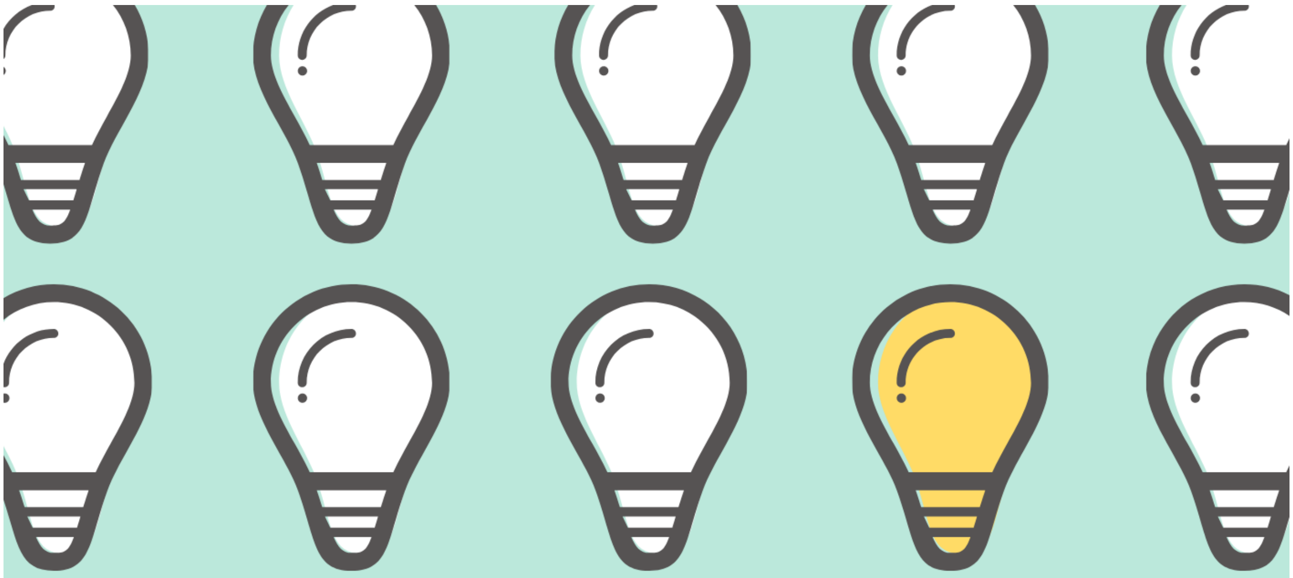
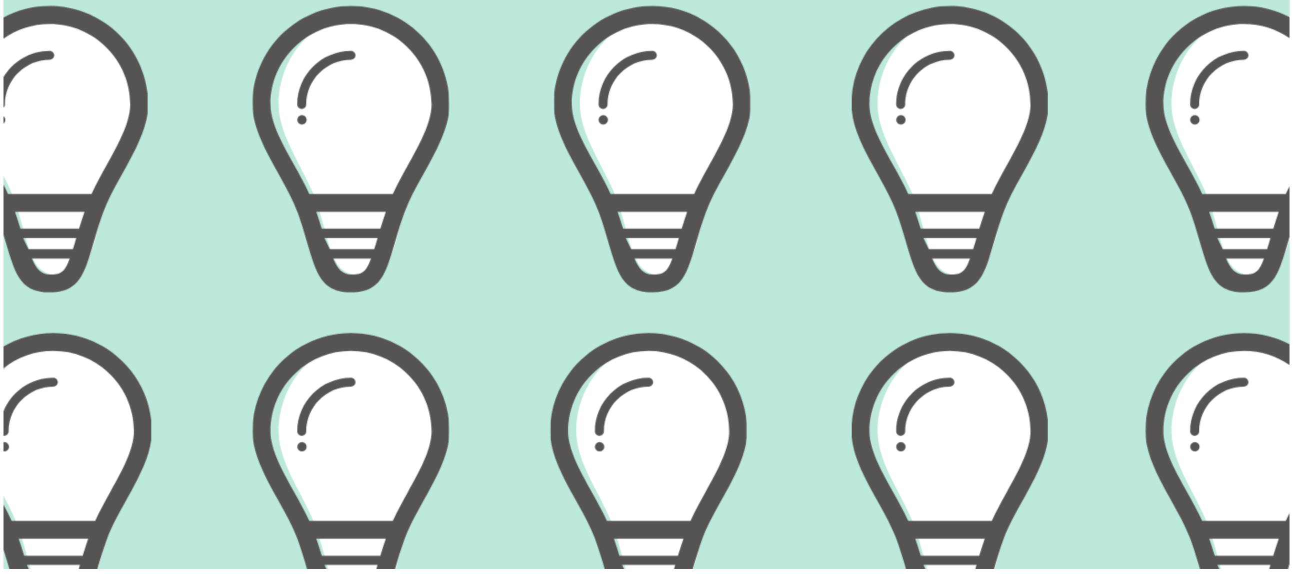


