SPRING ANNOTATIONS

SPRING BOOT ANNOTATIONS

==========

-> Spring boot annotations is a form of metadata that provides data about a program. It doesn't affect the application

we are developing in any way.

CORE SPRING FRAMEWORK ANNOTATIONS

---

1. @Required - It applies to a bean setter method. It indicates that the annotated bean must be populated at the configuration time with the required property or else it will throw an exception i.e., BeanInitializationException.

2. @Autowired - Spring provides annotation-based auto-wiring by providing @Autowired annotation. It is used to autowire spring bean with setter methods, instance variable and constructor. When we use @Autowired annotation the spring container autowires the bean by matching datatypes.

3. @Configuration - It is a class level annotation. The class which is annotated with @Configuration is used by the Spring Container as the source of bean definitions.

4. @ComponentScan - It is used when we want to scan a package for beans. It is used with @Configuration annotation. We can also specify the base packages to scan for Spring Components.

5. @Bean - It is a method level annotation. It is an alternative of the XML <bean> tag. It tells the method to produce a bean to be managed by the Spring Container.

SPRING FRAMEWORK STREOTYPE ANNOTATIONS

---

1. @Component - It is a class level annotation. It is used to mark a java class as a bean. A java class annotated with @Component annotation is found during the classpath. Spring framework picks it up and configures it in the application context into a Spring bean.

2. @Controller - It is a class level annotation. It is a specialization of @Component. It is used to mark a java class as a web request handler. It is often used to serve web pages. By default, it returns a string which indicates which route to redirect. It is usually used with @RequestMapping.

3. @Service - It is a class level annotation. It tells the Spring that the class contains the business logic.

4. @Repository - It is a class level annotation. The repository is a DAO (Data Access Object) that access the database directly. The repository does all the operations related to database.

SPRING BOOT ANNOTATIONS

---

1. @EnableAutoConfiguration - It auto-configures the bean that is present in the classpath and configures it to run the methods. The use of this annotation is reduced in the Spring Boot 1.2.0 release as the developers provided an alternative for this annotation i.e., @SpringBootApplication.

2. @SpringBootApplication - It is a combination of three annotations—@EnableAutoConfiguration, @ComponentScan, @Configuration

SPRING MVC AND REST ANNOTATIONS

---

1. @RequestMapping - It is used to map web requests. It has many optional elements like consumes, header, method, name, param, path, produces and value. It can be used for class as well as methods.

2. @GetMapping - It maps the HTTP GET request on the specific handler method. It is used to create a web service endpoint which fetches.

3. @PostMapping - It maps the HTTP POST request on the specific handler method. It is used to create a web service endpoint which creates.

4. @PutMapping - It maps the HTTP PUT request on the specific handler method. It is used to create a web service endpoint which creates or updates.

5. @DeleteMapping - It maps the HTTP DELETE request on the specific handler method. It is used to create a web service endpoint which deletes.

6. @PatchMapping - It maps the HTTP PATCH request on the specific handler method.

7. @RequestBody - It is used to bind HTTP request with an object in the method parameter. Internally it uses HTTP MessageConvertor to convert the body of the request. When we annotate the method parameter with @RequestBody, the Spring framework binds the incoming http request body with that parameter.

8. @ResponseBody - It is used to bind the method return value to the response body. It tells the Spring framework to serialize and return an object in JSON or XML format.

9. @PathVariable - It is used to extract values from the URI. It is most suitable for RESTful web service where the URL contains the path variables. We can define multiple @PathVariable in a method.

10. @RequestParam - It is used to extract the query parameters from the URL. It is most suitable for web applications. It can specify default values if the query parameters are missing from the URL.

11. @RestController - It is considered as a combination of @Controller and @ResponseBody. The @RestController annotation is itself annotated with @ResponseBody annotation. So it eliminates the need for annotating each method with @ResponseBody.

OTHER ANNOTATIONS USED IN PROJECT

==========

1. @EnableJpaRepositories - It enables the Jpa repositories that contain in the given packages.

2. @EntityScan - used to tell the Spring boot to where to scan the entities of the application.

3. @EnableSwagger2 - Swagger2 is an open-source project which is used to generate REST APIdocuments for RESTful web services. It also provides a user interface to access our RESTful web services via web browser. To enable this Swagger2 in our Spring Boot Application we have to use this annotation @EnableSwagger2.

4. @SpringBootServletInitializer - It's an interface to run the Spring application from a traditional WAR deployment. It binds Servlet, Filter and ServletContextInitializer beans from the application context to the server. The application can be run both by deploying the war on a Tomcat server and executing it as a self-executable web archive with embedded Tomcat.

5. @Valid - It comes from Java Validation API. It is used on method parameters and fields. We can use it on complex objects whenever there is a need to validate them.

6. @Validation - It comes from Spring Framework Validation. It is a variant of @Valid which allows group validations. It is used on method and method parameters when using group validations. It can be used on classes to support method parameter constraint validations.

7. @Api - Marks a class as a Swagger resource. By default, Swagger core will only include and introspect those classes which are annotated with @Api annotation and ignore all other classes.

8. @ApiOperation - This annotation is used to describe the exposed REST API. It describes an operation or typically a HTTP method against a specific path.

9. @ApiResponse - The @ApiResponse describes a concrete possible response.

10. @ApiResponses - This annotation is used to describe the expected responses for the REST API. The @ApiResponse describes a concrete possible response. It cannot be used directly on the method and needs to be included in the array value of @ApiResponses (whether there's one response or more).

11. @ApiParam - This annotation is used to describe the exposed REST API. It takes the following parameters-

value — The value is a short description of the parameter

required — If the parameter is optional or required.

defaultValue — Specify defaultValue of the parameter.