Springs Boot

Table Of Contents

[1. What is spring 5](#_Toc518394910)

[2. What is Spring Boot 5](#_Toc518394911)

[3. Maven 5](#_Toc518394912)

[4. Creating a Spring Boot Project 5](#_Toc518394913)

[5. Starting a spring boot application 7](#_Toc518394914)

[6. Spring Boot Application start-up (Internal Actions) 8](#_Toc518394915)

[7. Creating a Controller Class 8](#_Toc518394916)

[8. Returning Objects from Controller 10](#_Toc518394917)

[9. Internal Actions of Spring Boot 11](#_Toc518394918)

[10. Embedded Tomcat Server 11](#_Toc518394919)

[11. Working of MVC in Spring Boot 12](#_Toc518394920)

[12. Working of MVC in Spring Boot 13](#_Toc518394921)

[13. Creating the Business Service 14](#_Toc518394922)

[14. Business Service with Argument 15](#_Toc518394923)

[15. Creating a resource using POST request type 16](#_Toc518394924)

[16. Implementing PUT and Delete Resource 18](#_Toc518394925)

[17. Starting Spring Boot Application 19](#_Toc518394926)

[18. Configuring Spring Boot Application 20](#_Toc518394927)

[19. Introduction to JPA and CRUD operation 20](#_Toc518394928)

[20. Packaging and deploying Spring Boot 22](#_Toc518394929)

[21. Spring Boot Actuator 22](#_Toc518394930)

[22. @SpringBootApplication 22](#_Toc518394931)

[23. Main Method significance 23](#_Toc518394932)

# What is spring

Spring Boot Basically provides features that allow programmers to concentrate on the business logic rather spending time on the configuration of the application. Below are few of the features provided by Spring.

1. Application Frame Work.
2. Programming and Configuration model - Such as configuration on the deployment of the application.
3. Infrastructure support - Such as establishing connection to Database.

# What is Spring Boot

1. *Opiniated* – It basically provides a default Configuration that states here is an abstract structure for a java application.
2. *Convention over Configuration* – Spring Boot Comes with a series of default configuration that suits most of the business purpose hence saving time spent on the configuration.
3. *Standalone*.
4. *Production Ready*.

# Maven

Maven is a build and dependency management tool that is used to build java projects. The configuration information about a project is provided to maven through the POM.xml file (Project Object Management) which contains all the libraries and external JAR information required for building the project.

Using Maven we can provide the configuration to automatically download and add the JARS from the Maven repository to the build path, which save the time incurred in doing it manually.

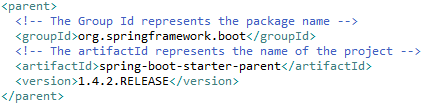
# Creating a Spring Boot Project

On the Spring Tool Suite select New -> Maven Project and in the prompt select the “Create Simple Project” by doing so a simple maven project gets created with default configuration. Once the details are provided Maven automatically download the respective packages, JARS and creates the project.

In order to configure the current project as SPRING BOOT project the “pom.xml” file has to be configured accordingly with the respective information.

The configuration is as follows.

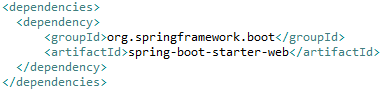
1. Modify the pom.xml file to add the parent project which can be done through the <parent> tag with the following information.



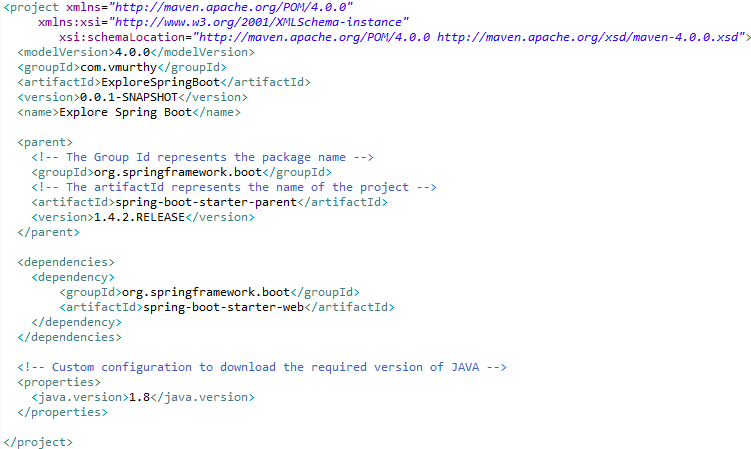
*Where the*

* **groupId**: Represents the name of the package of the parent project.
* **ArtifactId**: The name of the parent project.

