Top 75+ Spring Boot Interview Questions in 2025

November 8, 2023

Spring Boot is a Java-based framework used for creating stand-alone, production-grade Spring-based applications. It provides a wide range of features that make it easier to develop and deploy Spring-based applications. This blog talks about the top spring boot interview questions that will help you ace your upcoming interview sessions.

The spring boot interview questions are divided into different categories. Let’s get started!

Yes, you are in the right place. In this article, we have listed all the latest and most frequently asked spring boot interview questions with proper explanations and examples that help you understand the overall concept from the scratch. You can also use the code provided for some of the questions and run it on your machines to get better clarity on the concepts.

This article helps you to go through all the major spring boot interview questions and attend the interview confidently. This article also covers a lot of spring boot interview questions for freshers and experienced.

Spring Boot Interview Questions for Freshers

With the help of Spring boot interview questions for freshers, candidates can prepare for their interviews and increase their chances of getting hired. This guide covers some of the most commonly asked Spring boot interview questions that freshers are likely to encounter during their interviews.

1. What is Spring Boot?

Spring Boot is called a microservice framework that is built on top of the spring framework. This can help developers to focus more on convention rather than configuration.

1. The main aim of Spring boot is to give you a production-ready application. So, the moment you create a spring-boot project, it is runnable and can be executed/deployed on the server.
2. It comes with features like autoconfiguration, auto dependency resolution, embedded servers, security, health checks which enhances the productivity of a developer.

2. How to create Spring Boot project in eclipse?

One of the ways to create a spring boot project in eclipse is by using **Spring Initializer.**

You can go to the official website of spring and add details such as version, select maven or Gradle project, add your groupId, artifactId, select your required dependencies and then click on CREATE PROJECT.

Once the project is created, you can download it and extract and import it in your eclipse or STS.

And see your project is ready! To Install Spring Boot in Eclipse – Go to Eclipse IDE, click on “Help”->then go to Eclipse marketplace->and type Spring IDE and click on the finish button.

3. How to deploy spring boot application in tomcat?

Whenever you will create your spring boot application and run it, Spring boot will automatically detect the embedded tomcat server and deploy your application on tomcat. After successful execution of your application, you will be able to launch your rest endpoints and get a response.

4. What is the difference between Spring and Spring Boot?

Difference between Spring and Spring boot are as follows:

Spring –

1. Is a dependency injection framework.
2. It is basically used to manage the life cycle of java classes (beans). It consists of a lot of boilerplate configuration.
3. Uses XML based configuration.
4. It takes time to have a spring application up and running and it’s mainly because of boilerplate code.

Spring boot-

1. It is a suite of pre- configured frameworks and technologies which helps to remove boilerplate configuration.
2. Uses annotations.
3. It is used to create a production-ready code.

5. What is actuator in spring boot?

An actuator is one of the best parts of spring boot which consists of production-ready features to help you monitor and manage your application.

With the help of an actuator, you can monitor what is happening inside the running application.
Actuator dependency figures out the metrics and makes them available as a new endpoint in your application and retrieves all required information from the web. You can identify beans, the health status of your application, CPU usage, and many more with the actuator. By using @Endpoint annotation, you can create a custom endpoint.

6. How to change port in spring boot?

The default port number to start your SpringBoot application is **8080**.

Just to change the port number, you need to add ***server.port=8084***c(your port number) property in your application.properties file and start your application.

7. How to create war file in spring boot?

To create a war file in spring boot you need to define your packaging file ***as war*** in your pom.xml(if it is maven project).

Then just do ***maven clean and install*** so that your application will start building. Once the build is successful, just go into your Target folder and you can see .war file generated for your application.

8. What is JPA in spring boot?

JPA is basically ***Java Persistence API***. It's a specification that lets you do ORM when you are connecting to a relational database which is Object-Relational Mapping.

So, when you need to connect from your java application to relational database, you need to be able to use something like JDBC and run SQL queries and then you get the results and convert them into Object instances.

ORM lets you map your entity classes in your SQL tables so that when you connect to the database , you don't need to do query yourself, it's the framework that handles it for you.

And JPA is a way to use ORM, it's an API which lets you configure your entity classes and give it to a framework so that the framework does the rest.

9. How to save image in database using spring boot?

1. First configure mysql in your spring boot application.
2. Then you can map your entities with your db tables using JPA.
3. And with the help of save() method in JPA you can directly insert your data into your database

```
@RestController
@RequestMapping("/greatleasrning")
public class Controller {
@Autowired
private final GreatLearningRepository greatLearningRepository;
public Controller(GreatLearningRepository greatLearningRepository) {
}
```

In above case, your data which may be in JSON format can be inserted successfully into database.

```
@RequestMapping(method = RequestMethod.POST)
ResponseEntity<?> insert(@RequestBody Course course) {
greatLearningRepository.save(course);
 return ResponseEntity.accepted().build();
}
}
```

10. What is auto configuration in spring boot?

AutoConfiguration is a process by which Spring Boot automatically configures all the infrastructural beans. It declares the built-in beans/objects of the spring specific module such as JPA, spring security and so on based on the dependencies present in your applications class path.

For example: If we make use of Spring JDBC, the spring boot autoconfiguration feature automatically registers the DataSource and JDBCTemplete bean.
This entire process of automatically declaring the framework specific bean without the need of writing the xml code or java config code explicity is called Autoconfiguration which is done by springBoot with the help of an annotation called ***@EnableAutoconfiguration*** alternatively ***@SpringBootApplication***.

11. How to resolve whitelabel error page in spring boot application?

This is quite common error in spring boot application which says 404(page not found).

We can mostly resolve this in 3 ways:

1. *Custom Error Controller*– where you will be implementing ErrorController interface which is provided by SpringFramework and then overriding its getErrorPath() so that you can return a custom path whenever such type of error is occurred.
2. ***By Displaying Custom error page***– All you have to do is create an error.html page and place it into the src/main/resources/templates path. The BasicErrorController of of springboot will automatically pick this file by default.
3. ***By disabling the whitelabel error page***– this is the easiest way where all you need to do is *server.error.whitelabel.enabled* property to false in the *application.properties* file to disable the whitelabel error page.

