# Top 50 Spring Boot Interview Questions That Are A Must in 2023

## ****Spring**** ****Boot Interview Questions****

### **Q1.**[Spring](https://www.edureka.co/blog/spring-tutorial/)**vs Spring Boot**

|  |  |
| --- | --- |
| **Spring** | **Spring** Boot |
| A web application framework based on Java | A module of Spring |
| Provides tools and libraries to create customized web applications | Used to create a Spring application project which can just run/ execute |
| Spring is more complex than Spring Boot | Spring Boot is less complex than the Spring framework |
| Takes an unopinionated view | Takes an opinionated view of a platform |

### **Q2. What is Spring Boot and mention the need for it?**

Spring Boot is a Spring module that aims to simplify the use of the Spring framework for Java development. It is used to create stand-alone Spring-based applications that you can just run. So, it basically removes a lot of configurations and dependencies. Aiming at the Rapid Application Development, Spring Boot framework comes with the auto-dependency resolution, embedded HTTP servers, auto-configuration, management endpoints, and [Spring Boot CLI](https://www.edureka.co/blog/spring-boot-setup-helloworld-microservices-example/).

So, if you ask me why should anybody use Spring Boot, then I would say, Spring Boot not only improves productivity but also provides a lot of conveniences to write your own business logic.

### **Q3. Mention the advantages of Spring Boot**

The advantages of Spring Boot are as follows:

* Provides auto-configuration to load a set of default configuration for a quick start of the application
* Creates stand-alone applications with a range of non-functional features that are common to large classes of projects
* It comes with embedded tomcat, servlet containers jetty to avoid the usage of WAR files
* Spring Boot provides an opinionated view to reduce the developer effort and simplify maven configurations
* Provides CLI tool to develop and test applications
* Comes with Spring Boot starters to ensure dependency management and also provides various security metrics
* Consists of a wide range of APIs for monitoring and managing applications in dev and prod.
* Integrates with Spring Ecosystem like Spring [JDBC](https://www.edureka.co/blog/connect-mysql-database-in-java), Spring ORM, Spring Data, Spring Security easily by avoiding boilerplate code.

### **Q4. Mention a few features of Spring Boot.**

Few important features of Spring Boot are as follows:

1. Spring CLI – Spring Boot CLI allows you to Groovy for writing Spring boot application and avoids boilerplate code.
2. Starter Dependency – With the help of this feature, Spring Boot aggregates common dependencies together and eventually improves productivity
3. Auto-Configuration – The auto-configuration feature of Spring Boot helps in loading the default configurations according to the project you are working on. In this way, you can avoid any unnecessary WAR files.
4. Spring Initializer – This is basically a web application, which can create an internal project structure for you. So, you do not have to manually set up the structure of the project, instead, you can use this feature.
5. Spring Actuator –  This feature provides help while running Spring Boot applications.
6. Logging and Security – The logging and security feature of Spring Boot, ensures that all the applications made using Spring Boot are properly secured without any hassle.

### **Q5. Explain how to create a Spring Boot application using Maven.**

Well, there are various approaches to [create a Spring Boot application](https://www.edureka.co/blog/microservices-with-spring-boot) using maven, but if I have to name a few, then following are the ways to create a Spring Boot project/ application using [maven](https://www.edureka.co/blog/maven-in-java/):

* Spring Boot CLI
* Spring Starter Project Wizard
* Spring Initializr
* Spring Maven Project

### **Q6. Mention the possible sources of external configuration.**

There is no doubt in the fact that Spring Boot allows the developers to run the same application in different environments. Well, this is done with the support it provides for external configuration. It uses environment variables, properties files, command-line arguments, YAML files, and system properties to mention the required configuration properties. Also, the @value annotation is used to gain access to the properties. So, the most possible sources of external configuration are as follows:

* **Application Properties –** By default, Spring Boot searches for the application properties file or its YAML file in the current directory, classpath root or config directory to load the properties.
* **Command-line properties –** Spring Boot provides command-line arguments and converts these arguments to properties. Then it adds them to the set of environment properties.
* **Profile-specific properties –**  These properties are loaded from the application-{profile}.properties file or its YAML file. This file resides in the same location as that of the non-specific property files and the{profile} placeholder refers to an active profile.

### Q7. Can you explain what happens in the background when a Spring Boot Application is “Run as Java Application”?

When a Spring Boot application is executed as “Run as Java application”, then it automatically launches up the tomcat server as soon as it sees, that you are developing a web application. To learn more about Java, it’s recommended to join [Java training course](https://www.edureka.co/java-j2ee-training-course) today at Edureka.

### **Q8. What are the Spring Boot starters and what are available the starters?**

Spring Boot starters are a set of convenient dependency management providers that can be used in the application to enable dependencies. These starters, make development easy and rapid. All the available starters come under the org.springframework.boot group. Few of the popular starters are as follows:

* spring-boot-starter: – This is the core starter and includes logging, auto-configuration support, and YAML.
* spring-boot-starter-jdbc – This starter is used for HikariCP connection pool with JDBC
* spring-boot-starter-web – Is the starter for building web applications, including RESTful, applications using Spring MVC
* spring-boot-starter-data-jpa – Is the starter to use Spring Data JPA with Hibernate
* spring-boot-starter-security – Is the starter used for Spring Security
* spring-boot-starter-aop: This starter is used for aspect-oriented programming with AspectJ and  Spring AOP
* spring-boot-starter-test: Is the starter for testing Spring Boot applications

### **Q9. Explain Spring Actuator and its advantages.**

Spring Actuator is a cool feature of Spring Boot with the help of which you can see what is happening inside a running application. So, whenever you want to debug your application, and need to analyze the logs you need to understand what is happening in the application right? In such a scenario, the Spring Actuator provides easy access to features such as identifying beans, CPU usage, etc. The Spring Actuator provides a very easy way to access the production-ready REST points and fetch all kinds of information from the web. These points are secured using Spring Security’s content negotiation strategy.

### **Q10. What is Spring Boot dependency management?**

Spring Boot dependency management is basically used to manage dependencies and configuration automatically without you specifying the version for any of that dependencies.

