Spring Boot Interview Questions

1) What is Spring Boot?

Spring Boot is a Spring module which provides RAD (Rapid Application Development) feature to Spring framework.

It is used to create stand alone spring based application that you can just run because it needs very little spring configuration.

For more information [click here.](https://www.javatpoint.com/spring-boot-introduction)

2) What are the advantages of Spring Boot?

* Create stand-alone Spring applications that can be started using java -jar.
* Embed Tomcat, Jetty or Undertow directly. You don't need to deploy WAR files.
* It provides opinionated 'starter' POMs to simplify your Maven configuration.
* It automatically configure Spring whenever possible.

For more information [click here.](https://www.javatpoint.com/spring-boot-introduction)

3) What are the features of Spring Boot?

* Web Development
* SpringApplication
* Application events and listeners
* Admin features

For more information [click here.](https://www.javatpoint.com/spring-boot-features)

4) How to create Spring Boot application using Maven?

There are multiple approaches to create Spring Boot project. We can use any of the following approach to create application.

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

For more information [click here.](https://www.javatpoint.com/spring-maven-project)

5) How to create Spring Boot project using Spring Initializer?

It is a web tool which is provided by Spring on official site. You can create Spring Boot project by providing project details.

For more information [click here.](https://www.javatpoint.com/spring-initializr)

6) How to create Spring Boot project using boot CLI?

It is a tool which you can download from the official site of Spring Framework. Here, we are explaining steps.

Download the CLI tool from official site and For more information [click here.](https://www.javatpoint.com/spring-boot-cli)

7) How to create simple Spring Boot application?

To create an application. We are using STS (Spring Tool Suite) IDE and it includes the various steps that are explaining in steps.

For more information [click here.](https://www.javatpoint.com/spring-boot-application)

8) What are the Spring Boot Annotations?

The @RestController is a stereotype annotation. It adds @Controller and @ResponseBody annotations to the class. We need to import org.springframework.web.bind.annotation package in our file, in order to implement it.

For more information [click here.](https://www.javatpoint.com/spring-boot-annotations)

9) What is Spring Boot dependency management?

Spring Boot manages dependencies and configuration automatically. You don't need to specify version for any of that dependencies.

Spring Boot upgrades all dependencies automatically when you upgrade Spring Boot.

For more information [click here.](https://www.javatpoint.com/spring-boot-dm)

10) What are the Spring Boot properties?

Spring Boot provides various properties which can be specified inside our project's **application.properties** file. These properties have default values and you can set that inside the properties file. Properties are used to set values like: server-port number, database connection configuration etc.

For more information [click here.](https://www.javatpoint.com/spring-boot-properties)

11) What are the Spring Boot Starters?

Starters are a set of convenient dependency descriptors which we can include in our application.

Spring Boot provides built-in starters which makes development easier and rapid. For example, if we want to get started using Spring and JPA for database access, just include the **spring-boot-starter-data-jpa** dependency in your project.

For more information [click here.](https://www.javatpoint.com/spring-boot-starters)

12) What is Spring Boot Actuator?

Spring Boot provides actuator to monitor and manage our application. Actuator is a tool which has HTTP endpoints. when application is pushed to production, you can choose to manage and monitor your application using HTTP endpoints.

For more information [click here.](https://www.javatpoint.com/spring-boot-actuator)

13) What is thymeleaf?

It is a server side Java template engine for web application. It's main goal is to bring elegant natural templates to your web application.

It can be integrate with Spring Framework and ideal for HTML5 Java web applications.

For more information [click here.](https://www.javatpoint.com/spring-boot-thymeleaf-view)

14) How to use thymeleaf?

In order to use Thymeleaf we must add it into our pom.xml file like:

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

For more information [click here.](https://www.javatpoint.com/spring-boot-thymeleaf-view)

15) How to connect Spring Boot to the database using JPA?

Spring Boot provides **spring-boot-starter-data-jpa** starter to connect Spring application with relational database efficiently. You can use it into project POM (Project Object Model) file.

For more information [click here.](https://www.javatpoint.com/spring-boot-jpa)

16) How to connect Spring Boot application to database using JDBC?

Spring Boot provides starter and libraries for connecting to our application with JDBC. Here, we are creating an application which connects with Mysql database. It includes the following steps to create and setup JDBC with Spring Boot.

For more information [click here.](https://www.javatpoint.com/spring-boot-jdbc)

17) What is @RestController annotation in Spring Boot?

The @RestController is a stereotype annotation. It adds @Controller and @ResponseBody annotations to the class. We need to import org.springframework.web.bind.annotation package in our file, in order to implement it.

For more information [click here.](https://www.javatpoint.com/spring-boot-annotations)

18) What is @RequestMapping annotation in Spring Boot?

The **@RequestMapping** annotation is used to provide routing information. It tells to the Spring that any HTTP request should map to the corresponding method. We need to import org.springframework.web.annotation package in our file.

For more information [click here.](https://www.javatpoint.com/spring-boot-annotations)

19) How to create Spring Boot application using Spring Starter Project Wizard?

There is one more way to create Spring Boot project in STS (Spring Tool Suite). Creating project by using IDE is always a convenient way. Follow the following steps in order to create a Spring Boot Application by using this wizard.

For more information [click here.](https://www.javatpoint.com/spring-starter-project-wizard)

20) Spring Vs Spring Boot?

Spring is a web application framework based on Java. It provides tools and libraries to create a complete cutomized web application.

Wheras Spring Boot is a spring module which is used to create spring application project that can just run.



***Free Mock Assessment***

**Want to switch to MAANG Companies?**

Get a Personalised Learning Plan for **FREE!**

**Attempt Now**

**75000+ Developers Registered!**

## Welcome to Interviewbit, help us create the best experience for you!

Currently, You are a:

Top of Form



Student



Working Professional

Bottom of Form

## Begin your success journey!

Sign Up using

Top of Form

**Full name \***



**Email \***

**Password \***



Continue

By creating an account, I acknowledge that I have read and agree to InterviewBit’s [**Terms**](https://www.interviewbit.com/pages/terms/)and [**Privacy Policy**](https://www.interviewbit.com/pages/privacy/).

