Maven Tutorial for Beginners in 5 Steps

Defining what Maven does is very difficult.

Every Day Developer does a lot of things

- Manages Dependencies
    - Web Layer (Spring MVC)
    - Data Layer (JPA - Hibernate) etc..
- Build a jar or a war or an ear
- Run the application locally
    - Tomcat or Jetty
- Deploy to a T environment
- Add new dependencies to a project
- Run Unit Tests
- Generate Projects
- Create Eclipse Workspace

> *Maven helps us do all these and more...*

Getting Started

- Git Repository - https://github.com/in28minutes/getting-started-in-5-steps/tree/master/maven-in-5-steps
- Pre-requisites
    - Java & Eclipse - https://www.youtube.com/playlist?list=PLBBog2r6uMCSmMVTW__QmDLyASBvovyAO3
- We will use embedded maven in Eclipse

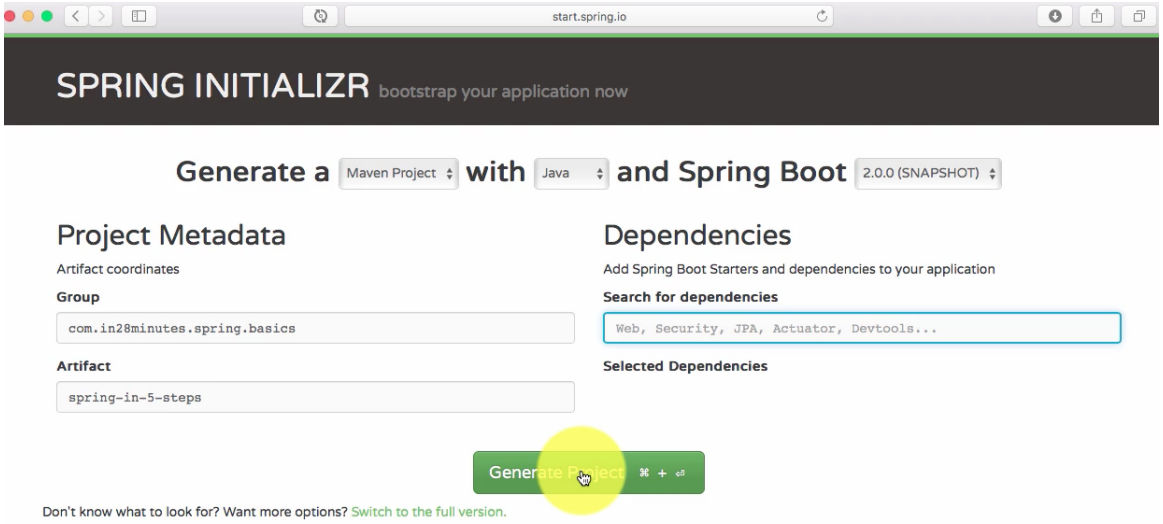
Overview

- Step 1 : Creating and importing a Maven Project
- Step 2 : Understanding Project Object Model - pom.xml
- Step 3 : Maven Build Life Cycle
- Step 4 : How does Maven Work?
- Step 5 : Important Maven Commands

Step 1 : Creating and importing a Maven Project

Creating a Spring Project with Spring Initializr is a cake walk.

> *Spring Initializr http://start.spring.io/ is great tool to bootstrap your Spring Boot projects.*



As shown in the image above, following steps have to be done

- Launch Spring Initializr and choose the following
    - Choose `com.in28minutes.learning.maven` as Group
    - Choose `maven-in-few-steps` as Artifact
    - Choose Dependency
        - Web
- Click Generate Project.
- Import the project into Eclipse.
- If you want to understand all the files that are part of this project, you can go here.

Step 2 : Understanding Project Object Model - pom.xml

Naming a project

How can other projects use our project? By using our project groupId and artifactId

```
<groupId>com.in28minutes.learning.maven</groupId>
<artifactId>maven-in-few-steps</artifactId>
```

Parent Pom

Similar to Java Inheritance. We inherit a lot of things from starter parent.

```
<parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.0.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
</parent>
```

Declaring Dependencies

Dependencies are frameworks that you would need to develop your project.

In the example below we are adding two dependencies.

