# APACHE MAVEN

JOHN BRYCE
Leading in IT Education
a matrix company

1. What is Apache Maven?
2. Maven's Objectives
3. What is a Software Build?
4. Installing Maven
5. Creating a New Maven Project in Eclipse
6. Maven Project Structure
7. POM
8. Maven Coordinates – GAV
9. Dependency Management
10. Maven Repositories

# What is Apache Maven?

https://www.apache.org/

Oversees more than 350 leading **Open Source projects**, including Apache HTTP Server -- the world's most popular Web server software

https://maven.apache.org/

An ASF project – a **tool** for **building** and **managing** Java-based projects.

**Maven** - a Yiddish word, meaning *accumulator of knowledge* - מבין

# Maven's Objectives

- Provide a standard, uniform **build system**
- Provide a clear definition of what the project consists of
- Manage **dependencies**
- Provide a way to **share JARs** across several projects
- Provide an easy way to publish **project information**
- Handle **versioning** and releases
- **Make the day-to-day work of Java developers easier**

# What is a Software Build?

**Software build** is the process of converting source code files into standalone software artifact(s) that can be run on a computer.

**Build tools** automate the process of software build and the associated processes including:

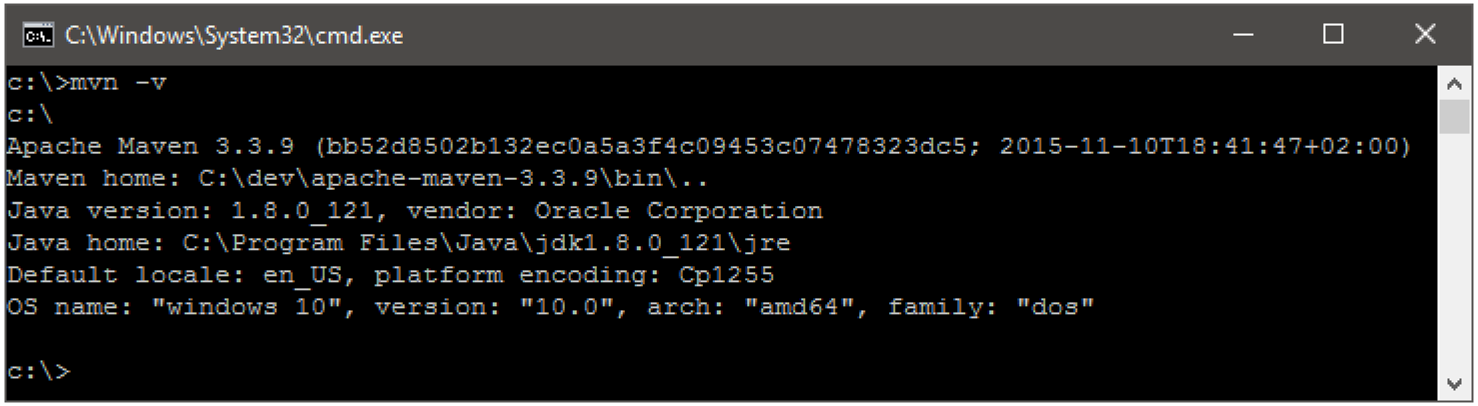
- Compiling computer source code into binary code
- Packaging binary code
- Running automated tests

**Maven is one of most popular build tools for Java!**

(but it's not only a build tool…)

# Installing Maven

1. Ensure JAVA_HOME environment variable is set and points to your JDK installation
2. Download Maven from: https://maven.apache.org/download.cgi
   On Windows, choose: **apache-maven-<version>-bin.zip**
3. Extract the ZIP file in any directory
4. Add the **bin** directory of the created directory **apache-maven-<version>** to the PATH environment variable
5. Confirm Maven is properly installed with command line: **mvn -v**
   The result should look similar to:



```
c:\>mvn -v
c:\
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-10T18:41:47+02:00)
Maven home: C:\dev\apache-maven-3.3.9\bin\..
Java version: 1.8.0_121, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_121\jre
Default locale: en_US, platform encoding: Cp1255
OS name: "windows 10", version: "10.0", arch: "amd64", family: "dos"

c:\>
```