Already have an account? Log in

Bottom of Form

# Spring Boot Interview QuestionsLast updated on Aug 26, 2022

[**Download PDF**](javascript:void(0))

Share with:



Copied

* [**Spring Boot Interview Questions For Freshers**](https://www.interviewbit.com/spring-boot-interview-questions/#spring-boot-interview-questions-for-freshers)
  + [1.What are the advantages of using Spring Boot?](https://www.interviewbit.com/spring-boot-interview-questions/#pros-of-using-spring-boot)
  + [2.What are the Spring Boot key components?](https://www.interviewbit.com/spring-boot-interview-questions/#spring-boot-components)
  + [3.Why Spring Boot over Spring?](https://www.interviewbit.com/spring-boot-interview-questions/#why-spring-boot-over-spring)
  + [4.What is the starter dependency of the Spring boot module?](https://www.interviewbit.com/spring-boot-interview-questions/#starter-dependency-of-spring-boot)
  + [5.How does Spring Boot works?](https://www.interviewbit.com/spring-boot-interview-questions/#how-spring-boot-works?)
  + [6.What does the @SpringBootApplication annotation do internally?](https://www.interviewbit.com/spring-boot-interview-questions/#what-does-@springbootannotation-do)
  + [7.What is the purpose of using @ComponentScan in the class files?](https://www.interviewbit.com/spring-boot-interview-questions/#purpose-of-@componentscan)
  + [8.How does a spring boot application get started?](https://www.interviewbit.com/spring-boot-interview-questions/#how-spring-boot-get-started)
  + [9.What are starter dependencies?](https://www.interviewbit.com/spring-boot-interview-questions/#starter-dependencies)
  + [10.What is Spring Initializer?](https://www.interviewbit.com/spring-boot-interview-questions/#spring-initializer)
  + [11.What is Spring Boot CLI and what are its benefits?](https://www.interviewbit.com/spring-boot-interview-questions/#spring-boot-cli)
  + [12.What are the most common Spring Boot CLI commands?](https://www.interviewbit.com/spring-boot-interview-questions/#common-spring-boot-cli-commands)
* [**Advanced Spring Boot Questions**](https://www.interviewbit.com/spring-boot-interview-questions/#advanced-spring-boot-questions)
  + [13.What Are the Basic Annotations that Spring Boot Offers?](https://www.interviewbit.com/spring-boot-interview-questions/#basic-annotations-in-spring-boot)
  + [14.What is Spring Boot dependency management?](https://www.interviewbit.com/spring-boot-interview-questions/#what-is-spring-dependency-management)
  + [15.Can we create a non-web application in Spring Boot?](https://www.interviewbit.com/spring-boot-interview-questions/#non-web-application-in-spring-boot)
  + [16.Is it possible to change the port of the embedded Tomcat server in Spring Boot?](https://www.interviewbit.com/spring-boot-interview-questions/#changing-port-of-embedded-tomcat-server)
  + [17.What is the default port of tomcat in spring boot?](https://www.interviewbit.com/spring-boot-interview-questions/#default-port-of-tomcat)
  + [18.Can we override or replace the Embedded tomcat server in Spring Boot?](https://www.interviewbit.com/spring-boot-interview-questions/#override-or-replace-embedded-tomcat)
  + [19.Can we disable the default web server in the Spring boot application?](https://www.interviewbit.com/spring-boot-interview-questions/#disabling-dafault-web-server)
  + [20.How to disable a specific auto-configuration class?](https://www.interviewbit.com/spring-boot-interview-questions/#disable-specific-auto-config-class)
  + [21.Explain @RestController annotation in Sprint boot?](https://www.interviewbit.com/spring-boot-interview-questions/#@rest-controller)
  + [22.What is the difference between @RestController and @Controller in Spring Boot?](https://www.interviewbit.com/spring-boot-interview-questions/#difference-between-@restcontroller-and-@controller)
  + [23.Describe the flow of HTTPS requests through the Spring Boot application?](https://www.interviewbit.com/spring-boot-interview-questions/#follow-of-https)
  + [24.What is the difference between RequestMapping and GetMapping?](https://www.interviewbit.com/spring-boot-interview-questions/#requestmapping-and-getmapping)
  + [25.What is the use of Profiles in spring boot?](https://www.interviewbit.com/spring-boot-interview-questions/#use-of-profiles-in-spring-boot)
  + [26.What is Spring Actuator? What are its advantages?](https://www.interviewbit.com/spring-boot-interview-questions/#spring-actuator)
  + [27.How to enable Actuator in Spring boot application?](https://www.interviewbit.com/spring-boot-interview-questions/#enable-actuator-in-spring-boot)
  + [28.What are the actuator-provided endpoints used for monitoring the Spring boot application?](https://www.interviewbit.com/spring-boot-interview-questions/#actuator-provided-endpoints)
  + [29.How to get the list of all the beans in your Spring boot application?](https://www.interviewbit.com/spring-boot-interview-questions/#getting-list-of-all-beans)
  + [30.How to check the environment properties in your Spring boot application?](https://www.interviewbit.com/spring-boot-interview-questions/#checking-environment-properties)
  + [31.How to enable debugging log in the spring boot application?](https://www.interviewbit.com/spring-boot-interview-questions/#enabling-debugging-log)
  + [32.Where do we define properties in the Spring Boot application?](https://www.interviewbit.com/spring-boot-interview-questions/#define-properties)
  + [33.What is dependency Injection?](https://www.interviewbit.com/spring-boot-interview-questions/#what-is-dependency-injection)
  + [34.What is an IOC container?](https://www.interviewbit.com/spring-boot-interview-questions/#what-is-ioc-container)
* [**Spring Boot MCQs**](https://www.interviewbit.com/spring-boot-interview-questions/#spring-boot-mcqs)

[**Spring boot**](https://www.interviewbit.com/blog/spring-boot-architecture/) is the hottest topic of discussion in interviews if it comes to Java Application development. Because of its fast, low configuration, inbuild server, and monitoring features, it helps to build a stand-alone java application from scratch with very robust and maintainable.

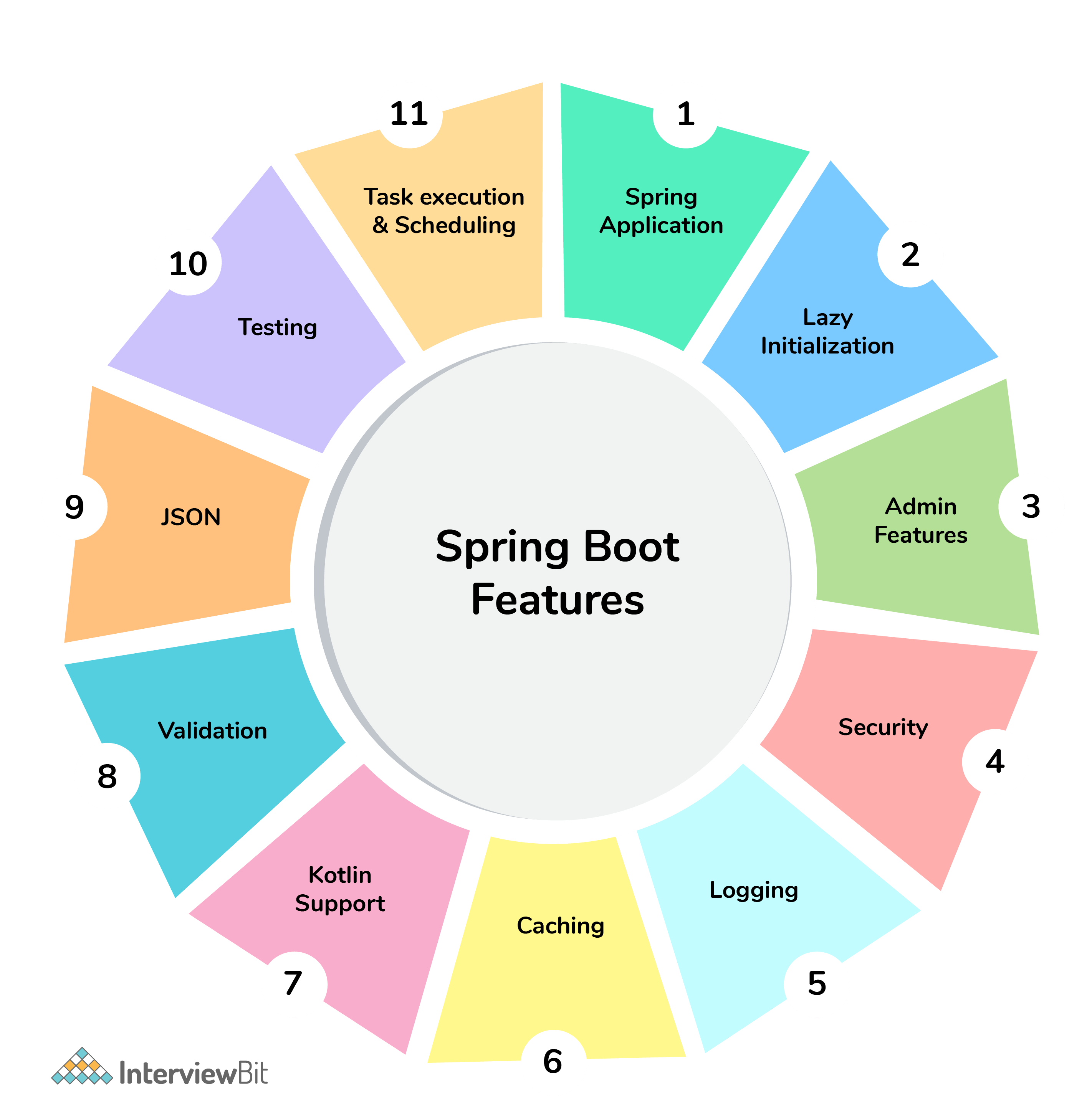
The article will walk you through the Spring Boot interview questions for basic to advanced level.

