14/08/2023

Q. Write description of all the spring Annotations.

*) spring core

1) @ Required :
- @ Required on setter methods to mark dependencies that we want to popwlate through XML.
- otherwise, Bean Initialization Exception will be thrown.
- This annotation is typically used in conjunction with dependency injection to ensure that all required dependencies are properly injected into a bean before it is used.

2) @ Autowired :
- We can use the @autowired to mark a dependency which spring is going to resolve & inject.
- We can use this annotation with a constructor, setter or field injection

3) @ Qualifier :
- @ Qualifier is an annotation in spring boot used to specify a specific bean that should be injected into a field or constructor. This can be useful when multiple beans of the same type are available in the application context & you want to specify which one to use.
- The @qualifier annotation is used with the @ Autowired anotation to specify the specific bean to be injected.

4) @ Configuration :
- This annotation indicates that a class contains bean definitions for the application context. Spring container can use the class to generate bean instance & manage their lifecycle.
- classes annotated with this annotation are typically used to define application-level bean & their dependencies & to configure various aspects of the spring container such as the component scan base package & the properties to be used by the application.

5) @ComponentScan
- Used to enable component scanning in a spring boot appl".
- Allows to automatically detect & register beans in appl".
- Used to specify additional packages to scan or to exclude packages from scan.

6) @Bean:
- Used to define in a @Configuration class. When method is annotated with @Bean, the return value of method will be registered as a bean in the application context.

7) @Lazy
- Way to tell spring framework to lazily initialize a bean meaning that the bean will not be created until it is actually needed by application.
- Useful for optimizing the application's performance, as it can help reduce the no. of beans that are created & initialized at startup.
- @Lazy can be used to specify the bean should be created as a singletone, meaning only one instance of the bean will be created & shared among all components that depends on it.

8) @Value:
- Used for injecting property values into beans. It's compatible with constructor, setter, & field injection
- Injecting static values isn't useful, so we can use placeholder strings in @Value to wire values defined in external sources.

* Spring Framework
9) @Component:
- Another way to declare a bean is to ~~work~~ a mark a class with @component annotation. Doing this turns the class into spring bean at the auto-scan time.

10) @controller:
- Marks the class as a web controller, capable of handling the http requests. Spring will looks all the methods of the

class marked with the @ controller annotation & establish the routing table to know which methods serve which endpoints.

11) @Service :
- Mark a specialization of a @Component. It tells spring that it's safe to manage them with more freedom than regular component. Services have no encapsulated state.

12) @Repository :
- It is a specialization of the @component annotation allowing for implementation classes to be autodetected through classpath scanning.
- A repository is a mechanism for encapsulation storage, retrieval, and search behavior which emulates a collection of objects.

* Spring Boot
13) @ Enable Auto Configuration
- Used to enable the autto configuration features.
- Enables the auto-configuration of spring ApplicationContext by scanning the classpath component & registering the beans.

14) @springBootApplication :
- it combines the @EnableAutoconfiguration, @Configuration & @ComponentScan annotations in a spring boot application.
- i) @Enable AutoConfiguration - This enables spring boot's autoconfiguration mechanism. Auto-configuration refers to creating beans automatically by scanning the classpath.
- ii) @ComponentScan - To scan the current package & its sub-packages in order to identify annotated classes & configure them as spring beans. Annotations like @component, @Configuration, @Service, @Repository are specified on classes to mark them as spring beans.
- iii) @ Configuration - Used to tell that annotated class has one or more bean method definitions, so that spring can load & detect beans at runtime & make them available in spring container.

# * Spring MVC & Spring REST

15) @RequestMapping:
- When a request is sent to the web application, spring boot will use @RequestMapping annotation to determine which java method should handle the request.

16) @CookieValue:
- Used to get the value of any HTTP cookie without iterating over all the cookies fetched from the request. This annotation can be used to map the value of a cookie to the controller method parameter.

17) @CrossOrigin:
- Cross-Origin Resource sharing is a security concept that allows restricting the resources implementing in web browsers. It prevents the Javascript code producing or consuming the requests against different origin.
- Spring MVC provides support for enabling CORS for REST API endpoints so that the API consumers, such as web clients & mobile devices, can make calls to REST API's.

