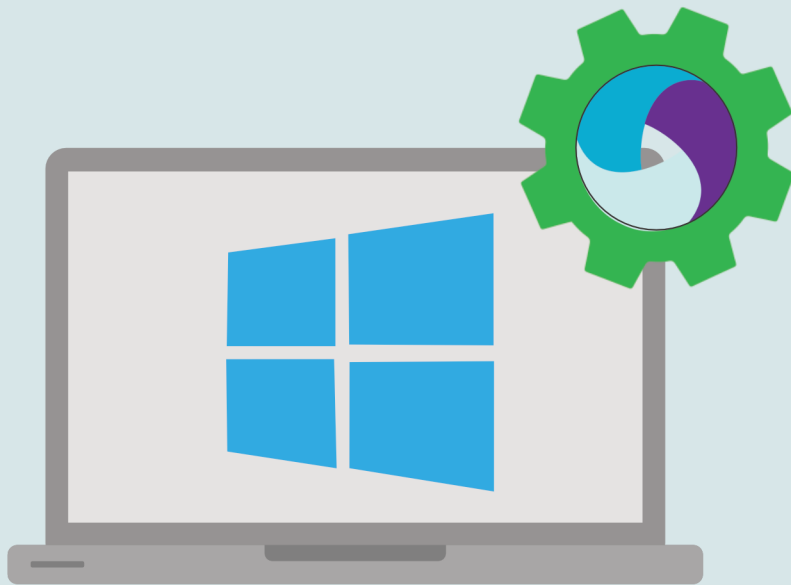


Practical Desktop App Test Automation with Appium



Zhimin Zhan, Courtney Zhan

Practical Desktop App Test Automation with Appium

Test Windows desktop apps wisely with Appium

Zhimin Zhan and Courtney Zhan

This book is for sale at

<http://leanpub.com/practical-desktop-app-test-automation-with-appium>

This version was published on 2023-07-24



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2019 - 2023 Zhimin Zhan and Courtney Zhan

Contents

Preface	i
Who should read this book	ii
How to read this book	iii
Send me feedback	iii
1. Introduction	1
1.1 Test automation benefits	1
1.2 Test Automation success factors	2
1.3 Why traditional commercial testing tools all failed?	2
1.4 Choose a right automation framework	4
1.5 Appium + WinAppDriver is the solution, period	5
1.6 Synergy of web app and native app testing	6
1.7 Desktop App Automation is harder than Web's	6
1.8 Challenges with Appium + WinAppDriver	7
1.9 Next action	8
2. First Appium Automation Script	9
2.1 Appium's Client-Server Architecture	9
2.2 Set up Appium Server v1 with WinAppDriver	9
Install and start Appium server	10
WinAppDriver	12
Enable Developer mode on Windows 10	12
2.3 Set up Appium Server v2 with WinAppDriver	13
Install and start Appium server	13
WinAppDriver	14
2.4 Set up Appium Client	15
Ruby	15
Ruby Library for Appium v1 - appium_lib	16
Ruby Library for Appium v2 - appium_lib	17

CONTENTS

2.5	First Appium Desktop App Test	17
	Open 'Calculator' App for Appium v1	17
	Open 'Calculator' App for Appium v2	18
	Drive controls on the app	19
2.6	Control Inspector	20
2.7	Test Tools	20
	TestWise IDE	20
	Visual Studio Code	21
2.8	Review	22
3.	Test Syntax Framework	23
3.1	RSpec Framework	23
3.2	Transform automation scripts to test scripts	27
	Put it into a structure	27
	Use RSpec hooks	28
	Add Assertions	30
	Add one more test case	30
3.3	Run RSpec tests	30
	Run RSpec test script file from the command line	30
	Run a specific test case from the command line	31
	Run RSpec tests in TestWise	31
	Run RSpec tests in Visual Studio Code	32
3.4	Alternative test syntax frameworks	33
4.	Find App Window	34
4.1	A closer look at the script	34
4.2	Launch UWP App by AppId	35
	How to find a UWP's App ID?	35
4.3	Launch Classic App by executable	36
4.4	Launch app with arguments	37
4.5	Find existing App Window	37
5.	Appium Locators	39
5.1	Tools to identify UI elements	39
	Appium Inspector	39
	WinAppDriver UI Recorder	39
	Accessibility Insights	39
	Inspect.exe	39
	Page Source	39

CONTENTS

5.2	Appium's six locators	40
	Accessibility Id	40
	Name	40
	XPath	40
	Tips on simplifying XPath	40
	Class Name	40
	ID	40
	Tag Name	41
5.3	Chain find_element to find child elements	41
5.4	Guide on selecting locator?	41
5.5	Advice on performance	41
	Which locator is faster?	41
	Limit the scope	41
5.6	Find multiple elements	41
6.	Getting started	42
6.1	An End-To-End Notepad scenario	42
	Start Notepad and type some text	42
	Find the main window	42
	Send Keys to the main window	42
	Send keys to an editor control	42
	Click a menu item	42
	Drive popup window	43
	Close the app	43
6.2	Get Calculator App's Information	43
	Get Window's title	43
	Get Appium server info	43
	Get an element's location on the page or screen	43
	Get an element's size in pixels	43
	Get an element's dimensions and coordinates	44
	Get an element's attribute	44
7.	Keyboard and Mouse	45
7.1	Keyboard	45
	Type non-visible characters	45
	Clear	45
	Type special keys	45
	Select menu item by key combinations	45

CONTENTS

	Select menu item by keyboard shortcut	45
7.2	Mouse	46
	Click a menu	46
	Click a button on the toolbar	46
	Double Click	46
	Context Click	46
7.3	Advanced User Interactions	46
	Double Click	46
	Move window	47
	Context click	47
	Drag and drop	47
	Click by offset	47
8.	Assertion	48
8.1	Assert Window title	48
8.2	Assert Text in an Edit control	48
8.3	Assert a control's attributes	48
8.4	Assert text present	48
8.5	Assert CheckBox is checked	48
8.6	Assert not	49
8.7	Assert RadioButton is checked	49
8.8	Assert Disabled	49
8.9	Assert ComboBox option	49
8.10	Check equal of two elements	49
9.	Review	50
9.1	Syntax errors	50
	How to avoid syntax errors?	50
9.2	Set up source control	50
	Git Installation	50
	Set up Git for local working folder	50
	Set up Git for a shared folder on a network drive	51
	Frequently used Git commands after set up	51
9.3	GUI/Object map	51
9.4	Custom libraries	51
9.5	Debugging	51
9.6	What is the best learning method?	51
10.	Test Maintenance	52

CONTENTS

10.1	Linear test steps are hard to maintain	52
10.2	Maintainable automated test design	52
	Intuitive to read	52
	Reusable function	52
	Page Object Model	52
	When to use Reusable Functions or Page Objects?	53
10.3	Don't Repeat Yourself	53
	DRY with Reusable Functions	53
	DRY with Page Objects	53
10.4	Maintain with ease	53
10.5	Wrap Up	53
11.	Improve Efficiency	54
11.1	Simple project structure	54
11.2	Test execution	54
	Run test cases in a test script file (F10)	54
	Run individual test case (Shift+F10)	54
11.3	Quick navigation	54
	Go to Test Script File (Ctrl+T)	55
	Go to Test Case (Ctrl+Shift+T)	55
11.4	Fast Editing with Snippets	55
11.5	Script library	55
11.6	Test refactoring	55
11.7	Debug test scripts	55
	Keep the app window open	55
	Attach test execution to an existing window	56
11.8	Wrap up	56
12.	Functional Test Refactoring	57
12.1	Functional test refactoring	57
	Functional test refactoring goals	57
12.2	Tool support	57
12.3	Case study	57
	Extract Function	57
	Move to Helper	58
	Move	58
	Extract to Page Function	58
	Introduce Page Object	58

CONTENTS

	Rename	58
12.4	Wrap up	58
13.	Test Data	59
13.1	Test data needs to be reusable	59
13.2	Generate Test Data on the fly	59
	Get date dynamically	59
	Intuitive date utility by ActiveSupport	59
	Get a random boolean value	59
	Generate a number	60
	Get a random string at a fixed length	60
	Get a random string in a collection	60
	Generate random person names, emails, addresses	60
	Generate a test file at fixed sizes	60
13.3	Retrieve data from Database	60
13.4	Reset database	60
14.	Case Study: Test TestWise	61
14.1	Launch App cleanly	61
14.2	App Version	61
14.3	Custom execution with Environment Variables	61
14.4	Common UI elements	61
	ToolBar	61
	Checkbox	62
	Tab	62
	HyperLink	62
	Text Edit Controls	62
	Button	62
14.5	Launch App with Argument	62
14.6	Reusable Test Data	62
14.7	Open a specific test file	63
14.8	Test Automation Support in App	63
14.9	Context Click	63
14.10	Drag and drop	63
14.11	Verify a file moved	63
14.12	Wrap up	63
15.	Continuous Testing	64
15.1	CT Overview	64

CONTENTS

15.2	Prerequisite	64
	Test Scripts are source controlled in Git	64
	The CT server is up running	64
	Appium set up on build machines	64
	Build Agents on build machines are up running	65
15.3	Continuous Testing Steps	65
	Trigger a build on the server via web interface	65
	Build Agent preparation	65
	Parallel Test Execution with Build Agents	65
15.4	Parallel Testing Lab	65
15.5	Ongoing maintenance	65
15.6	Test Stats or Reports	66
16.	Appium in other languages	67
16.1	Appium with Python	67
	Install Appium Python Client	67
	First Python Appium Script	67
	First Python Appium Test	67
16.2	Appium with C#	67
	Install Appium C# Client Library	68
	First C# Appium Script	68
	First C# Appium Test	68
16.3	Wrap up	68
Resources	69
	Books	69
	Web Sites	69
	Tools	69
References	70

Preface

Most business software applications fall in the following three categories:

- **Desktop App**

installed and runs on a personal or work computer, such as Microsoft Word on Windows and KeyNote on macOS. Also known as native apps.

