1. What is Tight Coupling?

When a class (Class A) is dependent on another class’s object (Class B), then we say class A is "tightly" Coupled with Class B. Spring helps us to create classes in a way that Tight Coupling can be removed and Loose Coupling can be done.

2. What is Loose Coupling?

Loose Coupling removes the dependency of an object (class B) on a class (class A). Loose Coupling is approached by creating an interface and a setter & getter method, or by using a constructor which takes the interface object.

3. What are Beans in Spring?

When a class is annotated or decorated using the @Component, such a class is called a Bean in Spring. Beans are maintained by Application Context.

4. Explain Bean creation process?

The process of Bean creation has the following phases

(I) Starts with a class (c1) which has the annotation @Component.

(II) Checks if the component annotated class (c1) is dependent.

(III) If yes, then Spring will create a bean for that class (c2) too.

(IV) A connection or auto wiring will occur between the two classes (c1 and c2) using @Autowired annotation and also through the constructor (c2) or the default case set Class Function (interface the Interface).

5. What is the importance of the annotation @Primary?

This annotation is used on a class that needs to be taken by spring on a primary basis. For instance, if Class X is @Component annotated and is dependent on both Class1 and Class2 (both @Component annotated) then the compiler would report an error. To show the primary class between Class1 and Class2 we use @Primary.

6. What is Dependency Injection?

Dependency Injection is where Spring searches for beans; once the appropriate bean is found, it auto wired the bean to the dependent class. Dependency Injection is the process where Spring framework looks for the beans and identifies the dependencies, and creates the instances of beans and auto wired them.

7. Explain Inversion of Control (IOC)?

In Tight Coupling the dependent class takes the responsibility of creating its dependency. Whereas, in Loose Coupling, we use @Autowired annotation over the dependency class (or reference) and Spring takes control of creating the instance and injects the dependency.

8. What are the roles of an IOC (Inversion of Control) Container?

IOC Container does the following things-

(I) Find Beans

(II) Identify their dependencies and wire the dependencies

(III) Manage Lifecycle of the Bean (creation, processing, and destruction)

9. What is Application Context?

It is an advanced version of IOC Container. It provides all the functionalities of Bean Factory and also provides things like AOP, Internationalization capabilities, web application context (request, session, etc).

10. Explain the process of creating an Application Context in Spring?

The Application Context can be defined in two ways (I) using XML, (II) using @Configuration. Once the configuration is done in any of the ways defined above, the Application Context is created using new [Class Path Xml Application Context]. The looks for the XML files, using this is one of the two ways. The other way is to use [Annotation Config Application Context].

11. Explain Component Scan?

Component Scan is one method of asking Spring to detect Spring-managed components, the input for this search is the packages. Two methods are available to define a Component Scan-

(I) Java Configuration; wherein, we use the @Component annotation to which we specify all the packages, for which Spring does the search.

(II) XML Configuration- we use

<context:component-scan base package=”com.demo.compscanex”/>

12. How do you perform the same (above question) in Spring Boot?

In Spring boot the annotation used to perform the scan is @SpringBootApplication. This annotation on a class would automatically initiate the component scan on the package where they are in.

13. Differentiate @Component, @Repository and @Service and @Controller?

typically a web application is developed in layers like the controller (which is the initial point of client communication), business (where the actual code or logic of the application is written) and DAO (where the database connections and interaction happens). In such an architecture web application, @Component can be used in any of the layers. Whereas, the @Controller is used in the controller/web layer. @Service is used in the business layer and @Repository is used in the DAO layer.

14. List out the different scopes of Bean?

(I) Singleton: throughout the spring context only one instance is created.

(II) Prototype: a new bean is created whenever requested.

(III) Request: Every HTTP Request creates a bean.

(IV) Session: A bean for every HTTP Session.

15. List out the types of Dependency Injection?

The types of Dependency Injection-

(I) Setter Injection and (II) Constructor Injection.

16. What is the difference between the Configuration types XML and Annotation?

These are the two ways of setting up the configuration, and they perform in the say way. Though, when the annotation approach is taken very less amount of code is written and the result would be the same as compared to the XML approach.

