SOFTWARE ENGINEERING

UNIT – 3

TOPIC – 9

BUILD TOOLS: INTRODUCTION TO MAVEN

1. Understanding Maven

Definition: Maven is a tool used for managing and building software projects, mainly in Java. It automates key tasks such as compiling code, running tests, and packaging the final software. Maven also manages dependencies by automatically downloading and organizing external libraries that your project needs.

In simple terms, Maven helps developers by simplifying repetitive tasks and keeping projects well-organized, ensuring that all the necessary tools and libraries are in place.

2. The Importance of Maven in Software Development

- **Without Maven**: Developers must handle every step manually, including finding libraries, compiling code, running tests, and packaging the project.
- **With Maven**: Maven automates these tasks. It structures your project, downloads the libraries, compiles the code, runs tests, and packages everything for you. This results in faster development and fewer errors.

Example: If you're building a Java project that requires a library for image processing, you normally need to search for the correct library, download it, and set it up manually. **With Maven**, you just specify the library in the POM file, and Maven automatically fetches and integrates it into your project.

3. Maven Operations

Maven relies on a **POM file** (Project Object Model) to function. This file serves as an instruction guide for Maven, telling it what libraries are needed and how to build the project.

4. Exploring the POM File (Project Object Model)

The **POM file** is the backbone of any Maven project. It is written in XML and specifies how the project is structured, what dependencies are needed, and how the project should be built.

Simple Example of a POM File: Think of the POM file like a recipe. It lists the ingredients (libraries) and the steps needed to create the final product.

```
xml
Copy code
<project>
    <groupId>com.example</groupId>
    <artifactId>my-project</artifactId>
    <version>1.0</version>

    <dependencies>
        <dependency>
            <groupId>org.example</groupId>
            <artifactId>image-library</artifactId>
            <version>2.1</version>
        </dependency>
    </dependencies>
</project>
```

Breaking it down:

- **groupId**: Identifies the organization or group that makes the project.
- **artifactId**: The name of your project (e.g., `my-project`).
- **version**: The version number of your project (e.g., `1.0`).
- **dependencies**: This section lists the libraries your project needs. Here, it specifies the **image-library** from `org.example`.
- **Without Maven**: You would need to find and download the **image-library** manually.
- **With Maven**: You just list it in the POM file, and Maven automatically downloads it for you.

5. Build Lifecycle in Maven

Maven follows a series of steps called the **build lifecycle** to create and manage your project.

- **Without Maven**: You have to manually compile your code, run tests, and package everything into a JAR file.
- **With Maven**: You can run a command like `mvn package`, and it will automatically do all these steps in the right order:
    1. **Clean**: Deletes old files from previous builds.
        - Command: `mvn clean`
    2. **Test**: Runs tests to check if the code works.
        - Command: `mvn test`
    3. **Package**: Bundles everything into a single file for you to use or share.
        - Command: `mvn package`
    4. **Install**: Installs the package into your local repository.
        - Command: `mvn install`
    5. **Deploy**: Sends the package to a remote repository for others to use.
        - Command: `mvn deploy`
- **Example**: You can use commands like `mvn clean`, `mvn compile`, or `mvn package` to perform specific actions during the build process.

6. Understanding JAR, WAR, and Target Files

In the **package** phase, Maven produces files that vary depending on project type:

- **JAR (Java ARchive)**: Contains compiled Java code, resources, and dependencies for **standalone applications**.
- **WAR (Web Application Archive)**: Packages web components (servlets, JSP, HTML, CSS) for **web applications**. Deploying this on a server enables web access.

All these files are stored in the **target folder**, created automatically during the build and containing both intermediate and final build files.

7. Managing Libraries with Maven Repositories

Maven uses **repositories** to store and find the libraries your project needs.

Types of Repositories:

1. **Local Repository**: This is on your computer. Maven checks here first to see if you have the libraries you need.
2. **Central Repository**: This is an online store of libraries managed by the Maven community. If a library isn't in your local repository, Maven downloads it from here.
3. **Remote Repository**: If a library is not available in the central repository, you can specify an additional location for Maven to look.

