Spring Boot Interview Questions

Read Discuss Courses Practice

Spring boot is a Java-based spring framework used in Java Application development. Spring Boot is a microservice-based framework that makes a production-ready Java application in very less time. Spring is fast, has a low configuration, an inbuilt server, and monitoring features which help to build a Java application very fast from scratch with robustness and maintainability. We just need to use the proper configuration for utilizing a particular functionality because most of the functionalities are auto-configured. Spring Boot is very beneficial if we want to develop REST API. It also has extra support for embedded application servers like Tomcat, Jetty, etc.

Spring Boot Interview Questions

In this article, we've covered the **top 50 spring boot interview questions and answers**, which are categorized as basic to advanced interview questions. These questions are designed for both freshers as well as experienced candidates. By preparing with these diverse interview questions, you can increase your chances of acing your next spring boot interview round.

Table of Content:

- *Advanced Spring Boot Interview Questions for Experienced*

Spring Boot Interview Questions For Freshers

1. What is Spring Boot?

- Java developers may use the Spring Boot framework independent, executable spring applications that are production-grade. Rather than a thorough Spring configuration setup, you might start out with simple configurations.
- To read more, refer to the article – Introduction to Spring Boot

2. What are the advantages of using Spring Boot?

- **Easy to use:** The majority of the boilerplate code required to create a Spring application is reduced by Spring Boot. This shortens the time it takes to create apps and makes it simple for developers to get started with Spring Boot.
- **Rapid Development:** Spring Boot's opinionated approach and auto-configuration enable developers to quickly develop apps without the need for time-consuming setup, cutting down on development time.
- **Scalable:** Spring Boot apps are intended to be scalable. This implies they may be simply scaled up or down to match your application's needs.
- **Production-ready:** Metrics, health checks, and externalized configuration are just a few of the features that Spring Boot includes and are designed for use in production environments. This helps facilitate the implementation of Spring Boot applications to the production environment and contributes to the dependability of your applications.

3. Why do we prefer Spring Boot over Spring?

Here is a table that summarises why we use Spring Boot over

Feature	Spring	Spring Boot
Ease of use	More complex	Easier
Production readiness	Less production-ready	More production-ready
Scalability	Less scalable	More scalable
Speed	Slower	Faster
Customization	Less Customizable	More Customizable

- **To read more, refer to the article –** Difference between Spring and Spring Boot

4. Explain the working of Spring Boot

- Spring Boot works by scanning your application for annotations. These annotations tell Spring Boot what kind of application you are building and how to configure it. For example, if you have an annotation that indicates that your application is a web application, Spring Boot will automatically configure a servlet container and a web server.
- Spring Boot also provides a number of starters. These are dependency descriptors that make it easy to add common features to your application. For example, if you want to add a database to your application, you can add the Spring Boot starter for JDBC. This will automatically add the necessary dependencies to your project and configure the database connection for you.
- **Here are the main steps involved in how Spring Boot works:**
  1. You start by creating a new Spring Boot project.
  2. You add the necessary dependencies to your project.
  3. You annotate your application with the appropriate annotations.
  4. You run your application.

- **To read more, refer to the article –**Best Practices For Structuring Spring Boot Application

- A Spring application gets started by calling the main() method in the SpringApplication class. This method takes a SpringApplicationBuilder object as a parameter, which is used to configure the application. The SpringApplicationBuilder object can be used to set the application's name, version, and other properties.
- Once the SpringApplication object is created, the run() method is called. This method starts the application by creating an application context and initializing it.
- Once the application context is initialized, the run() method starts the application's embedded web server. The embedded web server is used to serve the application's web pages.
- **Example:**

Java

```java
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class MyApplication
{
  public static void main(String[] args) {
    SpringApplication.run(MyApplication.class, args);
  }
}
```

