Top 15 Spring Boot Interview Questions with Answers for Java/JEE Programmers

Hello guys, if you are preparing for your next Java interview and Job description mentioned about Spring framework, then you should also prepare some **Spring Boot interview questions** to avoid disappointment. The Spring Boot is now the standard way to use Spring framework for Java development and almost all the companies are moving from the traditional way of using Spring Framework to more modern Spring Boot way. Gone are the days, where questions like have you used Spring Boot was asked to the developer. Nowadays interview expects more knowledge of Spring Boot from candidates and there are reasons for it, which we'll explore in this article.

Spring Boot aims to simplify Java development with Spring by removing major pain points with respect to configuration, dependency management and ease of development. As Craig Walls put in Spring boot in Action, **It's probably the best thing happened to Java after JDK 1.5 release and the introduction of Spring Framework itself some 15 years back.**

It introduces a host of features e.g. starter dependency, auto-configuration, embedded server, Spring Boot CLI, Spring Actuator, Spring Initializer etc to take the Java development with Spring to next level and that's why Spring Boot interview questions are becoming increasingly common in Java interviews.

In order to answer Spring Boot question with confidence, you not only know what problem Spring Boot solves but also in-depth knowledge of some of its core features like auto-configuration and starter dependencies. These two features eliminate a lot of configuration and setup work from Spring-based Java application.

Btw, If you are not familiar with them I first suggest you go through a beginners class like Spring Boot Essentials to get up-to-speed. Once you have good knowledge of Spring Boot, you can attempt these questions.

10 Spring Boot Interview Questions for Java Developers

Here is my list of some of the most common Spring Boot Interview Questions from Java interviews. These questions cover most of Spring Boot features e.g. auto-configuration, starter dependencies, Spring Actuator, Spring Initializer, Spring Boot CLI etc.

They are also collected from Java interviews for both intermediate and experienced Java developers e.g. from 2 to 5 years of experience.

1. What is Spring Boot? Why should you use it?

Spring Boot is another Java framework from Sring umbrella which aims to simplify the use of Spring Framework for Java development. It removes most of the pain associated with dealing with Spring e.g. a lot of configuration and dependencies and a lot of manual setups.

Why should you use it? Well, Spring Boot not only provides a lot of convenience by auto-configuration a lot of things for you but also improves the productivity because it lets you focus only on writing your business logic.

For example, *you don't need to setup a Tomcat server* to run your web application. You can just write

code and run it as Java application because it comes with an embedded Tomcat server. You can also create a JAR file or WAR file for deployment based on your convenience.

In short, there are many reasons to use Spring Boot. In fact, it's now the standard way to develop Java application with Spring framework.

2. What are some important features of using Spring Boot?
This is a good subjective question and used by the interviewer to gauge the experience of a candidate with Spring Boot. Anyway, following are some of the important features of Spring Boot framework:

1. Starter dependency
This feature aggregates common dependencies together. For example, if you want to develop Spring MVC based RESTful services then instead of including Spring MVC JAR and Jackson JAR file into classpath you can just specify spring-boot-web-starter and it will automatically download both those JAR files. Spring Boot comes with many such starter dependencies to improve productivity.

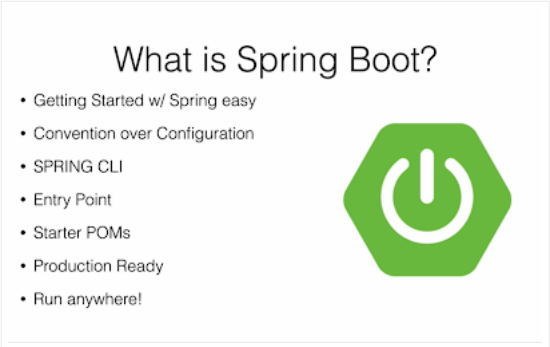
2. Auto-Configuration
This is another awesome features of Spring Boot which can configure many things for you. For example, If you are developing Spring web application and `Thymeleaf.jar` is present on the classpath then it can automatically configure Thymeleaf template resolver, view resolver, and other settings. A good knowledge of auto-configuration is required to become an experienced Spring Boot developers.

3. Spring Initializer
A web application which can create initial project structure for you. This simplifies initial project setup part.

4. Spring Actuator
This feature provides a lot of insights of a running Spring boot application. For example, you can use `Actuator` to find out which beans are created in Spring's application context and which request path are mapped to controllers.

5. Spring CLI
This is another awesome feature of Spring Boot which really takes Spring development into next level. It allows you to use Groovy for writing Spring boot application which means a lot more concise code.

If you are interested in learning more about these essential Spring Boot features then Dan Vega's **Learn Spring Boot - Rapid Spring Application Development** is a great place to start with.



3. What is auto-configuration in Spring boot? how does it help? Why Spring Boot is called opinionated?
There are a lot of questions in this one question itself, but let's first tackle auto-configuration. As explained in the previous example, it automatically configures a lot of things based upon what is present in the classpath.

