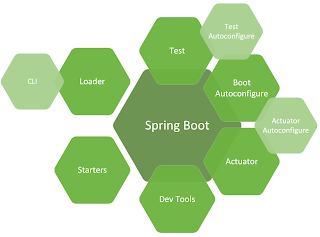
Hello Java developers, there is no doubt that the [Spring Boot](https://javarevisited.blogspot.com/2018/05/top-5-courses-to-learn-spring-boot-in.html) is now the standard way to develop Java application using Spring Framework, and that's why Spring Boot Questions are increasingly becoming popular on Java interviews. In the past, I have shared many [Spring MVC interview questions](https://javarevisited.blogspot.com/2018/11/top-20-spring-mvc-interview-questions-answers-for-java-developers.html) in this blog like the @RestController vs. @Controller, but I haven't got a chance to share any questions on [Spring Boot](https://javarevisited.blogspot.com/2018/07/top-5-books-to-learn-spring-boot-and-spring-cloud-java.html), [Spring Cloud](https://javarevisited.blogspot.com/2018/04/top-5-spring-cloud-courses-for-java.html), and [Microservices](https://medium.com/javarevisited/top-5-courses-to-learn-microservices-in-java-and-spring-framework-e9fed1ba804d). After a couple of personal requests from my readers, I thought to write about, and here comes my list of **top 20 Spring Boot interview questions**for Java developers.  
  
In this list, you will tackle all-important **Spring Boot concepts** from an interview point of view like **auto-configuration**, **starter dependencies**,**Spring Boot CLI**, and **Actuator**. Once you have gone through this list, you should have sufficient knowledge to tackle any Spring Boot question on the real interviews.  
  
Arguably, Spring Boot is the most significant and exciting thing that happens in Java development for a long time, I guess, after the release of Java 5. The main advantage of using [Spring Boot](https://javarevisited.blogspot.com/2018/02/top-5-spring-microservices-courses-with-spring-boot-and-spring-cloud.html) is that it greatly **simplifies the use of the Spring framework for Java development**.  
  
It addresses most of the concerns a new developer or a team has about using a framework like configurations, dependencies, setup hurdles, etc. It's the kind of application development framework Java programmers have been looking for a long time, and that's why it's very important from Java and Spring point of view.  
  
Btw, this is the third list of Spring interview questions on Javarevisited. Earlier I have shared some general Spring Framework interview questions, and more recently, I have written about some advanced Spring MVC and REST interview questions for Java developers.  
  
If you are preparing for Spring interview you can take a look at them and if you still need more questions you can also take help from this mega list of [**200+ Spring interview questions**](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-interview-questions-and-answers%2F) course on Udemy which is probably the biggest list of Spring interview questions I have come across.

**20 Spring Boot Interview Questions with Answer**

In this article, I am going to share some of the frequently asked questions from the Spring Boot framework, covering essential concepts like starter dependencies, auto-configuration, command-line interfaces, and Actuator in detail.  
  
The article is divided into 5 sections, the first section includes general Spring Boot interview questions like Comparision with Core Spring and Spring MVC.  
  
The second section focuses on the auto-configuration feature of Spring boot which removes most of the configuration required to work with the Spring framework.  
  
The third section focuses on Spring Boot's starter dependencies which consolidate common dependencies into one for ease of management.  
  
The fourth section focuses on interview questions about Spring Boot's Actuator, which provides insights about running Spring boot application, a very useful tool in production.  
  
Fifth, and the last section deals with Spring Boot CLI interview questions, which offers an attractive option for developing Spring Boot application using Groovy.