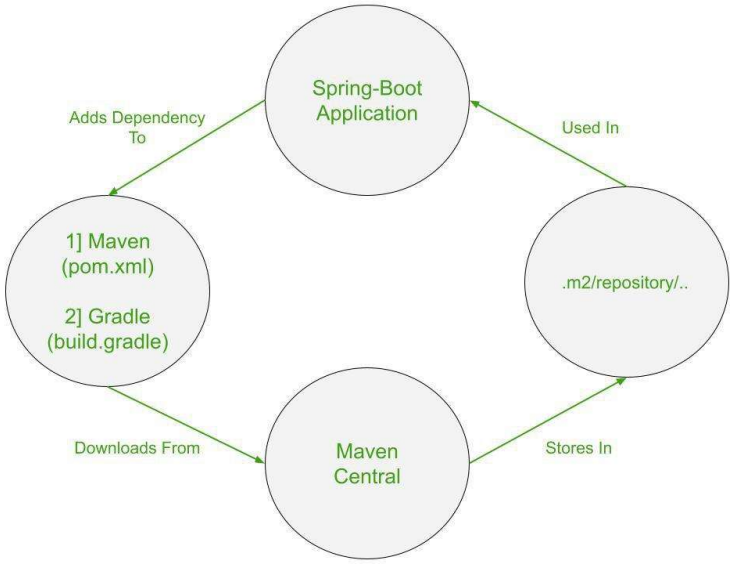
6. What are the Spring Boot Annotations?:

- **@SpringBootApplication:** This is the main annotation used to bootstrap a Spring Boot application. It combines three annotations: **@Configuration**,**@EnableAutoConfiguration**, and **@ComponentScan**. It is typically placed on the main class of the application.
- **@RestController:** This annotation is used to define a RESTful web service controller. It is a specialized version of the @Controller annotation that includes the @ResponseBody annotation by default.
- **@RequestMapping:** This annotation is used to map HTTP requests to a specific method in a controller. It can be applied at the class level to define a base URL for all methods in the class, or at the method level to specify a specific URL mapping.
- **@Autowired:** This annotation is used to automatically inject dependencies into a Spring-managed bean. It can be applied to fields, constructor parameters, or setter methods.
- **@Service:** This annotation is used to indicate that a class represents a service component in the application. It is typically used to annotate classes that contain

- **@Repository:** This annotation is used to indicate that a class represents a data repository or data access object (DAO). It is typically used for classes that interact with the database or other persistent storage.
- **@Component:** This annotation is the most generic annotation for any Spring-managed component. It is used to mark a class as a Spring bean that will be managed by the Spring container.
- **@Configuration:** This annotation is used to indicate that a class contains configuration methods for the application context. It is typically used in combination with @Bean annotations to define beans and their dependencies.
- **To read more, refer to the article** – Spring Core Annotations

7. What is Spring Boot dependency management?

- The Spring Boot framework offers a feature called Spring Boot dependency management that makes it easier to manage dependencies in a Spring Boot project. It makes sure that all necessary dependencies are appropriate for the current Spring Boot version and are compatible with it.
- **For example**, if you want to use Spring Boot to create a web application, you can add the Spring Boot starter web dependency to your application. This dependency will contain all of the dependencies that you need to use Spring Boot's web features, such as Spring MVC and Tomcat.
- **To read more, refer to the article** – **Spring Boot – Dependency Management**



8. What are the Spring Boot Starters?

- In Spring Boot, starters are a collection of pre-configured dependencies that make it

- These starters include all of the dependencies, version control, and configuration needed to make certain features of a Spring Boot application functional.
- To use a Spring Boot Starter, you simply add the dependency to your project's pom.xml file. For example, to add the Spring Boot starter web dependency, you would add the following dependency to your pom.xml file:

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

To read more, refer to the article – Spring Boot – Starters

9. What is Spring Boot Actuator?

- Spring Boot Actuator is a component of the Spring Boot framework that gives your applications production-ready operational monitoring and management capabilities. You can manage and monitor your Spring Boot application while it is running thanks to a set of pre-built endpoints and metrics that are provided.
- To use Spring Boot Actuator, you simply need to add the spring-boot-starter-actuator dependency to your project. Once you have added the dependency, Spring Boot will automatically configure your application to expose the actuator endpoints.
- **To read more, refer to the article –** Spring Boot Actuator

10. What is thymeleaf?

- Thymeleaf is a Java-based server-side templating engine used in Java web applications to render dynamic web pages. It is a popular choice for server-side templating in the Spring ecosystem, including Spring Boot.
- The server-side processing of HTML templates created by developers using Thymeleaf results in dynamic content. It is simple to read and write because it adheres to a natural and intuitive syntax that closely resembles that of conventional HTML.
- **To read more, refer to the articles**
  - Spring Boot – How Thymeleaf Works?
  - Spring Boot – Thymeleaf with Example

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

- Spring Boot CLI is a command line tool that can be used to create, run, and manage

Spring Boot quickly and easily.
- It is built on top of the Groovy programming language and leverages Spring Boot's powerful features to streamline the development process.

