V Skills
Skills for a secure future

Certified Selenium Professional Sample Material

V-Skills Certifications

A Government of India
&
Government of NCT Delhi Initiative

V-Skills

Skills for a secure future

1. INTRODUCTION

1.1. Test Automation for Web Applications

Many, perhaps most, software applications today are written as web-based applications to be run in an Internet browser. The effectiveness of testing these applications varies widely among companies and organizations. In an era of highly interactive and responsive software processes where many organizations are using some form of Agile methodology, test automation is frequently becoming a requirement for software projects. Test automation is often the answer. Test automation means using a software tool to run repeatable tests against the application to be tested. For regression testing this provides that responsiveness.

There are many advantages to test automation. Most are related to the repeatability of the tests and the speed at which the tests can be executed. There are a number of commercial and open source tools available for assisting with the development of test automation. Selenium is possibly the most widely-used open source solution. This user's guide will assist both new and experienced Selenium users in learning effective techniques in building test automation for web applications.

This user's guide introduces Selenium, teaches its features, and presents commonly used best practices accumulated from the Selenium community. Many examples are provided. Also, technical information on the internal structure of Selenium and recommended uses of Selenium are provided.

Test automation has specific advantages for improving the long-term efficiency of a software team's testing processes. Test automation supports:

- ✓ Frequent regression testing
- ✓ Rapid feedback to developers
- ✓ Virtually unlimited iterations of test case execution
- ✓ Support for Agile and extreme development methodologies
- ✓ Disciplined documentation of test cases
- ✓ Customized defect reporting
- ✓ Finding defects missed by manual testing

1.2. To Automate or Not to Automate?

Is automation always advantageous? When should one decide to automate test cases?

It is not always advantageous to automate test cases. There are times when manual testing may be more appropriate. For instance, if the application's user interface will change considerably in the near future, then any automation might need to be rewritten anyway. Also, sometimes there simply is not enough time to build test automation. For the short term, manual testing may be more effective. If an application has a very tight deadline, there is currently no test automation available, and it's imperative that the testing get done within that time frame, then manual testing is the best solution.

1.3. Introducing Selenium

Selenium is a set of different software tools each with a different approach to supporting test automation. Most Selenium QA Engineers focus on the one or two tools that most meet the needs of their project, however learning all the tools will give you many different options for approaching different test automation problems. The entire suite of tools results in a rich set of testing functions specifically geared to the needs of testing of web applications of all types. These operations are highly flexible, allowing many options for locating UI elements and comparing expected test results against actual application behavior. One of Selenium's key features is the support for executing one's tests on multiple browser platforms.

1.4. Brief History of The Selenium Project

Selenium first came to life in 2004 when Jason Huggins was testing an internal application at ThoughtWorks. Being a smart guy, he realized there were better uses of his time than manually stepping through the same tests with every change he made. He developed a Javascript library that could drive interactions with the page, allowing him to automatically rerun tests against multiple browsers. That library eventually became Selenium Core, which underlies all the functionality of Selenium Remote Control (RC) and Selenium IDE. Selenium RC was ground-breaking because no other product allowed you to control a browser from a language of choice.

While Selenium was a tremendous tool, it wasn't without its drawbacks. Because of its Javascript based automation engine and the security limitations browsers apply to Javascript, different things became impossible to do. To make things worse, webapps became more and more powerful over time, using all sorts of special features new browsers provide and making this restrictions more and more painful.

In 2006 a plucky engineer at Google named Simon Stewart started work on a project he called WebDriver. Google had long been a heavy user of Selenium, but testers had to work around the limitations of the product. Simon wanted a testing tool that spoke directly to the browser using the 'native' method for the browser and operating system, thus avoiding the restrictions of a sandboxed Javascript environment. The WebDriver project began with the aim to solve the Selenium' pain-points.

Jump to 2008. The Beijing Olympics mark China's arrival as a global power, massive mortgage default in the United States triggers the worst international recession since the Great Depression, The Dark Knight is viewed by every human (twice), still reeling from the untimely loss of Heath Ledger. But the most important story of that year was the merging of Selenium and WebDriver. Selenium had massive community and commercial support, but WebDriver was clearly the tool of the future. The joining of the two tools provided a common set of features for all users and brought some of the brightest minds in test automation under one roof. Perhaps the best explanation for why WebDriver and Selenium are merging was detailed by Simon Stewart, the creator of WebDriver, in a joint email to the WebDriver and Selenium community on August 6, 2009.