# Maven Projects Structure

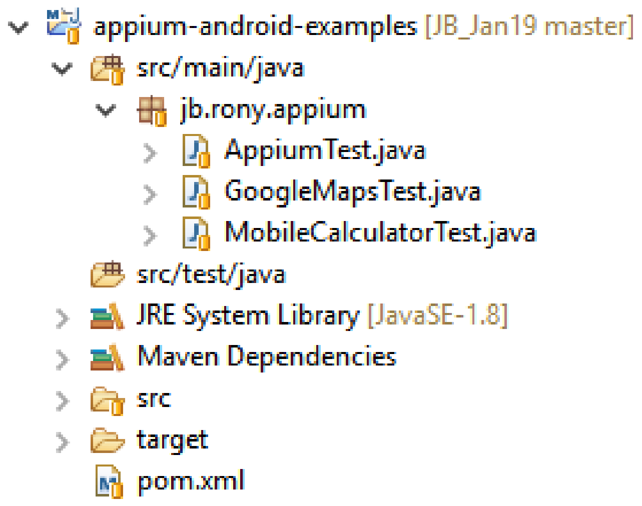
All Maven projects expect a common directories (folders) structure.

Motivation: Having a common directory layout would allow for users familiar with one Maven project to immediately feel at home in another Maven project.

Note: Each of the common directories may contain many sub-directories.

Examples:

| Directory | Purpose |
|---|---|
| **src/main/java** | Java code – all application sources |
| **src/main/resources** | All non-compiled files |
| **src/test/java** | Java code – all test sources |
| **target** | All build artifacts gets here |



```
appium-android-examples [JB_Jan19 master]
  src/main/java
    jb.rony.appium
      AppiumTest.java
      GoogleMapsTest.java
      MobileCalculatorTest.java
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  pom.xml
```

Also:
All Maven projects must have a **pom.xml** file at the project's root directory.

## POM

- POM stands for "Project Object Model". It is an XML representation of a Maven project held in a file named **pom.xml**.
- The POM contains all necessary information about a project, as well as configurations of plugins to be used during the build process.
- **Every Maven project must have a pom.xml file at it’s root folder.**