- **Web App**

runs in a web browser and requires Internet (or local network for internal-use app).

- **Mobile App**

runs a mobile device such as smart phone or tablet. Also known as native apps.

This book focuses on test automation for the first category: **Desktop App**, on Windows platform.

Since the Internet revolution began around 1996, there has been a significant shift in software development towards web applications. Over my extensive 20+ years as a programmer, I have primarily specialized in web application development, acknowledging the growing demand for online solutions.

However, it's important to note that desktop applications haven't disappeared entirely. Despite my inclination towards web development, desktop apps still hold relevance in certain contexts. It's true that not all software functions optimally within web browsers. For instance, mobile apps share similarities with desktop apps, albeit tailored for specific mobile operating systems, providing a more tailored user experience.

Throughout my test automation experience, I have predominantly focused on web applications. However, I want to emphasize that I do not intend to overlook automating desktop apps. Automating testing for desktop apps presents greater challenges compared to web apps, mainly due to the absence of a standardized framework like HTML and JavaScript in the web app domain.

Having ventured into automating desktop apps with the aid of a small wrapper library named RFormSpec, built on AutoIT3, I managed to achieve a certain level of success. Nonetheless,

I encountered issues with the execution reliability and maintainability of the test scripts, which did not meet my expectations.

In my search for a standardized solution akin to Selenium WebDriver for web applications, I came across a promising answer in 2019: **Appium + WinAppDriver**. This powerful combination, recommended by Microsoft, proved to be the ideal solution for automating desktop applications, offering the reliability and compatibility I was seeking.

“Coded UI Test for automated UI-driven functional testing is deprecated. Visual Studio 2019 is the last version where Coded UI Test will be fully available. We recommend using Selenium for testing web apps and Appium with WinAppDriver for testing desktop and UWP apps.” - docs.microsoft.com^a

^a<https://docs.microsoft.com/en-us/visualstudio/test/use-ui-automation-to-test-your-code?view=vs-2019>

Appium is an open-source automation framework for iOS, Android and Windows apps. Appium supports WebDriver, the same protocol also used in Selenium for automating web apps. Some might have heard that Appium dominates mobile testing, not for Desktop App testing. The reason is that WinAppDriver v1.0 was only released by Microsoft in October 2017. From my experience, it usually took Microsoft a few releases to achieve the desired level of maturity for its tools.

I put Appium + WinAppDriver in real use (in September 2019), testing my own native app: TestWise. As of 2021-11-02, TestWise’s automated regression suite has 304 Appium + WinAppDriver tests. 63,948 test executions over 834 days. By running this regression suite in the BuildWise Continuous Testing server, I release a new TestWise version regularly on a green build, passing all tests.

Who should read this book

This book is for testers or software engineers who are writing (or want to learn) automated tests in Appium to verify Windows native apps. In order to get the most of this book, basic Ruby coding skill is required.

How to read this book

I recommend readers to read through Chapters 2–14 in order, only skip Chapter 11 if you have decided on the testing editor or IDE. You may skip Chapter 1 to start by writing a test first and then come back to Chapter 1.

Appium and Selenium share a lot in common underneath the obvious differences, such as test automation characteristics and the WebDriver standard. Several chapters (6–10, 12) in “Practical Web Test Automation” are mostly applicable to Appium as well. I expect many of this book’s readers have read “Practical Web Test Automation”, so we won’t do copy-n-paste the content here. Instead, we will write these topics in a concise form. My intention is to keep this book independent.

Some chapters contain hands-on exercises (with step by step guides). Typically it will take about 10–30 minutes to complete an exercise. The main point is: to master test automation, you have to do it. Readers can choose to follow the exercises while or after reading a chapter.

By the way, the framework and all tools listed in this book, you can download and use freely.

Send me feedback

I would appreciate your comments, suggestions, reports on errors in the book and the test scripts. You may submit your feedback on the book’s site.

Functional testing via User Interface is practical and light on theory, so is this book. I hope you find this book useful.

Zhimin Zhan

Brisbane, Australia

1. Introduction

Test automation for Desktop App has been around for a long time, with a bad reputation. It seemed that large software companies all had a similar experience: buy an expensive test automation tool license, play a bit and never use it again. Back in 1999, I had a go with WinRunner (from Mercury Interactive, which was later acquired by HP) when I found out no one was using this expensive software. A tester told me it was hard to create, even more effort to maintain the recorded test scripts. The only memory I have now is that I used WinRunner to fill my timesheets.

1.1 Test automation benefits

The benefits of test automation are plenty. Below are five common ones:

- **Reliable.**

Tests perform the same operations precisely each time they are run, therefore eliminating human errors.

- **Fast.**

Test execution is faster than done manually.

- **Repeatable.**

Once tests are created, they can be run repeatedly with little effort.

- **Cost-Saving.**

Test execution can be scheduled to run at lunchtime or after working hours.

- **Regression Testing.**

“The intent of regression testing is to ensure that a change, such as a bug fix, did not introduce new faults” [Myers, Glenford 04]. Comprehensive manual regression testing is almost impossible to conduct for two reasons: the time required and human errors. As Steve McConnell pointed out, “The only practical way to manage regression testing is to automate it.” [McConnell]

1.2 Test Automation success factors

In a software team with successful test automation implemented, the team members

- runs all UI tests frequently (multiple times a day) as regression testing
- trusts the testing process, if green (all tests pass), they release to production.
- is comfortable on maintaining the test scripts along with frequent changes to the app

Few software projects can achieve that. This is not to belittle IT engineers/managers, rather a failure of our IT education system.

FACT: few software engineers (in Test) received proper education or training in test automation, which is unfortunate. Considering this: 30–40% of staff in a typical software team are testers. Programmers and business analysts spend a lot of time performing functional testing as well. However, to my knowledge, dedicated functional testing courses are rarely offered at universities. Therefore, it is not surprising that many wrong decisions on test automation are made due to a lack of knowledge.

1.3 Why traditional commercial testing tools all failed?

Traditional test automation tools, such as QTP, have existed over two decades, with poor history records. While some tool vendors are still around, the trend of functional test automation is clearly moving towards free and open-source automation frameworks, such as WebDriver.

The reasons for the failures of traditional testing tools are:

- **Proprietary test syntax**

There are huge financial benefits for software vendors to lock clients in their proprietary test syntax so that they can charge clients a big price. Understandably, they don't like standards. As a result, the poor customers will suffer these:

- The script syntax may be unintuitive

Commonly, the syntax was decided by one or two engineers who might have left the company many years ago; In other words, it has been in patching mode.

- Steep learning curve

Being proprietary, the test engineers find it hard to leverage their existing skills. For example, if the test syntax is in a plain-text scripting language, I could use a regular expression to extract dynamic data out. However, with those tools, I had to learn their way.

- Lack of documentation, resources and examples

The vendors had no incentive to provide up-to-date documentation and examples. They want customers to pay support.

- Poor technical support

In theory, after paying the support package, customers shall get good technical support. However, in reality, it never is the case, at least from my past 20+ years of experience. The reason: the support staff are often just out-sourced. The people, who answer the customers' calls and emails, don't have access to the real engineers anyway. Technically, a vendor rarely made a bug fix or enhancement based on a customer's request, a common answer is 'this would be addressed in the next release'.

- **Record-n-Playback**

Record-n-Playback test automation tools gave test automation bad impressions: expensive, hard-to-learn, and above all, does not work. As Lisa Crispin and Janet Gregory put out in their classic book 'Agile Testing': "Record/Playback scripts are notoriously costly from a maintenance perspective." [AT09].

However, in reality, all test automation with purely-recorded test scripts failed. The reason is simple, it is hard to maintain and software changes. It took me years to figure out that most of them never reached the test script maintenance stage, the test automation attempt has failed long before realising the maintenance effort.

- **Discouraging team collaboration**

One of the most frustrating things, when I worked with these commercial tools, is that I couldn't run the scripts without its tool. Let's say one company paid one license for Tester Toby, who is our automation guy. A programmer wants to run (*not develop or edit*) an automation script to verify a defect but he cannot because it requires a runtime license!