12. How to fetch data from database in spring boot?

You can use the following steps to connect your application with MySQL database.
1. First create a database in MySQL with create DATABASE student;
2. Now, create a table inside this DB:
CREATE TABLE student(studentid INT PRIMARY KEY NOT NULL AUTO_INCREMENT, studentname VARCHAR(255));
3. Create a SpringBoot application and add JDBC, MySQL and web dependencies.
4. In application.properties, you need to configure the database.

```
spring.datasource.url=jdbc:mysql://localhost:3306/studentDetails
spring.datasource.username=system123
spring.datasource.password=system123
spring.jpa.hibernate.ddl-auto=create-drop
```

5. In your controller class, you need to handle the requests.

```
package com.student;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class JdbcController {
@Autowired
JdbcTemplate jdbc;
@RequestMapping("/save")
public String index(){
jdbc.execute("insert into student (name)values(GreatLearnings)");
return "Data Entry Successful";
}
}
```

6. Run the application and check the entry in your Database.

13. How to use logger in spring boot?

There are many logging options available in SpringBoot. Some of them are mentioned below:

Using log4j2:

```
import org.apache.logging.log4j.Logger;
import org.apache.logging.log4j.LogManager;
// [...]
Logger logger = LogManager.getLogger(LoggingController.class);
```

Using Lombok:

All you need to do is add a dependency called ***org.projectlombok*** in your pom.xml as shown below:

```
<dependency>
 <groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<version>1.18.4</version>
<scope>provided</scope>
</dependency>
```

Now you can create a loggingController and add the ***@Slf4j*** annotation to it. Here we would not create any logger instances.

```
@RestController
@Slf4j
public class LoggingController {

@RequestMapping("/logging")
public String index() {
log.trace("A TRACE Message");
log.debug("A DEBUG Message");
log.info("An INFO Message");
log.warn("A WARN Message");
log.error("An ERROR Message");

return "Here are your logs!”;
}
}
```

So, there are many such ways in spring boot to use logger.

14. What is bootstrapping in spring boot?

One of the way to bootstrap your spring boot application is using Spring Initializer.
you can go to the official website of spring and select your version, and add you groupID, artifactId and all the required dependencies.

And then you can create your restEndpoints and build and run your project.
There you go, you have bootstrapped your spring boot application.

15. How to create jar file in spring boot?

To create a jar file in spring boot you need to define your packaging file as ***jar*** in your pom.xml(if it is maven project).

Then just do maven build with specifying ***goals as package*** so that your application will start building.

Once the build is successful, just go into your Target folder and you can see .jar file generated for you application.

16. What is dependency injection in spring boot?

Dependency injection is a way through which the Spring container injects one object into another. This helps for loose coupling of components.

For example: if class student uses functionality of department class, then we say student class has dependency of Department class. Now we need to create object of class Department in your student class so that it can directly use functionalities of department class is called dependency injection.

17. How to store image in MongoDB using spring boot?

One of the way for storing image in MongoDB is by using Spring Content. And also you should have the below dependency in your pom.xml.

```
<dependency>
<groupId>com.github.paulcwarren</groupId>
<artifactId>spring-content-mongo-boot-starter</artifactId>
<version>0.0.10</version>
</dependency>
```

You should have a GridFsTemplate bean in your applicationContext.

```
@Configuration
public class Config

@Bean
public GridFsTemplate gridFsTemplate() throws Exception {
return new GridFsTemplate(mongoDbFactory(), mappingMongoConverter());
}
...
```

Now add attributes so that your content will be associated to your entity.

```
@ContentId
private String contentId;

@ContentLength
private long contentLength = 0L;

@MimeType
private String mimeType = "text/plain";
And last but not the least, add a store interface.
@StoreRestResource(path="greatlearningImages")
public interface GreatLearningImageStore extends ContentStore<Candidate, String> {
}
```

That’s all you have to do to store your images in mongoDb using Springboot.

18. How to configure hibernate in spring boot?

The important and required dependency to configure hibernate is:

1. **spring-boot-starter-data-jpa**
2. **h2** (you can also use any other database)

Now, provide all the database connection properties in application.properties file of your application in order to connect your JPA code with the database.

Here we will configure H2 database in application.properties file:

```
spring.datasource.url=jdbc:h2:file:~/test
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=test
spring.datasource.password=test
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
```

Adding the above properties in your application.properties file will help you to interact with your database using JPA repository interface.

19. Mention the advantages of Spring Boot.

Advantages of Spring Boot –

1. It allows convention over configuration hence you can fully avoid XML configuration.
2. SpringBoot reduces lots of development time and helps to increase productivity.
3. Helps to reduce a lot of boilerplate code in your application.
4. It comes with embedded HTTP servers like tomcat, Jetty, etc to develop and test your applications.
5. It also provides CLI (Command Line Interface) tool which helps you to develop and test your application from CMD.

20. Explain what is thyme leaf and how to use thymeleaf?

Thymeleaf is a server-side java template engine which helps processing and creating HTML, XML, JavaScript , CSS, and text. Whenever the dependency in pom.xml (in case of maven project) is find, springboot automatically configures Thymeleaf to serve dynamic web content.

Dependency: spring-boot-starter-thymeleaf

We can place the thyme leaf templates which are just the HTML files in ***src/main/resources/templates/*** folder so that spring boot can pick those files and renders whenever required.

Thymeleaf will parse the index.html and will replace the dynamic values with its actual value that is been passed from the controller class.
That's it, once you run your Spring Boot application and your message will be rendered in web browsers.

21. What is the need for Spring Boot DevTools?

This is one of the amazing features provided by Spring Boot, where it restarts the spring boot application whenever any changes are being made in the code.

Here, you don't need to right-click on the project and run your application again and again. Spring Boot dev tools does this for you with every code change.
Dependency to be added is: spring-boot-devtools

The main focus of this module is to improve the development time while working on Spring Boot applications.

22. Can we change the port of the embedded Tomcat server in Spring boot?

Yes, you can change the port of embedded Tomcat server in Spring boot by adding the following property in your ***application.properties*** file.

server.port=8084

The default port number of the tomcat server to run the spring boot application is 8080, which is further possible to change it.

