

## Spring Boot Tutorial

Spring Boot Tutorial provides basic and advanced concepts of Spring Framework. Our Spring Boot Tutorial is designed for beginners and professionals both.

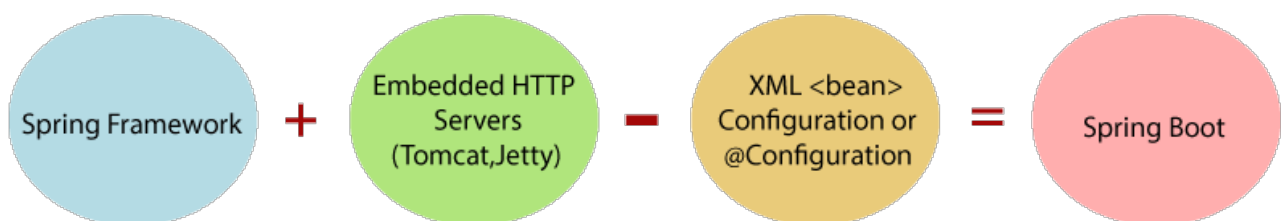
Spring Boot is a Spring module that provides the RAD (Rapid Application Development) feature to the Spring framework.

Our Spring Boot Tutorial includes all topics of Spring Boot such, as features, project, maven project, starter project wizard, Spring Initializr, CLI, applications, annotations, dependency management, properties, starters, Actuator, JPA, JDBC, etc.

## What is Spring Boot

Spring Boot is a project that is built on the top of the Spring Framework. It provides an easier and faster way to set up, configure, and run both simple and web-based applications.

It is a Spring module that provides the **RAD (*Rapid Application Development*)** feature to the Spring Framework. It is used to create a stand-alone Spring-based application that you can just run because it needs minimal Spring configuration.



In short, Spring Boot is the combination of **Spring Framework** and **Embedded Servers**.

In Spring Boot, there is no requirement for XML configuration (deployment descriptor). It uses convention over configuration software design paradigm that means it decreases the effort of the developer.

We can use Spring **STS IDE** or **Spring Initializr** to develop Spring Boot Java applications.

## Why should we use Spring Boot Framework?

We should use Spring Boot Framework because:

The dependency injection approach is used in Spring Boot.

It contains powerful database transaction management capabilities.

It simplifies integration with other Java frameworks like JPA/Hibernate ORM, Struts, etc.

It reduces the cost and development time of the application.

Along with the Spring Boot Framework, many other Spring sister projects help to build applications addressing modern business needs. There are the following Spring sister projects as follows:

**Spring Data:** It simplifies data access from the relational and **NoSQL** databases.

**Spring Batch:** It provides powerful **batch** processing.

**Spring Security:** It is a security framework that provides robust **security** to applications.

**Spring Social:** It supports integration with **social networking** like LinkedIn.

**Spring Integration:** It is an implementation of Enterprise Integration Patterns. It facilitates integration with other **enterprise applications** using lightweight messaging and declarative adapters.

## Advantages of Spring Boot

It creates **stand-alone** Spring applications that can be started using Java **-jar**.

It tests web applications easily with the help of different **Embedded** HTTP servers such as **Tomcat, Jetty**, etc. We don't need to deploy WAR files.

It provides opinionated '**starter**' POMs to simplify our Maven configuration.

It provides **production-ready** features such as **metrics, health checks**, and **externalized configuration**.

There is no requirement for **XML** configuration.

It offers a **CLI** tool for developing and testing the Spring Boot application.

It offers the number of **plug-ins**.

It also minimizes writing multiple **boilerplate codes** (the code that has to be included in many places with little or no alteration), XML configuration, and annotations.

It **increases productivity** and reduces development time.

## Limitations of Spring Boot

Spring Boot can use dependencies that are not going to be used in the application. These dependencies increase the size of the application.

## Goals of Spring Boot

The main goal of Spring Boot is to reduce **development, unit test, and integration test** time.

Provides Opinionated Development approach

Avoids defining more Annotation Configuration

Avoids writing lots of import statements

Avoids XML Configuration.