## 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

**Crack your next tech interview with confidence!**

Take a free mock interview, get instant⚡️ feedback and recommendation💡

**Attempt Now**

## 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:

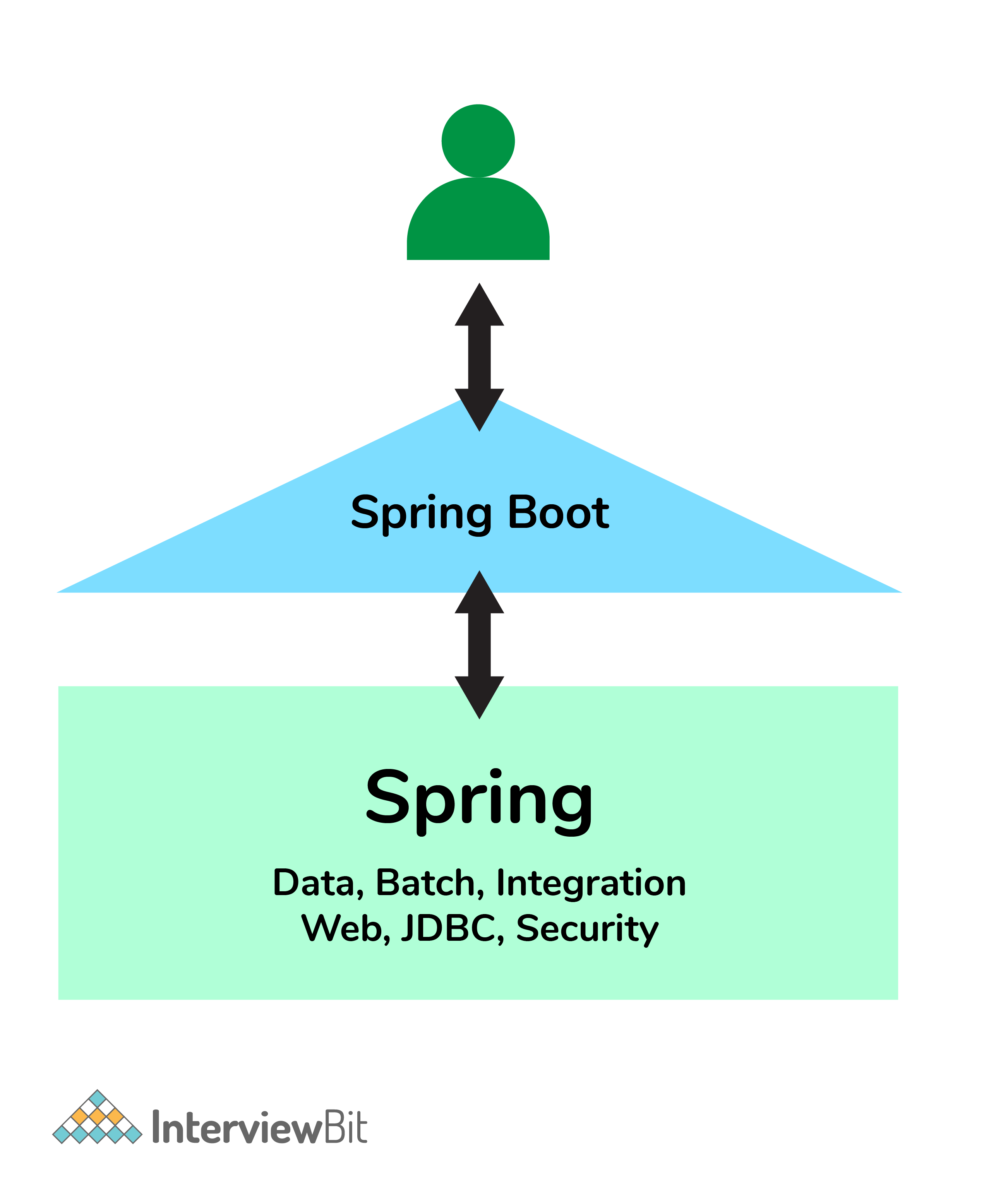
* Spring Boot auto-configuration.
* Spring Boot CLI.
* Spring Boot starter POMs.
* Spring Boot Actuators.

### 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:

Spring vs Spring Boot

**You can download a PDF version of Spring Boot Interview Questions.**

[**Download PDF**](javascript:void(0))