12. Explain Spring Data

- Spring Data is a powerful framework that can be used to develop data-oriented applications. It is easy to use and provides a number of features that make it a valuable tool for developers.
- It aims to simplify the development of data-centric applications by offering abstractions, utilities, and integration with various data sources.
- **Here are some of the projects that are part of Spring Data:**
  - **Spring Data JPA:** This project provides support for accessing data from relational databases using JPA.
  - **Spring Data MongoDB:** This project provides support for accessing data from MongoDB.
  - **Spring Data Neo4j:** This project provides support for accessing data from Neo4j.
  - **Spring Data Elasticsearch:** This project provides support for accessing data from Elasticsearch.

13. What are Profiles in Spring?

- For various environments or application scenarios, you can configure and manage various sets of bean definitions and configurations using Spring profiles. You can define particular configurations that are only activated when the corresponding profile is active by using profiles.
- To use Spring Profiles, you simply need to define a set of profiles in your application's configuration. You can then use the **spring.profiles.active** property to specify which profile you want to use.

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

- Yes, you can create a non-web application in Spring Boot. Spring Boot is not just for web applications.
- Using spring boot you can create applications like Microservices, Console applications, and batch applications.

15. Explain Spring MVC

- Spring MVC is a web MVC framework built on top of the Spring Framework. It provides a comprehensive programming model for building web applications.
- The MVC architectural pattern separates the application into three layers:
  - **The model:** The model represents the data that is used in the application.
  - **The view:** The view is responsible for displaying the data to the user.
  - **The controller:** The controller is responsible for handling user requests and interacting with the model and view.
- **To read more, refer to the article** – Spring MVC – Basic Example using JSTL

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

- Spring Boot can be used to create command-line applications. This is done by excluding the Spring Boot web starter dependency from the project's **pom.xml** file. This will prevent Spring Boot from automatically configuring the application for web applications.
- To create a command-line application in Spring Boot, you can use the Spring Boot CLI. The following command will create a new command-line application called "**my-command-line-project**[name]":

```
spring init --build=maven --dependencies=none my-command-line-project[name]
```

- This will create a new project called "my-command-line-project[name]" with the Spring Boot starter dependency excluded. You can then start developing your command-line application.
- To run the command-line application, you can use the following command:

```
java -jar my-command-line-project[name].jar
```

17. What Are Spring Boot DevTools Used For?

- The Spring Boot framework includes a module called Spring Boot DevTools that provides a number of development-time features and enhancements to increase developers' productivity.
- **Spring Boot DevTools can be used for the following purposes:**
  - Automatic application restart
  - Fast application startup:
  - Actuator endpoints

18. What is the starter dependency of the Spring boot module?

- To use a Spring Boot starter dependency, you simply need to add it to your project's pom.xml file. For example, to add the Spring Boot starter web dependency, you would add the following dependency to your pom.xml file:

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

To read more, refer to the article – Spring Boot – Starters

19. Define some common Spring Boot CLI commands?

- **spring init:** This command initializes a new project using Spring Initializr. You can use this command to create a new project with a specific set of dependencies.
- **spring run:** This command runs a Spring Boot application from the command line. You can use this command to quickly start up a Spring Boot application for testing or debugging purposes.
- **spring test:** Runs tests for a Spring Boot application. This command automatically detects and runs tests in your project.
- **spring start:** Starts a new interactive Spring Boot shell. The shell provides a command-line environment for running and managing Spring Boot applications.
- **spring help:** Displays the help documentation for the Spring Boot CLI, listing available commands and their usage.

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

- Yes, it is possible to change the port of the embedded Tomcat server in a Spring Boot application.
- A simple way is to set the server. port property in your application's application.properties file. For example, to set the port to 8081, you would add the following property to your application.properties file:

```
server.port=8081
```

- The default port of the embedded Tomcat server in Spring Boot is 8080.
- You can change the default port by setting the server. port property in your application's **application.properties** file.
- **To read more, refer to the article –** Spring Boot – Project Deployment Using Tomcat

22. Can we disable the default web server in the Spring Boot application?

- Yes, you can disable the default web server in the Spring Boot application.
- To do this, you need to set the **server.port** property to -1 in your application's application.properties file.

