Step-by-Step Selenium Setup

(Using IntelliJ IDEA + Maven + Java)

&

Launching Chrome Browser in 3 Different Ways

🛠️ Why Start with Selenium Setup?

Before writing any automation tests, it's essential to correctly set up the Selenium environment. This is one of the most searched and required steps for beginners in automation. A smooth setup ensures a better learning experience.

📌 Topics Covered

- Introduction to Automation
- Challenges in Manual Testing
- Why Automation is the Solution
- Importance of Programming in Automation
- Introduction to Selenium

- Step-by-step Selenium setup
- Launching Chrome browser using 3 different methods
- Selenium Components
- Advantages and Disadvantages of Selenium WebDriver

💡 Introduction to Automation

Automation testing involves using tools to execute test cases automatically, validate functionality, and reduce manual effort. It enhances speed, accuracy, and repeatability in testing processes.

📦 Challenges in Manual Testing

- Time-consuming for repetitive tasks
- Prone to human errors
- Difficult to scale across different environments
- Poor regression test coverage in large systems

⚙️ How Automation Solves These Challenges

- Faster execution of test cases
- Reusable test scripts
- More test coverage in less time
- Enables continuous testing in DevOps

👦💻 Role of Programming in Automation

Understanding programming (e.g. Java, Python) is essential in automation. It helps in:

- Writing custom test scripts
- Implementing logic and decision-making
- Debugging and maintaining test cases
- Integrating tools like Selenium, TestNG, Maven, etc.

🌐 Introduction to Selenium

Selenium is a widely used open-source tool for automating web applications. It supports multiple browsers, platforms, and programming languages.

🔧 Step-by-Step Selenium Setup

✅ 1. Install Java JDK

Download and install Java Development Kit (JDK) from the Oracle JDK site.

Configure environment variables:

- JAVA_HOME → Path to your JDK folder
- Add JAVA_HOME/bin to your system PATH
-

✅ 2. Install IntelliJ IDEA

Download IntelliJ IDEA (Community or Ultimate) from JetBrains.

Launch the IDE and follow the initial setup wizard.

✅ 3. Create a New Maven Project

1. Open IntelliJ IDEA → New Project

2. Select Maven → Click Next
3. Provide GroupId (e.g., com.selenium.learn) and ArtifactId (e.g., FirstSeleniumProject)
4. Finish the setup and open the project.

✅ 4. Add Selenium & WebDriverManager Dependencies

5. Open the `pom.xml` file and add the following inside `<dependencies>`:

```
<dependencies>
  <!-- Selenium Java -->
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.19.1</version>
  </dependency>

  <!-- WebDriverManager -->
  <dependency>
    <groupId>io.github.bonigarcia</groupId>
    <artifactId>webdrivermanager</artifactId>
```

```
    <version>5.7.0</version>
  </dependency>
</dependencies>
```

After adding, click "Load Maven Changes" in the top-right corner.

✅ 5. Create a Test Class

Right-click on src/main/java → New > Java Class
Name it LaunchBrowser

Add the sample code to launch the browser (see next section)

✅ 6. Run the Test

Right-click on the class → Run 'LaunchBrowser.main()' The Chrome browser should launch as configured.

🚀 Launching Chrome Browser – 3 Methods ✔ Method 1 – Using System.setProperty()

```
System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
WebDriver driver = new ChromeDriver();
```

- ChromeDriver must be downloaded manually
- Requires managing driver versions manually
- Less scalable and error-prone in the long run

✔ Method 2 – Using WebDriverManager

WebDriverManager.chromedriver().setup();

WebDriver driver = new ChromeDriver();

- Automatically handles driver versions
- Cleaner and more efficient setup
- Requires internet access to download drivers

✔ Method 3 – Relying on PATH Configuration

WebDriver driver = new ChromeDriver();

- Most simplified method
- Works when ChromeDriver is already added to the system PATH
- No additional configuration needed during test execution

📌 Recommended Approach

This method is considered the best for small to mid-sized projects where the driver setup is already managed. It avoids manual property settings and keeps the code clean.

🔍 Summary of Launch Methods

Method	Setup Required	Recommendation
System.setProperty()	Manual driver path setup	Not ideal
WebDriverManager setup	Auto-handled driver	Good for flexibility
Direct new ChromeDriver()	PATH must be set	Best for simplicity

⚙️ Selenium Components (Quick Overview)

Component	Description
Selenium WebDriver	Core API for browser automation
Selenium IDE	Record and playback tool

Selenium Grid	Distributed execution of tests across environments

✅ Pros and Cons of Selenium WebDriver

Advantages	Disadvantages
Free and open-source	Requires programming knowledge
Supports multiple languages and browsers	No built-in reporting or dashboards
Integrates with testing frameworks	Cannot automate desktop or mobile apps

📌 Conclusion

Selenium setup is the foundation of any automation journey. Launching Chrome using the direct new ChromeDriver() method is the most efficient and beginner-friendly way — when the system PATH is correctly configured.

Once the environment is set, creating, running, and scaling test cases become much easier. This setup empowers testers to move forward confidently into real-time browser automation.

Now that the setup is complete, the next step is interacting with web elements—identifying buttons, input fields, checkboxes, and links. The upcoming blog will cover Finding Web Elements & Working with Links, exploring basic advanced techniques to handle different types of web elements in Selenium.

Stay tuned for the next article in this Selenium series!