|  |
| --- |
| Here are the required items and configurations for the new automation VM:   1. [**Java Development Kit**](https://www.oracle.com/java/technologies/downloads/#jdk21-windows)  [**NOT** the Java Runtime; Maven requires **JDK** to compile sources] – we are using version **21**. 2. [**Apache Maven**](https://maven.apache.org/download.cgi) – we are using version **3.9.6**. You can get the latest stable version Please ensure the archive contents are extracted to **C:\Program Files\Apache\Maven**. 3. The following **System** Environment Variables:    1. **JAVA\_HOME** – value should be the JDK installation path [**C:\Program Files\Java\jdk-21**]    2. **M2\_HOME** – value should be the Apache Maven extraction path [**C:\Program Files\Apache\Maven**]    3. **M2** – value should be **%M2\_HOME%\bin**    4. **MAVEN\_OPTS** – value should be **-Xms256m -Xmx512m**    5. The **Path** variable should contain entries for:       1. **C:\Program Files\Common Files\Oracle\Java\javapath**       2. **%JAVA\_HOME%\bin**       3. **%M2%** 4. The following command should be executed in the directory where **\test\se** is extracted during deployment[s]: **mvn test -DxmlFilePath="src/test/java/smokeTest/smokeTest.xml"**   IntelliJ IDEA Community Edition  <https://www.jetbrains.com/idea/download/download-thanks.html?platform=windows&code=IIC>  Java Development Kit (21)  <https://www.oracle.com/java/technologies/downloads/#jdk21-windows>  Maven  https://stackoverflow.com/a/38549868/22396150  <https://maven.apache.org/download.cgi> **Get the Binary Zip Archive file.**  Installed java in C:\Program Files\Java\jdk-21  Next installed IntelliJ  Using admin privileges on ideaIC-2025.1.exe  Uninstall old version silently keeping configs checked  Installing to C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.1    **NOTE: Don’t need Visual Studio as IntelliJ covers all you need there. IntelliJ wraps around Git** |

Adding the **path to the environment variable** in Selenium automation (especially in **Java with Selenium WebDriver**) is done to make the browser **driver executable (e.g., chromedriver.exe)** accessible system-wide.

What is the Browser Driver?

Selenium doesn’t directly control browsers. It uses **browser-specific drivers** (like chromedriver, geckodriver, etc.) as a **bridge** to interact with the browser.

Why Add Driver Path to Environment Variables?

By adding the path to the **PATH environment variable**, the OS can **locate the driver executable** **from anywhere**, without you specifying the full path in your code every time.

✅ **Benefits:**

1. **Cleaner code** – You don’t have to hardcode full paths:

System.setProperty("webdriver.chrome.driver", "C:\\drivers\\chromedriver.exe");becomes unnecessary.

1. **Portability** – The code works on any system where the path is set.
2. **Avoids file path issues** when moving between machines or projects.

Without Setting the Path?

If you **don’t set it**, you **must** write:

System.setProperty("webdriver.chrome.driver", "C:\\WebDrivers\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

After installing the **Java Development Kit (JDK)** and **Apache Maven**, you need to **set up your environment properly** and then you can **create, build, and run Selenium Java projects** using Maven. Here's a precise step-by-step guide:

Example:

**🔹 Step 1: Set Environment Variables**

**✅ For JDK:**

1. Copy your JDK path (e.g., C:\Program Files\Java\jdk-21\bin)
2. Add it to Environment Variables → System variables → Path
3. Also, set a new variable:
   * Name: JAVA\_HOME
   * Value: C:\Program Files\Java\jdk-21

**✅ For Maven:**

1. Copy your Maven bin path (e.g., C:\apache-maven-3.9.6\bin)
2. Add it to the system Path variable
3. Set another variable:
   * Name: MAVEN\_HOME
   * Value: C:\apache-maven-3.9.6

✅ **System Environment Variables – Detailed Explanation**

🔹 a. JAVA\_HOME = C:\Program Files\Java\jdk-21

**Purpose:**

* Tells tools (Maven, IDEs like IntelliJ, Eclipse, Jenkins) where **Java is installed**.

**Effect:**

* Maven uses this to find java.exe
* Useful for scripts that rely on %JAVA\_HOME%

🔹 b. M2\_HOME = C:\Program Files\Apache\Maven

**Purpose:**

* Specifies the **Maven installation directory**.

**Effect:**

* Some older tools or scripts use M2\_HOME to locate Maven files.
* Helpful when configuring Maven in IDEs manually.

🔹 c. M2 = %M2\_HOME%\bin

**Purpose:**

* Points to the **bin directory** of Maven, where the mvn command lives.

**Effect:**

* %M2% becomes shorthand for C:\Program Files\Apache\Maven\bin
* Can be added to Path for quick access to mvn in terminal.

🔹 d. MAVEN\_OPTS = -Xms256m -Xmx512m

**Purpose:**

* Sets **Java heap memory limits** when Maven runs.

**Effect:**

* Prevents memory errors in large builds.
* Useful when compiling or running large test suites via Maven.

✅ **Path Variable Entries:**

i. C:\Program Files\Common Files\Oracle\Java\javapath

**Purpose:**

* System default for Java launcher (used by Windows).