By providing or avoiding the above points, Spring Boot Framework reduces **Development time, Developer Effort, and increases productivity.**

## Prerequisite of Spring Boot

To create a Spring Boot application, following are the prerequisites. In this tutorial, we will use **Spring Tool Suite** (STS) IDE.

Java 1.8

Maven 3.0+

Spring Framework 5.0.0.BUILD-SNAPSHOT

An IDE (Spring Tool Suite) is recommended.

## Spring Boot Features

Web Development

SpringApplication

Application events and listeners

Admin features

Externalized Configuration

Properties Files

YAML Support

Type-safe Configuration

Logging

Security

### Web Development

It is a well-suited Spring module for web application development. We can easily create a self-contained HTTP application that uses embedded servers

like **Tomcat**, **Jetty**, or Undertow. We can use the **spring-boot-starter-web** module to start and run the application quickly.

## SpringApplication

The SpringApplication is a class that provides a convenient way to bootstrap a Spring application. It can be started from the main method. We can call the application just by calling a static run() method.

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

## Application Events and Listeners

Spring Boot uses events to handle the variety of tasks. It allows us to create factories file that is used to add listeners. We can refer it to using the **ApplicationListener** key.

Always create factories file in META-INF folder like **META-INF/spring.factories**.

## Admin Support

Spring Boot provides the facility to enable admin-related features for the application. It is used to access and manage applications remotely. We can enable it in the Spring Boot application by using **spring.application.admin.enabled** property.

## Externalized Configuration

Spring Boot allows us to externalize our configuration so that we can work with the same application in different environments. The application uses YAML files to externalize configuration.

## Properties Files

Spring Boot provides a rich set of **Application Properties**. So, we can use that in the properties file of our project. The properties file is used to set properties like **server-port =8082** and many others. It helps to organize application properties.

## YAML Support

It provides a convenient way of specifying the hierarchical configuration. It is a superset of JSON. The `SpringApplication` class automatically supports YAML. It is an alternative of properties file.

## Type-safe Configuration

The strong type-safe configuration is provided to govern and validate the configuration of the application. Application configuration is always a crucial task which should be type-safe. We can also use annotation provided by this library.

## Logging

Spring Boot uses Common logging for all internal logging. Logging dependencies are managed by default. We should not change logging dependencies if no customization is needed.

## Security

Spring Boot applications are spring bases web applications. So, it is secure by default with basic authentication on all HTTP endpoints. A rich set of Endpoints is available to develop a secure Spring Boot application.

## Spring Boot Version

The latest version of Spring Boot is **3.0**. It introduces a lot of new features along with some modifications and replacement.

## Spring Boot 3.0

Let's have a sneak peek at Spring Boot 3.0.

### What's New

- Infrastructure Upgrade
- Spring Framework 6.0

### What's Changed

- Configuration Properties
- Gradle Plugin
- Actuators endpoints - to monitor and interact with your application

### What's Evolving

- Security
- Metrics

The pivotal team has upgraded the **infrastructure** in which the following tools are involved:

Supports **Java 8** or above versions

Supports Apache **Tomcat 8** or above versions

Supports **Thymeleaf 3**

Supports **Hibernate 5.2**

In **Spring Framework 5**, the Pivotal team upgraded the following:

Reactive Spring

- Servlet stack

  - Servlet Container

  - Servlet API

  - Spring MVC

- Reactive Stack**

  - Netty, Servlet 3.1, Undertow

  - Reactive HTTP Layer

  - Spring WebFlux

Functional API

Kotlin Support

The latest version of Spring Boot is 2.2.1. This release of Spring Boot includes 110 fixes, dependency upgrades, and improvements.

In the Spring Boot v2.2.1, the annotation **@ConfigurationProperties** scanning is now disabled by default. We need to be explicitly opted into by adding the **@ConfigurationPropertiesScan** annotation.

