Spring framework is actually an enterprise java framework which lets us write enterprises java applications.

Spring boot is basically a way in which we can bootstrap or quickly start up a simple spring application.

Spring boot gives us Opinionated, Convention over configuration, Stand alone, Production ready

In order to run the spring boot the version of java must be equal or higher than 8 to find that In CMD

Type **java -version.**

In order to know whether the java home is set or not we need type the command in CMD as

**echo $JAVA\_HOME**.

In STS we need to tell that application is a spring application by mentioning the annotation above the class which consists of main method as **@SpringBootApplication** and this annotation also tells that it is the starting point for our spring boot application.

**SpringApplication.run(source, args); -**

It tells to springboot to start the application which is the FirstDemo App and then create a servlet container and host this application in that servlet container and makes it available.

**Source – first argument is the class which has main method that’s annotated with a springBoot application.**

**Args – second argument is the arguments that we are passing to the main method.**

**SpringApplication.run(FirstDemo.class, args);**

1.Sets up default configuration

2.Starts Spring application context

3.Performs class path scan

4.Starts Tomcat server.

Standalone means we don’t have to create a servlet container.

**Controller:-**  It is basically a java class that has certain annotations marked in it. This annotations lets spring know what is the URL that we are mapping into.

No arg constructor makes the object easier to initialize.

@RequestMapping(value = "/topics")

**public** List<Topic> method1()

{

**return** Arrays.*asList*(**new** Topic("1", "Lakshman", "Analyst"),

**new** Topic("2", "Sai Ram", "Developer"),

**new** Topic("3", "Pavani", "Homemaker"));

}

Dependency section in maven pom file tells maven what jars to download and the parent section configures what versions of those jars to download.

Embedded Tomcat Server :-

1.Convenience

2.Servlet container config is now application config

3.Standalone application

4.Useful for microservices architecture.

**Business Service:-** In business service we place the code which we want to give to methods every time. So there will be no need to create a code everytime.

In spring business services are typically singleton’s. when the application starts up spring creates an instance of this service and then keeps that in its memory.

We declare the class as business service class by using the annotation **@Service.**

All the classes which have @Service

**DELETE request:-**

list.removeIf(t -> t.getId().equals(id));

**Swagger to Spring Boot :-**

Swagger is the vast set of open source tools that can help us with API development in general. It has tons of tools to help us with developing API, working with API is making calls to them and finally Documenting API.

Adding Swagger to Spring Boot:-

1. Getting the Swagger to Spring dependency
2. Enabling Swagger in our code
3. Configuring Swagger
4. Adding details as annotations to APIs

AddressBookResource:- It basically has three API endpoints

1. Getting all the contacts

2. Getting contact given an ID

3. Posting the contact so that it gets added to the list.

<http://localhost:8080/swagger-ui.html>

localhost:8080/v2/api-docs

**@GetMapping** is a composed annotation that acts as a shortcut for **@RequestMapping(method = RequestMethod.GET).**

**@GetMapping** is the newer annotaion. It supports **consumes**.

**@PostMapping** is a specialized version of @RequestMapping annotation that acts as a shortcut for **@RequestMapping(method = RequestMethod. POST).**

v2/api-docs

The way to customize a swagger is by creating instance of an object called a docket.

Docket is an object that’s going to contain all the customized properties that you intend for swagger to pick up when it’s generating those documentation.

Spring boot also has a Command Line Interface (CLI) which lets us to create the spring applications. It allows us to run groovy scripts, which means that we have a familiar java-like syntax without so much boilerplate code.

Spring-derby which is the database that lives in our classpath

**DATABASE:-**

1.@Entity – defines the objects of the class are to be relational database table.

In relational database every table must have primary key. In order to mention primary key in class we have to use the annotation of free type @Id for String id.

2. In order to use the TopicService class methods in RDBMS we have to create the interface which extends the CrudRepository.

CrudRepository will have logic for any entity class and it is a generic type of two parameters

1. Name of the entity class that we are working with here in my case is Topic
2. Type of Id here in my case is String

For doing crud operations on any entity class we don’t have to implement the data layer. We just use the CrudRepository that comes with spring JPA.

findAll() – Runs the query to get all the topics convert each of those rows into topic instances and get it back.

Save() – used for add the field into RDMS and also for updating purposes if we give the id and topic

As the arguments then it will check for the id, if it finds then it will update else it will insert the record.

findOne()

delete()

Actuator dependency figures out what are those matrix and makes them available as a new endpoint

O auth

The relationship between the entities is based on two factors

1.Multiplicity – One instance of an entity is related to how many instances of another entity.

2.Directionality – From which side of the relationship do we access the other.

The cascade attribute transfers operations done on one object onto its related child objects.

The mappedBy property is used to tell the persistence provider which variable we are using to represent the parent class in our child class. This property is needed when we use a Bi-directional relationship.

Composite key – A Composite key is a combination of two or more columns whose values together make a unique value. (@Embeddable)

If a single column is considered as the primary key, then @id is used and when the primary key consosts of multiple columns these columns need to be grouped into a different object.

Inheritance Mapping Strategies are :-

1.Single table

2.Joined Table

3.Table-Per-Class.

@Inheritance annotation in the POJO class defines the inheritance strategy.