### 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?

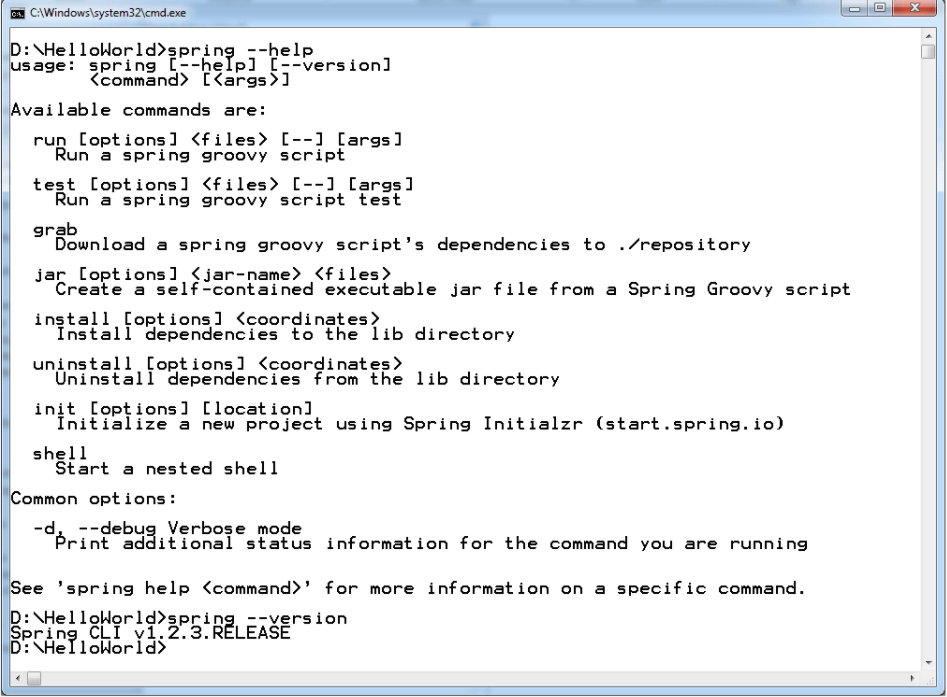
Spring Boot CLI is a command-line interface that allows you to create a spring-based java application using Groovy.

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 Sprint boot?

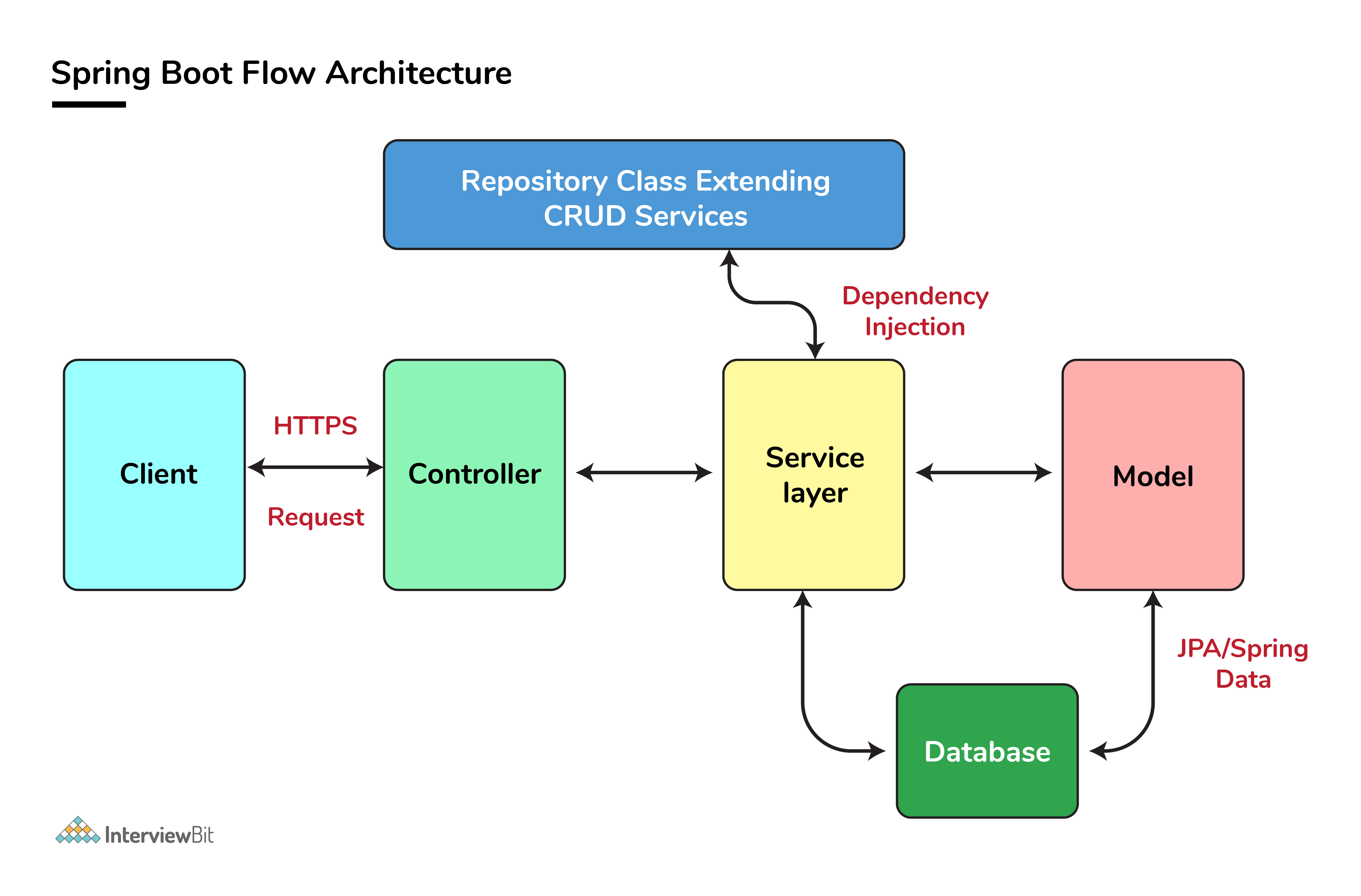
It is a combination of @Controller and @ResponseBody, used for creating a restful controller. It converts the response to JSON or XML. It ensures that data returned by each method will be written straight into the response body instead of returning a template.

### 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.

**Important Resources**

* [Spring vs Spring Boot](https://www.interviewbit.com/blog/spring-vs-spring-boot/)
* [Difference Between Spring MVC and Spring Boot](https://www.interviewbit.com/blog/difference-between-spring-mvc-and-spring-boot/)
* [Spring Interview Questions](https://www.interviewbit.com/spring-interview-questions/)
* [Spring Security Interview Questions](https://www.interviewbit.com/spring-security-interview-questions/)

## **Spring Boot MCQs**

1.

Spring is used for?

Java Framework

Web Development Framework

MVC Framework

All of the Above

2.

Default HTML template engine in Spring Boot?

JSP

HTML

Thymeleaf

None of the Above

3.

Annotation used for handling GET requests?

@PostMapping

@GetMapping

@PutMapping

None of the Above

4.

Annotation used for Rest Controller?

@SpringBootApplication

@RestController

@Controller

All of the Above

5.

Which annotation is not Spring Boot Annotation?

@Controller

@RestController

@Service

@Metadata

6.

Minimum Java version needed for Spring Boot?

Java 10

Java 8

Java 7

Java 9

7.

Which of the following is used by Maven?

Pom.xml

Config.xml

META-INF

None of the Above

8.

Is Dependency needed to create a Spring Boot web application?

Spring-boot-starter-web

Spring-boot-starter-test

Spring-boot-html

None of the Above

9.

Starting points of Spring Boot Application?

@Controller

@SpringBootApplication

@Service

None of the Above

10.

Database object must be annotated with?