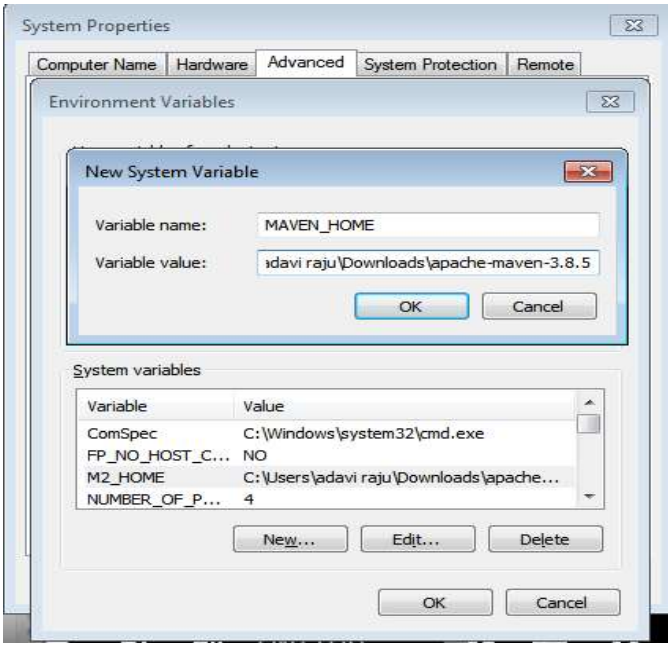
Example: When your project needs a library, Maven will first look in your local repository. If it's not there, it will automatically fetch it from the central repository.

8. Installing and Setting Up Maven

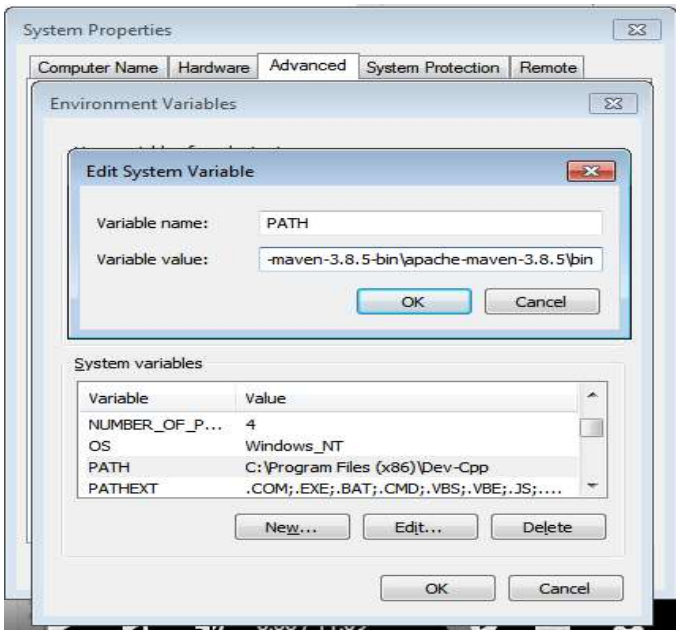
To start using Maven, you need to install it and set it up on your computer.

1. **Download Maven**: Get the latest version from the Apache website.
2. **Extract Files**: Unzip the downloaded files.

3. **Set Environment Variables**: You need to set up `JAVA_HOME` (where Java is installed) and `MAVEN_HOME` (where Maven is located) in your system settings.



4. **Add Maven to PATH**: This allows you to use Maven commands from any folder on your computer.



5. **Verification**: You can check if Maven is installed correctly by typing `mvn -version` in your terminal. If everything is set up right, it will show the version of Maven.


Command Prompt
Microsoft Windows [Version 10.0.19045.4894]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Madhu>mvn -version
Apache Maven 3.8.5 (3599d3414f046de2324203b78ddcf9b5e4388aa0)
Maven home: E:\apache-maven-3.8.5
Java version: 11.0.13, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-11.0.13
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
C:\Users\Madhu>