18) @GetMapping:
- The GET HTTP request is used to get single or multiple resources & @GetMapping annotation for mapping HTTP GET requests onto specific handler methods.

19) @PostMapping:
- The Post HTTP method is used to create a resource & @PostMapping annotation for mapping HTTP Post requests onto specific handler methods.

20) @PUTMapping:
- The PUT HTTP method is used to update the resource & @PutMapping annotation for mapping HTTP PUT requests onto specific handler methods.

21) @PatchMapping:
- The Patch Http method is used when you want to apply a

partial update to the resource & @PatchMapping annotation for mapping HTTP PATCH requests onto specific handler methods.

22) @DeleteMapping :-
- The DELETE HTTP method is used to delete the resources & @DeleteMapping annotation for mapping HTTP DELETE requests onto specific handler methods.

23) @ExceptionHandler
- Used to handle the specific exceptions & sending the custom responses to the client.

24) @InitBinder
- Plays the role to identify the methods which initialize the WebdataBinder.
- The @initBinder method supports many arguments as by @RequestMapping methods.

25) @MatrixVariable
- This indicates that a method parameter should be bound to a name-value pair within a path segment.

26) @PathVariable
- Used to retrieve data from URL path.
- This allows you to access dynamic values from URL & use them in your code.

27) @RequestAttribute
- It binds a method parameter to request attribute. It provides convenient access to the request attributes from a controller method. With the help of @RequestAttribute annotation, we can access objects that are populated on server-side.

28) @RequestBody :-
- It is used to bind HTTP request with an object in a method parameter.

29) @RequestHeader:
- Used to get the details about the HTTP request header.
- Use this annotations as a method parameter.

30) @RequestParam:
- Used to extract the query parameters from the URL. it is also known as a query parameter. It is most suitable for web application.

31) @RequestPart:
- This annotation associates a part of a multipart request with the method arguments, which is useful for sending complex multi-attributes data as payload e.g. JSON or XML.

32) @ResponseBody.
- it binds the method return value to the response body. it tells the spring boot framework to serialize a return an object into JSON & XML formate.

33) @ResponseStatus:
- This marks a method or exception class with the status code and reason message that should be returned.
- The status code is applied to the HTTP response when the handler method is invoked or whenever the specified exception is thrown.

34) @ControllerAdvice:
- @ControllerAdvice is a specialization of the @component annotation which allows to handle exceptions across the whole application in one global handling component.

35) @RestController:
- This is a convenience annotation for creating Restful controllers.
- It is specialization of @component & is autodetected through classpath scanning. It adds the @controller & @ResponseBody annotations. It converts the response to JSON or XML.

36) @RestControllerAdvice:
- Tells a controller that the object returned is automatically serialized into JSON & passed it to the HttpResponse Object. only need to return Java body Object instead of ResponseEntity

37) @SessionAttributes:
- Used to store model attributes in the Http servelet session between requests. It is a type-level annotation that declares the session attributes used by a specific controller.

38) @SessionAttributes:
- Used for pre-existing session attributes that are managed globally, outside the controller, eg. in filter, interceptor etc.
- Suitable for use cases like storing/accessing user authentication information etc.

* Spring cloud Annotations:

39) @EnableConfigServer
- Makes your spring boot application act as configuration server.

40) @EnableEurekaServer:
- Used to make your spring boot application acts as a Eureka Server.
- By default, the Eureka Server registers itself into the discovery.

41) @EnableCircuitBreaker:
- @EnableCircuitBreaker annotation will scan the classpath for any compatible Circuit Breaker implementation.

42) @HystrixCommand:
- To run method as Hystrix command synchronously you need to annotate method with @HystrixCommand annotation.
- Graceful degradation can be achieved by declaring name of fallbacks method in @HystrixCommand.

## Spring Data Access.

43) @Transactional
- @Transactional is a spring annotation that can be applied to methods or classes to indicate that the annotated code should be executed within a Transaction.

## Caching

44) @Cacheable:
- its a method level annotation. it defines a cache for a method's return value. The spring framework manages the requests & responses of the method to the cache that is specified in the annotation attribute.