23. How to disable a specific auto-configuration class?

- To disable a specific auto-configuration class in a Spring Boot application, you can use the **@EnableAutoConfiguration** annotation with the exclude attribute. This allows you to exclude the auto-configuration class from being applied during the application startup process.
- For example, the following code would exclude the

Java

```
org.springframework.boot.autoconfigure.security.servlet.SecurityAutoConfiguration clas

@SpringBootApplication
@EnableAutoConfiguration(exclude = { SecurityAutoConfiguration.class })
public class MyApplication {
    public static void main(String[] args)
    {
        SpringApplication.run(MyApplication.class, args);
    }
}
```

24. Describe the flow of HTTPS requests through the Spring Boot application.

- The flow of HTTPS requests through a Spring Boot application is as follows:
  1. The client (e.g., a web browser) initiates an HTTPS request to the server.
  2. The server's SSL/TLS stack establishes a secure connection with the client.
  3. The client sends the request headers and body to the server.
  4. The server decrypts the request headers and body.
  5. The server handles the request and generates a response

7. The server sends the response back to the client.
8. The client decrypts the response headers and body.

25. What is the difference between RequestMapping and GetMapping?

Points	@RequestMapping	@GetMapping
Annotations	@RequestMapping	@GetMapping
Purpose	Handles various types of HTTP requests (GET, POST, etc.)	Specifically handles HTTP GET requests
Example	@RequestMapping(value = “/example”, method = RequestMethod.GET)	@GetMapping(“/example”)
Convenience	Less Convenient	More Convenient

Advanced Spring Boot Interview Questions for Experienced

26. How to enable Actuator in the Spring boot application?

- Here are the steps to enable the actuator
  - **Add Actuator dependency:** spring-boot-starter-actuator
  1. **Enable endpoints in application.properties:** management.endpoints.web.exposure.include
  2. **Run your Spring Boot app:** Run your Spring Boot application as usual. The Actuator endpoints will be exposed on the management port. By default, this is port 8080.
  3. **Access Actuator endpoints at URLs on the management port:** You can now access the Actuator endpoints at URLs like:
  4. 1. Health: http://localhost:8080/actuator/health
     2. Metrics: http://localhost:8080/actuator/metrics
     3. Beans: http://localhost:8080/actuator/beans

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

- Using the ApplicationContext object in Spring Boot, you can retrieve a list of all the beans in your application.
- The ApplicationContext is responsible for managing the beans and their dependencies.

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

- The Environment object in a Spring Boot application can be used to check the environment's properties. Access to the configuration settings for the application, including those defined in property files, command-line arguments, environment variables, and other sources, is made possible by the Environment.
- You can get the Environment instance by calling the getEnvironment() method on the SpringApplication class.

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

- **Add the logging level property to application.properties:** logging. level.<package> = DEBUG
- **Optionally, you can configure the log pattern to include useful information:** logging. pattern.console=%d{yyyy-MM-dd HH:mm:ss} – %msg%n
- **Run your Spring Boot app.** You will now see debug-level logs in the console and/or log files.
- **The log level can also be changed at runtime using the actuator endpoint:**
- **Curl -X POST \http://localhost:8080/actuator/loggers/<logger-name> \ -H 'content-type: application/json' \-d '{"configuredLevel": "DEBUG"}'**

30. What is dependency Injection?

- A design pattern called Dependency Injection (DI) enables us to produce loosely coupled components. In DI, an object's ability to complete a task depends on another object, the dependency object. However, the dependent object requests that another object (the injector object) provide the dependency object for it rather than creating the dependency object itself.
- **To read more, refer to the article –** Spring – Dependency Injection by Setter Method

31. What are the types of Dependency Injections?

- **Constructor injection:** This is the most common type of DI in Spring Boot. In

- **Setter injection:** In setter injection, the dependency object is injected into the dependent object's setter method.
- **Field injection**: In field injection, the dependency object is injected into the dependent object's field.
- **To read more, refer to the article** – Spring Dependency Injection with Example – GeeksforGeeks

32. What error do you see if H2 is not present in the classpath?

```
Caused by: java.lang.ClassNotFoundException: org.h2.Driver
```

33. What is an IOC container?

- A key component of a spring core framework or runtime environment that controls the creation, configuration, and dependency injection of objects (often referred to as beans) in an application is an IoC (Inversion of Control) container, also referred to as a DI (Dependency Injection) container.
- **To read more, refer to the article** – Spring – IoC Container