**1. Spring Boot Common Interview Questions**

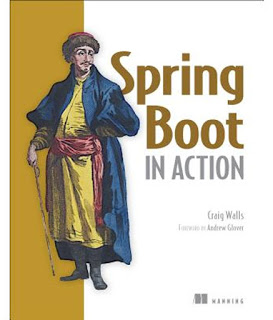
The main purpose of Spring Boot is to simplify the use of the Spring framework for Java development. It introduces a suite of features that address the pain associated with using the Spring framework for Java development as a lot of configuration, several JAR files, and a difficult startup curve. Now, let's see some of the common Spring Boot questions from this topic:  
  
  
**1. What is the Spring Boot?**  
Spring Boot is nothing but a Java framework that aims to simplify the use of the Spring framework for Java development. It introduces a slew of features like auto-configuration, embedded servlet container, starter dependencies, Actuator, and Spring Boot CLI to make Java web development with Spring MVC even easier.  
  
  
**2. What is the advantage of using Spring Boot?**  
There are several advantages of using Spring Boot the most important one is convenience and productivity. Spring Boot application can be run as a normal Java application like inside the main method. It also comes with starter dependency which means instead of explicitly loading all common libraries, you can load one starter and it will automatically import others.  
  
The Spring CLI features make writing Spring Boot application even easier using Groovy. You can even write a working Spring Boot application in less than 140 characters, something which you can tweet. You can further check out [**Learn Spring Boot in the 100 Steps**](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-boot-tutorial-for-beginners%2F) course on Udemy to learn more about what Spring Boot offers in terms of efficient configuration and development.

[](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-boot-tutorial-for-beginners%2F)

**3. What is the difference between Spring Boot and Spring MVC?**  
Even though both are part of the bigger Spring framework umbrella they are two different frameworks and solve different problems. Spring MVC makes Java web development easier by providing a consistent structure using the Model View Controller design pattern.  
  
On the other hand, Spring Boot aims to simplify the use of Spring Framework including [Spring MVC](https://www.java67.com/2017/11/top-5-free-core-spring-mvc-courses-learn-online.html) for Java development by adding some painful areas with respect to configuration, dependency management, and running the application.  
  
For example, you need a web-server to run a Spring MVC application but Spring Boot comes with embedded Tomcat or Jetty which can be used to run your Java web application.  
  
  
**4. What is the difference between Core Spring and Spring Boot?**  
Again, they are two different frameworks but comes under the same umbrella of the Spring framework. Core Spring generally refers to Spring container which provides [dependency injection](https://javarevisited.blogspot.com/2015/06/difference-between-dependency-injection.html) and [inversion of control](https://javarevisited.blogspot.com/2012/12/inversion-of-control-dependency-injection-design-pattern-spring-example-tutorial.html#axzz5N1cdCqrn), a key feature of Spring framework which promoted writing Java application in a different way 15 years back. While Spring Boot is now doing the same for the Spring framework, it is promoting a new way to use the Spring framework in the Java project.

**2. Spring Boot Auto-Configuration Interview Questions**

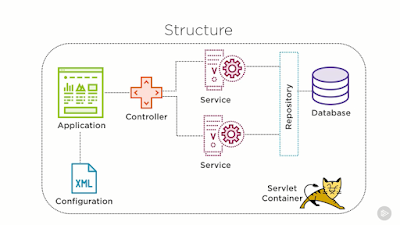
One of the important features of Spring Boot is the auto-configuration, which makes it possible for the Spring framework to intelligently detect what kind of application you're building and automatically configure the components necessary to support the application's need.  
  
It's built upon the conditional configuration feature introduced in Spring 4 which determines which configuration would be used and which will be ignored at runtime based upon dependencies on application's classpath, environment, and other factors, but with [Spring Boot](https://medium.com/javarevisited/10-advanced-spring-boot-courses-for-experienced-java-developers-5e57606816bd), you don't even need to write the explicit configuration for common scenarios.  
  
  
**5. What is Spring Boot auto-configuration?**  
Spring Boot auto-configuration is a runtime process that considers several factors to decide what Spring configuration should and should not be applied at the application startup-time. Some of its work may should magical to you.  
  
For example, if Spring's JdbcTemplate is available on classpath and if there is a DataSource bean then it can auto-configure a JdbcTemplateBean.  
  
