Selenium for Java Developers

JUG Saxony Day
Dresden, Germany
September 26, 2025

Boni García
https://bonigarcia.dev/

What is Selenium?

> *Selenium WebDriver (a.k.a. Selenium) is a* ***browser automation library***

- Multilanguage
- Cross-browser
- Open-source and community-driven since 2004

https://selenium.dev/

Fork me on GitHub

Selenium Hello World

```java
public class HelloWorldSelenium {

    public static void main(String[] args) {
        // Open Chrome
        WebDriver driver = new ChromeDriver();

        // Navigate to web page
        String url = "https://bonigarcia.dev/selenium-webdriver-java/";
        driver.get(url);

        // Check page title
        String title = driver.getTitle();
        System.out.println(String.format("The title of %s is %s", url, title));

        // Close Chrome
        driver.quit();
    }

}
```

Java

```maven
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.35.0</version>
</dependency>
```

Maven

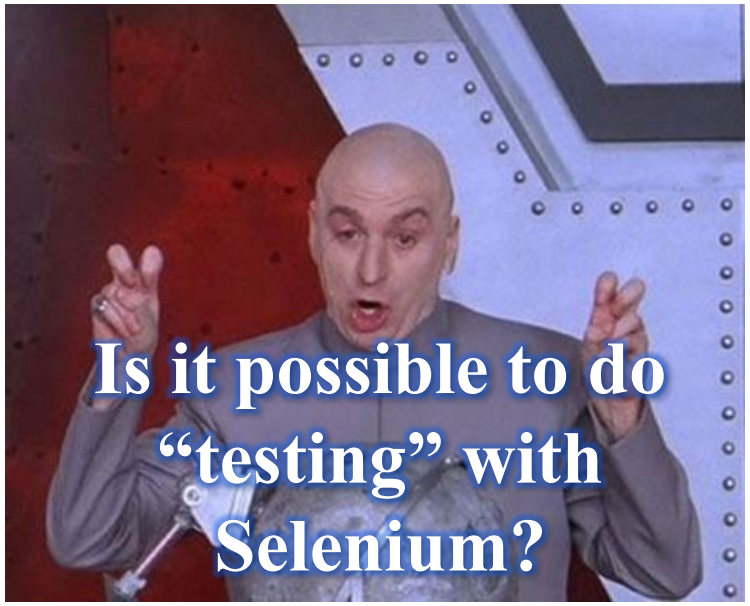
```gradle
dependencies {
    implementation("org.seleniumhq.selenium:selenium-java:4.35.0")
}
```

Gradle

What is NOT Selenium?

> *Selenium is **not** a **testing framework***

- ✗ Test runner
- ✗ Assertions
- ✗ Reporting
- ✗ Integration with CI/CD
- ✗ Other testing features



Ecosystem

WebDriverManager

Selenide
CONCISE UI TESTS IN JAVA

Serenity BDD

Galen Framework

Selenium-Jupiter

Selenium

JUnit 5

TestNG

AssertJ

cucumber

REST-assured

mockito

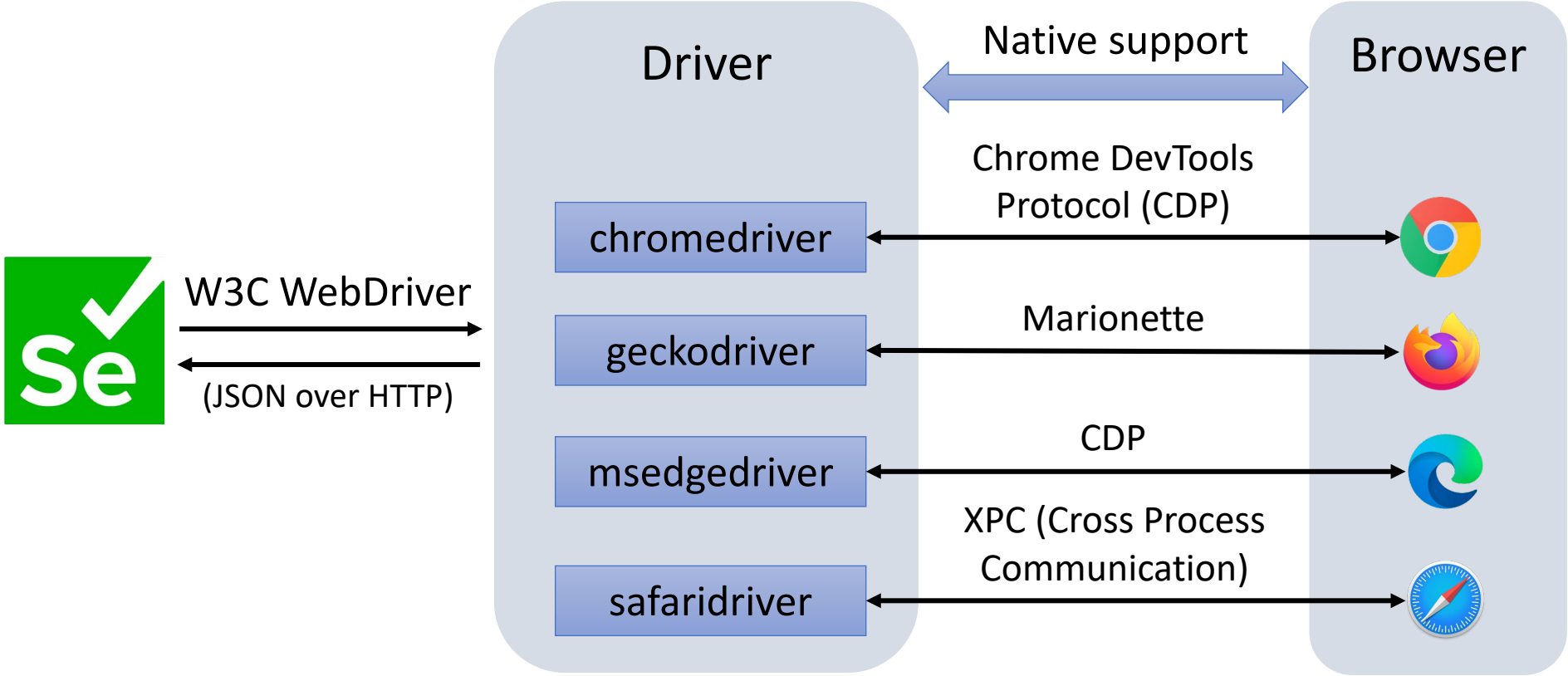
Datafaker

Allure Report

EXTENT REPORT

Java

Selenium Architecture



Ecosystem – WebDriverManager

WebDriverManager

“Automated driver management and other helper features for Selenium WebDriver in Java

https://bonigarcia.dev/webdrivermanager/

Selenium Hello World (E2E Test with WDM)

