

# Build Automation Tool

---



---

***Maven***<sup>TM</sup>

# Why Automated Builds



Automated Build are best practice.



Manual procedures are mistake prone.



Automated builds are self documentary.



Automated Builds can be triggered by other tools(eg Jenkins,cron jobs).

# Code Build or Integration



Code Build is a process by which source code is converted into a standalone form that can be run on a particular type of platform.



One of the most important step of the software build is the compilation process, where the source code files are converted into executable files.



The process of building software is usually managed by build tool.



Build has been created when a certain point in development has been reached or the code is deemed ready for implementation., either for testing or release.

# Maven Introduction

Apache Maven is a software project management and comprehension tool.

**Maven** is a build automation tool used primarily for Java projects.

Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages.

Maven addresses two aspects of building software: how software is built, and its dependencies.

# Project Object Model(POM)

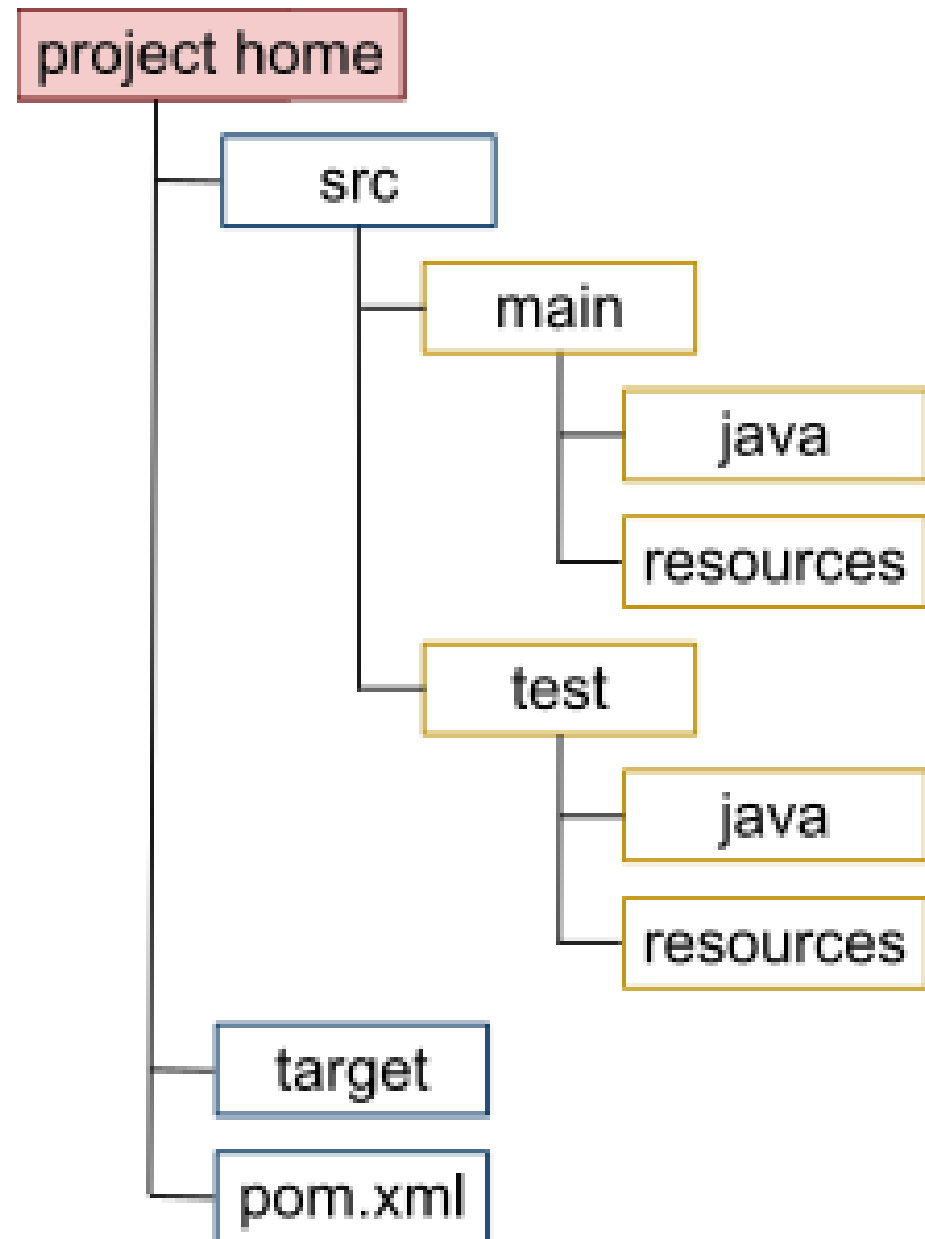
---

- Maven projects are configured using a Project Object Model, which is stored in a pom.xml-file.

```
<project>
  <!-- model version is always 4.0.0 for Maven 2.x POMs -->