For example, it can configure `JdbcTemplate` if its present and a `DataSource` bean are available in the classpath. It can even do some basic web security stuff if Spring security is present in the classpath.

Btw, if you are not familiar with spring security library then check out Spring Security Masterclass to learn more about it. It's one of the most important tools to secure modern-day Java application.

Anyway, the point is auto-configuration does a lot of work for you with respect to configuring beans, controllers, view resolvers etc, hence it helps a lot in creating a Java application.

Now, the big questions come, why it's considered opinionated? Well because it makes a judgment on its own. Sometimes it imports things which you don't want, but don't worry, Spring Boot also provides ways to override auto-configuration settings.

It's also disabled by default and you need to use either `@SpringBootApplication` or `@EnableAutoConfiguration` annotations on the Main class to enable the auto-configuration feature. See Spring Boot Essentials for learning more about them.

4. What is starter dependency in Spring Boot? how does it help?
This question is generally asked as a follow-up of the previous question as it's quite similar to auto-configuration and many developers get confused between both of them. As the name suggests, starter dependency deal with dependency management.

After examining several Spring-based projects Spring guys notice that there is always some set of libraries which are used together e.g. Spring MVC with Jackson for creating RESTful web services. Since declaring a dependency in Maven's `pom.xml` is the pain, they combined many libraries into one based upon functionality and created this starter package.

This not only frees you from declaring many dependencies but also fees you from compatibility and version mismatch issue. Spring Boot starter automatically pulls compatible version of other libraries so that you can use them without worrying about any compatibility issue.

5. What is the difference between @SpringBootApplication and @EnableAutoConfiguration annotation?
Even though both are essential Spring Boot application and used in the Main class or Bootstrap class there is a subtle difference between them. The `@EnableAutoConfiguration` is used to enable auto-configuration but `@SpringBootApplication` does a lot more than that.

It also combines `@Configuration` and `@ComponentScan` annotations to enable Java-based configuration and component scanning in your project.

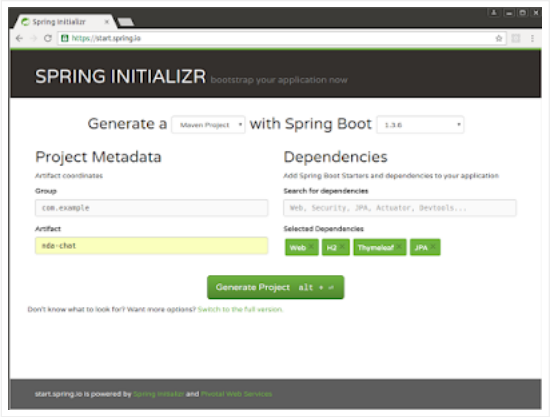
The `@SpringBootApplication` is in fact combination of `@Configuration`, `@ComponentScan` and `@EnableAutoConfiguration` annotations. You can also check Spring Boot MasterClass to learn more about this annotation and it's used.

Also, this Spring Boot question was recently asked to one of my friends in his last interview with a big Investment bank. He was interviewing for a front-office Java web application which uses Spring Boot in the back-end.

6. What is Spring Initializer? why should you use it?
One of the difficult things to start with a framework is initial setup, particularly if you are starting from scratch and you don't have a reference setup or project. Spring Initializer addresses this problem in Spring Boot.

It's nothing but a web application which helps you to create initial Spring boot project structure and provides Maven or Gradle build file to build your code.

I highly recommend to use it if you are starting the first time. If you want some assistance, you can check out this Spring Boot MasterClass.



7. What is Spring Actuator? What are its advantages?
This is an interesting Spring Boot question and mostly asked on Java roles which also has some support responsibility. Spring Actuator is another cool Spring Boot feature which allows seeing inside a running application.

Yes, you read it correctly. It allows you to see inside an application. Since Spring Boot is all about auto-configuration it makes debugging difficult and at some point in time, you want to know which beans are created in Spring's Application Context and how Controllers are mapped. Spring Actuator provides all that information.

It provides several endpoints e.g. a REST endpoint to retrieve this kind of information over the web. It also provides a lot of insight and metrics about application health e.g. CPU and memory usage, number of threads etc.

It also comes with a remote shell which you can use to securely go inside Spring Boot application and run some command to expose the same set of data. You can even use JMX to control this behavior at runtime.

