# **Creating a Spring Boot Project with Eclipse and Maven**

## **Introduction**

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.

## **Let’s get Started**

**Naming a project**

You define dependencies in your pom.xml. Maven would download the dependencies and make them available for use in your project.

Just like you can identify a Java class with a class name and a package name, you can identify a maven artifact by a GroupId and an ArtifactId.

|  |
| --- |
| <groupId>com.example.learning.maven</groupId>  <artifactId>maven-in-few-steps</artifactId> |

**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**> |

### **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.*

Predefined folder structure

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

**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.*

**Important Maven Commands**

* mvn –version -> Find the maven 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 a jar for the project
* help:effective-settings -> Debug Maven Settings
* help:effective-pom -> Look at the complete pom after all inheritances from parent poms are resolved
* dependency:tree -> Look at all the dependencies and transitive dependencies
* dependency:sources -> Download source code for all dependencies
* –debug -> Debug flag. Can be used with all the above commands

**Creating Spring Boot Projects with Eclipse and Maven**

There are three options to create Spring Boot Projects with Eclipse and Maven

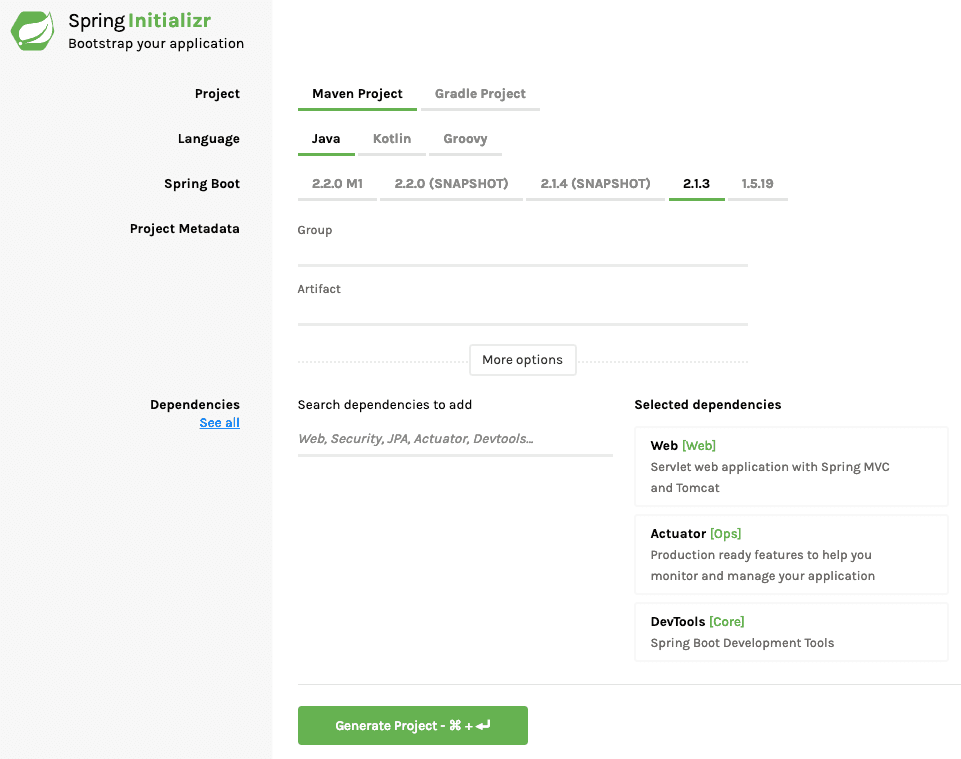
* Spring Initializr - https://start.spring.io
* Use STS or STS Eclipse Plugin and Create a Spring Boot Maven Project directly from Eclipse
* Manually Create a Maven Project and add Spring Boot Starter Dependencies.

We will use a Spring Boot Starter Web as an example.

**Option 1 - Bootstrapping Spring Boot Project with Spring Initializr**

Creating a Web application with Spring Initializr is a cakewalk. We will use Spring Web MVC as our web framework.

*Spring Initializr* [*http://start.spring.io/*](https://start.spring.io/) *is great tool to bootstrap your Spring Boot projects.*

**

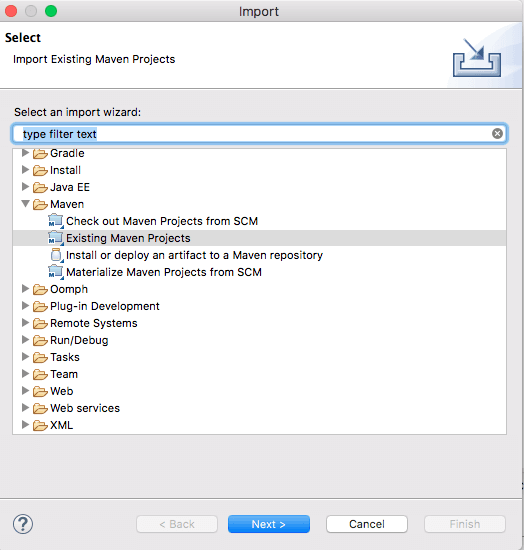
As shown in the image above, following steps have to be done

* Launch Spring Initializr and choose the following
  + Choose com.example as Group
  + Choose student-services as Artifact
  + Choose following dependencies
    - Web
    - Actuator
    - DevTools
* Click Generate Project.