- **Expensive**

Test Automation software, I think, has to be one of the most expensive by category. I clearly remembered one sponsor speech at ANZTB 2010 conference (the first conference I attended as a speaker): after demonstrating the software, a company sales representative said "It only costs A\$25,000 a license". A few years back, I was

requested to review IBM Rational Functional Tester (RFT), I wrote the same tests in RFT tests, Selenium WebDriver using RFT as Eclipse IDE and Watir. Watir and Selenium WebDriver excelled in the RFT in every aspect, and these two frameworks are free (and featured in the Agile Testing book which was popular at that time). RFT's price was about A\$11,000 a license.

I am NOT saying the test tools shall be free (as costing \$0), far from that. As a matter of fact, I prefer paying a reasonable price for it, this way, I might get continuous support and updates.

Free software means freedom

I attended one presentation of Richard Stallman, the founder of the Free Software Foundation. Richard said: “*Free, unfortunately, has two meanings in English: free as in freedom and free as in free beer*”. (Official explanation on GNU's website^a: “***‘free software’ is a matter of liberty, not price***”).

^a<https://www.gnu.org/philosophy/free-sw.en.html>

Why expensive GUI testing software is bad? It is now easier to explain with the popular new term “DevOps”. In a true DevOps team, Continuous Testing is a key process that every team member is responsible for quality control, continuously. In other words, every one is involved. Managers, please take this consideration when choosing a commercial testing tool.

1.4 Choose a right automation framework

I never believed those commercial testing tools with proprietary script syntax would work (for both web and native apps), so I did not waste my time learning/using them. The history proved me correct.

To make a test automation solution work, in my opinion, a right automation framework shall satisfy the following three criteria:

1. **Open test script framework, ideally a standard**

Test framework needs to be open-source, ideally a standard.

(I have been using Selenium WebDriver for testing web apps since 2011)

2. Support by underlying technology vendors (not tool vendors)

As a testing framework, obviously, itself must be very reliable. For testing Windows native apps, Microsoft's support is ideal. WinAppDriver (for Appium) is developed by Microsoft.

(Selenium WebDriver is supported by all major browser vendors: Google, Microsoft, Mozilla and Apple. These companies provide the WebDriver-compliant driver, such as ChromeDriver)

3. Flexible scripting with programming language

Programming is the only way to make test scripts flexible enough to cope with frequent application changes. The so-called 'scriptless automation' claimed by some tool vendors has been a complete failure and a joke.

- The success of Selenium WebDriver

Selenium WebDriver was released in 2011 and has dominated web testing since. Different from the proprietary syntax used by expensive commercial test automation tools, Selenium comes in 5 different programming language bindings: Ruby, Python, Java, JavaScript and C#.

- The role 'software tester' is being replaced by SET (Software Engineer in Test)

In top IT companies, the job title "tester" has been renamed to SET or SDET (software development engineer in test). This means testing in those companies requires programming skills.



Only after you decided the framework, then choose a tool to achieve better productivity. Commercial testing tools don't want you to know this, that's why they mix the tool and framework together.

1.5 Appium + WinAppDriver is the solution, period

In the preface, I quoted that Microsoft deprecated its own Coded UI Testing tool and recommended using "Appium with WinAppDriver for testing desktop and UWP apps." [This Microsoft blog¹](#) explained why and clearly stated: "*We recommend using Selenium for testing web-applications and Appium with WinAppDriver for testing desktop (WPF, WinForms, Win32) and UWP apps*".

¹<https://devblogs.microsoft.com/devops/changes-to-coded-ui-test-in-visual-studio-2019/>

While some tool vendors might still go around doing sales pitches (much less now in recent years), I think, it is a matter of time that those tools will die just like Microsoft's own Coded UI Test. The fact is: no company knows Windows native apps better than Microsoft.

1.6 Synergy of web app and native app testing

Astute readers might pick up some similarities between “Selenium WebDriver” and “Appium WinAppDriver” (and “Appium Android Driver” for mobile testing). Yes, they share the same gene, I call it “%ium” + “%Driver”.

- **Script Tier** (top)

Appium for native apps; Selenium for web apps. This defines the script syntax.

- **Driver Tier** (below)

WinAppDriver for native apps; WebDriver (e.g. ChromeDriver) for browsers. This provides the capability to drive the application.

Prior to Appium/Selenium, testing frameworks and tools for testing desktop, web and mobile apps are different. As we know, a modern application often offers two forms: Web and Mobile. There is clear demand to test the application on two platforms using the same technology. Now with “%ium” + “%Driver”, we can.

Katie Coons, a software engineer at Facebook said this at F8 2015 Conference: *“For all of the end-2-end tests at Facebook, we use WebDriver. WebDriver is an open-source JSON wire protocol I encourage you all to check it out if you haven't already. One of the great advantages of WebDriver is that it has got applications across many different platforms. so if I'm writing an end-to-end test for Android, for iOS, for the web, the API that I'm going to use look just the same. This is great, only keep our developers efficient but also make them really flexible and working across platforms”.*

1.7 Desktop App Automation is harder than Web's

Web technologies, such as HTML and CSS, are defined by W3C. In other words, all web pages are based on the same standard. Desktop apps are all different, therefore, automated testing desktop apps will be more challenging than web apps.

Compared to websites, automated testing desktop apps are more challenging:

- **Inspecting to locate a control is harder**

Control Inspection is a built-in browser feature. For desktop apps, we need to use a third-party tool, which is quite limited compared to web browsers.

- **App deployment**

When testing a website, an automated tester does not need to worry about the app's deployment. To the tester, he is testing something inside the Chrome app. When testing a new build of a desktop app, the first step is to deploy it using automation scripts.

- **Test Execution is slow**

Generally speaking, driving a desktop app, as compared to a website, is a lot slower.

- **The app might crash**

Chrome browser is very stable, web test automation testers rarely need to worry about what to do if Chrome crashes. But, unfortunately, to test a desktop app, you need.

1.8 Challenges with Appium + WinAppDriver

Future wise, the demand for automated testing Windows native apps was not high, as new development often towards Web + Mobile. However, I am sure many of you are still using native apps every day, so in my opinion, it is worth learning Appium + WinAppDriver.

There are challenges to introduce Appium + WinAppDriver to a software project:

- **Still relatively immature**

The latest version² is v2.2.2 (2022-12-03), since its first release in 2017.

- **Lack of tools**

With free framework, testing tools vendors have less interest in developing a tool for it (*they cannot ask big prices*).

I extended TestWise to support Appium + WinAppDriver, not aware of others.

- **Hard to get support (*false assumption*)**

Previously, a sale of test automation software often includes a 3-year support package (quite expensive as well). Usually, the support was useless. However, by purchasing the support, the manager felt 'safe'. IT managers with this kind of mindset won't choose 100% free Appium + WinAppDriver.

²<https://github.com/appium/appium-windows-driver/releases>

Of course, it is wrong. For test automation, it is better to engage an external test automation coach than paying for a ‘promised support’, which often is a frustrating conversation with someone at outsourced call center in India.

1.9 Next action

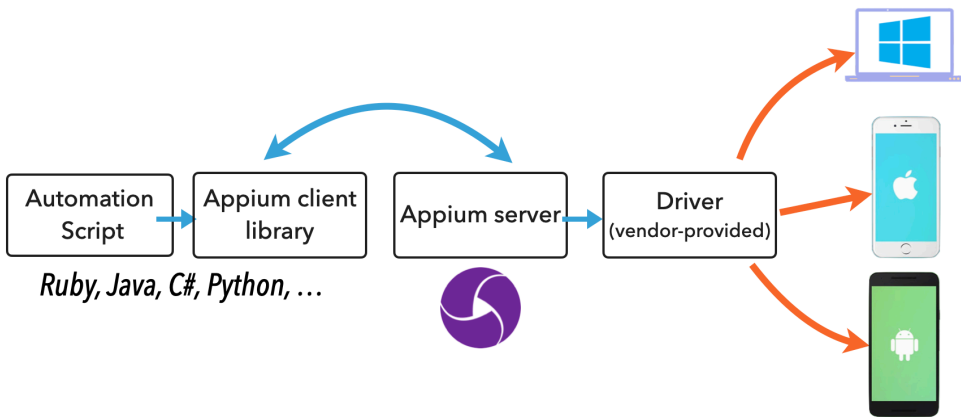
Enough theory for now. Let’s roll up our sleeves and write some automated scripts.

2. First Appium Automation Script

Let's write an Appium automation script to drive a Windows native app. We need to set up Appium on your machine first.

2.1 Appium's Client-Server Architecture

Appium at its heart is a server written in Node.js. The server works using a client-server architecture, a client connects to the server to available services hosted on the server. Any communication between the client and server is in the form of requests and responses.



First of all, the Appium server needs to be up running. Appium scripts, in a programming language such as Ruby, invoke an Appium client library sends requests regarding automation to the Appium server. The server then send translated commands to the driver to automate the application.

2.2 Set up Appium Server v1 with WinAppDriver

Appium v2 is released in July 2023. I have verified it with my TestWise regression suite, with 300+ Appium tests, and found some of features, e.g. mouse right-clicks, are not yet

implemented in WinAppDriver (not Appium's fault). Therefore, I recommend start with Appium v1 first. Appium syntax is relatively stable, even with this major version update, the core remains the same. You can find the book source codes for both v1 and v2.

