Spring Framework Interview Questions

Curated questions for Spring Core, Boot, Data, AOP, and real-world scenarios. Use the filter to jump to what you need.

Filter questions (e.g., IoC, DI, @Transactional, AOP, Boo Showing 129 of 129

Spring Basics

Dependency Injection/IoC, bean scopes/lifecycle, configuration styles (Java/XML/annotations), Boot auto-configuration, profiles, properties, and more.

1. ### What is Spring Framework?

   Spring is an open-source framework for Java-based enterprise applications. It provides a comprehensive programming and configuration model for modern Java-based enterprise applications.

2. ### What are the key components of Spring Framework?

   The key components of Spring Framework include: Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, and Test.

3. ### What is Inversion of Control (IoC) in Spring Framework?

   Inversion of Control (IoC) is a design principle that allows objects to be decoupled from each other. In Spring Framework, IoC is achieved through Dependency Injection (DI), which is a way to supply an external dependency to a class.

4. ### What is Dependency Injection in Spring Framework?

   Dependency Injection (DI) is a design pattern that allows objects to be decoupled from their dependencies. In Spring Framework, DI is used to provide objects with their required dependencies, either through constructors, setters, or fields.

5. ### What is AOP in Spring Framework?

   AOP (Aspect Oriented Programming) is a programming paradigm that allows developers to modularize cross-cutting concerns, such as logging, security, and transactions. In Spring Framework, AOP is used to add behavior to existing objects without having to modify their code.

6. ### What is Bean Factory in Spring Framework?

   Bean Factory is the root interface of Spring's DI container and it is responsible for instantiating, configuring, and managing a number of objects, known as beans. Bean Factory is responsible for creating and managing beans and providing them to other objects when requested.

7. ### What is the difference between ApplicationContext and BeanFactory in Spring Framework?

   BeanFactory is the basic implementation of the Spring container, while ApplicationContext is an extension of BeanFactory, which provides additional functionalities such as internationalization, event-propagation, and annotation-based context configuration.

8. ### What is a DispatcherServlet in Spring Framework?

   DispatcherServlet is the front controller class in Spring MVC, which acts as the central point to receive all incoming requests. DispatcherServlet is responsible for coordinating with the appropriate handlers to process the incoming request and returning the response.

9. ### What is the role of @Autowired in Spring Framework?

   @Autowired is a Spring annotation used for automatic Dependency Injection. It can be applied to fields, setter methods, or constructors to inject an object's dependencies.

10. ### What is the use of @Controller annotation in Spring Framework?

    @Controller is a stereotype annotation that is used in Spring MVC to indicate that a particular class serves as a controller. It is used to handle user requests and to provide the appropriate response.

11. ### What is @Repository annotation in Spring Framework?

    @Repository is a stereotype annotation that is used in Spring to indicate that a particular class serves as a DAO (Data Access Object) component. It is used to manage the persistence of data in the application.

12. ### What is the difference between @Component, @Service and @Repository annotations in Spring Framework?

    @Component is a generic stereotype annotation in Spring, used for any Spring-managed component. @Service is used to indicate that a particular class serves as a service layer component. @Repository is used to indicate that a particular class serves as a DAO (Data Access Object) component. These annotations are used to give meaning to the components in the application and are used in a loosely-coupled fashion.

13. ## What is the use of @Transactional annotation in Spring Framework?

    @Transactional is a Spring annotation used to demarcate transactions in a Spring-based application. It is used to manage transaction boundaries and to ensure that changes made to the data are rolled back in case of an error.

14. ## What is the use of @Qualifier annotation in Spring Framework?

    @Qualifier is a Spring annotation used to disambiguate between multiple beans of the same type that are available in the application context. It can be used with @Autowired to specify which bean to inject in case of multiple beans of the same type.

15. ## What is the difference between @Primary and @Qualifier annotations in Spring Framework?

    @Primary is used to specify the default bean that should be used for a particular type, in case there are multiple beans of the same type. @Qualifier, on the other hand, is used to specify a particular bean by its name or alias, in case there are multiple beans of the same type.

16. ## What is the use of @Value annotation in Spring Framework?

    @Value is a Spring annotation used to inject values from properties files or from command line arguments into fields or methods. It can also be used to assign a default value in case a value is not available.

17. ## What is the difference between @RequestMapping and @GetMapping annotations in Spring Framework?

    @RequestMapping is a generic annotation used for mapping URLs to methods in a Spring MVC application. @GetMapping is a specialized form of @RequestMapping that is used to handle HTTP GET requests.

18. ## What is the use of @RestController annotation in Spring Framework?

    @RestController is a specialized form of @Controller that is used to handle RESTful web services in a Spring MVC application. It is used to return data as JSON or XML instead of a view.

19. ## What is the use of @ExceptionHandler annotation in Spring Framework?

    @ExceptionHandler is a Spring annotation used to handle exceptions in a Spring MVC application. It is used to specify a method that will be invoked in case of an exception in the application.

20. ## What is the use of @PostConstruct and @PreDestroy annotations in Spring Framework?

    @PostConstruct and @PreDestroy are standard annotations defined in the Java API for Common Annotations and are used in Spring to specify methods that should be called after a bean has been constructed or before a bean is destroyed, respectively.

21. ## What is the difference between ApplicationContext and BeanFactory in Spring Framework?

    BeanFactory is the root interface for accessing a Spring IoC container. It provides basic facilities for managing the lifecycle of beans and resolving dependencies. ApplicationContext is an extension of BeanFactory and provides additional features such as the ability to load configuration files, the ability to publish events, and the ability to resolve messages. ApplicationContext is more commonly used in Spring applications compared to BeanFactory.

