WebDriverManager

Table of Contents

Motivation

Selenium WebDriver allows to control different types of browsers (such as Chrome, Firefox, Edge, and so on) programmatically using different programming languages. This is very useful to implement automated tests for web applications. Nevertheless, in order to use WebDriver, we need to pay a prize. For security reasons, the automated manipulation of a browser can only be done using native features of the browser. In practical terms, it means that a binary file must be placed in between the test using the WebDriver API and the actual browser. One the one hand, the communication between the WebDriver object and that binary is done using the (W3C WebDriver specification, formerly called *JSON Wire Protocol*. It consists basically on a REST service using JSON for requests and responses. On the other hand, the communication between the binary and the browser is done using native capabilities of the browser. Therefore, the general schema of Selenium WebDriver can be illustrated as follows:

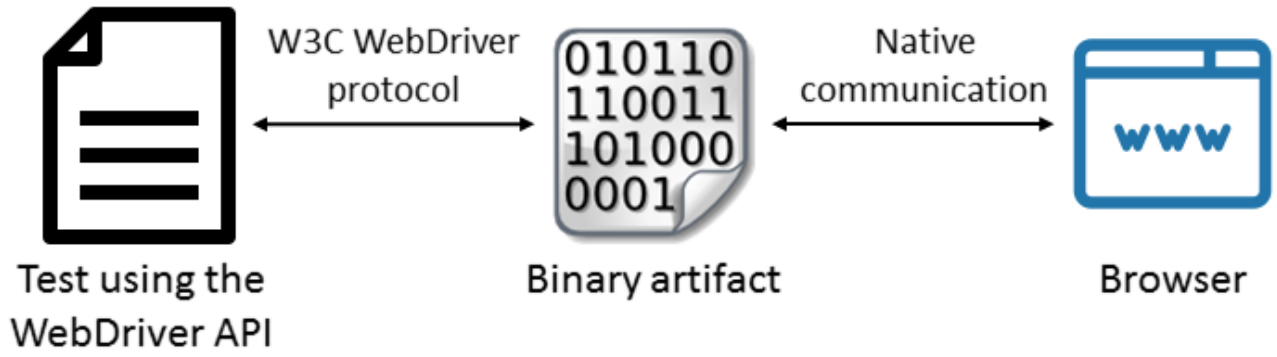

Figure 1. WebDriver general scenario

From a tester point of view, the need of this binary component is a pain in the neck, since it should be downloaded manually for the proper platform running the test (i.e. Windows, Linux, Mac). Moreover, the binary version should be constantly updated. The majority of browsers evolve quite fast, and the corresponding binary file required by WebDriver needs to be also updated. The following picture shows a fine-grained diagram of the different flavor of WebDriver binaries and browsers:

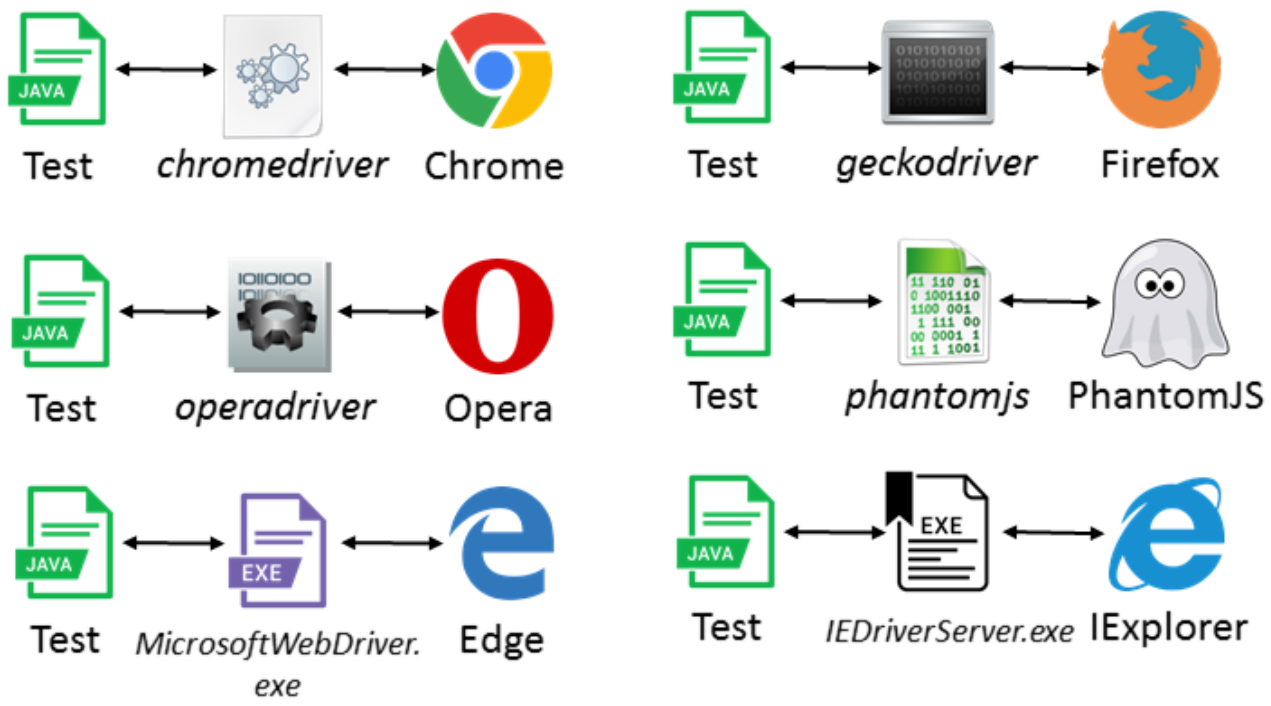

Figure 2. WebDriver scenario for Chrome, Firefox, Opera, PhantomJS, Edge, and Internet Explorer

