Spring Annotations

@SpringBootApplication

It is class-level annotation. It is combination of @Configuration, @EnableAutoConfiguration, @ComponentScan

@Configuration

It is class-level annotation. Used to declare a class with one or more @Bean methods. These Beans are managed by Spring container.

@ComponentScan

It is class-level annotation. To scan all the spring components in the current package and its sub-packages.

@Bean

It is method-level annotation. Used to specify that a method returns a bean to be managed by Spring IoC container.

@Component

It is a class-level annotation. It is used to mark a Java class as a bean. It means that Spring framework will auto detect these classes for dependency injection.

@Autowired

It is method-level annotation. It is used for automatic dependency injection. To automatically inject object into fields of the class.

@Qualifier

This annotation is used to specify the exact bean to be injected when multiple beans of the same type are present.

@Controller

It is a class-level annotation. It's used to mark a class as a web request handler. A controller class handles incoming HTTP requests.

@RestController

It is a class-level annotation. This annotation is a special controller used in RESTful Web services, and it's the combination of @Controller and @ResponseBody annotation

@ResponseBody

To send data to the response. @ResponseBody annotation tells a controller that the object returned is automatically serialized into JSON and passed back into the HttpResponse object t. When you use the @ResponseBody annotation on a method, Spring converts the return value into JSON and writes it to the HTTP response automatically. Each method in the Controller class must be annotated with @ResponseBody.

@RequestMapping

It is used to map requests to controller methods. It has various attributes to match by URL, HTTP method, request parameters, headers, and media types. You can use it at the class level to express shared mappings or at the method level to narrow down to a specific endpoint mapping.

@GetMapping

@GetMapping annotation is used for mapping HTTP GET requests onto specific handler methods.

Specifically, @GetMapping is a composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.GET).

@PostMapping

The POST HTTP method is used to create a resource and @PostMapping annotation for mapping HTTP POST requests onto specific handler methods.

Specifically, @PostMapping is a composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.POST).

@PutMapping

The PUT HTTP method is used to update the resource and @PutMapping annotation for mapping HTTP PUT requests onto specific handler methods.

Specifically, @PutMapping is a composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.PUT).

@DeleteMapping

The DELETE HTTP method is used to delete the resource and @DeleteMapping annotation for mapping HTTP DELETE requests onto specific handler methods.

Specifically, @DeleteMapping is a composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.DELETE).

@PatchMappings

The PATCH HTTP method is used when you want to apply a partial update to the resource and @PatchMapping annotation for mapping HTTP PATCH requests onto specific handler methods.

Specifically, @PatchMapping is a composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.PATCH).

@Service

the business logic resides within the service layer so we use the @Service Annotation to indicate that a class belongs to that layer. It is also a specialization of @Component Annotation like the @Repository Annotation. One most important thing about the @Service Annotation is it can be applied only to classes. It is used to mark the class as a service provider.

@Repository

@Repository Annotation is a specialization of @Component annotation which is used to indicate that the class provides the mechanism for storage, retrieval, update, delete and search operation on objects.

@EnableAutoConfiguration

The @EnableAutoConfiguration annotation enables Spring Boot to auto-configure the application context. Therefore, it automatically creates and registers beans based on both the included jar files in the classpath and the beans defined by us.

For example, when we define the spring-boot-starter-web dependency in our classpath, Spring boot auto-configures Tomcat and Spring MVC. However, this auto-configuration has less precedence in case we define our own configurations.

@EnableAutoConfiguration annotation tells Spring Boot to “guess” how you want to configure Spring, based on the jar dependencies that you have added.

Since the spring-boot-starter-web dependency added to classpath leads to configuring Tomcat and Spring MVC, the auto-configuration assumes that you are developing a web application and sets up Spring accordingly.

@EnableWebMvc

Imports spring webmvc related configurations. With Spring Boot's auto configure, this annotation can be omitted.

@Value

Spring @Value annotation is used to assign default values to variables and method arguments.

The @Value annotation is mostly used to get value or inject a value for specific property keys from the property files, environment variables, system properties, and command line arguments.

@PropertySource

Spring @PropertySource annotation is used to provide properties file to Spring Environment.

This annotation is used with @Configuration classes.

To provide our own properties file into the classpath we can use @PropertySource and to read the values form the property file we use @Values

@Transient

This annotation is used to ignore a column in a database. It will not be considered in Entity.