So we can change the port of tomcat following ways given below:-

- Using application.properties
- Using application.yml
- Using EmbeddedServletContainerCustomizer interface.
- Using WebServerFactoryCustomizer interface.
- Using Command-Line Parameter.

23. Mention the steps to connect Spring Boot application to a database using JDBC

Below are the steps to connect your Spring Boot application to a database using JDBC:

Before that, you need to add required dependencies that are provided by spring-boot to connect your application with JDBC.

Step 1: First create a database in MySQL with create DATABASE student;

Step 2: Now, create a table inside this DB:
CREATE TABLE student(studentid INT PRIMARY KEY NOT NULL AUTO_INCREMENT,

studentname VARCHAR(255));

Step 3: Create a springBoot and add JDBC,mysql and web dependencies.
Step 4: In application.properties, you need to configure the database.

```
spring.datasource.url=jdbc:mysql://localhost:3306/studentDetails
spring.datasource.username=system123
spring.datasource.password=system123
spring.jpa.hibernate.ddl-auto=create-drop
```

Step 5: In your controller class, you need to handle the requests.

```
package com.student;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.web.bind.annotation.RestController;
@RestController
  public class JdbcController {
@Autowired
JdbcTemplate jdbc;
   @RequestMapping("/save")
public String index(){
jdbc.execute("insert into student
(name)values(GreatLearnings)");
        return "Data Entry Successful";
}
}
```

Step 6: Run the application and check the entry in your Database.

Step 7: You can also go ahead and open the URL and you will see “Data Entry Successful” as your output.

24. What are the @RequestMapping and @RestController annotation in Spring Boot used for?

The ***@RequestMapping*** annotation can be used at class-level or method level in your controller class.

The global request path that needs to be mapped on a controller class can be done by using ***@RequestMapping*** at class-level. If you need to map a particular request specifically to some method level.

Below is a simple example to refer to:

```
@RestController
@RequestMapping("/greatLearning")
public class GreatLearningController {
@RequestMapping("/")
String greatLearning(){
return "Hello from greatLearning ";
}
@RequestMapping("/welcome")
String welcome(){
return "Welcome from GreatLearning";
}
}
```

The ***@RestController*** annotation is used at the class level.

You can use @RestController when you need to use that class as a request handler class.All the requests can be mapped and handled in this class.

@RestController itself consists ***@Controller*** and ***@ResponseBody*** which helps us to remove the need of annotating every method with @ResponseBody annotation.

Below is a simple example to refer to for use of @RestController annotation:

```
@RestController
@RequestMapping(“bank-details”)
public class DemoRestController{
@GetMapping(“/{id}”,produces =”application/json”)
public Bank getBankDetails(@PathVariable int id){
return findBankDetailsById();
}
}
```

Here, @ResponseBody is not required as the class is annotated with @RestController.

25. What do you understand by auto-configuration in Spring Boot and how to disable the auto-configuration?

AutoConfiguration is a process by which Spring Boot automatically configures all the infrastructural beans. It declares the built-in beans/objects of the spring-specific module such as JPA, spring-security, and so on based on the dependencies present in your application’s classpath.
For example: If we make use of Spring JDBC, the spring boot autoconfiguration feature automatically registers the DataSource and JDBCTemplete bean.
This entire process of automatically declaring the framework-specific bean without the need of writing the XML code or java-config code explicitly is called Autoconfiguration which is done by spring-boot with the help of an annotation called ***@EnableAutoconfiguration*** alternatively ***@SpringBootApplication.***

1. You can exclude the attribute of @EnableAutoConfiguration where you don’t want it to be configured implicity in order to disable the spring boot’s auto-configuration feature.

2. Another way of disabling auto-configuration is by using the property file:

For example:

```
spring.autoconfigure.exclude=
org.springframework.boot.autoconfigure.mongo.MongoAutoConfiguration,
org.springframework.boot.autoconfigure.data.MongoDataConfiguration,
```

In the above example, we have disabled the autoconfiguration of MongoDB.

26. Can you give an example for ReadOnly as true in Transaction management?

Yes, example for ReadOnly as true in Transaction Management is:

Suppose you have a scenario where you have to read data from your database like if you have a STUDENT database and you have to read the student details such as studentID, and studentName.

So in such scenarios, you will have to set read-only on the transaction.

27. Mention the advantages of the YAML file than Properties file and the different ways to load

YAML file in Spring boot.

YAML gives you more clarity and is very friendly to humans. It also supports **maps, lists, and other scalar types.**

YAML comes with hierarchical nature which helps in avoiding repetition as well as indentations.

If we have different deployment profiles such as development, testing, or production and we may have different configurations for each environment, so instead of creating new files for each environment we can place them in a single YAML file.
But in the case of the properties file, you cannot do that.

For example:

```
spring:
profiles:
active:
-test
---
spring:
profiles:
active:
-prod
---
spring:
profiles:
active:
-development
```

28. What do you understand by Spring Data REST?

By using Spring Data Rest, you have access to all the RESTful resources that revolves around Spring Data repositories.

Refer the below example:

```
@RepositoryRestResource(collectionResourceRel = "greatlearning", path = "sample")
public interface GreatLearningRepo extends CustomerRepository< greatlearning,
Long> {
}
```

Now you can use the POST method in the below manner:

```
{
“Name”:”GreatLearning”
}
```

And you will get response as follow:

```
{
“Name”:”GreatLearning”
}
__________
{
"name": "Hello greatlearning "
"_links": {
"self": {
"href": "<a href="http://localhost:8080/sample/1">http://localhost:8080/
greatlearning /1</a>"
},
" greatlearning ": {
“href": "<a href="http://localhost:8080/sample/1">http://localhost:8080/
greatlearning /1</a>"
}
}
```

In the above, you can see the response of the newly created resource.

29. What do you think is the need for Profiles?

The application has different stages-such as the *development stage, testing stage, production stage* and may have different configurations based on the environments.

With the help of spring boot, you can place profile-specific properties in different files such as

application-{profile}.properties

In the above*, you can replace the profile with whatever environment you need, for example, if it is a development profile, then* ***application-development.properties*** *file will have development specific configurations in it.*

So, in order to have profile-specific configurations/properties, you need to specify an active profile.

