**Spring**

### 1) What is Spring?

It is a lightweight, loosely coupled and integrated framework for developing enterprise applications in java.

### 2) What are the advantages of spring framework?

1. Predefined Templates
2. Loose Coupling
3. Easy to test
4. Lightweight
5. Fast Development
6. Powerful Abstraction
7. Declarative support

### 3) What are the modules of spring framework?

1. Test
2. Spring Core Container
3. AOP, Aspects and Instrumentation
4. Data Access/Integration
5. Web

### 4) What is IOC and DI?

IOC (Inversion of Control) and DI (Dependency Injection) is a design pattern to provide loose coupling. It removes the dependency from the program.

### 5) What is the role of IOC container in spring?

IOC container is responsible to:

* create the instance
* configure the instance, and
* assemble the dependencies.

**6) What are the types of IOC container in spring?**

There are two types of IOC containers in spring framework.

1. Bean Factory
2. Application Context.

### 7) What is the difference between Bean Factory and Application Context?

|  |  |
| --- | --- |
| **BeanFactory** | **ApplicationContext** |
| It is an interface defined in org. springframework.beans.factory.**BeanFactory** | It is an interface defined in org. springframework.context.**ApplicationContext** |
| It uses Lazy initialization | It uses Eager/ Aggressive initialization |
| It explicitly provides a resource object using the syntax | It creates and manages resource objects on its own |
| It doesn’t supports internationalization | It supports internationalization |
| It doesn’t supports annotation based dependency | It supports annotation based dependency |

### 8) What is the difference between constructor injection and setter injection?

|  |  |  |
| --- | --- | --- |
| **No.** | **Constructor Injection** | **Setter Injection** |
| 1) | No Partial Injection | Partial Injection |
| 2) | Doesn't override the setter property | Overrides the constructor property if both are defined. |
| 3) | Creates new instance if any modification occurs | Doesn't create new instance if you change the property value |
| 4) | Better for too many properties | Better for few properties. |
|  |  |  |

### 9) What is autowiring in spring? What are the autowiring modes?

Autowiring enables the programmer to inject the bean automatically. We don't need to write explicit injection logic.

The autowiring modes are given below:

|  |  |  |
| --- | --- | --- |
| **No.** | **Mode** | **Description** |
| 1) | no | this is the default mode, it means autowiring is not enabled. |
| 2) | byName | injects the bean based on the property name. It uses setter method. |
| 3) | byType | injects the bean based on the property type. It uses setter method. |
| 4) | constructor | It injects the bean using constructor |

### 10) What are the different bean scopes in spring?

There are 5 bean scopes in spring framework.

|  |  |  |
| --- | --- | --- |
| **No.** | **Scope** | **Description** |
| 1) | singleton | The bean instance will be only once and same instance will be returned by the IOC container. It is the default scope. |
| 2) | prototype | The bean instance will be created each time when requested. |
| 3) | request | The bean instance will be created per HTTP request. |
| 4) | session | The bean instance will be created per HTTP session. |
| 5) | Global session | The bean instance will be created per HTTP global session. It can be used in portlet context only. |

### 11) In which scenario, you will use singleton and prototype scope?

Singleton scope should be used with EJB **stateless session bean** and prototype scope with EJB **stateful session bean**.

**12) What are the transaction management supports provided by spring?**

Spring framework provides two type of transaction management supports:

1. **Programmatic Transaction Management**: should be used for few transaction operations.
2. **Declarative Transaction Management**: should be used for many transaction operations.

**13) What are classes for spring JDBC API?**

1. JdbcTemplate
2. SimpleJdbcTemplate
3. NamedParameterJdbcTemplate
4. SimpleJdbcInsert
5. SimpleJdbcCall.

### 14) What are the different features of Spring Framework?

Following are some of the major features of Spring Framework :

* **Lightweight:** Spring is lightweight when it comes to size and transparency.
* **Inversion of control (IOC):** The objects give their dependencies instead of creating or looking for dependent objects. This is called Inversion Of Control.
* **Aspect oriented Programming (AOP):** Aspect oriented programming in Spring supports cohesive development by separating application business logic from system services.
* **Container:**Spring Framework creates and manages the life cycle and configuration of the application objects.
* **MVC Framework:** Spring Framework’s MVC web application framework is highly configurable. Other frameworks can also be used easily instead of Spring MVC Framework.
* **Transaction Management:** Generic abstraction layer for transaction management is provided by the Spring Framework. Spring’s transaction support can be also used in container less environments.
* **JDBC Exception Handling:** The JDBC abstraction layer of the Spring offers an exception hierarchy, which simplifies the error handling strategy.

**15. What are the various ways of using Spring Framework?**

Spring Framework can be used in various ways. They are listed as follows:

1. As a Full-fledged Spring web application.
2. As a third-party web framework, using Spring Frameworks middle-tier.
3. For remote usage.
4. As Enterprise Java Bean which can wrap existing POJOs (Plain Old Java Objects).

### **16. What’s the difference between @Component, @Controller, @Repository & @Service annotations in Spring?**

**@Component:** This marks a java class as a bean. It is a generic stereotype for any Spring-managed component. The component-scanning mechanism of spring now can pick it up and pull it into the application context.

**@Controller:** This marks a class as a Spring Web MVC controller. Beans marked with it are automatically imported into the Dependency Injection container.

**@Service:** This annotation is a specialization of the component annotation. It doesn’t provide any additional behavior over the @Component annotation. You can use @Service over @Component in service-layer classes as it specifies intent in a better way.

**@Repository:** This annotation is a specialization of the @Component annotation with similar use and functionality. It provides additional benefits specifically for DAOs. It imports the DAOs into the DI container andmakes the unchecked exceptions eligible for translation into Spring DataAccessException.

### **17. What do you understand by @Required annotation?**

@Required is applied to bean property setter methods. This annotation simply indicates that the affected bean property must be populated at the configuration time with the help of an explicit property value in a bean definition or with autowiring. If the affected bean property has not been populated, the container will throw BeanInitializationException.

### **18. What do you understand by @Autowired annotation?**

The **@Autowired** annotation provides more accurate control over where and how autowiring should be done. This annotation is used to autowire bean on the setter methods, constructor, a property or methods with arbitrary names or multiple arguments. By default, it is a type driven injection.

### **19. What do you understand by @Qualifier annotation?**

When you create more than one bean of the same type and want to wire only one of them with a property you can use the **@Qualifier** annotation along with **@Autowired** to remove the ambiguity by specifying which exact bean should be wired.