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

    @Controller is a stereotype annotation in Spring that is used to indicate that a particular class serves as a controller in a Spring MVC application. @RestController is a specialized form of @Controller that is used to handle RESTful web services in a Spring MVC application. It is used to return data as JSON or XML instead of a view.

23. ## What is the use of AOP in Spring Framework?

    AOP (Aspect-Oriented Programming) is a programming paradigm in Spring that allows the separation of cross-cutting concerns, such as logging or security, from the core business logic of the application. It provides a way to modularize these concerns and to apply them consistently across the application. AOP is used in Spring to provide a clean separation of concerns and to reduce code duplication.

24. ## What is the difference between Constructor-based and Setter-based Dependency Injection in Spring Framework?

    Constructor-based Dependency Injection is a form of dependency injection in which dependencies are provided to a class through its constructor. Setter-based Dependency Injection is a form of dependency injection in which dependencies are provided to a class through setter methods. Both forms of dependency injection have their own advantages and disadvantages and the choice between them depends on the requirements of the particular application.

25. ## What is the use of @Scheduled annotation in Spring Framework?

    @Scheduled is a Spring annotation used to schedule tasks in a Spring-based application. It is used to specify a method that should be invoked periodically, after a fixed delay, or at a specific time.

26. ## What is the use of @Primary annotation in Spring Framework?

    @Primary is a Spring annotation used to specify the primary bean among multiple candidate beans. It is used to disambiguate which bean should be used when multiple beans are eligible to be injected into a single dependency.

27. ## What is the difference between @Autowired and @Inject annotations in Spring Framework?

    @Autowired is a Spring annotation used for automatic wiring of beans in a Spring application. @Inject is a standard annotation defined in the Java API for Dependency Injection and is used for the same purpose in a Spring application.

28. ## What is the difference between @ComponentScan and @Import annotations in Spring Framework?

    @ComponentScan is a Spring annotation used to specify the packages to be scanned for annotated components. It is used to automatically detect and register beans in a Spring application. @Import is a Spring annotation used to import one or more configuration classes into the current configuration class.

29. ## What is the difference between @Component and @Service annotations in Spring Framework?

    @Component is a general-purpose stereotype annotation in Spring that is used to indicate that a particular class serves as a component in a Spring-based application. @Service is a stereotype annotation in Spring that is used to indicate that a particular class serves as a service in a Spring-based application. The main difference between the two is their intended use and the level of abstraction they provide.

30. ## What is the use of @Qualifier annotation in Spring Framework?

    @Qualifier is a Spring annotation used to disambiguate between multiple candidates for injection. It is used to specify which of several possible beans should be injected, when multiple beans match the expected type.

31. ## What is the use of @Profile annotation in Spring Framework?

    @Profile is a Spring annotation used to specify the activation of certain beans in certain profiles. It is used to control which beans are created and used in a particular environment, such as development, testing, or production.

32. ## What is the use of @Value annotation in Spring Framework?

    @Value is a Spring annotation used to inject values from properties files or system properties into a bean. It is used to set values for fields and method parameters from an external source, such as a properties file or system environment variables.

33. ## What is the use of @Lazy annotation in Spring Framework?

    @Lazy is a Spring annotation used to specify that a bean should be initialized lazily. It is used to defer the creation of a bean until it is actually needed, rather than creating it upfront, which can improve performance and reduce memory usage in some cases.

34. ## What is the use of @Scope annotation in Spring Framework?

    @Scope is a Spring annotation used to specify the scope of a bean in a Spring application. It is used to control the lifecycle and visibility of a bean, and to determine how many instances of a bean should be created and managed in a single IoC container.

35. ## What is the use of @Async annotation in Spring Framework?

    @Async is a Spring annotation used to specify that a method should be executed asynchronously. It is used to execute methods in a background thread, allowing the main thread to continue executing and improving performance in some cases.

36. ## What is the use of @Transactional annotation in Spring Framework?

    @Transactional is a Spring annotation used to specify that a method or an entire class should participate in a transaction. It is used to manage the transaction boundary for a particular method or a set of methods in a class, and to ensure that a consistent and atomic set of operations is executed within the scope of a single transaction.

37. ## What is the use of @PostConstruct and @PreDestroy annotations in Spring Framework?

    @PostConstruct and @PreDestroy are annotations used in Spring to specify a method to be executed after a bean has been created and initialized, and before a bean is destroyed, respectively. They are used to perform additional configuration and cleanup tasks for a bean after it has been created or before it is destroyed.

38. ## What is the use of ApplicationEvent and ApplicationListener in Spring Framework?

    ApplicationEvent and ApplicationListener are classes in Spring used to implement the Observer Design Pattern. ApplicationEvent is used to represent an event in a Spring application, while ApplicationListener is used to listen to ApplicationEvent objects and perform some action in response.

39. ## What is the difference between BeanFactory and ApplicationContext in Spring Framework?

    BeanFactory and ApplicationContext are both IoC containers in Spring Framework, but ApplicationContext is a sub-interface of BeanFactory. BeanFactory provides basic IoC container functionality, while ApplicationContext provides additional features, such as support for internationalization (I18N) messages, event publishing, and JNDI access.

40. ## What is the difference between Spring Web MVC and Spring REST in Spring Framework?