Windows Application Driver (WinAppDriver in short) is a software that supports Selenium-like UI Test Automation for Windows Applications, including Universal Windows Platform (UWP), Windows Forms (WinForms) and Classic Windows (Win32) apps on *Windows 10* and Windows Server 2016+ PCs. Here are prerequisites for Appium + WinAppDriver tests:

- Appium Server is up running
- WinAppDriver installed
- Developer mode is enabled

Install and start Appium server

There are two ways to install and start up an Appium server v1.

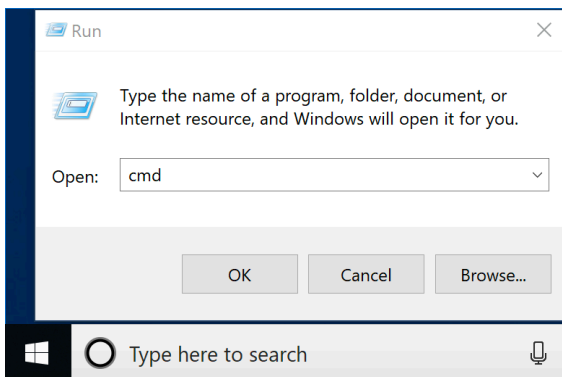
From the command line

Make sure you have Node.js (\geq v18) installed first.



Start a command window on Windows

To start a command window on Windows 10, right click the Windows Logo (at the bottom left) → 'Run' → type 'cmd' and click 'OK'.



Installing Appium is easy, just run the command below in a command window.

```
> npm install -g appium@1.19.1
```

Start up the Appium server.

```
> appium
```

If you see the output like below, the Appium server is up running on your machine.

```
[Appium] Welcome to Appium v1.19.1
```

```
[Appium] Appium REST http interface listener started on 0.0.0.0:4723
```



We prefer this way, as its launching speed is much faster than the Appium Desktop.

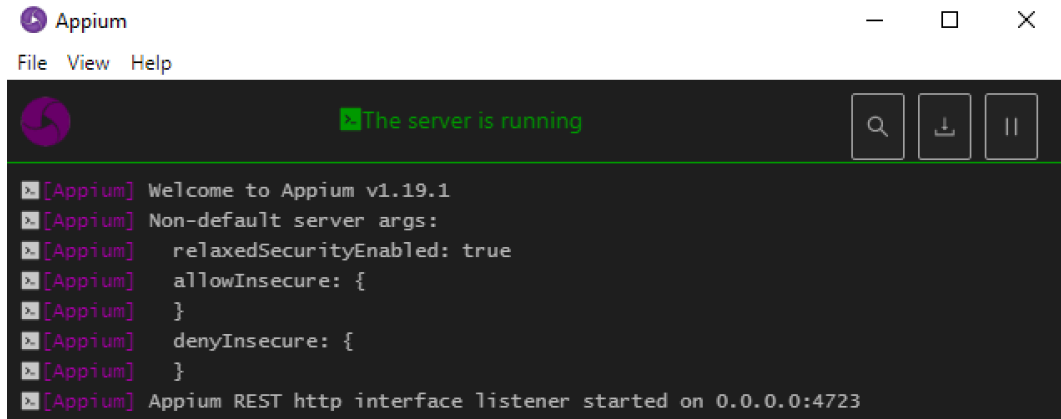
Appium Desktop

Download latest the release from [its Github page](https://github.com/appium/appium-desktop/releases)¹. There are several distributions (for different operating systems), download and run Windows installer (*filename: Appium-windows-VERSION.exe*).



¹<https://github.com/appium/appium-desktop/releases>

Click “Start Server” button.



If you see the above, your Appium Server is up running.

WinAppDriver

Download and install the latest stable [WinAppDriver release](https://github.com/microsoft/WinAppDriver/releases)².

I highlight the stable version here, this is because Appium Server is very strict on the matching AppDriver version. Otherwise, you may get an error like the below: “*UnknownError: An unknown server-side error occurred while processing the command. Original error: Could not verify WinAppDriver install; re-run install*”.

Enable Developer mode on Windows 10

Type “developer features” in “Type here to search” box, and select “Use developer features” to go to the settings page. Enable “Developer mode” there.

²<https://github.com/microsoft/WinAppDriver/releases>

For developers

Use developer features

These settings are intended for development use only.

[Learn more](#)

☐ Microsoft Store apps

Only install apps from the Microsoft Store.

☐ Sideload apps

Install apps from other sources that you trust, like your workplace.

 ☒ Developer mode

Install any signed and trusted app and use advanced development features.

In case you wonder why, here is [the reply from WinAppDriver developer](#)³: “Enabling DeveloperMode requires Administrative access to the machine, and therefore enforces that the user has the right set of permissions to control the machine”.

2.3 Set up Appium Server v2 with WinAppDriver

Install and start Appium server

```
> npm install -g appium
```

Start up the Appium server.

```
> appium
```

If you see the output like below, the Appium server is up running on your machine.

³<https://github.com/Microsoft/WinAppDriver/issues/165>

```
[Appium] Welcome to Appium v2.0.0  
[Appium] Appium REST http interface listener started on 0.0.0.0:4723
```



Appium Desktop has been deprecated in v2, replaced with Appium Inspector.

WinAppDriver

Appium can be used to drive different types of apps via different drivers, including ‘windows’, ‘mac2’, ‘xcuitest’ and ‘espresso’, ..., etc. We will use “Windows App Driver” to drive desktop apps on Windows platform.

```
> appium driver install windows
```

The output shall be like below:

```
√ Installing 'windows' using NPM install spec 'appium-windows-driver'  
i Driver windows@2.2.2 successfully installed  
- automationName: Windows  
- platformNames: ["Windows"]
```

If you start the Appium server again from the command line (if already running, press Ctrl+C to stop it).

```
> appium
```

The output shall be like below.

```
[Appium] Welcome to Appium v2.0.0
[Appium] Attempting to load driver windows...
[debug] [Appium] Requiring driver at C:\Users\ME\node_modules\appium-windows-driver
[Appium] Appium REST http interface listener started on 0.0.0.0:4723
[Appium] Available drivers:
[Appium]   - windows@2.2.2 (automationName 'Windows')
[Appium] No plugins have been installed. Use the "appium plugin" command to install
the one(s) you want to use.
```

2.4 Set up Appium Client

Appium client = Appium scripts + Appium client library. The client library is in one of five official programming languages, including Java, Ruby, JavaScript, Python and C#. Obviously, the test script language matches the client library.

One great feature of Appium (the same for Selenium-WebDriver) is that we could write Appium test scripts in different programming languages, including Java, Ruby, JavaScript, Python and C#. For this book, I will use Ruby, a popular scripting language well suited for writing test scripts. I will show using Appium with other languages such as C# and Python in later chapters, which you will find all quite similar.

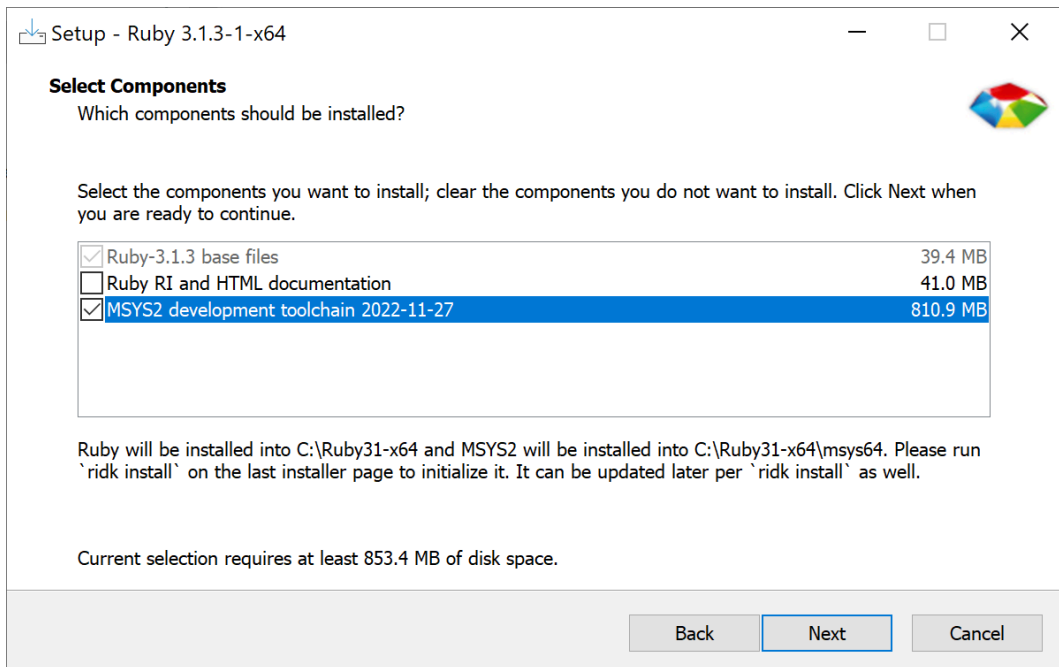
Ruby

Ruby is a free, dynamic, object-oriented, general-purpose scripting language. In my opinion, Ruby is the best language for scripting automated tests (*Disclaimer: I have programmed Java, C# and JavaScript professionally for years*).

