In Java world, Spring Framework/Boot is a must-have skill set. XML-Based Configuration was for a long time main mean. But overloaded xmls with unreadable structure has become quickly headache for developers. Spring has started to switch from XML-based configuration to Annotation-based Configuration. Nowadays, greenfield projects are structured almost always in Annotation-based manner. Based on my experience and with some research I have tried to summarize the most important Spring annotations that a developer needs in a daily usage.

***Context…***

These annotations are there for injecting and creating beans for a specific spring context:

* *@Autowired* — used for injecting a bean in a runtime. It can be used with method, variable or constructor (preferred).
* *@Configuration* — class level annotation that declares one or more @Bean methods to be injected by Spring container.
* *@Scope* — used for declaring scope of the bean. For Instance: singleton or prototype etc.
* *@Qualifier* — used for the cases if there are more than one implementation of the interface, where spring does not know which bean to inject(throws *NoUniqueBeanDefinitionException*). Together with @Autowired this annotation eliminates ambiguous situations.
* *@Bean* — method level annotation that indicates Spring container should inject specific Bean that is returned from the method that is annotated with this annotation.
* *@Profile*— these annotation is used mainly for dev/test purposes to activate/deactivate specific bean injections based on chosen profile.
* *@ComoponentScan*— specifies directory in which @Component should be scanned for a specific configuration.

***Stereotyping…***

These annotations are used for associate the class to relevant application tier.

* *@Controller* — stereotypes bean as a MVC Controller.
* *@Service*— stereotypes bean as a Service. It is mainly used for indicating that class is holding business logic.
* *@Repository*— stereotypes bean as a Repository. It is used for showing that class has to do something with persistence, it throws database related exceptions For Example.
* *@Component* — it is Spring generic annotation indicating that class should be injected as a bean.

***MVC…***

These annotations are used in terms of MVC software architecture pattern:

* *@Controller* — used for showing that class is a MVC Controller.
* *@RequestMapping*— mapping specific URL to a class or handler method. This is an entry point from external callers to the application. Usually used together with @Controller.
* *@RequestParam* — used for binding handler method variable with request parameter.
* *@PathVariable*— used for binding handler method variable with request template parameter.
* *@RequestBody* — used for deserializing JSON body of HttpRequestto handler methods java object.
* *@ResponseBody*— used for serializing handler methods return object to HttpResponse JSON.

***Aspect Oriented…***

Spring wraps AspectJ with its own annotations.

* *@Aspect* — specifies class as an Aspect.
* *@Pointcut*— used to declare a pointcut.
* *@AfterRunning*— used to be run if pointcut runs successfully.
* *@AfterThowing*— used to be run if pointcut runs on failure.
* *@Around* — used for wrapping a pointcut
* *@After* — used to be run after pointcut completion
* *@Before* — used to be run before pointcut executes

***Cloud…***

Spring adds some cloud related annotations for cloud/microservices purposes.

* *@EnableEurekaServer*— used for implementing microservices with spring. This class level Annotation makes your service Eureka discovery service.
* *@EnableDiscoverClient*— used for registering as a discoverable service for others.
* *@EnableCircuitBreaker* — used for configuring Hystrix circuit breaker protocol.
* *@EnableConfigServer* — used for making annotated project a config server, that other services can get configurations from.

This is by far not the complete list of annotations that are available by Spring, but most frequently used ones. In the course of this week I will add examples.