@Table

@Entity

@Column

@Repository

### ****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:

[[](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=curriculum)](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)

### [Spring Framework Certification Course](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)

* *[Instructor-led Sessions](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)*
* *[Real-life Case Studies](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)*
* *[Assignments](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)*
* *[Lifetime Access](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)*

[Explore Curriculum](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)

* 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:

### Programming & Frameworks Training

### [FULL STACK WEB DEVELOPMENT INTERNSHIP PROGRAM](https://www.edureka.co/full-stack-web-development?utm_source=blogbanner&utm_campaign=trending_course" \o "Full Stack Web Development Internship Program" \t "_blank)

### [Full Stack Web Development Internship Program](https://www.edureka.co/full-stack-web-development?utm_source=blogbanner&utm_campaign=trending_course" \o "Full Stack Web Development Internship Program" \t "_blank)

*[Reviews](https://www.edureka.co/full-stack-web-development?utm_source=blogbanner&utm_campaign=trending_course" \o "Full Stack Web Development Internship Program" \t "_blank)*

**[5](https://www.edureka.co/full-stack-web-development?utm_source=blogbanner&utm_campaign=trending_course" \o "Full Stack Web Development Internship Program" \t "_blank)**[(2069)](https://www.edureka.co/full-stack-web-development?utm_source=blogbanner&utm_campaign=trending_course" \o "Full Stack Web Development Internship Program" \t "_blank)

### [JAVA CERTIFICATION TRAINING COURSE](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Java Certification Training Course" \t "_blank)

### [Java Certification Training Course](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Java Certification Training Course" \t "_blank)

*[Reviews](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Java Certification Training Course" \t "_blank)*

**[4](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Java Certification Training Course" \t "_blank)**[(62089)](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Java Certification Training Course" \t "_blank)

### [PYTHON SCRIPTING CERTIFICATION TRAINING](https://www.edureka.co/python-scripting?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Scripting Certification Training" \t "_blank)

### [Python Scripting Certification Training](https://www.edureka.co/python-scripting?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Scripting Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/python-scripting?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Scripting Certification Training" \t "_blank)*

**[5](https://www.edureka.co/python-scripting?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Scripting Certification Training" \t "_blank)**[(12287)](https://www.edureka.co/python-scripting?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Scripting Certification Training" \t "_blank)

### [C PROGRAMMING CERTIFICATION COURSE](https://www.edureka.co/c-programming-course?utm_source=blogbanner&utm_campaign=trending_course" \o "C Programming Certification Course" \t "_blank)

### [C Programming Certification Course](https://www.edureka.co/c-programming-course?utm_source=blogbanner&utm_campaign=trending_course" \o "C Programming Certification Course" \t "_blank)

*[Reviews](https://www.edureka.co/c-programming-course?utm_source=blogbanner&utm_campaign=trending_course" \o "C Programming Certification Course" \t "_blank)*

**[5](https://www.edureka.co/c-programming-course?utm_source=blogbanner&utm_campaign=trending_course" \o "C Programming Certification Course" \t "_blank)**[(1325)](https://www.edureka.co/c-programming-course?utm_source=blogbanner&utm_campaign=trending_course" \o "C Programming Certification Course" \t "_blank)

### [C++ PROGRAMMING COURSE](https://www.edureka.co/cpp-programming-course?utm_source=blogbanner&utm_campaign=trending_course" \o "C++ Programming Course" \t "_blank)

### [C++ Programming Course](https://www.edureka.co/cpp-programming-course?utm_source=blogbanner&utm_campaign=trending_course" \o "C++ Programming Course" \t "_blank)

*[Reviews](https://www.edureka.co/cpp-programming-course?utm_source=blogbanner&utm_campaign=trending_course" \o "C++ Programming Course" \t "_blank)*

**[5](https://www.edureka.co/cpp-programming-course?utm_source=blogbanner&utm_campaign=trending_course" \o "C++ Programming Course" \t "_blank)**[(877)](https://www.edureka.co/cpp-programming-course?utm_source=blogbanner&utm_campaign=trending_course" \o "C++ Programming Course" \t "_blank)

### [PYTHON DJANGO CERTIFICATION TRAINING COURSE](https://www.edureka.co/python-django?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Django Certification Training Course" \t "_blank)

### [Python Django Certification Training Course](https://www.edureka.co/python-django?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Django Certification Training Course" \t "_blank)

*[Reviews](https://www.edureka.co/python-django?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Django Certification Training Course" \t "_blank)*

**[5](https://www.edureka.co/python-django?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Django Certification Training Course" \t "_blank)**[(7165)](https://www.edureka.co/python-django?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Django Certification Training Course" \t "_blank)

### [NODE.JS CERTIFICATION TRAINING COURSE](https://www.edureka.co/nodejs-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Node.js Certification Training Course" \t "_blank)

### [Node.js Certification Training Course](https://www.edureka.co/nodejs-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Node.js Certification Training Course" \t "_blank)

*[Reviews](https://www.edureka.co/nodejs-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Node.js Certification Training Course" \t "_blank)*

**[5](https://www.edureka.co/nodejs-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Node.js Certification Training Course" \t "_blank)**[(7578)](https://www.edureka.co/nodejs-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Node.js Certification Training Course" \t "_blank)

### [ADVANCED JAVA CERTIFICATION TRAINING](https://www.edureka.co/advanced-java-sp?utm_source=blogbanner&utm_campaign=trending_course" \o "Advanced Java Certification Training" \t "_blank)

### [Advanced Java Certification Training](https://www.edureka.co/advanced-java-sp?utm_source=blogbanner&utm_campaign=trending_course" \o "Advanced Java Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/advanced-java-sp?utm_source=blogbanner&utm_campaign=trending_course" \o "Advanced Java Certification Training" \t "_blank)*

**[5](https://www.edureka.co/advanced-java-sp?utm_source=blogbanner&utm_campaign=trending_course" \o "Advanced Java Certification Training" \t "_blank)**[(4761)](https://www.edureka.co/advanced-java-sp?utm_source=blogbanner&utm_campaign=trending_course" \o "Advanced Java Certification Training" \t "_blank)

### [SPRING FRAMEWORK CERTIFICATION COURSE](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Spring Framework Certification Course" \t "_blank)

### [Spring Framework Certification Course](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Spring Framework Certification Course" \t "_blank)

*[Reviews](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Spring Framework Certification Course" \t "_blank)*

**[5](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Spring Framework Certification Course" \t "_blank)**[(9833)](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Spring Framework Certification Course" \t "_blank)

Next

* 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**

[[](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=batches)](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=batches" \t "_blank)

### [Spring Framework Certification Course](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=batches" \t "_blank)

[Weekday / Weekend BatchesSee Batch Details](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=batches" \t "_blank)

* 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);      }  } |

**Step 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

## **Top 10 Technologies to Learn in 2022 | Edureka**

This Edureka video on 𝐓𝐨𝐩 𝟏𝟎 𝐓𝐞𝐜𝐡𝐧𝐨𝐥𝐨𝐠𝐢𝐞𝐬 𝐭𝐨 𝐋𝐞𝐚𝐫𝐧 𝐢𝐧 𝟐𝟎𝟐𝟐 will introduce you to all the popular and trending technologies in the market which you should focus on in 2022. These are the trending technologies that you need to learn in order to have a successful career in the year 2022.