Download and install the official [Ruby Installer for Windows](https://rubyinstaller.org/downloads/)⁴, there are several distributions, if you are not sure, get the recommended one, which is Ruby+Devkit 3.1.3 (x64) at the time of writing.

Please make sure the 'MSYS2 development toolchain' is checked in the installation wizard.

⁴<https://rubyinstaller.org/downloads/>



You might find this guide: [“10-Minute Guide to Set up Test Automation using Selenium WebDriver with Ruby⁵”](https://zhiminzhan.medium.com/10-minute-guide-to-set-up-test-automation-using-selenium-webdriver-with-ruby-a2454ac86e95) useful.

Ruby Library for Appium v1 - appium_lib

Ruby libraries are called Gems. Installing gems is easy.

```
> gem install appium_lib --version 11.2.0
```

If you see error messages like below,

⁵<https://zhiminzhan.medium.com/10-minute-guide-to-set-up-test-automation-using-selenium-webdriver-with-ruby-a2454ac86e95>

```
LoadError:
  cannot load such file -- 3.1/rubyeventmachine
```

The most likely reason is 'eventmachine-1.2.7-x64-mingw32' not compatible with the Ruby installation on your OS. Here is a workaround:

```
> gem uninstall eventmachine
> gem install eventmachine --platform ruby
```

Then re-run `gem install appium_lib` command.

Ruby Library for Appium v2 - appium_lib

The `appium_lib` version v12+ works with Appium v2.

```
> gem install appium_lib
```

2.5 First Appium Desktop App Test

We will use the built-in Calculator app in our first example.

Open 'Calculator' App for Appium v1

Open Notepad, type or paste the following text into it.

```
require 'appium_lib'
desired_caps = {
  caps: {
    platformName: 'Windows',
    deviceName:   'SurfacePro',
    app:          "Microsoft.WindowsCalculator_8wekyb3d8bbwe!App"
  } }
driver = Appium::Driver.new(desired_caps, true).start_driver
```

Save the file as “**start-calc.rb**” in a folder (e.g.C:\Users\YOU\learn-appium), please note the file extension is “.rb”.

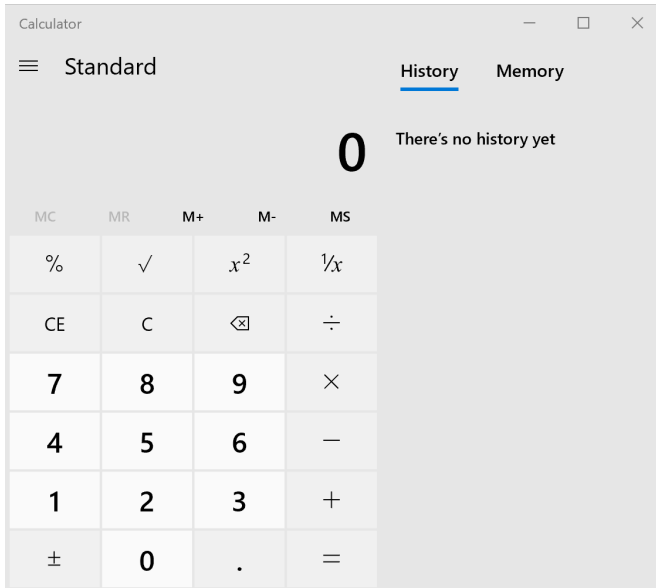
Open a Command window, change to that folder,

```
> cd C:\Users\YOU\learn-appium
```

and run this script.

```
> ruby start-calc.rb
```

You will see Calculator App starts, hooray!



Please note, if you have used Appium v1 with Ruby, the following section was not required before, but seems mandatory now.

```
appium_lib: {  
  server_url: "http://127.0.0.1:4723"  
}
```

Without it, I got this error.

The requested resource could not be found, or a request was received using an HTTP method\ that is not supported by the mapped resource
(Selenium::WebDriver::Error::UnknownCommandError)

Open 'Calculator' App for Appium v2

The opts to start Appium is different.

```
require 'appium_lib'

opts = {
  caps: {
    automationName: "windows",
    platformName: "Windows",
    deviceName: "Dell",
    app: "Microsoft.WindowsCalculator_8wekyb3d8bbwe!App"
  },
  appium_lib: {
    server_url: "http://127.0.0.1:4723"
  }
}

driver = Appium::Driver.new(opts).start_driver
```

Please note, if you have used Appium v1 with Ruby, the following section was not required before, but seems mandatory now.

```
appium_lib: {
  server_url: "http://127.0.0.1:4723"
}
```

Without it, I got this error.

The requested resource could not be found, or a request was received using an HTTP method\ that is not supported by the mapped resource
(Selenium::WebDriver::Error::UnknownCommandError)

Drive controls on the app

You may add the following test steps (to `start-calc.rb`) to drive the Calculator app.

```
#...
driver.find_element(:name, "One").click
driver.find_element(:name, "Plus").click
driver.find_element(:name, "Three").click
driver.find_element(:name, "Equals").click

result = driver.find_element(:accessibility_id, "CalculatorResults").text
puts(result)
```

Run the script again (`ruby start-calc.rb`), this time, you will see the script is ‘using’ the Calculator.

```
Display is 4
```

2.6 Control Inspector

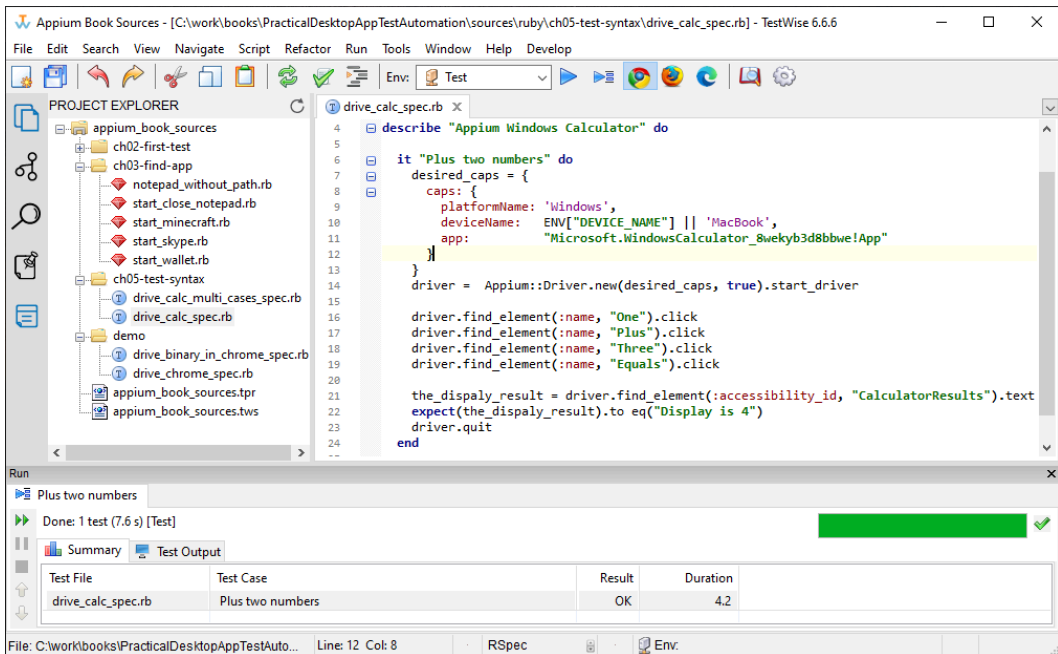
To drive an app, more specifically, the controls in the app, we need to locate them first. Different from web apps, where the browser has a built-in inspector, we’ll need to use a third-party inspecting utility. I will cover that in Chapter 4.

2.7 Test Tools

Appium test scripts are plain text files, which means you may use any text editors such as NotePad to edit them. Of course, NotePad won’t be a good choice. Typically, we use an integrated development environment (IDE) or a programmer editor.

TestWise IDE

TestWise is a functional testing IDE to help testers develop and maintain automated test scripts efficiently (*Disclaimer: I created TestWise*). You can create, edit, run and debug test scripts in TestWise.