## New Features

Support constructor binding for property nested inside a JavaBean

Add config property for CodecConfigurer.maxInMemorySize in WebFlux

Make test slices' type exclude filters public

Support amqp:// URLs in spring.rabbitmq.addresses

## Dependency upgrades

Some dependencies have been upgraded in Spring Boot v2.2.1 are as follows:

Mongodb 3.11.2

Spring Security 5.2.1.RELEASE

Slf4j 1.7.29

Spring Hateoas 1.0.1.RELEASE

Hibernate Validator 6.0.18.Final

Hibernate 5.4.8.Final

Jetty 9.4.22.v20191022

Spring Framework 5.2.1

Spring AMQP 2.2.1

H2 1.4.200

Spring Security 5.2

Spring Batch 4.2

Some important and widely used third-party dependencies are upgraded in this release are as follows:

Micrometer 1.3.1

Flyway 6.0.7

Elasticsearch 6.8.4

JUnit 5.5

Jackson 2.10

## Spring vs. Spring Boot vs. Spring MVC

### Spring vs. Spring Boot

**Spring:** Spring Framework is the most popular application development framework of Java. The main feature of the Spring Framework is **dependency Injection** or **Inversion of Control** (IoC). With the help of Spring Framework, we can develop a **loosely** coupled application. It is better to use if application type or characteristics are purely defined.

**Spring Boot:** Spring Boot is a module of Spring Framework. It allows us to build a stand-alone application with minimal or zero configurations. It is better to use if we want to develop a simple Spring-based application or RESTful services.

**Tight coupling:** means classes and objects are dependent on one another.

**Loose coupling:** means reducing the dependencies of a class that uses the different class directly

The primary comparison between Spring and Spring Boot are discussed below:

Spring	Spring Boot
<b>Spring Framework</b> is a widely used Java EE framework for building applications.	<b>Spring Boot Framework</b> is widely used to develop <b>REST APIs</b> .
It aims to simplify Java EE development that makes developers more productive.	It aims to shorten the code length and provide the easiest way to develop <b>Web Applications</b> .
The primary feature of the Spring Framework is <b>dependency injection</b> .	The primary feature of Spring Boot is <b>Autoconfiguration</b> . It automatically configures the classes based on the requirement.
It helps to make things simpler by allowing us to develop <b>loosely coupled</b> applications.	It helps to create a <b>stand-alone</b> application with less configuration.
The developer writes a lot of code ( <b>boilerplate code</b> ) to do the minimal task.	It <b>reduces</b> boilerplate code.
To test the Spring project, we need to set up the sever explicitly.	Spring Boot offers <b>embedded server</b> such as <b>Jetty</b> and <b>Tomcat</b> , etc.
It does not provide support for an in-memory database.	It offers several plugins for working with an embedded and <b>in-memory</b> database such as <b>H2</b> .
Developers manually define dependencies for the Spring project in <b>pom.xml</b> .	Spring Boot comes with the concept of <b>starter</b> in pom.xml file that internally takes care of downloading the dependencies <b>JARs</b> based on Spring Boot Requirement.



## Spring Boot vs. Spring MVC

**Spring Boot:** Spring Boot makes it easy to quickly bootstrap and start developing a Spring-based application. It avoids a lot of boilerplate code. It hides a lot of complexity behind the scene so that the developer can quickly get started and develop Spring-based applications easily.

**boilerplate code** : section of code that are repeated in multiple places with little to no variation

**Spring MVC:** Spring MVC is a Web MVC Framework for building web applications. It contains a lot of configuration files for various capabilities. It is an HTTP oriented web application development framework.

Spring Boot and Spring MVC exist for different purposes. The primary comparison between Spring Boot and Spring MVC are discussed below:

Spring Boot	Spring MVC
<b>Spring Boot</b> is a module of Spring for packaging the Spring-based application with sensible defaults.	<b>Spring MVC</b> is a model view controller-based web framework under the Spring framework.
It provides default configurations to build <b>Spring-powered</b> framework.	It provides <b>ready to use</b> features for building a web application.
There is no need to build configuration manually.	It requires build configuration manually.
There is <b>no requirement</b> for a deployment descriptor.	A Deployment descriptor is <b>required</b> .
It avoids boilerplate code and wraps dependencies together in a single unit.	It specifies each dependency separately.
It <b>reduces</b> development time and increases productivity.	It takes <b>more</b> time to achieve the same.