### **Q11. Mention the minimum requirements for a Spring boot System.**

Spring Boot 2.1.7.RELEASE requires

* Java 8 +
* Spring Framework 5.1.9 +

**Explicit build support**

* Maven 3.3+
* Gradle 4.4+

**Servlet Container Support**

* Tomcat 9.0 – Servlet Version 4.0
* Jetty 9.4 –  Servlet Version 3.1
* Undertow 2.0 – Servlet Version 4.0

### **Q12. Explain what is thymeleaf and how to use thymeleaf?**

Thymeleaf is a server-side Java template engine used for web applications. It aims to bring natural template for your web application and can integrate well with Spring Framework and HTML5 Java web applications. To use Thymeleaf, you need to add the following code in the pom.xml file:

|  |  |
| --- | --- |
| 1  2  3  4 | <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-thymeleaf</artifactId>  </dependency> |

### **Q13. Can we change the port of the embedded Tomcat server in Spring boot?**

Yes, we can change the port of the embedded tomcat server by using the application properties file. In this file, you have to add a property of “server.port” and assign it to any port you wish to. For example, if you want to assign it to 8081, then you have to mention server.port=8081. Once you mention the port number, the application properties file will be automatically loaded by Spring Boot and the required configurations will be applied on to the application.

### **Q14. What is the need for Spring Boot DevTools?**

Spring Boot Dev Tools are an elaborated set of tools and aims to make the process of developing an application easier. If the application runs in the production, then this module is automatically disabled, repackaging of archives are also excluded by default. So, the Spring Boot Developer Tools applies properties to the respective development environments.  To include the DevTools, you just have to add the following dependency into the pom.xml file:

|  |  |
| --- | --- |
| 1  2  3  4 | <dependency>      <groupId>org.springframework.boot</groupId>      <artifactId>spring-boot-devtools</artifactId>  </dependency> |

### **Q15. Mention the steps to create a Spring Boot project using Spring  Initializer.**

Spring Initializr is a web tool provided by Spring. With the help of this tool, you can create Spring Boot projects by just providing project details. The following steps need to be followed to create a Spring Boot project using Spring Initializer:

* Choose the maven project and the required dependencies. Then, fill in the other required details like Group, Artifact, and then click on Generate Project.
* Once the project is downloaded, extract the project onto your system
* Next, you have to import this project using the import option on the Spring Tool Suite IDE
  + While importing the project, remember that you have to choose the project type to be Maven and the source project should contain the pom.xml file.

Once, all the above steps are followed you will see that the Spring Boot project is created with all the required dependencies.

### **Q16. Mention the steps to connect Spring Boot application to a database using JDBC.**

Spring Boot starter projects provide the required libraries to connect the application with JDBC. So, for example, if you just have to create an application  and connect it with [MySQL](https://www.edureka.co/blog/mysql-tutorial/) database, you can follow the below steps:

**Step 1:** Create a database in MySQL

|  |  |
| --- | --- |
| 1 | CREATE DATABASE example; |

**Step 2:**Then you have to create a table inside this database.

|  |  |
| --- | --- |
| 1 | CREATE TABLE customers(customerid INT PRIMARY KEY NOT NULL AUTO\_INCREMENT, customername VARCHAR(255)); |

**Step 3:** Now, create a Spring Boot project and provide the required details

**Step 4:** Add the JDBC, MySQL and web dependencies.

**Step 5:** Once the project is created, you have to configure the database into application properties

|  |  |
| --- | --- |
| 1  2  3  4 | spring.datasource.url=jdbc:mysql://localhost:3306/example  spring.datasource.username=root  spring.datasource.password=edureka  spring.jpa.hibernate.ddl-auto=create-drop |

**Step 6:** The main application.java class should have the following code:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | package com.edureka;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  public class SampleApplication {      public static void main(String[] args) {          SpringApplication.run(SampleApplication.class, args);      }  } |

**tep 7:** Next, you have to create a controller to handle the HTTP requests, by mentioning the following code:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | package com.edureka;  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("/insert")  public String index(){  jdbc.execute("insert into customers(name)values('Aryya')");  return "Data Entry Successful";  }  } |

**Step 8:** Finally, execute this project as a Java application.  
**Step 9:** Next, open the URL (localhost:8080/insert), and you will see the output as Data Entry Successful. You can also go forward and check if the data is entered into the table.

### **Q17. How to enable HTTP/2 support in Spring Boot?**

You can enable the HTTP/2 support in Spring Boot by: server.http2.enabled=true

### **Q18.  What are the @RequestMapping  and @RestController annotation in Spring Boot used for?**

|  |  |
| --- | --- |
| **@RequestMapping** | **@RestController** |
| This annotation is used to provide the routing information and tells to Spring that any HTTP request must be mapped to the respective method. | This annotation is used to add the @ResponseBody and @Controller annotation to the class |
| To use this annotation, you have to import org.springframework.web.  bind.annotation.RequestMapping; | To use this annotation, you have to import org.springframework.web.  bind.annotation.RestController; |

**Example:** Consider you have a method example() which should map with /example URL.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | package com.edureka;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RestController;  @RestController  public class SampleController {  @RequestMapping("/example")  public String example(){  return"Welcome To Edureka";  }  } |

### **Q19. What is Spring Boot CLI and how to execute the Spring Boot project using boot CLI?**

Spring Boot CLI is a tool supported by the official [Spring Framework](https://www.edureka.co/blog/what-is-spring-framework/). The steps to execute a Spring Boot project are as follows:

* Download the CLI tool from the official site and extract the zip file. The bin folder present in the Spring setup is used to execute the Spring Boot application.
* Since Spring Boot CLI executes groovy files, you need to create a groovy file for Spring Boot application. So, to do that, open terminal and change the current directory to the bin folder. Now, open a groovy file (for example Sample.groovy)
* In this file create a controller as follows:

@RestController public class Sample {

@RequestMapping("/example")

String index(){

<h1>"Welcome To Edureka"</h1>;

} }

* Then execute the groovy file by mentioning:

|  |  |
| --- | --- |
| 1 | ./spring run Sample.groovy; |

Once, the project is executed go to the URL(localhost:8080:/example) and you will see the output as **Welcome To Edureka**

In case you are facing any challenges with these Spring Boot interview questions, please comment on your problems in the comment section below.

### **Q20. Mention the differences between JPA and**[Hibernate.](https://www.edureka.co/blog/what-is-hibernate-in-java/)

|  |  |
| --- | --- |
| **JPA** | **Hibernate** |
| JPA is a Data Access Abstraction used to reduce the amount of boilerplate code | Hibernate is an implementation of Java Persistence API and offers benefits of loose coupling |

### Q21. How can we create a custom endpoint in Spring Boot Actuator?

To create a custom endpoint in Spring Boot 2.x, you can use the @Endpoint annotation. Spring Boot also exposes endpoints using @WebEndpointor, @WebEndpointExtension over HTTP with the help of [Spring MVC](https://www.edureka.co/blog/spring-mvc-tutorial/), [Jersey](https://www.edureka.co/blog/java-web-services-tutorial/), etc.

