Spring Boot Interview Questions

Description

What is Spring Boot?

Spring Boot aims to help the developers reduce the development time by mostly focusing their time on writing business logic. Spring Boot contains predefined set of dependencies which focus on reducing the time for application configuration. Spring Boot reduces the boiler plate code. Autoconfiguration is one of the feature which makes the developers life easier as Spring boot configures the classes by looking at the dependencies present.

Difference between Spring and Spring Boot

Spring	Spring Boot
Spring framework was released in 2002 and the main feature was dependency injection.	Spring framework has a lot of modules and Spring Boot is one of the modules which simplifies creating Spring based applications.
Difficult for developers to create applications, as a lot of configurations are required.	Spring Boot was designed to create an easy to start Spring application which you can go till Production with minimal configuration.
Development time required is more.	It reduces lots of development time and increases productivity
There is no embedded server, we need to add it manually to the IDE	It provides Embedded HTTP servers like Tomcat, Jetty etc. to develop and test our web applications very easily.

Spring	Spring Boot
Extra care needed while adding dependencyin pom.xml. We might end up adding another dependency with a version that might not be compatible with other dependencies. In short transitive dependency.	Spring Boot has starter projects or entries in pom.xml which will take care of downloading all the required dependencies and we don't need to worry on theversions that are downloaded.
Does not provide any in memory database.	Provides H2 which is the in memory database.
Framework used to create Enterprise applications.	Most widely used to create REST web services and Cloud micro services.

Features of Spring Boot

- Create stand-alone Spring applications
- Provides Embedded servers such as Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated 'starter' dependencies to simplify your build configuration
- Automatically configure Spring and 3rd party libraries whenever possible
- Provide production-ready features such as metrics, health checks, and externalized configuration.
- Absolutely no code generation and no requirement for XML configuration.
- Spring Boot starter projects used for dependency management.
- Integrates well with other Spring projects such as Spring Data, Spring Security, Spring Batch, Spring Credhub etc.
- Spring Initializer is a web tool which is used to create Spring Boot applications, all you have to do is select the spring boot version, fill the details like group name, artifact etc, select the dependencies presented to you, download the project and you are ready to go.

What are the different ways of creating Spring Boot Applications?

- Spring Boot CLI
- Spring Starter Project Wizard using STS IDE
- Spring Initializr
- Spring Maven Project

What is Spring Initializr?

Spring Initializr is an online portal provided by Official Spring to generate Spring Boot projects based on the details provided. Based on the dependencies we provide, it creates a that is available for download, and post that we can import in our IDE.

What is Externalization in Spring Boot

Externalization in Spring Boot is provided using below factors.
Application property files : Whatever configurations the application would need, are placed in either application.properties file or in YAML files. The extension of YAML files is application.yml. Both have their own style of expressing the configuration. Property files are placed in either current directory, classpath root or config directory from where the properties could be loaded.

Profile property files : Now that we know configurations are placed in either application.properties or application.yml files. We can now define properties specific to the environment in which your application is deployed. Example if your application is deployed in a DEV environment or UAT environment, the properties will be picked up based on the profile that we will set to our files. Example application-uat.yml , this file has UAT as the profile set and will be used by the application running on UAT environment.

Command-line properties : Whatever arguments we provide in the command line it will be added as properties and it will be added as environment properties.

What are Spring Boot Starter Projects

Once we add Spring boot starter dependencies, all the associated jars will be downloaded.
E.g. In case of starter-web dependency, all the required jars will be downloaded so that we can start building RESTful web services or Spring MVC based applications.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <version>2.3.0.RELEASE</version>
</dependency>
```

- spring-boot-starter-data-jpa â€“ Spring Data JPA with Hibernate
- spring-boot-starter-security â€“ Used for Spring Security
- spring-boot-starter-aop: This starter is used for aspect-oriented programming with AspectJ and Spring AOP
- spring-boot-starter-test: Is the starter for testing Spring Boot applications

Explanation of Spring Boot actuators

Spring boot actuators are used to check the metrics and various other aspects of a running production application. There are various endpoints by which we can get all the information.

Some of the features include.

- Beans – Displays a complete list of all the Spring beans in your application.
- Health – Shows application health information.
- Info – Displays arbitrary application info.
- Metrics – Shows â€˜metricsâ€™ information for the current application

Explanation of Spring Boot devtools

Spring Boot includes a set of tools that can make the application development experience a little more pleasant and faster.

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <optional>true</optional>
    </dependency>
</dependencies>
```

Applications that use spring-boot-devtools automatically restart whenever files on the classpath change. This can be a useful feature when working in an IDE, as it gives a very fast feedback loop for code changes. By default, any entry on the classpath that points to a directory is monitored for changes. Note that certain resources, such as static assets and view templates, do not need to restart the application.

What is auto configuration in Spring boot

Enable auto-configuration of the Spring Application Context, attempts to guess and configure beans that you are likely needed. Auto-configuration classes are usually applied based on your classpath and what beans you have defined. For example, if you have tomcat-embedded.jar on your classpath you are likely to want a TomcatServletWebServerFactory (unless you have defined your own ServletWebServerFactory bean).
When using @SpringBootApplication, the auto-configuration of the context is automatically enabled and adding this annotation has therefore no additional effect.