### 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": "<a href="http://localhost:8080/sample/1">http://localhost:8080/sample/1</a>"  },  "sample": {  "href": "<a href="http://localhost:8080/sample/1">http://localhost:8080/sample/1</a>"  }  } |

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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <dependency>      <groupId>org.springframework.boot</groupId>      <artifactId>spring-boot-starter-web</artifactId>      <exclusions>          <exclusion>              <groupId>org.springframework.boot</groupId>              <artifactId>spring-boot-starter-tomcat</artifactId>          </exclusion>      </exclusions>  </dependency>  <dependency>      <groupId>org.springframework.boot</groupId>      <artifactId>spring-boot-starter-jetty</artifactId>  </dependency> |

### ****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](https://www.mygreatlearning.com/academy/learn-for-free/courses/dockerize-spring-boot-application/?gl_blog_id=25325) 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](https://www.mygreatlearning.com/blog/java-tutorial-for-beginners/?gl_blog_id=25325) (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](https://www.mygreatlearning.com/jpa/free-courses/?gl_blog_id=25325) is basically **Java Persistence API**. It’s a specification that lets you do ORM when you are connecting to a [relational database](https://www.mygreatlearning.com/blog/what-is-rdbms/?gl_blog_id=25325) 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](https://www.mygreatlearning.com/blog/sql-tutorial-for-beginners/?gl_blog_id=25325) and then you get the results and convert them into Object instances.

ORM lets you map your entity classes in your [SQL](https://www.mygreatlearning.com/academy/learn-for-free/courses/introduction-to-sql/?gl_blog_id=25325) 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](https://www.mygreatlearning.com/academy/learn-for-free/courses/my-sql-basics/?gl_blog_id=25325) 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](https://www.mygreatlearning.com/json-format/free-courses/?gl_blog_id=25325) 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](https://www.mygreatlearning.com/xml/free-courses/?gl_blog_id=25325) 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](https://www.mygreatlearning.com/blog/mysql-tutorial/?gl_blog_id=25325).  
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](https://www.mygreatlearning.com/blog/jdbc-tutorial/?gl_blog_id=25325), 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](https://www.mygreatlearning.com/academy/learn-for-free/courses/intro-to-bootstrap/?gl_blog_id=25325) 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](https://www.mygreatlearning.com/spring/tutorials/spring-dependency-injection/?gl_blog_id=25325) 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](https://www.mygreatlearning.com/academy/learn-for-free/courses/mongodb-tutorial/?gl_blog_id=25325) 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](https://www.mygreatlearning.com/academy/learn-for-free/courses/html-tutorial/?gl_blog_id=25325), [XML](https://www.mygreatlearning.com/blog/xml-tutorial/?gl_blog_id=25325), [JavaScript](https://www.mygreatlearning.com/academy/learn-for-free/courses/intro-to-javascript/?gl_blog_id=25325) , [CSS](https://www.mygreatlearning.com/academy/learn-for-free/courses/css-tutorial/?gl_blog_id=25325), 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](https://www.mygreatlearning.com/blog/java-super-keyword-and-wrapper-class/?gl_blog_id=25325).

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:\demo\target>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:\demo\target>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.

1. **@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:-**

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

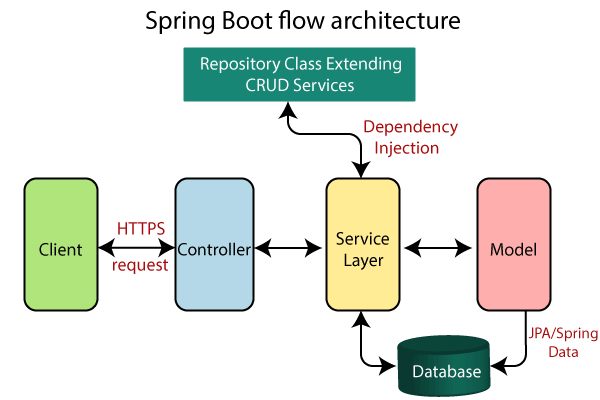
@Restcontroller is annotated within @controller and @Responsebody. This annotation is applied to mark the respective class as a request handler in your application.

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?****

|  |  |
| --- | --- |
| **@controller** | **@RestController** |
| Controller is a common annotation that is used to mark a class as a spring MVC controller. | Rest controller is a Springspecial controller used in Restful web services and the wrapped within the @controller and @Responsebody |

### ****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](https://www.mygreatlearning.com/blog/everything-you-need-to-know-about-microservice-architecture/?gl_blog_id=25325) 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 [J](https://www.mygreatlearning.com/blog/java-virtual-machine/)[VM](https://www.mygreatlearning.com/blog/java-virtual-machine/?gl_blog_id=25325) 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.properties

### ****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.WebMvcTestContextBootsrapper.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>

1. 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";

}

}

1. 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} + '!'"/>

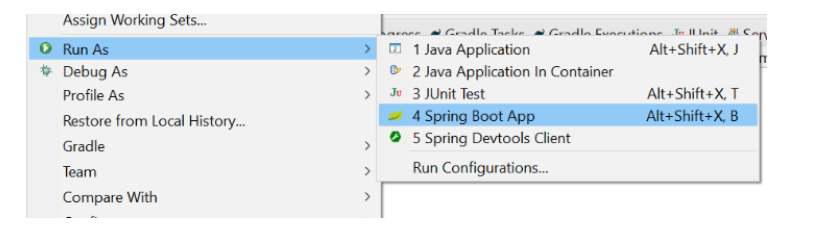
<h4> Welcome to Thymeleaf Demo in Spring Boot</h4>

</body>

</html>

1. 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

1. By changing in application.properties file

server.port=8081

1. 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.

Also Read: [Top 25 Common Interview Questions](https://www.mygreatlearning.com/blog/common-interview-questions/?gl_blog_id=25325)

## **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](https://www.mygreatlearning.com/academy/career-paths/java-developer/?gl_blog_id=25325) 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](https://www.mygreatlearning.com/blog/everything-you-need-to-know-about-microservice-architecture/?gl_blog_id=25325) 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.

**Q1. What Is Spring Boot and What Are Its Main Features?**

Spring Boot is essentially a framework for rapid application development built on top of the Spring Framework. With its auto-configuration and embedded application server support, combined with the extensive documentation and community support it enjoys, Spring Boot is one of the most popular technologies in the Java ecosystem as of date.

Here are a few salient features:

* [**Starters**](https://www.baeldung.com/spring-boot-starters) – a set of dependency descriptors to include relevant dependencies at a go
* [**Auto-configuration**](https://www.baeldung.com/spring-boot-annotations#enable-autoconfiguration) – a way to automatically configure an application based on the dependencies present on the classpath
* [**Actuator**](https://www.baeldung.com/spring-boot-actuators) – to get production-ready features such as monitoring
* [**Security**](https://www.baeldung.com/security-spring)
* [**Logging**](https://www.baeldung.com/spring-boot-logging)

**Q2. What Are the Differences Between Spring and Spring Boot?**

The Spring Framework provides multiple features that make the development of web applications easier. These features include dependency injection, data binding, aspect-oriented programming, data access and many more.

