<https://developer.okta.com/blog/2018/08/22/basic-crud-angular-7-and-spring-boot-2>

spring boot Annotations:

<https://howtodoinjava.com/spring-boot2/spring-boot-annotations/>

Spring boot is all about auto configuration

1)SpringBootApplication:

It marks the class as spring boot main class. Application will start executing from this class.

It enables below annotations

1)SpringBootConfiguration: It indicates that a class provides Spring Boot application configuration. It can be used as an alternative to the Spring’s standard @Configuration annotation so that configuration can be found automatically.

2)Component scan:

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.

3)Enable auto configuration: It will guess and configure bean that we will need . Auto-configuration classes are usually applied based on your classpath and what beans you have defined. For example, if you have tomcat-embedded.jar on your classpath you are likely to want a [TomcatServletWebServerFactory](https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/web/embedded/tomcat/TomcatServletWebServerFactory.html" \o "class in org.springframework.boot.web.embedded.tomcat) (unless you have defined your own [ServletWebServerFactory](https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/web/servlet/server/ServletWebServerFactory.html" \o "interface in org.springframework.boot.web.servlet.server) bean).

**@AutoConfigureBefore, @AutoConfigureAfter, @AutoConfigureOrder**

We can use the @AutoConfigureAfter or @AutoConfigureBefore annotations if our configuration needs to be applied in a specific order (before of after).

If we want to order certain auto-configurations that should not have any direct knowledge of each other, we can also use @AutoConfigureOrder. That annotation has the same semantic as the regular @Order annotation but provides a dedicated order for auto-configuration classes.

|  |
| --- |
| @AutoConfigureAfter Example |
| @Configuration  @AutoConfigureAfter(CacheAutoConfiguration.class)  @ConditionalOnBean(CacheManager.class)  @ConditionalOnClass(CacheStatisticsProvider.class)  public class RedissonCacheStatisticsAutoConfiguration  {      @Bean      public RedissonCacheStatisticsProvider redissonCacheStatisticsProvider(){          return new RedissonCacheStatisticsProvider();      }  } |