**20.  What do you understand by @RequestMapping annotation?**

@RequestMapping annotation is used for mapping a particular HTTP request method to a specific class/ method in controller that will be handling the respective request. This annotation can be applied at both levels:

* **Class level**: Maps the URL of the request
* **Method level**: Maps the URL as well as HTTP request method.

### 21.**What are the difference between Spring AOP and AspectJ AOP?**

|  |  |
| --- | --- |
| **Spring AOP** | **AspectJ AOP** |
| Runtime weaving through proxy is done | Compile time weaving through AspectJ Java tools is done |
| It supports only method level PointCut | It suports field level Pointcuts |
| It is DTD based | It is schema based and Annotation configuration |

### 22**. What do you mean by Proxy in Spring Framework?**

An object which is created after applying advice to a target object is known as a Proxy. In case of client objects the target object and the proxy object are the same.

### **23. In Spring, what is Weaving?**

The process of linking an aspect with other application types or objects to create an advised object is called Weaving. In Spring AOP, weaving is performed at runtime.

**24. Name Some of the Design Patterns Used in the Spring Framework?**

* **Singleton Pattern** – singleton-scoped beans
* **Factory Pattern** – Bean Factory classes
* **Prototype Pattern** – prototype-scoped beans
* **Adapter Pattern** – Spring Web and Spring MVC
* **Proxy Pattern** – Spring Aspect-Oriented Programming support
* **Template Method Pattern** – *JdbcTemplate*, *HibernateTemplate*, etc.
* **Front Controller** – Spring MVC *DispatcherServlet*
* **Data Access Object** – Spring DAO support
* **Model View Controller**– Spring MVC

**25. What Is Spring JdbcTemplate Class and How to Use It?**

The Spring JDBC template is the primary API through which we can access database operations logic that we’re interested in:

* Creation and closing of connections
* Executing statements and stored procedure calls
* Iterating over the *ResultSet* and returning results

### **26. What Is Spring WebFlux?**

[Spring WebFlux](https://docs.spring.io/spring/docs/current/spring-framework-reference/web-reactive.html#webflux) is Spring's reactive-stack web framework, and it's an alternative to Spring MVC. In order to achieve this reactive model and be highly scalable, the entire stack is non-blocking.

### **27. What Is the Default Bean Scope in Spring Framework?**

By default, a Spring Bean is initialized as a singleton.

**28. Which Is the Best Way of Injecting Beans and Why?**

The recommended approach is to use constructor arguments for mandatory dependencies and setters for optional ones. This is because constructor injection allows injecting values to immutable fields and makes testing easier.

**29. What is Spring configuration file?**

Spring configuration file is an XML file. This file contains the classes information and describes how these classes are configured and introduced to each other

**30.What is Spring AOP?**

Aspect-oriented programming, or AOP, is a programming technique that allows programmers to modularize crosscutting concerns, or behavior that cuts across the typical divisions of responsibility, such as logging and transaction management. The core construct of AOP is the aspect, which encapsulates behaviors affecting multiple classes into reusable modules.

**Spring Boot**

### 1. What are the advantages of using Spring Boot?

The advantages of Spring Boot are listed below:

* Easy to understand and develop spring applications.
* Spring Boot is nothing but an existing framework with the addition of an embedded HTTP server and annotation configuration which makes it easier to understand and faster the process of development.
* Increases productivity and reduces development time.
* Minimum configuration.
* We don’t need to write any XML configuration, only a few annotations are required to do the configuration.

### 2. What are the Spring Boot key components?

Below are the four key components of spring-boot:

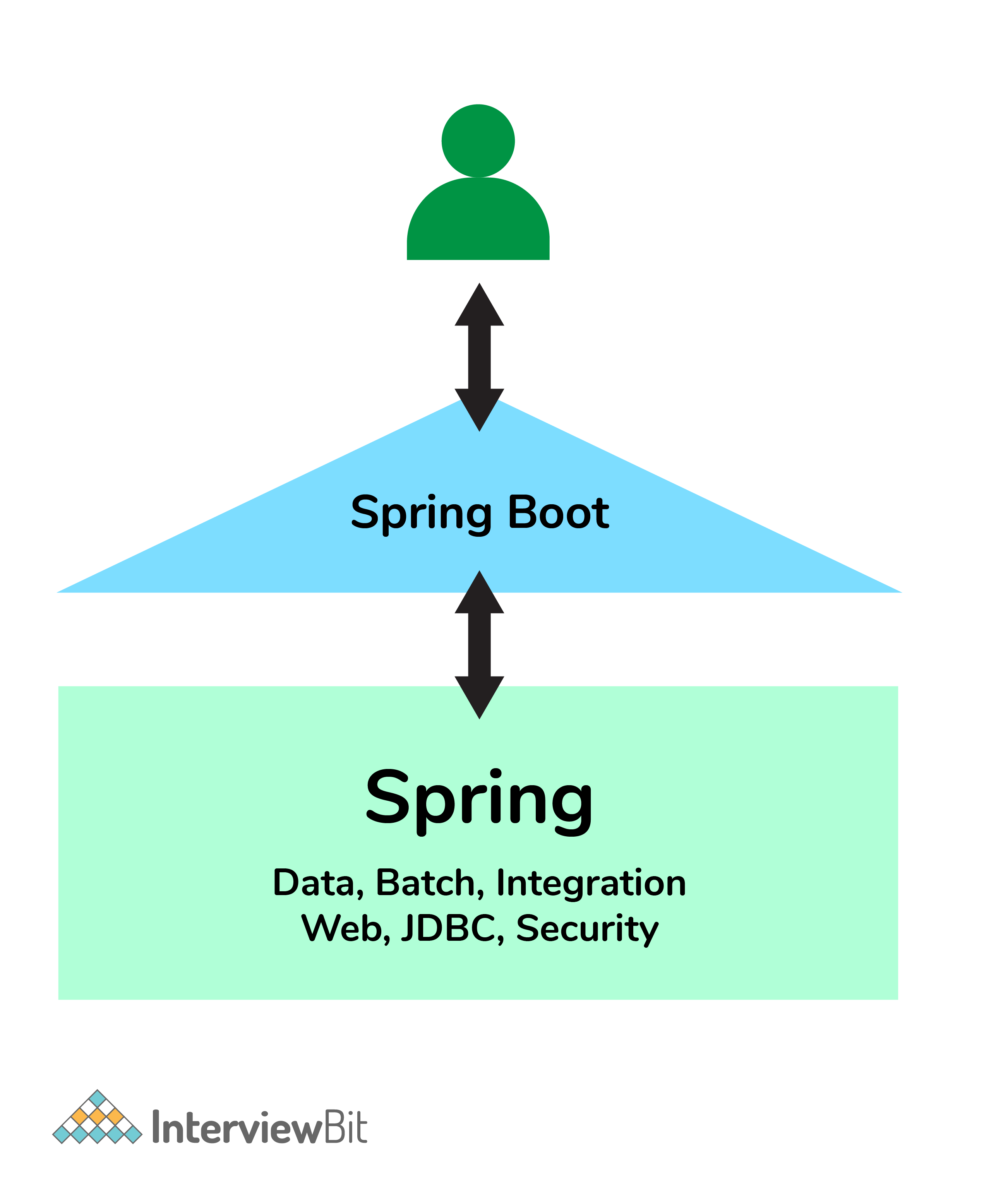
* Spring Boot auto-configuration.
* Spring Boot CLI.
* Spring Boot starter POMs.
* Spring Boot Actuators.

