**Spring**

1. Spring works on 3 principles
   1. DI / IOC
   2. AOP
   3. Abstraction
2. Every class in spring Is called as a bean
3. Steps to create a spring Maven Project
   1. File -> new -> Other -> maven -> Maven Project
   2. Archetype : quickstart
   3. Group Id : package name
   4. Artifact Id : project name
   5. Update pom.xml file
      1. Java version : its different till jdk 8 and beyond jdk 9   
         Within <properties> tag
      2. Add dependencies
         1. Spring-context
         2. To add version at 1 place and use it using ${spring.version}  
            <properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<java.version>11</java.version>

**<spring.version>5.3.23</spring.version>**

**</properties>**

1. Create a class => **follow the naming conventions : getters/ setters**
2. Create a xml file with the name  
   spring-core.xml [ filename can be anything ]
   * 1. Configure our beans using <bean>
     2. Id attribute => instance of the class
3. Load xml file
   * 1. In main method  
        ApplicationContext context = new

ClassPathXmlApplicationContext("spring-core.xml");

* + 1. The moment the xml file is loaded all the beans configured using <bean> tag are loaded and instantiated.

1. Dependency injection
   1. Constructor
      1. XML : <constructor-arg>
      2. Annotation : **@Autowired or @Value**
   2. Setter
      1. XML : <property>
      2. Annotation : @Autowired or @Value
   3. Field injection : only via annotations : @Autowired or @Value
2. Namespaces
   1. Context namespace:  
      <context:annotation-config/> : scanning dependency injection  
      <context:component-scan basePackages =”<path to package>/> : scanning @Component(“”) and all DI annotations   
        
      By default the id generated by spring is the camel case of the classname
   2. P namespace : property
   3. C namespace : constructor
3. Scoping : specifies the no of instances per application  
   scope=””  
   @Scope
   1. Singleton : eager loading
   2. Prototype : lazy loading
   3. Request
   4. Session
4. Lazy initialization :
   1. XML : lazy-init =”true”
   2. Annotation : @lazy
5. Java based configuration : replace xml file with a class
   1. @Configuration : the class annotated with this annotation is the place where application specific configuration lives
   2. @ComponentScan : for the beans annotated with @Component and @Bean annotation
6. @Bean : used on the method level for any DI