Spring Framework Annotations

Sl.No	Annotation	Description
01.	@Autowired	Annotation @Autowired is used to inject object dependency implicitly for a **constructor, field or method**. This is known as "autowired by type" since the object to be injected is discovered by its type. The items declared @Autowired need not have to be public.
02.	@Configurable	Used on classes to inject properties of domain objects. Types whose properties are injected without being instantiated by Spring can be declared with @Configurable annotation.
03.	@Qualifier	It can be used to create more than one bean of the same type and wire only one of the types with a property. It provides greater control on the dependency injection process and can be used with @Autowired annotation.
04.	@Required	Used to mark class members that are mandatory. The Spring auto-configuration fails if a particular property specified with this annotation cannot be injected.
05.	@ComponentScan	Make Spring scan the package for the @Configuration clases.
06.	@Configuration	It is used on classes that define beans.
07.	@Bean	It indicates that a method produces a bean which will be mananged by the Spring container.
08.	@Lazy	Makes a @Bean or @Component to be initialized only if it is requested.
09.	@Value	It is used to inject values into a bean's attribute from a property file. @Value annotation indicates a default value expression for the field or parameter.
10.	@Resource	Annotation used to inject an object that is already in the Application Context. It searches the instance by name. It also works on setter methods.
11.	@Primary	Annotation used when no name is provided telling Spring to inject an object of the annotated class first. Used along with @Component.
12.	@Component	Generic stereotype annotation used to tell Spring to create an instance of the object in the Application Context. It's possible to define any name for the instance, the default is the class name as camel case.
13.	@Controller	Stereotype annotation for presentation layer.
14.	@Repository	Stereotype annotation for persistence layer.
15.	@Service	Stereotype annotation for service layer.
16.	@Conditional	Annotation is used to load beans into Application Context only if the given condition is met.

Spring-Boot Web Annotations

Sl.No	Annotation	Description
01.	@SpringBootApplication	This annotation is used to qualify the main class for a Spring Boot project. The class used with this annotation must be present in the base path. @SpringBootApplication scans for sub-packages by doing a component scan.
02.	@EnableAutoConfiguration	Based on class path settings, property settings, new beans are added by Spring Boot by using this annotation.
03.	@Controller	Allows detection of component classes in the class path automatically and register bean definitions for the classes automatically.
04.	@RestController	Used in controllers that will behave as RESTful resources. @RestController is a convenience annotation that combines @Controller and @ResponseBody.
05.	@ResponseBody	Makes Spring to convert the returned object to a response body. This is useful for classes exposed as RESTful resources.
06.	@RequestMapping	Used to map web requests to specific handler classes and methods, based on the URI.
07.	@RequestParam	This annotation is used to bind request parameters to a method parameter in your controller.
08.	@PathVariable	This annotations binds the placeholder from the URI to the method parameter and can be used when the URI is dynamically created or the value of the URI itself acts as a parameter.

Profile

Sl.No	Annotation	Description
01.	spring.profiles.active	Property to be set in application.properties in order to tell Spring what profiles are active.
02.	@Profile("dev")	Annotation used to define which profile can execute the annotated method.

Aspect Annotations

Sl.No	Annotation	USE	Description
01.	@Aspect	Type	Declares a class to be an aspect.
02.	@After	Method	Declares a method to be called after a pointcut completes.

Sl.No	Annotation	USE	Description
03.	@AfterReturning	Method	Declares a method to be called after a pointcut returns successfully.
04.	@AfterThrowing	Method	Declares a method to be called after a pointcut throws an exception.
05.	@Around	Method	Declares a method that will wrap the pointcut.
06.	@Before	Method	Declares a method to be called before proceeding to the pointcut.
07.	@DeclareParents	Static Field	Declares that matching types should be given new parents,that is, it introduces new functionality into matching types.
08.	@Pointcut	Method	Declares an empty method as a pointcut placeholder method.

Spring-Boot Testing Annotations

Sl.No	Annotation	Description
01.	@AfterTransaction	Annotation used to identify which method needs to be invoked after a transaction is completed.
02.	@BeforeTransaction	Used to identify the method to be invoked before a transaction starts executing.
03.	@ContextConfiguration	Declares the annotated classes which will be used to load the context for the test. The location of the configuration file has to be povided to Spring.
04.	@DirtiesContext	This annotation indicates the test(s) modify or corrupt the SpringApplicationContext and that it should be closed. Hence, context is reloaded before the next test is executed.
05.	@ExpectedException	The test method is expected to throw a particular exception, else the test fails.
06.	@WebAppConfiguration	Used to create web version of the application context.
07.	@Repeat	Specifies the test method to be executed multiple times.
08.	@Transactional	Describes transaction attributes on a method or class.
09.	@Rollback	Indicates if the transaction of a test method must be rolled back after the execution of the test completed.
10.	@Commit	Indicates that the transaction of a test method must be committed after the execution of the test completed.
11.	@Timed	Indicates the time limit for the test method. If the test has not completed execution before the time expires, the test fails.
12.	@TestPropertySource	Annotation specifies the property sources for the test class.

Sl.No	Annotation	Description
13.	@Sql	Annotation declares a test class/method to run SQL scripts against a database.

Workflow and Tools

Sl.No	Category	Tool	Description
01.	Development	Spring Tool Suite (STS)	STS is an Eclipse-based development environment that can be used to develop Spring applications easily.
02.	Development	Eclipse	IDE used to develop Java applications. It has a plugin for develping Spring applications. Useful when working simultaneously on Spring and non-Spring based apps.
03.	Development	IntelliJ IDEA	Provides a smooth environment and user experience for developing Spring applications. Provides a comprehensive view of the project forspeedy navigation, error highlighting, plugins for numerous purposes, code completion.
04.	Unit Testing	JUnit	JUnit is a regression Testing Framework used to implement unit testing in Java with an enhanced speed and quality.
05.	Unit testing	Mockito	It is an open source mocking framework to create mock classes and interfaces providing realistic tests to predict the behaviour of the application.
06.	Unit testing	JaCoCo	Provides support for code-coverage measurement and generates a detailed test report.
07.	API Testing	JMeter	Apache's JMeter is an open source testing tool which includes load, functional, regression and performance tests.
08.	API Testing	Postman	Postman is a powerful tool used to test web services. It provides a feature to create test collections and can be used for API testing.
09.	API Testing	REST-Assured	Makes API tesing in Java simple by providing behaviour driven development. It can integrate with any existing Java automation framework.
10.	Monitoring	spring-boot-actuators	Enables monitoring and managing Spring boot application by shipping production ready featues like health, metrics, HTTP tracing etc.
11.	Monitoring	Micrometer	Micrometer is an application-neutral facade or abstraction for reporting application metrics which integrates with many monitoring systems, e.g., Graphite, New Relic and Statsd. Micrometer integrates with the metrics endpoint of spring-boot-actuators.

↥ back to top