17. List out the ways Auto wiring is done?

(I) by Type

(II) by Name

(III) Constructor (same as by Type, but through constructor)

18. What is Dirty Read?

When a transaction (t1) is meant to read the changes that are performed by another transaction (t2) and provided transaction t2 is not committed yet; then in such a situation, the transaction t1 is called Dirty Read transaction.

19. Why Spring Boot?

Spring-based applications have a lot of configuration (boiler-plate code). In Spring MVC, a lot of configuration is required (like component scan, dispatcher servlet, view resolver, etc). Whereas, in Spring Boot the boiler-plate code is not required.

20. Spring vs Spring MVC vs Spring Boot?

Spring: the most important feature of Spring is Dependency Injection or Inversion of Control.

Spring MVC: provides a decoupled approach in developing web applications. Concepts like [Dispatcher Servlet], [Model And View], [View Resolver] makes web application development easy.

Spring Boot: makes the configuration very easy and automatic using a feature called Auto Configuration, in which the [Dispatcher Servlet] is done by Spring internally.

21. What is the role of @SpringBootApplication?

This annotation is used to launch up the entire application. Internally, @SpringBootApplication does the following,

@SpringBootConfiguration: same as @Configuration in a Spring Application.

@EnableAutoConfiguration: auto-configures the classes available in the class path.

@ComponentScan: all the classes available under a package will be scanned when this annotation is applied.

22. What does an Embedded Server mean in Spring Boot?

To deploy any web application a server like Tomcat is required. In Spring Boot a server (like Tomcat) is available as part of the application in a jar. The concept of Embedded Server makes the deployment of application very easy and independent.

23. Why do we use application.properties?

The file application.properties is used to configure things like database details, log generation, security (username/password), serialization, etc.

24. What is Spring JDBC?

Spring JDBC uses methods like update (query), execute (query) and query (SQL, result Set Extractor) to interact with the database.

25. What is the difference between JDBC and Spring JDBC?

In JDBC, the checked exceptions need to be written; whereas, in Spring JDBC those exceptions are made into Runtime Exceptions. Which means, exception handling is not manually done in Spring JDBC.

26. What is JPA?

Java Persistence API (JPA) defines the mapping from Java Object to a Database Table. The procedure to map a Java object to a row in a database table is defined in JPA. JPA provides a lot of useful annotations, using which the relationship between classes and tables are defined.

27. What is Hibernate? How is it different from JPA?

Once the mapping is done, Hibernate (a JPA Implementation) will help us create query under the hood and interact with the database.

28. Describe the cases in which the Dependency Injection is done through Constructors and Setters?

When the dependencies are required/mandatory, the Constructor approach is selected. And when the dependencies are optional then the Setters approach is used.

29. What is the importance of POM.XML file?

Project Object Model (POM) is an XML formatted file in which all the configuration for a maven project is defined. The most commonly used tags in POM.XML are <groupid>, <artifactId>, <version>, <packaging> and a few more.

30. What does the @RequestParam annotation do?

This allows the server side to read from data and automatically bind it to a parameter coming into the method.

31. Annotation: Spring Boot Application?

Spring Boot @SpringBootApplication annotation is used to mark a configuration class that declares one or more @Bean methods and also triggers auto-configuration and component scanning. It’s same as declaring a class with @Configuration, @EnableAutoConfiguration and @ComponentScan annotations.

32. Annotation: Component

The @Component annotation marks a java class as a bean so the component-scanning mechanism of spring can pick it up and pull it into the application context

33. Annotation: Auto wire

The @Autowired annotation can be used to autowire bean on the setter method just like @Required annotation, constructor, a property or methods with arbitrary names and/or multiple arguments.

34. Annotation: Entity

JPA will include any class annotated with @Entity in the persistence management setup. You don't need persistence.xml if you use annotations.

35. Annotation: Id

The @Idannotation is inherited from javax.persistence.Id, indicating the member field below is the primary key of current entity.