### **Q22. Explain Spring Data.**

Spring Data aims to make it easy for the developers to use relational and non-relational databases, cloud-based data services, and other data access technologies. So, basically, it makes it easy for data access and still retains the underlying data.

### Q23. What do you understand by auto-configuration in Spring Boot and how to disable the auto-configuration?

Auto-configuration is used to automatically configure the required configuration for the application. For example, if you have a data source bean present in the classpath of the application, then it automatically configures the [JDBC template](https://www.edureka.co/blog/connect-mysql-database-in-java). With the help of auto-configuration, you can create a Java application in an easy way, as it automatically configures the required beans, controllers, etc.

To disable the auto-configuration property, you have to exclude attribute of @EnableAutoConfiguration, in the scenario where you do not want it to be applied.

|  |  |
| --- | --- |
| 1 | @EnableAutoConfiguration(exclude={DataSourceAutoConfiguration.class}) |

If the class is not on the classpath, then to exclude the auto-configuration, you have to mention the following code:

|  |  |
| --- | --- |
| 1 | @EnableAutoConfiguration(excludeName={Sample.class}) |

Apart from this, Spring Boot also provides the facility to exclude list of auto-configuration classes by using the spring.autoconfigure.exclude property. You can go forward, and add it either in the application.properties or add multiple classes with comma-separated.

### Q24. What are the differences between @SpringBootApplication and @EnableAutoConfiguration annotation?

|  |  |
| --- | --- |
| **@SpringBootApplication** | **@EnableAutoConfiguration** |
| Used in the main class or bootstrap class | Used to enable auto-configuration  and component scanning in your project |
| It is a combination of @Configuration, @ComponentScan and @EnableAutoConfiguration annotations. | It is a combination of @Configuration and @ComponentScan annotations |

### Q25. What are the steps to deploy Spring Boot web applications as JAR and WAR files?

To deploy a Spring Boot web application, you just have to add the following plugin in the pom.xml file:

|  |  |
| --- | --- |
| 1  2  3  4 | <plugin>      <groupId>org.springframework.boot</groupId>      <artifactId>spring-boot-maven-plugin</artifactId>  </plugin> |

By using the above plugin, you will get a JAR executing the package phase. This JAR will contain all the necessary libraries and dependencies required. It will also contain an embedded server. So, you can basically run the application like an ordinary JAR file.  
**Note:** The packaging element in the pom.xml file must be set to **jar** to build a JAR file as below:

|  |  |
| --- | --- |
| 1 | <packaging>jar</packaging> |

Similarly, if you want to build a WAR file, then you will mention

|  |  |
| --- | --- |
| 1 | <packaging>war</packaging> |

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

Example for ReadOnly as TRUE in transaction management could be as follows:

Consider a scenario, where you have to read data from the database. For example, let us say you have a customer database, and you want to read the customer details such as customerID, and customername. To do that, you will set**read-only on the transaction** as we do not want to check for the changes in the entities.

### Q27. Can you explain how to deploy to a different server with Spring Boot?

To deploy a different server with Spring Boot, follow the below steps:

* Generate a WAR from the project
* Then, deploy the WAR file onto your favorite server

Note: The steps to deploy the WAR file on the server is dependent on the server you choose.

Want to upskill yourself to get ahead in your career? Check out this video

### Q28: What is the best way to expose custom application configuration with Spring Boot?

One way to expose the custom application [configuration in Spring](https://www.edureka.co/blog/spring-tutorial/) Boot is by using the **@Value annotation**. But, the only problem with this annotation is that all the configuration values will be distributed throughout the application. Instead, you can use a centralized approach.

By centralized approach, I mean that you can define a configuration component using the @ConfigurationProperties as follows:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | @Component  @ConfigurationProperties("example")  public class SampleConfiguration {  private int number;  private boolean value;  private String message; |

According to the above snippet, the values configured in application.properties will be as follows:

|  |  |
| --- | --- |
| 1  2  3 | example.number: 100  example.value: true  example.message: Dynamic Message |

### **Q29. Can we create a non-web application in Spring Boot?**

Yes, we can create a non-web application by removing the web dependencies from the classpath along with changing the way Spring Boot creates the application context.

### Q 30. What are the steps to connect an external database like MySQL or Oracle?

To connect an external database, you have to follow the below steps:

* Start by adding the dependency for MySQL Connector to pom.xml
* Then remove H2 Dependency from pom.xml
* Now, set up your [MySQL database](https://www.edureka.co/blog/mysql-tutorial/) and configure your connection to the MySQL database
* Restart your project

## ****Spring**** ****Boot Interview Questions****

### Q31. Mention the advantages of the YAML file than Properties file and the different ways to load YAML file in Spring boot.

The advantages of the YAML file than a properties file is that the data is stored in a hierarchical format. So, it becomes very easy for the developers to debug if there is an issue. The SpringApplication class supports the YAML file as an alternative to properties whenever you use the SnakeYAML library on your classpath. The different ways to load a YAML file in Spring Boot is as follows:

* Use YamlMapFactoryBean to load YAML as a Map
* Use YamlPropertiesFactoryBean to load YAML as Properties

### Q32. How is Hibernate chosen as the default implementation for JPA without any configuration?

When we use the Spring Boot Auto Configuration, automatically the spring**-boot-starter-data-jpa**dependency gets added to the pom.xml file. Now, since this dependency has a transitive dependency on JPA and Hibernate, Spring Boot automatically auto-configures Hibernate as the default implementation for JPA, whenever it sees Hibernate in the classpath.

### Q33. What do you understand by Spring Data REST?

Spring Data REST is used to expose the RESTful resources around Spring Data repositories. Consider the following example:

|  |  |
| --- | --- |
| 1  2  3 | @RepositoryRestResource(collectionResourceRel = "sample", path = "sample")  public interface SampleRepository          extends CustomerRepository<sample, Long> { |

Now, to expose the REST services, you can use the POST method in the following way:

|  |  |
| --- | --- |
| 1  2  3 | {  "customername": "Rohit"  } |

Response Content

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | {  "customername": "Rohit"  "\_links": {  "self": {  "href": "<http://localhost:8080/sample/1>"  },  "sample": {  "href": "<http://localhost:8080/sample/1>"  }  } |

Observe that the response content contains the href of the newly created resource.

### Q34. What is the difference between RequestMapping and GetMapping?

The @GetMapping is a composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.GET). Both these methods support the consumes. The consume options are :

consumes = “text/plain”  
consumes = {“text/plain”, “application/\*”}

### Q35. In which layer, should the boundary of a transaction start?

The boundary of the transaction should start from the Service Layer since the logic for the business transaction is present in this layer itself.