"Why are the projects merging? Partly because webdriver addresses some shortcomings in selenium (by being able to bypass the JS sandbox, for example. And we've got a gorgeous API), partly because selenium addresses some shortcomings in webdriver (such as supporting a broader range of browsers) and partly because the main selenium contributors and I felt that it was the best way to offer users the best possible framework."

1.5. Selenium's Tool Suite

Selenium is composed of multiple software tools. Each has a specific role.

Selenium 2 (aka. Selenium Webdriver): Selenium 2 is the future direction of the project and the newest addition to the Selenium toolkit. This brand new automation tool provides all sorts of awesome features, including a more cohesive and objects oriented API as well as an answer to the limitations of the old implementation.

As you can read in Brief History of The Selenium Project, both the Selenium and WebDriver developers agreed that both tools have advantages and that merging the two projects would make a much more robust automation tool.

Selenium 2.0 is the product of that effort. It supports the WebDriver API and underlying technology, along with the Selenium 1 technology underneath the WebDriver API for maximum flexibility in porting tests. In addition, Selenium 2 still runs Selenium 1's Selenium RC interface for backwards compatibility.

Selenium 1 (aka. Selenium RC or Remote Control: As you can read in Brief History of The Selenium Project, Selenium RC was the main Selenium project for a long time, before the WebDriver/Selenium merge brought up Selenium 2, the newest and more powerful tool. Now Selenium 1 is deprecated and is not actively supported (mostly in maintenance mode).

Selenium IDE: Selenium IDE (Integrated Development Environment) is a prototyping tool for building test scripts. It is a Firefox plugin and provides an easy-to-use interface for developing automated tests. Selenium IDE has a recording feature, which records user actions as they are performed and then exports them as a reusable script in one of many programming languages that can be later executed.

Even though Selenium IDE has a "Save" feature that allows users to keep the tests in a table-based format for later import and execution, it is not designed to run test passes nor is it designed to build all the automated tests you will need. Specifically, Selenium IDE doesn't provide iteration or conditional statements for test scripts. At the time of writing there is no plan to add such thing. The reasons are partly technical and partly based on the Selenium developers encouraging best practices in test automation which always requires some amount of programming. Selenium IDE is simply intended as a rapid prototyping tool. The Selenium developers recommend for serious, robust test automation either Selenium 2 or Selenium 1 to be used with one of the many supported programming languages.

Selenium-Grid: Selenium-Grid allows the Selenium RC solution to scale for large test suites and for test suites that must be run in multiple environments. Selenium Grid allows you to run tests in parallel, that is, different tests can be run at the same time on different remote machines. This has two advantages. First, if you have a large test suite, or a slow-running test suite, you can boost its performance substantially by using Selenium Grid to divide test suite to run different tests at the same time using those different machines. Also, if you must run test suite on multiple environments you can have different remote machines supporting and running tests in them at the same time. In each case Selenium Grid greatly improves the time it takes to run suite by making use of parallel processing.

Differences between Selenium 1 and 2

Selenium 1

- ✓ supports real browsers only – no HtmlUnit support, needs a graphical desktop environment (slower but more realistic compared to HtmlUnit)
- ✓ JavaScript based approach
- ✓ when used in unit tests, you need a proxy application to control the browser (Selenium-RC)
- ✓ limited by the browser's Javascript security model
- ✓ complex API, API has evolved over time (dictionary-based approach)
- ✓ API methods for specific HTML elements – e.g. checking checkboxes, selecting radio buttons, selecting elements from dropdown menus etc.

Selenium 2

- ✓ Selenium 2 has been created from Selenium 1 and the Webdriver project
- ✓ backwards compatible to Selenium 1
- ✓ Webdriver API
  - ✓ simple, object oriented API
  - ✓ supports real browsers and headless HTMLUnit (HTMLUnit should be faster than testing with a real browser)
  - ✓ uses most appropriate mechanism to control the browser (IE: automation controls, Firefox: Extension, Chrome: Webdriver API)
  - ✓ not limited by browser's Javascript security model
  - ✓ more realistic simulation of the user's actions by using OS level events (typing, mouse, ...)
  - ✓ no API methods for specific HTML elements (e.g. checking checkboxes etc.) – you use click and type commands only (this could be a disadvantage)
  - ✓ you can no longer manipulate non-visible fields (e.g. hidden fields) (this could be a disadvantage)
  - ✓ methods to wait for elements to appear – very useful for testing ajax applications
- ✓ you do no longer need selenium-rc (proxy to control the browser) in your unit tests. Webdriver is able to control the browser directly
- ✓ Selenium-RC and Selenium-Grid from Selenium 1 have been merged into Selenium Server
- ✓ supports all major desktop browsers (IE, Firefox, Opera, Webkit, Chrome), mobile browsers (IPhone, Android) and headless HtmlUnit