45) @CachePut:
- This always lets the method execute. it is generally used when want caches to be updated with the result of method execution.

46) @CacheEvict.
- Used to remove one or more entries from a cache.
- When a method annotated with @CacheEvict is called, spring will remove the cached data associated with the specified cache name & keys along with the method execution.

47) @CacheConfig:
- Used at class level to share common cache related settings.
- All the methods of the class annotated with @Cacheable gets a common cache related settings provided by @CacheConfig.

## Spring Scheduler.

48) @scheduled
- The @Scheduled is a method-level annotation applied at runtime to mark the method to be scheduled.
- It takes one attribute from cron, fixedDelay, or fixedRate for specifying the schedule & execution in different formats.

## Spring Testing

49) @Bootstrapwith.
- It is a class-level annotation that you can used to configure how spring TestContext framework is bootstrapped.
- You can use @Bootstrapwith to specify a custom TestContextBootstrapper.

50) @ContextConfiguration:
- it defines class-level metadata that is used to determine how to load & configure an ApplicationContext for integration tests.

51) @WebAppConfiguration:
- it is a class-level annotation that is used to declare that application Context loaded for an integration test should be a ~~WebAppConfiguration~~ WebApplicationContext.

52) @Timed:
- An annotation for ~~making~~ marking a method, construct, or class as Timed. This ~~not~~ metric will be registered in the application MetricRegistry.

53) @Repeat:
- Test annotation for use with JUnit 4 to indicate that a test method should be invoked repeatedly.
- The scope of execution to be repeated includes execution of the test method itself as well as any setup or tear down of the test fixture.

54) @Commit:
- @Commit is a test annotation that is used to indicate that a test-managed transaction should be committed after the test method has completed.

55) @RollBack:
- @RollBacks is a test annotation that is used to indicate whether a test-managed transaction should be rolled back after the test method has completed.

56) @DirtiesContext:
- @DirtiesContext is a spring testing annotation. It indicates the associated test or class modifies the ApplicationContext.
- it tells the testing frameworks to close & recreate the context for later tests.

57) @BeforeTransaction:
- This indicates that the annotated void method should be run before a transaction is started, for test methods that have been configured to run within a transaction by using spring's @Transactional annotation.

58) @AfterTransaction:
- indicates that the annotated void method should be run after a transaction is ended; for test methods that have been configured to run within a transaction by using spring's @Transactional annotation.

59) @Sql:
- Used to annotate a test class or test method to configure SQL scripts () & statements () to be executed against a given database during integration tests.

60) @SqlConfig:
- @SqlConfig defines metadata that is used to determine how to parse & execute SQL scripts configured via the @sql annotation.

61) @SqlGroup:
- It is a container annotation that aggregates several @sql annotations.
- You can use @SqlGroup, natively to declare several nested @sql annotations.

62) @SpringBootTest:
- This annotation loads the complete spring application context.

- In contrast, a test slice annotation only loads beans required to test a particular layer. & because of this, we can avoid unnecessary mocking & ~~di~~ ~~st~~ side effects.

63) @DataJpaTest.
- Tests annotated with @DataJpaTest are transactional & roll backs at the end of each test. They also used an embedded in-memory database.

64) @DataMongoTest:
- Annotation that can be used for a MongoDB Test that focuses only on MongoDB components.
- Using this annotation will disable full auto-configuration & instead apply only configuration relevant to MongoDB tests.

65) @WebMvcTest:
- Used for spring MVC tests. it disables full auto-configuration & instead applies only configuration relevant to MVC tests. it auto configure MockMvc instance as well.

66) @AutoConfigureMockMVC:
- Annotation that can be applied to a test class to enable & configure auto-configuration of MockMVC.

67) @MockBean:
- We can use the @MockBean to add mocks objects to spring application context. The mock will replace any existing bean of the same type in the application context.

68) @JsonTest:
- Annotation for a JSON test that focuses only on JSON Serlization.

69) @TestPropertySource:
- It is a class-level annotation that is used to specify which properties files should be loaded when running the test class.