ANTON SMIRNOV

# PRACTICAL TIPS AND TRICKS FOR SELENIUM TEST AUTOMATION

A Testing Journal



# Practical Tips and Tricks for Selenium Test Automation.

**Learn to create a framework for automation testing fundamentals fast.**

This version was published on 2021-02-22

## Contact details:

- antony.s.smirnov@gmail.com

**Related Websites:**

- Practical Tips and Tricks for Selenium Test Automation: https://test-engineer.site/

- Author’s Software Testing Blog: https://test-engineer.site/

## Table of Contents

# Introduction.

Pretty often you can see test automation framework successfully running tests and reporting results but not doing what it’s supposed to do: providing a reliable way for team members to build automated tests, and get reliable results.

This often happens when a test automation framework is built without planning in advance and understanding how it will be used.

At first, the team realizes that they need automated tests. One of the engineers decides to take care of it (or gets assigned) — using the tools they are familiar with; they automate the first bunch of tests.

Since initially, it’s a proof of concept, some things are being implemented via the fastest and most obvious solution, which is not always utilizing the industry’s best practices. Such solutions introduce technical debt. If not addressed early, the impact of technical debt grows once the framework is expanded.

As a result, few iterations later, the team gets a test automation framework that can pretty well-run tests that were in the mind of the author building it. But making a step aside, expanding coverage to additional features, or trying to get other engineers owning tests creation via such framework becomes a challenging task.

Have you ever wondered how to set up a test automation framework? Well, in this book you will learn about everything you’ll need to successfully create such a framework.

We’re going to look at the pros and cons of preconfigured testing environments and those that are created dynamically.

This book is based on more than 5+ years of experience in the field of test automation. During this time, a huge collection of solved questions has accumulated, and the problems and difficulties characteristic of many beginners have become clearly visible. In the course of working in different places, I have repeatedly had to create a framework for testing automation from scratch. It was obvious and reasonable for me to summarize this material in the form of a book that will help novice testers quickly build an automation testing framework on a project and avoid many annoying mistakes.

This book does not aim to fully disclose the entire subject area with all its nuances, so do not take it as a textbook or Handbook — for decades of development testing has accumulated such a volume of data that its formal presentation is not enough, and a dozen books.

Also, reading just this one book is not enough to become a "senior automated testing engineer". Then why do we need this book?

First, this book is worth reading if you are determined to engage in automated testing – it will be useful as a "beginner" and have some experience in automation.

Secondly, this book can and should be used as reference material.

Thirdly, this book — a kind of "map", which has links to many external sources of information (which can be useful even experienced automation engineer), as well as many examples with explanations.

This book is not intended for people with high experience in test automation. From time to time, I use a learning approach and try to “chew” all the approaches and build the stages step by step.

Some people more experienced in software test automation also having may find it slow, boring, and monotonous.

This book is intended for people who first approach the creation of an automation testing framework, especially if their goal is to add automation to their test approach.

First of all, I wrote this book for a tester with experience in the field of “manual” software testing, the purpose of which is to move to a higher level in the tester career.

## Summary:

**We can safely say that this book is a kind of guide for beginners in the field of automation software testing.**

I have a huge knowledge of the field of test automation. I also have quite a lot of experience building automation on a project from scratch. I have repeatedly had to develop and implement the framework of testing automation on projects.

The learning approach focuses on a huge chunk of theory on building the automation testing framework. The book also discusses the theory of test automation in detail.

However, the direction of automation to support testing is no longer limited to testing, so this book is suitable for anyone who wants to improve the use of automation: managers, business analysts, users, and, of course, testers.

Testers use different approaches for testing on projects. I remember when I first started doing testing, I was drawing information from traditional books and was unnecessarily confused by some concepts that I rarely had to use. And most of the books, to my great regret, did not address the aspects and approaches to test automation. Most books on testing begin by showing how you can test a software product with basic approaches. But I do not consider the approaches and implementations of test automation at the testing stage.

**My main** goal is to help you start building an automation testing framework using a strategy and have the basic knowledge you need to do so.

This book focuses on theory rather than a lot of additional libraries, because once you have the basics, building a library and learning how to use it becomes a matter of reading the documentation.

This book is not an "exhaustive" introduction. This is a guide to getting started in building an automation testing framework. I focused on the examples.

I argue that in order to start implementing an automation testing framework, you need a basic set of knowledge in testing and management to start adding value to automation projects.