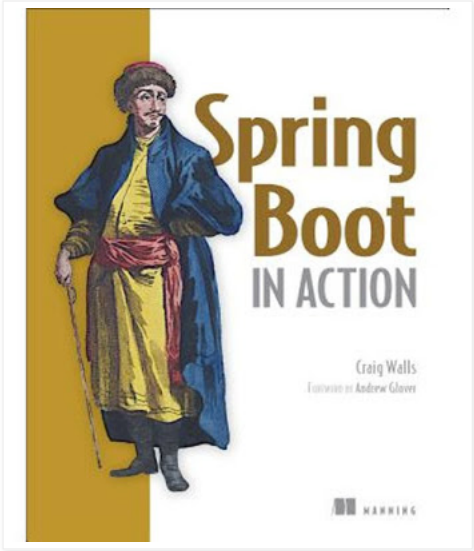
Btw, it's important to secure your Spring Actuator endpoints because it exposes a lot of confidential information and a potentially dangerous one-two. For example, by using `/showdown` endpoint you can kill a Spring Boot application.

But, don't worry. You can use Spring Security to secure Spring Actuator endpoints. If you are not familiar with Spring Security then you can check out Spring Security Certification Class to learn about it.

8. What is Spring Boot CLI? What are its benefits?
Spring Boot CLI is a command line interface which allows you to create Spring-based Java application using Groovy. Since it's used Groovy, it allows you to create Spring Boot application from the command line without ceremony e.g. you don't need to define getter and setter method, or access modifiers, return statements etc.

It's also very powerful and can auto-include a lot of library in Groovy's default package if you happen to use it.

For example, if you use `JdbcTempalte`, it can automatically load that for you. If you are interested in learning Spring Boot CLI, I recommend reading **Spring Boot in Action** book, another masterpiece from Craig Walls after Spring in Action.



9. Where do you define properties in 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 you can use Spring Initializer to create this file, albeit empty.

You don't need to do any special configuration to instruct Spring Boot load this file. If it exists in classpath then Spring Boot automatically loads it and configure itself and application code according.

For example, you can use to define a property to change the embedded server port in Spring Boot, which is also our next question.

10. Can you change the port of Embedded Tomcat server in Spring boot? If Yes, How?
Yes, we can change the port of Embedded Tomcat Server in Spring Boot by adding a property called server.port in the `application.properties` file.

As explained in the previous question, this property file is automatically loaded by Spring Boot and can be used to configure both Spring Boot as well as application code.

If you need an example, you can see this step by step tutorial to change the port of Embedded Tomcat Server in Spring Boot. You can further see Learn Spring Boot in 100 Steps to learn more about essential Spring Boot concepts and how to use them in a real project.

11. What is the difference between an embedded container and a WAR?
The main difference between an embedded container and a WAR file is that you can Spring Boot application as a JAR from the command prompt without setting up a web server. But to run a WAR file, you need to first set up a web server like Tomcat which has Servlet container and then you need to deploy WAR there.

12. What embedded containers does Spring Boot support?
Spring Boot support three embedded containers: Tomcat, Jetty, and Undertow. By default, it uses Tomcat as embedded containers but you can change it to Jetty or Undertow.

13. What are some common Spring Boot annotations?
Some of the most common Spring Boot annotations are `@EnableAutoConfiguration`, `@SpringBootApplication`, `@SpringBootConfiguration`, and `@SpringBootTest`.

The `@EnableAutoConfiguration` is used to enable auto-configuration on Spring Boot application, while `@SpringBootApplication` is used on the Main class to allow it to run a JAR file. `@SpringBootTest` is used to run unit test on Spring Boot environment.

14. Can you name some common Spring Boot Starter POMs?
Some of the most common Spring Boot Start dependencies or POMs are spring-boot-starter, spring-boot-starter-web, spring-boot-starter-test. You can use spring-boot-starter-web to enable Spring MVC in Spring Boot application.

15. Can you control logging with Spring Boot? How?
Yes, we can control logging with Spring Boot by specifying log levels on application.properties file. Spring Boot loads this file when it exists in the classpath and it can be used to configure both Spring Boot and application code.

Spring Boot uses Commons Logging for all internal logging and you can change log levels by adding following lines in the `application.properties` file:

```
logging.level.org.springframework=DEBUG
logging.level.com.demo=INFO
```

That's all about some of the **common Spring Boot Interview Questions for Java developers**. If you are preparing for Java development interview where Spring Boot is required as a skill then you should be familiar with these interview questions. They not only help you to do well in your interview but also encourage you to learn key Spring Boot concepts in details to make the best use of it.

Further Learning

Spring Framework Master Class - Beginner to Expert
Master Microservices with Spring Boot and Spring Cloud
Master Hibernate and JPA with Spring Boot in 100 Steps
Spring Boot Reference Guide
5 Free Spring Framework and Spring Boot Courses
Spring Boot: Efficient Development, Configuration, and Deployment

Thanks for reading this article so far. If you like these Spring Boot Interview Questions then please share with your friends and colleagues. If you have any feedback or doubt then please drop a note.

Posted by Javin Paul

Labels: interview questions, spring boot, spring framework

No comments:

Post a Comment

Enter your comment...

Comment as: Google Accou ▼

Publish Preview

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)