A minimal POM must contain all the following:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>il.co.jb</groupId>
  <artifactId>my-project</artifactId>
  <version>1.0</version>
</project>
```

**groupId**, **artifactId**, **version** (GAV) are all required fields. More on the GAV in the next slide.

# Maven Coordinates - GAV

JOHN BRYCE
Leading in IT Education
a matrix company

**groupId**, **artifactId** and **version** **(GAV)** are mandatory fields in the **pom.xml** file.

The GAV combination is the **unique identifier** of a Maven project.

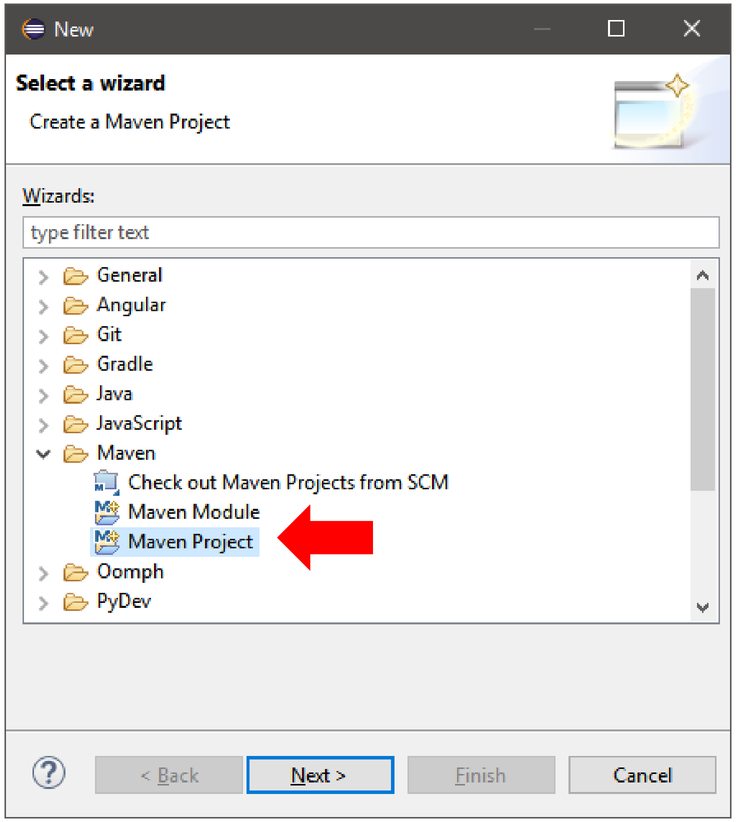
This combo is also necessary when specifying dependencies (discussed on the next slide).

**Naming Conventions:**

- **groupId** – Usually represents the ID of the organization or company. Different projects of the same organization will usually have the same groupId.
  Example: a fictional company named "Super Corp" that uses the domain name: supercorp.com, will use the groupId: **com.supercorp** for all it's projects.
  (Note that this convention is similar to the naming convention for Java packages – domain name in reverse order).
- **artifactId** - This is the project ID. Usually matches the name of the project.
  Example: **super-app**
- **version** – This is the project version. Usually comes in the format: {major}.{minor}.{maintenance}. Example: **2.0.15**

# Create A Maven Project in Eclipse

Although Maven is a command-line utility where all commands are executed by the **mvn** tool, all popular Java IDEs come with Maven integration and support many Maven-related features out of the box.

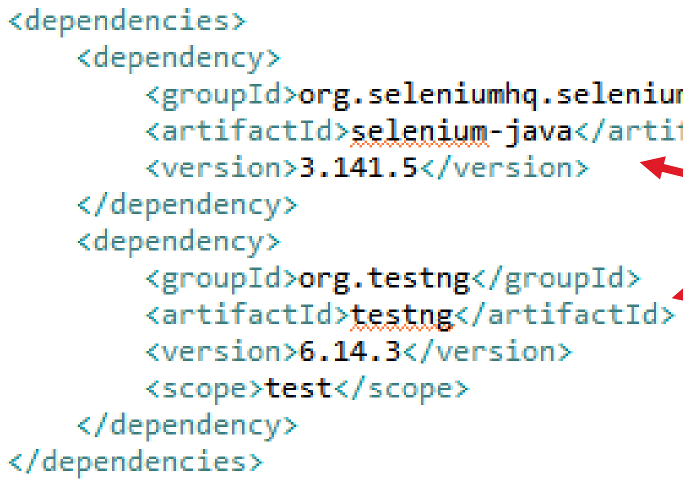


**To Create a new Maven project in Eclipse:**

1. On the top main menu: File -> New -> Other
2. Choose “Maven Project” under the “Maven” folder. Click “Next”.
3. Leave “Use default Workspace location” selected and click “Next”.
4. Leave the default selection of Artifact Id: “maven-archetype-quickstart” and click “Next”
5. Specify values for: Group Id, Artifact Id, Version and Package. Use **naming conventions** as discussed in the previous slides.
6. Click “Finish”. You now have a new Maven project in your Eclipse workspace.

# Dependency Management

- Dependencies are **software libraries** that a software project needs. Example: Selenium is a library for web browsers automation.
- In Java, libraries come in the form of JAR files.
- Dependency management is a **core feature** of Maven.
- Dependencies are specified in the **pom.xml**
- To add a dependency, we need to know its **groupId**, **artifactId** and **version**. Then we add it to the POM, under the <dependencies> section:



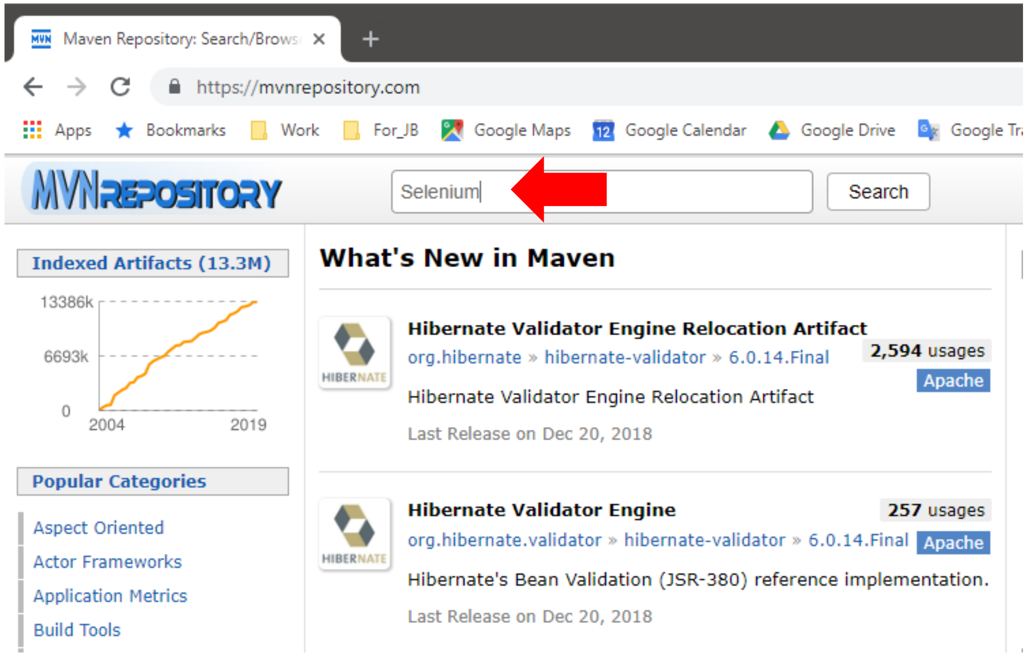
```
<dependencies>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>3.141.5</version>
    </dependency>
    <dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
        <version>6.14.3</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Added two dependencies, for two libraries: **Selenium** & **TestNG**