    Spring Web MVC is a full-featured Model-View-Controller (MVC) framework for building web applications in Spring, while Spring REST is a sub-project of Spring Web MVC that focuses on building RESTful web services. Spring REST provides a set of annotations and tools for

building RESTful web services, while Spring Web MVC provides a full-featured MVC framework for building both RESTful and non-RESTful web applications.

41. ### What is the use of DispatcherServlet in Spring Framework?

    DispatcherServlet is the front-controller in Spring Framework's Web MVC framework. It is responsible for handling all incoming requests, mapping them to appropriate controllers, and returning the response to the client. DispatcherServlet acts as a central hub for request processing, and delegates requests to handlers, while also applying common behaviors, such as exception handling and view resolution.

42. ### What is the difference between @Controller and @RestController in Spring Framework?

    @Controller and @RestController are both annotations in Spring used to define controllers in a Spring MVC application. The main difference between them is that @RestController is a specialized version of @Controller that is specifically designed for building RESTful web services. @RestController automatically adds @ResponseBody to all methods, which means that the return value of a method will be automatically serialized and returned as the response body of an HTTP request.

43. ### What is the difference between @RequestMapping and @GetMapping in Spring Framework?

    @RequestMapping and @GetMapping are both annotations in Spring used to define mapping between HTTP requests and controller methods. @RequestMapping is a more generic version of the annotation, and can be used to map requests for any HTTP method (GET, POST, PUT, DELETE, etc.). @GetMapping, on the other hand, is a specialized version of @RequestMapping that is specifically designed to map GET requests.

44. ### What is the use of Spring Security in Spring Framework?

    Spring Security is a framework in Spring that provides security services for Java applications. It is used to secure applications and APIs, and to provide authentication and authorization for users accessing the application. Spring Security provides a flexible and comprehensive security solution, and can be easily integrated into a Spring application to secure both the front-end and back-end components.

45. ### What is the use of @Autowired annotation in Spring Framework?

    @Autowired is a Spring annotation used for dependency injection. It is used to automatically wire a bean with its dependencies, so that the dependencies are injected into the bean at runtime. @Autowired can be used to annotate fields, setter methods, or constructors to have the dependencies automatically injected by Spring.

46. ### What is the use of @Qualifier annotation in Spring Framework?

    @Qualifier is a Spring annotation used in conjunction with @Autowired to resolve ambiguities in dependency injection. When there are multiple candidate beans for a single dependency, @Qualifier can be used to specify which bean should be injected. This allows you to select the specific bean you want to be injected based on its name or any other qualifying information.

47. ### What is the use of @Primary annotation in Spring Framework?

    @Primary is a Spring annotation that is used to specify the primary bean for autowiring. When there are multiple candidate beans for a single dependency, @Primary can be used to specify which bean should be used as the primary bean for autowiring. If multiple beans are marked as primary, the first one will be used. @Primary can be used to override any @Qualifier annotations.

48. ### What is the use of @Value annotation in Spring Framework?

    @Value is a Spring annotation used to inject values into fields, setter methods, or constructors from properties defined in a properties file or environment variables. It can also be used to specify a default value if the specified property is not found. This allows you to externalize the configuration of your application, and make it easily configurable without having to modify the code.

49. ### What is the difference between singleton and prototype scope in Spring Framework?

    Singleton and prototype are two different scopes in Spring. The singleton scope is the default scope, and ensures that only one instance of a bean is created and shared across the entire application. The prototype scope, on the other hand, creates a new instance of a bean each time it is requested. This allows you to create multiple instances of a bean, each with its own state.

50. ### What is the use of @Service annotation in Spring Framework?

    @Service is a Spring annotation used to indicate that a class is a service layer component. It is typically used to indicate that a class is part of the service layer, and provides business logic for the application. @Service can be used to annotate classes that provide the business logic for your application, and can be easily wired into other components, such as controllers, to provide the required services.

51. ### What is the use of @Controller annotation in Spring Framework?

    @Controller is a Spring annotation used to indicate that a class is a controller component in the presentation layer. It is typically used to handle incoming HTTP requests, and provide the necessary processing logic to produce a response. @Controller can be used to annotate classes that act as the front-end controllers for your application, handling incoming requests and sending responses to the client.

52. ### What is the use of @RequestMapping annotation in Spring Framework?

@RequestMapping is a Spring annotation used to map incoming HTTP requests to specific methods in a controller. It is typically used to map an incoming URL to a specific method in a controller, which can then handle the request and produce a response. @RequestMapping can be used to specify the HTTP methods that a particular method should handle, as well as to specify the URL pattern that the method should handle.

53. What is the use of @ResponseBody annotation in Spring Framework?

@ResponseBody is a Spring annotation used to indicate that a method return value should be bound to the response body. It is typically used in RESTful web services, where the response should be in a specific format, such as JSON or XML. @ResponseBody can be used to indicate that the return value of a method should be automatically serialized and written to the response body, without the need for additional configuration.

54. What is the use of @RestController annotation in Spring Framework?

@RestController is a convenience annotation in Spring that combines the functionality of @Controller and @ResponseBody. It is typically used to create RESTful web services, where the response should be in a specific format, such as JSON or XML. @RestController can be used to create a controller that automatically serializes the return value of its methods and writes it to the response body, without the need for additional configuration.

55. What is the use of @PathVariable annotation in Spring Framework?

@PathVariable is a Spring annotation used to extract values from the URL and bind them to method parameters. It is typically used in RESTful web services, where the URL contains variables that need to be extracted and used in the processing logic. @PathVariable can be used to extract values from the URL and bind them to method parameters, allowing you to use these values in your processing logic.