### 3. Why Spring Boot over Spring?

Below are some key points which spring boot offers but spring doesn’t:

* Starter POM.
* Version Management.
* Auto Configuration.
* Component Scanning.
* Embedded server.
* InMemory DB.
* Actuators

Spring Boot simplifies the spring feature for the user:

Spring vs Spring Boot

**You can download a PDF version of Spring Boot Interview Questions.**

[**Download PDF**](javascript:void(0))

### 4. What is the starter dependency of the Spring boot module?

Spring boot provides numbers of starter dependency, here are the most commonly used -

* Data JPA starter.
* Test Starter.
* Security starter.
* Web starter.
* Mail starter.
* Thymeleaf starter.

### 5. How does Spring Boot works?

Spring Boot automatically configures your application based on the dependencies you have added to the project by using annotation. The entry point of the spring boot application is the class that contains @SpringBootApplication annotation and the main method.

Spring Boot automatically scans all the components included in the project by using @ComponentScan annotation.

### 6. What does the @SpringBootApplication annotation do internally?

The @SpringBootApplication annotation is equivalent to using @Configuration, @EnableAutoConfiguration, and @ComponentScan with their default attributes. Spring Boot enables the developer to use a single annotation instead of using multiple. But, as we know, Spring provided loosely coupled features that we can use for each annotation as per our project needs.

### 7. What is the purpose of using @ComponentScan in the class files?

Spring Boot application scans all the beans and package declarations when the application initializes. You need to add the @ComponentScan annotation for your class file to scan your components added to your project.

### 8. How does a spring boot application get started?

Just like any other Java program, a Spring Boot application must have a main method. This method serves as an entry point, which invokes the SpringApplication#run method to bootstrap the application.

@SpringBootApplication

**public** **class** **MyApplication** {

**public** **static** **void** **main**(String[] args) {

SpringApplication.run(MyApplication.class);

// other statements

}

}

### 9. What are starter dependencies?

Spring boot starter is a maven template that contains a collection of all the relevant transitive dependencies that are needed to start a particular functionality.  
Like we need to import spring-boot-starter-web dependency for creating a web application.

<dependency>

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

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

</dependency>

### 10. What is Spring Initializer?

Spring Initializer is a web application that helps you to create an initial spring boot project structure and provides a maven or gradle file to build your code. It solves the problem of setting up a framework when you are starting a project from scratch.

### 11. What is Spring Boot CLI and what are its benefits?

Spring Boot CLI is a command-line interface that allows you to create a spring-based java application using Groovy.

Example: You don’t need to create getter and setter method or access modifier, return statement. If you use the JDBC template, it automatically loads for you.

### 13. What Are the Basic Annotations that Spring Boot Offers?

The primary annotations that Spring Boot offers reside in its org.springframework.boot.autoconfigure and its sub-packages. Here are a couple of basic ones:

@EnableAutoConfiguration – to make Spring Boot look for auto-configuration beans on its classpath and automatically apply them.

@SpringBootApplication – used to denote the main class of a Boot Application. This annotation combines @Configuration, @EnableAutoConfiguration, and @ComponentScan annotations with their default attributes.

### 14. What is Spring Boot dependency management?

Spring Boot dependency management is used to manage dependencies and configuration automatically without you specifying the version for any of that dependencies.

### 15. Can we create a non-web application in Spring Boot?

Yes, we can create a non-web application by removing the web dependencies from the classpath along with changing the way Spring Boot creates the application context.

### 16. Is it possible to change the port of the embedded Tomcat server in Spring Boot?

Yes, it is possible. By using the **server.port** in the **application.properties**.

### 17. What is the default port of tomcat in spring boot?

The default port of the tomcat server-id 8080. It can be changed by adding **sever.port** properties in the **application.property** file.

### 18. Can we override or replace the Embedded tomcat server in Spring Boot?

Yes, we can replace the Embedded Tomcat server with any server by using the Starter dependency in the **pom.xml** file. Like you can use spring-boot-starter-jetty as a dependency for using a jetty server in your project.

### 19. Can we disable the default web server in the Spring boot application?

Yes, we can use **application.properties** to configure the web application type i.e **spring.main.web-application-type=none.**

### 20. How to disable a specific auto-configuration class?

You can use exclude attribute of @EnableAutoConfiguration if you want auto-configuration not to apply to any specific class.

