**Introduction to Spring Boot Framework**

Use Cases Manual

Sandbox Link [Spring Boot](https://share.percipio.com/cd/J676mja_Z)

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

we will explore the different options of creating Spring Boot Projects with Maven and Eclipse.

**You will learn**

* How to bootstrap a simple project with Spring Initializr?
* How to use Spring Starter Eclipse Plugin to create a simple project with Spring Boot, Maven and Eclipse?
* How to create a Spring Boot Project manually step by step?

**Tools you will need**

* Maven 3.0+ is your build tool
* Your favorite IDE. We use Eclipse.
* JDK 1.8+

**Introduction to Maven**

Why Maven?

* You don’t want to store all the libraries in your project!
* You want to tell I need A, B, C and you would want the tool to download the libraries and make them available to you.
* That’s Maven. The tool which you use to manage the libraries. If you need a new version of the library, you can change the version and your project is ready!
* Also, you don’t need to worry about what libraries your library needs to work. For example, Spring might need other libraries - logging, xml etc.
* Once you declare a dependency on Spring, Maven would download

**Spring**

**And all dependencies of Spring**

**Big Picture of Maven**

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…***

**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.optum.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**

Build Lifecycle is a sequence of steps

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

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

***Maven follows Convention over Configuration.***

**Pre-defined folder structure**

**Source Code**

${basedir}/src/main/java

${basedir}/src/main/resources

**Test Code**

${basedir}/src/test

**How does Maven Work?**

* Maven repository stores all the versions of all dependencies.
* The jar dependencies are stored on your machine in a folder called maven local repository.

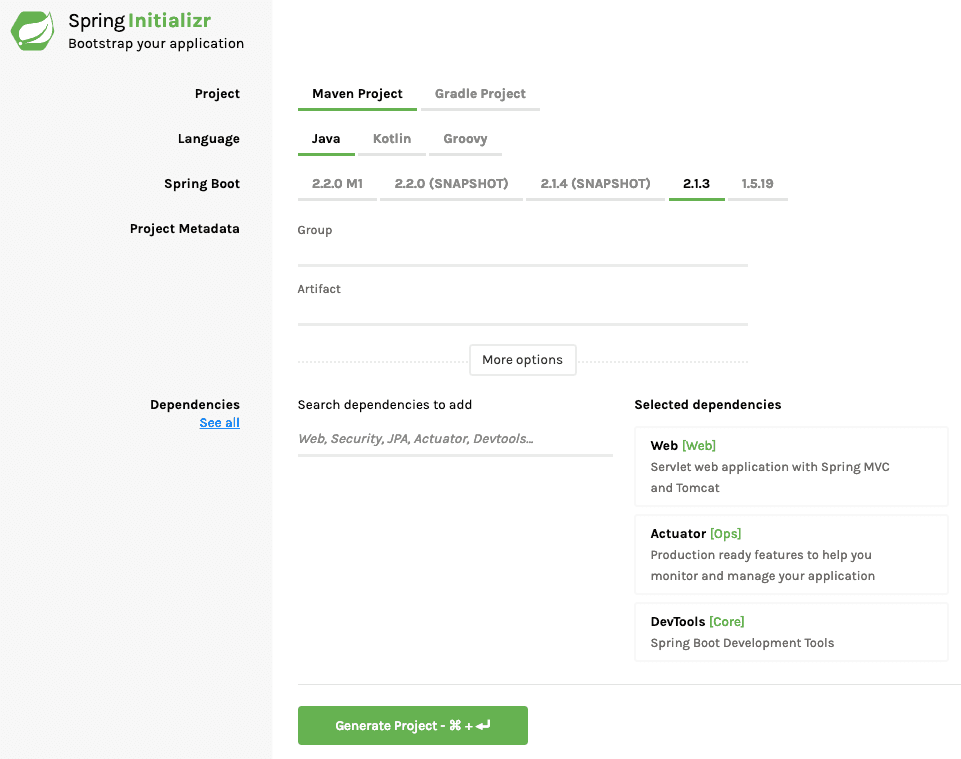
**Creating Spring Boot Projects with Eclipse and Maven**

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

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

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

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.optum.springboot as Group

Choose student-services as Artifact

Choose following dependencies

Web

Actuator

DevTools

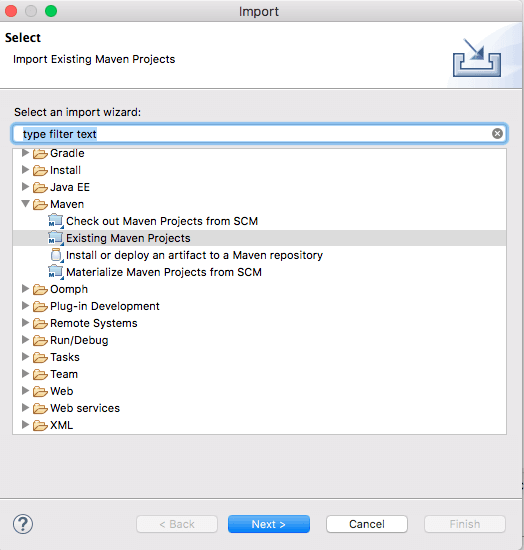
Click Generate Project.

This would download a zip file to your local machine.

**Import using Eclipse**

Unzip the zip file and extract to a folder.

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



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

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

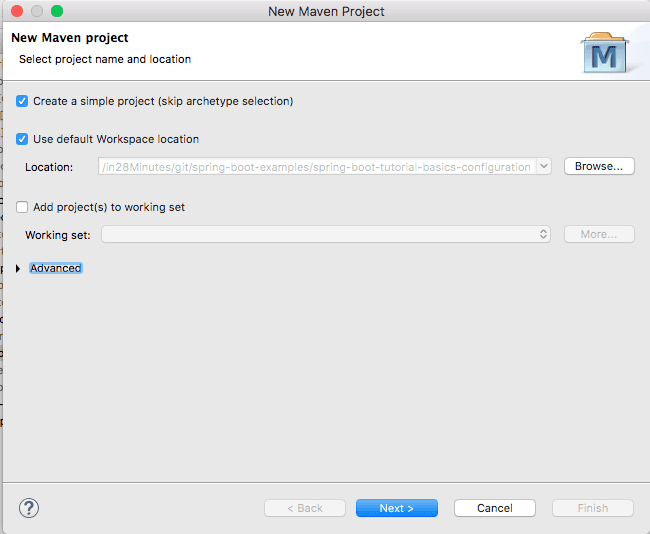
<https://spring.io/tools/sts/> all provides the complete download of STS as well as the Update Sites for STS Eclipse Plugin**.**

**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>

**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 launching point of the web application.**

**/src/main/java/com/optum/springboot/SpringBootWebApplication.java**

**package com.optum.springboot.tutorial;**

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);

}

}