1. In order to further download all the JARS required for the Web Application we can either a list of all the required JARS or we can add the default dependency which in turn downloads the required JAR for a WEB application.



*Complete pom.xml file:*



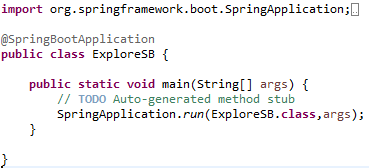
# Starting a spring boot application

In order to start spring boot application as a standalone web app, we create a class containing the main method as usual and in order to mark the incorporating class as a spring boot application the class will be annotated with “@*SpringBootApplication*”.

Once it is done the next step is to implement steps to instruct the spring boot to start the current application, create a servlet container and host the current application on it. The way it can be done is by using the static method “SpringApplication.run(arg1, arg2)” which takes in 2 argument. First one being the name of the class containing the main method and the second representing the parameters to the class.

The SpringApplication.run() method is used to start the spring application from a class during the course of which it does the following tasks.

1. Create the appropriate ApplicationContext.
2. Register a CommandLinePropertySource to expose command line arguments as spring properties.
3. Refresh the Application context and load all the singleton beans.
4. Trigger any CommandLineRunner beans.



# Spring Boot Application start-up (Internal Actions)

During the start of a spring boot application a sequence if internal action takes place to provide the expected behaviour.

1. Sets up the default configuration.
2. Starts the spring application context.
3. Performs class path scan during the course of which it identifies the intent of all the containing classes. For example we may have a class which is annotated as @*Service* once the scan is complete spring boot would now where to look for business service classes. And say we have a class annotated with @*Controller*, the spring boot would know the class that needs to be invoked when the flow is to be sent to a controller class.
4. As a final step it would start the Tomcat server.

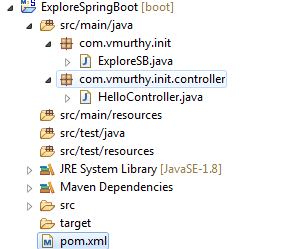
# Creating a Controller Class

The following are the steps to be followed while creating a controller class in Spring Boot Application.

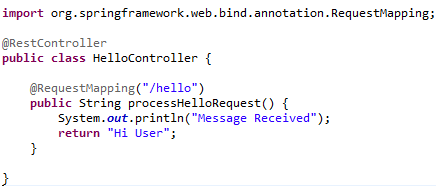
1. First create a class having implementation to depict the behaviour of a controller.
2. Annotated the class with “*@RestController*” to let spring boot be aware of the intent of the class during class path scan.
3. Create a method annotated as *“@RequestMapping*” defining the trigger during which the method will be executed. By default “*@RequestMapping”* maps to the “*get”* request in case of a “put” or “post” the same has to be explicitly mentioned in the annotation.

Note: The controller class should always be after the package containing the main class. Say the main class is contained in the package “*com.vmurthy.init*” the controller class is to be placed in the package “*com.vmurthy.init.hello”.*

Project Structure:



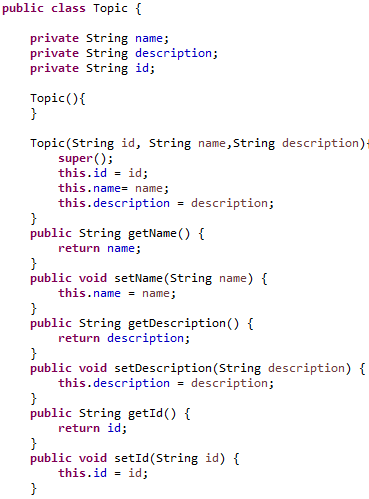
*Controller Class:*



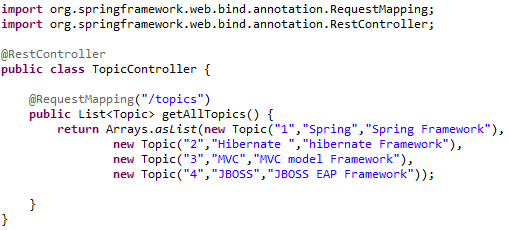
# Returning Objects from Controller

The below example illustrates a method call returning a list of objects.