Fork me on GitHub

```java
class HelloWorldChromeJupiterTest {

    WebDriver driver;

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

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

    @AfterEach
    void teardown() {
        driver.quit();
    }

    @Test
    void test() {
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/");
        assertThat(driver.getTitle()).contains("Selenium WebDriver");
    }

}
```

```xml
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.35.0</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>5.13.4</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.assertj</groupId>
    <artifactId>assertj-core</artifactId>
    <version>3.27.6</version>
    <scope>test</scope>
</dependency>
<dependency>
   <groupId>io.github.bonigarcia</groupId>
   <artifactId>webdrivermanager</artifactId>
   <version>6.3.2</version>
   <scope>test</scope>
</dependency>
```

Selenium Manager

Selenium Manager

> *Selenium Manager is the official driver manager of the Selenium project, and it is shipped out of the box with every Selenium release*

https://www.selenium.dev/documentation/selenium_manager/

Selenium Hello World (E2E Test)

Fork me on GitHub

```java
class HelloWorldFirefoxJupiterTest {

    WebDriver driver;

    @BeforeEach
    void setup() {
        driver = new FirefoxDriver();
    }

    @AfterEach
    void teardown() {
        driver.quit();
    }

    @Test
    void test() {
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/");
        assertThat(driver.getTitle()).contains("Selenium WebDriver");
    }

}
```

```xml
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.35.0</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>5.13.4</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.assertj</groupId>
    <artifactId>assertj-core</artifactId>
    <version>3.27.6</version>
    <scope>test</scope>
</dependency>
```

Selenium Manager – Drivers

- Selenium Manager automatically discovers, downloads, and caches the drivers required by Selenium

1. Browser version discovery
2. Driver version discovery
3. Driver download and cache

Chrome 134

chromedriver
134.0.6998.165

`~/.cache/selenium`

Selenium Manager – Browsers

- Selenium Manager automatically discovers, downloads, and caches the browsers driven with Selenium when these browsers are not installed in the local system

	Test		
	✓	✓	✓*
	✓	✓	✓
	✓	✓	✓

*Requires admin permissions

Selenium Manager – Browsers

- **Chrome for Testing** (CfT) is a specialized version of Google Chrome designed specifically for browser automation
  - It is not evergreen (no auto-updates)
  - It is lighter that regular Google Chrome (e.g., user-related components, such as the user profile or password manager are not available in CfT)

https://googlechromelabs.github.io/chrome-for-testing/

Selenium Manager – Browsers

Fork me on GitHub

```
class ChromeVersionTest {

    WebDriver driver;

    @BeforeEach
    void setup() {
        ChromeOptions options = new ChromeOptions();
        options.setBrowserVersion("beta");
        driver = new ChromeDriver(options);
    }

    @Test
    void test() {
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/");
        String title = driver.getTitle();
        assertThat(title).contains("Selenium WebDriver");
    }

    @AfterEach
    void teardown() {
        driver.quit();
    }

}
```

Specific browser versions (including "beta", "dev", or "nightly") are supported

Selenium API

- Document Object Model (DOM) manipulation
- Impersonate user actions (keyboard, mouse)
- Waiting strategies
- Execute JavaScript
- Make screenshots
- Manage browser (e.g., headless, history, …)
- Chrome DevTools Protocol
- WebDriver BiDi
- …

