* **Introducing Apache Maven**

Maven is a build automation and project management tool from the Apache Group, primarily for Java projects. It simplifies the build process by providing a standard, automated framework for:

* Dependency Management: Automatically downloads and configures project libraries.
* Uniform Build System: Uses a standard project structure and a default build lifecycle.
* Project Information: Generates reports and documentation from your project's metadata.
* **Installing Maven**

1. **Prerequisites:** Ensure a Java Development Kit (JDK) is installed and the JAVA\_HOME environment variable is configured correctly.
2. **Download and Extract:** Download the latest binary zip file from the Apache Maven website and extract it to a directory.
3. **Set Environment Variables:** Add the Maven bin directory to your system's PATH variable.
4. **Verify Installation:** Open a new terminal and run mvn -version to confirm that Maven is installed correctly.

* **Maven repositories**

Maven uses repositories to store project artifacts (like JARs) and dependencies.

* Local Repository: A directory on your local machine (~/.m2/repository by default) that caches downloaded dependencies.
* Central Repository: The main public repository maintained by the Maven community, containing most common open-source libraries.
* Remote Repository: A custom repository, often internal to a company, used for private libraries or third-party artifacts not in the central repository.
* **Creating a Simple Maven Project in Eclipse**

1. **Launch Eclipse:**

* Open your Eclipse IDE.

1. **Create New Maven Project:**
   * Go to File > New > Project...
   * In the "New Project" wizard, type "Maven" in the filter box and select Maven Project. Click Next.
2. **Choose Project Type:**
   * Check the box for Create a simple project (skip archetype selection). This creates a basic Maven-enabled Java project. Click Next.
3. **Configure Project Details:**
   * **Group Id:** Enter a unique identifier for your organization or project group (e.g., com.example.myproject).
   * **Artifact Id:** Enter the name of your project (e.g., my-maven-app).
   * **Version:** Specify the project version (e.g., 1.0.0-SNAPSHOT).
   * Click Finish.
4. **Project Structure:**

Eclipse will create a Maven project with a standard structure, including:

* + **src/main/java**: For your main Java source code.
  + **src/main/resources**: For main project resources.
  + **src/test/java:** For your test Java source code.
  + **src/test/resources**: For test resources.
  + **pom.xml**: The Project Object Model file, which defines project configurations, dependencies, and build instructions.

1. **Managing Dependencies:**

Open the pom.xml file to add or manage project dependencies. You can find dependency information from the Maven Central Repository and add them within the <dependencies> tag in your pom.xml. Eclipse will automatically download and integrate these dependencies upon saving the pom.xml.

1. **Building and Running:**
   * Right-click on your project in the Package Explorer.
   * Select Run As > Maven Build... to execute Maven goals (e.g., package, install).
   * To run your Java application, right-click on the main class and select Run As > Java Application.

* **The Project Object Model (POM)**

The POM is the core of a Maven project, defined in a pom.xml file located in the project's root directory. This XML file contains all project information and configuration details, including:

* Project metadata: groupId, artifactId, version
* Dependencies: External libraries needed by the project
* Build setup: Plugins and lifecycle phases
* **The Build Lifecycle**

Maven provides three built-in build lifecycles, each with a sequence of phases. When you execute a phase, all preceding phases are also run in order.

**Default (Build) Lifecycle**

The main lifecycle for building and deploying a project. Key phases include:

* validate: Checks if the project is correct.
* compile: Compiles the source code.
* test: Runs unit tests.
* package: Packages the compiled code into a distributable format, like a JAR or WAR.
* install: Installs the package into your local repository.
* deploy: Deploys the package to a remote repository.

**Clean Lifecycle**

Manages project cleanup. The main phase is clean, which deletes the target directory created by the build.

**Site Lifecycle**

Handles the creation and deployment of project documentation.

Maven configuration

* Project-level: The pom.xml file in your project directory defines the build configuration, dependencies, and plugins.
* User-level: The settings.xml file (in ~/.m2/) configures global settings, like local repository location and proxy settings, for all projects used by a specific user.
* **Build profiles**

Profiles are customizable sets of configuration values that can be defined in the pom.xml or settings.xml. They allow you to modify a project's build for different environments (e.g., development, testing, production) and can be activated through the command line or other triggers.

* **Running a Maven project**

To run a specific phase, use the mvn command in your project's root directory.

* mvn clean: Cleans the project.
* mvn package: Compiles and packages the code.
* mvn install: Builds and installs the artifact to the local repository.
* mvn clean install: Cleans and then installs the project.
* **Creation of a Maven project**

A standard Maven project structure can be generated using a template called an archetype.

* **Creating a Maven project via the command line**

1. Open a terminal.
2. Run the **mvn archetype:generate** command, specifying the archetype you want to use. A common choice is maven-archetype-quick start for a simple Java project.

**bash**

**mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false**

1. Maven will generate the project directory (my-app) with a standard layout and a pom.xml file.

* **Creating a Maven project in an IDE (e.g., Eclipse)**

1. Go to File > New > Maven Project.
2. Choose an archetype from the list. The maven-archetype-quick start is a good starting point.
3. Enter the Group Id, Artifact Id, and other project details.
4. The IDE will generate the project structure, including the pom.xml file.