## Spring Boot Architecture

Spring Boot is a module of the Spring Framework. It is used to create stand-alone, production-grade Spring Based Applications with minimum efforts. It is developed on top of the core Spring Framework.

Spring Boot follows a layered architecture in which each layer communicates with the layer directly below or above (hierarchical structure) it.

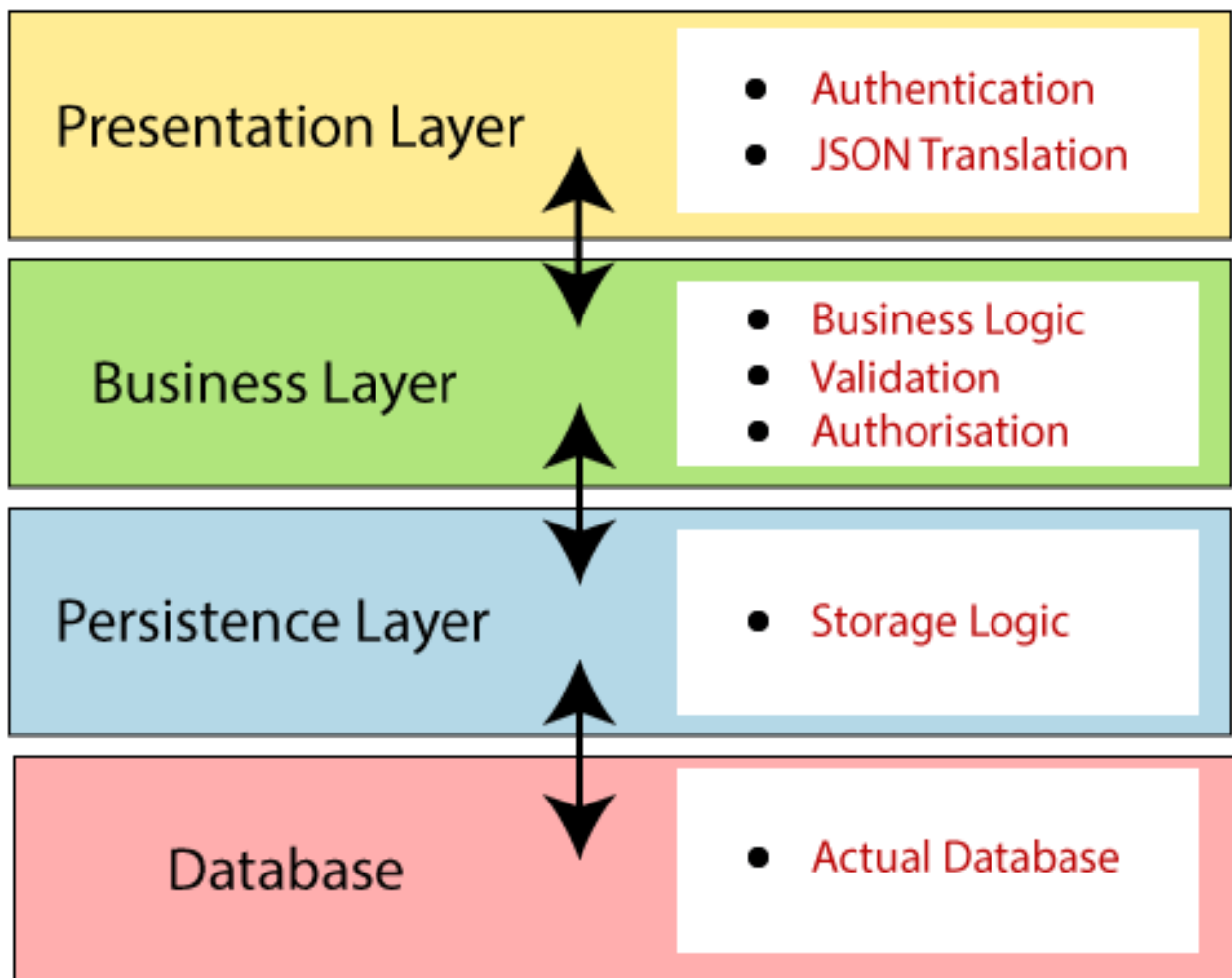
Before understanding the **Spring Boot Architecture**, we must know the different layers and classes present in it. There are **four** layers in Spring Boot are as follows:

**Presentation Layer**

**Business Layer**

**Persistence Layer**

**Database Layer**



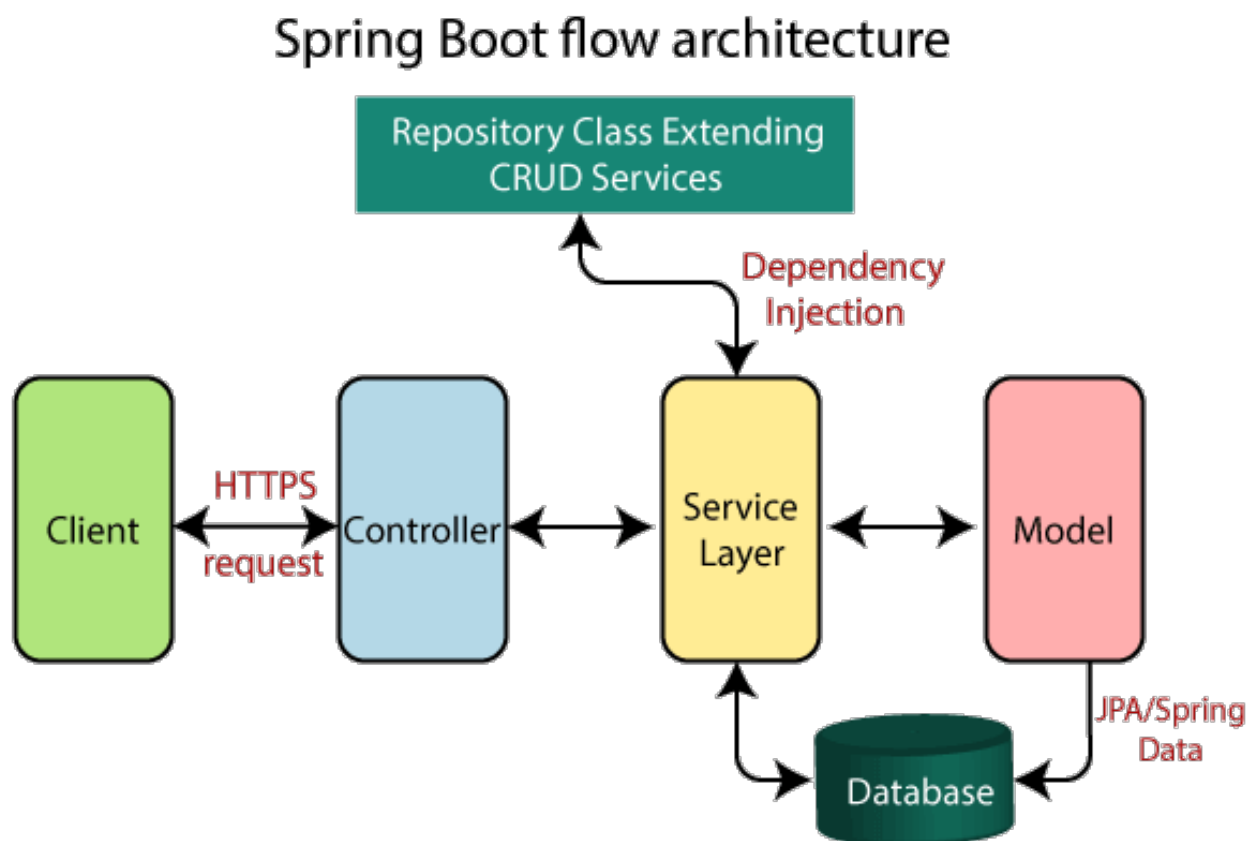
**Presentation Layer:** The presentation layer handles the HTTP requests, translates the JSON parameter to object, and authenticates the request and transfer it to the business layer. In short, it consists of **views** i.e., frontend part.

**Business Layer:** The business layer handles all the **business logic**. It consists of service classes and uses services provided by data access layers. It also performs **authorization** and **validation**.

**Persistence Layer:** The persistence layer contains all the **storage logic** and translates business objects from and to database rows.