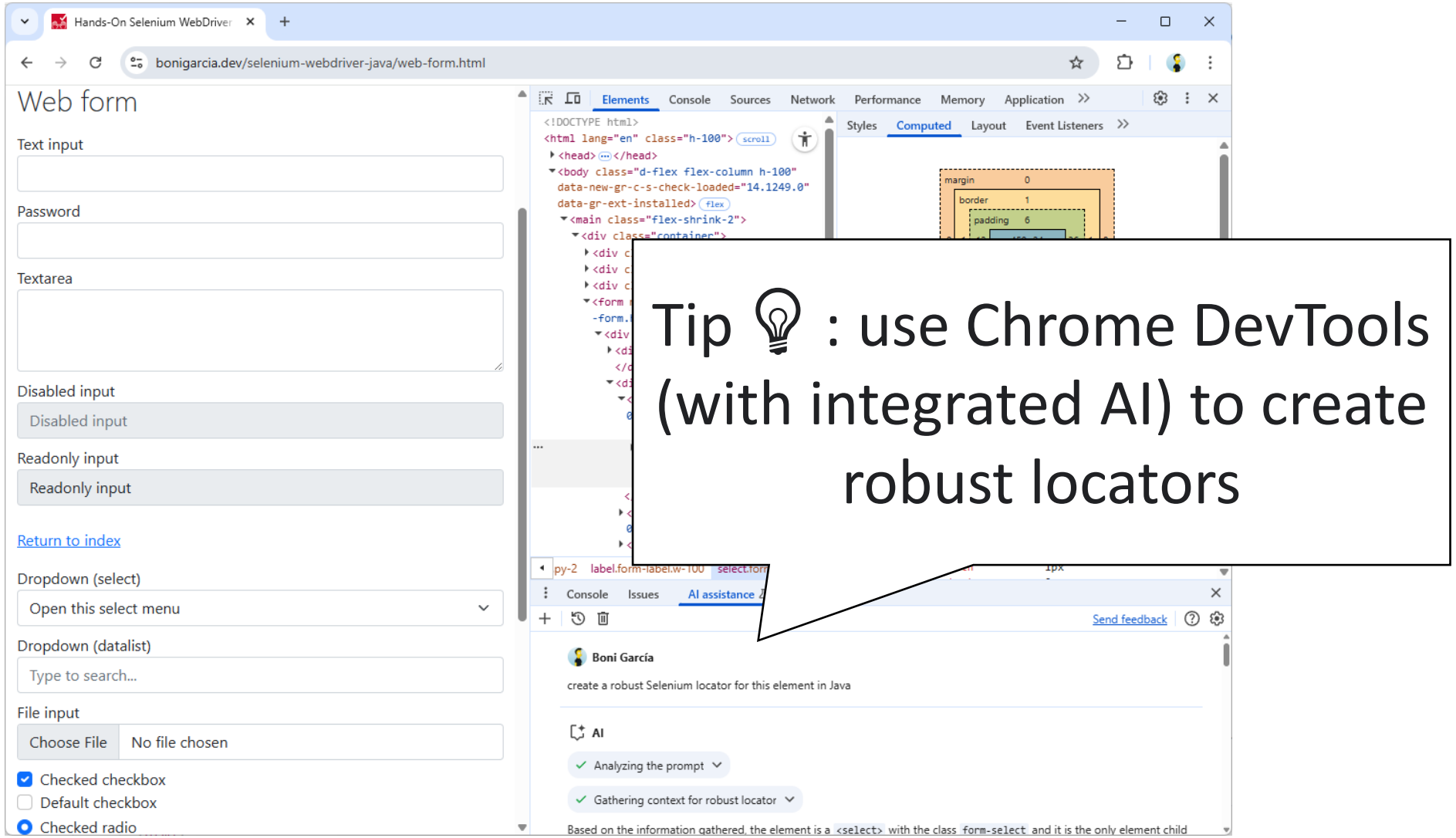
Selenium API – Locators

- **Locators** are used to identify and interact with web elements

Id	Name	Link text	Partial link text	Tag name	Class name	CSS selector	XPath

```
WebElement username = driver.findElement(By.id("username"));
WebElement textByName = driver.findElement(By.name("my-text"));
WebElement linkByText = driver.findElement(By.linkText("Return to index"));
WebElement linkByPartialText = driver.findElement(By.partialLinkText("index"));
WebElement textarea = driver.findElement(By.tagName("textarea"));
WebElement alert = driver.findElement(By.className("alert"));
WebElement hidden = driver.findElement(By.cssSelector("input[type=hidden]"));
WebElement radio = driver.findElement(By.xpath("//*[@type='radio' and @checked]"));
```

Selenium API – Locators



Selenium API – Waiting Strategies

- **Waiting strategies** are used to handle synchronization between the Selenium script and the actual response speed of the web element

Implicit Wait | Explicit Wait | Fluent Wait

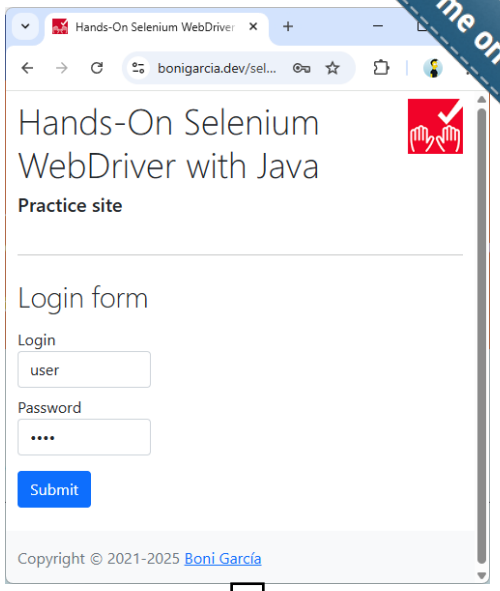
```
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
```

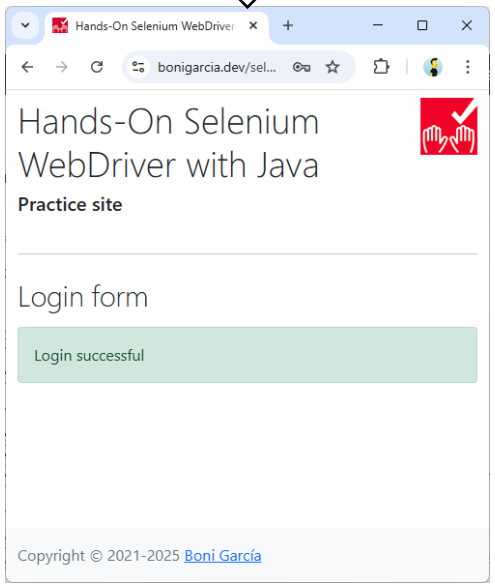
```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
WebElement landscape = wait.until(ExpectedConditions
        .presenceOfElementLocated(By.id("landscape")));
```

```
Wait<WebDriver> wait = new FluentWait<>(driver)
        .withTimeout(Duration.ofSeconds(10))
        .pollingEvery(Duration.ofSeconds(1))
        .ignoring(NoSuchElementException.class);
WebElement landscape = wait.until(ExpectedConditions
        .presenceOfElementLocated(By.id("landscape")));
```

Selenium API – Waiting Strategies

```
class LoginSeleniumTest {

    // Fixture

    @Test
    void test() throws Exception {
        // Open system under test (SUT)
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/login-form.html");

        // Log in
        driver.findElement(By.id("username")).sendKeys("user");
        driver.findElement(By.id("password")).sendKeys("user");
        driver.findElement(By.cssSelector("button[type='submit']")).click();

        // Assert expected text
        WebElement successElement = driver.findElement(By.id("success"));  // FIXME: flaky
        assertThat(successElement.getText()).contains("Login successful");

        // Take screenshot
        File screenshot = ((TakesScreenshot) driver).getScreenshotAs(FILE);
        Path destination = Paths.get("login-selenium.png");
        Files.move(screenshot.toPath(), destination, REPLACE_EXISTING);
    }

}
```