Similarly, if Spring MVC is on the classpath, Spring's DisapatcherServlet will be configured and [Spring MVC](https://javarevisited.blogspot.com/2017/06/how-spring-mvc-framework-works-web-flow.html) will be enabled.  
  
  
**6. How does auto-configuration work? How does it know what to configure?**  
As explains in the previous example, Spring Boot auto-configuration checks classpath and makes decisions like if a Thymeleaf is present on the classpath then it can configure a Thymelead template resolver, view resolver, and a template engine automatically.  
  
If Spring Data JPA is on the classpath then it can automatically create repository implementations from repository interfaces.  Similarly, if Spring Security is present on the classpath then it can configure a very basic web security setup.  
  
It is seriously powerful as it takes more than 200 such decisions ever time an application starts up. If you need more examples, you can check out [**Spring Boot in Action**](https://www.amazon.com/Spring-Boot-Action-Craig-Walls/dp/1617292540?tag=javamysqlanta-20) book from Craig Walls, the best book to learn Spring Boot framework.

[](https://www.amazon.com/Spring-Boot-Action-Craig-Walls/dp/1617292540?tag=javamysqlanta-20)

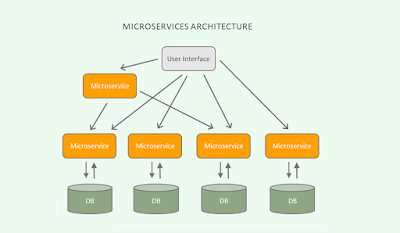
**7. What are some common Spring Boot annotations? (**[**answer**](https://www.java67.com/2019/01/top-5-spring-boot-annotations-java-programmers-should-know.html)**)**  
Some of the common Spring Boot annotations are @EnableAutoConfiguration, which enables auto-configuration and @SpringBootApplication which combines @Configuration, @ComponentScan, and @EnableAutoConfiguration into one to provide Java configuration, component scanning and enable Spring Boot's auto-configuration features. Both annotations are usually placed in the main class.  
  
  
**8. What does @EnableAutoConfiguration annotation do? (**[**answer**](https://javarevisited.blogspot.com/2018/05/the-springbootapplication-annotation-example-java-spring-boot.html)**)**  
The @EnableAutoConfiguration annotation enables auto-configuration in the Spring Boot project. This one line of code literally saves you from writing pages of configurations that would be required otherwise.

**3. Spring Boot Starter Dependency Interview Questions**

This is another interesting feature of Spring Boot which makes it easier to select which build and runtime dependencies your application will need by combining commonly needed dependencies. With Spring Boot starter dependency you can just include one dependency and rest assured that it will other common dependencies without worrying about the compatible version.  
  
  
**9. What is starter dependency in Spring Boot? What is the advantage of it?**  
The starter dependency feature of Spring Boot solves the problem of dependency management in your project. For example, if you want to build a Java web application using JPA and Thymeleaf template what are the dependencies you will need?  
  
You not only need to figure out a list of dependencies but also their compatible versions. This can be painful and risky if you don't have much experience. Spring Boot solves this problem by introducing "Starter".  
  
Instead of manually adding a library or specifying in Maven or Gradle build file you specify Starter. Each Starter aggregates common and related dependencies like to develop a Spring MVC based application instead of adding all JAR files you can just specify spring-boot-starter-web starter in your Maven or Gradle build file and it will import Spring MVC Jar and commonly used library like Jackson.  
  
Internally, Starters are nothing but Maven POM that defines transitive dependencies on other libraries that work together to provide some common functionality.  
  
Btw, If you want to learn more about starter dependency then I suggest you check out [**Spring Boot Fundamentals**](https://pluralsight.pxf.io/c/1193463/424552/7490?u=https%3A%2F%2Fwww.pluralsight.com%2Fcourses%2Fspring-boot-fundamentals) by Kesha Williams on Pluralsight.