//use of exclude

@EnableAutoConfiguration(exclude={className})

### 21. Explain @RestController annotation in Sprint boot?

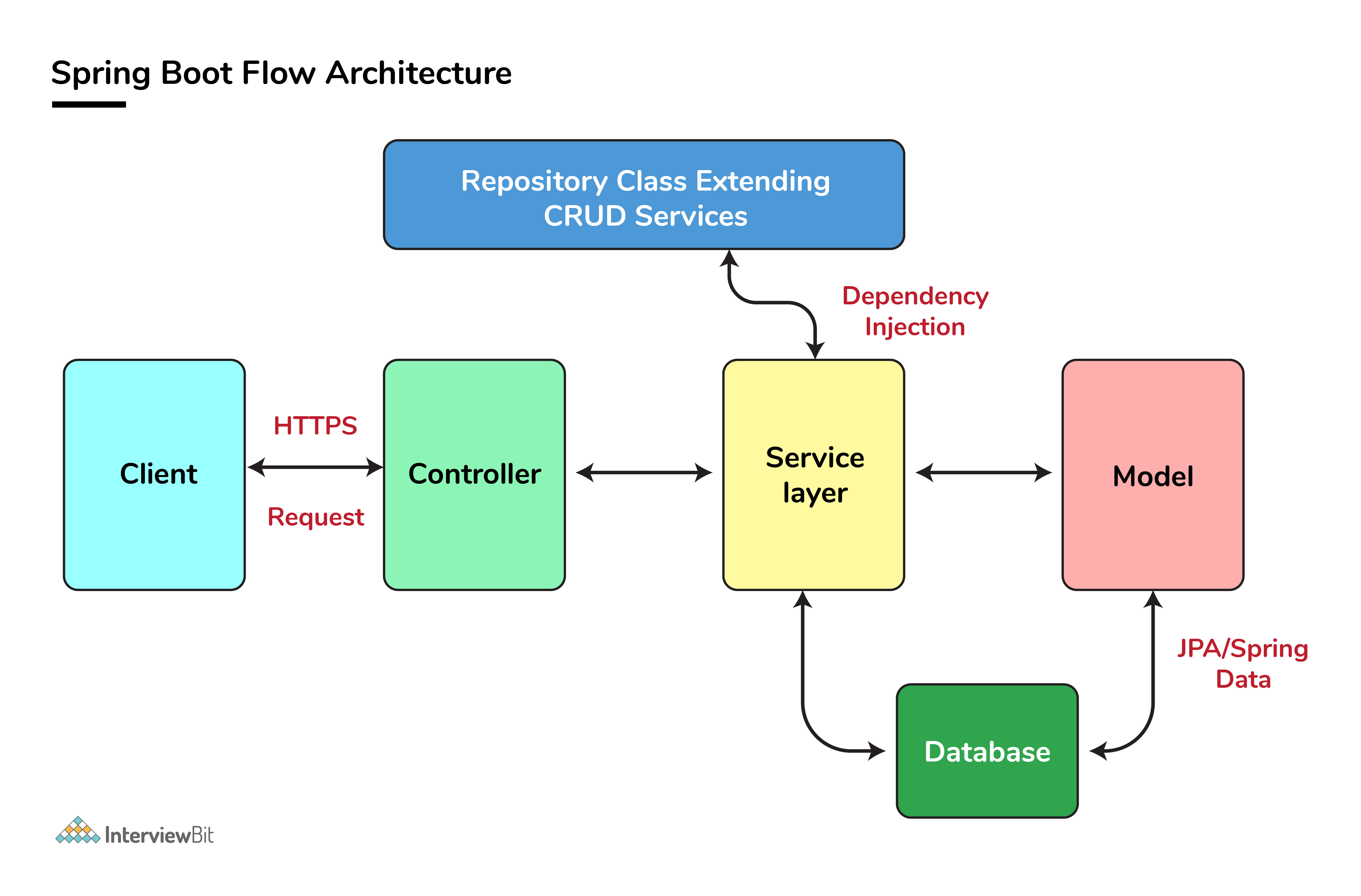
It is a combination of @Controller and @ResponseBody, used for creating a restful controller. It converts the response to JSON or XML. It ensures that data returned by each method will be written straight into the response body instead of returning a template.

### 22. What is the difference between @RestController and @Controller in Spring Boot?

@Controller Map of the model object to view or template and make it human readable but @RestController simply returns the object and object data is directly written in HTTP response as JSON or XML.

### 23. Describe the flow of HTTPS requests through the Spring Boot application?

On a high-level spring boot application follow the MVC pattern which is depicted in the below flow diagram.

Spring Boot Flow Architecture

### 24. 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 getMapping is only an extension of RequestMapping which helps you to improve on clarity on request.

### 25. What is the use of Profiles in spring boot?

While developing the application we deal with multiple environments such as dev, QA, Prod, and each environment requires a different configuration. For eg., we might be using an embedded H2 database for dev but for prod, we might have proprietary Oracle or DB2. Even if DBMS is the same across the environment, the URLs will be different.

To make this easy and clean, Spring has the provision of Profiles to keep the separate configuration of environments.

### 26. What is Spring Actuator? What are its advantages?

An actuator is an additional feature of Spring that helps you to monitor and manage your application when you push it to production. These actuators include auditing, health, CPU usage, HTTP hits, and metric gathering, and many more that are automatically applied to your application.

### 27. How to enable Actuator in Spring boot application?

To enable the spring actuator feature, we need to add the dependency of “spring-boot-starter-actuator” in pom.xml.

<dependency>

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

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

</dependency>

### 28. What are the actuator-provided endpoints used for monitoring the Spring boot application?

Actuators provide below pre-defined endpoints to monitor our application -

* Health
* Info
* Beans
* Mappings
* Configprops
* Httptrace
* Heapdump
* Threaddump
* Shutdown

### 29. How to get the list of all the beans in your Spring boot application?

Spring Boot actuator “/Beans” is used to get the list of all the spring beans in your application.

### 30. How to check the environment properties in your Spring boot application?

Spring Boot actuator “/env” returns the list of all the environment properties of running the spring boot application.

### **1.  What is Spring Boot?**

Spring Boot is called a microservice framework that is built on top of the spring framework. This can help developers to focus more on convention rather than configuration.

1. The main aim of Spring boot is to give you a production-ready application. So, the moment you create a spring-boot project, it is runnable and can be executed/deployed on the server.
2. It comes with features like autoconfiguration, auto dependency resolution, embedded servers, security, health checks which enhances the productivity of a developer.

### **2. How to create Spring Boot project in eclipse?**

One of the ways to create a spring boot project in eclipse is by using **Spring Initializer.**

You can go to the official website of spring and add details such as version, select maven or Gradle project, add your groupId, artifactId, select your required dependencies and then click on CREATE PROJECT.

Once the project is created, you can download it and extract and import it in your eclipse or STS.

And see your project is ready! To Install Spring Boot in Eclipse – Go to Eclipse IDE, click on “Help”->then go to Eclipse marketplace->and type Spring IDE and click on the finish button.

### **3. How to deploy spring boot application in tomcat?**

Whenever you will create your [spring boot application](https://www.mygreatlearning.com/academy/learn-for-free/courses/dockerize-spring-boot-application/?gl_blog_id=25325) and run it, Spring boot will automatically detect the embedded tomcat server and deploy your application on tomcat.  
After successful execution of your application, you will be able to launch your rest endpoints and get a response.