Fork me on GitHub

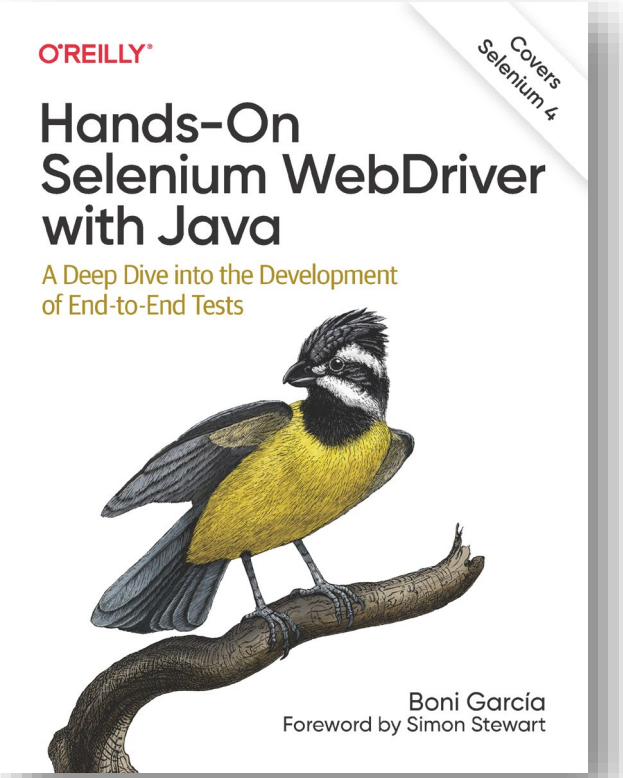
Selenium API – Waiting Strategies

Fork me on GitHub

```
class SlowLoginSeleniumTest {

    // Fixture

    @Test
    void test() throws Exception {
        // Open system under test (SUT)
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/login-slow.html");

        // Log in
        driver.findElement(By.id("username")).sendKeys("user");
        driver.findElement(By.id("password")).sendKeys("user");
        driver.findElement(By.cssSelector("button[type='submit']")).click();

        // Assert expected text
        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
        WebElement successElement = wait.until(ExpectedConditions.presenceOfElementLocated(By.id("success")));
        assertThat(successElement.getText()).contains("Login successful");

        // Take screenshot
        File screenshot = ((TakesScreenshot) driver).getScreenshotAs(FILE);
        Path destination = Paths.get("slow-login-selenium.png");
        Files.move(screenshot.toPath(), destination, REPLACE_EXISTING);
    }

}
```

Explicit wait

Selenium API – Examples



https://github.com/bonigarcia/selenium-webdriver-java

https://github.com/bonigarcia/selenium-examples

https://github.com/bonigarcia/browser-automation-apis/

Ecosystem – WebDriverManager

Fork me on GitHub

```
class DockerChromeTest {

    WebDriver driver;
    WebDriverManager wdm;

    @BeforeEach
    void setupTest() {
        wdm = WebDriverManager.chromedriver().browserInDocker();
        driver = wdm.create();
    }

    @Test
    void test() {
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/");
        assertThat(driver.getTitle()).contains("Selenium WebDriver");
    }

    @AfterEach
    void teardown() {
        wdm.quit();
    }

}
```

Docker Selenium
Docker images for Selenium Grid

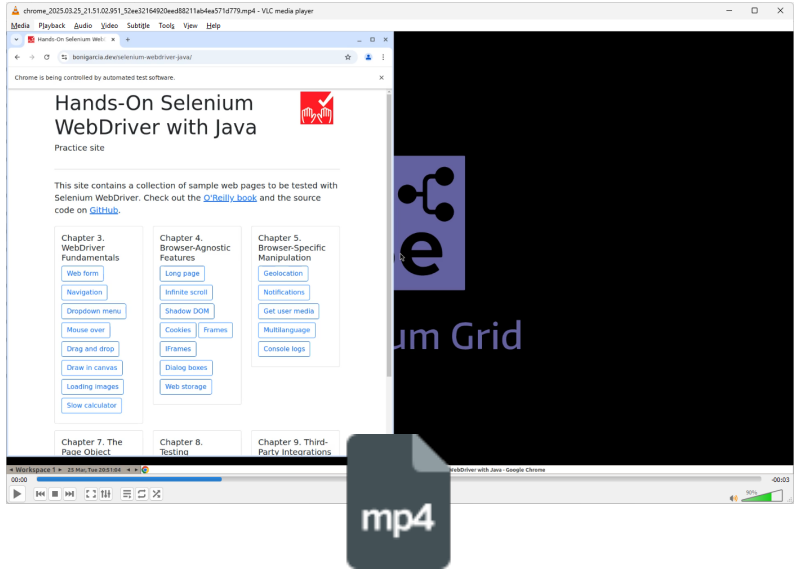
WebDriverManager

https://bonigarcia.dev/webdrivermanager/

Ecosystem – WebDriverManager

Fork me on GitHub

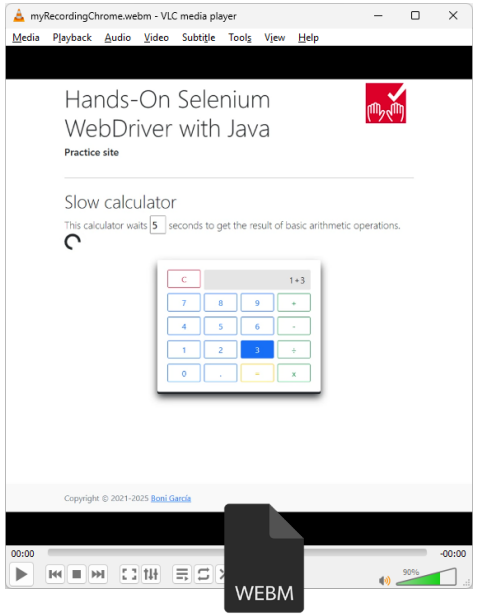
```
class DockerChromeRecordingTest {

    WebDriver driver;
    WebDriverManager wdm;

    @BeforeEach
    void setupTest() {
        wdm = WebDriverManager.chromedriver().browserInDocker().enableRecording();
        driver = wdm.create();
    }

    @Test
    void test() {
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/");
        assertThat(driver.getTitle()).contains("Selenium WebDriver");
    }

    @AfterEach
    void teardown() {
        wdm.quit();
    }

}
```



