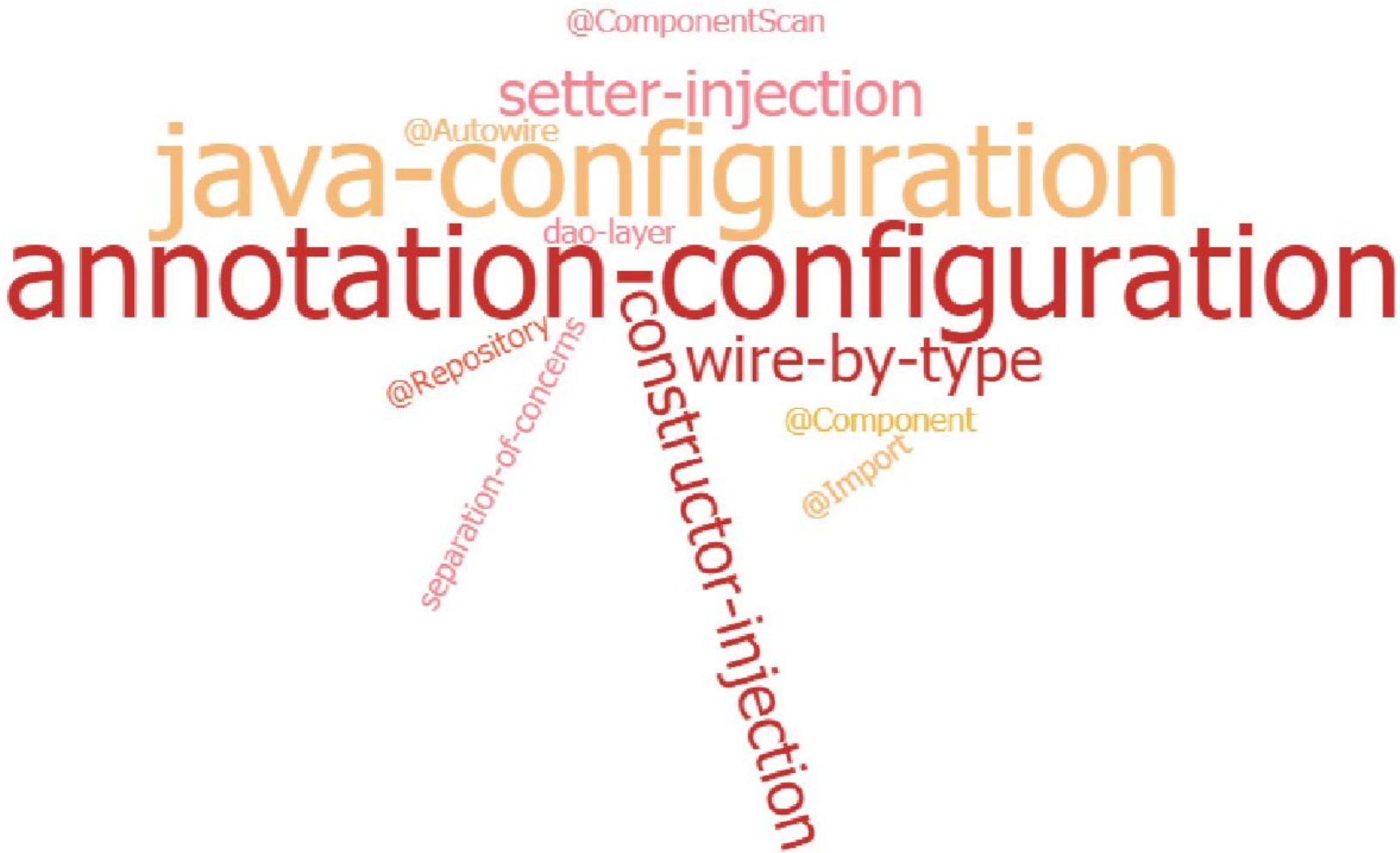

@ComponentScan
setter-injection
@Autowire
java-configuration
dao-layer
annotation-configuration
@Repository
separation-of-concerns
constructor-injection
wire-by-type
@Component
@Import

Java Configuration

- @Configuration - Classes annotated with `@Configuration` contain bean definitions.
- @Bean - The `@Bean` annotation is used to tell Spring that the result of the annotated method will be a bean that has to be managed by it. The `@Bean` annotation together with the method are treated as a bean definition, and the method name becomes the bean id.
- @Import - having multiple configuration classes is recommended for the same reason it is recommended to have multiple XML configuration files, to group together beans with the same responsibility and to make the application testable.
- @ImportResource - is possible also, to import a configuration from XML files and bootstrap everything with a `AnnotationConfigApplicationContext` instance, using the `@ImportResource` annotation on the configuration class.

Java Configuration

Advantages:

- Compile time checking
- No need for XML configuration

Disadvantage:

- Java class serving a configuration purpose only
- Impractical for large applications

Annotation Configuration

- @Qualifier - from Spring to specify name of the bean to inject
- @Component - template for any Spring-managed component(bean).
- @ComponentScan - Scans packages for bean definitions
- @Autowire - core annotation for this group; is used on dependencies to instruct Spring IoC to take care of injecting them. Can be used on fields, constructors, and setters.