*File 1*: *Topic.java* : Containing the model of the object being returned by the application.



*File 2:* TopicController.java : Providing the implementation of the controller class defining the trigger access all the topics.



*Output*:

*[{"name":"Spring","description":"Spring Framework","id":"1"},{"name":"Hibernate ","description":"hibernate Framework","id":"2"},{"name":"MVC","description":"MVC model Framework","id":"3"},{"name":"JBOSS","description":"JBOSS EAP Framework","id":"4"}]*

When any method returns the object the spring MVC automatically converts the object into JSON and sends it to the calling system. The JSON has the key as the property name and the value as same as what is contained by the properties.

# Internal Actions of Spring Boot

When we say default configuration what happens is that based on the parent version mentioned in the pom.xml file there exists a predefined list of JARS gets downloaded and tagged to the build path.

This predefined list of JARS is termed as ***Bill of Material*** and is designed in such a way that they are compatible with one another and helps us in building the business application as desired.

The element parent defines the version of JARS to be downloaded while the element “dependency” represents the list of JARS to be downloaded.

# Embedded Tomcat Server

For developing a spring boot project the programmer need not download and configure the Tomcat server as it is packaged as part of the initial configuration or the parent project. Here is the list of advantages by having the tomcat server embedded within the spring boot parent project.

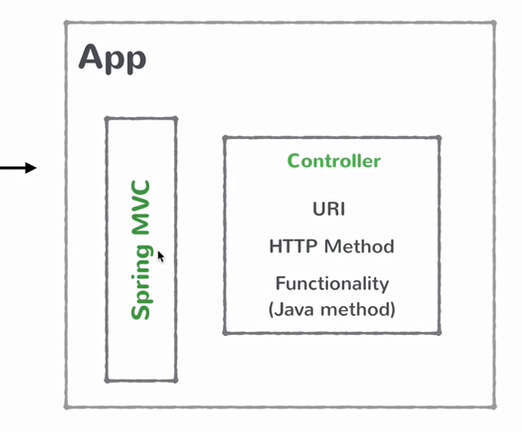
1. Convenience - As it saves the times incurred in the process of downloading and configuring the host server.
2. It enables the project to be executed as a standalone project.
3. Useful for Micro service architecture – that is in the event say we have 10 micro service using spring boot framework save the time spent in deploying them individually.

Note: It is not mandatory that any spring boot application should employ Tomcat Server it is just the default configuration and it can be modified to use the appropriate server.

# Working of MVC in Spring Boot

Every method that is defined as controller will have two important configurations first one is the URI that triggers the method and the second one is the type of the HTTP request (get, post, put).

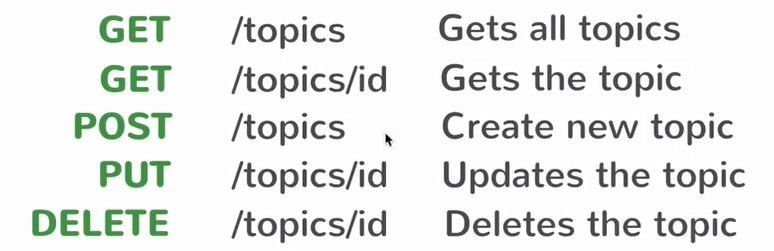
So when a request is received Spring MVC component looks up the build path and identifies the method that is to triggered for processing the request once the method return an object the spring MVC converts the data into JSON object and sends it over to the calling system.



# Working of MVC in Spring Boot

During the early stages of developing a REST web service, the first set of entity to be identified is the resources which define the API supported by the web service.

Here is t list of resources support by Course API.

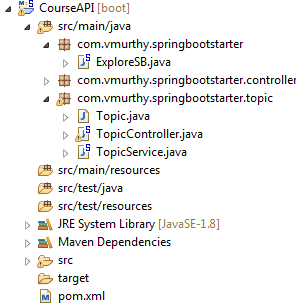


# Creating the Business Service

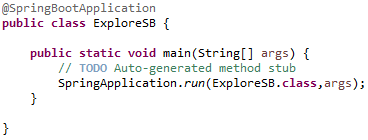
All the classes that house the business logic of the application will be annotated with @Service annotation.

