**Maven**

**What is build?**

The process of converting source code into an executable file.

eg : The final output or product (e.g., executable, JAR file) generated from the source code.

**Traditional build process**

The sequence of steps and activities performed to create the build.

several steps are there, here are some examples

**Source Code Compilation**: Manually compile the source code files (e.g., .java files in Java) into object code or bytecode using a compiler.

**Dependency Management**: Manually download and include all necessary external libraries and dependencies required for the project.

**Code Packaging**: Manually bundle the compiled code and resources into an executable format, such as a JAR file for Java applications.

**Configuration Management**: Manually manage and configure environment-specific settings for different deployment environments.

**Testing**: Write and execute test cases manually, often without automated testing frameworks, to ensure code correctness.

**Deployment**: Manually transfer the packaged application to the target deployment environment, such as copying files to a server.

**What is a build tool?**

A build tool is software that automates the process of converting the source code files into executable files.

This process can include compiling the code, packaging, running tests, and deploying the software.

**List of Build Tools**

**Maven**

* **Language:** Java
* **Configuration:** XML (POM.xml)
* **Features:**
  + Automatic dependency management
  + Extensive plugin ecosystem
  + command based

**Gradle**

* **Language:** Java, Groovy, Kotlin
* **Configuration:** Groovy/Kotlin DSL
* **Features:**
  + Excellent for complex and multi-project builds
  + High performance and customizable
  + script based

**Ant**

* **Language:** Java
* **Configuration:** XML (build.xml)
* **Features:**
  + Highly flexible and extensible
  + Suitable for custom-build processes
  + script based

**Ivy**

* **Language:** Java
* **Configuration:** XML (ivy.xml)
* **Features:**
  + Focuses on dependency management
  + Fine-grained control over dependencies
  + Integrates well with Apache Ant
  + script based

**npm (Node Package Manager)**

* **Configuration**: package.json
* **Languages**: JavaScript
* **Features**: Manages dependencies and scripts, widely used in the JavaScript ecosystem.
* command-based

**What is Maven?**

A build automation tool used primarily for Java projects.

Simplifies project management by automating the build process.

Manages project dependencies, builds, documentation, and reporting.

**In 2002**, a developer named **Jason van Zyl** decided to tackle these problems.

He created a tool that simplifies the build process, manages dependencies automatically, and provides a standardized project structure. He named this tool **Maven**.

**Features of Maven**

1. **Dependency Management:** Automatically download and manage libraries and plugins.
2. **Consistent Project Structure:** Standard directory layout for projects.
3. **Build Automation:** Automate compilation, testing, packaging, and deployment.
4. **Plugins:** Extend functionality through plugins.
5. **Project Object Model:** POM.xml file manges project information, dependencies, plugins, and build configurations.
6. **Build life cycle:** Pre-defined build lifecycles such as default, clean and site.
7. **Central Repository and Local Cache:** Uses a central repository to download dependencies and local repository to storing downloaded dependencies.
8. **Ease of Use:** Simplifies complex build processes with straightforward commands.

**POM.xml**

The Project Object Model (POM) is an XML file that contains information about the project and configuration details used by Maven to build the project. It includes details such as:

* Project coordinates (groupId, artifactId, version)
* Dependencies
* Build plugins
* Build configurations

**Types of Repositories**

1. **Local Repository:** Cached on your local machine.
2. **Central Repository:** Public repository maintained by the Maven community.
3. **Remote Repository:** Custom or third-party repositories.

**Maven Life Cycle**

Maven has three built-in build life cycles:

1. **defaul**t: Handles project deployment.
2. **clean**: Handles project cleaning.
3. **site**: Handles project documentation.

**Maven Phases**

**> default**

Each life cycle is made up of phases, which are steps in the build process. The default life cycle includes:

**Validate:** Check if the project is in the correct format and identify the POM.xml file. command → *mvn clean*

**compile:** converts the source code into machine understandable language → *mvn* *compile*

**test:** Run the test cases of unit test. command → *mvn* *test*

**package:** Takes the compiled code and packages it (e.g., into a JAR). command → *mvn* *package*

**integration test:**  Runs integration tests. command → *mvn* *integration-test*

**verify:** Run checks on the results of integration tests. command → mvn verify

**install:** Install the package into the local repository. command → *mvn* *install*

**deploy:** Copy the final package to the remote repository. command → *mvn* *deploy*

**> clean**

The **clean** lifecycle in Maven is used to clean the project, removing all files generated by the previous build.

**pre-clean**: Perform any operations needed before cleaning. This phase is usually executed as part of the clean phase, so we don't typically run it separately.

**clean**: Remove all files generated by the previous build. command → *mvn clean*

**post-clean**: Perform any operations needed after cleaning. This phase is usually executed as part of the clean phase, so we don't typically run it separately.

**> site**

The **site** lifecycle in Maven is used to create project documentation and site reports.

**pre-site**: Perform any operations needed before generating the site documentation. This phase is usually executed as part of the site phase, so we don't typically run it separately.

**site**: Generate the project's site documentation. command → *mvn site*

**post-site**: Perform any operations needed after generating the site documentation, and prepare the site for deployment. This phase is usually executed as part of the site phase, so you don't typically run it separately.

**site-deploy**: Deploy the generated site documentation to a web server or repository. command → *mvn site-deploy*

**Hands-On Example**

1. **Create a Maven Project:** mvn archetype:generate
2. **Navigate to Project Directory:** cd my-app
3. **Build the Project:** mvn clean install