Maven

PV260 Software Quality

Stanislav Chren

23. 2. 2017

What is Maven

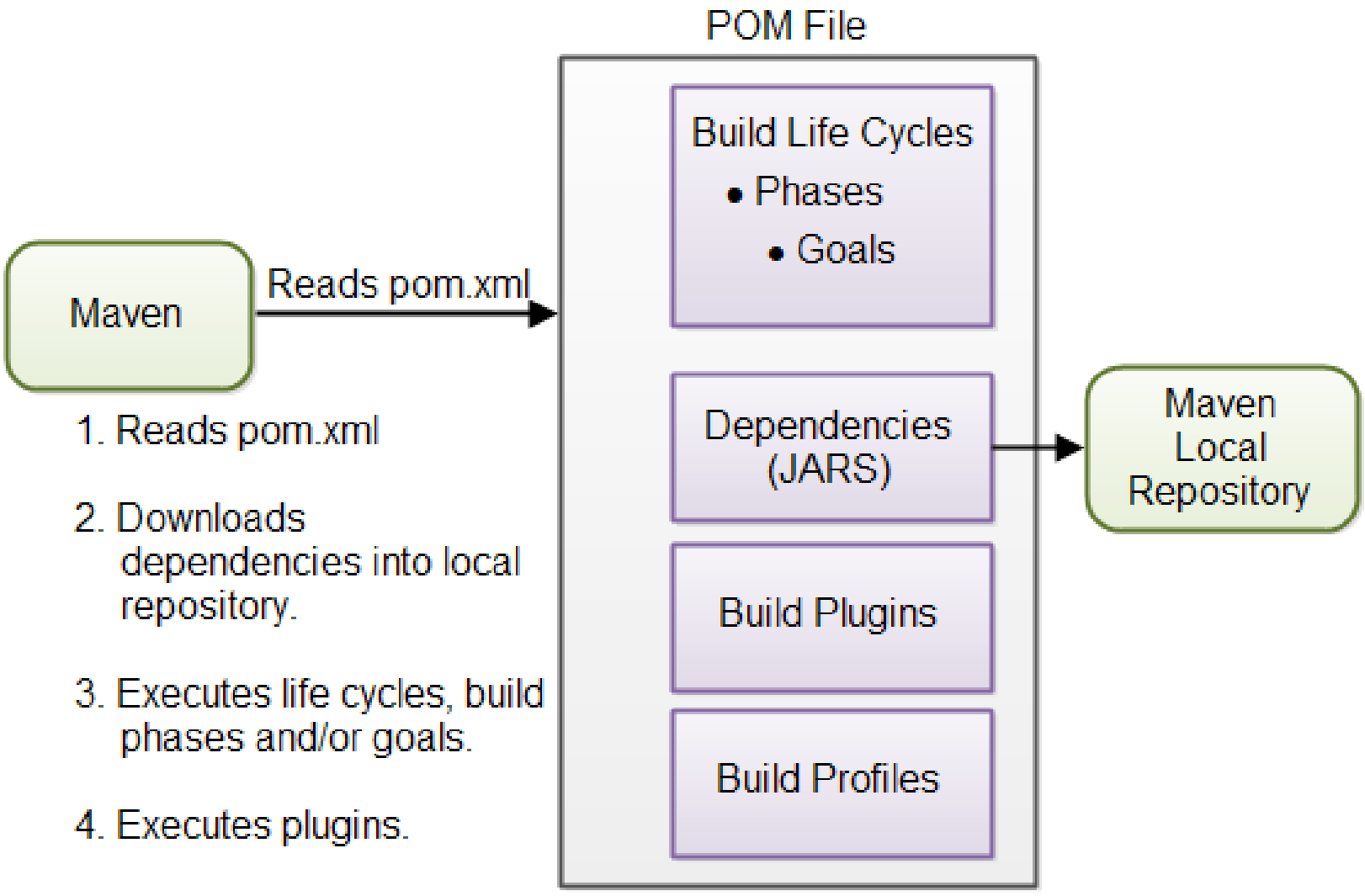
Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

Our Use case

- Building Java projects
- Dependency management

download from `http://maven.apache.org/index.html`

Overview


POM File
Build Life Cycles
• Phases
• Goals
Maven
Reads pom.xml
Dependencies (JARS)
Maven Local Repository
Build Plugins
Build Profiles
1. Reads pom.xml
2. Downloads dependencies into local repository.
3. Executes life cycles, build phases and/or goals.
4. Executes plugins.
All executed according to selected build profile.

