——原生代码和 web 内容交互。比如，像 Phonegap，可以帮助开发者使用网页技术开发应用，然后用原生代码封装，这些就是混合应用。重要的是，appium是一个跨平台的工具：它允许测试人员在不同的平台（iOS，Android）使用同一套API来写自动化测试脚本，这样大大增加了iOS和Android测试套件间代码的复用性。

**特点:**

appium是跨平台的，可以用在OSX，Windows以及Linux桌面系统上运行。appium类库封装了标准Selenium客户端类库，为用户提供所有常见的JSON格式selenium命令以及额外的移动设备控制相关的命令，如多点触控手势和屏幕朝向。appium选择了Client/Server的设计模式。只要client能够发送http请求给server，那么的话client用什么语言来实现都是可以的，这就是appium及Selenium(WebDriver)如何做到支持多语言的原因；

**Appium生态工具:**

**Appium Desktop:**内嵌Appium Server和Inspector的综合工具.**Appium Server :**Appium的核心工具,命令行工具.

**Appium Clients:**各种语言的客户端封装库,用于连接appium server.**Appium Crawler**自动遍历工具

**Appium 的实现原理**：

它属于 C/S 架构，Appium Client 通过第三方库向 Appium Server 发起请求，Appium Server 接受请求，然后和移动平台上的代理工具打交道，代理工具在运行过程中不断接收来自 Appium Server 的请求，并解析出要执行的操作，最后调用移动平台原生的测试框架完成测试操作

**Win10环境搭建:**

1.安装 jdk，配置环境变量.

2.安装 npm install -g appium(Appium Server安装),先安装nvm,再设置切换node和npm的源,在切换npm为淘宝源.

3.安装 android studio 开发工具,配置环境变量.

4.安装npm install -g appium-doctor 查看环境配置有无问题.

5.安装Python与pip3 install Appium-Python-Client 和IDE选VSCode,Python多版本隔离工具选env,测试框架有unittest,pytest,nose.选pytest.

**Appium Desktop安装:**官网下载安装: <https://github.com/appium/appium-desktop/releases>

主要功能:UI分析,录制用例,元素查找测试,Attach已有的session ,云测试

**配置待测应用以启动appium:**以Android Studio的模拟器的youtube为例,配置如下,进行启动.

{

"platformName": "android",

"deviceName": "bluestack",

"appPackage": "com.google.android.youtube",

"appActivity": "com.google.android.apps.youtube.app.WatchWhileActivity",

"autoGrantPermissions": true

}

**基础demo:**

配置要打开的app,连接模拟器,进行控制,进行UI分析,点击顶部的**选择元素**进行元素分析,录制简单测试用例且保存.

点击开始录制,选择代码方式,展开样式代码进行操作,操作完成后,保存代码.

**录制用例并执行:**使用appium-desktop录制用例,安装Python依赖pip install Appium-Python-Client,增加隐式等待增强稳定性,重新运行.

**增加隐式等待:**driver.implicitly\_wait(10)

**隐式等待:**服务端(appium)会在特定超时时间内重试多次寻找控件. **显示等待:**在客服端(用例端)根据更灵活的条件循环等待条件满足.

**在VSCode运行:**$sppium后,直接在IDE运行.

**测试步骤三要素:定位,交互,断言.**

**常用定位手段: id, Accessibility ID, XPath(XML路径语言,用于XML中的节点定位)**

**App DOM:**关键Attribute的clickble,content-desc,resource-id,text,bounds,其中android与ios的DOM属性与节点结构类似,名字和属性的命名不同.

**元素定位符与复用:**findElementByxxx,findElement(by,value),findElement主要用于 Page Object模式.

**常用自动化动作支持:** click,sendkys,swiper,touch action

**手势操作:** Press release longPress, moveTo,tapwait,perform

**使用测试框架管理测试用例:**Java的Unit4,Junit5,TestNG. Python的unittest和pytest

**设备交互API:**模拟电话,短信,横竖屏切换,app处理,键盘,粘贴板,录屏.

**monkey测试:**

**monkey测试：**测试软件的稳定性，健壮性的方法，一般可以通过测试过程中打印的日志来发现问题。Monkey是通过命令行来对APP进行测试的工具，允许在模拟器里或真机上。向系统发送伪随机用户时间流实现对应用程序进行压力测试；

**环境搭建：**win10需按照ADB工具, <https://developer.android.com/studio/run/win-usb>,然后再配置变量

https://blog.csdn.net/u013250071/article/details/78416274

**实践：**在手机开发者选项中，勾上USB调试。使用adb命令查看已连接设备：$adb devices

关闭adb的后台进程: adb kill-server 让手机脱离USB线的TCP连接方式: adb tcpip

开启tcp连接方式连接手机: adb connect 收集日志数据,用于后续的分析,比如耗电量: adb bugreport

**发送压测命令：**对随机应用执行100条monkey命令：adb shell monkey 1000

对特定应用进行monkey测试：adb shell monkey -P XXX.apk -v 测试次数

**APP信息:**获取当前界面信息: adb shell dumpsys activity top

获取任务列表: adb shell dumpsys activity activities

获得内存信息: adb shell dumpsys meminfo com.android.settings

获取cpu信息: adb shell dumpsys cpuinfo

获取特定包基本信息: adb shell dumpsys package xx

获取当前activity: adb shell dumpsys activity top

**APP入口:**adb logcat| findsdr -i displayed 或 appt dump badging xx.apk | grep lanunchable-activity 或

apkanalyzer 最新版本的sdk才有

**启动应用:** adb shell am start -n xxx -S 日志查询: adb logcat

当前应用查询 adb logcat | findstr Displayed

**ADB SHELL:**本身Linux的shell,可以调用Android内置命令.

adb shell dumpsys adb shell pm adb shell am adb shell ps adb shell monkey

清理应用缓存:adb shell pm clear 包名

显示常用uiautomator命令: adb shell uiautomator

显示常用输入命令: adb shell input