**Spring Framework**

1. Framework is predefined functionalities/code provided by any third party which can be use to reduce development efforts, time and minimize the errors.
2. Spring Framework is also known as framework of framework.
3. Spring distributed into a multiple module which can be use either separately or can be use in the combination of multiple modules.
4. Some of the important modules are listed below

**Spring IOC/Core**

Spring MVC

**Spring JDBC**

**Spring ORM (JPA/Hibernate)**

**Spring REST**

Spring Security

Spring JMS

Spring Batch

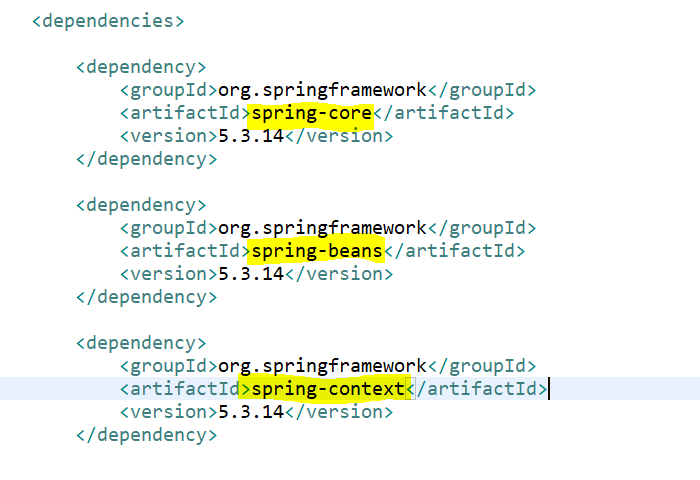
Spring Cloud etc…

1. To Implement any of the spring module you have to follow 3 steps at higher level.
   1. Add a jar files (Dependencies)
   2. Add Configuration of the module (by Spring suggested way)
   3. Customize the spring functionalities (Optional)
   4. Use/Implement the functionalities provided by Spring

Spring Docs : <https://docs.spring.io/spring-framework/docs/current/reference/html/>

**Spring IOC (Inverse Of Control)**

1. Spring IOC is also called as spring core.
2. This is the base module of the spring.
3. It can be implemented for core java application, Web App, Restful API Application etc…
4. In this module you will majorly implement following functionalities
   1. Spring Configuration.
   2. Spring Bean Class.
   3. Spring Container.
   4. Create/Get the spring bean class object using Spring Container.
   5. Spring Bean scope
   6. \*\*\*Spring CI(Constructor Injection), SI(Setter Injection), DI(Dependency Injection)
   7. \*\*\*Spring Autowire
5. Dependencies (jar) for Spring IOC
   1. Spring-core
   2. Spring bean
   3. Spring context



**Spring Configuration**

1. Spring configuration can be done using 2 ways
   1. **Using XML file and XML Tags**
      1. Create resources folder
      2. Create xml file
      3. Add following Xsd inside xml file

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"*

*http://www.springframework.org/schema/beanshttp://www.springframework.org/schema/beans/spring-beans.xsd"*>

<!-- bean definitions here -->

</beans>

* 1. **Using Configuration class and Annotations**.
     1. Configuration Class is a java class with **@Configuration** annotation
     2. Use **@ComponentScan** annotation to provide the location of you component classes, so that spring will search all the component/bean classes in the given package.

**Spring Bean class**

1. These are the java classes whose object creation and object maintained by Spring container.
2. To convert any java class into spring bean class you have to configure it inside XML file or you can use Annotations provided by spring.
3. XML tags to set Java class as spring bean class.

<bean id=*"ObjName"* class=*"package.ClassName"*>

</bean>

**Spring Container**

1. Spring containers are the classes which create, holds and manage the spring bean classes.
2. There are 2 types of spring container available
   1. ApplicationContext
      1. Application context container is the advance container and support more functionalities than BeanFactory.
      2. This can be use for large applications like Web Application
      3. To get an application Context container there are multiple options provided by spring.
         1. Using **ClassPathXmlApplicationContext** when configuration is using XML file.
         2. Using **AnnotationConfigApplicationContext** when configuration is using Annotation.
   2. BeanFactory
      1. This container support less functionalities than the Application Context.
      2. This container can be use for small applications like Core Java application.
3. The objects created by spring can be get using the method provided by container.

container.getBean("ID", ClassName.**class**);

**Spring Bean Scope**

1. There are total 5 spring bean scopes
   1. **Singleton**: Is the default scope if you not explicitly provided. In this scope only a single object of the class will be created for the given id inside a spring container.
   2. **Prototype**: In this scope multiple objects will be created for a single id whenever use request to a container.
   3. **Request**: This scope is applicable for web application, in this scope new object will be created for every request.
   4. **Session**: This scope is applicable for web application, In this scope only one object gets created for a session.
   5. **GlobleSession**: This scope is applicable for web application, in this scope only one object present in a application which is share by all users.

**Setter Injection (SI)**

1. This is the process where you can pass the values for the variables using a setter method.
2. This can be achieved by using following tag in the XML file

<property name=*"variableName"* value=*"value"*></property>

**Constructor Injection (CI)**

1. This is the process where you can pass the values for the variables using a constructor.
2. This can be achieved by using following tag in the XML file

<constructor-arg value=*"value"*></constructor-arg>

**Dependency Injection (DI)**

1. This is a process where object of one class injected inside a Object of another class (HAS-A relation in core java).
2. DI can be achieved using SI or CI.
3. DI using SI

<property name=*"addr"* **ref=*"ad"***></property> <!-- DI by SI -->

1. DI using CI

<constructor-arg **ref=*"ad"***></constructor-arg> <!-- DI by CI -->

**Autowire**

1. Autowire is a process where you can automate the dependency injection(DI) process.
2. There are different type of autowire
   1. ByName: is use for a DI using SI. DI happens based on the name of your variable.
   2. ByType: is use for a DI using SI. DI happens based on the type of your variable.
   3. Constructor: is use for a DI using CI
   4. No: no DI injection happened automatically in this option.

**Annotations**

1. @Configuration
   1. This annotation is use to define any class as a configuration for the Spring.
2. @ComponentScan
   1. Using this annotation you can instruct spring to scan the given package for the bean class.
   2. Here you have to provide the package name which needs to scan for the bean classes.
3. @Component
   1. Using this annotation, you can define any java class as a spring bean class.
   2. The object of these java classes will create by spring after scanning the package
   3. There are child annotations which can be used for the same purpose,

@Controller, @RestController, @Service, @Repository, @ControllerAdivce etc.