Ecosystem – WebDriverManager

Fork me on GitHub

```
class RecordEdgeTest {

    WebDriver driver;
    File targetFolder;
    WebDriverManager wdm;

    @BeforeEach
    void setup() {
        wdm = WebDriverManager.edgedriver().watch();
        driver = wdm.create();
    }

    @Test
    void test() {
        driver.get(
                "https://bonigarcia.dev/selenium-webdriver-java/slow-calculator.html");
        wdm.startRecording();
        // test logic
        wdm.stopRecording();
    }

    @AfterEach
    void teardown() {
        driver.quit();
    }

}
```



BrowserWatcher

https://bonigarcia.dev/browserwatcher/

Ecosystem – WebDriverManager

Fork me on GitHub

```
class GatherLogsFirefoxTest {

    WebDriverManager wdm;
    WebDriver driver;

    @BeforeEach
    void setup() {
        wdm = WebDriverManager.firefoxdriver().watch();
        driver = wdm.create();
    }

    @Test
    void test() {
        driver.get(
                "https://bonigarcia.dev/selenium-webdriver-java/console-logs.html");
        List<Map<String, Object>> logMessages = wdm.getLogs();

        // handle logs
    }

    @AfterEach
    void teardown() {
        driver.quit();
    }

}
```

Ecosystem – Selenide

- Selenide is a Java-based open-source framework for UI test automation, built on top of Selenium WebDriver
  - It simplifies test writing by providing a more concise, fluent API and automating common tasks like auto waits
  - It aims writing automated tests in Java easier and more readable

https://selenide.org/

Ecosystem – Selenide

Fork me on GitHub

```java
class SelenideJupiterTest {

    @Test
    void testSelenide() {
        open("https://bonigarcia.dev/selenium-webdriver-java/login-form.html");

        $(By.id("username")).val("user");
        $(By.id("password")).val("user");
        $("button").pressEnter();
        $(By.id("success")).shouldBe(visible)
                .shouldHave(text("Login successful"));
    }

}
```

```xml
<dependency>
    <groupId>com.codeborne</groupId>
    <artifactId>selenide</artifactId>
    <version>7.10.1</version>
    <scope>test</scope>
</dependency>
```

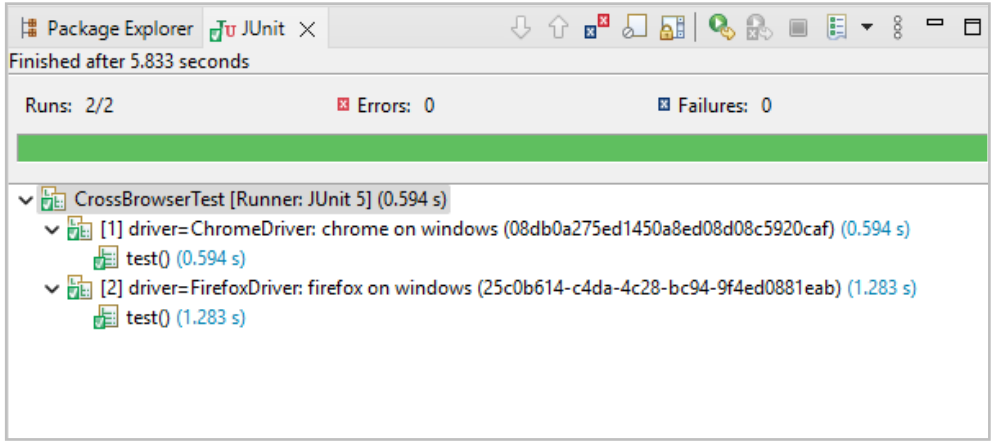
Ecosystem – Unit Testing Frameworks

Fork me on GitHub

```java
class CrossBrowserTest extends CrossBrowserParent {

    @Test
    void test() {
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/");
        assertThat(driver.getTitle()).contains("Selenium WebDriver");
    }

}
```

```java
public class CrossBrowserProvider implements ArgumentsProvider {

    @Override
    public Stream<? extends Arguments> provideArguments(
            ExtensionContext context) {
        ChromeDriver chrome = new ChromeDriver();
        FirefoxDriver firefox = new FirefoxDriver();

        return Stream.of(Arguments.of(chrome), Arguments.of(firefox));
    }

}
```

```java
@ParameterizedClass
@ArgumentsSource(CrossBrowserProvider.class)
class CrossBrowserParent {

    @Parameter
    WebDriver driver;

    @AfterEach
    void teardown() {
        driver.quit();
    }

}
```



