**Spring Annotations:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Annotation** | **Description** | **Target** | **Retention Policy** |
| **Core Spring Framework Annotations** | | | |
| @Required | * It indicates affected bean must be populated with required property at configuration time otherwise it throws BeanInitializationException.   Ex: @Required public void setAge(Integer age){} | Method | Runtime |
| @Autowired | It injects dependency implicitly.  Ex: @Autowired private Person person; | Field, Method, Constructor, Parameter | Runtime |
| @Qualifier | * This annotation is used along with @Autowired. when you need more control on dependency injection. * It can be specified on individual constructor arguments or method parameters. * It also used to avoid confusion which occurs when you create more than one bean of same type.   Ex: https://www.logicbig.com/how-to/code-snippets/jcode-spring-framework-qualifier.html | Field, Method, Class (Type), Constructor, Parameter | Runtime |
| @ComponentScan | * Make Spring scan the package for the @Configuration classes. |  |  |
| @Configuration | * Marks class as source of bean definition * Alternative for xml configuration class | Class (Type) | Runtime |
| @Bean | * indicates that a method produces a bean to be managed by the Spring container. * alternative of <bean> tag in xml | Method | Runtime |
| @Lazy | * Makes @Bean or @Component be initialized on demand rather than eagerly.   Ex: @Component @Lazy public class BeanThree | Field, Method, Class (Type), Constructor, Parameter | Runtime |
| @Value | * indicates a default value expression for the field or parameter, typically something like   Ex: @Value("${aBean.age ?: 21}") private int age; | Field, Method, Constructor, Parameter | Runtime |
| Spring Framework Stereotype Annotations | | | |
| @Component | Used to indicate spring component.  It marks class as component so component scan can add it application context.  Ex: @Component("thisIsBeanA") public class BeanA {} | Class (Type) | Runtime |
| @Controller | Used to indicate that the class is spring controller.  Used to identify controllers in Spring MVC and Spring WebFlux.  Ex: @Controller public class MyController{} | Class (Type) | Runtime |
| @Service | This specialized form of @Component.  Used to marks the class which performs some service such as execute business logic, call external APIs, perform calculations.  Ex: @Service public class BusinessServiceImpl{} |  |  |
| @Repository | Used to mark java classes that accesses database.  It has automatic translational feature for example when exception occurs there is handler for exception no need to add try catch block.  Ex: @Repository public class DetailsRepositoy | Class (Type) | Runtime |
| Spring Boot Annotations | | | |
| @EnableAutoConfiguration | It tells spring boot to start adding beans based on the class path settings, other beans, various property settings.  It is created using following annotations:  @Target(value=TYPE)  @Retention(value=RUNTIME)  @Documented  @Inherited  @AutoConfigurationPackage  @Import(value=AutoConfigurationImportSelector.class)  Fields   |  |  | | --- | --- | | Modifier and Type | Fields and Description | | static String | ENABLED\_OVERRIDE\_PROPERTY |   Optional Elements   |  |  | | --- | --- | | Modifier and Type | Fields and Description | | Class<?>[] exclude | Exclude specific auto-configuration classes such that they will never be applied. | | String[] excludeName | Exclude specific auto-configuration class names such that they will never be applied. | | Class (Type) | Runtime |
| @SpringBootApplication | It is used on the application while setting up the spring boot project.  It does a component scan but it will only scans its sub packages therefore the class annotated with @SpringBootApplication must kept in base package so it can scan all the sub packages of spring boot project.  It is created using following annotations:  @Target(value=TYPE)  @Retention(value=RUNTIME)  @Documented  @Inherited  @SpringBootConfiguration  @EnableAutoConfiguration  @ComponentScan(excludeFilters={@ComponentScan.Filter(type=CUSTOM,classes=TypeExcludeFilter.class),})  Optional Elements   |  |  | | --- | --- | | Modifier and Type | Fields and Description | | Class<?>[] exclude | Exclude specific auto-configuration classes such that they will never be applied. | | String[] excludeName | Exclude specific auto-configuration class names such that they will never be applied. | | Class<?>[] scanBasePackageClasses | Type-safe alternative to scanBasePackages() for specifying the packages to scan for annotated components. | | String[] scanBasePackages | Base packages to scan for annotated components. | | Class (Type) | Runtime |
| @SpringBootConfiguration | Indicates that a class provides spring boot application configuration. an alternative to spring 's configuration.  So that configuration found automatically.  It is created using following annotation:  @Target(value=TYPE)  @Retention(value=RUNTIME)  @Documented  @Configuration | Class (Type) | Runtime |
| **Spring MVC and REST Annotations** | | | |
| @RequestMapping | It is used to map web request onto specific handler class or handler method.  When it used on class level it creates base uri for which the controller will be used.  When it used on method level it will give you URI on which handler methods will be executed.  If sometime you want to perform different operations using same URI. In such situations you can use method attribute of RequestMapping.   |  |  | | --- | --- | | Modifier and Type | Fields and Description | | String[] consumes | The consumable media types of the mapped request, narrowing the primary mapping. | | String[] headers | The headers of the mapped request. | | RequestMethod[] | The HTTP request methods to map to, narrowing the primary mapping: GET, POST, HEAD, OPTIONS, PUT, PATCH, DELETE, TRACE. | | String name | Assign name to mapping | | String[] params | The parameters of the mapped request. | | String[] path | The path mapping URIs. | | String[] produces | The producible media types of the mapped request. | | String[] value | The primary mapping expressed by this annotation. |   Ex: @RestController  @RequestMapping("/home")  public class IndexController {    @RequestMapping(value={"", "/page", "page\*","view/\*,\*\*/msg"})  String indexMultipleMapping(){  return "Hello from index multiple mapping.";  }} | Method, Class (Type) | Runtime |