56. What is the use of @Qualifier annotation in Spring Framework?

@Qualifier is a Spring annotation used to disambiguate bean references in cases where multiple beans match the expected type. It is typically used in conjunction with the @Autowired annotation to specify which bean should be injected, in cases where there are multiple beans of the same type. @Qualifier can be used to specify a unique identifier for a bean, allowing you to select a specific bean when there are multiple options.

57. What is the use of @Value annotation in Spring Framework?

@Value is a Spring annotation used to inject property values into fields, methods, or constructors. It is typically used to inject values from properties files, such as application.properties, into your application. @Value can be used to specify a default value for a property, as well as to specify the name of the property that should be injected into a field or method.

58. What is the use of @Repository annotation in Spring Framework?

@Repository is a Spring annotation used to indicate that a class is a data access object (DAO) in the persistence layer. It is typically used to define DAOs that provide access to a database, perform CRUD operations, and return data in a specific format. @Repository can be used to annotate classes that implement the persistence layer for your application, providing data access and storage functionality.

59. What is the use of @Aspect annotation in Spring Framework?

@Aspect is a Spring annotation used to define a class as an aspect in the aspect-oriented programming (AOP) paradigm. It is typically used to define cross-cutting concerns, such as logging, security, and transaction management, that can be applied to multiple parts of an application. @Aspect can be used to define classes that encapsulate the AOP logic for your application, allowing you to modularize and reuse common functionality across multiple parts of your code.

60. What is the difference between @Component and @Controller in Spring Framework?

@Component and @Controller are both used to annotate classes in the Spring framework, but they are used for different purposes. @Component is a generic stereotype used to annotate any class that is a component of the application. @Controller, on the other hand, is specifically used to annotate classes that represent controllers in the MVC pattern. In general, controllers are responsible for processing user requests, whereas components are more general-purpose classes that may provide services to other parts of the application.

61. What is the difference between @Autowired and @Resource in Spring Framework?

@Autowired and @Resource are both used to inject beans into fields, methods, or constructors in the Spring framework, but they work slightly differently. @Autowired is a Spring-specific annotation that automatically injects the first matching bean it finds in the application context. @Resource, on the other hand, is a standard Java EE annotation that can be used to inject beans based on their name or type. In general, @Autowired is preferred in Spring applications, but @Resource can be used if compatibility with Java EE is required.

62. What is the difference between @Scope("singleton") and @Scope("prototype") in Spring Framework?

@Scope("singleton") and @Scope("prototype") are used to specify the scope of a bean in the Spring framework, and they determine how many instances of a bean will be created and managed by the application context. @Scope("singleton") specifies that only one instance of the bean should be created and shared by the entire application, whereas @Scope("prototype") specifies that a new instance of the bean should be created every time it is requested from the application context.

63. What is the use of ApplicationContext in Spring Framework?

ApplicationContext is an interface in the Spring framework that provides the context in which a Spring application runs. It is responsible for managing the lifecycle of beans, including creating and initializing beans, injecting dependencies, and managing their destruction. ApplicationContext also provides a centralized, unified way of accessing resources, such as configuration files and data sources, and it can be used to publish and receive events between beans.

64. What is the difference between BeanFactory and ApplicationContext in Spring Framework?

BeanFactory and ApplicationContext are both used to manage the lifecycle of beans in the Spring framework, but ApplicationContext is a more advanced and feature-rich interface that provides additional functionality beyond that of BeanFactory. BeanFactory is the simplest form of container in the Spring framework, providing basic support for bean instantiation, configuration, and lifecycle management. ApplicationContext extends BeanFactory and provides additional functionality, such as support for internationalization (i18n), event publication and listener registration, and easier integration with Spring's AOP features.

65. What is the difference between ApplicationContext and WebApplicationContext in Spring Framework?

ApplicationContext and WebApplicationContext are both used in the Spring framework to provide the context in which a Spring application runs, but WebApplicationContext is specifically designed to be used in web applications. ApplicationContext is a more general-purpose interface that can be used in any type of Java application, while WebApplicationContext is a subinterface of ApplicationContext that provides additional functionality specifically for web applications, such as access to the ServletContext and the ability to resolve resources within a web context.

66. What is the difference between ApplicationContextAware and BeanNameAware in Spring Framework?

ApplicationContextAware and BeanNameAware are both interfaces in the Spring framework that allow a bean to be aware of its environment and context within the application. ApplicationContextAware is used to give a bean access to the ApplicationContext in which it is running, so that it can access other beans, resources, and configuration data. BeanNameAware, on the other hand, is used to give a bean access to its own bean name, so that it can be aware of how it is named and referred to within the application context.

67. What is the difference between BeanFactory and FactoryBean in Spring Framework?

BeanFactory and FactoryBean are both used in the Spring framework to manage the creation and configuration of beans, but they serve different purposes. BeanFactory is a container interface that provides basic support for bean instantiation, configuration, and lifecycle management. FactoryBean, on the other hand, is a special type of bean that is used to create other beans. It is a factory that can be used to create objects that are not directly managed by the application context, such as objects created by a third-party library or objects that are expensive to create.

68. What is the difference between BeanPostProcessor and BeanFactoryPostProcessor in Spring Framework?

BeanPostProcessor and BeanFactoryPostProcessor are both used in the Spring framework to process beans after they have been created and configured, but they serve different purposes. BeanPostProcessor is used to post-process individual beans and make modifications to them after they have been created and configured, such as adding additional properties or changing the behavior of a bean. BeanFactoryPostProcessor, on the other hand, is used to post-process the entire bean factory, allowing you to make changes to the bean definitions or the configuration metadata used by the bean factory to create and manage beans.