### **Q36.**How does path=”sample”, collectionResourceRel=”sample” work with Spring Data Rest?

|  |  |
| --- | --- |
| 1  2  3 | @RepositoryRestResource(collectionResourceRel = "sample", path = "sample")  public interface SampleRepository extends  PagingAndSortingRepository<Sample, Long> |

* path – This section is used to mention the segment under which the resource is to be exported.
* collectionResourceRel – This value is used to generate links to the collection resource.

### Q37. Explain how to register a custom auto-configuration.

In order to register an auto-configuration class, you have to mention the fully-qualified name under the @EnableAutoConfiguration key META-INF/spring. factories file. Also, if we build the with maven, then this file should be placed in the resources/META-INT directory.

### **Q38. How do you Configure Log4j for logging?**

Since Spring Boot supports Log4j2 for logging a configuration, you have to exclude Logback and include Log4j2 for logging. This can be only done if you are using the starters project.

### **Q39. Mention the differences between WAR and embedded containers**

|  |  |
| --- | --- |
| WAR | Embedded Containers |
| WAR benefits a considerable measure from Spring Boot | Only one component of Spring Boot and is utilized during improvements |

### Q40. What do you think is the need for Profiles?

Profiles are used to provide a way to segregate the different parts of the application configuration and make it available for various environments. So, basically, any @Component or a @Configuration can be marked with a @Profile to limit as it is loaded. Consider you have multiple environments,

* Dev
* QA
* Stage
* Production

Now, let’s say, you want to have different application configuration in each of the environments, you can use profiles to have different application configurations for different environments. So, basically, Spring and Spring Boot provide features through which you can specify:

* The active profile for a specific environment
* The configuration of various environments for various profiles.

### Q41. What are the steps to add a custom JS code with Spring Boot?

The steps to add a [custom JS code](https://www.edureka.co/blog/javascript-tutorial/) with Spring Boot are as follows:

* Now, create a folder and name it **static** under the resources folder
* In this folder, you can put the static content in that folder

**Note:** Just in case, the browser throws an unauthorized error, you either disable the security or search for the password in the log file, and eventually pass it in the request header.

### Q42. How to instruct an auto-configuration to back off when a bean exists?

To instruct an auto-configuration class to back off when a bean exists, you have to use the @ConditionalOnMissingBean annotation. The attributes of this annotation are as follows:

* **value:** This attribute stores the type of beans to be checked
* **name:** This attribute stores the name of beans to be checked

### Q43. Why is Spring Data REST not recommended in real-world applications?

Spring Data REST is not recommended in real-world applications as you are exposing your database entities directly as [REST Services](https://www.edureka.co/blog/what-is-rest-api/). While designing RESTful services, the two most important things that we consider is the domain model and the consumers. But, while using Spring Data REST, none of these parameters are considered. The entities are directly exposed. So, I would just say, you can use Spring Data REST, for the initial evolution of the project.

### Q44. What is the error you see if  H2 is not in the classpath?

If H2 is not present in the classpath, then you see the following error:

Cannot determine embedded database driver class for database type NONE

To resolve this error, add H2 to the pom.xml file, and restart your server.  
The following code snippet can be added to add the dependency:

|  |  |
| --- | --- |
| 1  2  3  4  5 | <dependency>      <groupId>com.h2database</groupId>      <artifactId>h2</artifactId>      <scope>runtime</scope>  </dependency> |

### Q45. What is the way to use profiles to configure the environment-specific configuration with Spring Boot?

Since it is a known fact that a Profile is nothing but a key to identify an environment lets consider the following two profiles in the example:

* dev
* prod
* Consider the following properties present in the application properties file:

example.number: 100  
example.value: true  
example.message: Dynamic Message

Now, say you want to customize the application.properties for dev profile, then you need to create a file with name application-dev.properties and override the properties that you want to customize. You can mention the following code:

example.message: Dynamic Message in Dev

Similarly, if you want to customize the application.properties for prod profile, then you can mention the following code snippet:

example.message: Dynamic Message in Prod

Once you are done with the profile-specific configuration, you have to set the active profile in an environment. To do that, either you can

* Use -Dspring.profiles.active=prod in  arguments
* Use spring.profiles.active=prod in application.properties file

### Q46. Mention the dependencies needed to start up a JPA Application and connect to in-memory database H2 with Spring Boot?

The dependencies are needed to start up a JPA Application and connect to in-memory database H2 with Spring Boot

* web starter
* h2
* data JPA starter
* To include the dependencies refer to the following code:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <dependency>      <groupId>org.springframework.boot</groupId>      <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>      <groupId>com.h2database</groupId>      <artifactId>h2</artifactId>      <scope>runtime</scope>  </dependency>  <dependency>      <groupId>org.springframework.boot</groupId>      <artifactId>spring-boot-starter-data-jpa</artifactId>  </dependency> |

### Q47. What do you understand by Spring Boot supports relaxed binding?

Relaxed binding, is a way in which, the property name does not need to match the key of the environment property. In Spring Boot, relaxed binding is applicable to the type-safe binding of the configuration properties. For example, if a property in a bean class with the @ConfigurationPropertie annotation is used sampleProp, then it can be bounded to any of the following environment properties:

* sampleProp
* sample-Prop
* sample\_Prop
* SAMPLE\_PROP

### Q48.  Where is the database connection information specified and how does it automatically connect to H2?

Well, the answer to this question is very simple. It is because of the Spring Boot auto-configuration that, configures the dependencies of the application. So, the database connection information, and automatically connecting the database to H2 is done by the auto-configuration property.

### Q49. What is the name of the default H2 database configured by Spring Boot?

The name of the default H2 database is **testdb.  Refer below:**

spring.datasource.name=testdb # Name of the datasource.

**Note:** Just incase if you are using H2 in-memory database, then exactly that is the name of Spring Boot which is used to setup your H2 database.

## ****Spring**** ****Boot Interview Questions****

### **Q50. Do you think, you can use jetty instead of tomcat in spring-boot-starter-web?**

Yes, we can use jetty instead of tomcat in spring-boot-starter-web, by removing the existing dependency and including the following:

|  |  |
| --- | --- |
|  |  |

With this, we come to an end to this article on Spring Boot Interview Questions. I hope this set of Spring Boot Interview Questions and Answers will help you in preparing for your interviews. All the best! If you want to learn Spring and wish to use it while developing Java applications, then check out the [***Spring Certification***](https://www.edureka.co/spring-certification-course)Training by Edureka, a trusted online learning company with a network of more than 250,000 satisfied learners spread across the globe.