1.6. Choosing Selenium Tool

Many people get started with Selenium IDE. If you are not already experienced with a programming or scripting language you can use Selenium IDE to get familiar with Selenium commands. Using the IDE you can create simple tests quickly, sometimes within seconds.

We don't, however, recommend you do all test automation using Selenium IDE. To effectively use Selenium you will need to build and run tests using either Selenium 2 or Selenium 1 in conjunction with one of the supported programming languages. Which one you choose depends on you.

At the time of writing the Selenium developers are planning on the Selenium-WebDriver API being the future direction for Selenium. Selenium 1 is provided for backwards compatibility. Still, both have strengths and weaknesses which are discussed in the corresponding chapters of this document.

We recommend those who are completely new to Selenium to read through these sections. However, for those who are adopting Selenium for the first time, and therefore building a new test suite from scratch, you will probably want to go with Selenium 2 since this is the portion of Selenium that will continue to be supported in the future.

1.7. Supported Browsers and Platforms

In Selenium 2.0, the supported browsers vary depending on whether you are using Selenium-WebDriver or Selenium-RC.

Selenium-WebDriver
Selenium-WebDriver supports the following browsers along with the operating systems these browsers are compatible with

- ✓ Google Chrome
- ✓ Internet Explorer 6, 7, 8, 9, 10 - 32 and 64-bit where applicable
- ✓ Firefox: latest ESR, previous ESR, current release, one previous release
- ✓ Safari
- ✓ Opera
- ✓ HtmlUnit
- ✓ phantomjs
- ✓ Android (with Selendroid or appium)
- ✓ iOS (with ios-driver or appium)

Selenium 1.0 and Selenium-RC.
This is the old, support platform for Selenium 1.0. It should still apply to the Selenium 2.0 release of Selenium-RC.

Browser	Selenium IDE	Selenium 1 (RC)	Operating Systems
Firefox 3.x	Record and playback tests	Start browser, run tests	Windows, Linux, Mac
Firefox 3	Record and playback tests	Start browser, run tests	Windows, Linux, Mac
Firefox 2	Record and playback tests	Start browser, run tests	Windows, Linux, Mac
IE 8	Test execution only via Selenium RC*	Start browser, run tests	Windows
IE 7	Test execution only via Selenium RC*	Start browser, run tests	Windows
IE 6	Test execution only via Selenium RC*	Start browser, run tests	Windows
Safari 4	Test execution only via Selenium RC	Start browser, run tests	Windows, Mac
Safari 3	Test execution only via Selenium RC	Start browser, run tests	Windows, Mac

Browser	Selenium IDE	Selenium 1 (RC)	Operating Systems
Safari 2	Test execution only via Selenium RC	Start browser, run tests	Windows, Mac
Opera 10	Test execution only via Selenium RC	Start browser, run tests	Windows, Linux, Mac
Opera 9	Test execution only via Selenium RC	Start browser, run tests	Windows, Linux, Mac
Opera 8	Test execution only via Selenium RC	Start browser, run tests	Windows, Linux, Mac
Google Chrome	Test execution only via Selenium RC	Start browser, run tests	Windows, Linux, Mac
Others	Test execution only via Selenium RC	Partial support possible**	As applicable

Tests developed on Firefox via Selenium IDE can be executed on any other supported browser via a simple Selenium RC command line.

Selenium RC server can start any executable, but depending on browser security settings there may be technical limitations that would limit certain features.

1.8. Flexibility and Extensibility

You'll find that Selenium is highly flexible. There are many ways you can add functionality to both Selenium test scripts and Selenium's framework to customize test automation. This is perhaps Selenium's greatest strength when compared with other automation tools. These customizations are described in various places throughout this document. In addition, since Selenium is Open Source, the source code can always be downloaded and modified.