JUnit 5

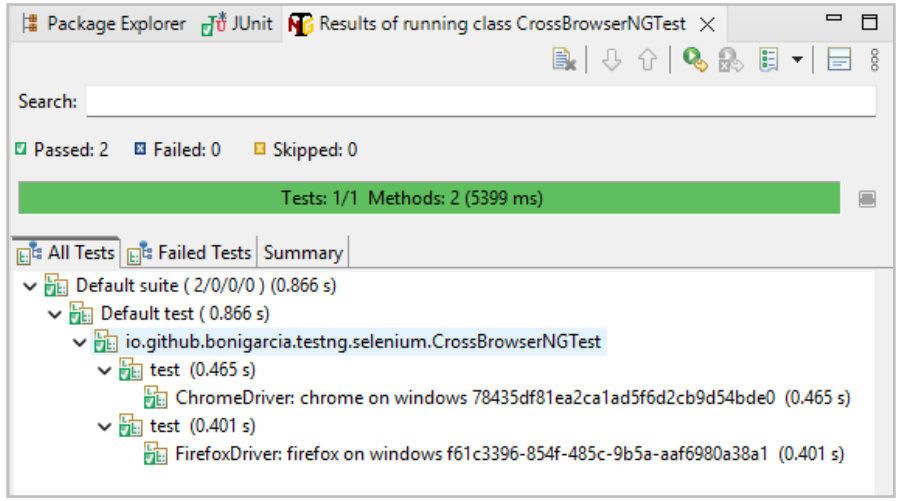
https://junit.org/

Ecosystem – Unit Testing Frameworks

Fork me on GitHub

```java
public class CrossBrowserNGTest extends CrossBrowserParent {

    @Test(dataProvider = "browserProvider")
    public void test(WebDriver driver) {
        this.driver = driver;

        // Test logic
    }

}
```

```java
public class CrossBrowserParent {

    WebDriver driver;

    @DataProvider(name = "browserProvider")
    public static Object[][] data() {
        ChromeDriver chrome = new ChromeDriver();
        FirefoxDriver firefox = new FirefoxDriver();

        return new Object[][] { { chrome }, { firefox } };
    }

    @AfterMethod
    void teardown() {
        driver.quit();
    }

}
```



TestNG

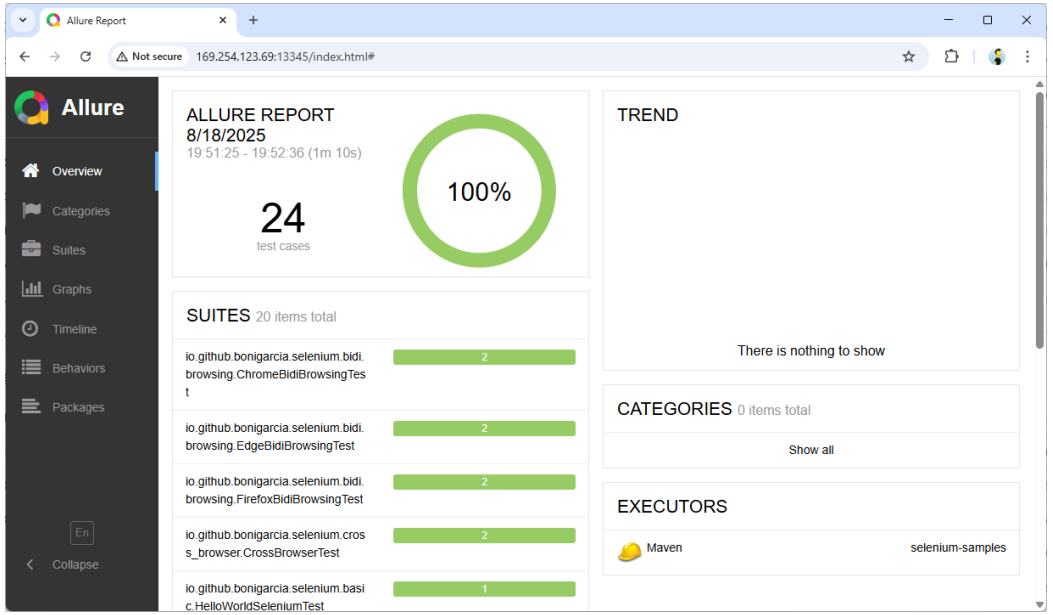
https://testng.org/

Ecosystem – Reporting

Fork me on GitHub

```
<dependencies>
   <dependency>
      <groupId>io.qameta.allure</groupId>
      <artifactId>allure-junit5</artifactId>
      <version>2.29.1</version>
      <scope>test</scope>
    </dependency>
</dependencies>
<build>
   <plugins>
      <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-surefire-plugin</artifactId>
         <version>3.5.3</version>
         <configuration>
            <properties>
               <property>
                  <name>listener</name>
                  <value>io.qameta.allure.junit5.AllureJunit5</value>
               </property>
            </properties>
          </configuration>
       </plugin>
       <plugin>
          <groupId>io.qameta.allure</groupId>
          <artifactId>allure-maven</artifactId>
          <version>2.15.2</version>
       </plugin>
    </plugins>
</build>
```

```
mvn test
mvn allure:report
mvn allure:serve
```



Allure Report

https://allurereport.org/

Ecosystem – Reporting

```
class ReportingJupiterTest {

    WebDriver driver;
    static ExtentReports reports;

    @BeforeAll
    static void setupClass() {
        reports = new ExtentReports();
        ExtentSparkReporter htmlReporter = new ExtentSparkReporter("extentReport.html");
        reports.attachReporter(htmlReporter);
    }

    @BeforeEach
    void setup(TestInfo testInfo) {
        reports.createTest(testInfo.getDisplayName());
        driver = new ChromeDriver();
    }

    @AfterEach
    void teardown() {
        driver.quit();
    }

    @AfterAll
    static void teardownClass() {
        reports.flush();
    }

    // Tests

}
```

Fork me on GitHub

EXTENT REPORT

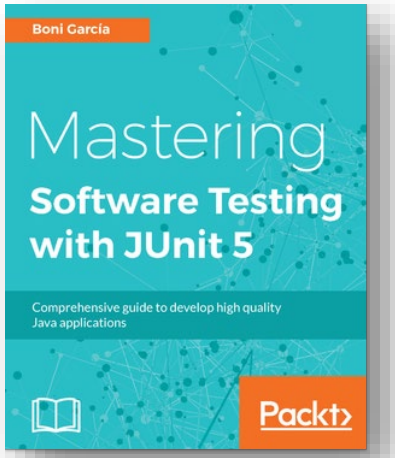
https://extentreports.com/

Ecosystem – Selenium-Jupiter

Selenium-Jupiter

“ JUnit extension for Selenium WebDriver

https://bonigarcia.dev/selenium-jupiter/



https://github.com/bonigarcia/mastering-junit5

Fork me on GitHub

Ecosystem – Selenium-Jupiter

- Selenium-Jupiter uses JUnit 5's **dependency injection**:

```
@ExtendWith(SeleniumJupiter.class)
class HelloWorldChromeSelJupTest {

    @Test
    void test(ChromeDriver driver) {
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/");
        assertThat(driver.getTitle()).contains("Selenium WebDriver");
    }

}
```

```
<dependency>
   <groupId>io.github.bonigarcia</groupId>
   <artifactId>selenium-jupiter</artifactId>
   <version>6.3.2</version>
   <scope>test</scope>
</dependency>
```