Over the years, Spring has been growing more and more complex, and the amount of configuration such application requires can be intimidating. This is where Spring Boot comes in handy — it makes configuring a Spring application a breeze.

Essentially, while Spring is unopinionated, **Spring Boot takes an opinionated view of the platform and libraries, letting us get started quickly.**

Here are two of the most important benefits Spring Boot brings in:

* Auto-configure applications based on the artifacts it finds on the classpath
* Provide non-functional features common to applications in production, such as security or health checks

Please check out our other tutorial for a [detailed comparison between vanilla Spring and Spring Boot](https://www.baeldung.com/spring-vs-spring-boot).

**Q3. How Can We Set Up a Spring Boot Application With Maven?**

We can include Spring Boot in a Maven project just like we would any other library. However, the best way is to inherit from the *spring-boot-starter-parent* project and declare dependencies to [Spring Boot starters](https://www.baeldung.com/spring-boot-starters). Doing this lets our project reuse the default settings of Spring Boot.

Inheriting the *spring-boot-starter-parent* project is straightforward — we only need to specify a *parent* element in *pom.xml*:

<**parent**>

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

<**artifactId**>spring-boot-starter-parent</**artifactId**>

<**version**>2.4.0.RELEASE</**version**>

</**parent**>

We can find the latest version of *spring-boot-starter-parent* on [Maven Central](https://search.maven.org/search?q=g:org.springframework.boot%20AND%20a:spring-boot-starter-parent&core=gav).

**Using the starter parent project is convenient but not always feasible.** For instance, if our company requires all projects to inherit from a standard POM, we can still benefit from Spring Boot's dependency management using a [custom parent](https://www.baeldung.com/spring-boot-dependency-management-custom-parent).

**Q4. What Is Spring Initializr?**

Spring Initializr is a convenient way to create a Spring Boot project.

We can go to the [Spring Initializr](https://start.spring.io/) site, choose a dependency management tool (either Maven or Gradle), a language (Java, Kotlin or Groovy), a packaging scheme (Jar or War), version and dependencies, and download the project.

This **creates a skeleton project for us** and saves setup time so that we can concentrate on adding business logic.

Even when we use our IDE's (such as STS or Eclipse with STS plugin) new project wizard to create a Spring Boot project, it uses Spring Initializr under the hood.

**Q5. What Spring Boot Starters Are Available Out There?**

Each starter plays a role as a one-stop shop for all the Spring technologies we need. Other required dependencies are then transitively pulled in and managed in a consistent way.

All starters are under the *org.springframework.boot* group and their names start with *spring-boot-starter-*. **This naming pattern makes it easy to find starters, especially when working with IDEs that support searching dependencies by name.**

At the time of this writing, there are more than 50 starters at our disposal. Here, we'll list the most common:

* *spring-boot-starter*: core starter, including auto-configuration support, logging and YAML
* *spring-boot-starter-aop*: for aspect-oriented programming with Spring AOP and AspectJ
* *spring-boot-starter-data-jpa:* for using Spring Data JPA with Hibernate
* *spring-boot-starter-security*: for using Spring Security
* *spring-boot-starter-test*: for testing Spring Boot applications
* *spring-boot-starter-web*: for building web, including RESTful, applications using Spring MVC

For a complete list of starters, please see [this repository](https://github.com/spring-projects/spring-boot/tree/master/spring-boot-project/spring-boot-starters).

To find more information about Spring Boot starters, take a look at [Intro to Spring Boot Starters](https://www.baeldung.com/spring-boot-starters).

**Q6. How to Disable a Specific Auto-Configuration?**

If we want to disable a specific auto-configuration, we can indicate it using the *exclude* attribute of the *@EnableAutoConfiguration* annotation.

For instance, this code snippet neutralizes *DataSourceAutoConfiguration*:

// other annotations

@EnableAutoConfiguration(exclude = DataSourceAutoConfiguration.class)

**public** **class** **MyConfiguration** { }

If we enabled auto-configuration with the *@SpringBootApplication* annotation — which has *@EnableAutoConfiguration* as a meta-annotation — we could disable auto-configuration with an attribute of the same name:

// other annotations

@SpringBootApplication(exclude = DataSourceAutoConfiguration.class)

**public** **class** **MyConfiguration** { }

We can also disable an auto-configuration with the *spring.autoconfigure.exclude* environment property. This setting in the *application.properties* file does the same thing as before:

spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration

**Q7. How to Register a Custom Auto-Configuration?**

To register an auto-configuration class, we must have its fully qualified name listed under the *EnableAutoConfiguration* key in the *META-INF/spring.factories* file:

org.springframework.boot.autoconfigure.EnableAutoConfiguration=com.baeldung.autoconfigure.CustomAutoConfiguration

If we build a project with Maven, that file should be placed in the *resources/META-INF* directory, which will end up in the mentioned location during the *package* phase.

**Q8. How to Tell an Auto-Configuration to Back Away When a Bean Exists?**

To instruct an auto-configuration class to back off when a bean already exists, we can use the *@ConditionalOnMissingBean* annotation.

The most noticeable attributes of this annotation are:

* *value* – the types of beans to be checked
* *name* – the names of beans to be checked

When placed on a method adorned with *@Bean*, the target type defaults to the method's return type:

@Configuration

**public** **class** **CustomConfiguration** {

@Bean

@ConditionalOnMissingBean

**public** CustomService **service**() { ... }

}

**Q9. How to Deploy Spring Boot Web Applications as Jar and War Files?**

Traditionally, we package a web application as a WAR file and then deploy it into an external server. Doing this allows us to arrange multiple applications on the same server. When CPU and memory were scarce, this was a great way to save resources.

But things have changed. Computer hardware is fairly cheap now, and the attention has turned to server configuration. A small mistake in configuring the server during deployment may lead to catastrophic consequences.

**Spring tackles this problem by providing a plugin, namely *spring-boot-maven-plugin*, to package a web application as an executable JAR.**

To include this plugin, just add a *plugin* element to *pom.xml*:

<**plugin**>

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

<**artifactId**>spring-boot-maven-plugin</**artifactId**>

</**plugin**>

With this plugin in place, we'll get a fat JAR after executing the *package* phase. This JAR contains all the necessary dependencies, including an embedded server. So, we no longer need to worry about configuring an external server.

We can then run the application just like we would an ordinary executable JAR.

Notice that the *packaging* element in the *pom.xml* file must be set to *jar* to build a JAR file:

<**packaging**>jar</**packaging**>

If we don't include this element, it also defaults to *jar*.

To build a WAR file, we change the *packaging* element to *war*:

<**packaging**>war</**packaging**>

and leave the container dependency off the packaged file:

<**dependency**>

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

<**artifactId**>spring-boot-starter-tomcat</**artifactId**>

<**scope**>provided</**scope**>

</**dependency**>

After executing the Maven *package* phase, we'll have a deployable WAR file.

**Q10. How to Use Spring Boot for Command-Line Applications?**

Just like any other Java program, a Spring Boot command-line 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

}

}

The *SpringApplication* class then fires up a Spring container and auto-configures beans.

