Maven Fundamentals
Maven is a project management and comprehension tool used primarily in Java projects. Its main goal is to simplify and automate the build, test, package, and deployment processes of applications while following a standardized and repeatable approach. Maven adheres to the principle of "Convention over Configuration," meaning that with minimal configuration, significant results can be achieved.

Key Features of Maven

- Dependency Management: Automates the inclusion and updating of libraries required for the project.
- Project Build: Simplifies the process of compiling, testing, and packaging the project into various formats (JAR, WAR, etc.).
- Standard Lifecycle: Defines common phases such as clean, compile, test, package, verify, install, and deploy.
- Repositories: Manages dependencies and plugins stored in local, remote, or centralized repositories.
- Plugins: Extends Maven’s functionality through plugins that can handle tasks such as compiling, testing, and code analysis.

Project Configuration
Maven uses a file called pom.xml (Project Object Model) to define the project’s structure and configuration. This file contains information about the project, its dependencies, plugins, and settings necessary to build the project.

Dependency Management
One of Maven’s most powerful aspects is its ability to manage dependencies. In the pom.xml file, you can specify which external libraries your project requires. Maven will automatically download these libraries from a remote repository (like Maven Central) and add them to your project’s classpath.

- Transitive Dependencies: If a dependency has its own dependencies, Maven will automatically download those as well, efficiently resolving dependency trees.
- Dependency Scopes: Maven allows you to define the scope of a dependency, such as compile, test, runtime, provided, or system, which determines when and how the dependency is included.

Maven Repositories
Maven uses repositories to store dependencies and plugins. There are three types of repositories:

- Local Repository: Located on the user's machine (~/.m2/repository by default). Maven first searches for dependencies here before looking in remote repositories.
- Central Repository: The default repository for Maven, containing most public dependencies.
- Remote Repositories: Can be configured to fetch dependencies from external locations not included in the central repository.

Build Lifecycle
Maven defines a build lifecycle consisting of phases, and each phase has a set of associated tasks. Some of the most important phases include:

- validate: Validates that the project is correct and that all necessary information is available.
- compile: Compiles the project's source code.
- test: Runs unit tests.
- package: Packages the compiled code into a distributable format like JAR or WAR.
- install: Installs the package into the local repository.
- deploy: Copies the final package to the remote repository to share with other developers or projects.

Maven Plugins
Maven allows you to extend its functionality through plugins. Some popular plugins include:

- maven-compiler-plugin: For compiling source code.
- maven-surefire-plugin: For running unit tests.
- maven-jar-plugin: For packaging the application as a JAR.

References

O'Brien, T. (n.d.). Maven: The complete reference. Sonatype. Retrieved from https://books.sonatype.com/mvnref-book/reference/

Apache Software Foundation. (n.d.). Apache Maven documentation. Retrieved from https://maven.apache.org/guides/

O'Brien, T. (2008). Maven: A beginner's guide. O'Reilly Media. Retrieved from https://www.safaribooksonline.com/library/view/maven-a-beginners/9780596517335/

Sonatype. (n.d.). Maven by example. Sonatype. Retrieved from https://books.sonatype.com/mvnex-book/reference/

Baeldung. (n.d.). A guide to Maven. Retrieved from https://www.baeldung.com/maven