### **4. What is the difference between Spring and Spring Boot?**

Difference between Spring and Spring boot are as follows:

**Spring –**

1. Is a dependency injection framework.
2. It is basically used to manage the life cycle of [java classes](https://www.mygreatlearning.com/blog/java-tutorial-for-beginners/?gl_blog_id=25325) (beans). It consists of a lot of boilerplate configuration.
3. Uses XML based configuration.
4. It takes time to have a spring application up and running and it’s mainly because of boilerplate code.

**Spring boot-**

1. It is a suite of pre- configured frameworks and technologies which helps to remove boilerplate configuration.
2. Uses annotations.
3. It is used to create a production-ready code.

### **5. What is actuator in spring boot?**

An actuator is one of the best parts of spring boot which consists of production-ready features to help you monitor and manage your application.

With the help of an actuator, you can monitor what is happening inside the running application.  
Actuator dependency figures out the metrics and makes them available as a new endpoint in your application and retrieves all required information from the web. You can identify beans, the health status of your application, CPU usage, and many more with the actuator. By using @Endpoint annotation, you can create a custom endpoint.

### **6. How to change port in spring boot?**

The default port number to start your SpringBoot application is **8080**.

Just to change the port number, you need to add **server.port=8084**c(your port number) property in your application.properties file and start your application.

### **7. How to create war file in spring boot?**

To create a war file in spring boot you need to define your packaging file **as war** in your pom.xml(if it is maven project).

Then just do **maven clean** **and install** so that your application will start building. Once the build is successful, just go into your Target folder and you can see .war file generated for your application.

### **8. What is JPA in spring boot?**

[JPA](https://www.mygreatlearning.com/jpa/free-courses/?gl_blog_id=25325) is basically **Java Persistence API**. It’s a specification that lets you do ORM when you are connecting to a [relational database](https://www.mygreatlearning.com/blog/what-is-rdbms/?gl_blog_id=25325) which is Object-Relational Mapping.

So, when you need to connect from your java application to relational database, you need to be able to use something like JDBC and run [SQL queries](https://www.mygreatlearning.com/blog/sql-tutorial-for-beginners/?gl_blog_id=25325) and then you get the results and convert them into Object instances.

ORM lets you map your entity classes in your [SQL](https://www.mygreatlearning.com/academy/learn-for-free/courses/introduction-to-sql/?gl_blog_id=25325) tables so that when you connect to the database , you don’t need to do query yourself, it’s the framework that handles it for you.

And JPA is a way to use ORM, it’s an API which lets you configure your entity classes and give it to a framework so that the framework does the rest.

### **9. How to save image in database using spring boot?**

1. First configure [mysql](https://www.mygreatlearning.com/academy/learn-for-free/courses/my-sql-basics/?gl_blog_id=25325" \t "_blank) in your spring boot application.
2. Then you can map your entities with your db tables using JPA.
3. And with the help of save() method in JPA you can directly insert your data into your database

@RestController

@RequestMapping("/greatleasrning")

public class Controller {

@Autowired

private final GreatLearningRepository greatLearningRepository;

public Controller(GreatLearningRepository greatLearningRepository) {

}

In above case, your data which may be in [JSON format](https://www.mygreatlearning.com/json-format/free-courses/?gl_blog_id=25325) can be inserted successfully into database.

@RequestMapping(method = RequestMethod.POST)

ResponseEntity<?> insert(@RequestBody Course course) {

greatLearningRepository.save(course);

return ResponseEntity.accepted().build();

}

}

### **10. What is auto configuration in spring boot?**

AutoConfiguration is a process by which Spring Boot automatically configures all the infrastructural beans. It declares the built-in beans/objects of the spring specific module such as JPA, spring security and so on based on the dependencies present in your applications class path.

**For example:** If we make use of Spring JDBC, the spring boot autoconfiguration feature automatically registers the DataSource and JDBCTemplete bean.  
This entire process of automatically declaring the framework specific bean without the need of writing the [xml code](https://www.mygreatlearning.com/xml/free-courses/?gl_blog_id=25325) or java config code explicity  is called Autoconfiguration which is done by springBoot with the help of an annotation called **@EnableAutoconfiguration** alternatively **@SpringBootApplication**.

### **11. How to resolve whitelabel error page in spring boot application?**

This is quite common error in spring boot application which says 404(page not found).

We can mostly resolve this in 3 ways:

1. **Custom Error Controller**– where you will be implementing ErrorController  interface which is provided by SpringFramework and then overriding its getErrorPath() so that you can return a custom path whenever such type of error is occurred.
2. **By Displaying Custom error page**– All you have to do is create an error.html page and place it into the src/main/resources/templates path. The BasicErrorController of of springboot will automatically pick this file by default.
3. **By disabling the whitelabel error page**– this is the easiest way where all you need to do is server.error.whitelabel.enabled property to false in the application.properties file to disable the whitelabel error page.

### **12. How to fetch data from database in spring boot?**

You can use the following steps to connect your application with [MySQL database](https://www.mygreatlearning.com/blog/mysql-tutorial/?gl_blog_id=25325).  
1. First create a database in MySQL with create DATABASE student;  
2. Now, create a table inside this DB:  
CREATE TABLE student(studentid INT PRIMARY KEY NOT NULL AUTO\_INCREMENT, studentname VARCHAR(255));   
3. Create a SpringBoot application and add [JDBC](https://www.mygreatlearning.com/blog/jdbc-tutorial/?gl_blog_id=25325), MySQL and web dependencies.  
4. In application.properties, you need to configure the database.

spring.datasource.url=jdbc:mysql://localhost:3306/studentDetails

spring.datasource.username=system123

spring.datasource.password=system123

spring.jpa.hibernate.ddl-auto=create-drop

5. In your controller class, you need to handle the requests.

package com.student;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class JdbcController {

@Autowired

JdbcTemplate jdbc;

@RequestMapping("/save")

public String index(){

jdbc.execute("insert into student (name)values(GreatLearnings)");

return "Data Entry Successful";

}

}

6. Run the application and check the entry in your Database.

### **13. How to use logger in spring boot?**

There are many logging options available in SpringBoot. Some of them are mentioned below:

* Using log4j2:

import org.apache.logging.log4j.Logger;

import org.apache.logging.log4j.LogManager;

// [...]

