**Getting Started with Selenium, TestNG, and a Web Driver in IntelliJ**

**Author: Truong Long**

1. **Introduction**

Selenium is a popular open-source tool for automating web browsers. It allows you to write tests that can interact with web pages and perform actions such as clicking buttons, filling out forms, and verifying that certain elements are present or have certain values.

TestNG is a testing framework that can be used with Selenium to organize and run your tests. It provides features such as parallel execution, test dependency management, and flexible reporting.

Web drivers are libraries that allow Selenium to interact with specific web browsers, such as Chrome or Edge. In order to run Selenium tests on a particular browser, you will need to install and configure the appropriate web driver.

In this guide, I will walk you through the steps of installing and setting up Selenium, TestNG, and a web driver in IntelliJ, a popular Java development environment. This guide is intended for those who are new to Selenium and are looking to get started with the tool. By the end of this guide, you will have a basic Selenium test running in IntelliJ.

1. **Prerequisites**

Before you can start using Selenium, TestNG, and a web driver in IntelliJ, you will need to make sure that you have the following prerequisites installed on your machine:

Java Development Kit (JDK): Selenium is written in Java, so you will need to have the JDK installed in order to use it. You can download the latest version of the JDK from the Oracle website (https://www.oracle.com/java/technologies/javase-downloads.html). Make sure to choose the correct version for your operating system.

IntelliJ: IntelliJ is a popular Java development environment that you can use to write and run Selenium tests. You can download the latest version of IntelliJ from the JetBrains website (https://www.jetbrains.com/idea/download/). Make sure to choose the version that is appropriate for your operating system.

Once you have these prerequisites installed, you are ready to proceed with installing Selenium and TestNG in IntelliJ.

1. **Installing Selenium and TestNG in IntelliJ**
2. **Installation of Selenium and TestNG:**
3. Open IntelliJ and create a new project.
4. In the "Project" pane on the left side of the screen, right-click on the name of your project and select "New > Module."
5. In the "New Module" window, select "Java" and click "Next."
6. In the "Project Location" field, specify a location for your project on your local machine. Click "Finish" to create the project.
7. In the "Project" pane, right-click on the "lib" folder and select "New > Directory."
8. Name the new directory "dependencies" and click "OK." This is where you will store the Selenium and TestNG JAR files.
9. Download the Selenium and TestNG JAR files from the following locations:

* Selenium: <https://www.selenium.dev/downloads/>
* TestNG: <https://testng.org/doc/download.html>

1. Save the JAR files to the "dependencies" directory that you created in step 6.
2. In the "Project" pane, right-click on the name of your project and select "Open Module Settings."
3. In the "Project Settings" window, select the "Modules" tab.
4. Click the "+" button and select "JARs or directories."
5. In the "Choose File or Directory" window, navigate to the "dependencies" directory and select the Selenium and TestNG JAR files. Click "OK."
6. Click "Apply" and then "OK" to close the "Project Settings" window.

Your project is now set up to use Selenium and TestNG. In the next section, we will show you how to configure TestNG in IntelliJ.

1. **TestNG configuration:**
2. In the "Project" pane, right-click on the name of your project and select "New > TestNG Class."
3. In the "Create TestNG Class" window, specify a package name and a class name for your test class. Click "OK."
4. A basic TestNG test class template will be generated. Replace the contents of the class with the following code:

import org.testng.annotations.Test;

public class MyTest {

@Test

public void test() {

// write your test code here

}

}

1. Right-click on the test class and select "Run 'MyTest'."
2. A "Run TestNG Test" window will appear. Make sure that the "MyTest" class is selected, and then click "OK."
3. The test will run and a results window will appear. If the test passes, you will see a green "PASSED" message. If the test fails, you will see a red "FAILED" message.

Your TestNG configuration is now complete. You can use this basic test template to write and run your Selenium tests. In the next section, we will show you how to set up a web driver so that you can run your tests in a specific browser.