Ecosystem – Selenium-Jupiter

Fork me on GitHub

- Selenium-Jupiter also provides **Docker** support (e.g., for recording):

```
@EnabledIfDockerAvailable
@ExtendWith(SeleniumJupiter.class)
class DockerChromeRecordingSelJupTest {

    @Test
    void testDockerChromeRecording(
            @DockerBrowser(type = CHROME, recording = true) WebDriver driver) {
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/");
        assertThat(driver.getTitle()).contains("Selenium WebDriver");
    }

}
```

Fork me on GitHub

Ecosystem – Selenium-Jupiter

- Selenium-Jupiter use **test templates** for cross-browser testing:

```java
@EnabledIfDockerAvailable
@ExtendWith(SeleniumJupiter.class)
class CrossBrowserSelJupTest {

    @TestTemplate
    void testCrossBrowser(WebDriver driver) {
        driver.get("https://bonigarcia.dev/selenium-webdriver-java/");
        assertThat(driver.getTitle()).contains("Selenium WebDriver");
    }

}
```

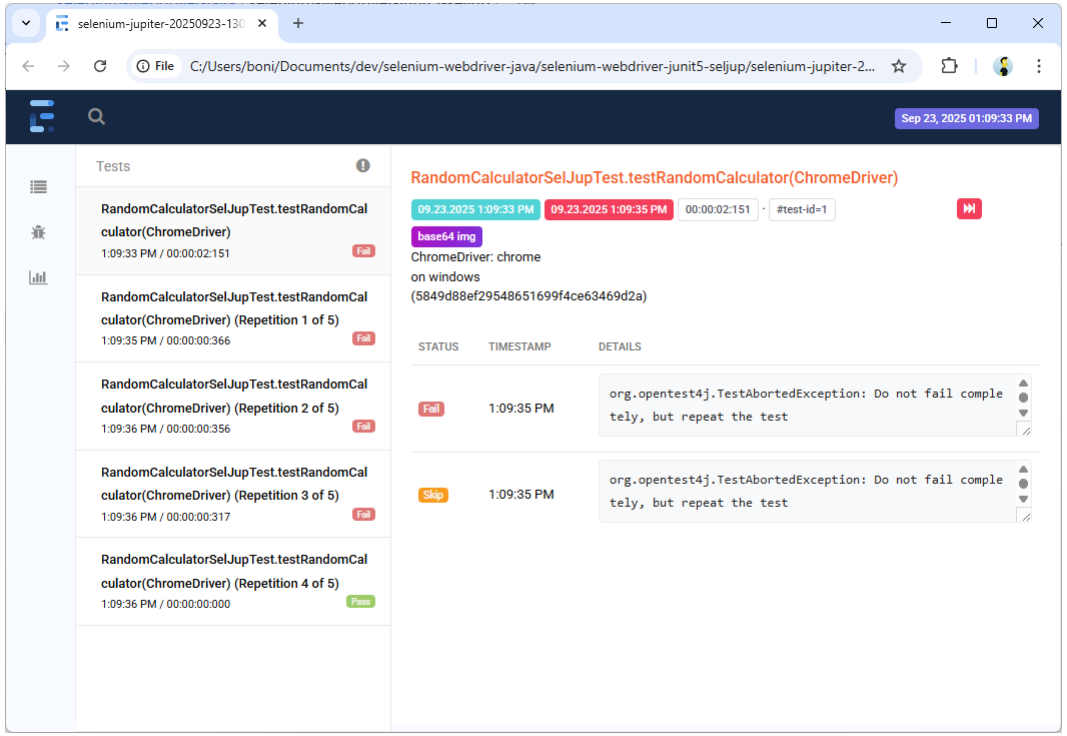
```
Runs: 3/3    Errors: 0    Failures: 0

CrossBrowserSelJupTest [Runner: JUnit 5] (3.077 s)
  testCrossBrowser(WebDriver) (3.077 s)
    [Browser [type=chrome]] (3.077 s)
    [Browser [type=edge]] (1.798 s)
    [Browser [type=firefox-in-docker, version=beta]] (17.436 s)
```

```json
{
   "browsers": [
      [
         {
            "type": "chrome"
         }
      ],
      [
         {
            "type": "edge",
             "arguments" : [
                "--headless"
             ]
         }
      ],
      [
         {
            "type": "firefox-in-docker",
            "version": "beta"
         }
      ]
   ]
}
```