34. What is Spring Bean?

- An object that is managed by the Spring IoC container is referred to as a spring bean. The creation, configuration, and dependencies of objects in a Spring application are managed using this fundamental idea of the Spring Framework.
- A Spring bean can be any Java object, but it typically has the following characteristics:
  - It is annotated with the @Component annotation or one of its derivatives.
  - It has a default constructor.
  - It has a public no-argument constructor.
  - It has a public setter method for each of its dependencies.
- **To read more, refer to the article** – Spring – BeanFactory

35. What are Inner Beans in Spring?

In Spring, an Inner Bean refers to a bean that is defined within the scope of another bean's definition. It is a way to declare a bean inside the configuration of another bean, without explicitly giving it a unique identifier.

- To define an Inner Bean in Spring, you can declare it as a nested <bean> element

XML

```
<bean id="outerBean" class="com.example.OuterBean">
  <property name="innerBean" ref="innerBean"/>
</bean>
<bean id="innerBean" class="com.example.InnerBean"/>
```

Inner Beans can also be defined using the Java-based configuration in Spring, using the appropriate annotations.

To read more, refer to the article – Spring @Bean Annotation with Example

36. What is the difference between Constructor and Setter Injection?

Points	Constructor Injection	Setter Injection
Dependency	Dependencies are provided through constructor parameters.	Dependencies are set through setter methods after object creation.
Immutability	Promotes immutability as dependencies are set at creation.	Dependencies can be changed dynamically after object creation.
Dependency Overriding	Harder to override dependencies with different implementations.	Allows easier overriding of dependencies using different setter values.
Configurability	Limited configurability as dependencies are set during object creation and cannot be changed later.	Provides flexibility as dependencies can be changed dynamically at runtime.

37. What is Bean Wiring?

- Bean wiring is a mechanism in Spring that is used to manage the dependencies

- There are also two main types of bean wiring:
  - **Autowiring:** This is a feature of Spring that allows the Spring container to automatically wire beans together. Autowiring can be done by name or by type.
  - **Manual wiring:** This is the process of manually wiring beans together. This is done by the developer, who specifies the dependencies of a bean in the bean configuration file.
- **To read more, refer to the article –** Spring – Autowiring

38. What does the @SpringBootApplication annotation do internally?

- The @SpringBootApplication annotation is a convenience annotation that combines three annotations: **@EnableAutoConfiguration, @ComponentScan, and @Configuration**.
- Internally, the @SpringBootApplication annotation is processed by the Spring Boot framework. The framework will first check if the annotation is present on the main class of your application. If it is, the framework will then enable auto-configuration, scan the packages for @Component annotated classes, and register the beans in the Spring container.
- **To read more, refer to the article –** How to Run Spring Boot Application?

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

- The **@ComponentScan** annotation is used to tell Spring to scan a package and automatically detect Spring components, configurations, and services to configure.
- **The @ComponentScan annotation can be used in the following ways:**
  - **Without arguments:** This tells Spring Boot to scan the current package and sub-packages for @Component annotated classes.
  - **With basePackageClasses:** This tells Spring Boot to scan the specified classes and their sub-classes for @Component annotated classes.
  - **With basePackages:** This tells Spring Boot to scan the specified packages for @Component annotated classes.
- **To read more, refer to the article –**Spring @ComponentScan Annotation with Example

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

- Add the spring-boot-starter-jdbc dependency to your project. This dependency will

- Create an application.properties file and configure the database connection properties. The following are some of the properties that you need to configure:
  - spring.datasource.url: The JDBC URL for your database.
  - spring.datasource.username: The username for your database.
  - spring.datasource.password: The password for your database.
- Create a JdbcTemplate bean in your application. The JdbcTemplate bean is a Spring bean that provides you with a simple way to execute SQL queries and statements against your database.
- Use the JdbcTemplate bean to execute SQL queries and statements against your database.
- **To read more, refer to the articles**
  - Spring JDBC Example
  - Spring – JDBC Template
  - Spring Boot – JDBC

41. What are the @RequestMapping and @RestController annotations in Spring Boot used for?

- **@RequestMapping:** @RequestMapping is used to map HTTP requests to handler methods in your controller classes. It can be used at the class level and method level.
- @RequestMapping supports mapping by:
  - HTTP method – GET, POST, PUT, DELETE
  - URL path
  - URL parameters
  - Request headers
- **@RestController:** @RestController is a convenience annotation that combines @Controller and @ResponseBody. It indicates a controller where every method returns a domain object instead of a view.

