1. **What is spring boot? Why did you use spring boot in your project? Why not spring?**

**Spring Boot –** is a spring module

**Elaborating –** Spring boot is a framework for RAD build using spring framework with extra support of auto configuration and embedded application servers(Like tomcat, Jetty)

**It provides RAD –** Rapid Application Development.

**It helps us in creating efficient fast stand-**alone applications, which you can just run it removes a lot of configurations and dependencies.

1. **What RAD that you are talking about? How you can achieve RAD using spring boot?**

RAD is modified waterfall model, which focuses on developing software in a short span of time

Phases of RAD are as follows:

**Business Modelling:**

Business model is designed for the product to be developed.

**Data Modelling:**

Data Model is designed, the relation between these data objects are established using information gathered in first phase.

**Process Modelling:**

Process model is designed. Process description for adding, deleting, retrieving or modifying a data object are given

**Application Generation**

The actual product is built using coding. Convert process and data models into actual prototypes.

**Testing and Turnover**

Product is tested and if changes are required, the whole process starts again.

**Note:** By using spring application, fourth step will consume more time. By using spring boot this will be reduced.

**(3) Is this possible to change the port of embedded tomcat server in spring boot?**

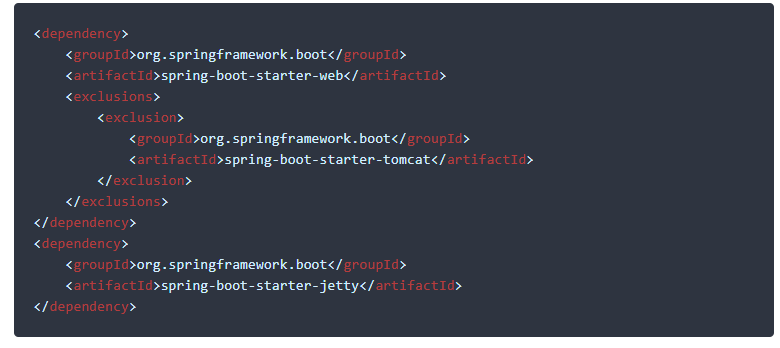
Put server. Port properties in application. Properties



**(4)Can we override or replace embedded tomcat server in spring boot?**

Yes, we can replace the embedded tomcat with any other servers by using starter dependencies

**Example:** You can use spring-boot-starter-jetty as a dependency for each project as you need



**(5)Can we disable the default web server in web application?**

The major strong point in spring is to provide the flexibility to build your web application loosely coupled. Spring provides features to disable the web server in a quick configuration. Yes, we can use the application. Properties to configure web application type i.e. Spring.main.web-application-type = none.



**(6)How to disable a specific auto-configuration class?**

You can use the exclude attribute of @EnableAutoConfiguration, if you find any specific auto configuration classes that you do not are being applied.

//By using exclude

@EnableAutoConfiguration (exclude = {DataSourceAutoConfiguration.class})



**@Configuration:**

It is a class level annotation.

Annotating a class with @configuration indicates that the class can be used by

Spring container as a source of bean definitions can use the class.

**Example:**

@Configuration

Class AppConfig {

@Bean

Public Student getStudentObject () {

return new Student ()

}

}

**@EnableAutoConfiguration:**

This Annotation auto configures the beans that are present in the classpath.This simplifies the developers work by guessing the required beans from classpath and configure it to run the application.

For Example, if you have tomcat-embedded.jar in the class path, then you will need a Tomcat Embedded servlet container factory bean to configure the tomcat server. This will be searched

And configured without any manual xml configurations.