This would download a zip file to your local machine.

Unzip the zip file and extract to a folder.

In Eclipse, Click File -> Import -> Existing Maven Project as shown below.



Navigate or type in the path of the folder where you extracted the zip file to in the next screen.



Once you click Finish, Maven would take some time to download all the dependencies and initialize the project.

That’s it. Your first Spring Project is Ready.

**Option 2 - Using STS or STS Eclipse Plugin to create Spring Boot Maven Project**

With Spring tool suite, you can directly create a spring boot project from Eclipse.

You should either download the complete installation of STS or You can install the STS Eclipse plugin

***https://spring.io/tools/sts/all provides the complete download of STS as well as the Update Sites for STS Eclipse Plugin.***

In Eclipse/STS, start with File -> New -> Spring Starter Project.

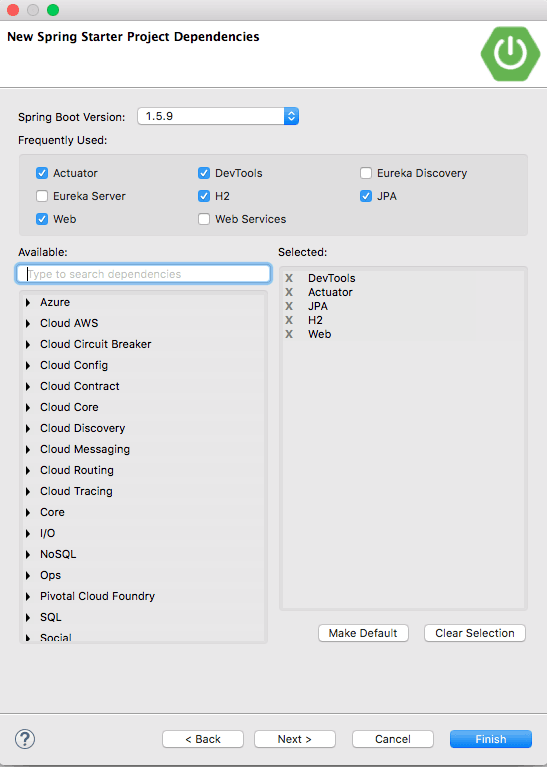
In the next screen, you can choose the following for your project.

* Group ID
* Artifact ID
* Root Package
* Version
* Description
* Java Version
* Language
* Packaging

Make sure you choose Maven as Type.



In the next screen, you can choose the dependencies that you would want to add to your Spring Boot project.



Once you click Finish, Maven would take some time to download all the dependencies and initialize the project.

That’s it. Your first Spring Project is Ready.

**Option 3 - Manually Create a Maven Spring Boot Project**

The last option is to create the project manually.

In Eclipse, start with File > New > Maven Project

Choose Create a simple project as shown in the screenshot below:



In the next screen, provide these details for your project and click Finish.

* Group ID
* Artifact ID
* Version

This would create a basic Maven project with Zero dependencies.

Next add in the appropriate Spring Boot Starters into the pom.xml

|  |
| --- |
| <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-web</**artifactId**> </**dependency**>  <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-devtools</**artifactId**>  <**scope**>runtime</**scope**> </**dependency**>  <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-test</**artifactId**>  <**scope**>test</**scope**> </**dependency**> |

Starter Web is used for developing Spring Boot Web Applications or RESTful Services. Starter Test provides unit testing and integration test capabilities with Spring Test, Mockito and JUnit. One thing we are missing is the version for these dependencies. We will add Spring Boot Starter Parent as the parent pom in the pom.xml

|  |
| --- |
| <**parent**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-parent</**artifactId**>  <**version**>2.3.1.RELEASE</**version**>  <**relativePath** /> <!-- lookup parent from repository --> </**parent**> |

Let’s configure the Java version to use as 1.8

|  |
| --- |
| <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  <java.version>1.8</java.version>  <maven-jar-plugin.version>3.1.1</maven-jar-plugin.version> </properties> |

Next step is to create a Spring Boot Application class which will be the launching point of the web application.

***/src/main/java/com/in28minutes/springboot/tutorial/SpringBootWebApplication.java***

|  |
| --- |
| package com.example; import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication; @SpringBootApplication public class SpringBootWebApplication {  public static void main(String[] args) {  SpringApplication.run(SpringBootWebApplication.class, args);  } } |

All that you need to do is to add @SpringBootApplication and use SpringApplication.run() static method to launch the Spring Application context.

When you launch this class a java application, you would see that an embedded tomcat server would launch up and you are ready to add in features to this application.

**Voila!!** We have successfully completed this Exercise.