1. **Setting up a Web Driver**
2. To use Selenium to automate a web browser, you will need to install and configure the appropriate web driver. There are web drivers available for various browsers, including Chrome and Edge. In this guide, we will show you how to set up the Chrome driver, but the process is similar for other drivers as well.
3. Download the Chrome driver from the following location: <https://chromedriver.chromium.org/downloads>. Make sure to choose the version of the Chrome driver that matches the version of Chrome installed on your machine.
4. Save the Chrome driver executable file to a location on your local machine. For example, you could create a new directory called "webdrivers" in your project directory and save the Chrome driver executable file there.
5. In your Selenium test class, add the following import statement at the top of the file:

import org.openqa.selenium.chrome.ChromeDriver;

1. In the **@BeforeMethod** or **@BeforeClass** annotation of your test class, add the following code to set up the Chrome driver:

System.setProperty("webdriver.chrome.driver", "/path/to/chromedriver");

WebDriver driver = new ChromeDriver();

Make sure to replace **/path/to/chromedriver** with the actual path to the Chrome driver executable file on your local machine. For example, if you saved the Chrome driver executable file in the "webdrivers" directory in your project directory, the path would be **"webdrivers/chromedriver"**.

1. You can now use the **driver** object to interact with the Chrome browser in your Selenium tests.

For example, you can use the **driver.get()** method to open a URL in the browser, or the **driver.findElement()** method to locate an element on the page and perform an action on it.

Here is an example of a simple Selenium test that opens the Google homepage and searches for a keyword:

@Test

public void test() {

driver.get("https://www.google.com");

WebElement searchBox = driver.findElement(By.name("q"));

searchBox.sendKeys("selenium");

searchBox.submit();

}

1. When you are finished with your tests, be sure to close the browser window by calling the **driver.quit()** method. This is important to ensure that the browser is properly closed and any resources are released.

Your web driver is now set up and ready to use with Selenium. In the next section, we will show you how to create and run your first Selenium test.

1. **Running Your First Selenium Test**
2. In your Selenium test class, create a new **@Test** method. This is where you will write the code for your Selenium test.
3. Use the **driver** object and Selenium's API to interact with the web page and perform the actions that you want to test. For example, you might want to navigate to a particular URL, fill out a form, click a button, or verify that certain elements are present or have certain values.

Here is an example of a simple Selenium test that navigates to the Google homepage, searches for a keyword, and verifies that the search results page is displayed:

@Test

public void test() {

// navigate to the Google homepage

driver.get("https://www.google.com");

// enter a search query and submit the form

WebElement searchBox = driver.findElement(By.name("q"));

searchBox.sendKeys("selenium");

searchBox.submit();

// verify that the search results page is displayed

String pageTitle = driver.getTitle();

Assert.assertEquals(pageTitle, "selenium - Google Search");

}

1. To run your Selenium test, right-click on the test class and select "Run 'MyTest'."
2. A "Run TestNG Test" window will appear. Make sure that the "MyTest" class is selected, and then click "OK."
3. The test will run and a results window will appear. If the test passes, you will see a green "PASSED" message. If the test fails, you will see a red "FAILED" message, along with an error message indicating what went wrong.

Congratulations, you have successfully run your first Selenium test! You can now use Selenium to automate web browser interactions and test the functionality of your web applications.

1. **Conclusion**

In this guide, I have shown you how to install and set up Selenium, TestNG, and a web driver in IntelliJ. You should now have a basic Selenium test running in the development environment, and you are ready to start exploring the full capabilities of the tool.

Here are a few tips for troubleshooting and further learning:

* If you encounter any errors or issues while following this guide, make sure to check the error messages and do some research to find a solution. There are many resources available online, such as Stack Overflow and the Selenium documentation, that can help you troubleshoot problems.
* To learn more about Selenium and how to use it effectively, you can explore the Selenium doc: <https://www.selenium.dev/documentation/>. And try out some of the examples provided. You can also consider taking an online course or attending a workshop to gain more in-depth knowledge of Selenium and web automation testing.
* As you start writing more complex tests and automating larger portions of your web applications, you may find it helpful to use additional tools and libraries to make your work easier and more efficient. Some popular options include frameworks like Selenide:<https://selenide.org/.> And Page Object Model: <https://www.selenium.dev/documentation/en/guidelines_and_recommendations/page_object_models/>.

I hope that this guide has helped you get started with Selenium, TestNG, and web drivers in IntelliJ. Wish you the best of luck in your automation testing endeavors!