Typical Project Structure

```
my-app
|-- pom.xml
`-- src
    |-- main
    |   |-- java
    |   |   `-- com
    |   |       `-- mycompany
    |   |           `-- App.java
    |   `-- resources
    |       `-- META-INF
    |           `-- application.properties
    `-- test
        `-- java
            `-- com
                `-- mycompany
                    `-- AppTest.java
```

POM File

- Mandatory for each Maven project
- Specifies project and build configuration

```
<project>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
    ....
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
      ...
      </plugin>
    </plugins>
  </build>
</project>
```

Dependencies

- Project dependencies are defined inside `<dependency>` elements.
- Example

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.11</version>
  <scope>test</scope>
</dependency>
```

- Dependency artifacts are searched in local, remote and central repositories
- If the artifact is not present in the local repository, it is downloaded from central or specified remote repositories.
- Maven also handles the transitive dependencies.

Dependency Scopes

- Dependency scope is used to limit the transitivity of a dependency, and also to affect the classpath used for various build tasks.

Scope	Description
compile	Default. Dependencies are available in all classpaths of a project
provided	Provided by the JDK or a container at runtime.
runtime	Not required for compilation, but is for execution. Only the runtime and test classpaths.
test	Only the test compilation and execution phases
system	The JAR is provided explicitly without the repository search.

Maven CMD Interface

- Run from CMD using `mvn` followed by the name of a build *life cycle*, *phase* or *goal*, e.g. `mvn clean`
- When Maven builds a software project it follows a build life cycle. The build life cycle is divided into build phases, and the build phases are divided into build goals

Lifecycle	Description
default	handles everything related to compiling and packaging.
clean	handles everything related to removing temporary files from the output directory, including generated source files, compiled classes, previous JAR files etc.
site	handles everything related to generating documentation.

Default Lifecycle Build Phases

- Usage: `mvn phaseName` or `mvn phaseName:goal`

Phase	Description
validate	Validates that the project is correct and all necessary information is available. This also makes sure the dependencies are downloaded
compile	Compiles the source code of the project.
test	Runs the tests against the compiled source code using a suitable unit testing framework.
package	Packs the compiled code in its distributable format, such as a JAR.
install	Install the package into the local repository.
deploy	Copies the final package to the remote repository.

Creating New Project

- The basic project structure is generated using the *archetypes*
- Maven contains archetypes for various types of projects.
- The project creation wizard is executed by: `mvn archetype:generate`
- For a sample HelloWorld project, you can run:

```
mvn -B archetype:generate \
    -DarchetypeGroupId=org.apache.maven.archetypes \
    -DgroupId=com.mycompany.app \
    -DartifactId=my-app
```

- More archetypes: https://maven.apache.org/guides/introduction/introduction-to-archetypes.html

Plugins

- Plugins are used for customizing the Maven build
- Example - making the resulting jar file executable and bundled with all dependencies:

```
<plugin>
  <artifactId>maven-assembly-plugin</artifactId>
  <configuration>
    <archive>
    <manifest>
      <mainClass>fully.qualified.MainClass</mainClass>
    </manifest>
    </archive>
    <descriptorRefs>
      <descriptorRef>jar-with-dependencies</descriptorRef>
    </descriptorRefs>
  </configuration>
</plugin>
```

  Project is then built by: `mvn clean compile assembly:single`

- More info about plugins:
  https://maven.apache.org/plugins/

Tutorials

- http://tutorials.jenkov.com/maven/maven-tutorial.html
- https://maven.apache.org/guides/getting-started/index.html
- http://www.tutorialspoint.com/maven/

Task

1. Generate at least three different project types (one of them should be the default/quickstart). Examine their structure
2. Add following dependencies to the quickstart project:
   - Hibernate
   - Mockito
   - Slf4j
3. Give them appropriate scope. Hibernate should be available in packaged jar, Mockito is only available for testing and Slf4j will be supplied by the environment.
4. Try to execute various life cycles/phases
5. Add the plugin configuration so that the resulted jar is bundled with it its dependencies and is executable via `java -jar <NAME>.jar`