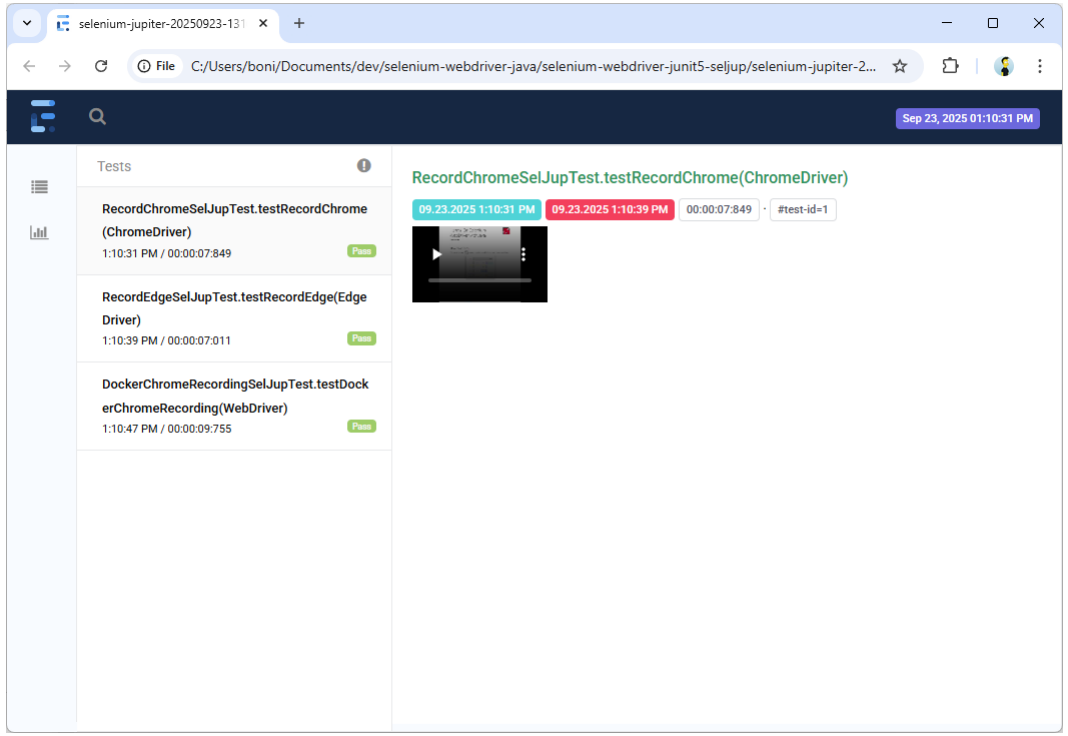
Fork me on GitHub

Ecosystem – Selenium-Jupiter

- As of version 6.3.0, Selenium-Jupiter provides built-in reporting capabilities through ExtentReports



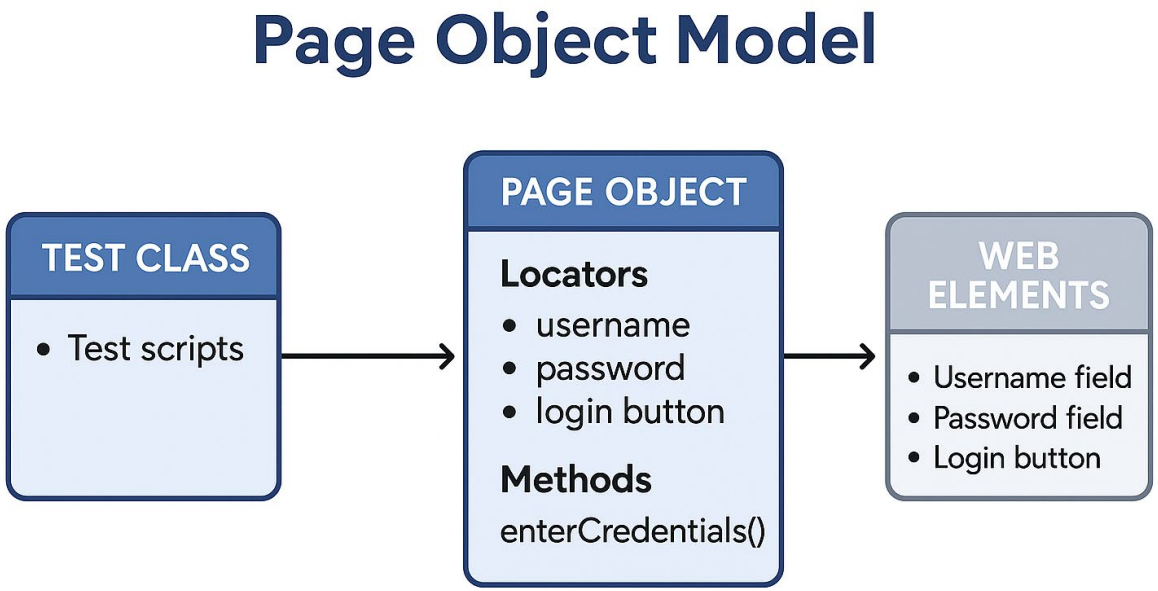

```
mvn test -Dtest=Random*
```




```
mvn test -Dtest=*Record*
```

Design patterns – POM

- **Page Object Model (POM)** is a design pattern for test automation that help us in making reusable and maintainable code



Design patterns – POM

Fork me on GitHub

```java
class ExtendedLoginJupiterTest {

    ExtendedLoginPage login;

    @BeforeEach
    void setup() {
        login = new ExtendedLoginPage("chrome");
    }

    @AfterEach
    void teardown() {
        login.quit();
    }

    @Test
    void testLoginSuccess() {
        login.with("user", "user");
        assertThat(login.successBoxPresent()).isTrue();
    }

    @Test
    void testLoginFailure() {
        login.with("bad-user", "bad-password");
        assertThat(login.successBoxPresent()).isFalse();
    }

}
```

```java
public class ExtendedLoginPage extends ExtendedBasePage {

    By usernameInput = By.id("username");
    By passwordInput = By.id("password");
    By submitButton = By.cssSelector("button");
    By successBox = By.id("success");

    public ExtendedLoginPage(String browser, int timeoutSec) {
        this(browser);
        setTimeoutSec(timeoutSec);
    }

    public ExtendedLoginPage(String browser) {
        super(browser);
        visit("https://bonigarcia.dev/selenium-webdriver-java/login-form.html");
    }

    public void with(String username, String password) {
        type(usernameInput, username);
        type(passwordInput, password);
        click(submitButton);
    }

    public boolean successBoxPresent() {
        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(2));
        try {
            WebElement success = wait.until(
                    ExpectedConditions.visibilityOfElementLocated(successBox));
            return success.isDisplayed();
        } catch (TimeoutException e) {
            return false;
        }
    }
}
```

Conclusions

- Selenium is a browser automation library, not a testing framework
- The Selenium Java ecosystem is very rich for E2E testing:
  - JUnit/TestNG: unit testing framework
  - WebDriverManager: automated driver management and other features
  - Selenide: E2E testing framework providing a fluent API for Selenium
  - Selenium-Jupiter: JUnit extension to reduce the test boilerplate
- POM is a convenient design pattern for reusability and maintainability
- Find open-source examples in:
  - https://github.com/bonigarcia/selenium-webdriver-java
  - https://github.com/bonigarcia/selenium-examples
  - https://github.com/bonigarcia/browser-automation-apis/

Selenium for Java Developers

Thank you so much!

Boni García
boni.garcia@uc3m.es

Get these slides on:

https://bonigarcia.dev/