**Database Layer:** In the database layer, **CRUD** (create, retrieve, update, delete) operations are performed.

### Spring Boot Flow Architecture



Now we have validator classes, view classes, and utility classes. Spring Boot uses all the modules of Spring-like Spring MVC, Spring Data, etc. The architecture of Spring Boot is the same as the architecture of Spring MVC, except one thing: there is no need for **DAO** and **DAOImpl** classes in Spring boot.

Creates a data access layer and performs CRUD operation.  
The client makes the HTTP requests (PUT or GET).

The request goes to the controller, and the controller maps that request and handles it. After that, it calls the service logic if required.

## Spring Initializr

**Spring Initializr** is a **web-based tool** provided by the Pivotal Web Service. With the help of **Spring Initializr**, we can easily generate the structure of the **Spring Boot Project**. It offers extensible API for creating JVM-based projects.

It also provides various options for the project that are expressed in a metadata model. The metadata model allows us to configure the list of dependencies supported by JVM and platform versions, etc. It serves its metadata in a well-known that provides necessary assistance to third-party clients.


## Supported Interface

It supports **IDE STS, IntelliJ IDEA Ultimate, NetBeans, Eclipse**. You can download the plugin from <https://github.com/AlexFalappa/nb-springboot>. If you are using VSCode, download the plugin from <https://github.com/microsoft/vscode-spring-initializr>

. Use Custom Web UI <http://start.spring.io> or <https://start-scs.cfapps.io>

. It also supports the command-line with the **Spring Boot CLI** or **cURL** or **HTTPIe**.

The following image shows the Spring Initializr UI:



Spring Initializr  
Bootstrap your application

Project

Language

Spring Boot

Project Metadata

Dependencies

Maven Project

Gradle Project

Java

Kotlin

Groovy

2.2.2 (SNAPSHOT)

2.2.1

2.1.11 (SNAPSHOT)

2.1.10

Group

com.example

Artifact

demo

Options

Name

demo

Description

Demo project for Spring Boot

Package Name

com.example.demo

Packaging

Jar

War

Java

13

11

8

Q

≡

Generate - Ctrl + G

Explore - Ctrl + Space

Share...

© 2013-2019 Pivotal Software  
start.spring.io is powered by  
[Spring Initializr](#) and [Pivotal Web Services](#)

## Generating a Project

Before creating a project, we must be friendly with UI. Spring Initializr UI has the following labels:

**Project:** It defines the **kind** of project. We can create either **Maven Project** or **Gradle Project**. We will create a **Maven Project** throughout the tutorial.  
**Language:** Spring Initializr provides the choice among three languages **Java**, **Kotlin**, and **Groovy**. Java is by default selected.

**Spring Boot:** We can select the Spring Boot **version**. The latest version is **2.6.6**.

**Project Metadata:** It contains information related to the project, such as **Group**, **Artifact**, etc. **Group** denotes the **package** name; **Artifact** denotes the **Application** name. The default **Group** name is **com.example**, and the default **Artifact** name is **demo**.

**Dependencies:** Dependencies are the collection of artifacts that we can add to our project.

There is another **Options** section that contains the following fields:

**Name:** It is the same as **Artifact**.

**Description:** In the description field, we can write a **description** of the project.

**Package Name:** It is also similar to the **Group** name.

**Packaging:** We can select the **packing** of the project. We can choose either **Jar** or **War**.

**Java:** We can select the **JVM** version which we want to use. We will use **Java 8** version throughout the tutorial.

There is a **Generate** button. When we click on the button, it starts packing the project and downloads the **Jar** or **War** file, which you have selected.

In the service layer, all the business logic performs. It performs the logic on the data that is mapped to JPA with model classes.

A JSP page is returned to the user if no error occurred.