Logger logger = LogManager.getLogger(LoggingController.class);

* Using Lombok:

All you need to do is add a dependency called **org.projectlombok**in your pom.xml as shown below:

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<version>1.18.4</version>

<scope>provided</scope>

</dependency>

Now you can create a loggingController and add the **@Slf4j**annotation to it. Here we would not create any logger instances.

@RestController

@Slf4j

public class LoggingController {

@RequestMapping("/logging")

public String index() {

log.trace("A TRACE Message");

log.debug("A DEBUG Message");

log.info("An INFO Message");

log.warn("A WARN Message");

log.error("An ERROR Message");

return "Here are your logs!”;

}

}

So, there are many such ways in spring boot to use logger.

### **14. What is bootstrapping in spring boot?**

One of the way to [bootstrap](https://www.mygreatlearning.com/academy/learn-for-free/courses/intro-to-bootstrap/?gl_blog_id=25325) your spring boot application is using Spring Initializer.  
you can go to the official website of spring  and select your version, and add you groupID, artifactId and all the required dependencies.

And then you can create your restEndpoints and build and run your project.  
There you go, you have bootstrapped your spring boot application.

### **15. How to create jar file in spring boot?**

To create a jar file in spring boot you need to define your packaging file as **jar** in your pom.xml(if it is maven project).

Then just do maven build with specifying **goals as package** so that your application will start building.

Once the build is successful, just go into your Target folder and you can see .jar file generated for you application.

### **16. What is dependency injection in spring boot?**

[Dependency injection](https://www.mygreatlearning.com/spring/tutorials/spring-dependency-injection/?gl_blog_id=25325) is a way through which the Spring container injects one object into another. This helps for loose coupling of components.

**For example:** if class student uses functionality of department class, then we say student class has dependency of Department class. Now we need to create object of class Department in your student class so that it can directly use functionalities of department class is called dependency injection.

### **17. How to store image in MongoDB using spring boot?**

One of the way for storing image in [MongoDB](https://www.mygreatlearning.com/academy/learn-for-free/courses/mongodb-tutorial/?gl_blog_id=25325) is by using Spring Content. And also you should have the below dependency in your pom.xml.

<dependency>

<groupId>com.github.paulcwarren</groupId>

<artifactId>spring-content-mongo-boot-starter</artifactId>

<version>0.0.10</version>

</dependency>

You should have a GridFsTemplate bean in your applicationContext.

@Configuration

public class Config

@Bean

public GridFsTemplate gridFsTemplate() throws Exception {

return new GridFsTemplate(mongoDbFactory(), mappingMongoConverter());

}

...

Now add attributes so that your content will be associated to your entity.

@ContentId

private String contentId;

@ContentLength

private long contentLength = 0L;

@MimeType

private String mimeType = "text/plain";

And last but not the least, add a store interface.

@StoreRestResource(path="greatlearningImages")

public interface GreatLearningImageStore extends ContentStore<Candidate, String> {

}

That’s all you have to do to store your images in mongoDb using Springboot.

### **18. How to configure hibernate in spring boot?**

The important and required dependency to configure hibernate is:

1. **spring-boot-starter-data-jpa**
2. **h2** (you can also use any other database)

Now, provide all the database connection properties in application.properties file of your application in order to connect your JPA code with the database.

Here we will configure H2 database in application.properties file:

spring.datasource.url=jdbc:h2:file:~/test

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=test

spring.datasource.password=test

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

Adding the above properties in your application.properties file will help you to interact with your database using JPA repository interface.

### **19. Mention the advantages of Spring Boot.**

**Advantages of Spring Boot –**

1. It allows convention over configuration hence you can fully avoid XML configuration.
2. SpringBoot reduces lots of development time and helps to increase productivity.
3. Helps to reduce a lot of boilerplate code in your application.
4. It comes with embedded HTTP servers like tomcat, Jetty, etc to develop and test your applications.
5. It also provides CLI (Command Line Interface) tool which helps you  to develop and test your application from CMD.

### **20. Explain what is thyme leaf and how to use thymeleaf?**

Thymeleaf is a server-side java template engine which helps processing and creating [HTML](https://www.mygreatlearning.com/academy/learn-for-free/courses/html-tutorial/?gl_blog_id=25325), [XML](https://www.mygreatlearning.com/blog/xml-tutorial/?gl_blog_id=25325), [JavaScript](https://www.mygreatlearning.com/academy/learn-for-free/courses/intro-to-javascript/?gl_blog_id=25325) , [CSS](https://www.mygreatlearning.com/academy/learn-for-free/courses/css-tutorial/?gl_blog_id=25325), and text. Whenever the dependency in pom.xml (in case of  maven project) is find, springboot automatically configures Thymeleaf to serve dynamic web content.

**Dependency: spring-boot-starter-thymeleaf**

We can place the thyme leaf templates which are just the HTML files in **src/main/resources/templates/** folder so that spring boot can pick those files and renders whenever required.

Thymeleaf will parse the index.html and will replace the dynamic values with its actual value that is been passed from the controller class.  
That’s it, once you run your Spring Boot application and your message will be rendered in web browsers.

### **21. What is the need for Spring Boot DevTools?**

This is one of the amazing features provided by Spring Boot, where it restarts the spring boot application whenever any changes are being made in the code.

 Here, you don’t need to right-click on the project and run your application again and again. Spring Boot dev tools does this for you with every code change.  
**Dependency to be added is: spring-boot-devtools**

The main focus of this module is to improve the development time while working on Spring Boot applications.

### **22. Can we change the port of the embedded Tomcat server in Spring boot?**

Yes, you can change the port of embedded Tomcat server in Spring boot by adding the following property in your **application.properties** file.

server.port=8084

The default port number of the tomcat server to run the spring boot application is 8080, which is further possible to change it.

So we can change the port of tomcat following ways given below:-

* Using application.properties
* Using application.yml
* Using EmbeddedServletContainerCustomizer interface.
* Using WebServerFactoryCustomizer interface.
* Using Command-Line Parameter.

### **23. Mention the steps to connect Spring Boot application to a database using JDBC**

Below are the steps to connect your Spring Boot application to a database using JDBC:

Before that, you need to add required dependencies that are provided by spring-boot to connect your application with JDBC.

