**Spring Boot Annotations**

1. **@SpringBootApplication**

* Firstly it use to mark java class as starting point of application
* And also this annotation is equivalent to three annotation which is

**@EnableAutoConfiguration + @ComponentScan + @Configuration**

* Now in **@EnableAutoConfiguration** will Enable the Auto configuration means it will Automatically Configure if any Spring configuration based on the Jar dependency If any present in classpath so it will auto configured in our project.(**We can use attribute “exclude” for exclude the specific auto configuration**)
* For Example if we found spring-boot-starter-web Dependency in pom.xml then Spring boot application will automatically configure the web application context and JPA persistence context.
* **@ComponentScan** this is just a simple annotation where it will scan our All the Components is present in Specified package for example it will scan like @Component, @Service, @Repository and @Controller and then it will register as a Bean in Spring Application Context.
* If we need to scan out of the package so in that case we need to add

@ComponentScan(basePackage = {“otherPackageComponent”}) ) in our Configuration class.

* **@Configuration** will help us to identifying Beans or class which will returning Beans for Spring Dependency Injection which will manage by the IOC Container. **In simple word this Annotation marks a class as source of Bean Definition.**

1. **@Bean**

* It will tell us ````````````````````````````````````````````````````````````````````````````````````````````````````````````+this will returning the Bean to be managed by Spring Container.
* It will create a managed instance of particular return type of method.

1. **@Component**

* This annotation will tell us to this class is the Component which will manage by spring and we will use it with autowiring.

1. **@Autowired**

* This annotation is provide a feature which we can achieve Dependency Injection Automatically.
* It will inject automatically the Object from Spring Container by matching type of this particular Object.
* It help us to autowire the bean without creating an object using new keyword.
* We can applied to Constructor, Field, or Setter Method.

**@Component**

* **@Controller**
* **@Service**
* **@Repository**

1. **@Controller**

* This Annotation will use for define the class as component which will used a Restful web service to handle Requests and Responses.
* **@Controller –** Spring framework will always looking for HTML or view page for the returning string from method.
* If we annoted @Controller it will always expect as return type Model and View
* **@RestController** – It will return any thing we have in our method.

1. **@Service**

* This Annotation will use for define the class as component which will used for Service Layer.
* In Simple meaning this annotation is use for having the Business Logic of the application it this Specific Class.

1. **@Repository**

* This Annotation will use for define the class as component which will used for Represent the DAO layer of the application to deals with the database CRUD operations.

1. **@Qualifier**

* This annotation will Qualify the Specific Bean if there is more than One Bean is available.
* For example @Autowired annotation will check the type of Object and if there is two or more Same type of object is present then we will use @Qualifier annotation with Autowiring to avoid ambiguity problem.

1. **@Lazy**

* In spring boot a while defining spring bean then it will create Spring Beans in Container and now if we want this Spring bean is created as per requirement while we inject this bean then it will have to create then we will use @Lazy annotation.

**Stereotype Annotation** @Component, @Service, @RestController, @Controller, @Repository

1. **@Value**

* It will use for to inject the value explicitly from application.properties file with define key as attribute
* For example @Value(${server.port}) – it will inject the port no. of application in define variable.

1. **@PropertySource**

* This annotation is use in Class level for defining the source of the properties which we will use by @Value attribute.

1. **@ConfigurationProperties**

* It will use for configure the specific properties form properties file.
* For Example:

In application.properties file

devname.name = abcd

devname.age = 22

In somClass.class file

@ConfigurationProperties(prefix = “devname”)

public class somClass{

private String name;

private String age;

}

* In this Example devname is the property in properties file and we will use or inject directly in Variable.

1. **@Profile**

* This will use for defining the which environment profile use for application.
* spring.active.profile = dev

1. **@Scope**

* If we define the scope of the bean we will use this annotation.
* For example we use this @Scope(“singletone”) or @Scope(“prototype”)

**Scope of Bean :-**

**@Scope** – It will tell scope of the bean means if we want to declare the scope of the Bean.

In that commonly we use singleton or prototype.

Type of Scope

**Singleton** – by default scope of spring bean – it will return same object reference every time in entire application.

**Prototype** – it will return always new object reference – it will create new object every time

**Request** – for HHTP request objects – it will create new instance for each HTTP request.

**Session** – for only session objects – it will create single instance of the bean per web socket session

1. @**RestController**

This is specialized version of Controller that include @ResponseBody means it directly return data rather than view.

It will return data in XML and Json format.

1. **@RequestMapping**
2. **@GetMapping**
3. **@PostMapping**
4. **@PutMapping**
5. **@DeleteMapping**
6. **@RequestBody**

It will bind the data from request into Java Object.

1. **@PathVariable**
2. **@RequestParam**
3. **@ControllerAdvice**

Provide global exception handling.

1. **@ExceptionHandler**
2. **@Entity**

* It will tell to JPA or Hibernate this class will perform to Database Related Operations.

1. **@Table**

* It will define to JAP or Hibernate to create the Table as the name provide and if we did not use this annotation the @Entity annotation will do automatically as per class name

1. **@Column**
2. **@GeneratorType**
3. **@Transactional**
4. **@Data**
5. **@CrossOrigin**
6. **@Secured**
7. **@PreAuthorize**
8. **@PermitAll**
9. **Caching Annotation**
10. **@EnableCaching**
11. **@Cacheable**
12. **@CachePut**
13. **@CacheEvict**

**AOP(Aspect Oriented Annotation)**

1. **@Aspect**
2. **@Pointcut**
3. **@AfterRunning**
4. **@AfterThowing**
5. **@Around**
6. **@Before**