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

Points	@SpringBootApplication	@EnableAutoConfiguration
When to use	When you want to use auto-configuration	When you want to customize auto-configuration

Points	@SpringBootApplication	@EnableAutoConfiguration
Entry point	Typically used on the main class of a Spring Boot application, serving as the entry point.	Can be used on any configuration class or in conjunction with @SpringBootApplication.
Component Scanning	Includes @ComponentScan annotation to enable component scanning.	Does not perform component scanning by itself.
Example	@SpringBootApplication public class MyApplication { public static void main(String[] args) { SpringApplication.run(MyApplication.class, args); } }	@Configuration @EnableAutoConfiguration public class MyConfiguration { }

43. What are the steps to connect an external database like MySQL or Oracle?

- **To connect an external database like MySQL or Oracle to a Spring Boot application, you need to follow these steps:**
  - Add the dependency for the JDBC driver of the database you want to connect to. For example, to connect to MySQL, you would add the mysql:mysql-connector-java dependency.
  - Create an application.properties file and configure the database connection properties. The following are some of the properties that you need to configure:
    - spring.datasource.url: The JDBC URL for your database.
    - spring.datasource.username: The username for your database.
    - spring.datasource.password: The password for your database.
    - spring.datasource.driver-class-name: The fully qualified class name of the JDBC driver for your database.
  - Create a JdbcTemplate bean in your application. The JdbcTemplate bean is a Spring bean that provides you with a simple way to execute SQL queries and statements against your database.
  - Use the JdbcTemplate bean to execute SQL queries and statements against your database.

- **To read more, refer to the articles –**Spring Boot – CRUD Operations using MySQL Database

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

- **Advantages:**
  - **Easy to Edit and Modify:** YAML files are easier to edit and modify manually, especially when dealing with large or nested configurations. The hierarchical structure and indentation make it less error-prone when making changes.
  - **Conciseness:** YAML syntax allows for a more concise representation of data compared to the key-value pairs in properties files. This can lead to shorter and more readable configuration files.
  - **Complex data types:** YAML supports complex data types like lists and maps, allowing for more flexible and expressive configuration. This is particularly useful when dealing with nested or multi-valued properties.
- **Ways to load YAML**:
  - **Using the @ConfigurationProperties annotation:** The @ConfigurationProperties annotation can be used to load YAML files into Spring Boot applications. This annotation can be used on a class that contains properties that are defined in a YAML file.
  - **Using the YamlPropertiesFactoryBean class:** The YamlPropertiesFactoryBean class can be used to load YAML files into Spring Boot applications. This class can be used to create a Properties object that contains the data from the YAML file.
- **To read more, refer to the article –** Difference between YAML(.yml) and .properties file in Java SpringBoot

45. Mention the differences between WAR and embedded containers

Feature	WAR	Embedded containers
Packaging	A WAR file is a Java archive that contains all of the files needed to deploy a web application to a web server.	An embedded container is a web application server included in the same JAR file as the application code.
Configuration	WAR deployment requires	Embedded containers typically use

Feature	WAR	Embedded containers
	(e.g., web.xml, context.xml) to define the web application.	within the application code to configure the web application and container settings.
Security	WAR files can be deployed to a web server that is configured with security features.	Embedded containers typically do not have the same level of security as web servers, but they can be made more secure by using security features that are provided by the Java Runtime Environment (JRE).
	than embedded containers, as they can be deployed to any web server that supports the WAR file format.	than WAR files, as they can only be deployed to the specific web application server that is included in the JAR file.

46. What Do you understand about Spring Data Rest?

- Spring Data REST is a framework that exposes Spring Data repositories as RESTful web services. It allows you to expose repositories as REST endpoints with minimal configuration.
- **Spring Data REST uses the following technologies:**
  - **Spring Data:** Spring Data provides a unified abstraction for accessing data from different data stores.
  - **Spring MVC:** Spring MVC is a web framework that provides a way to build RESTful web services.
- **For more information, Please Refer to these articles**
  - Spring – MVC Framework
  - Introduction to the Spring Data Framework

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

- Here are the reasons why not to choose Spring Data REST

- **Performance** – Since Spring Data REST generates endpoints dynamically, the performance may not be optimal for very large-scale applications.
- **Versioning** – It can be difficult to version the REST APIs exposed by Spring Data REST.
- **Relationships** – Handling relationships between entities can be tricky with Spring Data REST.
- **Filtering** – There are limited options for filtering the results returned by the endpoints.

- **For more information, Please Refer to this article-** Spring – REST Controller

48. What is Data JPA?

- Spring Data JPA is a module that provides a high-level abstraction over the Java Persistence API (JPA). Spring Data JPA makes it easy to develop applications that interact with relational databases.
- JPA is a specification that defines the Java standard for Object-Relational Mapping (ORM). It provides an API for mapping Java objects to relational database tables and performing CRUD (Create, Read, Update, Delete) operations using object-oriented concepts.

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

- Spring Boot automatically configures Hibernate as the default JPA implementation when you add the spring-boot-starter-data-jpa dependency to your project. This dependency includes the Hibernate JAR file as well as the Spring Boot auto-configuration for JPA.
- You can override the default Hibernate configuration by providing your own JPA properties or Hibernate properties in your application.properties file. But by default, Hibernate is chosen and configured for you.
- **For More Details, Refer to These articles**
  - Hibernate Architecture
  - Java – JPA vs Hibernate

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

- Here are the steps on how to deploy to a different server with Spring Boot:
  - **Build your Spring Boot application.** You can do this by running the mvn clean

- **Create a deployment package.** The deployment package will depend on the type of server you are deploying to. For example, if you are deploying to a Tomcat server, you will need to create a WAR file.
- **Deploy the deployment package to the server.** The specific steps for deploying the deployment package will depend on the type of server you are deploying to. For example, if you are deploying to a Tomcat server, you can deploy the WAR file by copying it to the webapps directory of the Tomcat server.
- **Start the server.** Once the deployment package has been deployed, you can start the server.

Spring Boot Interview Questions – FAQs

1. What will be the Spring Boot Interview Questions for 5 Years Experience?

In the interview, candidates with over 5 years of experience are primarily questioned about these concepts.

1. ***Auto-configuration:*** *Spring Boot automatically configures beans based on project dependencies, saving time in setup.*
2. ***Starters****: Dependency management artifacts for easy integration of common Spring Boot features like web apps, data access, and security.*
3. ***Production-ready applications:*** *Spring Boot provides embedded servers, actuators, and metrics for creating production-ready apps.*
4. ***Best practices:*** *Use dependency injection, version control, and thorough testing when developing Spring Boot apps.*
5. ***Challenges****: Understanding auto-configuration and selecting appropriate dependencies may be challenging.*
6. ***Improving skills****: Enhance Spring Boot skills through documentation, conferences, and contributing to the project.*

2. What are the most common Spring Boot interview questions?

The most common Spring Boot interview questions are:

- *What is Spring Boot?*
- *What are the advantages of using Spring Boot?*

- *What is the difference between Spring Boot and Spring Framework?*
- *What are the starter dependencies in Spring Boot?*
- *What is the purpose of the @SpringBootApplication annotation?*
- *What is the purpose of the @Configuration annotation?*
- *What is the purpose of the @Bean annotation?*
- *What is the purpose of the @Autowired annotation?*
- *What is the purpose of the @Value annotation?*
- *What is the purpose of the @Profile annotation?*
- *What is the purpose of the @EnableAutoConfiguration annotation?*
- *What is the default port of the embedded Tomcat server in Spring Boot?*
- *How to change the port of the embedded Tomcat server in Spring Boot?*
- *How to enable actuator in Spring Boot?*
- *How to access actuator endpoints in Spring Boot?*

3. How can I prepare for Spring Boot interview questions?

There are a few things you can do to prepare for Spring Boot interview questions:

- *Learn about Spring Boot*
- *Practice answering common Spring Boot interview questions*
- *Create a Spring Boot project and experiment with the different features*
- *Attend Spring Boot meetups and conferences*
- *Join the Spring Boot community on Stack Overflow and other forums*

4. What will be the Spring Boot Interview Questions for 2 Years Experience?

For candidates with up to 2 years of experience, interviews will typically focus on the core concepts of Spring Boot, such as auto-configuration, starters, actuator, and CLI. Questions may also be asked about how to create, configure, run, and deploy Spring Boot applications.-

- *Basics of Spring Boot*
- *Components of Spring Boot*
- *Create a Spring Boot application*
- *Configure Spring Boot application*

- *Deploy a Spring Boot application*
- *Best practices for developing Spring Boot applications*

5. What will be the Spring Boot Interview Questions for 3 Years Experience?

In the interview, candidates with over 3 years of experience are primarily questioned about these concepts.