  <modelVersion>4.0.0</modelVersion>
  <!-- project coordinates, i.e. a group of values which uniquely identify this project -->
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0</version>
  <!-- library dependencies -->
  <dependencies>
    <dependency>
      <!-- coordinates of the required library -->
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <!-- this dependency is only used for running and compiling tests -->
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

A directory  
structure for a  
Java project  
auto-generated  
by Maven

---



# Directory structure of a Maven project

---

Directory name	Purpose
project home	Contains the pom.xml and all subdirectories.
src/main/java	Contains the deliverable Java sourcecode for the project.
src/main/resources	Contains the deliverable resources for the project, such as property files.
src/test/java	Contains the testing Java sourcecode (JUnit or TestNG test cases, for example) for the project.
src/test/resources	Contains resources necessary for testing.

# Apache Maven Project

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example.training</groupId>
  <artifactId>maven-training</artifactId>
  <version>1.0</version>
</project>
```

- Maven uniquely identifies a project using:

**groupId:** Arbitrary project grouping identifier

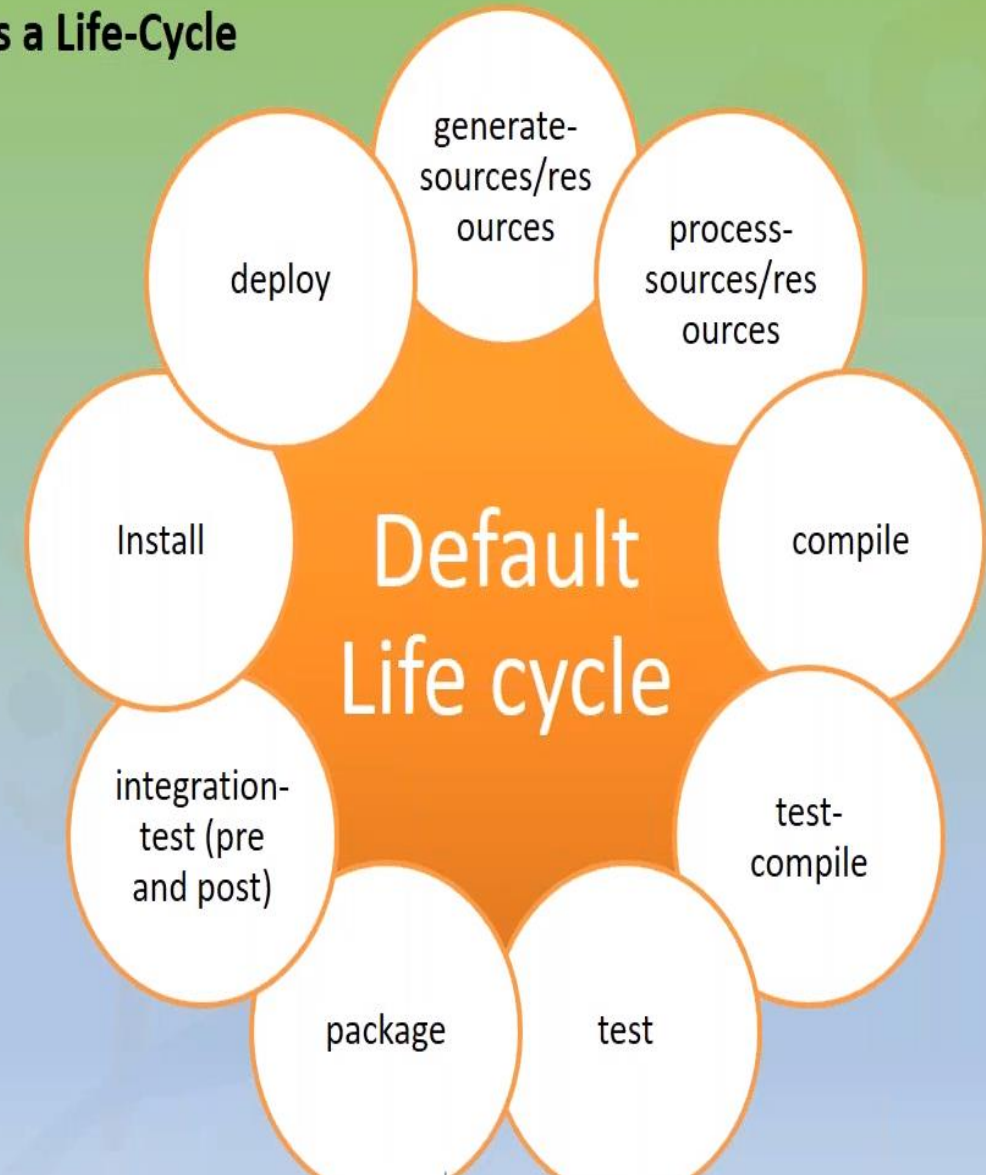
**artifactid:** Arbitrary name of the project.

