Annotations of spring → shiva

1)* @Auto wired :- Used for automatic injection of beans. Spring @Qualifier is used in conjuction with this to avoid confusion when we have 2 or more bean.

2)* @configuration :- Used to indicate that a class declares one or more ~~bean~~ @Bean methods. These classes are processed by the spring container to generate bean definations & service requests for those beans at runtime

3). @Bean :- Indicates that a method produces a bean to be managed by spring container and it is most important.

4). @componentScan :- configures component scanning directives for use with @configuration classes.

5). @component :- Indicates that an annotated class is a component Such classes are considered as candidates for auto detection when using annotation based configuration & class path scanning

6). @service :- Indicates that an annotated class is a service. this annotation serves as a specialization of @component, allowing for implementation classes to be auto detected.

dao
Entity with all annotation.
cross origin

7). @Repository: ~~Indicates~~ serves as a specialization of @component and advisable to use with "dao".

8). @Transactional: is the spring declarative transaction management annotation.

9). @Enable websecurity:- is used with @configuration class to have the spring security configuration.

10). @Qualifier:- It is used along with @autowired annotation and used when you need more control of the dependancy injection.

11). @value:- This annotation is used at the field, constru-ctor parameter & method parameter level.

12). @Spring Boot Application:- Used on the application class while setting up a spring Boot ~~Application~~ project.

13). Request Mapping: Used at both class & method level. Used to map web requests onto specific handler class & handler methods.

14) @CrossOrigin:- this is used at both class & method level to enable cross origin requests.

15) @Get Mapping:- This annotation is used for mapping HTTP GET requests onto specific handler methods

16). @Post Mapping:- Maps HTTP post requests.

17). @PutMapping:- Mapping HTTP PUT requests

18). @PatchMapping:- Mapping HTTP PATCH requests
Alternative to @RequestMapping (method = Request-
- Method. PATCH).

19). @DeleteMapping:- Mapping HTTP DELETE requests.

20). @ExceptionHandler:- To handle exception at the controller level. Defines the class of exception it will Catch.

21). @InitBinder:- Plays the role of identifying the methods which initialize the webdatabinder which is a DataBinder that binds the request parameter to JavaBean objects.

22). @Mappings & @Mapping:- Used on fields. the @Mapping annotation is a meta annotation that indicates a webmapping annotation. For mapping different field names, use @Mappings.

23). @PathVariable:- Annotates request handler method arguments. When URI value acts as a para-
-meter. You can specify that parameterusing @Path Variable.

24). @RequestAttribute:- Binds the request attribute to a handlermethod parameter. It is used to access the objects which are populated on server side.

25). @RequestBody:- Indicates thata method parameter should be baund to a HTTP request body.

26)@RequestHeader :- To map controller parameter to request header value.

27) @RequestParam:- Used with @RequestMapping to retrieve the URL parameter & map it to the method argument

28) @RestController:- Used at class level. By using this One no longer need to add @ResponseBody.

29). @EnableConfigServer:- class level. while developing a project with a number of services, you need to have a centralized & straightforward manner to configure & retrieve the configurations about all the services that you are going to develop

30). @EnableEurekaServer:- Spring Boot has made it easy to design a Eureka Server by just annoting the entry class.

31). @EnableDiscoveryClient:- added with @EnableEureka Server to the application entry point.

32)@ Transactional:- This annotation is simply a metadata that can be consumed by some runtime infrastr-ucture. the mere presence of it is not enough to activate transactional behaviour.

33). @Scheduled: Used on methods along with the trigger metadata.

34)@WebAppConfiguration: Used to declare that the ApplicationContext loaded for an integrated test should be WebApplication Context

35). @ContextConfiguration:- class level annotation. Declares the the annoted classes that will be used to Load the Context

36). @Repeat: Used if uses want to run a test method several times.

to
Specify ~~that method~~
37). @Timed:- Used if user want to finish ~~its~~ test method ~~and~~ in given period of time in 'ms'

38). @Commit:- After execution of a testmethod, the transaction of the transactional test method can be committed using this

39). @Roll back:- Indicates whether the transaction of a transactional test method must be rolled back after test completes execution.

40). @Before Transaction:- Methods with ~~these~~ This annotation indicates that they ~~should~~ should be executed before any transaction starts executing.

41)@After Transaction:- This annotation indicate that the method should be executed after a transaction ends for test methods.

42). @Sql:- Used on a test class or test method to run sql scripts against a database...

43). @Sql Config:- this annotation defines the metadata that is used to determine how to parse & execute SQL scripts configured via @sql annotation.

44). @Sql Group:- Can hold several @sql annotations as it is a container and also, can declare nested @sql annotation.

45).@Spring BootTest:- Used to start the spring context for integration tests. Brings up full auto configuration context

46). @DataJpaTest:- Will only provide auto Configuration required to test spring data JPA.

~~47) @Web~~