**Step 1**: First create a database in MySQL with create DATABASE student;

**Step 2**:  Now, create a table inside this DB:  
CREATE TABLE student(studentid INT PRIMARY KEY NOT NULL AUTO\_INCREMENT,

studentname VARCHAR(255));

**Step 3**: Create a springBoot and add JDBC,mysql and web dependencies.  
**Step 4**: In application.properties, you need to configure the database.

spring.datasource.url=jdbc:mysql://localhost:3306/studentDetails

spring.datasource.username=system123

spring.datasource.password=system123

spring.jpa.hibernate.ddl-auto=create-drop

**Step 5**: In your controller class, you need to handle the requests.

package com.student;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class JdbcController {

@Autowired

JdbcTemplate jdbc;

@RequestMapping("/save")

public String index(){

jdbc.execute("insert into student

(name)values(GreatLearnings)");

return "Data Entry Successful";

}

}

**Step 6**: Run the application and check the entry in your Database.

**Step 7**: You can also go ahead and open the URL and you will see “Data Entry Successful” as your output.

### **24. What are the @RequestMapping and @RestController annotation in Spring Boot used for?**

The **@RequestMapping** annotation can be used at class-level or method level in your controller class.

The global request path that needs to be mapped on a controller class can be done by using **@RequestMapping** at class-level. If you need to map a particular request specifically to some method level.

Below is a simple example to refer to:

@RestController

@RequestMapping("/greatLearning")

public class GreatLearningController {

@RequestMapping("/")

String greatLearning(){

return "Hello from greatLearning ";

}

@RequestMapping("/welcome")

String welcome(){

return "Welcome from GreatLearning";

}

}

The **@RestController** annotation is used at the class level.

You can use @RestController when you need to use that class as a request handler class.All the requests can be mapped and handled in this class.

**@RestController** itself consists **@Controller** and **@ResponseBody** which helps us to remove the need of annotating every method with @ResponseBody annotation.

**Below is a simple example to refer to for use of @RestController annotation:**

@RestController

@RequestMapping(“bank-details”)

public class DemoRestController{

@GetMapping(“/{id}”,produces =”application/json”)

public Bank getBankDetails(@PathVariable int id){

return findBankDetailsById();

}

}

Here, @ResponseBody is not required as the class is annotated with @RestController.

### **25. What do you understand  by auto-configuration in Spring Boot and how to disable the auto-configuration?**

AutoConfiguration is a process by which Spring Boot automatically configures all the infrastructural beans. It declares the built-in beans/objects of the spring-specific module such as JPA, spring-security, and so on based on the dependencies present in your application’s classpath.  
**For example:** If we make use of Spring JDBC, the spring boot autoconfiguration feature automatically registers the DataSource and JDBCTemplete bean.  
This entire process of automatically declaring the framework-specific bean without the need of writing the XML code or java-config code explicitly  is called Autoconfiguration which is done by spring-boot with the help of an annotation called **@EnableAutoconfiguration** alternatively **@SpringBootApplication.**

1. You can exclude the attribute of @EnableAutoConfiguration where you don’t want it to be configured implicity in order to disable the spring boot’s auto-configuration feature.

2. Another way of disabling auto-configuration is by using the property file:

**For example:**

spring.autoconfigure.exclude=

org.springframework.boot.autoconfigure.mongo.MongoAutoConfiguration,

org.springframework.boot.autoconfigure.data.MongoDataConfiguration,

In the above example, we have disabled the autoconfiguration of MongoDB.

### **26. Can you give an example for ReadOnly as true in Transaction management?**

Yes, example for ReadOnly as true in Transaction Management is:

Suppose you have a scenario where you have to read data from your database like if you have a STUDENT database and you have to read the student details such as studentID, and studentName.

 So in such scenarios, you will have to set read-only on the transaction.

### **27. Mention the advantages of the YAML file than Properties file and the different ways to load**

YAML file in Spring boot.

YAML gives you more clarity and is very friendly to humans. It also supports **maps, lists, and other scalar types.**

YAML comes with hierarchical nature which helps in avoiding repetition as well as indentations.

If we have different deployment profiles such as  development, testing, or production and we may have different configurations for each environment, so instead of creating new files for each environment we can place them in a single YAML file.  
But in the case of the properties file, you cannot do that.

**For example:**

spring:

profiles:

active:

-test

---

spring:

profiles:

active:

-prod

---

spring:

profiles:

active:

-development

### **28. What do you understand by Spring Data REST?**

By using Spring Data Rest, you have access to all the RESTful resources that revolves around Spring Data repositories.

Refer the below example:

@RepositoryRestResource(collectionResourceRel = "greatlearning", path = "sample")

public interface GreatLearningRepo extends CustomerRepository< greatlearning, Long> {

}

Now you can use the POST method in the below manner:

{

“Name”:”GreatLearning”

}

And you will get response as follow:

{

“Name”:”GreatLearning”

}

\_\_\_\_\_\_\_\_\_\_

{

"name": "Hello greatlearning "

"\_links": {

"self": {

"href": "<a href="http://localhost:8080/sample/1">http://localhost:8080/ greatlearning /1</a>"

},

" greatlearning ": {

“href": "<a href="http://localhost:8080/sample/1">http://localhost:8080/ greatlearning /1</a>"

}

}

In the above, you can see the response of the newly created resource.

### **29. What do you think is the need for Profiles?**

The application has different stages-such as the development stage, testing stage, production stage and may have different configurations based on the environments.

With the help of spring boot, you can place profile-specific properties in different files such as

**application-{profile}.properties**

In the above, you can replace the profile with whatever environment you need, for example, if it is a development profile, then **application-development.properties** file will have development specific configurations in it.

So, in order to have profile-specific configurations/properties, you need to specify an active profile.

### **30. How to insert data in mysql using spring boot?**

First configure mysql in your spring boot application.

Then you can map your entities with your db tables using JPA.

And with the help of save() method in JPA, you can directly insert your data into your database.

@RestController

@RequestMapping("/greatleasrning")

public class Controller {

@Autowired

private final GreatLearningRepository greatLearningRepository;

public Controller(GreatLearningRepository greatLearningRepository) {

this. greatLearningRepository = greatLearningRepository;

}

In the above case, your data which may be in JSON format can be inserted successfully into the database.

@RequestMapping(method = RequestMethod.POST)

ResponseEntity<?> insert(@RequestBody Course course) {

greatLearningRepository.save(course);

return ResponseEntity.accepted().build();

}

}

### **31. How to create a login page in spring boot?**

You can create a simple and default login page in spring boot, you can make use of Spring security. Spring security secures all HTTP endpoints where the user has to login into the default HTTP form provided by spring.