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
</dependency>

<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
</dependency>
```

Step 3 : Maven Build Life Cycle

When we run "mvn clean install", we are executing the complete maven build life cycle.

Build LifeCycle is a sequence of steps

- Validate
- Compile
- Test
- Package
- Integration Test
- Verify
- Install
- Deploy

> *Maven follows Convention over Configuration.*

Pre defined folder structure

- Source Code
    - ${basedir}/src/main/java
    - ${basedir}/src/main/resources
- Test Code
    - ${basedir}/src/test

Step 4 : How does Maven Work?

Maven Repository contains all the jars indexed by artifact id and group id.

Once we add a dependency to our pom.xml, maven asks the maven repository for the jar dependencies giving group id and the artifact id as the input.

- Maven repository stores all the versions of all dependencies. JUnit 4.2,4.3,4.4

The jar dependencies are stored on your machine in a folder called maven local repository. All our projects would refer to the jars from the maven local repository.

> *Local Repository : a temp folder on your machine where maven stores the jar and dependency files that are downloaded from Maven Repository.*

Step 5 : Important Maven Commands

- mvn –version
- mvn compile (compiles source files)
- mvn test-compile (compiles test files) - one thing to observe is this also compiles source files
- mvn clean - deletes target directory
- mvn test - run unit tests
- mvn package - creates the jar
- help:effective-settings
- help:effective-pom
- dependency:tree
- dependency:sources
- –debug

Complete Code Example

/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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>com.in28minutes.learning.maven</groupId>
        <artifactId>maven-in-few-steps</artifactId>
        <version>0.0.1-SNAPSHOT</version>

        <packaging>jar</packaging>

        <name>maven-in-few-steps</name>
        <description>Demo project for Spring Boot</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>2.0.0.RELEASE</version>
                <relativePath/> <!-- lookup parent from repository -->
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
        </properties>

        <dependencies>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-web</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <build>
                <plugins>
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                        </plugin>
                </plugins>
        </build>
<!--

Remote Maven Repository
->
Local Maven Repository

 - Local Repository => Local System

 - Remote Maven repository => Central Repositories
   - stores all the versions of all dependencies. JUnit 4.2,4.3,4.4

 - mvn install vs mvn deploy
   - copies the created jar to local maven repository - a temp folder on my machine where maven stores the file
```

```
-->
	<repositories>
		<repository>
			<id>spring-snapshots</id>
			<name>Spring Snapshots</name>
			<url>https://repo.spring.io/snapshot</url>
			<snapshots>
				<enabled>true</enabled>
			</snapshots>
		</repository>
		<repository>
			<id>spring-milestones</id>
			<name>Spring Milestones</name>
			<url>https://repo.spring.io/milestone</url>
			<snapshots>
				<enabled>false</enabled>
			</snapshots>
		</repository>
	</repositories>

	<pluginRepositories>
		<pluginRepository>
			<id>spring-snapshots</id>
			<name>Spring Snapshots</name>
			<url>https://repo.spring.io/snapshot</url>
			<snapshots>
				<enabled>true</enabled>
			</snapshots>
		</pluginRepository>
		<pluginRepository>
			<id>spring-milestones</id>
			<name>Spring Milestones</name>
			<url>https://repo.spring.io/milestone</url>
			<snapshots>
				<enabled>false</enabled>
			</snapshots>
		</pluginRepository>
	</pluginRepositories>


</project>
```

/src/main/java/com/in28minutes/learning/maven/maveninfewsteps/MavenInFewStepsApp

```
package com.in28minutes.learning.maven.maveninfewsteps;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MavenInFewStepsApplication {

	public static void main(String[] args) {
		SpringApplication.run(MavenInFewStepsApplication.class, args);
	}
}
```

/src/main/resources/application.properties

/src/test/java/com/in28minutes/learning/maven/maveninfewsteps/MavenInFewStepsApp

```
package com.in28minutes.learning.maven.maveninfewsteps;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
@SpringBootTest
public class MavenInFewStepsApplicationTests {

	@Test
	public void contextLoads() {
	}

}
```

> *Join our free Spring Boot in 10 Steps Course.*

> *Find out how in28Minutes reached 100,000 Learners on Udemy in 2 years. The in28minutes Way - Our approach to creating awesome learning experiences.*