**Version:** version of the project



# Maven Build Life Cycle

Every Maven Build has a Life-Cycle

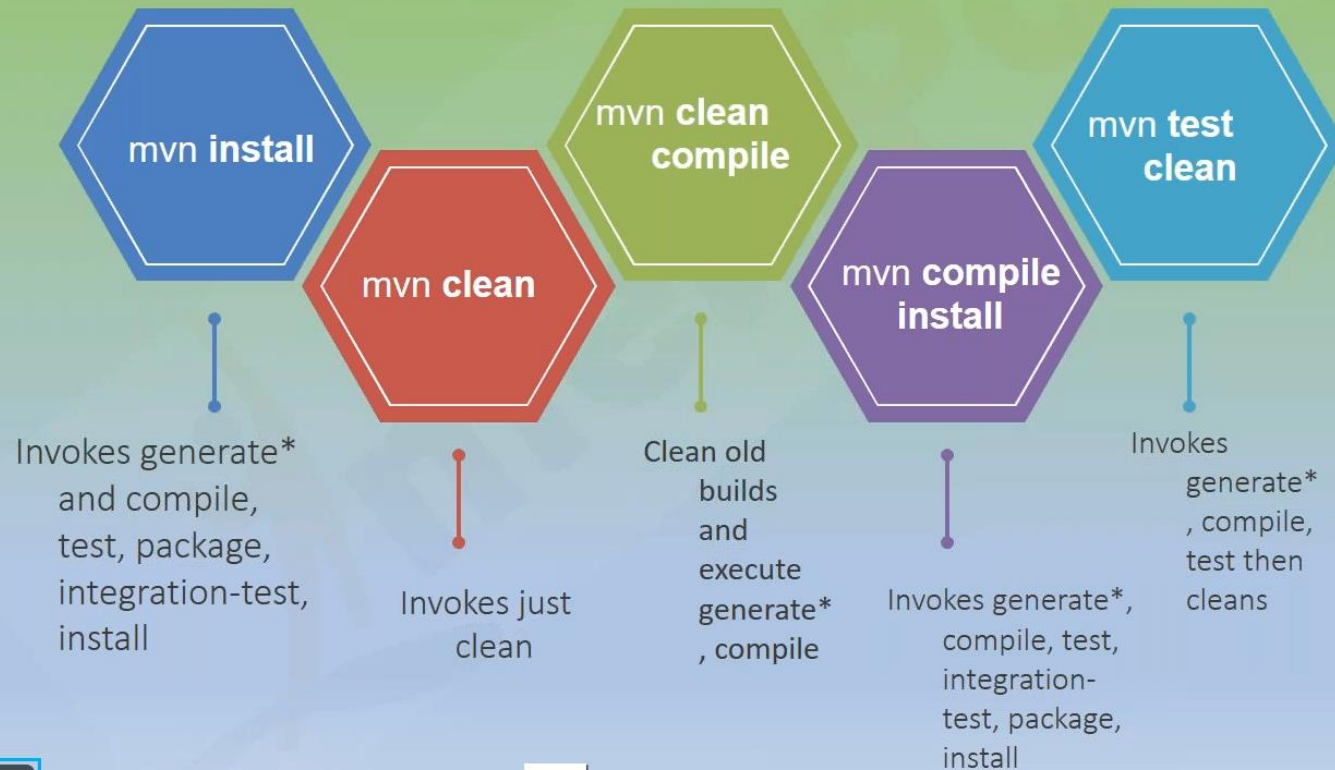


# Maven Build Lifecycle

---

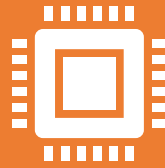
Phase	Handles	Description
prepare-resources	resource copying	Resource copying can be customized in this phase.
validate	Validating the information	Validates if the project is correct and if all necessary information is available.
compile	compilation	Source code compilation is done in this phase.
Test	Testing	Tests the compiled source code suitable for testing framework.
package	packaging	This phase creates the JAR/WAR package as mentioned in the packaging in POM.xml.
install	installation	This phase installs the package in local/remote maven repository.
Deploy	Deploying	Copies the final package to the remote repository.

voke a Maven build you set a lifecycle "goal"



# Apache Maven Goals

# Apache Maven Dependency Management



Maven's dependency-handling mechanism is organized around a coordinate system identifying individual artifact such as software libraries and modules.



No need to store the libraries in SCM.



A central maven repository.

# Apache Maven Dependency Management

- Adding a Dependency.
- Dependency consist of
  1. GAV
  2. Scope: compile, test, provided (default=compile)
  3. Type: jar, pom, war

```
<project>
  ...
  <dependencies>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>2.5</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>
```

# References

- 1. download <https://github.com/devopsguru91/MavenCounterWebApp.git>
  - 2. `git clone https://github.com/devopsguru91/MavenCounterWebApp.git`
  - 3. `mvn package`
- 
- 1. `git clone https://github.com/devopsguru91/MavenCalcTest.git`
  - 2. `mvn clean test`
- 
- 1. `git clone https://github.com/devopsguru91/MavenTestDemo.git`