We need to add **spring-boot-starter-security** dependency in your pom.xml or build.gradle and a default username and password can be generated with which you can log in.

### **32. What is the main class in spring boot?**

Usually in java applications, a class that has a main method in it is considered as a main class. Similarly, in spring boot applications main class is the class which has a public static void main() method and which starts up the SpringApplicationContext.

### **33. How to use crud repository in spring boot?**

In order to use crud repository in spring boot, all you have to do is extend the crud repository which in turn extends the Repository interface as a result you will not need to implement your own methods.

Create a simple spring boot application which includes below dependency:  
**spring-boot-starter-data-jpa**, **spring-boot-starter-data-rest**

And extend your repository interface as shown below**:**

package com.greatlearning;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource

public interface GreatLearning extends CrudRepository<Candidate, Long>

{

public List<Candidate> findById(long id);

//@Query("select s from Candidate s where s.age <= ?")

public List<Candidate> findByAgeLessThanEqual (long age);

}

### **34. How to run spring-boot jar from the command line?**

In order to run spring boot jar from the command line, you need to update you pom.xml(or build.gradle) of your project with the maven plugin.

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

Now, Build your application and package it into the single executable jar. Once the jar is built you can run it through the command prompt  using the below query:

java -jar target/myDemoService-0.0.1-SNAPSHOT.jar

And you have your application running.

### **35. What is Spring Boot CLI and how to execute the Spring Boot project using boot CLI?**

Spring Boot CLI is nothing but a command-line tool which is provided by Spring so that you can develop your applications quicker and faster.

To execute your spring boot project using CLI, you need first to download CLI from Spring’s official website and extract those files. You may see a bin folder present in the Spring setup which is used to execute your spring boot application.

As Spring boot CLI allows you to execute groovy files, you can create one and open it in the terminal.  
And then execute  **./spring run filename.groovy;**

### **36. what is the rest controller in spring boot?**

The **@RestController** annotation is used at the class level.

You can use @RestController when you need to use that class as a request handler class.All the requests can be mapped and handled in this class.

**@RestController** itself consists **@Controller** and **@ResponseBody** which helps us to remove the need of annotating every method with @ResponseBody annotation.

**Below is a simple example to refer to for use of @RestController annotation:**

@RestController

@RequestMapping(“bank-details”)

public class DemoRestController{

@GetMapping(“/{id}”,produces =”application/json”)

public Bank getBankDetails(@PathVariable int id){

return findBankDetailsById();

}

}

Here, @ResponseBody is not required as the class is annotated with @RestController.

### **37. How to handle 404 error in spring boot?**

Consider a scenario, where there are no stockDetails in the DB and still, whenever you hit the GET method you get 200(OK) even though the resource is not found which is not expected. Instead of 200, you should get 404 error.  
So to handle this, you need to create an exception, in the above scenario “StockNotFoundException”.

GetMapping("/stocks/{number}")

public Stock retriveStock(@PathVariable int number)

{

Stock stock = service.findOne(number);

if(Stock ==null)

//runtime exception

throw new StockNotFoundException("number: "+ number);

return stock;

}

Now, create a Constructor from [Superclass](https://www.mygreatlearning.com/blog/java-super-keyword-and-wrapper-class/?gl_blog_id=25325).

Right-click on the file -> Go to Source ->And generate constuctors from superclass-> and check the RuntimeException(String)-> and generate.

And add an annotation called **@ResponseStatus** which will give you 404 (not found) error.

package com.greatlearning;

import org.springframework.http.HttpStatus;

import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(HttpStatus.NOT\_FOUND)

public class StockNotFoundException extends RuntimeException

{

public StockNotFoundException(String message)

{

super(message);

}

}

Now, you can hit the same URL again and there you go, you get a 404 error when a resource is not found.

### **38. Which is the spring boot latest version?**

The latest version of spring boot is**2.6.0**. It came out with a lot of dependency upgrades, java 15 support and much more.

Yes, now as you are brushed up with spring boot interview questions and answers. We have also tried to cover all the springboot interview questions for experienced professionals. Hope you can easily crack the spring boot interview now!

Please feel free to comment below if you have any queries related to the above questions or answers. Also, do comment if you find any other questions that you think must be included in the above list of questions.

## **Spring Boot Interview Questions for Experienced**

As an experienced professional, you should be prepared to answer questions about your experience with Spring Boot. In this section, we will share some of the most common Spring Boot interview questions for experienced professionals.

### **39. How to check the environment properties in your Spring boot application?**

If we need to set the different target environments, Spring Boot has a built-in mechanism.

One can simply define an application environment.properties file in the src/main/resources directory and then set a Spring profile with the same environment name.

For example, if we define a “production” environment, that means we’ll have to define a production profile and then application-production.properties.

This environment file will be loaded and will take precedence over the default property file. You should note that the default file will still be loaded. It’s just that when there is a property collision, the environment-specific property file takes precedence.

### **40. Where do we define properties in the Spring Boot application?**

**Command Line Properties**

Command-line properties are converted into Spring Boot Environment properties by the spring boot application.

Command-line properties have more precedence over the other property sources.

Spring Boot uses the 8080 port number, by default, to start the Tomcat. Let us see how one can change the port number by using command-line properties.

c:\demo\target>java -jar demo-0.0.1-SNAPSHOT.jar --server.port=9090

**Properties File**

Properties files are used to keep one or more properties in a single file to run the application in a different environment. Properties are kept in the application.properties file under the classpath in a typical spring boot application. The location of the application.properties file is at src/main/resources directory. The code of application.properties file is as below:

sever.port=9090

spring.application.name = demoservice

**YAML File**

Spring Boot also supports YAML-based properties configurations to run the application. The user can use,  application.yml file instead of the application.properties file. The YAML file is kept inside the classpath. The sample application.yml file is given below −

spring:

application:

name: demoservice

server:

port: 9090

**Externalized Properties**

The user can keep properties in different locations or paths instead of keeping the properties file under classpath. While running the JAR file, the user can specify the properties file path. The application developer can use the following command to specify the location of the properties file while running the JAR −

-Dspring.config.location = C:\application.properties

-C:\demo\target>java -jar -Dspring.config.location=C:\application.properties demo-0.0.1-SNAPSHOT.jar