**@ComponentScan**

It scans a package and all of its sub packages, looking for classes that could be automatically registered as beans in spring container

**(7) What does the @SpringBootApplication annotation do internally?**

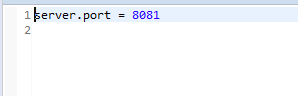
As per the spring boot doc, the @SpringBootApplication annotation is equivalent to using @configuration, @EnableAutoConfiguration, and @ComponentScan with their default attributes. Spring Boot enable the developer to use a single annotation instead of using multiple. However, as we know, spring provided loosely coupled features that we can use for each individual annotation as per project

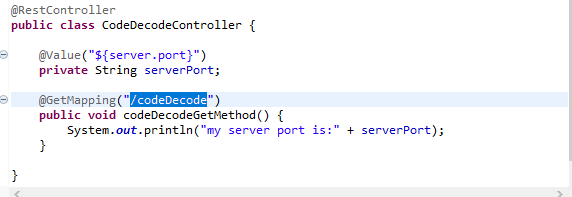
**(8) How to use property defined in application. Properties file into your java class.**

Use the @Value annotation to access the properties, which is defined in application. Properties file

@Value (“${server.port}”)

private String serverPort;





**(9)Explain @Rest Controller annotation in spring?**

@RestController is a convenience annotation for creating restful controllers. It is a specialization of @component and is auto detected through class path scanning. It adds @controller and @response Body annotations. It converts the response to JSON or XML.

Which eliminates the need to annotate every request handling method of the controller class with the

@ResponseBody annotation. It is typically used in combination with annotated handler methods based on the @Request mapping annotation.

Indicates that the data returned by each method will be written straight into the response body instead of rendering a template.

**(10) Difference between @RestController and @Controller in Spring Boot?**

To answer this we first understand the difference between a web application and a REST API.

It is that the response from a web application is generally a view (HTML+CSS+JAVASCRIPT)

Because they are intended for, human viewers while rest API just returns data in form of JSON OR XML because most of the REST clients are programs.

Same goes with @RestController and @Controller annotation.

@Controller Map of the model object to view or template and makes it human readable

But @RestController simply returns the object and object data is directly written into http response as JSON OR XML.

**(11) What is the difference between @RequestMapping and @GetMapping?**

RequestMapping can be used with GET, POST, PUT and many other request methods using the method attribute on the annotation. Whereas get mapping is only the extension of request mapping, which helps you to improve clarity on request.

**(12)What is the use of profiles in spring boot?**

When developing applications for the enterprise, we typically deal with multiple environments such as Dev, QA, and prod.The configuration properties for the environment are different.

For Example, we might be using an embedded H2 database for Dev, but Prod could have the proprietary Oracle or DB2.Even if the DBMS is the same across environments, the URLS would definitely be different.

To make this easy and clean, spring has the provision of profiles, to help separate the configuration for each environment.so that instead of maintaining this programmatically, the properties can be kept in separate files such as application-dev. properties and application-prod.properties.The default application. Properties points to the currently active profile using spring.profile.active so that the correct configuration is picked up.

**(13) What is Spring Actuator? What are its advantages?**

Actuator is a manufacturing term that refers to a mechanical device for moving or controlling something. Actuators can generate a large amount of motion from small change.

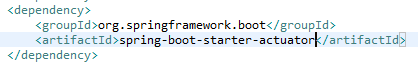
In spring boot, Actuator Is an additional feature that helps you monitor and manage your application when you push it to production. These features includes Auditing, Health, and Metrics gathering and may more features that can be automatically applied to your application.

You can enable this feature by adding the dependency: spring-boot-starter-actuator in pom.xml

Using spring actuator, you can access those flows like what bean is created, what is the CPU usage, http hits that your server has handled.

**How to use Actuator in Spring Boot Application:**

Add dependency in pom.xml



**Hit:**

<http://localhost:8081/actuator/health>

O/P:

{“status”:”UP”}, without doing anything because of dependency

Actuator endpoints:

<http://localhost:8090/actuator/>

To explicitly include / expose all endpoints use this in properties file:

Management.endpoints.web.exposure.include = \*

To expose only selected endpoints

Management.endpoints.jmx.exposure.include = health.info.env.beans

To get environmental configuration about the server.

<http://localhost:8080/actuator/env>

To get all the spring beans loaded in the context.

