**🎯 TASK 8: Run a Simple Java Maven Build Job in Jenkins**

Build a Java HelloWorld program with Maven via Jenkins (CI/CD basics)

**🛠 Tools Needed:**

✅ Jenkins  
✅ Java JDK 8 or 11  
✅ Maven  
✅ (Optional) Git  
✅ Command Prompt / Terminal

**📂 Deliverables:**

* ✅ Java HelloWorld App (HelloWorld.java)
* ✅ pom.xml
* ✅ Jenkins Freestyle Job to build it
* ✅ Screenshot of successful BUILD SUCCESS

**📖 Full Minute Step-by-Step Guide**

**🔹 Step 1: Install Prerequisites**

✅ **Install JDK 8 or 11**  
Check with:

nginx

Copy code

java -version

✅ **Download and unzip Maven** (not installer — use ZIP)  
👉 <https://maven.apache.org/download.cgi>  
👉 Extract to:

makefile

Copy code

C:\Tools\apache-maven-3.9.10

✅ **Add Maven to System PATH**

1. Right-click **This PC → Properties → Advanced system settings**
2. Click **Environment Variables**
3. Under **System variables** → Find Path → Edit
4. Add:

makefile

Copy code

C:\Tools\apache-maven-3.9.10\bin

1. Click OK everywhere.

✅ **Check Maven install:**

nginx

Copy code

mvn -v

You should see Maven version info ✅

**🔹 Step 2: Create Java Maven Project**

👉 Open Command Prompt  
👉 Create folder:

makefile

Copy code

C:\Users\Raichal\Projects\hello-java-maven

👉 Inside it, create:

css

Copy code

src\main\java

👉 Inside src\main\java, create file HelloWorld.java  
📜 Code:

java

Copy code

public class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, Jenkins + Maven!");

}

}

👉 In C:\Users\Raichal\Projects\hello-java-maven create pom.xml  
📜 Code:

xml

Copy code

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>hello</artifactId>

<version>1.0</version>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**🔹 Step 3: Run Maven Build Locally (test once)**

Go to:

bash

Copy code

cd C:\Users\Raichal\Projects\hello-java-maven

Run:

go

Copy code

mvn clean package

✅ You should see BUILD SUCCESS

**🔹 Step 4: Start Jenkins**

👉 From Docker:

arduino

Copy code

docker run -p 8080:8080 jenkins/jenkins:lts

Or from Windows Service if installed natively.

**🔹 Step 5: Add Maven in Jenkins**

* Go to Jenkins UI → http://localhost:8080/
* Manage Jenkins → **Global Tool Configuration**
* Scroll down to **Maven**
* Click **Add Maven**
  + Name: Maven-3.9.10
  + Uncheck Install automatically
  + Enter:

makefile

Copy code

C:\Tools\apache-maven-3.9.10

* Save

**🔹 Step 6: Create Jenkins Freestyle Job**

* Dashboard → New Item
* Enter Name: hello-java-job
* Choose **Freestyle project**
* OK

**🔹 Step 7: Configure Source Code / Workspace**

**If using local folder**:

* Scroll to **Build Environment**
* Check **Use custom workspace**
* Enter:

makefile

Copy code

C:\Users\Raichal\Projects\hello-java-maven

**🔹 Step 8: Add Build Step**

* Scroll to **Build**
* Click **Add build step**
* Choose **Invoke top-level Maven targets**
* **Goals:**

go

Copy code

clean package

* In **Advanced**:
  + **POM**:

makefile

Copy code

C:\Users\Raichal\Projects\hello-java-maven\pom.xml

**🔹 Step 9: Save & Build**

* Click **Save**
* Click **Build Now**

**🔹 Step 10: Check Console Output**

* Click on build number (#1)
* Console Output

✅ You should see:

csharp

Copy code

[INFO] BUILD SUCCESS

**📸 Step 11: Take Screenshot**

Take a clear screenshot of that console output with BUILD SUCCESS

**✅ Recap:**

| **Step** | **What You Did** |
| --- | --- |
| Install JDK, Maven, Jenkins | Prerequisites |
| Created hello-java-maven folder | Java app & pom |
| Verified mvn clean package locally | Tested build |
| Configured Jenkins + Maven tool | Setup CI |
| Created Freestyle project | Job config |
| Set local workspace & pom.xml | Build settings |
| Ran Jenkins Build | Checked console |
| Took screenshot | Deliverable |

**📌 Interview Questions & Answers:**

**1️⃣ What is Jenkins?**

**Answer:**  
Jenkins is an open-source **automation server** used for **continuous integration and continuous delivery (CI/CD)**.  
It helps developers automatically build, test, and deploy applications by integrating with various tools like Git, Maven, Docker, etc.

**Why it's useful:**

* Automates software development workflows
* Detects issues early via automated builds and tests
* Supports plugins for extending functionality

**2️⃣ How do you create a Jenkins job?**

**Answer:**  
To create a Jenkins job:

1. Go to Jenkins Dashboard
2. Click **New Item**
3. Enter a job name
4. Choose a job type (like **Freestyle project**, **Pipeline**, etc.)
5. Click **OK**
6. Configure job details (source code management, build steps, post-build actions)
7. Click **Save**
8. Click **Build Now** to run it

**3️⃣ What is Maven used for?**

**Answer:**  
Maven is a **build automation and project management tool** for Java applications.  
It simplifies compiling, testing, packaging, and managing project dependencies through a **pom.xml** file.

**Key features:**

* Handles project builds
* Manages dependencies
* Defines project structure and configurations
* Automates tasks like creating JAR/WAR files

**4️⃣ How does Jenkins use build tools like Maven?**

**Answer:**  
Jenkins integrates with build tools like Maven through **build steps** inside Jenkins jobs.

In a Jenkins job:

* You can add a **Maven build step**
* Specify Maven goals like clean install or clean package
* Jenkins then triggers Maven to execute those goals in the job’s workspace
* The output (success or failure) is captured in the **Console Output**

**5️⃣ What is the difference between compile and package in Maven?**

| **Compile** | **Package** |
| --- | --- |
| Compiles source code into .class files | Compiles **and bundles** compiled files into a distributable format like a .jar or .war |
| Maven command: mvn compile | Maven command: mvn package |
| Doesn't create a final archive | Creates final output for deployment |

**6️⃣ Where do you configure tools in Jenkins?**

**Answer:**  
In Jenkins:

1. Go to **Manage Jenkins** (from Dashboard)
2. Click **Global Tool Configuration**
3. Configure tools like:
   * JDK
   * Maven
   * Git
   * Gradle
4. Set installation paths or auto-install options
5. Click **Save**

**7️⃣ How do you debug a failed Jenkins build?**

**Answer:**  
To debug:

1. **Check Console Output** — see where it failed
2. **Verify build steps** — check commands or goals used
3. **Ensure correct tool configurations** — JDK, Maven, etc.
4. **Check workspace permissions and paths**
5. **Use Maven debug options** if needed:

go

Copy code

mvn clean package -X

1. **Test the build manually in the local terminal** — to isolate whether it’s a code issue or a Jenkins config issue
2. **Check Jenkins system logs** (Manage Jenkins → System Log)