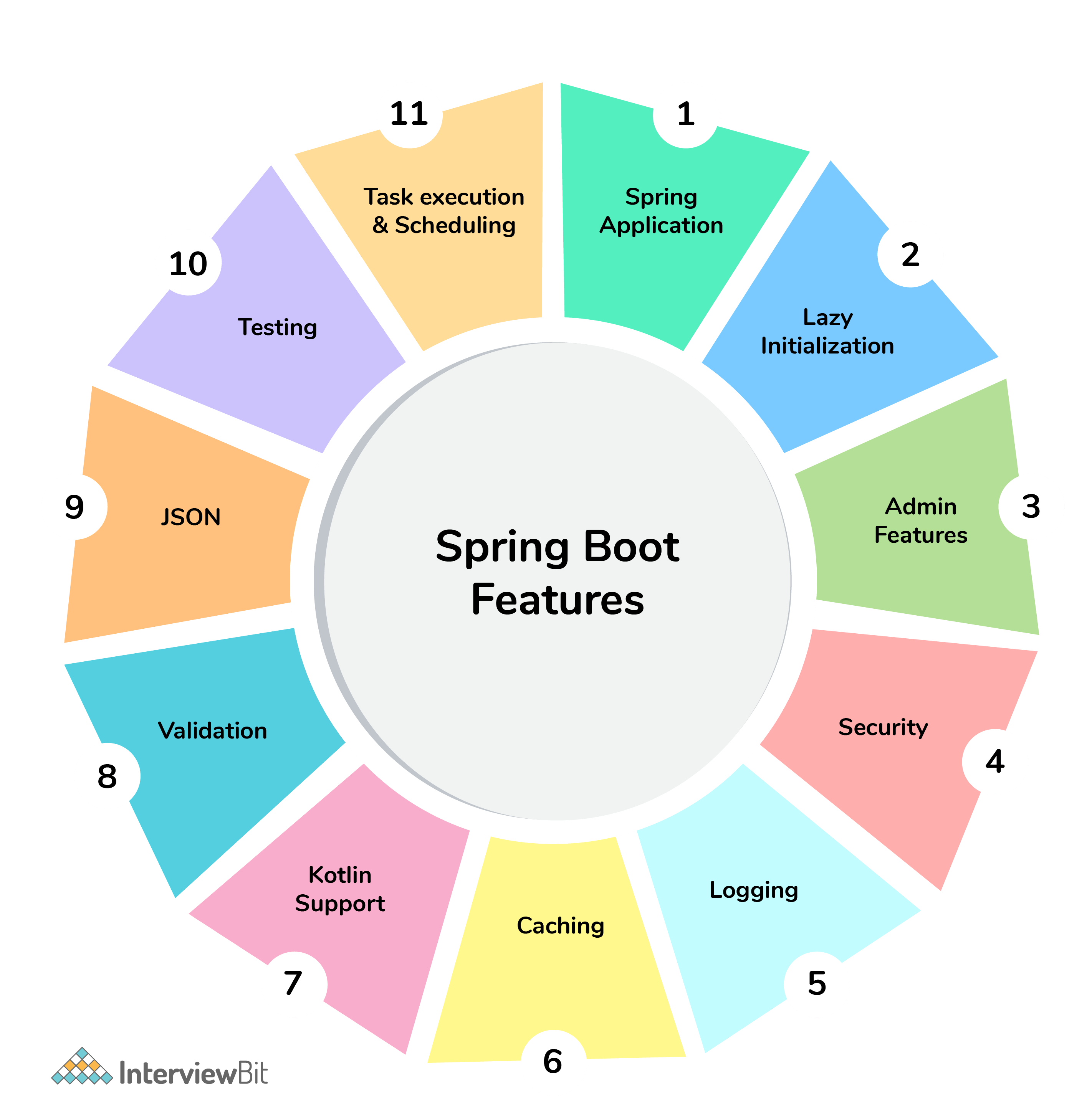
Got a question for us? Please mention it in the comments section of “Spring Boot Interview Questions” and we will get back to you.

## What is Spring boot?

[**Sprint boot**](https://www.interviewbit.com/java-interview-questions/) is a Java-based spring framework used for Rapid Application Development (to build stand-alone microservices). It has extra support of auto-configuration and embedded application server like tomcat, jetty, etc.

## Features of Spring Boot that make it different?

* Creates stand-alone spring application with minimal configuration needed.
* It has embedded tomcat, jetty which makes it just code and run the application.
* Provide production-ready features such as metrics, health checks, and externalized configuration.
* Absolutely no requirement for XML configuration.

Spring Boot Features



## Spring Boot Interview Questions For Freshers

### 1. What are the advantages of using Spring Boot?

The advantages of Spring Boot are listed below:

* Easy to understand and develop spring applications.
* Spring Boot is nothing but an existing framework with the addition of an embedded HTTP server and annotation configuration which makes it easier to understand and faster the process of development.
* Increases productivity and reduces development time.
* Minimum configuration.
* We don’t need to write any XML configuration, only a few annotations are required to do the configuration.

### 2. What are the Spring Boot key components?

Below are the four key components of spring-boot:

* Spring Boot auto-configuration.
* Spring Boot CLI.
* Spring Boot starter POMs.
* Spring Boot Actuators.

### 3. Why Spring Boot over Spring?

Below are some key points which spring boot offers but spring doesn’t:

* Starter POM.
* Version Management.
* Auto Configuration.
* Component Scanning.
* Embedded server.
* InMemory DB.
* Actuators

Spring Boot simplifies the spring feature for the user:

### 4. What is the starter dependency of the Spring boot module?

Spring boot provides numbers of starter dependency, here are the most commonly used -

* Data JPA starter.
* Test Starter.
* Security starter.
* Web starter.
* Mail starter.
* Thymeleaf starter.

### 5. How does Spring Boot works?

Spring Boot automatically configures your application based on the dependencies you have added to the project by using annotation. The entry point of the spring boot application is the class that contains @SpringBootApplication annotation and the main method.

Spring Boot automatically scans all the components included in the project by using @ComponentScan annotation.

### 6. What does the @SpringBootApplication annotation do internally?

The @SpringBootApplication annotation is equivalent to using @Configuration, @EnableAutoConfiguration, and @ComponentScan with their default attributes. Spring Boot enables the developer to use a single annotation instead of using multiple. But, as we know, Spring provided loosely coupled features that we can use for each annotation as per our project needs.

