**Annotations Used in Alliedacademies Application**

1. **@SpringBootApplication:**

**Many Spring Boot developers like their apps to use auto-configuration, component scan and be able to define extra configuration on their "application class". A single @SpringBootApplication annotation can be used to enable those three features, that is:**

**@EnableAutoConfiguration: enable Spring Boot’s auto-configuration mechanism**

**@ComponentScan: enable @Component scan on the package where the application is located (see the best practices)**

**@Configuration: allow to register extra beans in the context or import additional configuration classes**

**The @SpringBootApplication annotation is equivalent to using @Configuration, @EnableAutoConfiguration, and @ComponentScan with their default attributes.**

1. **@PropertySource**

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

**This annotation is to be used in conjunction with Java based configuration and the @Configuration annotation.**

**We can use @value annotation inorder to read the values from properties file.**

1. **@Autowired**

**Spring @Autowired annotation is used for automatic dependency injection. Spring framework is built on dependency injection and we inject the class dependencies through spring bean configuration file.**

1. **@Configuration**

**Use @Configuration annotation on top of any class to declare that this class provides one or more @Bean methods and may be processed by the Spring container to generate bean definitions and service requests for those beans at runtime.**

1. **@ComponentScan**

**The @ComponentScan annotation is used with the @Configuration annotation to tell Spring the packages to scan for annotated components.**

**@ComponentScan also used to specify base packages and base package classes using thebasePackageClasses or basePackages attributes of @ComponentScan.**

1. **@Component**

**The @Component annotation marks a java class as a bean so the component-scanning mechanism of spring can pick it up and pull it into the application context.**

1. **@Controller**

**@Controller annotation marks a class as a Spring Web MVC controller. It too is a @Component specialization, so beans marked with it are automatically imported into the DI container. When we add the @Controller annotation to a class, we can use another annotation i.e. @RequestMapping; to map URLs to instance methods of a class.**

1. **@Service**

**The @Service annotation is also a specialization of the component annotation. It doesn’t currently provide any additional behavior over the @Component annotation, but it’s a good idea to use @Service over @Component in service-layer classes because it specifies intent better.**

1. **@Repository**

**The @Repository annotation is a specialization of the @Component annotation with similar use and functionality. In addition to importing the DAOs into the DI container, it also makes the unchecked exceptions (thrown from DAO methods) eligible for translation into Spring DataAccessException.**

1. **@Bean**

**@Bean is used to explicitly declare a single bean, rather than letting Spring do it automatically for us.**

**Another big difference is that @Component is a class level annotation where as @Bean is a method level annotation and ,by default, name of the method serves as the bean name.**

1. **@RestController**

**@RestController is a specialized version of the controller. It includes the @Controller and @ResponseBody annotations and as a result, simplifies the controller implementation.**

1. **@Table**

**The @Table annotation allows you to specify the details of the table that will be used to persist the entity in the database.**

**The @Table annotation provides four attributes, allowing you to override the name of the table, its catalogue, and its schema, and enforce unique constraints on columns in the table.**

1. **@Column**

**Represents the column name in the table.**

1. **@Id**

**Represents the primary key in the table.**

1. **@RequestMapping**

**@RequestMapping can be applied to the controller class as well as methods.**

**This annotation is used to map web requests onto specific handler classes and/or handler methods.**

1. **@PathVariable**

**The @PathVariable annotation in spring binds the URI template variables to the handler method parameters of a controller. A URI template can contain one or more path variables enclosed by the curly braces ({ }) .**

1. **@RequestParam**

**The @RequestParam annotation binds a web request parameter (i.e. query string) to a method parameter in a controller.**

1. **@PathParam**

**The @PathParam annotation is a type of parameter that you can extract for use in your resource class. URI path parameters are extracted from the request URI, and the parameter names correspond to the URI path template variable names specified in the @Path class-level annotation.**

1. **@QueryParam**

**The @QueryParam annotation is a type of parameter that you can extract for use in your resource class. Query parameters are extracted from the request URI query parameters.**

1. **@Consumes**

**The @Consumes annotation is used to specify the MIME media types of representations a resource can consume that were sent by the client.**

1. **@Produces**

**The @Produces annotation is used to specify the MIME media types of representations a resource can produce and send back to the client: for example, "text/plain".**

1. **@Provider**

**The @Provider annotation is used for anything that is of interest to the JAX-RS runtime, such as MessageBodyReader and MessageBodyWriter. For HTTP requests, the MessageBodyReader is used to map an HTTP request entity body to method parameters. On the response side, a return value is mapped to an HTTP response entity body by using a MessageBodyWriter. If the application needs to supply additional metadata, such as HTTP headers or a different status code, a method can return a Response that wraps the entity and that can be built using Response.ResponseBuilder.**

1. **@ModelAttribute**

**The @ModelAttribute is an annotation that binds a method parameter or method return value to a named model attribute and then exposes it to a web view.**

1. **@RequestBody and @ResponseBody**

**@RequestBody and @ResponseBody annotations are used to bind the HTTP request/response body with a domain object in method parameter or return type. Behind the scenes, these annotation uses HTTP Message converters to convert the body of HTTP request/response to domain objects.**

1. **Entity**

**Specifies that the class is an entity. This annotation can be applied on Class, Interface of Enums.**