Spring Framework & Microservices

In Java

* Interfaces
* Collection Framework

Framework: It is like a semi-implemented application which takes care of lot of common features every application needs so that developers need not write code on these common features, which are:

1. Design Pattern
2. Type Conversion
3. Exception Handling

Spring Framework

It is a Java framework that helps you to create various types of applications like web, webservices, cloud based applications

It gives you many modules

1. Spring Core / Spring Context: Takes care of design patterns & Dependency Injection (IoC)
2. Spring Web MVC: Create web & REST webservice application
3. Spring Boot: Quickly automate configurations
4. Spring Cloud: Create spring applications that can be run on cloud environments like microservices
5. Spring AOP: Aspect Oriented Programming
6. Spring JPA: To interact with the Database

Spring Core

It gives you all the design patterns & Dependency Injection feature

In Maven repository

spring-context

Types of DI

1. Setter Injection: It uses setter methods to initialize the object
2. Constructor Injection: It uses constructor arguments to initialize the object

You can configure application related data in XML file spring container can pass these data to the object via constructor or setters these are called as setter injection not only these data spring can also pass objects via constructor/setters

Setter Injection: Use <property>

Constructor Injection: Use <constructor-arg>

Try to comment all the <property> tags of DBSource bean & use constructor to initialize username, url, password using <constructor-arg>

DBSource(String username, String password, String url) { … }

Supplying an object into another object

class A {   
  
}  
class B {   
 A a1;  
}

<bean id = “b3” class = “com.A”>  
</bean>

<bean id = “b4” class = “com.B”>  
 <property name = “a1” ref = “b3” />  
</bean>

Annotation configuration

Spring can configure the beans using different types of annotations like

@Component, @Service, @Repository, @Controller, @RestController

All the above annotations are written on top of the class, any class having above annotations are instantiated spring container

@Service   
class A { }

@Repository  
class B { }

@Autowired: This is used to inject the object into another object like an alternative way of using <property ref >

@Qualifier: This is used when you want to specify which object you want spring to inject to the object

Question: If we have annotation can we completely ignore XML configuration

Answer: Still you can’t ignore because some third party classes if spring needs to instantiate then the only way is to use <bean> tag in the XML

Note: If spring needs to configure the classes with annotations then we must write <component-scan> tag in the XML file

Activity:

Create two <bean> for DBSource class and check which object the spring container is supplying to the TestDao & if not, make the spring container to supply any one of the DBSource object to the TestDao

Bean life cycle methods

These methods are automatically called when objects are created/destroyed/properties are set

There are annotations for these life-cycle methods

@PostConstruct: A method having this annotation is automatically called after bean is created by spring container

@PreDestroy: A method having this annotation is automatically called before bean is removed from the spring container