- Maven **downloads** the dependencies (JARs) from **remote (online) repositories** and stores them in a **local repository** (no need to download the same library twice if it's required by more than one project).

# Maven Repositories

- A Maven repository is used to hold build artifacts and dependencies.
- There are two types of repositories: **local** and **remote**.
- **Remote repositories** are online repositories, usually accessed via the HTTP protocol and allow downloading dependencies they host.
- The Maven Central repository is at: https://mvnrepository.com/ it hosts **millions** of artifacts!
- The local repository refers to a copy on your own computer - that is a cache of the remote downloads.
- On Windows, the local repository is usually located at: **C:\Users\<current_user>\.m2\repository**
- To find the GAV for a dependency (library) that you want to add to your project, search the Maven repository:



What's New in Maven

**Hibernate Validator Engine Relocation Artifact** — 2,594 usages
org.hibernate » hibernate-validator » 6.0.14.Final — Apache
Hibernate Validator Engine Relocation Artifact
Last Release on Dec 20, 2018

**Hibernate Validator Engine** — 257 usages
org.hibernate.validator » hibernate-validator » 6.0.14.Final — Apache
Hibernate's Bean Validation (JSR-380) reference implementation.
Last Release on Dec 20, 2018

JOHN BRYCE
Leading in IT Education
a matrix company

# Thank You!