69. What is the difference between BeanDefinitionRegistryPostProcessor and BeanFactoryPostProcessor in Spring Framework?

BeanDefinitionRegistryPostProcessor and BeanFactoryPostProcessor are both used in the Spring framework to process bean definitions and make changes to them, but BeanDefinitionRegistryPostProcessor is a more advanced interface that provides access to the underlying BeanDefinitionRegistry. BeanDefinitionRegistryPostProcessor can be used to add, remove, or modify bean definitions, while BeanFactoryPostProcessor can only modify the configuration metadata used by the bean factory to create and manage beans.

70. What is the difference between @Autowired and @Resource in Spring Framework?

@Autowired and @Resource are both annotations used in the Spring framework for dependency injection, but they serve different purposes and have different features. @Autowired is used to automatically wire a bean property with a matching bean from the application context, based on type. @Resource, on the other hand, is used to wire a bean property with a matching bean from the application context, based on either name or type. @Autowired has more advanced features, such as support for constructor injection and support for fine-tuning the matching algorithm, while @Resource is a simpler, more straightforward option for dependency injection.

71. What is the difference between @Component and @Service in Spring Framework?

@Component and @Service are both annotations used in the Spring framework to indicate that a class is a component and should be managed as a bean by the application context, but they serve different purposes and are used in different contexts. @Component is a general-purpose annotation that can be used to annotate any type of component, including service components, data access components, and presentation components. @Service is a specialized annotation that is used specifically for service components, which are the components that provide the business logic and services of the application.

72. What is the difference between @Controller and @RestController in Spring Framework?

@Controller and @RestController are both annotations used in the Spring framework to indicate that a class is a controller and should be managed as a bean by the application context, but they serve different purposes and are used in different contexts. @Controller is used

for traditional web controllers that handle web requests and generate view-based responses, while @RestController is used for RESTful web services that handle REST requests and generate JSON or XML responses.

73. What is the difference between @Scope and @Lazy in Spring Framework?

@Scope and @Lazy are both annotations used in the Spring framework to control the scope and behavior of beans, but they serve different purposes. @Scope is used to specify the scope of a bean, such as singleton, prototype, request, session, or global session, and determines when and how often a new instance of the bean will be created. @Lazy, on the other hand, is used to specify that a bean should be initialized lazily, meaning that it will not be created until it is actually needed.

74. What is the difference between @Qualifier and @Primary in Spring Framework?

@Qualifier and @Primary are both annotations used in the Spring framework to resolve conflicts and disambiguate between multiple candidate beans for injection, but they serve different purposes. @Qualifier is used to specify the exact bean that should be injected, based on its unique bean name or custom qualifier. @Primary, on the other hand, is used to specify a primary bean that should be preferred over other candidate beans when multiple beans match the required type.

Scenario-Based Q&A

Practical situations covering transactions, @Transactional pitfalls, AOP, caching, REST/web, data access, testing, and troubleshooting.

1. What is the difference between ApplicationContext and BeanFactory in Spring?

BeanFactory is the basic container of the Spring framework, while ApplicationContext is an extension of BeanFactory with additional functionalities, such as internationalization, event publication, and access to remote resources.

2. What is the role of DispatcherServlet in Spring MVC?

DispatcherServlet is the front controller in Spring MVC that receives all incoming requests and maps them to appropriate controllers for processing.

3. What is the @Autowired annotation in Spring used for?

@Autowired is used to automatically wire bean dependencies in the Spring application context, eliminating the need for explicit configuration in XML files.

4. What is a Spring Bean and how is it created?

A Spring Bean is a managed object within the Spring framework's IoC container. It is created by defining a Java class with a specific configuration, such as using the @Component or @Bean annotation, and then adding it to the application context.

5. What is the role of the @ComponentScan annotation in Spring?

@ComponentScan is used to specify the packages that should be scanned for annotated components in the Spring application context. This helps to automatically detect and register beans with the context.

6. What is a Spring Bean Life Cycle?

The Spring Bean life cycle refers to the sequence of events that occur from the time a bean is created, initialized, and finally destroyed. These events are managed by the Spring framework and can be customized through various methods and interfaces, such as InitializingBean, DisposableBean, and @PostConstruct and @PreDestroy annotations.

7. What is the difference between @Controller and @RestController in Spring?

@Controller is used to define a Spring MVC controller that handles HTTP requests, while @RestController is a specialized version of @Controller that is specifically designed to return data in JSON or XML format over HTTP.

8. What is the purpose of the @Qualifier annotation in Spring?

@Qualifier is used to disambiguate between multiple instances of the same type that are present in the application context. It allows you to specify which bean should be used for autowiring, based on a logical name that you assign to the bean.

9. What is the use of AOP in Spring framework?

AOP (Aspect-Oriented Programming) is used in the Spring framework to provide a way to modularize cross-cutting concerns, such as logging, transaction management, security, and exception handling. It allows you to separate these concerns from the core business logic of the application, resulting in a more organized and maintainable codebase.

10. What is the difference between Spring and Spring Boot?

Spring is a framework for building Java applications, while Spring Boot is a framework for building production-ready Spring applications with minimal configuration. Spring Boot provides a convention-over-configuration approach, automatic configuration of required components, and production-ready features, making it easier to develop and deploy Spring applications.

11. What is the @Transactional annotation in Spring used for?

@Transactional is used to demarcate methods that participate in a transaction. When a method is annotated with @Transactional, any database operations performed within the method will be executed in a transactional context, ensuring consistency and atomicity of the data.