When we have any field within the business service class annotated with “@*Autowired*” the spring instantiates a single instance that is by default the scope is “Singleton”.

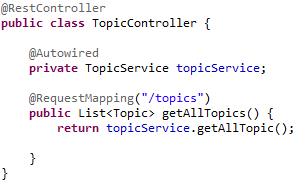
Project Structure:



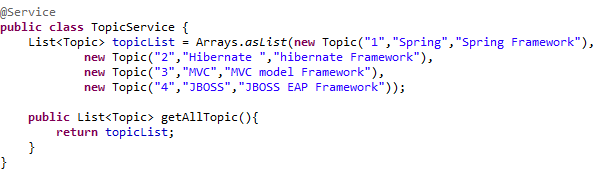
File 1: ExploreSB.java



File 2: TopicController.java



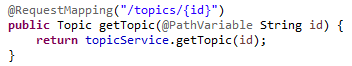
File 3: TopicService.java



# Business Service with Argument

In order for any business service to accept an argument and perform processing based on the argument in order to provide the result the incoming value should be mapped to the argument of the method.

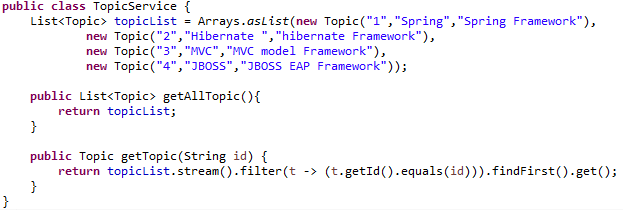
The mapping of the value from the request to the method can be done in the following ways using the @*PathVariable* annotation.



The value being sent will be marked within the curly braces in our case the value will be passed in {id} the name being same as that of the argument in the method. If the name is different the mapping can be achieved as follows.

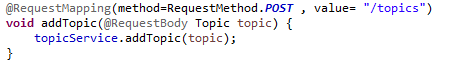


File 1: TopicService.java



# Creating a resource using POST request type

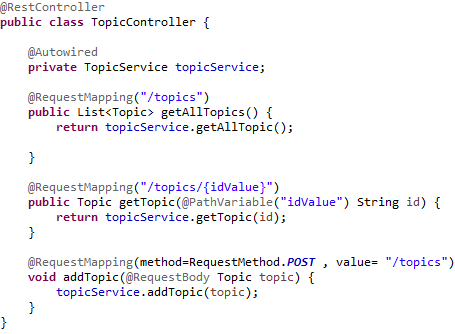
The following is the syntax to process data from the request method of type POST.



In the above mentioned snippet of code the attribute “method” defines the type of the request method and the value represents the URI that triggers the method.

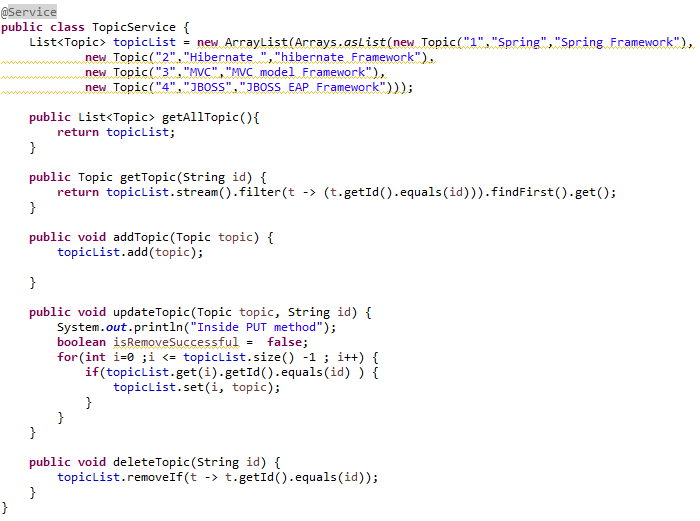
@*RequestBody* indicates that the request body need which is a JSON object should be converted into the object of type “Topic” for processing.

File Name: TopicController.java



# Implementing PUT and Delete Resource

File TopicService.java



File: TopicController.java



# Starting Spring Boot Application

Following are the ways in which Spring Boot Application can be started.

1. Spring INITIALZE

This is an online UI through which the dependency that are required for the project can be configured which includes the JARS Upon completing the configuration the downloadable file contains the base version on top of which the actual business service application can be built.