| @CookieValue | It is only used with method annotated with @RequestMapping  The HTTP cookie value bound to the @CookieValue parameter for given cookie name.  Ex: @RequestMapping("/cookieValue")  public void getCookieValue(@CookieValue "JSESSIONID" String cookie){  } | Parameter | Runtime |
| @CrossOrigin | Used to enable cross origin requests. In many cases the host that serves JavaScript is different form host that serves data means cross origin resource sharing (CORS).  By default it allows all the origins, all headers, HTTP methods specified in @RequestMapping with max age of 30 min. you can customized this by specifying corresponding attribute values.  Ex:  @CrossOrigin(maxAge = 3600)  @RestController  @RequestMapping("/account")  public class AccountController {    @CrossOrigin(origins = "http://example.com")  @RequestMapping("/message")  public Message getMessage() {  // ...  }    @RequestMapping("/note")  public Note getNote() {  // ...  }  } | Method, Class (Type) | Runtime |
| @GetMapping | It is variant of @RequestMapping  Mapping of HTTP GET request onto specific method handler.  Alternative to @RequestMapping(method=RequestMethod.GET)  Ex: | Method | Runtime |
| @PostMappng | It is variant of @RequestMapping  Mapping of HTTP POST request onto specific method handler.  Alternative to @RequestMapping(method=RequestMethod.POST)  Ex: | Method | Runtime |
| @PutMapping | It is variant of @RequestMapping  Mapping of HTTP PUT request onto specific method handler.  Alternative to @RequestMapping(method=RequestMethod.PUT)  Ex: | Method | Runtime |
| @DeleteMapping | It is variant of @RequestMapping  Mapping of HTTP DELETE request onto specific method handler.  Alternative to @RequestMapping(method=RequestMethod.DELETE)  Ex: | Method | Runtime |
| @PatchMapping | It is variant of @RequestMapping  Mapping of HTTP PATCH request onto specific method handler.  Alternative to @RequestMapping(method=RequestMethod.PATCH)  Ex: | Method | Runtime |
| @InitBinder | It plays role to identify the methods which initialize WebDataBinder (a databinder that binds the request parameter to Java Bean Objects)  It supports many arguments as @RequestMapping method except commands and form arguments.  Ex:  @InitBinder  public void dataBinding(WebDataBinder binder) {  binder.addValidators(userValidator, emailValidator);  SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");  dateFormat.setLenient(false);  binder.registerCustomEditor(Date.class, "dob", new CustomDateEditor(dateFormat, true));  } | Method | Runtime |
| @PathVariable | It is used to annotate request handler arguments  Ex: @RequestMapping("/welcomeString/{countryName}/{userName}")  public ModelAndView helloString(@PathVariable("countryName") String cn, @PathVariable("userName") String un)  {  ModelAndView model = new ModelAndView("page");  model.addObject("msg", "Country : " + cn + " <=======> Name : " + un);  return model;  } | Parameter | Runtime |
| @RequestAttribute | It used to bind the request attributes to handler method parameter.  It is used to access the objects which have been populated on server side.  Ex :  @RequestMapping("/")  public String handle (@RequestAttribute("visitorCounter") Integer counter) {  return String.format("Visitor number: %d", counter);  } | Parameter | Runtime |
| @RequestParam | It is used to bind the parameter values from query string to handler method arguments.  Ex: | Parameter | Runtime |
| @ReqeustBody | It indicates that method parameter should be bound to the value of the HTTP request body.  HttpMessageConvertor is responsible for converting from HTTP request message to Object. | Parameter | Runtime |
| @RequestHeader | Used to map controller parameter to request header value.  It checks the header with name specified within the annotation and binds its values to handler method parameter. | Parameter | Runtime |
| @ResponseBody | Indicates that the result type should be written in JSON or XML format. | Method, Class (Type) | Runtime |
| @ResponseStatus | It marks method or exception class with HTTP response status and reason must be returned. | Method, Class (Type) | Runtime |
| @ControllerAdvice | * For each controller you can use @ExceptionHandler on method that will be called when given exception occurs but it handles only those exceptions that occur within that controller. * To overcome this problem @ControllerAdvice is used which enable us to define @ExceptionHandler, @InitBinder, @ModelAttribute that applied to all that apply to all @RequestMapping methods. * It will be applied to all controllers. | Class (Type) | Runtime |
| @RestController | * Marks class as controller where every method returns a domain object instead of view. * No need to use @ResponseBody annotation because it combines @Controller and @ ResponseBody. * you just send domain object as HTTP response in the format that consumer understands like JSON. | Class (Type) | Runtime |
| @RestControllerAdvice | * It combines @ControllerAdvice and @ResponseBody | Class (Type) | Runtime |
| @SessionAttribute | * Used to bind method parameter to session attribute. * provides convenient access to existing or permanent session attributes. | Parameter | Runtime |
| @SessionAtrributes | * Used when you want add JavaBean object into session. * It is used to when you want to keep JavaBean Object in session for short time. * It can be used with conjunction with @ModelAttribute   Ex:  @ModelAttribute("person")  public Person getPerson(){}  // within the same controller as above snippet  @Controller  @SeesionAttributes(value="person", types={Person.class})  public class PersonController{} | Class (Type) | Runtime |
| **Spring Cloud Annotations** | | | |
|  |  |  |  |