12. ### What is the difference between @RequestParam and @PathVariable in Spring?

    @RequestParam is used to extract query parameters from an HTTP request, while @PathVariable is used to extract values from the URI template of a RESTful resource. @RequestParam is used for optional parameters, while @PathVariable is used for required parameters.

13. ### What is the use of the @Value annotation in Spring?

    @Value is used to assign values to properties from external sources, such as property files, environment variables, and system properties. This allows you to externalize the configuration of your application and provide a way to modify its behavior without recompiling the code.

14. ### What is the use of the @Bean annotation in Spring?

    @Bean is used to define a bean in the Spring application context. It can be used to explicitly configure a bean and its properties, as well as to specify the factory method that creates the bean instance.

15. ### What is the use of the @Autowired annotation in Spring?

    @Autowired is used to automatically wire dependencies in a Spring application. It tells the Spring framework to inject a required bean into a class property, constructor, or setter method, based on its type.

16. ### What is the use of the @Primary annotation in Spring?

    @Primary is used to indicate that a bean should be given priority when multiple instances of the same type are present in the application context. It is used to resolve conflicts between multiple beans of the same type that are eligible for autowiring.

17. ### What is the use of the @Profile annotation in Spring?

    @Profile is used to conditionally activate or deactivate a bean, based on the value of the active profile. It allows you to define different configurations for different environments, such as development, test, and production, and to choose the appropriate configuration at runtime based on the active profile.

18. ### What is the use of the @Async annotation in Spring?

    @Async is used to indicate that a method should be executed asynchronously, in a separate thread. It allows you to offload long-running tasks from the main thread, improving the responsiveness and performance of your application.

19. ### What is the use of the @Scheduled annotation in Spring?

    @Scheduled is used to schedule the execution of a method, either at fixed intervals or based on a cron expression. It allows you to run tasks periodically, such as sending emails, generating reports, and cleaning up data, without manual intervention.

20. ### What is the main advantage of using Spring Boot?

    The main advantage of using Spring Boot is its ability to simplify the development and deployment of Spring applications. Spring Boot provides a convention-over-configuration approach, automatic configuration of required components, and production-ready features, making it easier to develop and deploy Spring applications.

21. ### What is the use of the @SpringBootApplication annotation in Spring Boot?

    @SpringBootApplication is a convenience annotation that combines several annotations, including @Configuration, @EnableAutoConfiguration, and @ComponentScan. It is used to enable the auto-configuration of a Spring Boot application and to scan for components and configurations in the classpath.

22. ### What is the use of the application.properties or application.yml file in Spring Boot?

    The application.properties or application.yml file is used to configure a Spring Boot application. It can be used to specify various properties, such as the server port, database connection details, and application-specific settings, that can be used to customize the behavior of a Spring Boot application.

23. ### What is the difference between embedded tomcat and standalone tomcat in Spring Boot?

    Embedded tomcat is a lightweight version of tomcat that is included in a Spring Boot application and started when the application is run. It is suitable for development and testing, as it requires no external setup or configuration. Standalone tomcat is a full-fledged tomcat server that runs separately from a Spring Boot application and is used to deploy production-ready applications.

24. ### What is the use of the Spring Initializer in Spring Boot?

    The Spring Initializer is a web-based tool provided by Spring Boot that can be used to generate a basic structure for a new Spring Boot project. It allows you to choose the dependencies that you need for your application, such as spring-web, spring-data-jpa, and spring-security, and to generate a project structure and configuration files, such as pom.xml and application.properties, that you can use as a starting point for your application.

25. ## What is the use of the spring-boot-starter-parent POM in Spring Boot?

    The spring-boot-starter-parent POM is a parent POM provided by Spring Boot that provides default configurations for a Spring Boot project. It sets up a consistent version of dependencies, such as Spring, Apache Tomcat, and Jackson, and provides a number of helpful Maven plugins, such as the Spring Boot Maven plugin, that simplify the build process.

26. ## What is the use of the Spring Boot Actuator in Spring Boot?

    The Spring Boot Actuator is a set of tools provided by Spring Boot that provide production-ready features, such as monitoring, health checks, and auditing, for a Spring Boot application. It allows you to monitor the health of your application and to access production-related information, such as the status of the application, the configuration, and the logs, in a secure and convenient way.

27. ## What is the use of the CommandLineRunner interface in Spring Boot?

    The CommandLineRunner interface is used to run a piece of code after the Spring Boot application has started. It allows you to perform some action, such as setting up data or printing information, when the application is started. A class that implements the CommandLineRunner interface must provide a run method, which is executed after all the beans in the context have been initialized.

28. ## What is the use of the @ConditionalOnProperty annotation in Spring Boot?

    @ConditionalOnProperty is used to conditionally configure a bean, based on the value of a specified property. It allows you to configure different behaviors for different environments, based on the values of properties in the application.properties or application.yml file.

29. ## What is the use of the @ConditionalOnMissingBean annotation in Spring Boot?

    @ConditionalOnMissingBean is used to conditionally configure a bean, based on the absence of another bean of the same type. It allows you to configure a default behavior for a bean if no other bean of the same type has been defined in the application context.

30. ## What is the use of the @SpringBootApplication annotation in Spring Boot?

    @SpringBootApplication is a convenience annotation that is equivalent to using @Configuration, @EnableAutoConfiguration, and @ComponentScan. It enables auto-configuration of the Spring Boot application and scans for components in the current package and all its sub-packages.

31. ## What is the use of the @RestController annotation in Spring Boot?

    @RestController is a convenience annotation that is equivalent to using @Controller and @ResponseBody. It is used to mark a class as a RESTful web service controller and to automatically map the methods of the class to HTTP requests. The methods of a class annotated with @RestController are capable of handling HTTP requests and returning the response in a JSON or XML format.

32. ## What is the use of the @DataJpaTest annotation in Spring Boot?

    @DataJpaTest is used to test the persistence layer of a Spring Boot application. It provides a simple and fast way to test the persistence layer by starting only the components needed for JPA tests and ignoring the rest of the application. This annotation can be used to test the JPA repositories, entity classes, and the transaction management of the persistence layer.

33. ## What is the use of the @SpringBootTest annotation in Spring Boot?

    @SpringBootTest is used to run a Spring Boot application test. It starts the application context and provides a way to test the application components, such as services, controllers, and repositories. The @SpringBootTest annotation can be used to test the complete application, or a specific part of the application, by specifying the properties and configuration required for the test.

34. ## What is the use of the @Value annotation in Spring Boot?

    @Value is used to inject values from the application.properties or application.yml file into a Spring Boot application. It allows you to access the values of properties in the configuration file and use them in your application, for example to set the value of a field or to pass the value as a parameter to a method.

35. ## What is the use of the @EnableAutoConfiguration annotation in Spring Boot?

    @EnableAutoConfiguration is used to enable auto-configuration in a Spring Boot application. It is used to automatically configure the application based on the dependencies that are present on the classpath. This annotation enables the default configuration of the application, such as security, database, and other configurations, and reduces the amount of code required to be written by the developer.

36. ## What is the use of the @Autowired annotation in Spring Boot?

    @Autowired is used to automatically wire beans in a Spring Boot application. It is used to inject a bean from the Spring Application Context into a class. The @Autowired annotation can be used on fields, setter methods, or constructors to inject a bean into a class. This annotation helps to reduce the amount of code required for wiring beans and makes the code more readable.

37. ## What is the use of the @ComponentScan annotation in Spring Boot?

    @ComponentScan is used to specify the base packages to be scanned in a Spring Boot application. It is used to scan the components, such as controllers, services, and repositories, in a Spring Boot application. The @ComponentScan annotation can be used to scan

components in the current package and all its sub-packages, or to specify the packages to be scanned. This annotation helps to reduce the amount of code required for component scanning and makes the code more readable.

38. ## What is the use of the @ConfigurationProperties annotation in Spring Boot?

    @ConfigurationProperties is used to bind external configuration properties to a class in a Spring Boot application. It is used to map the values of properties in the application.properties or application.yml file to a class in the application. The @ConfigurationProperties annotation can be used to bind properties to a class, such as a POJO, and access the values of the properties in the class. This annotation helps to reduce the amount of code required to bind properties to a class and makes the code more readable.

39. ## What is the use of the @Profile annotation in Spring Boot?

    @Profile is used to specify the active profile for a Spring Boot application. It is used to activate a specific profile for a component, such as a controller, service, or repository. The @Profile annotation can be used to specify the active profile for a component, and the component will be loaded only if the specified profile is active. This annotation helps to reduce the amount of code required to activate a specific profile and makes the code more readable.

40. ## What is the use of the @ConditionalOnExpression annotation in Spring Boot?

    @ConditionalOnExpression is used to conditionally register a bean in a Spring Boot application. It is used to register a bean based on a SpEL (Spring Expression Language) expression. The @ConditionalOnExpression annotation can be used to conditionally register a bean, and the bean will be registered only if the SpEL expression evaluates to true. This annotation helps to reduce the amount of code required to conditionally register a bean and makes the code more readable.

41. ## What is the use of the @Value annotation in Spring Boot?

    @Value is used to inject a property value in a Spring Boot application. It is used to inject a value from the application.properties or application.yml file into a class. The @Value annotation can be used to inject a single value or multiple values into a class. This annotation helps to reduce the amount of code required to inject properties into a class and makes the code more readable.

42. ## What is the use of the @RestController annotation in Spring Boot?

    @RestController is used to define a RESTful web service in a Spring Boot application. It is used to create a RESTful endpoint in a Spring Boot application. The @RestController annotation is a combination of the @Controller and @ResponseBody annotations, and it is used to define a RESTful endpoint that returns a JSON or XML response. This annotation helps to reduce the amount of code required to define a RESTful endpoint and makes the code more readable.

43. ## What is the use of the @RequestMapping annotation in Spring Boot?

    @RequestMapping is used to map a request in a Spring Boot application. It is used to map a URL to a method in a RESTful endpoint. The @RequestMapping annotation can be used to map a URL to a method, and the method will be called when the specified URL is accessed. This annotation helps to reduce the amount of code required to map a URL to a method and makes the code more readable.

44. ## What is the use of the @ExceptionHandler annotation in Spring Boot?

    @ExceptionHandler is used to handle exceptions in a Spring Boot application. It is used to specify a method to be called when an exception is thrown in a RESTful endpoint. The @ExceptionHandler annotation can be used to specify a method to be called when an exception is thrown, and the specified method will be called to handle the exception. This annotation helps to reduce the amount of code required to handle exceptions and makes the code more readable.

45. ## What is the purpose of the @SpringBootApplication annotation in Spring Boot?