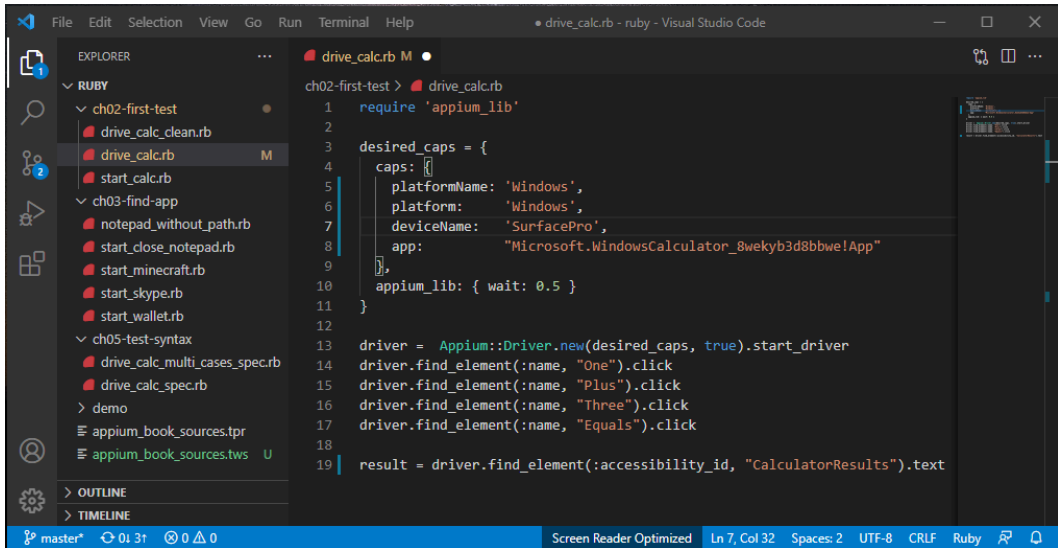
If you have read Zhimin's other books on web test automation, you will know that we use TestWise for Selenium WebDriver tests. Yes, you may use it for Appium test scripts as well, in pretty much the same way.



Check out Chapter 11 for simple instructions to use TestWise efficiently.

Visual Studio Code

Visual Studio Code is a free and popular source-code editor. While VS Code is mainly used by programmers, we can use it for developing test scripts as well.



The screenshot shows the Visual Studio Code editor with a Ruby file named `drive_calc.rb` open. The Explorer sidebar on the left shows a project structure with folders `ch02-first-test` and `ch03-find-app`, and various Ruby files. The main editor area displays the following code:

```
ch02-first-test > drive_calc.rb
1  require 'appium_lib'
2
3  desired_caps = {
4    caps: {
5      platformName: 'Windows',
6      platform: 'Windows',
7      deviceName: 'SurfacePro',
8      app: "Microsoft.WindowsCalculator_8wekyb3d8bbwe!App"
9    },
10   appium_lib: { wait: 0.5 }
11 }
12
13 driver = Appium::Driver.new(desired_caps, true).start_driver
14 driver.find_element(:name, "One").click
15 driver.find_element(:name, "Plus").click
16 driver.find_element(:name, "Three").click
17 driver.find_element(:name, "Equals").click
18
19 result = driver.find_element(:accessibility_id, "CalculatorResults").text
```

The status bar at the bottom indicates the file is on the `master` branch, has 31 changes, and is using the `UTF-8` encoding with `CRLF` line endings. It also shows the file is `Screen Reader Optimized` and the editor is in `Ruby` mode.

2.8 Review

Compared to Appium v1, v2 installation of driver is much easier. Please note, the set-up of Appium (server, scripting language and client library) is only needed to be done once. From now on, we will focus on the test scripts.

3. Test Syntax Framework

Test Automation Framework = Driver framework + Test Syntax framework

Despite Appium being widely known as a test automation framework, strictly speaking, it is the 'Automation' part of a test automation framework. Automation (also known as Driver) frameworks drive the application's UI; test syntax frameworks provide the test structure and assertion mechanism.

In this chapter, I will introduce RSpec, the syntax framework choice of this book.

3.1 RSpec Framework

RSpec is a popular Behaviour Driven Development (BDD) framework in Ruby. According to RubyGems, the download count of RSpec v3.8 alone exceeds 207 million!

More expressive

Compared to xUnit test frameworks, RSpec tests are easier to read. For example, for the JUnit test below:

```
class UserAuthenticationTest {  
    public void testCanLoginWithValidUsernameAndPassword {  
        // ...  
    }  
    public void testAccessDeniedForInvalidPassword() {  
        // ...  
    }  
}
```

Its RSpec version will be like this:

```
describe "User Authentication" do
  it "User can login with valid login and password" do
    # ...
  end

  it "Access denied for invalid password" do
    #...
  end
end
```

Execution Hooks

Execution hooks are similar to `setUp()` and `tearDown()` functions in JUnit. Test steps inside an execution hook are run before or after test cases depending on the nature of the hook. The example below shows the order of execution in RSpec:

```
describe "Execution Order Demo" do

  before(:all) do
    puts "Calling before(:all)"
  end

  before(:each) do
    puts "  Calling before(:each)"
  end

  after(:each) do
    puts "  Calling after(:each)"
  end

  after(:all) do
    puts "Calling after(:all)"
  end

  it "First Test Case" do
    puts "    In First Test Case"
  end

  it "Second Test Case" do
    puts "    In Second Test Case"
  end

end
```

Output

```
Calling before(:all)
  Calling before(:each)
    In First Test Case
  Calling after(:each)
  Calling before(:each)
    In Second Test Case
  Calling after(:each)
Calling after(:all)
```

What is the use of execution hooks? Let's look at the test script below (*please just focus on the structure of test scripts rather than test statement syntax, for now*). There are three login test cases in a single test script file.

```
describe "User Login" do
  include TestHelper # defined functions such as open_app, sign_in, ..., etc

  it "Can login as Registered User" do
    open_app
    sign_in("james", "pass")
    expect(page_text).to include("Welcome James")
    sign_off
    close_app
  end

  it "Can login as Guest" do
    open_app
    sign_in("guest", "changeme")
    expect(page_text).to include("Login OK")
    close_app
  end

  it "Can login as Administrator" do
    open_app
    sign_in("admin", "secret")
    assert_link_present_with_text("Settings")
    sign_off
    close_app
  end
end
```

By utilizing execution hooks, we can refine these test cases to:

```
describe "User Login" do
  include TestHelper

  before(:all) do
    open_app
  end

  after(:each) do
    sign_off
  end

  after(:all) do
    close_app
  end

  it "Can login as Registered User" do
    sign_in("james", "pass")
    expect(page_text).to include("Welcome James")
  end

  it "Can login as Guest" do
    sign_in("guest", "changeme")
    expect(page_text).to include("Login OK")
  end

  it "Can login as Administrator" do
    sign_in("admin", "secret")
    assert_link_present_with_text("Settings")
  end
end
```

By utilizing RSpec's `before(:all)`, `after(:each)` and `after(:all)` hooks, this version is not only more concise, more importantly, every test case is now more focused (distinguished from each other). Using these hooks effectively will make test scripts more readable and easier to maintain. For readers who are new to RSpec, don't worry, I will cover it more in later chapters.

3.2 Transform automation scripts to test scripts

In the previous chapter, we created an Appium automation script that drives the Calculator app.

```
require 'appium_lib'

# please note this for Appium v2.
# for v1, see Chapter 2 or check out the book's source code under v1
opts = {
  caps: {
    automationName: "windows",
    platformName: "Windows",
    deviceName: "Dell",
    app: "Microsoft.WindowsCalculator_8wekyb3d8bbwe!App"
  },
  appium_lib: {
    server_url: "http://127.0.0.1:4723",
    wait: 0.5
  }
}
driver = Appium::Driver.new(opts).start_driver

driver.find_element(:name, "One").click
driver.find_element(:name, "Plus").click
driver.find_element(:name, "Three").click
driver.find_element(:name, "Equals").click

the_display_result = driver.find_element(:accessibility_id, "CalculatorResults").text
puts(the_display_result)
driver.quit
```

Put it into a structure

```

require 'rspec'
require 'appium_lib'

describe "Appium Windows Calculator" do

  it "Plus two numbers" do
    opts = {
      caps: {
        automationName: "windows",
        platformName: "Windows",
        deviceName: "Dell",
        app: "Microsoft.WindowsCalculator_8wekyb3d8bbwe!App"
      },
      appium_lib: {
        server_url: "http://127.0.0.1:4723",
        wait: 0.5
      }
    }

    driver = Appium::Driver.new(opts).start_driver
    driver.find_element(:name, "One").click
    driver.find_element(:name, "Plus").click
    driver.find_element(:name, "Three").click
    driver.find_element(:name, "Equals").click

    the_display_result = driver.find_element(:accessibility_id, "CalculatorResults").text
    puts(the_display_result)
    driver.quit
  end
end

```

- **describe**

Scope of a test suite. A test suite may contain one or more test cases.

- **it**

Scope of a test case.

Use RSpec hooks

The initialization part is the first thing we do, so I put it in `before(:all)`. By the same token, `driver.quit` is in `after(:all)`.


```

before(:all) do
  opts = {
    caps: {
      automationName: "windows",
      platformName: "Windows",
      deviceName: "Dell",
      app: "Microsoft.WindowsCalculator_8wekyb3d8bbwe!App"
    },
    appium_lib: {
      server_url: "http://127.0.0.1:4723"
    }
  }

  @driver = Appium::Driver.new(opts).start_driver
end

after(:all) do
  driver.quit
end

# a convenient function to use driver in test scripts
def driver
  @driver
end

```

I added a convenient method `def driver`. This is to create an alias to the instance variable `@driver`. To put it simply, an instance variable in Ruby can be used anywhere in the script file, not limited to a local scope.