1. Spring Boot CLI (Command Line interface)

The CLI allows programmers to run various tests using groovy scripts. Say we have a groovy script defining the implementation of the controller class. That is just creating the groovy script and the rest of the required classes will be generated by the CLI and the output can be tested from a POST MAN or Browser as we test a spring boot application.

1. Spring Tool Suite IDE

Select the New Spring Starter project which would provide the same set of options as that of spring INITILZE the only difference is that it can be done through the IDE.

# Configuring Spring Boot Application

The spring boot project created through online or through CLI through or through Spring Test Suite will have the default configuration which can be modified as per the programmers need through the “*application.properties*” file.

As for determining the list of properties that can be modified the link <https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html> provides the information.

# Introduction to JPA and CRUD operation

The Spring boot comes with an out of the box interface “***CrudOperation***” with implementation which provides methods to perform standard operation on the data base like insert, update, delete, retrieve all, retrieve the required data.

In our example we will create a new interface called TopicRepository which extends CrudOperation interface so that we can define our custom data base operation methods on top of the predefined ones.

For requirement where the out of the box method in the CrudInterface does not pull the required data we can add custom methods with a specific naming convention and have JPA to provide the implementation for the same.

The method needs to be added in the “***CourseRepository***” interface which extends the “CrudRepository” built in interface.

Say For example if we want to find all the courses by name the naming convention for the method would be like:

findByName() -> where “find” would be the keywork to let the JPA know that the intent is to identify a record, the “By” refers to the criteria and “Name” refers to the property name of the entity class basically refers to the column to be used for filtering.

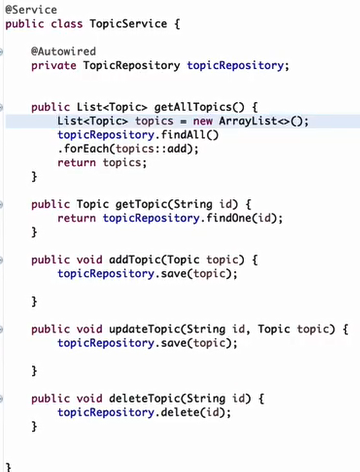
Example: To find all the records by description : findByDescription()

Example: to find all the course records by topic id: findByTopicId()

*Where*

* Topic – class
* Id - > property of the class Topic.

File 1: TopicService.java



# Packaging and deploying Spring Boot

In order to build and install the spring boot application through the command go to the project folder and execute the command: ***mvn clean install*** , this would produce a JAR file in a folder named “target”which can be installed using the command : ***java –jar target/project\_name.jar***

If the outcome of the build is to be produced in the form of a WAR file the “packaging” property in the pom.xml file has be modified into value “war”.

# Spring Boot Actuator

It Produces Production ready feature to help monitor and manage your application.

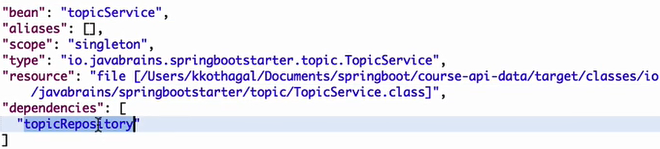
1. Resource : ***health***: called as : <http://localhost:8080/health>

Produces a JSON providing the data on the application status, disk space, data base used and such data the application.

1. Resource : ***beans*** : called as : <http://localhost:8080/beans>

Provides the list of beans available in the Spring boot application.

***Output:***



More Info is found on:



# @SpringBootApplication

The annotation @Spring boot application annotation enables the following features.

1. Enable auto configuration as done by the *@EnableAutoConfiguration*  (Spring boot application tends to automatically configure your spring application based on the JAR dependencies added but in the event if there is any custom configuration the default configuration is disabled.)
2. Enables @ComponentScan on all the application classes.
3. Performs the action of @Configuration: allow to register extra beans in the context.

# Main Method significance

The reason we use main method in a spring boot application is that if the application is packaged as WAR file, then it would be deployed on a server which takes care of executing it. But in a spring boot application that is packaged as JAR it also contains an embedded tomcat server and in order to initiate the application we have the main method.

# @RestController vs. @Controller

The @RestController annotation is very much similar to @Controller incorporating @Responsebody(which indicates that the method writes the response in the form of an object).

Code Snippet:

