SPRING FRAMEWORK

The theoretical basis for Spring Framework is the dependency inversion principle (it is one of the SOLID principles). It states that high level modules should not depend on low level ones, both of them should depend on abstraction. Dependency injection is a mechanism of injecting dependencies (so low level-modules) to high-level modules. Dependency injection uses inversion of control container (IoC) that holds the configuration settings and so on.

Nowadays we use annotation based configuration instead of the XML based settings + we boot the application with Spring Boot !!!

Most important annotations

@Component this is how we tell Spring to inject the given class with this annotation in the IoC container, so it is used to auto-detect and auto-configure beans using classpath scanning. We can use dependency injection whenever we want to use that class

@Service it is a @Component, so the functionality is the same as we have discussed for @Component, but we annotate classes that are services in the application

@Repository it is a @Component, but we annotate classes that are repositories, so we have the database-related operations in these classes

@Bean it is used to explicitly declare a single bean, rather than letting the framework do it automatically with scanning

@Autowire this is how we inject a dependency, we do not have to instantiate the class with the 'new' keyword, it is handler by the framework itself

@Qualitfier there may be a situation when you create more than one bean of the same type and want to wire only one of them with a property, in such case you can use @Qualifier annotation along with @Autowired to remove the confusion by specifying which exact bean will be wired

@Configuration it is a @Component as well. Indicates that the given class declares one or more @Bean methods. It indicates also that the class may be processed by the Spring container to generate bean definitions and service requests for those beans at runtime

@ComponentScan Spring container can detect and register the beans and the classes with the @Component annotation (as well as the @Services and @Repositorys). We can define the package name in which Spring is going to scan for the components.

@EnableAutoConfiguration auto-configures the beans that are present in the classpath. For example, if we have tomcat-embedded.jar in the classpath, we need the TomcatEmbeddedServletContainerFactory bean to configure the tomcat server. This will be done because of this annotation

@SpringBootApplication because most of the applications needs the @Configuration, @ComponentScan and the @EnableAutoConfiguration annotations, they have been merged into this single @EnableAutoConfiguration annotation