Now the main test case is extracted out, easier to read.

```

it "Plus two numbers" do
  driver.find_element(:name, "One").click
  driver.find_element(:name, "Plus").click
  driver.find_element(:name, "Three").click
  driver.find_element(:name, "Equals").click
  the_display_result = driver.find_element(:accessibility_id, "CalculatorResults").text
  puts the_display_result
end

```

Add Assertions

A test script without assertions (also known as checkpoints) is not a real test. In the example, I replace printing out the calculated result with an assertion.

```
it "Plus two numbers" do
  # ...
  the_display_result = driver.find_element(:accessibility_id, "CalculatorResults").text
  expect(the_display_result).to eq("Display is 4")
end
```

If an assertion in one test case failed, RSpec will mark that test case (and the whole test script file) as failed.

Add one more test case

Quite often, a test suite contains one or more related tests. For example, a user login test script usually contains at least two test cases: “Login OK” and “Login Failed”.

I add one more test into our test script file.

```
it "Minus" do
  driver.find_element(:name, "Six").click
  driver.find_element(:name, "Minus").click
  driver.find_element(:name, "One").click
  driver.find_element(:name, "Equals").click

  result = driver.find_element(:accessibility_id, "CalculatorResults").text
  expect(result.gsub("Display is ", "").to eq("5"))
end
```

3.3 Run RSpec tests

There are several ways to execute RSpec tests.

Run RSpec test script file from the command line

```
rspec calc_4_multi_cases_spec.rb
```

Test Pass

```
..
```

```
Finished in 5.31 seconds (files took 2.75 seconds to load)
2 examples, 0 failures
```

Test Fail

If there a test failures, RSpec shows F in the test output (like JUnit) and detail.

```
F.
```

Failures:

```
1) Appium Windows Calculator Plus
   Failure/Error: expect(result).to eq("Display is 3")

     expected: "Display is 3"
     got: "Display is 4"

     (compared using ==)
# ./calc_4_multi_cases_spec.rb:31:in `block (2 levels) in <top (required)>'
```

```
Finished in 5.71 seconds (files took 3.16 seconds to load)
2 examples, 1 failure
```

Run a specific test case from the command line

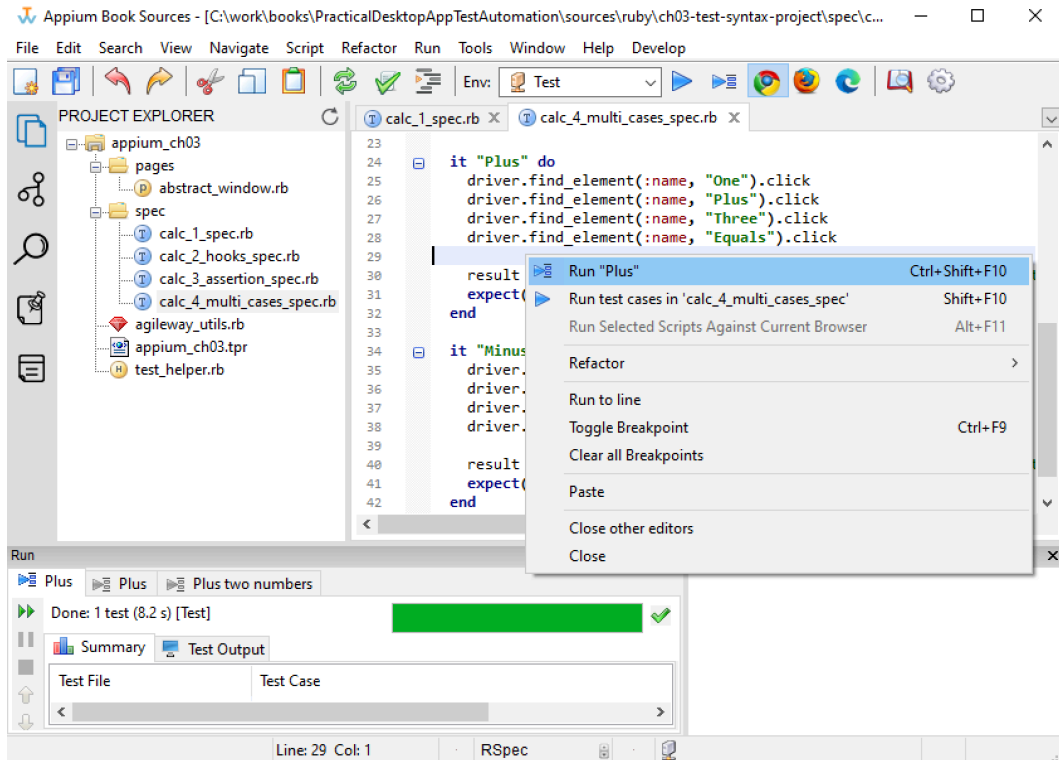
To run a specific test case within a RSpec test script file, simply append a line number in chosen test case range. For example, the command below will run the second test case (based on the line number) only in *calc_4_multi_cases_spec.rb*.

```
rspec calc_4_multi_cases_spec.rb:38
```

Run RSpec tests in TestWise

TestWise has built-in support for RSpec.

Open the test project (e.g. *ch03-test-syntax-project/appium_ch03.tpr*), click a test script file (**_spec.rb*) on the left to open in the editor, right-click a test case (under *it "..."*) and select Run "...".

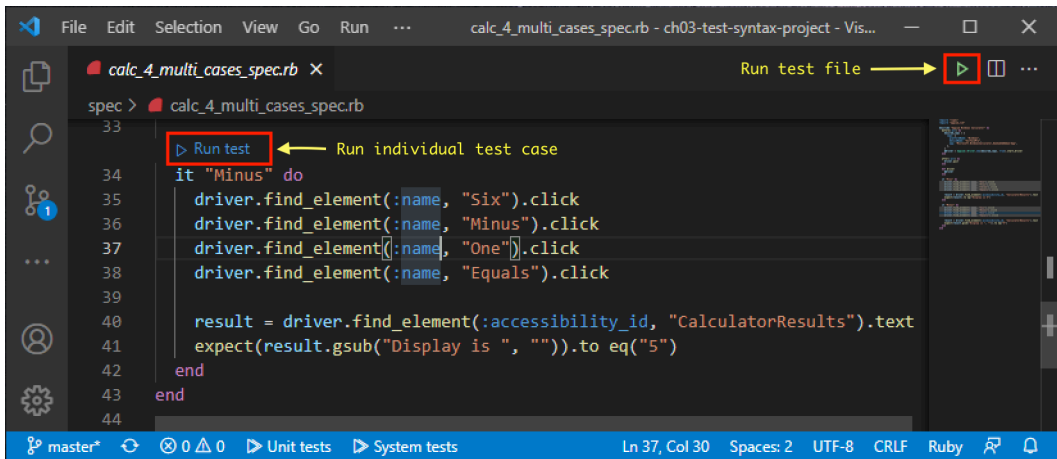


For more on using TestWise, please check out Chapter 11.

Run RSpec tests in Visual Studio Code

You need to install an extension to run RSpec tests in Visual Studio Code. [Ruby Test Runner](https://marketplace.visualstudio.com/items?itemName=MateuszDrewniak.ruby-test-runner)¹ is a good one.

¹<https://marketplace.visualstudio.com/items?itemName=MateuszDrewniak.ruby-test-runner>



3.4 Alternative test syntax frameworks

Appium, as a driver framework, can work with most test syntax frameworks, which fall into one of three categories:

- **JUnit variants** (known as xUnit)

The granddaddy of the test syntax frameworks is JUnit created by Kent Beck and Erich Gamma in 1997. As its name suggests, it is designed for unit testing. With JUnit-style frameworks being widely used, JUnit variants were created for integration and functional tests, such as PyTest for Python.

- **RSpec variants** in other languages

Mocha for JavaScript, Pytest for Python.

- **Gherkin syntax frameworks**, such as Cucumber and SpecFlow.

I don't recommend using Gherkin syntax (also known as Given-Then-When) for test automation, as it significantly increases test maintenance efforts with little benefit.

4. Find App Window

The first challenge is to start (*a new instance*) or find (*existing*) your app via Appium scripts.

4.1 A closer look at the script

1. Define Desired Capabilities

Desired Capabilities are settings in a JSON object, sent by Appium clients to the server to start a new automation session.

Example:

```
require 'appium_lib'
opts = {
  caps: {
    automationName: "windows",
    platformName: "Windows",
    deviceName: "Dell",
    app: "Microsoft.WindowsCalculator_8wekyb3d8bbwe!App"
  },
  appium_lib: {
    server_url: "http://127.0.0.1:4723",
    wait: 0.5
  }
}
```

The important part is the app. The setting for app is either the AppID for UWP apps or the exe path of classic apps. Calculator app for the above example.

2. Start App

The statement starts a new Appium session, which will launch the application (specified in the desired capabilities).

```
driver = Appium::Driver.new(opts).start_driver
```

3. Drive UI elements

Driver the controls like user actions.

```
driver.find_element(:name, "One").click
driver.find_element(:name, "Plus").click
driver.find_element(:name, "Three").click
driver.find_element(:name, "Equals").click
```

4. Get data from one UI element

Get the result from the Calculator. This is a different type of operation from the above, in the context testing, it fetches data from the app for verification.

```
display_result = driver.find_element(:accessibility_id, "CalculatorResults").text
puts(display_result)
```

We will cover verifications later, here I use puts (in Ruby) to print out the calculation result.