[](https://medium.com/javarevisited/10-free-spring-boot-tutorials-and-courses-for-java-developers-53dfe084587e)

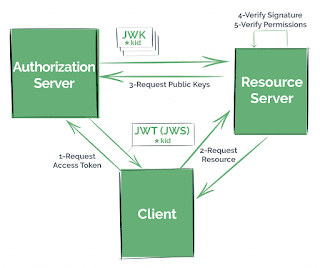
**10. How do you define properties in Spring Boot? Where?**  
You can define properties in the application.properties file. Spring Boot automatically loads this file and its properties can be used to configure both application and Spring boot. For example, if you want to change the default port of embedded tomcat server in Spring Boot then you can add server.port = 9000 in this file as shown in this [example](https://www.java67.com/2019/07/spring-boot-3-ways-to-change-port-of-tomcat.html). Also, when you use Spring Boot Initializer to create your project structure, it also creates this file but empty  
  
  
**11. What does @SpringBootApplication annotation do?**  
This is the main Spring Boot annotation which enables Spring's component-scanning and auto-configuration feature of Spring Boot. Every Spring Boot application which is not using Spring CLI needs this annotation. Internally, @SpringBootApplication is a combination of three annotations:  
  
Spring's @Configuration which enables Spring's Java-based configuration.  
  
Spring's ComponentScan which enables component scanning so that web controller classes and other components you write are automatically discovered and registers as spring beans in ApplicaitonContext.  
  
Spring Boot's @EnableAutoConfiguration which enables the auto-configuration feature of Spring Boot.  
  
The @SpringBootApplication annotation was introduced in Spring Boot 1. 2, which means in older versions you need to annotate your main class with all of these three annotations.  
  
If you want to learn more about this annotation I suggest you check [**Master Microservices with Spring Boot and Spring Cloud**](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fmicroservices-with-spring-boot-and-spring-cloud%2F) course by Ranga Karnam on Udemy.

[](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fmicroservices-with-spring-boot-and-spring-cloud%2F)

**12. What things affect what Spring Boot sets up?**  
Spring Boot's magic depends upon multiple things e.g. annotations, application's classpath, environment, etc. Spring CLI can automatically add packages to Groovy's default package if they are present in the application classpath. Similarly, you can enable auto-configuration by using @SpringBootApplication or @EnableAutoConfiguration annotations.

**4. Spring Boot Actuator Interview Questions**

This is probably the most important feature of a Spring Java application running on production. It provides insights into the inner working of a running Java application which is really important given so much automation. If you really need to know which beans are configured and how Spring MVC controllers are mapped to request path then this tool can help you.  
  
  
**13. What is the Spring Boot Actuator?**  
It's part of the Spring Boot application that provides deep insights about a running Spring boot application like beans in the Spring [application context,](https://javarevisited.blogspot.com/2012/11/difference-between-beanfactory-vs-applicationcontext-spring-framework.html#axzz6CgkxY2DJ) auto-configuration decisions, Spring MVC mappings, thread activity, and various applications health metrics.  
  
  
**14. What are the different ways Actuator provides to gain insight into a Spring Boot application?**  
Spring Boot Actuator provides a couple of ways to retrieve insights and other metrics like it provides REST endpoints, which means you can access metrics over HTTP. It also provides a remote shell to execute a command inside the Spring Boot application and expose much of the same data as the Actuator's endpoints. Also, all of the Actuator's endpoints are exposed as JMX MBeans, which means you can monitor and manage them in runtime using a JMX client like the jConsole.  
  
  
**15. Why you need to secure Spring Boot Actuator's endpoints?**  
You need to secure the Spring Boot Actuator's endpoints because it exposes a lot of information from running the Spring Boot application which can be considered sensitive. Some of Spring Boot endpoints like /shutdown are also dangerous as a rogue user can shut down the production instance resulting in loss of money and reputation.  
  
  
**16. How do you secure the Spring Boot Actuator's endpoint to restrict access?**  
Just like Spring MVC application, you can secure the Spring Boot Actuator's endpoints using Spring Security. You can add a Security starter as a build dependency and then security auto-configuration will take care of locking the application, including the Actuator endpoints.  
  