- *What are the different ways to start a Spring Boot application?*
- *What are the different ways to configure Spring Boot applications?*
- *How to use Spring Boot starters?*
- *How to use Spring Boot actuator?*
- *How to use Spring Boot CLI?*
- *How to use Spring Boot in a microservices architecture?*
- *How to secure a Spring Boot application?*
- *How to troubleshoot Spring Boot applications?*

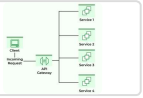
Last Updated : 14 Aug, 2023

5

Similar Reads

Difference Between Spring Boot Starter Web and Spring Boot Starter Tomcat

Spring Boot - Spring JDBC vs Spring Data JDBC


Java Spring Boot Microservices - Develop API Gateway Using Spring Cloud Gateway


Spring Boot | How to access database using Spring Data JPA

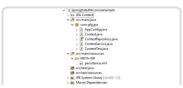
Difference between Spring MVC and Spring Boot


How to Create a Spring Boot Project in Spring Initializr and Run it in IntelliJ IDEA?

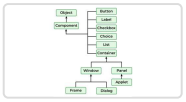

How to Create and Setup Spring Boot Project in Spring Tool Suite?


How to Run Your First Spring Boot Application in Spring Tool Suite?

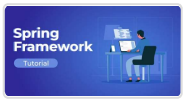

Spring Boot - Spring Data JPA


How to Make a Project Using Spring Boot, MySQL, Spring Data JPA and Maven?

Related Tutorials


Java AWT Tutorial

Spring MVC Tutorial



Spring Tutorial

Spring Boot Tutorial

Java 8 Features - Complete Tutorial

Previous
Spring Boot Interview Questions

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Easy | Normal | Medium | Hard | Expert

Article Tags : interview-questions, Java-Spring, Java-Spring-Boot, Picked, Advance Java, Java

Improve Article | Report Issue

GeeksforGeeks
A-143, 9th Floor, Sovereign Corporate Tower, Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Top 100 DSA Interview Problems

All Cheat Sheets

DSA Roadmap by Sandeep Jain

AngularJS

NodeJS

Express.js

Lodash

Computer Science

GATE CS Notes

Operating Systems

Computer Network

Database Management System

Software Engineering

Digital Logic Design

Engineering Maths

Python

Python Programming Examples

Django Tutorial

Python Projects

Python Tkinter

OpenCV Python Tutorial

Python Interview Question

Data Science & ML

Data Science With Python

Data Science For Beginner

Machine Learning Tutorial

Maths For Machine Learning

Pandas Tutorial

NumPy Tutorial

NLP Tutorial

Deep Learning Tutorial

DevOps

Git

AWS

Docker

Kubernetes

Azure

GCP

Competitive Programming

Top DSA for CP

Top 50 Tree Problems

Top 50 Graph Problems

Top 50 Array Problems

Top 50 String Problems

Top 50 DP Problems

Top 15 Websites for CP

System Design

What is System Design

Monolithic and Distributed SD

Scalability in SD

Databases in SD

High Level Design or HLD

Low Level Design or LLD

Top SD Interview Questions

Interview Corner

Company Wise Preparation

Preparation for SDE

GfG School

CBSE Notes for Class 8

CBSE Notes for Class 9

Internship Interviews

Competitive Programming

Aptitude Preparation

CBSE Notes for Class 11

CBSE Notes for Class 12

English Grammar

Commerce

Accountancy

Business Studies

Economics

Management

Income Tax

Finance

Statistics for Economics

UPSC

Polity Notes

Geography Notes

History Notes

Science and Technology Notes

Economics Notes

Important Topics in Ethics

UPSC Previous Year Papers

SSC/ BANKING

SSC CGL Syllabus

SBI PO Syllabus

SBI Clerk Syllabus

IBPS PO Syllabus

IBPS Clerk Syllabus

Aptitude Questions

SSC CGL Practice Papers

Write & Earn

Write an Article

Improve an Article

Pick Topics to Write

Write Interview Experience

Internships