30. How to insert data in mysql using spring boot?

First configure mysql in your spring boot application.

Then you can map your entities with your db tables using JPA.

And with the help of save() method in JPA, you can directly insert your data into your database.

```
@RestController
@RequestMapping("/greatleasrning")
public class Controller {
@Autowired
private final GreatLearningRepository greatLearningRepository;
public Controller(GreatLearningRepository greatLearningRepository) {
this. greatLearningRepository = greatLearningRepository;
}
```

In the above case, your data which may be in JSON format can be inserted successfully into the database.

```
@RequestMapping(method = RequestMethod.POST)
ResponseEntity<?> insert(@RequestBody Course course) {
greatLearningRepository.save(course);
return ResponseEntity.accepted().build();
}
}
```

31. How to create a login page in spring boot?

You can create a simple and default login page in spring boot, you can make use of Spring security. Spring security secures all HTTP endpoints where the user has to login into the default HTTP form provided by spring.

We need to add ***spring-boot-starter-security*** dependency in your pom.xml or build.gradle and a default username and password can be generated with which you can log in.

32. What is the main class in spring boot?

Usually in java applications, a class that has a main method in it is considered as a main class. Similarly, in spring boot applications main class is the class which has a public static void main() method and which starts up the SpringApplicationContext.

33. How to use crud repository in spring boot?

In order to use crud repository in spring boot, all you have to do is extend the crud repository which in turn extends the Repository interface as a result you will not need to implement your own methods.

Create a simple spring boot application which includes below dependency: **spring-boot-starter-data-jpa**, **spring-boot-starter-data-rest**

And extend your repository interface as shown below**:**

```
package com.greatlearning;
import java.util.List;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;
@RepositoryRestResource
public interface GreatLearning extends CrudRepository<Candidate, Long>
{
public List<Candidate> findById(long id);

//@Query("select s from Candidate s where s.age <= ?")
public List<Candidate> findByAgeLessThanEqual (long age);
}
```

34. How to run spring-boot jar from the command line?

In order to run spring boot jar from the command line, you need to update you pom.xml(or build.gradle) of your project with the maven plugin.

```
<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>
```

Now, Build your application and package it into the single executable jar. Once the jar is built you can run it through the command prompt using the below query:

java -jar target/myDemoService-0.0.1-SNAPSHOT.jar

And you have your application running.

35. What is Spring Boot CLI and how to execute the Spring Boot project using boot CLI?

Spring Boot CLI is nothing but a command-line tool which is provided by Spring so that you can develop your applications quicker and faster.

To execute your spring boot project using CLI, you need first to download CLI from Spring's official website and extract those files. You may see a bin folder present in the Spring setup which is used to execute your spring boot application.

As Spring boot CLI allows you to execute groovy files, you can create one and open it in the terminal.
And then execute ***./spring run filename.groovy;***

36. what is the rest controller in spring boot?

The ***@RestController*** annotation is used at the class level.

You can use @RestController when you need to use that class as a request handler class.All the requests can be mapped and handled in this class.

@RestController itself consists ***@Controller*** and ***@ResponseBody*** which helps us to remove the need of annotating every method with @ResponseBody annotation.

Below is a simple example to refer to for use of @RestController annotation:

```
@RestController
@RequestMapping(“bank-details”)
public class DemoRestController{
@GetMapping(“/{id}”,produces =”application/json”)
public Bank getBankDetails(@PathVariable int id){
return findBankDetailsById();
}
}
```

Here, @ResponseBody is not required as the class is annotated with @RestController.

37. How to handle 404 error in spring boot?

Consider a scenario, where there are no stockDetails in the DB and still, whenever you hit the GET method you get 200(OK) even though the resource is not found which is not expected. Instead of 200, you should get 404 error.
So to handle this, you need to create an exception, in the above scenario “StockNotFoundException”.

```
GetMapping("/stocks/{number}")
public Stock retriveStock(@PathVariable int number)
{
Stock  stock  = service.findOne(number);
if(Stock  ==null)
//runtime exception
throw new StockNotFoundException("number: "+ number);
return stock;
}
```

Now, create a Constructor from Superclass.

Right-click on the file -> Go to Source ->And generate constuctors from superclass-> and check the RuntimeException(String)-> and generate.

And add an annotation called ***@ResponseStatus*** which will give you 404 (not found) error.

```
package com.greatlearning;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(HttpStatus.NOT_FOUND)
public class StockNotFoundException extends RuntimeException
{
public StockNotFoundException(String message)
{
super(message);
}
}
```

Now, you can hit the same URL again and there you go, you get a 404 error when a resource is not found.

38. Which is the spring boot latest version?

The latest version of spring boot is **2.6.0**. It came out with a lot of dependency upgrades, java 15 support and much more.

Yes, now as you are brushed up with spring boot interview questions and answers. We have also tried to cover all the springboot interview questions for experienced professionals. Hope you can easily crack the spring boot interview now!

Please feel free to comment below if you have any queries related to the above questions or answers. Also, do comment if you find any other questions that you think must be included in the above list of questions.

Spring Boot Interview Questions for Experienced

As an experienced professional, you should be prepared to answer questions about your experience with Spring Boot. In this section, we will share some of the most common Spring Boot interview questions for experienced professionals.

39. How to check the environment properties in your Spring boot application?

If we need to set the different target environments, Spring Boot has a built-in mechanism.

One can simply define an application environment.properties file in the src/main/resources directory and then set a Spring profile with the same environment name.

For example, if we define a “production” environment, that means we’ll have to define a production profile and then application-production.properties.

This environment file will be loaded and will take precedence over the default property file. You should note that the default file will still be loaded. It’s just that when there is a property collision, the environment-specific property file takes precedence.

40. Where do we define properties in the Spring Boot application?

Command Line Properties

Command-line properties are converted into Spring Boot Environment properties by the spring boot application.

Command-line properties have more precedence over the other property sources.