Auto-configuration tries to be as intelligent as possible and will back-away as you define more of your own configuration.

Spring Boot auto-configuration attempts to automatically configure your Spring application based onthe jar dependencies that you have added. For example, if HSQLDB is on your classpath, and youhave not manually configured any database connection beans, then Spring Boot auto-configures an in-memory database.

@SpringBootApplication annotation

@SpringBootApplication = @Configuration + @ComponentScan + @EnableAutoConfiguration

Most of you might have seen this annotation in the Spring boot projects main class. This Single annotation serves 3 purposes.

@Configuration– The Class that has this annotation is the source of beans. This annotation is used for defining beans using Java configuration.

@ComponentScan – Spring container scans through all the packages and registers the beans in the Spring Application Context

@EnableAutoConfiguration – This is one of the intelligent features of Spring boot and tries to configure beans that you are likely needed looking at the jars present in the Classpath .

What is the purpose ofÂ `@RestController` Â annotation in Spring Boot?

The `@RestController` annotation is a combination of two annotations i.e. `@Controller` and `@ResponseBody` annotation. This annotation eliminates the need to annotate methods that map requests with the `@ResponseBody` annotation thereby making the code cleaner. The `@ResponseBody` annotation serializes the return object into HttpResponse.

What is the purpose of `@ConditionalOnMissingBean`Â annotation?

This annotation is used along with @Bean annotation. @ConditionalOnMissingBean is a way to tells the auto configuration class to initialize the bean only when it is not able to find the bean in application context. If the bean already exists then it will not get created.

```
@ConditionalOnMissingBean
@Bean
publicÂ DiscountÂ myDiscount()Â {Â Â Â Â
returnÂ newÂ Discount();
}
```

myDiscount will be initialized only when no other bean of type **myDiscount** exists in the context. If it does, then **myDiscount** will not be registered as a bean.

The main purpose of this annotation is to provide a fallback bean, in case no bean of that type is present.

How to change tomcat server to jetty in Spring Boot

Tomcat is the default server that comes with spring-boot-starter-web dependency. In order to change the default server from Tomcat to Jetty, we need to exclude the Tomcat dependency and Jetty dependency.

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

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

How to change default port of embedded tomcat server in Spring Boot

The `server.port` property in the application.properties file is used to assign a different port to Tomcat server.

```
server.port = 8181
```

How to disable default web server in Spring

Boot

The `spring.main.web-application-type=none` property in the application.properties file is used to disable the default web server and change the web application type.

What is Spring Boot starter parent (spring-boot-starter-parent dependency)?

spring-boot-starter-parent contains the default settings like default java version, default encoding as well as default plugin configuration like maven-jar-plugin, maven-failsafe-plugin etc. It also contains the working combination of dependencies and their version numbers. Everything present in the parent plugin is inherited by the child pom. Below is how the starter parent dependency looks like

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.0.RELEASE</version>
</parent>
```

Difference between yaml and properties files

Both yaml and properties files are used to store properties or configurations that can be accessed from your spring boot application. If you are using properties file in your application then the file name can be **application.properties**, but if you are using yaml files to store configuration then it will be **application.yml**. Yaml is considered more user friendly as far as readability is concerned. To understand this more, let’s take an example of creating database configuration using both properties and yaml files.

application.properties

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=password
```

application.yml

```
spring:
  datasource:
    url: jdbc:h2:mem:testdb
    driverClassName: org.h2.Driver
    username: sa
    password: password
```

We can also validate the yaml file using some online validator.

Explain about @ConfigurationProperties

@ConfigurationProperties is used to read data from the properties file.

This annotation is used to read configurations from the properties file and map it to Java bean. Along with this annotation we use @Configuration.

application.properties

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=password
```

```
@Configuration
@ConfigurationProperties(prefix = "spring.datasource")
public class ConfigProperties {

    private String url;
    private String driverClassName;
    private String username;
    private String password;;

    // standard getters and setters
}
```

Profiles in Spring boot

Profiles in spring boot are used to differentiate between the configurations specific to a particular environment. E.g. If your application is running in a Development environment for development purpose, and Production environment for the real users, the database connectivity for both environments will be different.
If we store dev specific configuration in application-dev.yml and production specific configuration in application-prod.yml , based on the environment which your application is running in, it will pick the configurations from that specific file. Here â€œdevâ€• and â€œprodâ€• are the profiles. Profiles are set in the environment variables, so while the application startup, it will read the configuration based on the profile.

Embedded tomcat in Spring boot

Remember the days when we used to download the Apache tomcat and set it up on our IDE? It’s no more the case now. Whenever you create a Spring boot application, it comes with an embedded tomcat along with it and this provides convenience to the developers. The servlet container config has a separate configuration, but now its part of the application config.

Various annotations in Spring boot

Core Spring annotations
@Required
@Autowired
@Configuration
@ComponentScan
@Bean
@Component
@Controller
@Repository

Spring boot annotations
@SpringBootApplication
@EnableAutoConfiguration
@Configuration

Spring REST annotations
@RestController
@GetMapping
@PostMapping
@PutMapping
@DeleteMapping
@PathVariable
@RequestParam
@RequestHeader
@RequestBody

Category

1. Interview

Date Created
March 15, 2021
Author
kk-ravi144gmail-com