**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

=================================================================================

**Spring Boot**

[**Sprint boot**](https://www.interviewbit.com/java-interview-questions/) is a Java-based spring framework used for Rapid Application Development (to build stand-alone microservices). It has extra support of auto-configuration and embedded application server like tomcat, jetty, etc.

## Features of Spring Boot that make it different?

* Creates stand-alone spring application with minimal configuration needed.
* It has embedded tomcat, jetty which makes it just code and run the application.
* Provide production-ready features such as metrics, health checks, and externalized configuration.
* Absolutely no requirement for XML configuration.

**1. What are the advantages of using Spring Boot? Spring Boot Features**

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:

### 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?

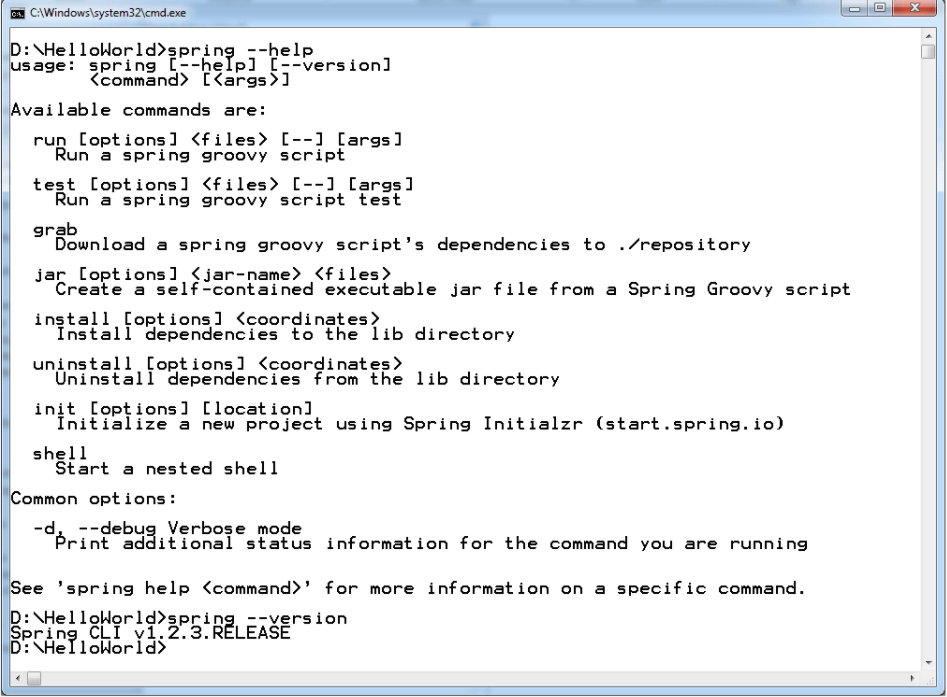
Spring Boot CLI is a command-line interface that allows you to create a spring-based java application using Groovy.

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.

### 12. What are the most common Spring Boot CLI commands?

-run, -test, -grap, -jar, -war, -install, -uninstall, --init, -shell, -help.

To check the description, run spring --help from the terminal.

Spring Boot CLI Commands

## Advanced Spring Boot Questions

### 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.

### 31. How to enable debugging log in the spring boot application?

Debugging logs can be enabled in three ways -

* We can start the application with --debug switch.
* We can set the logging.level.root=debug property in application.property file.
* We can set the logging level of the root logger to debug in the supplied logging configuration file.

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

You can define both application and Spring boot-related properties into a file called application.properties. You can create this file manually or use Spring Initializer to create this file. You don’t need to do any special configuration to instruct Spring Boot to load this file, If it exists in classpath then spring boot automatically loads it and configure itself and the application code accordingly.

### 33. What is dependency Injection?

The process of injecting dependent bean objects into target bean objects is called dependency injection.

* Setter Injection: The IOC container will inject the dependent bean object into the target bean object by calling the setter method.
* Constructor Injection: The IOC container will inject the dependent bean object into the target bean object by calling the target bean constructor.
* Field Injection: The IOC container will inject the dependent bean object into the target bean object by Reflection API.

### 34. What is an IOC container?

IoC Container is a framework for implementing automatic dependency injection. It manages object creation and its life-time and also injects dependencies into the class