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

Spring Boot application scans all the beans and package declarations when the application initializes. You need to add the @ComponentScan annotation for your class file to scan your components added to your project.

### 8. How does a spring boot application get started?

Just like any other Java program, a Spring Boot application must have a main method. This method serves as an entry point, which invokes the SpringApplication#run method to bootstrap the application.

@SpringBootApplication

**public** **class** **MyApplication** {

**public** **static** **void** **main**(String[] args) {

SpringApplication.run(MyApplication.class);

// other statements

}

}

### 9. What are starter dependencies?

Spring boot starter is a maven template that contains a collection of all the relevant transitive dependencies that are needed to start a particular functionality.  
Like we need to import spring-boot-starter-web dependency for creating a web application.

<dependency>

<groupId> org.springframework.boot</groupId>

<artifactId> spring-boot-starter-web </artifactId>

</dependency>

### 10. What is Spring Initializer?

Spring Initializer is a web application that helps you to create an initial spring boot project structure and provides a maven or gradle file to build your code. It solves the problem of setting up a framework when you are starting a project from scratch.

### 11. What is Spring Boot CLI and what are its benefits?

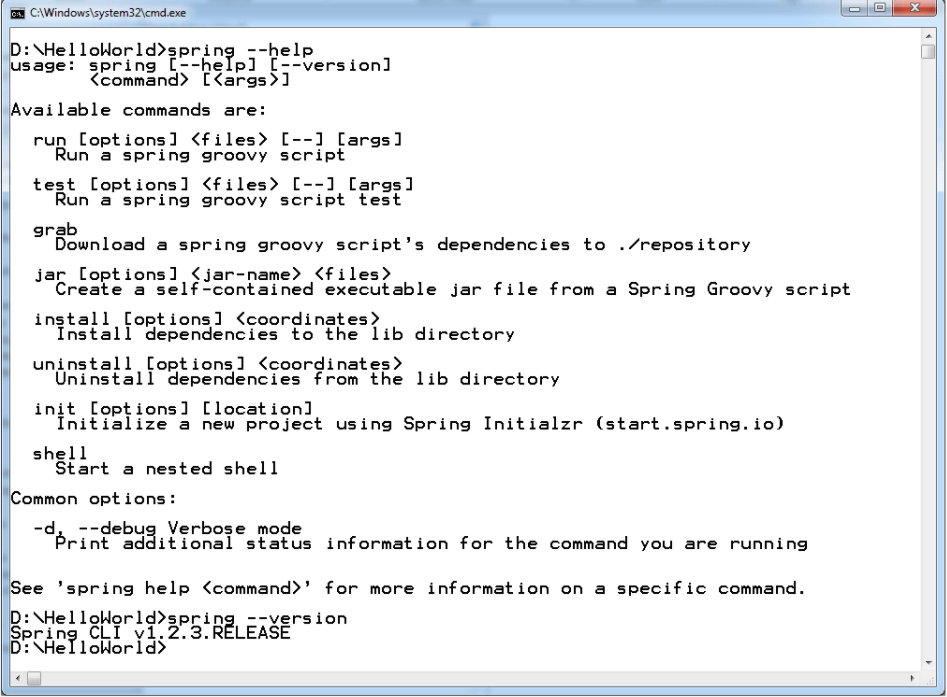
Spring Boot CLI is a command-line interface that allows you to create a spring-based java application using Groovy.

Example: You don’t need to create getter and setter method or access modifier, return statement. If you use the JDBC template, it automatically loads for you.

### 12. What are the most common Spring Boot CLI commands?

-run, -test, -grap, -jar, -war, -install, -uninstall, --init, -shell, -help.

To check the description, run spring --help from the terminal.

Spring Boot CLI Commands

## Advanced Spring Boot Questions

### 13. What Are the Basic Annotations that Spring Boot Offers?

The primary annotations that Spring Boot offers reside in its org.springframework.boot.autoconfigure and its sub-packages. Here are a couple of basic ones:

@EnableAutoConfiguration – to make Spring Boot look for auto-configuration beans on its classpath and automatically apply them.

@SpringBootApplication – used to denote the main class of a Boot Application. This annotation combines @Configuration, @EnableAutoConfiguration, and @ComponentScan annotations with their default attributes.

### 14. What is Spring Boot dependency management?

Spring Boot dependency management is used to manage dependencies and configuration automatically without you specifying the version for any of that dependencies.

### 15. Can we create a non-web application in Spring Boot?

Yes, we can create a non-web application by removing the web dependencies from the classpath along with changing the way Spring Boot creates the application context.

### 16. Is it possible to change the port of the embedded Tomcat server in Spring Boot?

Yes, it is possible. By using the **server.port** in the **application.properties**.

### 17. What is the default port of tomcat in spring boot?

The default port of the tomcat server-id 8080. It can be changed by adding **sever.port** properties in the **application.property** file.

### 18. Can we override or replace the Embedded tomcat server in Spring Boot?

Yes, we can replace the Embedded Tomcat server with any server by using the Starter dependency in the **pom.xml** file. Like you can use spring-boot-starter-jetty as a dependency for using a jetty server in your project.

### 19. Can we disable the default web server in the Spring boot application?

Yes, we can use **application.properties** to configure the web application type i.e **spring.main.web-application-type=none.**

### 20. How to disable a specific auto-configuration class?

You can use exclude attribute of @EnableAutoConfiguration if you want auto-configuration not to apply to any specific class.

//use of exclude

@EnableAutoConfiguration(exclude={className})