Spring Boot uses the 8080 port number, by default, to start the Tomcat. Let us see how one can change the port number by using command-line properties.

```
c:demotarget>java -jar demo-0.0.1-SNAPSHOT.jar --server.port=9090
```

Properties File

Properties files are used to keep one or more properties in a single file to run the application in a different environment. Properties are kept in the application.properties file under the classpath in a typical spring boot application. The location of the application.properties file is at src/main/resources directory. The code of application.properties file is as below:

```
sever.port=9090
spring.application.name = demoservice
```

YAML File

Spring Boot also supports YAML-based properties configurations to run the application. The user can use, application.yml file instead of the application.properties file. The YAML file is kept inside the classpath. The sample application.yml file is given below −

```
spring:
    application:
       name: demoservice
  server:
port: 9090
```

Externalized Properties

The user can keep properties in different locations or paths instead of keeping the properties file under classpath. While running the JAR file, the user can specify the properties file path. The application developer can use the following command to specify the location of the properties file while running the JAR −

```
-Dspring.config.location = C:application.properties

-C:demotarget>java -jar -Dspring.config.location=C:application.properties demo-
0.0.1-SNAPSHOT.jar
```

41. What is an IOC container?

IoC Container is a framework that is used for implementing automatic dependency injection. It manages object creation and its lifetime. It, it also injects dependencies into the class.

The IoC container is used to create an object of the specified class. It also injects all the dependency objects through a constructor, a property, or a method at run time and disposes it at the appropriate time. With this, one doesn't have to create and manage objects manually.

All the containers provide easy support for the Dependency Injection lifecycle as below.

Register: The container should know which dependency to instantiate when it encounters a particular type. This process is called registration.

Resolve: When using the IoC container, the objects need to be created manually. This is done by the container and is called resolution. The container should include some methods to resolve the specified type; the container creates an object of the specified type. It then injects the required dependencies if any and returns the object.

Dispose: The container should manage the lifetime of the dependent objects. IoC containers include different lifetime managers to manage an object's lifecycle and dispose it.

42. What are the basic Annotations that spring boot offers?

First of all, we have to know about the annotations. Annotations are used to instruct the intention of the programmers.

As the name suggests, spring boot annotations is a form of Metadata that provides the whole data about the program. In other ways, we can define it as annotations are used to provide supplemental information about the program. It is not part of the program.

It does not change the programs which are already compiled.

Core Spring Framework Annotation:-

1. **@Required:-**

@Required applies to the bean setter method.

This indicates that the annotated bean must be populated at the configuration time with the required property; if the following case is not satisfied, it throws an exception BeanInitializationException.

2. **@Autowired:-**

In the spring framework, spring provides annotation-based auto–wiring by providing @Autowired annotation.

It is used to auto-wire spring bean on setter methods, instance variables and constructors., When we use the annotation @Autowired, the spring container auto-wires the bean factory by matching the data type.

Other Annotations which are provided by Spring Boot, Spring Framework, and In Spring MVC are:-

3. @configuartion.
4. @Componentscan
5. @Bean
6. @component.
7. @Controller.
8. @service.
9. @Repository
10. @EnableAutoConfiguaration
11. @SpringBootApplication.
12. @RequestMapping
13. @GetMapping
14. @PostMapping.

43. What is spring Boot dependency Management?

Spring Boot manages dependencies and configuration automatically. Each release of spring boot provides a list of dependencies that it supports. The list of dependencies available as a part of Spring-boot dependencies can be used in maven, so we need to specify the version of the dependencies or add the dependencies version in our config file in our configuration.

Spring boot automatically manages and spring boot upgrades all dependencies automatically respectively or consistently at the time when we update the spring boot version.

Advantage of Dependency Management:-

1. Spring dependency management provides us the centralized dependency information by adding or specifying the dependencies version in a required place in the spring boot version. It also helps us to switch from one version to another version with ease.
2. This management helps us to avoid the mismatch of different versions of the Spring Boot library.
3. Here we simply have to write a library name specifying the version.

44. Can we create a non-web application in spring boot?

Yes, but the application could also be called as spring boot standalone application.

To create a non-web application, your application needs to implement CommandLineRunner interface and its Run method for the running of our application. So this run method always acts like the main of our non-web application.

45. What is the default port of the tomcat server in Spring Boot?

As we had already discussed about the default port, the tomcat server in spring boot is port 8080. Which is changeable based on the user or the programmer’s requirement.

46. Can we override or replace the embedded tomcat server in spring boot?

If we consider the fact, spring boot by default comes up with the embedded server once we add the “Spring –boot-starter” dependency. But the spring boot gives us the flexibility to use the tomcat.

If we don’t want to use the tomcat, then tomcat comes with three types of embed servers: Tomcat, jetty, and undertow.

47. Can we disable the default web server in the spring boot application?

Yes, as discussed above, there are 3 web servers available we can choose between them. Spring boot gives more priority for using the tomcat server.

48. Explain @Restcontroller annotation in spring boot?

Spring restcontroller annotation is an annotation that is itself annotated within two annotations.

@Restcontroller is annotated within @controller and @Responsebody. This annotation is applied to mark the respective class as a request handler in your application.

Spring Rest controller annotation is used to create restful web services using Spring MVC.

49. What is the difference between @RestController and @Controller in Spring Boot?

50. Describe the flow of HTTPS request through the spring boot app?

We all can see the above image of the spring boot flow architecture to understand the basic concept of the HTTPS request flow in the spring boot app.

We have the validator classes, view classes, and utility classes.

As we all know, spring boot uses the modules of spring-like MVC, spring data, etc.

So the concept also the same for several things, and also the architecture of spring boot is the same as the architecture of spring MVC; instead of one concept, there is no need for the DAO and DAOimpl classes in spring boot.

It creates a data access layer and started performing CRUD operations.

CRUD operation is nothing but Create Read Update and Delete operation, which is done by all of the programmers in their website.

The client makes the HTTP request in PUT or GET.

After this, the request goes to the controller, and the controller maps that respective request and handles it; if there is the requirement for calling some logic, it calls the service logic after handling the request.

All the business logic performs in the service layer.

Service layer 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 has occurred.

51. What is the difference between RequestMapping and GetMapping?

The @GetMapping is a composed annotation which is the short notation of @RequestMapping(method=RequestMethod.GET).

These both methods support the “Consumes.”

The consumes options are,

Consumes=”text/plain”

Consumes={“text/plain”,”application”};

52. How to get the list of all the beans in your spring boot application?

In the case of spring boot, you can use appContext.getBeanDefinitionNames() to get all the beans loaded by the spring container.