Injection Types

- Constructor Injection - The `@Autowire` annotation can be used on constructors to tell Spring to use autowiring in order to provide arguments for that constructor.
- Setter Injection - In conclusion, when creating a bean using setter injection, the bean is first instantiated by calling the constructor and then initialized by injecting the dependencies using setters.
- Field Injection - when using @Autowire on the field

Autowiring

- By Type - Out of the box, Spring will try to **autowire by type**, because rarely in an application is there need for more than one bean of a type. Spring will inspect the type of dependency necessary and will inject the bean with that exact type.
- By Name - **if Spring cannot decide which bean to autowire based on type (because there are more beans of the same type in the application), it defaults to autowiring by name**.

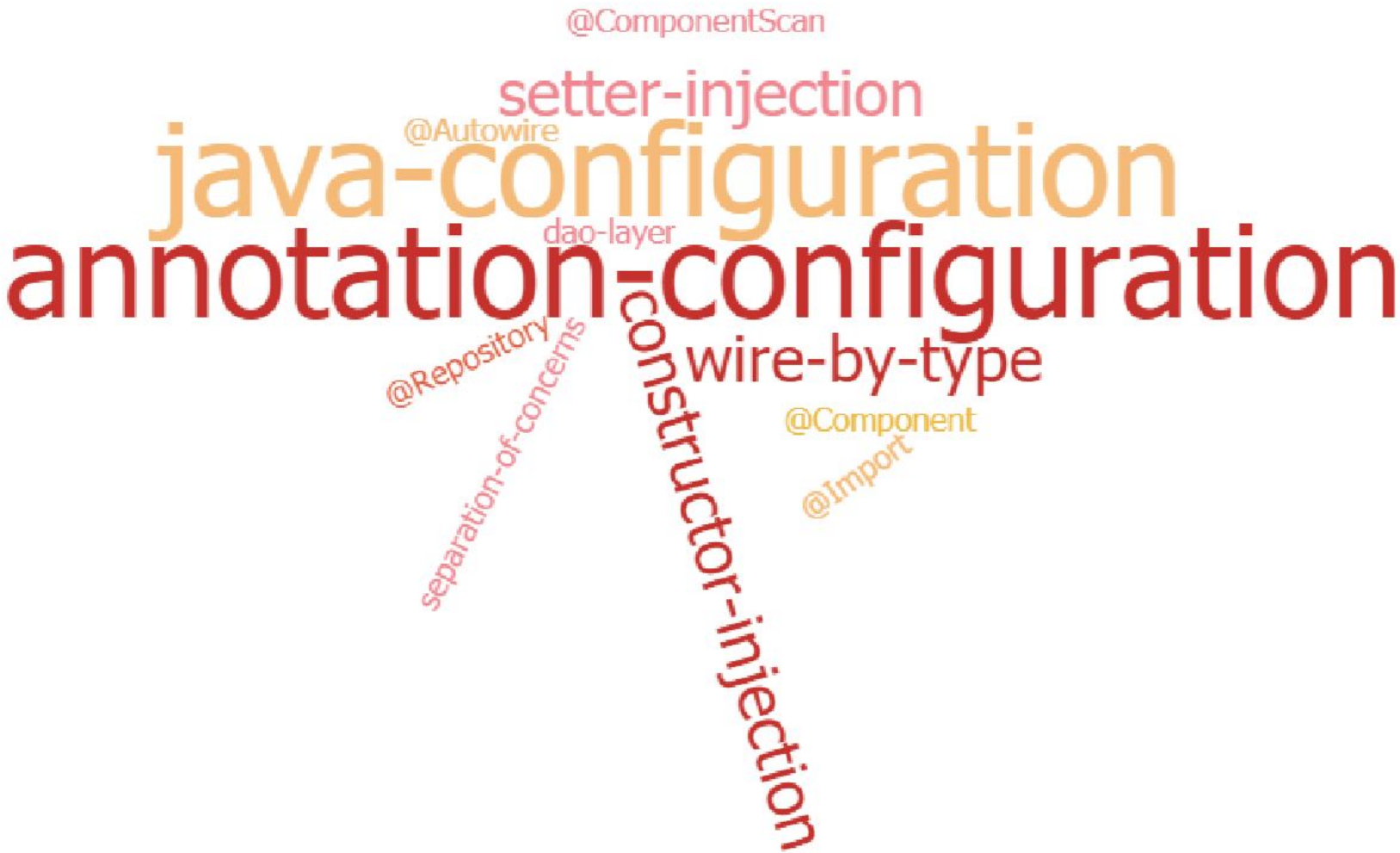
Annotation Configuration

Advantages:

- Effective when dealing with a lot of beans

Disadvantages:

- Need to own the source code
- Couples the code to Spring DI


@ComponentScan
setter-injection
@Autowire
java-configuration
dao-layer
annotation-configuration
@Repository
separation-of-concerns
constructor-injection
wire-by-type
@Component
@Import