1. MAVEN TOTURIAL

* Building a software project typically consists of such tasks
  + downloading dependencies, putting additional jars on a classpath
  + compiling source code into binary code
  + running tests
  + packaging compiled code into deployable artifacts such as JAR, WAR, and ZIP files
  + deploying these artifacts to an application server or repository.
* [Apache Maven](https://maven.apache.org/) **automates** these tasks, minimizing the risk of humans making errors while building the software manually and separating the work of compiling and packaging our code from that of code construction.
* The key features of Maven are:
  + **simple project setup that follows best practices:** Maven tries to avoid as much configuration as possible, by supplying project templates (named archetypes)
  + **dependency management:** it includes automatic updating, downloading and validating the compatibility, as well as reporting the dependency closures (known also as transitive dependencies)
  + **isolation between project dependencies and plugins:** with Maven, project dependencies are retrieved from the *dependency repositories* while any plugin's dependencies are retrieved from the *plugin repositories,* resulting in fewer conflicts when plugins start to download additional dependencies
  + **central repository system:** project dependencies can be loaded from the local file system or public repositories, such as [**Maven Central**](https://search.maven.org/classic/)

# **Project Object Model**



* Project Object Model (POM), represented by a pom.xml file.
* The POM also defines the relationships among modules of multi-module projects.
* Project Identifiers: Maven uses a set of identifiers, also called coordinates, to uniquely identify a project and specify how the project artifact should be packaged
  + *groupId* – a unique base name of the company or group that created the project
  + *artifactId* – a unique name of the project
  + version – a version of the project
  + *packaging* – a packaging method (e.g. *WAR*/*JAR*/*ZIP*)
* **Dependence**
  + external libraries that a project uses are called dependencies. The dependency management feature in Maven ensures automatic download of those libraries from a central repository, so you don't have to store them locally.
  + This is a key feature of Maven and provides the following benefits:
    - uses *less storage* by significantly reducing the number of downloads off remote repositories
    - makes *checking out a project* quicker
    - provides an effective platform for exchanging binary artifacts within your organization and beyond without the need for building artifact from source every time
* Repository
  + A repository in Maven is used *to hold build artifacts and dependencies* of varying types.
  + If an artifact or a plug-in is available in the local repository, Maven uses it. Otherwise, it is downloaded from a central repository and stored in the local repository.
* Properties
  + **Maven properties are value-placeholders and are accessible anywhere within a**pom.xml**by using the notation**${name}, where name is the property.
* Build
  + It provides information about the default Maven goal, the directory for the compiled project, and the final name of the application. The default build section looks like this:



* + The default output folder for compiled artifacts is named target, and the final name of the packaged artifact consists of the artifactId and version, but you can change it at any time.

1. Maven Build LifeCycle