Selenium

PV260 Software Quality

Matěj Karolyi

29. 4. 2021

Terms and Conditions

This PDF file is available only for students of the PV260 Software Quality course for learning purposes only.

Please do not modify or redistribute this PDF file.

What is Selenium

> *"Selenium automates browsers. That's it! What you do with that power is entirely up to you. Primarily, it is for automating web applications for testing purposes, but is certainly not limited to just that. Boring web-based administration tasks can (and should!) also be automated as well."*

Selenium homepage: `https://www.selenium.dev/`

Selenium IDE

> *"Open source record and playback test automation for the web."*

Available from `https://www.selenium.dev/selenium-ide/`

Works as an extension of a web browser (Firefox, Chrome, Safari).

- **Web Ready**: turn-key solution to quickly author reliable end-to-end tests. Works out of the box for any web app.
- **Easy Debugging**: easier test debugging with rich IDE features like setting breakpoints and pausing on exceptions.
- **Cross-browser Execution**: run your tests on any browser/OS combination in parallel using the Command-line Runner for Selenium IDE.

How to start working with Selenium WebDriver

1. Go to https://www.selenium.dev/download/ and download the latest "Selenium Standalone Server" jar file.
2. Go to https://sites.google.com/a/chromium.org/chromedriver/downloads and download the latest chromedriver.
3. Go to https://github.com/junit-team/junit4/wiki/Download-and-Install and download the jars for junit and hamcrest-core.
4. Create java project in your IDE (or use some already existing project).
5. Add jar files from previous steps to Libraries.
6. Write your first test.

Example test in Selenium

```
@Test

public void testSeleniumSoftwareIsOnWikipedia(){

    System.setProperty("webdriver.chrome.driver",
    "D:\\chromedriver_win32\\chromedriver.exe");

    WebDriver webDriver = new ChromeDriver();

    webDriver.get("https://cs.wikipedia.org");

    WebElement searchField = webDriver.findElement(By.name("search"));

    searchField.sendKeys("selenium");

    searchField.submit();

    assertTrue("There is Selenium software displayed.",

    webDriver.getPageSource().contains("Selenium (software)"));

    webDriver.close();

}
```

Extending Selenium Framework

When you want to cover your web application by Selenium-based tests, you probably want an extended framework based on Selenium.

It might be handy to create class MyWebApplication with methods like:
- `clickLink(String linkText)`
- `setFieldWithValue(String field, String value)`
- `clickButton(String buttonName)`

and many others. Some of those methods might be specific for the web UI of your application not very common on the Internet.

You might also create objects for specific pages of your application. For example `RegisterPage` with method like `register(User user)` or `register(String email, String userName, String password, String repeatPassword)`.

Extending Selenium Framework

(continuing)

You also probably want some basic configuration in your extension of Selenium framework. For example:

1. the browser you want to use
2. whether to close the browser at the end of the test or not
3. what is the maximum execution time after the tests time out
4. base URL of the application (quite probable it is different on production and on different test environments)
5. etc

Logging might be also very useful for Selenium based frameworks. Note that the tests might want to be run in some Continuous Integration system. In case a test fails, you want to know what is happening and why the test failed. Logs will help you achieve that. You might also want to take and save a screenshot of the browser when a test fails.

Links for more information

- Good video introduction for Selenium
  https://www.youtube.com/watch?v=-eJ2cZXyJ0E
- Documentation
  https://www.selenium.dev/selenium/docs/api/java/overview-summary.html
- Selenium Homepage
  https://www.selenium.dev/