**Purpose:**

* System default for Java launcher (used by Windows).

**Effect:**

* Lets you use java, javac, etc., from the command prompt.
* Safe to keep, but not mandatory if %JAVA\_HOME%\bin is present.

ii. %JAVA\_HOME%\bin

**Purpose:**

* Adds the bin folder of your installed JDK to the path.

**Effect:**

* Lets you run java, javac, etc., from **any folder** in Command Prompt.
* Preferred over the Oracle path above because it is **explicit**.

iii. %M2%

**Purpose:**

* Adds Maven's bin folder to your system path.

**Effect:**

* Lets you run mvn command globally in terminal.

**✅ What Will Happen if You Set All These Correctly**

* ✅ You can run java, javac, and mvn from anywhere in Command Prompt
* ✅ Maven will locate the correct JDK automatically via JAVA\_HOME
* ✅ Builds and tests run faster and more reliably due to MAVEN\_OPTS
* ✅ Tools like Jenkins, IntelliJ, Eclipse will auto-detect Java and Maven
* ✅ You won’t face errors like:
  + JAVA\_HOME is not set
  + 'mvn' is not recognized as an internal or external command

**Difference between Eclipse and IntelliJ IDEA** when it comes to Maven environment variables like M2, M2\_HOME, and MAVEN\_OPTS for Selenium projects.

**✅ Summary Answer**

No, it's **not mandatory** to set M2, M2\_HOME, or MAVEN\_OPTS to run Selenium code with Maven **in Eclipse**, but **IntelliJ may require or recommend** setting them explicitly, especially in manual configurations or larger builds.

**💡 Why It Works in Eclipse Without M2 or MAVEN\_OPTS**

* **Eclipse has built-in Maven support (M2E plugin)**.
* It **manages Maven internally** and doesn’t rely on system-level environment variables.
* As long as your pom.xml is correct and Maven is bundled or set inside Eclipse preferences, your Selenium dependencies work fine.

**❗ Why IntelliJ Suggests Setting M2\_HOME and MAVEN\_OPTS**

1. **IntelliJ Uses External Maven (Not bundled like Eclipse)**
   * It expects to find Maven in a system-defined location.
   * If M2\_HOME and %M2\_HOME%\bin aren't in your system’s environment, IntelliJ can't find Maven unless you configure it manually.
2. **Custom Memory Options with MAVEN\_OPTS**
   * IntelliJ suggests setting this for performance: **“MAVEN\_OPTS=-Xms256m -Xmx512m”**
   * Especially helpful for big projects or running tests with many Selenium drivers or parallel threads.
3. **IntelliJ → Settings → Build Tools → Maven**
   * It may show warnings like:  
     *"Maven home directory not found"*  
     until you:
     + Set M2\_HOME or
     + Manually specify Maven installation in settings

**🛠️ How to Fix or Avoid the Need (If You Prefer Not to Set Env Vars)**

**Option 1: Configure Maven Manually in IntelliJ**

* Go to:  
  File > Settings > Build, Execution, Deployment > Build Tools > Maven
* Set **Maven home directory** manually (e.g., C:\Program Files\Apache\Maven)
* Set **Java SDK** under Project Structure > Project > SDK

**Option 2: Set Environment Variables (Preferred for Reusability)**

Set these permanently:

M2\_HOME = C:\Program Files\Apache\Maven

M2 = %M2\_HOME%\bin

MAVEN\_OPTS = -Xms256m -Xmx512m

Add %M2% to Path

Then restart IntelliJ (and your machine, if needed)

JAvaDOC

**Javadoc URL for Selenium**

You're referring to this Javadoc URL for Selenium:

ruby

CopyEdit

https://javadoc.io/doc/org.seleniumhq.selenium/selenium-java/4.33.0/

This URL is used **for Selenium**, and can be attached to Selenium-related libraries.  
It **should not be attached to ExtentReports**, because ExtentReports is a **separate library**.

**✅ What You Can Do:**

**✔️ 1. Attach Selenium Javadoc to Selenium libraries only**

If you want to attach the **Selenium Javadoc** to your selenium-java dependency:

1. Go to File → Project Structure → Libraries
2. Select the Selenium-related library, like:

makefile

CopyEdit

Maven: org.seleniumhq.selenium:selenium-java:4.33.0

1. Click the **JavaDocs** tab
2. Click + → **Specify URL**
3. Paste:

ruby

CopyEdit

https://javadoc.io/doc/org.seleniumhq.selenium/selenium-java/4.33.0/

1. Click OK and Apply ✅

**✔️ 2. (Optional) Attach ExtentReports Javadoc (if available)**

If you're working with:

text

C:\Users\oraj\.m2\repository\com\aventstack\extentreports\5.1.2\extentreports-5.1.2.jar

And want to attach Javadoc:

* Visit: <https://search.maven.org>
* Search: extentreports 5.1.2
* Look for the **Javadoc** .jar (sometimes it's not provided)
* If it exists:
  + Download it
  + Go to: File → Project Structure → Libraries
  + Select extentreports
  + JavaDocs tab → + → Attach downloaded .jar

If **no Javadoc is released**, IntelliJ will show a message like:

“No Javadoc found for this library.”