**Spring Annotations**

* Spring Annotations are a form of metadata that provides data about a program.
* Annotations- are used to provide supplemental information about a program.
  + - It does not have a direct effect on the operation of the code they annotate.
    - It does not change the action of the compiled program.

1. **@Required**:

* It applies to the bean setter method.
* It indicates that the annotated bean must be populated at configuration time with the required property, else it throws an exception BeanInitilizationException.

1. @**RestController:**

* It can be considered as a combination of @Controller and @ResponseBody annotations.
* The @RestController annotation is itself annotated with the @ResponseBody annotation.
* It eliminates the need for annotating each method with @ResponseBody.

1. **@Controller**:

* The @Controller is a class-level annotation.
* It is a specialization of @Component. It marks a class as a web request handler.
* It is often used to serve web pages.
* By default, it returns a string that indicates which route to redirect.
* It is mostly used with @RequestMapping annotation.

1. **@Configuration**:

* It is a class-level annotation.
* The class annotated with @Configuration used by Spring Containers as a source of bean definitions.

1. **@Bean**:

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

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 is found during the classpath.
   * The Spring Framework pick it up and configure it in the application context as a Spring Bean.
2. **@ComponentScan:** 
   * It is used when we want to scan a package for beans.
   * It is used with the annotation @Configuration.
   * We can also specify the base packages to scan for Spring Components.
3. **@Service:**
   * It is also used at class level.
   * It tells the Spring that class contains the business logic.
4. **@Repository**:
   * It is a class-level annotation.
   * The repository is a DAOs (Data Access Object) that access the database directly.
   * The repository does all the operations related to the database.
5. **@Autowired:** 
   * Spring provides annotation-based auto-wiring by providing @Autowired annotation.
   * It is used to autowire spring bean on setter methods, instance variable, and constructor.
   * When we use @Autowired annotation, the spring container auto-wires the bean by matching data-type.
6. **@Primary:**

* It is used to give higher preference to a bean when there are multiple beans of the same type.

1. **@Qualifier:**

* It is used to eliminate the issue of which bean needs to be injected.

1. **@SpringBootApplication**: It is a combination of three annotations @EnableAutoConfiguration, @ComponentScan, and @Configuration.
2. **@ControllerAdvice:**

* It is useful to handle exceptions when you have multiple Spring REST API controllers doing a lot of different work.
* That means when writing any application you encounter exceptions and to handle them at each method level is tedious and not optimal.
* So in order to overcome that, spring has introduced the concept of @ControllerAdvice, which will intercept all the controllers and look for the exceptions thrown.
* This is at a global level means you only have one @ControllerAdvice for each application and it will intercept the exceptions thrown by the controllers in that particular application context.

1. @ExceptionHandler:

* This annotation is used to handle specific exceptions.
* The annotated method is invoked when the specified exceptions are thrown from a @Controller.
* We can define these methods either in a @Controller class or in @ControllerAdvice class.

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 Spring Boot 1.2.0 release because developers provided an alternative of the annotation, i.e. @SpringBootApplication.

1. **@RequestMapping**:

* It is used to map the web requests.
* It has many optional elements like consumes, header, method, name, params, path, produces, and value.
* We use it with the class as well as the method.

1. **@GetMapping**:

* It maps the HTTP GET requests on the specific handler method.
* It is used to create a web service endpoint that fetches It is used instead of using: @RequestMapping(method = RequestMethod.GET)

1. **@PostMapping:**

* It maps the HTTP POST requests on the specific handler method.
* It is used to create a web service endpoint that CREATES
* It is used instead of using @RequestMapping(method = RequestMethod.POST)

1. **@PutMapping:**

* It maps the HTTP PUT requests on the specific handler method.
* It is used to create a web service endpoint that creates or UPDATES
* It is used instead of using @RequestMapping(method = RequestMethod.PUT)

1. **@DeleteMapping:**

* It maps the HTTP DELETE requests on the specific handler method.
* It is used to create a web service endpoint that deletes a resource.
* It is used instead of using @RequestMapping(method = RequestMethod.DELETE)

1. **@PatchMapping:**

* It maps the HTTP PATCH requests on the specific handler method.
* It is used instead of using @RequestMapping(method = RequestMethod.PATCH)

1. **@RequestBody:**

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

1. **@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 and XML format.

1. @PathVariable:

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

1. **@RequestParam:**

* It is used to extract the query parameters form the URL.
* It is also known as a query parameter.
* It is most suitable for web applications.
* can specify default values if the query parameter is not present in the URL.

1. **@RequestHeader:**

* It is used to get the details about the HTTP request headers.
* We use this annotation as a method parameter. The optional elements of the annotation are name, required, value, defaultValue.
* For each detail in the header, we should specify separate annotations.
* We can use it multiple time in a method.

1. **@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 the server-side.