First Spring Boot Project

Table of Contents

[What is Spring Boot? 2](#_Toc57378168)

[Advantages of Spring Boot 2](#_Toc57378169)

[Configure a Spring Boot Project 3](#_Toc57378170)

[Eclipse 3](#_Toc57378171)

[Spring Tool Suite 3](#_Toc57378172)

[Project Structure 4](#_Toc57378173)

[JSP View in Spring Boot 6](#_Toc57378174)

[JPA (Java Persistent API) 6](#_Toc57378175)

[What is JPA? 6](#_Toc57378176)

[Why ORM? 6](#_Toc57378177)

[JPA API 6](#_Toc57378178)

[CRUD Operations 6](#_Toc57378179)

# Prerequisites

Following softwares and tools must be installed on the system before creating any Spring Boot Application

1. Any suitable Java version (LTS preferred)
2. Maven 3.0+
3. Eclipse/STS: **Spring Tool Suite** is recommended.

# Ways of creating a Spring Boot Application

There are following ways to create a Spring Boot project:

1. With Eclipse (old method)

Create a maven project in **Eclipse** and add starter dependencies as jars

1. Use Spring Initializr and Eclipse/STS
2. Use Spring Tool Suite IDE
3. Spring Boot Command Line Interface
4. Using Spring plugin in Eclipse

## Eclipse

1. Create a Maven project in Eclipse and add starter dependencies in the form of jar files
2. Use Spring plugin in Eclipse
3. By Spring Initializer: Eclipse is the most used IDE by Java Developers, so Spring Boot provides an initializer to make things easy for eclipse users.

### Steps to create Spring Boot Application in Eclipse

1. Step 1: Search for Spring Boot Initializer on the internet
2. Step 2: Go to <https://start.spring.io/>
3. Step 3: Choose the options for the project:
   1. Project – Maven Project,
   2. Language – Java,
   3. Spring Boot – 2.3.5,
   4. Project Metadata,
   5. Packaging – Jar,
   6. Java – 11
4. Step 4: Add Dependencies
   1. Web
5. Step 5: Click Generate: The project will be downloaded as a zip file. Extract it.
6. Open Eclipse and Import the extracted project as “Existing maven Project”

## Spring Tool Suite (STS)

1. Spring Tool Suite is an IDE to develop Spring applications.
2. It is an Eclipse-based development environment.
3. It provides a ready-to-use environment to implement, run, deploy, and debug the application.
4. It validates our application and provides quick fixes for the applications.

### Steps to create Spring Boot Application in STS

1. Step 1: Search for STS download
2. Step 2: Go to: <https://spring.io/tools>
3. Step 3: Download 64 bit STS windows version (here 4.8.1)
4. Step 4: Execute the file by double clicking it
5. Step 5: Go to File>> New >> Spring Starter Project
6. Step 6: Choose the options for the project:
   1. Project – Maven Project,
   2. Language – Java,
   3. Spring Boot – 2.5.2,
   4. Project Metadata, such as project name and package: **spring.boot.demo**
   5. Packaging – Jar,
   6. Java - 17
7. Step 4: Add Dependencies
   1. Web

**Note: We will use STS in our course to gain familiarity with it**

### Project Structure

#### Java Source Code

1. All the java source code must reside inside **src/main/java** folder.
2. The package we mentioned while making the project (here spring.boot.demo) is important.
3. It will be considered as a base package for all the java classes hereafter, i.e. all the Java files **must** either lie inside this package or **must be** written inside the sub package of this package for Spring Boot to scan them.
4. In this example, the class SpringDemoApplication is the java class from where the execution will begin.
5. SpringDemoApplication is annotated with **@SpringBootApplication** annotation. This annotation is the combination of following annotations:
   1. @Configuration,
   2. @EnableAutoConfiguration, and
   3. @ComponentScan
6. Just run the application as –
   1. Java Application in Eclipse or
   2. Run As >> Spring Boot App in STS.

#### Resources

1. All the static resources, configurations and themes are kept in this folder
2. **Static**: All the static resources such as HTML, CSS, JavaScript, images and other media files are contained in static folder.
3. **Templates**: It will contain any predefined themes such as for example: Thymeleaf.
4. File **application.properties**: All the properties such database configurations, context path, file path configurations must be done in application.properties file with the help of **key=value** pairs

#### Dependencies

We add **starter template jars** to our spring boot application such as: **spring-boot-starter-web**, **spring-boot-starter-data-jpa**, etc.

When we add starter jars, then Spring Boot pulls all the related jars. These Jar files contain the file spring.factories in META-INF folder, i.e. META-INF/spring.factories

If we use JPA, JPA configurations become active.

Spring boot scans the class path and if it finds JPA, all the configurations relating to JPA in spring.factories will become active, this will download spring.orm, hibernate, mysql connector etc

This is known as “Opinionated Defaults Configuration”

## JSP View in Spring Boot

In order to use JSP in a Spring Boot application, follow the below steps

1. Include **tomcat embedded jasper** maven dependency in pom.xml
2. Create some folders: a **webapp>>views** folder in **src>>main** folder
3. Place the JSPs in it
4. Set the following properties in **application.properties**
   1. spring.mvc.view.prefix=/views/
   2. spring.mvc.view.suffix=.jsp
5. Create a controller with a request handler to show JSP
6. Fire the request through the URL

# JPA (Java Persistent API)

## What is JPA?

* By name, JPA stands for Java Persistent API, i.e. it is an Application Programming Interface (a set of interfaces and classes) that is used to persist (manipulate and store) data.
* In other words JPA is a specification. It is a specification that lays down the specifications for Object Relationship Mapping (ORM). It is implemented by Hibernate, Open JPA, Eclipse link etc. Hibernate is the most used one.
* JPA specification is given by Oracle, Hibernate is written by Gavin King

## Why ORM?

* With ORM, developers can transact with database without having to write tedious SQL Queries
* Reduction of manual database handling saves time plus helps developers to write less erroneous code.
* Less errors mean less time is consumed in testing and development cycles

## JPA API

It provides following 2 interfaces:

1. EntityManagerFactory: It will provide the object of EntityManager object
2. EntityManager: It will provide methods to do CRUD

# CRUD Operations

1. Step 1: Create a new Spring Boot Starter project in STS
2. Step 2: Add the following dependencies in the Spring Boot Project:

* Spring Boot Starter Data JPA
* MySQL Connector

1. Step 3: Fill in the project details
2. Step 4: Create an entity (for ex User, **Book**, Person, Product etc.) that we need saving. Just remember to create it inside the parent package.
3. Step 5: Create an interface: BookRepository and extend it with CRUDRepository interface
   1. This BookRepository will be same as DAO in the data layer.
   2. It will extend an interface CRUDRepository
   3. CRUDRepository takes in 2 generic parameters: 1st: Entity Class and 2nd **type** of Entity Id. For Example: CRUDRepository<Book, Integer>
   4. We can also make it extend the child interface of CRUDRepository: JPARepository, it has some extra features.
4. Configure datasource in application.properties files
   1. spring.datasource.name=test
   2. spring.datasource.url=jdbc:mysql://localhost:3306/spring\_boot\_crud?serverTimezone=UTC
   3. spring.datasource.username=root
   4. spring.datasource.password=1234
   5. spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
   6. spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MySQL55Dialect
   7. spring.jpa.hibernate.ddl-auto=update
5. Get the ApplicationContext from SpringApplication.run() method
6. Get an object of BookRepository from applicationContext.getBean() method
7. Use save method on BookRepository object to save data