Notice we must pass a configuration class to the *run* method to work as the primary configuration source. By convention, this argument is the entry class itself.

After calling the *run* method, we can execute other statements as in a regular program.

**Q11. What Are Possible Sources of External Configuration?**

Spring Boot provides support for external configuration, allowing us to run the same application in various environments. **We can use properties files, YAML files, environment variables, system properties and command-line option arguments to specify configuration properties.**

We can then gain access to those properties using the *@Value* annotation, a bound object via the [*@ConfigurationProperties* annotation](https://www.baeldung.com/configuration-properties-in-spring-boot), or the *Environment* abstraction.

**Q12. What Does It Mean That Spring Boot Supports Relaxed Binding?**

Relaxed binding in Spring Boot is applicable to [the type-safe binding of configuration properties](https://www.baeldung.com/configuration-properties-in-spring-boot).

With relaxed binding, **the key of a property doesn't need to be an exact match of a property name.** Such an environment property can be written in camelCase, kebab-case, snake\_case, or in uppercase with words separated by underscores.

For example, if a property in a bean class with the *@ConfigurationProperties* annotation is named *myProp*, it can be bound to any of these environment properties: *myProp*, *my-prop*, *my\_prop*, or *MY\_PROP*.

**Q13. What Is Spring Boot DevTools Used For?**

Spring Boot Developer Tools, or DevTools, is a set of tools making the development process easier.

To include these development-time features, we just need to add a dependency to the *pom.xml* file:

<**dependency**>

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

<**artifactId**>spring-boot-devtools</**artifactId**>

</**dependency**>

The *spring-boot-devtools* module is automatically disabled if the application runs in production. The repackaging of archives also excludes this module by default. So, it won't bring any overhead to our final product.

By default, DevTools applies properties suitable to a development environment. These properties disable template caching, enable debug logging for the web group, and so on. As a result, we have this sensible development-time configuration without setting any properties.

**Applications using DevTools restart whenever a file on the classpath changes.** This is a very helpful feature in development, as it gives quick feedback for modifications.

By default, static resources, including view templates, don't set off a restart. Instead, a resource change triggers a browser refresh. Notice this can only happen if the LiveReload extension is installed in the browser to interact with the embedded LiveReload server that DevTools contains.

For further information on this topic, please see [Overview of Spring Boot DevTools](https://www.baeldung.com/spring-boot-devtools).

**Q14. How to Write Integration Tests?**

When running integration tests for a Spring application, we must have an *ApplicationContext*.

To make our life easier, Spring Boot provides a special annotation for testing — *@SpringBootTest*. This annotation creates an *ApplicationContext* from configuration classes indicated by its *classes* attribute.

**In case the *classes* attribute isn't set, Spring Boot searches for the primary configuration class.** The search starts from the package containing the test until it finds a class annotated with *@SpringBootApplication* or *@SpringBootConfiguration*.

For detailed instructions, check out our tutorial on [testing in Spring Boot](https://www.baeldung.com/spring-boot-testing).

**Q15. What Is Spring Boot Actuator Used For?**

Essentially, Actuator brings Spring Boot applications to life by enabling production-ready features. **These features allow us to monitor and manage applications when they're running in production.**

Integrating Spring Boot Actuator into a project is very simple. All we need to do is include the *spring-boot-starter-actuator* starter in the *pom.xml* file:

<**dependency**>

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

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

</**dependency**>

Spring Boot Actuator can expose operational information using either HTTP or JMX endpoints. But most applications go for HTTP, where the identity of an endpoint and the */actuator* prefix form a URL path.

Here are some of the most common built-in endpoints Actuator provides:

* *env* exposes environment properties
* *health* shows application health information
* *httptrace* displays HTTP trace information
* *info* displays arbitrary application information
* *metrics* shows metrics information
* *loggers* shows and modifies the configuration of loggers in the application
* *mappings* displays a list of all *@RequestMapping* paths

Please refer to our [Spring Boot Actuator tutorial](https://www.baeldung.com/spring-boot-actuators) for a detailed rundown.

**Q16. Which Is Better to Configure a Spring Boot Project — Properties or YAML?**

YAML offers many advantages over properties files:

* More clarity and better readability
* Perfect for hierarchical configuration data, which is also represented in a better, more readable format
* Support for maps, lists and scalar types
* Can include several [**profiles**](https://www.baeldung.com/spring-profiles) in the same file (since Spring Boot 2.4.0, this is possible for properties files too)

However, writing it can be a little difficult and error-prone due to its indentation rules.

For details and working samples, please refer to our [Spring YAML vs Properties](https://www.baeldung.com/spring-yaml-vs-properties) tutorial.

**Q17. What Basic Annotations Does Spring Boot Offer?**

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

[Spring Boot Annotations](https://www.baeldung.com/spring-boot-annotations) offers more insight into the subject.

**Q18. How to Change the Default Port in Spring Boot?**

We can [change the default port of a server embedded in Spring Boot](https://www.baeldung.com/spring-boot-change-port) using one of these ways:

* Using a properties file – We can define this in an *application.properties* (or *application.yml*) file using the property *server.port*.
* Programmatically – In our main *@SpringBootApplication* class, we can set the *server.port* on the *SpringApplication* instance.
* Using the command line – When running the application as a jar file, we can set the server.port as a java command argument:
* java -jar -Dserver.port=8081 myspringproject.jar

**Q19. Which Embedded Servers Does Spring Boot Support, and How to Change the Default?**

As of date, **Spring MVC supports Tomcat, Jetty and Undertow.** Tomcat is the default application server supported by Spring Boot's *web* starter.

**Spring WebFlux supports Reactor Netty, Tomcat, Jetty and Undertow** with Reactor Netty as default.

In Spring MVC, to change the default, let's say to Jetty, we need to exclude Tomcat and include Jetty in the dependencies:

<**dependency**>

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

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

<**exclusions**>

<**exclusion**>

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

<**artifactId**>spring-boot-starter-tomcat</**artifactId**>

</**exclusion**>

</**exclusions**>

</**dependency**>

<**dependency**>

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

<**artifactId**>spring-boot-starter-jetty</**artifactId**>

</**dependency**>

Similarly, to change the default in WebFlux to UnderTow, we need to exclude Reactor Netty and include UnderTow in the dependencies.

[Comparing Embedded Servlet Containers in Spring Boot](https://www.baeldung.com/spring-boot-servlet-containers) has more details on the different embedded servers we can use with Spring MVC.

**Q20. Why Do We Need Spring Profiles?**

When developing applications for the enterprise, we typically deal with multiple environments such as Dev, QA and Prod. The configuration properties for these environments are different.

For example, we might be using an embedded H2 database for Dev, but Prod could have the proprietary Oracle or DB2. Even if the DBMS is the same across environments, the URLs would definitely be different.

To make this easy and clean, **Spring has the provision of profiles to help separate the configuration for each environment.** So, instead of maintaining this programmatically, the properties can be kept in separate files such as *application-dev.properties* and *application-prod.properties*. The default *application.propertie*s points to the currently active profile using *spring.profiles.active* so that the correct configuration is picked up.