**Spring Boot Interview Question**

**Q1. Spring 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.

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****

* 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, 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 spring boot application using maven, but if I have to name a few, then following are the ways to create a Spring Boot project/ application using maven

* 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. 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.

### ****Q8. 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.

**Q9. 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:

<dependency>

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

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

</dependency>

**Q10. 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.

**Q11. 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:

<dependency>

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

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

</dependency>

**Q12. 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.

### ****Q13. How to enable HTTP/2 support in Spring Boot?****

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

### ****Q14.  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; |

### ****Q15. Mention the differences between JPA and Hibernate.****

|  |  |
| --- | --- |
| **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 |

### Q16. 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.

### ****Q17. 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.

### Q18. 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.

@EnableAutoConfiguration(exclude= {DataSourceAutoConfiguration.class})

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

@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.

### Q18. 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 |

**Q19. 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

**Q20. 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.

### Q21. 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/\*”}

### Q22. 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.

**Q23. 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.

**Q24. 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.

**Q25. 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 |

**Q26. 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.

**Q27. 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

**Q28. 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

**Q29. 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. 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.