By calling this method, we can show all of our beans present in our spring boot applications.

Spring Boot Microservices Interview Questions

If you’re looking for Spring Boot interview questions regarding microservices, you’ve come to the right place. In this article, we’ll share with you some of the most popular and insightful questions that will help you prepare for your next interview.

53. What are Microservices?

Microservices is a style of architecture wherein the key business capabilities of an application are exposed as loosely coupled services that can be independently deployed. Each service exposed is referred to as Microservice. For example, let us take the example of an eCommerce application. We can design and build separate microservices for key business functionalities of the eCommerce application like Authentication, Customer Account, Product Catalog, Product Ordering, Product Offering Qualification, Shopping Cart, Recommendation, Payment, Payment Method, Shipment Tracking, etc.

54. What are microservices in spring boot?

Microservices is an architectural style wherein the key business capabilities of an application are exposed as loosely coupled services.

Sprint boot is a framework that has evolved to be formidable for Java microservice development.

Spring Boot enables building production-ready applications faster and provides embedded servers that are easy to deploy with containers.

Spring Cloud which builds on top of Spring Boot, provides features to quickly build production-ready microservices. It’s possible to quickly set up services with minimal configurations Eg. Service Registration and discovery, circuit breakers, proxies, logging, log tracking, monitoring, etc.

55. What are Microservices in Java?

Microservices in Java are nothing but microservices with microservices architecture using Java programming language. The speciality of Microservices is that polyglot architecture is supported.

For example, if a team is working on one of the microservice using Java, Spring Boot, and MySQL, another team can work on another microservice using Python, Node JS, and NoSQL.

56. What is Microservices Architecture?

Microservice architecture is an architectural pattern of software development wherein an application’s core business capabilities are exposed as loosely coupled services that can be developed, deployed, and maintained independently of each other.

- Each service performs a unique function.
- Services are distributed across systems.
- Services are organized around business capabilities.
- Data management is decentralized
- Governance is decentralized

Polyglot architecture where different microservices can use a different version of the same programming language and/or different programming language and/or different architectures as well.

57. Why Microservices?

In the case of monolith applications, there are several problems like

a. Same code base for presentation, business layer, and data access layer. Application is deployed as a single unit.

b. Complex to maintain and scalability is an issue.

Microservice solves the above problems.

Microservices are ideal when a monolith or a legacy application needs to be modernized.

For new software development, if the key business drivers are to reduce time to market, scalable better software, lower costs, faster development, or cloud-native development, microservices are ideal.

Each service is independent and gives the flexibility to choose the programming language, database, and/or architecture.

Distinct services can be developed, deployed, and maintained independently.

58. What is an API gateway in microservices?

API Gateway in Microservices is a Microservices Architecture pattern.

API Gateway is a server and is a single-entry point into the system. API Gateway is responsible for routing the request, composition, and translation of the protocol. All the requests from the clients first come to the API Gateway and the API Gateway routes the request to the correct microservice.

API Gateway can also aggregate the results from the microservices back to the client. API Gateway can also translate between web protocols like HTTP, web socket, etc.

API Gateway can provide every client with a custom API as well.

An example of an API Gateway is Netflix API Gateway.

59. How to deploy microservices?

Microservices are developed and deployed quickly and in most cases automatically as part of the CI/CD pipeline. Microservices could be deployed in Virtual Machines or Containers. The virtual machines or containers can be On-premise or in the cloud as well.

There are different deployment approaches available for Microservices. Some of the possible deployment approaches for microservices are mentioned below.

- Multiple service instances per host
- Service instance per host
- Service instance per VM
- Service instance per Container
- Serverless deployment
- Service deployment platform

60. How to handle exceptions in microservices?

In the case of microservices, exception handling is important. If any exception/error is not handled, it will be propagated to all the downstream services creating an impact on the user experience. To make the services more resilient, handling exceptions becomes very important.

In the case of ‘500 – Internal Service Error’, Sprint Boot will respond like below.

```
(
"timestamp": "2020-04-02T01:31:08.501+00:00",
"path": "/shop/action",
"status": 500,
"error": "Internal Server Error",
"message": "",
"requestId": "a8c4c6d4-3"
}
```

Spring provides ControllerAdvice for exception handling in Spring Boot Microservices. @ControllerAdivce informs Spring Boot that a class will act like an Interceptor in case of any exceptions.

We can have any number of exception handlers to handle each exception.

Eg. For handling generic Exception and RunTimeException, we can have 2 exception handlers.

@ControllerAdvice public class ApplicationExceptionHandler { @ExceptionHandler(Exception.class) public ResponseEntity handleGenericException(Exception e){ ShopException shopException = new ShopException(100, “Items are not found”); return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR) .body(shopException); } @ExceptionHandler(RuntimeException.class) public ResponseEntity handleRunTimeException(RuntimeException e){ ShopException shopException = new ShopException(101, “Item is not found”); return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR) .body(shopException); } }

61. What is Spring Cloud?

Spring Cloud is an open-source library that provides tools for quickly deploying the JVM based application on the clouds. It provides a better user experience and an extensible mechanism due to various features like Distributed configuration, Circuit breakers, Global locks, Service registrations, Load balancing, Cluster state, Routing, Load Balancing, etc. It is capable of working with spring and different applications in various languages

61. Features of Spring Cloud

Major features are as below:

- Distributed configuration
- Distributed messaging
- service-to-service calls
- Circuit breakers
- Global locks
- Service registration
- Service Discovery
- Load balancing
- Cluster state
- Routing

62. How Do You Override A Spring Boot Project's Default Properties?

Spring Application loads properties from the application.properties files in the following locations and add them to the Spring Environment:

1. A /config subdirectory of the current directory.
2. The current directory
3. A classpath /config package
4. The classpath root

The list is ordered by precedence means that the properties that are defined in locations higher in the list override those defined in lower locations.