<http://localhost:8080/actuator/beans>

**Enabling HTTP Trace:**

Before 2.2 x spring boot, you can just add dependency and expose that using:

management.endpoints.web.exposure.include = \*

But after 2.2 x http trace does not work at all. This is not a bug in spring boot rather it is a legitimate change done because the default-implementation stores captured data in memory. Hence, it consumes much memory, and memory is a pretty costly and precious. That is why this feature is now turned of by default and has to be turned on by the user explicitly, if needed.

To fix this issue just create the bean of HttpTraceRepository, which is in memory repository. This will store the last 100 http request-response exchanges into your memory.

By Default all endpoints comes in deafultcontext path of the application, suffixed with /actuator.

If for some reason, we have existing endpoints in application starting with /actuator then we can customize the base path to something else.

Management.endpoints.web.base-path = /manage

Now we will be able to access all actuator endpoints under a new URL

Example: /manage/health

**Customize the management server port**

Add this to profile: management.server.port = 8090

**How to create custom endpoints**

This can be achieved by adding the following annotations:

@Endpoint and @component to class

@Read Operation, @Write Operation or @Delete Operation on method level

@Read Operation maps to HTTP GET

@Write Operation maps to HTTP POST

@Delete Operation maps to HTTP DELETE

By adding @Bean annotated with @Endpoint, any method annotated with @Read Operation, Write Operation, or @Delete Operation are automatically exposed over JMX or HTTP>

**Steps to deploy Spring Boot Web Applications as JAR and WAR**

To deploy Spring Boot web applications, you just have to add the following plugin in pom.xml file

<plugin>

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

<artifactId>spring-boot-maven-plugin</ artifactId >

</plugin>

By using the above plugin, you will get a jar executing the package phase. This JAR will contains all necessary libraries and dependencies required. It will also contain an embedded server. Therefore, you can run the application like an ordinary jar file.

Note: The Packaging element in the pom.xml file must be set to jar to build a JAR as below.

<Packaging>jar</packaging> or <packaging>war</packaging>

**Advantages of YAML file over Properties file:**

More clarity and better readability

Perfect for hierarchical configuration data, which is also represented in a better, more readable format

Support for maps, lists and scalar types.

**Global Exception Handling in Spring Boot:**

In Real World Projects, It is very important to handle error correctly and simultaneously provide meaningful error messages to the clients too.

Today we will see how to properly handle errors specifically in spring boot.

Prerequisite: Spring boot, creation of rest Apis

**How can we make Error response clear in spring?**

We are lucky enough that spring already comes with built in support for error handling

**Annotation used for Spring Boot:**

**@Rest Controller:** is the basic annotation for classes that handle REST operations

**@Controller Advice:**

The @Controller Advice annotation handles exceptions globally. It allows you to use the same ExceptonHandler for multiple controllers. This way, we can define how to treat an exception in just one place because this handler will be called whenever the exception is thrown from classes that are covered by Controller Advice.

As the name suggests, is “Advice” for multiple controllers.

Allows our class to be a global interceptor of exceptions thrown by methods annotated by request mapping.

**@Exception Handler:**

Spring annotation that provides a mechanism to treat exceptions that are thrown during execution of handlers (Controller operations).This annotation if used on methods of controller classes, will serve as the entry point for handling exceptions thrown within this controller only.

Altogether, the most common way is to use @ExceptionHanler on methods of @Controller Advice classes so that the exception handling will be applied globally or to a subset of controllers

@ExceptionHandler and @Controller Advice are used to define a central point for treating exceptions and wrapping them up in a class.

**@Response Status:**

Our error responses are always giving us the HTTP status 500 instead of a more descriptive status code.

To address this we can annotate our exception with @response status and pass in the desired http

Response status.

You can also override the existing exception handlers. Spring Boots built-in exception class **Response EntityExceptionHandler** has multiple methods that you can override to customize the exception handling further.

**Spring Boot + Hibernate Interview Questions for second/Higher rounds**

**Explain Architecture of your Application u worked on**

This is one of the most asked interview question.

We all work in IT industry where we have some clients, we serve the request, and they pay us

Hence, Most of the Applications work on client server architecture