5. Close App

There is still one step missing in the above script: the Calculator window remains after the script. As a convention, we close the app in the end, so that it won't cause too many applications running.

```
driver.quit
```

4.2 Launch UWP App by AppId

For a Microsoft Universal Windows Platform application (UWP), use its unique App ID.

```
caps: {
  #...
  app: "Microsoft.WindowsCalculator_8wekyb3d8bbwe!App"
}
```

How to find a UWP's App ID?

Start “Windows PowerShell”, use Get-AppxPackage command to get App details including App ID. For example, the command below returns Calculator App details.

```
> Get-AppxPackage | out-string -stream | select-string Calculator
```

The ‘PackageFamilyName’ value in the output is the App ID.

```
PackageFamilyName : Microsoft.WindowsCalculator_8wekyb3d8bbwe
```

Paste the appID in your script and append “!App” after it. This is important, without that, your app won’t start.

If you are not sure the App name, use the command below to list all installed UWP apps on your machine.

```
Get-AppxPackage | out-string -stream | select-string PackageFamilyName
```



Launch another UWP app with Appium

Write a new script to launch your favourite UWP app such as Minecraft and Skype.

4.3 Launch Classic App by executable

The AppId is only applicable to UWP apps. For Classic Win32 Apps, we need to specify the exe filename of the application. The script below opens NotePad, wait 3 seconds, then close it.

```
require 'appium_lib'

opts = {
  caps: {
    automationName: "windows",
    platformName: "Windows",
    deviceName: "Dell",
    app: "notepad.exe"
  },
  appium_lib: {
    server_url: "http://127.0.0.1:4723"
  }
}

driver = Appium::Driver.new(opts).start_driver
sleep 3
driver.quit
```

If the executable (.exe file) is in the PATH, you may just use the exe filename.


```
app: "notepad.exe"
```

However, I recommend using an absolute path to avoid confusion.

4.4 Launch app with arguments

Quite often, we want to start the application with arguments. For example, the below launches Notepad with an existing text file.

```
opts = {  
  caps: {  
    automationName: "windows",  
    platformName: "Windows",  
    app: "notepad.exe",  
    appArguments: "C:\\\\agileway\\\\TestWise6\\\\ReadMe.txt",  
  },  
  appium_lib: {  
    server_url: "http://127.0.0.1:4723",  
  },  
}
```

This is equivalent to `notepad C:\\agileway\\TestWise6\\ReadMe.txt`.

4.5 Find existing App Window

You can attach an Appium session to an existing Window. This will be very handy for debugging.

Here is how it works: Create a new session without launching a new app instance by setting `app: "Root"` in the desired capabilities.

```
opts = {  
  caps: {  
    automationName: "windows",  
    platformName: "Windows",  
    deviceName: "Dell",  
    app: "Root"  
  },  
  appium_lib: {  
    server_url: "http://127.0.0.1:4723",  
    wait: 0.5  
  }  
}  
  
@driver = Appium::Driver.new(opts).start_driver
```

You can think this is an Appium session for the whole desktop, then find the main window normal way.

The example below is to find the BuildWise Agent window and click the 'Stop' button in it. To make the test script working, install [BuildWise Agent](https://agileway.com.au/buildwise/download)¹ first and then click the 'Start' button to get it started.

```
win_title = "BuildWise Agent - Evaluation"  
main_win = driver.find_element(:name, win_title)  
main_win.find_element(:name, "Stop").click
```

¹<https://agileway.com.au/buildwise/download>

5. Appium Locators

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

5.1 Tools to identify UI elements

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Appium Inspector

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

WinAppDriver UI Recorder

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Accessibility Insights

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Inspect.exe

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Page Source

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

5.2 Appium's six locators

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Accessibility Id

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Name

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

XPath

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Tips on simplifying XPath

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Class Name

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

ID

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Tag Name

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

5.3 Chain find_element to find child elements

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

5.4 Guide on selecting locator?

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

5.5 Advice on performance

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Which locator is faster?

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Limit the scope

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

5.6 Find multiple elements

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

6. Getting started

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

6.1 An End-To-End Notepad scenario

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Start Notepad and type some text

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Find the main window

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Send Keys to the main window

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Send keys to an editor control

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Click a menu item

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Drive popup window

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Close the app

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

6.2 Get Calculator App's Information

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Get Window's title

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Get Appium server info

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Get an element's location on the page or screen

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Get an element's size in pixels

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Get an element's dimensions and coordinates

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Get an element's attribute

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

7. Keyboard and Mouse

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

7.1 Keyboard

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Type non-visible characters

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Clear

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Type special keys

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Select menu item by key combinations

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Select menu item by keyboard shortcut

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

7.2 Mouse

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Click a menu

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Click a button on the toolbar

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Double Click

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Context Click

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

7.3 Advanced User Interactions

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Double Click

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Move window

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Context click

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Drag and drop

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Click by offset

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

8. Assertion

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

8.1 Assert Window title

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

8.2 Assert Text in an Edit control

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

8.3 Assert a control's attributes

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

8.4 Assert text present

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

8.5 Assert CheckBox is checked

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

8.6 Assert not ...

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

8.7 Assert RadioButton is checked

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

8.8 Assert Disabled

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

8.9 Assert ComboBox option

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

8.10 Check equal of two elements

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

9. Review

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

9.1 Syntax errors

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

How to avoid syntax errors?

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

9.2 Set up source control

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Git Installation

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Set up Git for local working folder

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Set up Git for a shared folder on a network drive

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Frequently used Git commands after set up

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

9.3 GUI/Object map

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

9.4 Custom libraries

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

9.5 Debugging

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

9.6 What is the best learning method?

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

10. Test Maintenance

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

10.1 Linear test steps are hard to maintain

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

10.2 Maintainable automated test design

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Intuitive to read

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Reusable function

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Page Object Model

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

When to use Reusable Functions or Page Objects?

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

10.3 Don't Repeat Yourself

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

DRY with Reusable Functions

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Parameterizing functions

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

DRY with Page Objects

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

10.4 Maintain with ease

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

10.5 Wrap Up

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

11. Improve Efficiency

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

11.1 Simple project structure

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

11.2 Test execution

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Run test cases in a test script file (F10)

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Run individual test case (Shift+F10)

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

11.3 Quick navigation

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Go to Test Script File (Ctrl+T)

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Go to Test Case (Ctrl+Shift+T)

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

11.4 Fast Editing with Snippets

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

11.5 Script library

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

11.6 Test refactoring

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

11.7 Debug test scripts

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Keep the app window open

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Attach test execution to an existing window

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

11.8 Wrap up

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

12. Functional Test Refactoring

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

12.1 Functional test refactoring

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Functional test refactoring goals

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

12.2 Tool support

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

12.3 Case study

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Extract Function

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Move to Helper

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Move

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Extract to Page Function

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Introduce Page Object

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Rename

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

12.4 Wrap up

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

13. Test Data

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

13.1 Test data needs to be reusable

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

13.2 Generate Test Data on the fly

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Get date dynamically

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Intuitive date utility by ActiveSupport

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Get a random boolean value

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Generate a number

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Get a random string at a fixed length

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Get a random string in a collection

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Generate random person names, emails, addresses

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Generate a test file at fixed sizes

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

13.3 Retrieve data from Database

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

13.4 Reset database

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

14. Case Study: Test TestWise

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

14.1 Launch App cleanly

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

14.2 App Version

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

14.3 Custom execution with Environment Variables

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

14.4 Common UI elements

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

ToolBar

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Checkbox

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Tab

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

HyperLink

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Text Edit Controls

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Button

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

14.5 Launch App with Argument

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

14.6 Reusable Test Data

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

14.7 Open a specific test file

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

14.8 Test Automation Support in App

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

14.9 Context Click

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

14.10 Drag and drop

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

14.11 Verify a file moved

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

14.12 Wrap up

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

15. Continuous Testing

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

15.1 CT Overview

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

15.2 Prerequisite

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Test Scripts are source controlled in Git

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

The CT server is up running

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Appium set up on build machines

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Build Agents on build machines are up running

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

15.3 Continuous Testing Steps

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Trigger a build on the server via web interface

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Build Agent preparation

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Parallel Test Execution with Build Agents

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

15.4 Parallel Testing Lab

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

15.5 Ongoing maintenance

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

15.6 Test Stats or Reports

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

16. Appium in other languages

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

16.1 Appium with Python

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Install Appium Python Client

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

First Python Appium Script

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

First Python Appium Test

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

16.2 Appium with C#

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Install Appium C# Client Library

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

First C# Appium Script

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

First C# Appium Test

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

16.3 Wrap up

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Resources

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Books

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Web Sites

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

Tools

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.

References

This content is not available in the sample book. The book can be purchased on Leanpub at <http://leanpub.com/practical-desktop-app-test-automation-with-appium>.