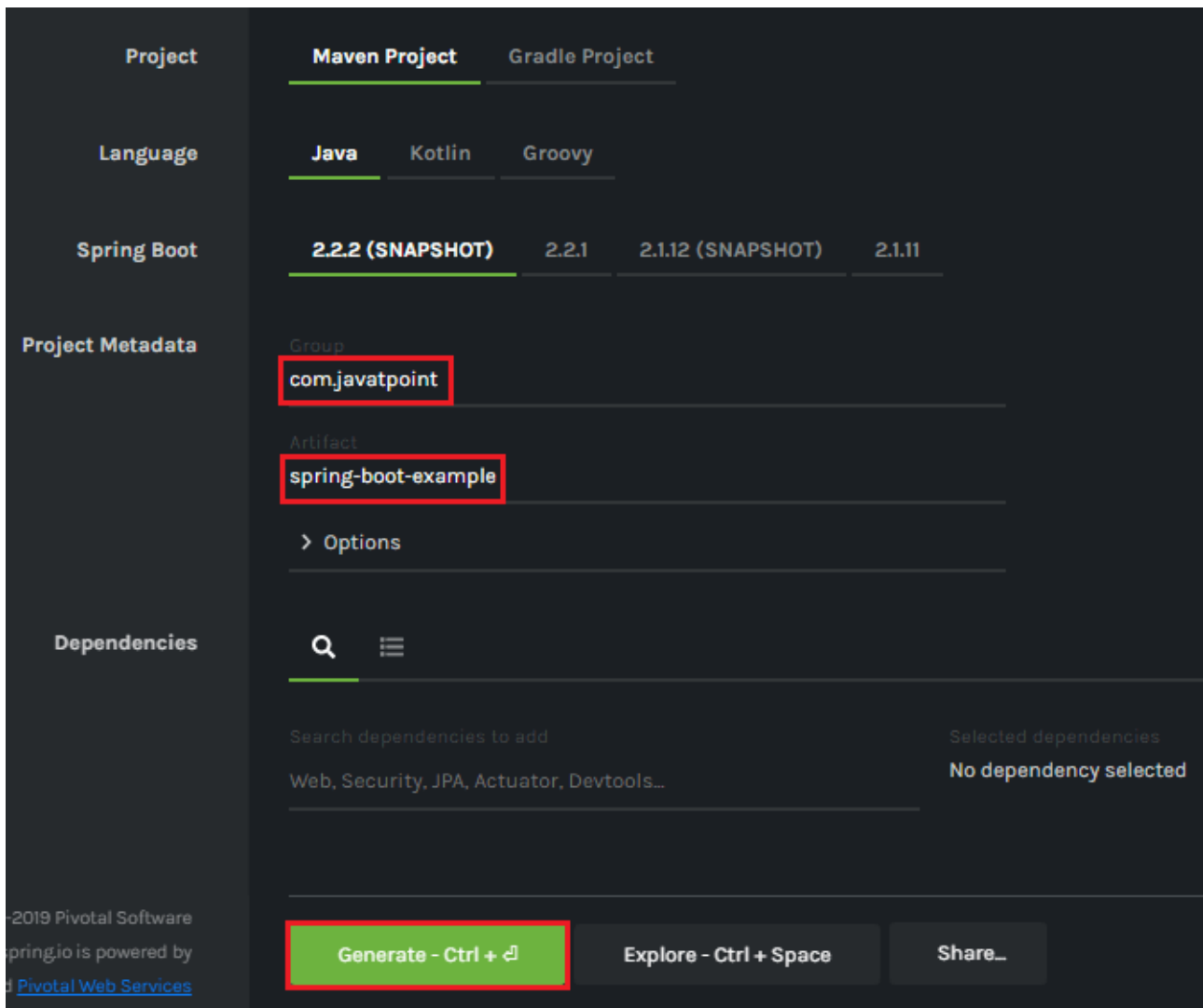
## Creating a Spring Boot Project

Following are the steps to create a simple Spring Boot Project.

**Step 1:** Open the Spring initializr <https://start.spring.io>.

**Step 2:** Provide the **Group** and **Artifact** name. We have provided **Group** name **com.javatpoint** and **Artifact** **spring-boot-example**.