Concerning Java, in order to locate these drivers, the absolute path of the binary controlling the browser should be exported in a given environment variable before creating a WebDriver instance, as follows:

```
System.setProperty("webdriver.chrome.driver", "/path/to/chromedriver");
System.setProperty("webdriver.opera.driver", "/path/to/operadriver");
System.setProperty("webdriver.ie.driver", "C:/path/to/IEDriverServer.exe");
System.setProperty("webdriver.edge.driver", "C:/path/to/MicrosoftWebDriver.exe");
System.setProperty("phantomjs.binary.path", "/path/to/phantomjs");
System.setProperty("webdriver.gecko.driver", "/path/to/geckodriver");
```

In order to simplify the life of Java WebDriver users, in March 2015 the utility WebDriverManager was first released. WebDriverManager is a library which automates all this process (download the proper binary and export the proper variable) for Java in runtime. The WebDriverManager API is quite simple, providing a singleton object for each of the above mentioned browsers:

```
WebDriverManager.chromedriver().setup();
WebDriverManager.firefoxdriver().setup();
WebDriverManager.operadriver().setup();
WebDriverManager.phantomjs().setup();
WebDriverManager.edgedriver().setup();
WebDriverManager.iedriver().setup();;
```

Setup

Using WebDriverManager is very easy. First, you need to import the dependency in your project (typically as *test* dependency). In Maven, it is done as follows:

```
<dependency>
    <groupId>io.github.bonigarcia</groupId>
    <artifactId>webdrivermanager</artifactId>
    <version>5.0.0</version>
    <scope>test</scope>
</dependency>
```

The dependency (typically *test*) to be declared in a Gradle project is as follows:

```
dependencies {
    testImplementation("io.github.bonigarcia:webdrivermanager:5.0.0")
}
```

Then, you need to declare *Selenium-Jupiter* extension in your JUnit 5 test, simply annotating your test with `@ExtendWith(SeleniumJupiter.class)`. Finally, you need to include one or more parameters in your `@Test` methods (or constructor) whose types implements the `WebDriver` interface (e.g. `ChromeDriver` to use Chrome, `FirefoxDriver` for Firefox, and so for). That's it. *Selenium-Jupiter* control the lifecycle of the `WebDriver` object internally, and you just need to use the `WebDriver` object in your test to drive the browser(s) you want. For example:

```
```

```
import static java.lang.invoke.MethodHandles.lookup;
import static org.assertj.core.api.Assertions.assertThat;
import static org.slf4j.LoggerFactory.getLogger;

import java.time.Duration;

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Wait;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.slf4j.Logger;

import io.github.bonigarcia.wdm.WebDriverManager;

/**
 * Test with Google Chrome browser.
 *
 * @author Boni Garcia
 * @since 1.0.0
 */
class ChromeTest {

    final Logger log = getLogger(lookup().lookupClass());

    WebDriver driver;

    @BeforeAll
    static void setupClass() {
        WebDriverManager.chromedriver().setup();
    }

    @BeforeEach
    void setupTest() {
        driver = new ChromeDriver();
    }

    @AfterEach
    void teardown() {
        if (driver != null) {
            driver.quit();
        }
    }

    @Test
    void test() {
        String sutUrl = "https://github.com/bonigarcia/webdrivermanager";
        driver.get(sutUrl);
```

```
        String title = driver.getTitle();
        log.debug("The title of {} is {}", sutUrl, title);

        Wait<WebDriver> wait = new WebDriverWait(driver,
                Duration.ofSeconds(30));
        wait.until(d -> d.getTitle().contains("Selenium WebDriver"));
        assertThat(driver.getTitle()).containsIgnoringCase("WebDriverManager");
    }

}
```

NOTE	As of JUnit 5.1, extensions can be registered programmatically via `@RegisterExtension` or automatically via Java's `ServiceLoader`. Both mechanisms can be used with *Selenium-Jupiter*.

Features

Driver Management

Under construction.

Browser Finder

Under construction.

WebDriver Builder

Under construction.

Browsers in Docker

Under construction.

Other Usages

WDM CLI

Under construction.

WDM Server

Under construction.

WDM in Docker

Under construction.

WDM Agent

Under construction.

Examples

Under construction.

Configuration

Under construction.

Known issues

Under construction.

Support

Under construction.

Contributing

Under construction.

Further documentation

Under construction.

About

WebDriverManager (Copyright © 2017-2021) is a project created by Boni Garcia (@boni_gg) licensed under Apache 2.0 License. This documentation is released under the terms of CC BY 3.0 (also available in PDF). There are several ways to get in touch with WebDriverManager:

- Questions about WebDriverManager are supposed to be discussed in StackOverflow, using the tag *webdrivermanager_java*.
- Comments, suggestions and bug-reporting should be done using the GitHub issues.
- If you think WebDriverManager can be enhanced, consider contribute to the project by means

of a pull request.