### 21. Explain @RestController annotation in Spring boot?

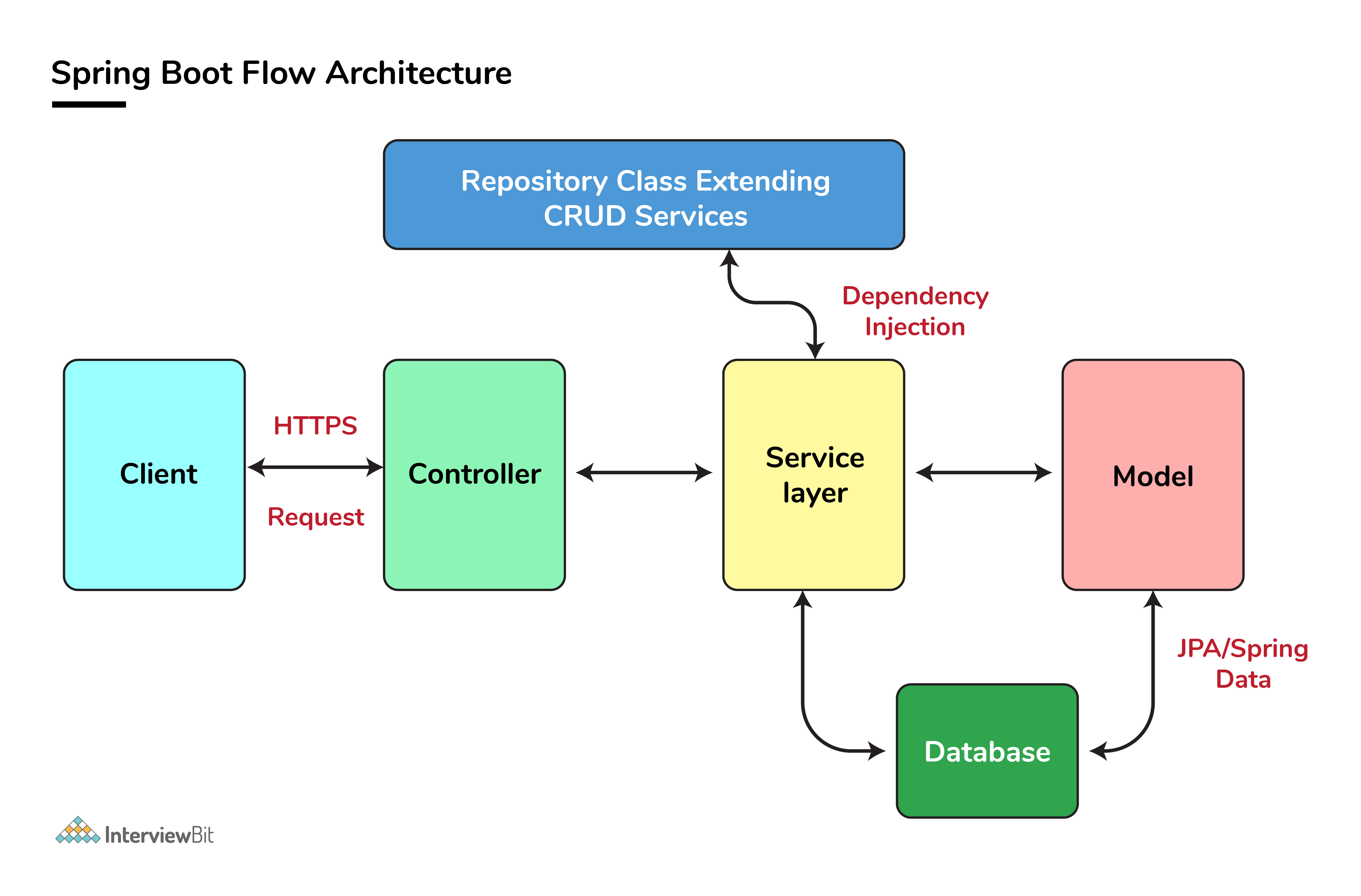
It is a combination of @Controller and @ResponseBody, used for creating a restful controller. It converts the response to JSON or XML. It ensures that data returned by each method will be written straight into the response body instead of returning a template.

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

@Controller Map of the model object to view or template and make it human readable but @RestController simply returns the object and object data is directly written in HTTP response as JSON or XML.

### 23. Describe the flow of HTTPS requests through the Spring Boot application?

On a high-level spring boot application follow the MVC pattern which is depicted in the below flow diagram.

Spring Boot Flow Architecture

### 24. What is the difference between RequestMapping and GetMapping?

RequestMapping can be used with GET, POST, PUT, and many other request methods using the method attribute on the annotation. Whereas getMapping is only an extension of RequestMapping which helps you to improve on clarity on request.

### 25. What is the use of Profiles in spring boot?

While developing the application we deal with multiple environments such as dev, QA, Prod, and each environment requires a different configuration. For eg., we might be using an embedded H2 database for dev but for prod, we might have proprietary Oracle or DB2. Even if DBMS is the same across the environment, the URLs will be different.

To make this easy and clean, Spring has the provision of Profiles to keep the separate configuration of environments.

### 26. What is Spring Actuator? What are its advantages?

An actuator is an additional feature of Spring that helps you to monitor and manage your application when you push it to production. These actuators include auditing, health, CPU usage, HTTP hits, and metric gathering, and many more that are automatically applied to your application.

### 27. How to enable Actuator in Spring boot application?

To enable the spring actuator feature, we need to add the dependency of “spring-boot-starter-actuator” in pom.xml.

<dependency>

<groupId> org.springframework.boot</groupId>

<artifactId> spring-boot-starter-actuator </artifactId>

</dependency>

### 28. What are the actuator-provided endpoints used for monitoring the Spring boot application?

Actuators provide below pre-defined endpoints to monitor our application -

* Health
* Info
* Beans
* Mappings
* Configprops
* Httptrace
* Heapdump
* Threaddump
* Shutdown

### 29. How to get the list of all the beans in your Spring boot application?

Spring Boot actuator “/Beans” is used to get the list of all the spring beans in your application.

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

Spring Boot actuator “/env” returns the list of all the environment properties of running the spring boot application.

### 31. How to enable debugging log in the spring boot application?

Debugging logs can be enabled in three ways -

* We can start the application with --debug switch.
* We can set the logging.level.root=debug property in application.property file.
* We can set the logging level of the root logger to debug in the supplied logging configuration file.

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

You can define both application and Spring boot-related properties into a file called application.properties. You can create this file manually or use Spring Initializer to create this file. You don’t need to do any special configuration to instruct Spring Boot to load this file, If it exists in classpath then spring boot automatically loads it and configure itself and the application code accordingly.

### 33. What is dependency Injection?

The process of injecting dependent bean objects into target bean objects is called dependency injection.

* Setter Injection: The IOC container will inject the dependent bean object into the target bean object by calling the setter method.
* Constructor Injection: The IOC container will inject the dependent bean object into the target bean object by calling the target bean constructor.
* Field Injection: The IOC container will inject the dependent bean object into the target bean object by Reflection API.

### 34. What is an IOC container?

IoC Container is a framework for implementing automatic dependency injection. It manages object creation and its life-time and also injects dependencies into the class.

# Spring Boot Interview Questions for 10 Years Experience

As a seasoned Spring Boot developer with 10 years of experience, you can expect to be asked some challenging interview questions during the hiring process. While the questions will vary depending on the company, industry, and specific role you are applying for, there are certain key concepts and topics that are likely to come up. In this article, we will discuss some of the most common [Spring Boot interview questions](https://talentedge.com/articles/spring-boot-interview-questions/) for 10 years experienced professionals.