36. Annotation: Service

Service annotation is used in your service layer and annotates classes that perform service tasks, often you don't use it but in many case you use this annotation to represent a best practice. For example, you could directly call a DAO class to persist an object to your database but this is horrible. It is pretty good to call a service class that calls a DAO. This is a good thing to perform the separation of concerns pattern.

37. Annotation: Not Blank

The @NotBlank annotation uses the [Not Blank Validator] class, which checks that a character sequence’s trimmed length is not empty. To put it simply, a String field constrained with @NotBlank must be not null and the trimmed length must be greater than zero.

38. Annotation: Not Null

The @NotNull Annotation is, actually, an explicit contract declaring the following:

A method should not return null.

A variable (like fields, local variables, and parameters) cannot hold null value.

39. Annotation: Table

The @Table annotation allows you to specify the details of the table that will be used to persist the entity in the database.

The @Table annotation provides four attributes, allowing you to override the name of the table, its catalogue, and its schema, and enforce unique constraints on columns in the table. For now, we are using just table name, which is EMPLOYEE.

40. Annotation: Generate Values

The @GeneratedValue annotation is to configure the way of increment of the specified column(field). For example when using MySQL, you may specify auto Increment in the definition of table to make it self-incremental.

41. Annotation: JSON Ignore Properties

ignores the specified logical properties in JSON serialization and deserialization. It is annotated at class level. [Json Ignore Properties] are considered to be ignored in JSON serialization and deserialization.

42. Annotation: Column

The @Column annotation is used to specify the details of the column to which a field or property will be mapped.

43. Annotation: Temporal

Temporal annotation in JPA implementation can only be used with the fields and property get methods. Also, this annotation can be specified for the persistent fields or properties java.util.Data or java.util.Calendar. This annotation is available since the release of JPA 1.0. @Temporal solves the one of the major issue of converting the date and time values from Java object to compatible database type and retrieving back to the application.

44. Annotation: Entity Listners

JPA [entity listeners] are not entities, they are simply java classes in which JPA lifecycle [call back] methods are implemented using annotations. Entities should defines [entity listeners] using annotation @EntityListeners

Multiple listeners can be listed.

Whenever a lifecycle event occurs, provider iterates through the list of all JPA [entity listeners] listed in @EntityListeners annotation and instantiates them in the order they are listed.

Provider invokes [call back] methods of listener and passes entity instance to it.

[call back] methods from entity are invoked if there are any.

45. Annotation: Created Date

The @CreatedDate won't work by itself if you just put @EntityListeners(AuditingEntityListener.class) on your entities. In order, it'll work you have to do a little more configuration.

Let's say that in your DB the field of @CreatedDate is String type, and you want to return the user that is currently logged in as a value for @CreatedDate

46. Annotation: Repository

The @Repository annotation is a specialization of the @Component annotation with similar use and functionality. In addition to importing the DAOs into the DI container, it also makes the unchecked exceptions (thrown from DAO methods) eligible for translation into Spring DataAccessException.

47. Annotation: Query

We can pass an additional parameter of type Sort to a Spring Data method declaration that has the @Query annotation. It'll be translated into the ORDER BY clause that gets passed to the database.

48. Annotation: Rest Controller

Spring RestController annotation is a convenience annotation that is itself annotated with @Controller and @ResponseBody. This annotation is applied to a class to mark it as a request handler.

Spring RestController annotation is used to create RESTful web services using Spring MVC. Spring RestController takes care of mapping request data to the defined request handler method. Once response body is generated from the handler method, it converts it to JSON or XML response.

49. what is the spring version used?

3.9

50. What is the JPA version used

2.2.0.M1

51. What is the spring JPA hibernate version used

5.1.0 final

52. What is a dependency in a POM file? What is the full form of POM?

Dependency management is a core feature of Maven. Managing dependencies for a single project is easy. Managing dependencies for multi-module projects and applications that consist of hundreds of modules is possible. Maven helps a great deal in defining, creating, and maintaining reproducible builds with well-defined classpaths and library versions.

Project Object Model.