**Step 3:** Now click on the **Generate** button.



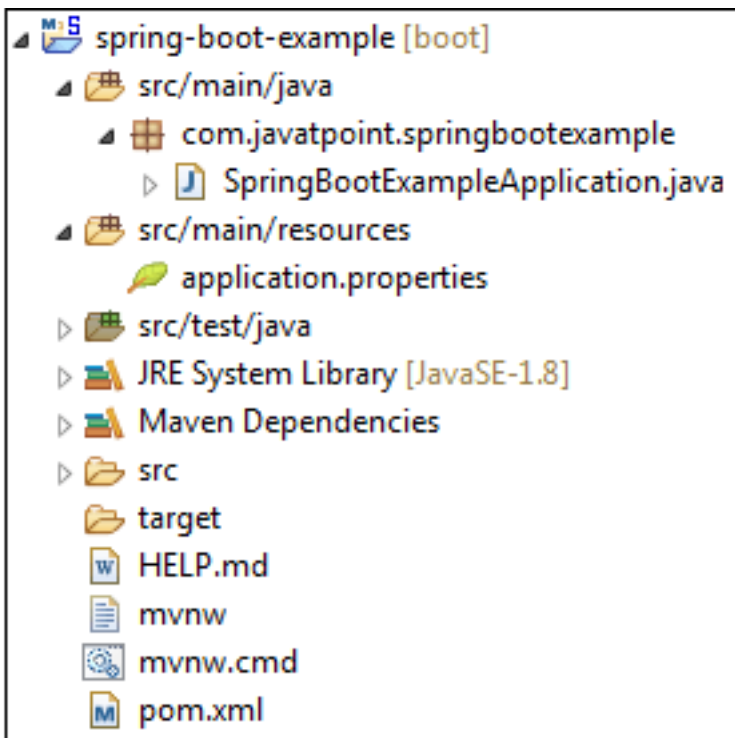
When we click on the Generate button, it starts packing the project in a **.rar** file and downloads the project.

**Step 4:** Extract the **RAR** file.

**Step 5:** Import the folder.

File -> Import -> Existing Maven Project -> Next -> Browse -> Select the project -> Finish

It takes some time to import the project. When the project imports successfully, we can see the project directory in the **Package Explorer**. The following image shows the project directory:



## SpringBootExampleApplication.java

```
package com.javatpoint.springbootexample;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class SpringBootExampleApplication
{
    public static void main(String[] args)
    {
        SpringApplication.run(SpringBootExampleApplication.class, args);
    }
}
```

## 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>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
```



```
<version>2.2.2.BUILD-SNAPSHOT</version>
<relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>com.javatpoint</groupId>
<artifactId>spring-boot-example</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>spring-boot-example</name>
<description>Demo project for Spring Boot</description>
<properties>
<java.version>1.8</java.version>
</properties>
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
<exclusions>
<exclusion>
<groupId>org.junit.vintage</groupId>
<artifactId>junit-vintage-engine</artifactId>
</exclusion>
</exclusions>
</dependency>
</dependencies>
<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>
<repositories>
<repository>
<id>spring-milestones</id>
<name>Spring Milestones</name>
<url>https://repo.spring.io/milestone</url>
</repository>
```

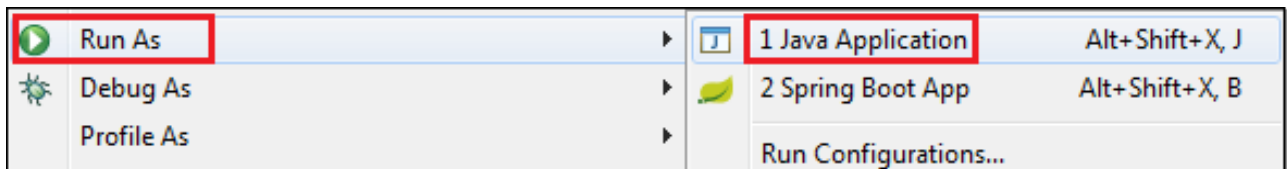
```

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

```

**Step 6:** Run the **SpringBootExampleApplication.java** file.

Right-click on the file -> Run As -> Java Applications



The following image shows the application runs successfully.

```

: Starting SpringBootExampleApplication on Anubhav-PC with PID 5096 (C:\Users\Anubhav
: No active profile set, falling back to default profiles: default
: Started SpringBootExampleApplication in 41.147 seconds (JVM running for 1856.017)

```