    @SpringBootApplication is a combination of three annotations, including @SpringBootConfiguration, @EnableAutoConfiguration, and @ComponentScan. It is used to enable the auto-configuration and component scanning features of Spring Boot. The @SpringBootApplication annotation is used to define the main class of a Spring Boot application, and it is typically placed on the class that contains the main method.

46. ## What is the use of the @AutoConfigureMockMvc annotation in Spring Boot?

    @AutoConfigureMockMvc is used to enable the auto-configuration of the MockMvc instance in a Spring Boot application. It is used to test the RESTful endpoints in a Spring Boot application. The @AutoConfigureMockMvc annotation can be used to enable the auto-configuration of the MockMvc instance, and the MockMvc instance can be used to test the RESTful endpoints in a Spring Boot application.

47. ## What is the purpose of the application.properties file in Spring Boot?

    The application.properties file is used to configure the properties of a Spring Boot application. It is used to specify the configuration of a Spring Boot application, such as the database connection, server port, and other configuration parameters. The application.properties file is typically located in the classpath of a Spring Boot application, and the properties defined in this file can be accessed using the @Value annotation.

48. ## What is the purpose of the application.yml file in Spring Boot?

The application.yml file is used to configure the properties of a Spring Boot application using the YAML format. It is used to specify the configuration of a Spring Boot application, such as the database connection, server port, and other configuration parameters. The application.yml file is typically located in the classpath of a Spring Boot application, and the properties defined in this file can be accessed using the @Value annotation.

49. What is the purpose of the Spring Initializer in Spring Boot?

The Spring Initializer is a web-based tool provided by Spring Boot to generate a basic structure of a Spring Boot application. It is used to generate a basic structure of a Spring Boot application, including the required dependencies, configuration files, and other files required for a Spring Boot application. The Spring Initializer can be accessed through the Spring Boot website, and it provides an easy and convenient way to generate the basic structure of a Spring Boot application.

50. What is the difference between Spring Boot and Spring MVC?

Spring Boot is a framework for building production-ready Spring applications. It provides a set of tools for creating standalone, production-grade Spring applications with minimal effort. Spring MVC is a module of the Spring framework that provides a Model-View-Controller architecture for building web applications. While Spring Boot uses Spring MVC as its default web framework, it also provides the option to use other web frameworks such as JAX-RS, JPA, and others. In essence, Spring Boot is a comprehensive toolkit for building production-ready applications, while Spring MVC is a specialized tool for building web applications.

51. How does Spring Boot handle configuration properties?

Spring Boot handles configuration properties in several ways, including properties files, YAML files, command-line arguments, and environment variables. The properties defined in these sources can be accessed in a Spring Boot application using the @Value annotation or the ConfigurationProperties class. Spring Boot also provides the option to bind properties to a Bean using the @ConfigurationProperties annotation. Additionally, Spring Boot provides a comprehensive set of properties for configuring various aspects of a Spring Boot application, such as the database connection, server port, and others.

52. What is the purpose of the @RestController annotation in Spring Boot?

@RestController is a specialized version of the @Controller annotation in Spring MVC. It is used to define a RESTful controller in a Spring Boot application. The @RestController annotation is used to define the RESTful endpoints in a Spring Boot application, and it combines the functionality of the @Controller and @ResponseBody annotations. The @RestController annotation is used to define RESTful endpoints that return JSON or XML responses, and it is one of the key annotations in a Spring Boot application.

53. What is the difference between @Bean and @Component in Spring Boot?

@Bean is a method-level annotation used in the configuration class to define a Bean. It is used to define a Bean that will be managed by the Spring container. On the other hand, @Component is a class-level annotation used to indicate that a class is a candidate for component scanning. It is used to define a Bean that will be automatically detected and registered with the Spring container. In essence, @Bean is used to manually define a Bean in the configuration class, while @Component is used to automatically detect and register Beans using component scanning.

54. What is the purpose of the @Autowired annotation in Spring Boot?

@Autowired is used to automatically wire dependencies in a Spring Boot application. It is used to inject a Bean into another Bean, or to inject a dependency into a class. The @Autowired annotation can be used on a constructor, field, or setter method to inject a Bean into another Bean. It is one of the key annotations in a Spring Boot application, and it is used to handle the dependency injection in a Spring Boot application.

55. What are the key features of the Spring Framework?

The key features of the Spring Framework include dependency injection, aspect-oriented programming, data access and transaction management, and support for multiple view technologies.

Tip: When answering Spring questions, describe bean wiring, configuration, and runtime behavior (context, proxies, transaction boundaries) with concrete examples.

SEARCH

Search topics... Go

INTERVIEW TOPICS

Java Q&A

Java 8 Q&A

Spring Q&A

Hibernate	Q&A
React JS	Q&A
SQL	Q&A
MongoDB	Q&A
JavaScript	Q&A
Kafka	Q&A
Design Patterns	Q&A
Microservices	Q&A
REST API	Q&A
Behavioral	Q&A

QUICK ACTIONS

Start System Design | Practice Coding | Earn a Certificate

RECOMMENDED READING

- Learn Java 17 Programming
- Learn JavaScript Quickly
- SQL QuickStart Guide
- System Design Interview

See all ➜

SPONSORED

Ads help keep content free.

NEED HELP?

Have a topic request or correction? We'd love to hear from you.

Contact Us

THANK YOU!

Your comments and complements are always the inspiration that keeps us motivated.

ABOUT

PROJECTS

ABOUT US

BLOG

AWARDS

ADDRESS

Sterling, Virginia, 20166, US

info@techventurellc.com

641 451 3463

641 451 3463

FOLLOW US

© 2022 Copyright: techventurellc.com