1. What is Spring Boot, and how does it differ from Spring Framework?

Spring Boot is an open-source Java-based framework used to create standalone, production-grade Spring-based applications with minimal configuration. It provides a set of pre-configured dependencies and auto-configuration modules that eliminate the need for boilerplate code, making it easier and faster to create applications. Spring Boot also integrates with other Spring Framework modules such as Spring Data, Spring MVC, and Spring Security.

The main difference between Spring Boot and Spring Framework is that Spring Boot is an opinionated framework that provides a streamlined way of developing Spring-based applications, while Spring Framework is a more traditional, modular framework that requires more configuration and setup.

1. What are the advantages of using Spring Boot?

Spring Boot offers several advantages for developers, including:

* Rapid application development: Spring Boot eliminates the need for a lot of boilerplate code, allowing developers to focus on writing business logic.
* Production-ready: Spring Boot provides many pre-built components and features such as security, data access, and monitoring, making it easier to create production-grade applications.
* Microservices support: Spring Boot is an ideal framework for creating microservices-based architectures, thanks to its lightweight and modular design.
* Easy integration: Spring Boot provides integration with several other Spring modules and third-party libraries, allowing developers to easily add features to their applications.

1. What is an embedded container in Spring Boot, and why is it important?

Spring Boot includes an embedded container, which is a lightweight server that can run inside a Spring Boot application. It eliminates the need for developers to deploy their applications on external containers like Tomcat or Jetty, making it easier to develop and deploy applications. The embedded container also supports hot reloading, which allows developers to make changes to their code and see the results immediately, without having to restart the application.

1. How does Spring Boot handle database connections?

Spring Boot provides a set of auto-configuration modules that make it easy to connect to databases. By default, it supports several popular databases like MySQL, PostgreSQL, and MongoDB. Developers can specify the database they want to use in the application.properties or application.yml file, and Spring Boot will automatically configure the database connection.

1. What is Spring Security, and how does it work?

Spring Security is a module of the Spring Framework that provides security features for Java-based applications. It offers authentication and authorization services for web and non-web applications. Spring Security uses a set of filters to intercept and process requests, and it supports several authentication mechanisms such as basic authentication, form-based authentication, and OAuth2.

1. How does Spring Boot handle error handling and exception handling?

Spring Boot provides a centralized error handling mechanism that allows developers to handle errors and exceptions in a consistent way across the application. It provides a set of error handling classes and annotations, such as @ControllerAdvice and @ExceptionHandler, that developers can use to define how errors and exceptions should be handled. Spring Boot also provides detailed error messages and stack traces, which can be useful for debugging.

1. What is Spring Data, and how does it work with Spring Boot?

Spring Data is a module of the Spring Framework that provides a set of high-level abstractions for working with databases. It offers a consistent API for interacting with various data sources, such as relational databases, NoSQL databases, and in-memory data stores. Spring Boot integrates with Spring Data to provide easy

1. What are the different types of Bean scopes in Spring Boot?

In Spring Boot, there are different types of Bean scopes available. These are:

* Singleton: This is the default scope in Spring Boot. A singleton bean is created only once in the container, and the same instance is returned every time the bean is requested.
* Prototype: This scope creates a new instance of the bean every time it is requested.
* Request: This scope creates a new instance of the bean for every HTTP request.
* Session: This scope creates a new instance of the bean for every HTTP session.
* Global Session: This scope creates a new instance of the bean for every global HTTP session.

1. What is the difference between Spring Boot and Spring MVC?

Spring Boot is an opinionated framework that is built on top of the Spring Framework, which is a modular framework for building enterprise applications in Java. Spring Boot provides an out-of-the-box configuration for Spring applications and minimizes the effort required to set up and run a Spring application. Spring MVC, on the other hand, is a module within the Spring Framework that is used for building web applications. It provides a model-view-controller architecture for developing web applications and is used for building RESTful web services.

1. What is the purpose of the @Autowired annotation?

The @Autowired annotation is used in Spring Boot to inject dependencies automatically into a Spring managed bean. It is used to wire up the dependencies of a bean without the need for explicit configuration in XML or Java. By using this annotation, we can eliminate the need for manually configuring dependencies and focus on writing the business logic of our application.

1. What is the difference between the @Component, @Service, and @Repository annotations in Spring Boot?

The @Component, @Service, and @Repository annotations are used to denote different types of beans in a Spring Boot application. The main differences between them are as follows:

* @Component: This is a generic annotation that is used to denote any Spring-managed component. It is used as a base annotation for all other annotations.
* @Service: This annotation is used to denote a business service in a Spring Boot application. It is used to encapsulate the business logic of an application.
* @Repository: This annotation is used to denote a data access object in a Spring Boot application. It is used to provide data access services to the application.

1. What is the use of profiles in Spring Boot?

In Spring Boot, profiles are used to define different configurations for different environments. With profiles, we can define different configurations for development, testing, and production environments. By using profiles, we can ensure that our application works correctly in different environments and that we can easily switch between configurations.

1. How can you externalize the configuration properties in Spring Boot?

In Spring Boot, we can externalize the configuration properties by using the application.properties or application.yml file. These files are located in the src/main/resources directory of the Spring Boot application. By using these files, we can configure various properties such as database settings, server settings, and other application-specific properties. We can also use the @Value annotation to inject configuration properties into a Spring bean.

1. What are the different types of databases that are supported by Spring Boot?

Spring Boot supports a wide range of databases. Some of the databases that are supported by Spring Boot include:

* MySQL
* PostgreSQL
* Oracle
* MongoDB
* SQL Server
* H2 Database
* Apache Derby
* Cassandra
* Neo4j

These databases are supported through various Spring Boot modules and can be easily configured in a Spring Boot application.

When preparing for [Spring Boot interview questions](https://talentedge.com/articles/spring-boot-interview-questions/), as an experienced developer, it’s crucial to have a deep understanding of the Spring Boot framework and its related technologies. You should be able to explain Spring Boot Auto Configuration, Spring Boot Actuator, Spring Data JPA, and Spring Security, as well as how these concepts integrate with other Spring modules.

The interviewer may also ask about your experience in designing and implementing RESTful web services, working with databases, testing, and debugging Spring Boot applications. In addition, they may inquire about your knowledge of microservices architecture, Docker, and cloud-based deployment.

To impress the interviewer, you should be able to demonstrate your expertise in all of these areas and show that you have the skills necessary to design, implement, and test robust and scalable Spring Boot applications. Overall, thorough preparation and a strong understanding of Spring Boot and its related technologies are key to succeeding in a Spring Boot Interview.