If the user does not want application.properties as the configuration file name, they can switch to another by specifying a spring.config.name environment property. The user can also refer to an explicit location using the spring.config.location environment property (comma-separated list of directory locations, or file paths).

```
$ java -jar myproject.jar --spring.config.name=myproject
```

or

```
$ java -jar myproject.jar --
spring.config.location=classpath:/default.properties,classpath:/override.propertie
s
```

63. How Is Spring Security Implemented In A Spring Boot Application?

Spring Security is a framework that majorly focuses on providing both authentication and authorization to Java EE-based enterprise software applications.

Adding Spring security:

Maven:

To include spring security, include the below dependency:

```
<dependencies>
<dependency>
<groupID>org.springframework.security</groupID>
<artifactId>spring-security-config</artifactID>
<version>5.5.0</version>
</dependeny>
<dependency>
<groupId>org.springframework.security</groupId>
<artifactId>spring-security-web</artifactId>
<version>5.5.0</version>
</dependency>
</dependencies>
```

Gradle:

To include spring security in Gradle based project use:

```
repositories {
mavenCentral()
}
dependencies {
compile 'org.springframework.security:spring-security-web:5.5.0'
compile 'org.springframework.security:spring-security-config:5.5.0'
}
```

64. Which Embedded Containers Are Supported By Spring Boot?

The embedded containers supported by spring boot are Tomcat (default), Jetty, and undertow servers

65. Where Do We Use WebMVC Test Annotation?

```
@Target(value=TYPE)
@Retention(value=RUNTIME)
@Documented
@Inherited
@BootstrapWith(value=org.springframe.boot.test.autoconfigure.web.servlet.WebMvcTes
tContextBootsrapper.class)
@ExtendWidth(value=org.springframework.test.contect.junit.jupiter.SpringExtension.
class)
@AutoConfigureCache
@AutoConfigureWebMvc
@AutoConfigureMockMvc
@ImportAutoConfiguration
public @interface WebMvcTest
```

Annotation can be used for a Spring MVC test that focuses only on Spring MVC components.

Using this annotation disables full auto-configuration and instead apply only configuration relevant to MVC tests (i.e., @Controller, @ControllerAdvice, @JsonComponent, Converter/GenericConverter, Filter, WebMvcConfigurer, and HandlerMethodArgumentResolver beans but not @Component, @Service, or @Repository beans).

By default, annotated tests with @WebMvcTest will also auto-configure Spring Security and MockMvc (including support for HtmlUnit WebClient and Selenium WebDriver). For more fine-grained control of MockMVC, the @AutoConfigureMockMvc annotation is used.

Usually @WebMvcTest is used in combination with @MockBean or @Import to create any collaborators required by your @Controller beans.

66. How to Configure Spring Boot Application Logging?

Spring Boot provides a LoggingSystem abstraction that configures logging based on the content of the classpath. If Logback is available, it is definitely the first choice.

Suppose the only change the user needs to make to logging is to set the levels of various loggers. In that case, they can do so in application.properties by using the "logging.level" prefix, as shown in the following example:

```
logging.level.org.springframework.web=DEBUG
logging.level.org.hibernate=ERROR
```

Java Spring boot interview questions

This section provides an overview of some of the most common Java Spring boot interview questions.

67. What is the Minimum Java version needed for Spring Boot?

Java 8 is the minimum version required.

68. How to use thymeleaf?

Steps are as follows:

1. First, create a Spring Boot Project using STS or Spring Initializer. Add dependency for Thymeleaf and Spring Web.

For Gradle:

```
implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
implementation 'org.springframework.boot:spring-boot-starter-web'
```

For Maven:

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

2. Create a Controller Class in package by either adding a new package or use the default package containing the main application class.

DemoController.java:

```
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class DemoController {

@GetMapping(value = "/thymeleafTemplate")
public String getTemplate(@RequestParam(name="name" , required=false,
defaultValue="World") String name, Model model) {
model.addAttribute("name",name);
return "thymeleafTemplate";
}
}
```

3. Add template in the resources folder.
   src/main/resources/templates/thymeleafTemplate.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title> Thymeleaf Spring Boot Demo </title>
</head>
<body>
<p th:text=" 'Hello, ' + ${name} + '!'"/>
<h4> Welcome to Thymeleaf Demo in Spring Boot</h4>
</body>
</html>
```

4. Build code.

Run the application using Integrated Development Environment: Run as -> Spring Boot App.

69. How to Use Spring Boot for Command-Line Applications?

To run Spring Boot for Command-Line Applications, Open the terminal window and change the directory to the root folder of your Spring Boot application.

If the user list files in this directory, they should see a pom.xml file. One can also run your Spring Boot application as an executable Java jar file

70. How Can You Change the Default Port in Spring Boot?

Default port is 8080; The user can change the default port by:

1. Command-line:

```
java -jar spring-5.jar --server.port=8083
```

2. By changing in application.properties file

```
server.port=8081
```

3. Programmatic Configuration:

```
@SpringBootApplication
public class CustomApplication {
public static void main(String[] args) {
SpringApplication app = new SpringApplication(CustomApplication.class);
app.setDefaultProperties(Collection.singletonMap("server.port", "8083"));
app.run(args);
}
```

71. Explain what happens in the background when a Spring Boot Application is "Run as Java Application"?

If you are using Eclipse IDE or an Eclipse maven plugin, make sure that as soon as you add a dependency or change the class file, it is compiled and available in the target folder. After that, it is just like any other Java application.

When you launch the java application, then the spring boot auto configuration kicks in.

It starts up tomcat when it sees that you are developing a web application!

72. What are the differences between JPA and Hibernate?

JPA is a standard, while Hibernate is not a standard.

The. session is used to handle data's persistence in hibernate, while in JPA, Entity Manager is used. The query language in Hibernate is Hibernate Query language, while in JPA, the query language is Java Persistence query language. Hibernate is one of the most JPA providers.

73. What are the Spring Boot key components?

The four key components of Spring Boot are –

- Spring Boot Starter: It is a key component of Spring Boot Framework and is primarily used to aggregate a group of dependencies into a single one. This helps in simplifying project dependencies.
- Spring Boot CLI: It is an acronym for Spring Boot Command Line Interface. It is essentially software that helps in testing Spring Boot Applications from Command Prompt.
- Spring Boot AutoConfigurator: Another major component of Spring Boot Framework, it is used for the development of Spring-based applications that often require a lot of configuration.
- Spring Boot Actuator: Spring Boot Actuator assists in managing endpoints to Spring Boot applications. It is also used to provide Spring Boot Application metrics.

74. What is the purpose of using @ComponentScan in the class files?

The @ComponentScan annotation is used to scan the components added to the project for the class file. It is also used to specify base packages and their classes using the basePackages attributes. As you specify the basePackages attribute, this will prompt Spring Boot to scan all the packages as well as subpackages of the classes that have been specified.

75. What is Spring Initializr?

Spring initializr is a web-based application that helps in generating spring boot projects for developers. It will provide the basic structure code without any application code. Using the Maven or a Gradle build specification, you can build the code. It is very helpful when you are starting a project from scratch as it eliminates the need for setting up a framework.

76. How do you Configure Log4j for logging?

Here is a simple example of the configuration of Log4j for logging –

- Start by creating a Java project. You can do so by navigating to MyEclipse and then File->New-> Java Project.
- Enter the project name and click Finish.
- Now, to add the log4j file, right-click on the created java project name. Then select Build Path -> Configure Build Path.
- Go to the library and click Add External JARs button.
- Now from the drop down menu, select the log4j file and click OK.
- Next, create a new Java file.
- Create a log4j,properties file. Now, create a new file on that folder with name log4j.properties and click Finish.
- Add the log4j.properties file to the Classpath.
- In Classpath, click on Advanced. Select the Add Folders option and click OK.
- Now browse the folder for log4j.properties file. Click OK and apply Run.
- Run the project.

- This will give the output on the console

```
[main] DEBUG Example  – Hello this is a debug message

[main] INFO  Example  –   Hello this is an info message
```

77. What are the HTTP methods that can be implemented in spring boot rest service?

Here are the essential HTTP methods that can be implemented in the spring boot rest service –

- GET: A key HTTP method, GET is used to know the representational view of the data. When used in read-only mode, it helps in keeping the data safe and secure. Also, the results are idempotent which means that you will get the same results no matter how many times it is used.
- POST: The POST restful API HTTP method works on resource collections. Using POST on the parent resource creates new resources associated with a proper hierarchy. It is highly useful for developers to prevent polluting the code and define resources in an explicit manner.
- PUT: The PUT restful API helps in updating a resource. It does so by replacing the content of that resource entirely. It is the most common way of updating information.
- PATCH: PATCH is also used to update resources. Unlike the PUT method, it only modifies resource content and does not replace it entirely. Unless you want to update every resource of your content, it is poor practice to apply PATCH.
- DELETE: This HTTP method is used to target and delete a single resource. However, its implementation can be inconsistent sometimes.

78. What are the steps to add a custom JS code with Spring Boot?

You can add a custom JS code with Spring Boot in these steps –

- Under the Resources folder, create a folder named Static.
- Put your static content in that folder.
- /src/main/resources/static is the suggested folder.
- Refer to the path to myStatic.js as – <script src="/js/myStatic.js"></script>

79. How to debug spring boot applications in eclipse?

- Start by opening an SSH port-forwarding tunnel to the server.
- Now, keep the SSH session running. Go to Eclipse> Toolbar> Run> Debug Configurations.
- Select Remote Java Application and click on NEW.
- Select the Socket Attach as Connection Type Standard.
- Select Host as LocalHost and Port as whatever you have configured.
- Click Apply and Debug.

Also Read: Top 25 Common Interview Questions

This brings us to the end of the Spring Boot interview questions. We hope these questions help you prepare effectively for your upcoming interview and boost your confidence. To further enhance your understanding and gain more in-depth knowledge, explore **free certification courses** on Spring Boot and related technologies to give you an extra edge in your interview preparation.

Spring Boot Interview Questions FAQs

This FAQ section on Spring Boot interview questions covers some of the most commonly asked questions about Spring Boot, including questions about its features, working, etc.

1. What is a spring boot? Why should you use it?
Spring Boot provides a good platform for Java developers to reduce overall development time and increase efficiency by integrating tests. One can choose Spring Boot because it provides powerful batch processing, eases dependency management, and no manual configurations are needed.

2. What is the main class in spring boot?
The main class in spring boot is configured automatically by the "public static void **main**()" method that starts up the Spring ApplicationContext.

3. What are the spring boot features?
Some of the important spring boot features are mentioned below:
Admin support: Springboot's admin support feature is used to manage application remotely and also provides the facility to enable admin-related features for the application.
Externalized Configuration: Spring Boot's externalized configuration helps the developers to work with the same application code in a different environment.
Profiles: Springboot's profile feature provides a way to segregate parts of your application and make it be available only in certain environments.
Logging: Springboot's logging feature uses "Commons Logging" for all internal logging.
Internationalization: Springboot's internationalization feature supports localized messages i.e your application can cater to different language preferences.
JSON: Spring Boot provides integration of three JSON libraries like Gson, Jackson, JSON-B.
Web applications: Spring Boot is one of the platforms that is well suited for web applications.
Security: Spring boot is by default secure with basic authentication on all HTTP endpoints.

4. How does spring boot handle exc
Springboot's exception handler is an annotation that is used to handle the specific exceptions with the help of @ExceptionHandler annotation.

5. How does spring boot Microservice discover dependent Microservices?

Eureka service can discover dependent microservices in spring boot to get the job done. This service will register all the client microservices through the eureka server to get the dependent microservice.

6. What is a bean in spring?
In Spring, the bean is defined as an object that is like a backbone of your application, managed by a Spring IoC container.

7. What is spring boot Microservices?
Spring Boot microservices enables production-ready applications to iterate fast and provide non-functional features. This is the reason why spring boot microservices has become the de facto standard for Java™. In microservices, you can write code for your single functionality. You can use different technology stacks for different microservices as per the skill set. You can develop this type of microservices with the help of Spring boot very quickly as spring boot gives priority to convention over configuration which increases the productivity of your developers.

8. What is the classpath in spring boot?
Classpath in spring boot is defined as a path where you place resources. During the development, stage maven will take files and place them in the appropriate place for you to use them.

9. How does spring boot application work?
Springboot can configure your application automatically based on the dependencies of the project by using @EnableAutoConfiguration annotation.