Though, you also need to do a bit of customization like setting up the right roles, username, password, etc. Btw, if you are not familiar with Spring Security then you can take a look at the [Learn Spring Security MasterClass](https://courses.baeldung.com/p/learn-spring-security-the-certification-class?utm_source=javarevisited&utm_medium=web&utm_campaign=lss&affcode=22136_bkwjs9xa) by Eugen Paraschiv to learn more about it.

[](https://www.java67.com/2017/12/top-5-spring-security-online-training-courses.html)

**5. Spring Boot CLI Interview Questions**

Spring Boot command line really makes it easy to develop Java web applications using Groovy. It takes advantage of Groovy's feature to remove the ceremony associated with writing Java code. With The Spring CLI, there is no need for accessor methods, access modifiers such as public, private, semicolons, or return keyword. In some cases, you can even get rid of import statements.  
  
  
**17. What is Spring Boot CLI?**([answer](https://javarevisited.blogspot.com/2018/11/top-5-spring-boot-features-java.html#axzz6CwWzuI40))  
It's a command-line interface provided by Spring Boot which makes it easy to write Spring Boot application using Groovy. It leverages the simplicity and convenience offered by auto-configuration and starter dependency and makes it even easier to write Spring boot application by taking Groovy's power to remove a lot of boiler-plate code from Java. In order to use Spring Boot CLI, you need to install it on your machine.  
  
  
**18. Can you write Spring application with Groovy?**  
Yes, Spring Boot CLI makes it possible to write Spring Boot application in Groovy. It also cut down a lot of noise which comes with writing code in Java.  
  
  
**19. What are the main advantages of the Spring Boot command-line interface (CLI)?**  
Apart from the convenience of writing Spring application with Groovy, Spring Boot CLI offers the following advantages:  
  
It can leverage Spring Boot auto-configuration and starter dependencies.  
It can detect when certain types are in use and automatically resolve the appropriate dependencies to support those types, though, not all.  
  
CLI has information about which packages some common types belong like [JdbcTempalte](https://javarevisited.blogspot.com/2020/05/10-jdbctemplate-examples-in-spring.html) and common Spring annotations like [@Controller](https://javarevisited.blogspot.com/2017/11/difference-between-component-service.html#axzz5p5OvBy6e). If those types are used, it can automatically add those packages to Groovy's default packages at compile time. You can further see  
  
  
**20. What does @Grab annotation do? When to use this?**  
The @Grab annotation is used to import types which Spring boot CLI cannot resolve automatically. It actually extends @Grab annotation from Gradle, a JAR dependency manager into [Groovy](https://javarevisited.blogspot.com/2017/08/top-5-books-to-learn-groovy-for-java.html#axzz5dPh77Fzl), so that, for many frequently used library you can import them by just adding module ID like you can just write @Grab("h2") instead of @Grab("com.h2database:h2:1.4.185) to import H2 in-memory database and Spring CLI will take care of version, etc.  
  
  
**21. What is Spring Initializer?**  
This is a bonus Spring Boot interview questions as we have already covered 20 questions. Well, the Spring Initializer is a web application that can generate Spring Boot project structure for you. Sometimes creating a project starter is the most difficult thing, especially if you are new to Spring development and Spring Initializer takes care of that. It doesn't generate any code but gives you basic project structure with Maven or Gradle build specification to build your code.  
  
  
**22. What is the difference between @SpringBootApplication and @EnableAutoConfiguration in Spring Boot?  (**[**answer**](https://www.java67.com/2018/05/difference-between-springbootapplication-vs-EnableAutoConfiguration-annotations-Spring-Boot.html)**)**  
The main difference is that @SpringBootApplicaiton annotation is relatively newer and only available from the Spring Boot 1.2 version while @EnableAutoConfiguration is present from the start. Also, @EnableAutoConfiguration just enables the auto-configuration feature of the Spring Boot application while @SpringBootApplication does three things.  
  
Read more: <https://javarevisited.blogspot.com/2020/05/top-20-spring-boot-interview-questions-answers.html#ixzz6OrEMtCOX>