In fact, when I started creating the automation testing framework first, I used only the initial level of knowledge in the field of testing and development.

I also want the book to be small and accessible so that people actually read it and apply the approaches described in it in practice.

## Acknowledgments.

This book was created as a “work in progress” on **leanpub.com**. My thanks go to everyone who bought the book in its early stages, this provided the continued motivation to create something that added value, and then spends the extra time needed to add polish and readability.

**I am also grateful to every QA engineer that I have worked with who took the time to explain their approach. You helped me observe what a good QA engineer does and how they work. The fact that you were good, forced me to ‘up my game’ and improve both my coding and testing skills. All mistakes in this book are my fault.**

# Chapter 1. Selenium History.

Selenium WebDriver is a collection of open-source APIs which are used to automate the testing of a web application.

Selenium WebDriver tool is used to automate web application testing to verify that it works as expected. It supports many browsers such as Firefox, Chrome, IE, and Safari. However, using the Selenium WebDriver, we can automate testing for web applications only. It does not qualify for window-based applications. It also supports different programming languages such as C#, Java, Perl, PHP, and Ruby for writing test scripts. Selenium WebDriver is platform-independent since the same code can be used on different Operating Systems like Microsoft Windows, Apple OS, and Linux. It is one of the components of the selenium family, which also includes Selenium IDE, Selenium Client API, Selenium Remote Control, and Selenium Grid.

The story starts in 2004 at ThoughtWorks in Chicago, with Jason Huggins building the Core mode as "JavaScriptTestRunner" for the testing of an internal Time and Expenses application (Python, Plone). Automatic testing of any applications is core to ThoughtWork's style, given the Agile leanings of this consultancy. He has help from Paul Gross and Jie Tina Wang. For them, this was a day job.
Jason started demoing the test tool to various colleagues. Many were excited about its immediate and intuitive visual feedback, as well as its potential to grow as a reusable testing framework for other web applications.

Soon after in 2004 fellow **ThoughtWorker** Paul Hammant saw the demo, and started discussions about the open sourcing of Selenium, as well as defining a 'driven' mode of Selenium where you'd get to use Selenium over the wire from a language of your choice, that would get around the 'same origin policy'. Other (then) colleagues, Aslak Hellesoy and Mike Melia, experimented with different ideas for the 'server' piece, including page rewriting to get around the same origin policy. Paul wrote the original server piece in Java, and Aslak and Obie Fernandez ported that the client driver to Ruby, setting the foundation for drivers in yet more languages.

**ThoughtWorkers** in various offices around the world picked up Selenium for commercial projects, and contributed back to Selenium from the lessons learned on these projects. Mike Williams, Darrell Deboer, and Darren Cotterill all helped with the increasing the capabilities and the robustness of it.

At Bea, Dan Fabulich and Nelson Sproul came to the conclusion that the driver/server to browser architecture was not the most useful or flexible, so forked the driver coder and crafted that into a standalone server that leveraged and bundled MortBay's Jetty as a web-proxy. When the code was merged back it became known as "Selenium Remote Control" and the old driven code line and capability was retired.

Pat Lightbody became involved at the same time, with a commercial idea that required him to quit his day job (Jive Software). The idea was "Hosted QA", and it was eventually moved into Gomez's service line. Pat worked with Dan and Nelson making Selenium RC stable for large scale deployment.

Pat had privately coded a grid for Hosted QA that took screenshots of browsers in various states and was looking after multiple customers concurrently. Jason had the same hosted QA idea a year before but did not quit his day job to do it.

In 2007 Dan moved to the rapidly growing Redfin, which also part-time sponsors his time on Selenium, and encourages a speaking agenda.

Jason Huggins left Thoughtworks in 2007 and joined the (then secret) Selenium support team inside **Google**.

Jennifer Bevan (and other unnamed Googlers) had coded their own grid capability for Selenium RC and deployed it internally for the testing of multiple public web applications. Google hosted a GTAC conference in New York and talked about their use of Selenium for the first time. Jennifer soon became a committer on the Selenium projects.

Haw-bin Chai in Chicago provided patches for XPath functionality and developed an extension called "UI Element" that makes the grammar of locators much simpler. He was invited into the Selenium development team in 2007.

Simon Stewart at ThoughtWorks had been working on a different web testing tool called WebDriver. It did not rely on JavaScript to do the heavy lifting but instead had a client for each browser that was coded from scratch. It also had a 'higher level' API than Selenium-RC and showed lots of promise. Simon presented the tool at GTAC and started work on compatibility with Selenium-RC, which gave rise to the obvious conclusion that the two projects should merge.

Simon, at Google from 2007 to 2012, and now at Facebook, gets to spend some of his time making that a reality.