Spring

① what is Spring ?

* Spring is a light weight Container which runs on JRE.

* Spring is most popular application development framework for enterprise Java.

* Spring helps to develops enterprise-class application using POJOs.

* The main use of pojo is We used to have heavy Containers the EJB etc. but We Can make use to run on light-weight Container.

* Spring uses ligut weight Container, the IOC for developing applications.

POJO (plain old Java object) - A class which does not inherit any class are called it as pojo class.

→ pojos basically define any entity. Like in your program, If you Want any Employee class, than you Can create a pojo as.

```
* public class Employee {
private String name;
private String id;
private double Salary;

}
```

→ The Major advantage of the pojo class is that we will not have to Create objects every time in other Java programs. Simple we Can access the objects by using the get() & set() methods.

→ A pojo class should not extend predefined classes.

→ It should not Implement prespecified interfaces.

→ It should not have any prespecified annotation.

→ In spring we are using
1) RTP
2) Has-A-relationship
we Can make our layers light-weight.

② what is IOC Containor ?

There are two Important modulus of Spring framework.

1) IOC (Inversion of Controller)
2) MVC (Cmodel view Controller)

It is the Concept where programmer do not create a objects explicitly instead of programmer describes how the object should be Created in a Configuration file The Configuration file will Contain the information of the class & also the data that should be present in the object. The Configuration file usually will be xml file. xml file are used to store & transport the data.

* Ioc is responsible to instantiate, Configure and assemble the objects.

* The Ioc Containor gets information from the xml file and works accordingly.

* The Important functionalities of IOC Containor
  1) to Instantiate the application class
  2) to Configure the objects.

Internally IoC Container will go to the Specified xml File and Searches for the Specified class id and instantiate class and loads the class, implements the methods or sets the values to the attributes and destroys the object.

$$\begin{tikzcd}
 & \text{IoC} \arrow[dl, dash] \arrow[dr, dash] & \\
\text{Core} \arrow[d] & & \text{J2EE} \arrow[d] \\
\text{BeanFactory (I)} \arrow[d] & & \text{ApplicationContext (I)} \arrow[d] \\
\text{XmlBeanFactory} & & \text{ClassPathXmlApplicationContext}
\end{tikzcd}$$

$$\begin{tikzcd}
\text{MVC} \arrow[d] \\
\text{WebApplicationContext (I)} \arrow[d] \\
\text{WebApplicationContextUtil}
\end{tikzcd}$$

③ what is dependency Injection ?

→ Dependency Injection using to Connect two class together and at the Same time keeping them independent.

→ These dependency between part express the meaning of association between two class(s).

→ for example class A is dependent of class B All this means, class B will get injected into class A by the IoC.

→ Dependency Injection makes our application loosely coupled.

→ it will pass the value from xml file to pojo class.

→ Dependency injection Can be acheived in two ways described bellow.

1) by using Constructor injection
2) by using Setter Injection (post Constructor)

→ Initialisation the value for data member of the bean class by using the Constructor is known as Constructor Injection.

→ The Constructor arg tag is used a a Configuration file.

Ex:- <Constructor_arg value = "234"</Construtor-arg>

→ The Container will set the values for the data member by referring the xml file where the data member name and values specify the property tag & for the injection setter also must be there at the bean class.

Ex:- <property name = "UserId" value = "1234"></property>

Injecting object

We can inject Objects which is required by another object with the heep spring Containers.

1) Create the depended bean class
2) Configure the bean in xml file using bean tag.
3) ~~Configure the bean in~~ Specify property tag where the bean names to be injected with the following Syntax.

```
<bean id = "pen" class = "com.ty.pen" >
<property name = "UserId" ref = "marker" ></property>
</bean>
```

④ @Autowired —

It is used to autowire Spring bean on setter methods, instance variable, and constructor. When we use @Autowired annotation, the spring container auto-wires the bean by matching data-type. [It will check whether the object of the bean is present or ~~the~~ not]

⑤ @Bean :-

It is a method-level annotation. It is an alternative of XML <bean> tag. It tells the method to produce a bean to be managed by spring container. whenever we are using third party object to instantiate [which gives object of the pojo class & configured it as a bean]

⑥ @Component

It is a class-level annotation. It is used to mark a Java class as a bean. A Java class annotated with @Component. The Spring framework pick it up and configure it in the application context as a Spring Bean.

It is used to create a object for Bean class.

(7) @Configuration:-

It indicates that ~~the~~ class can be used by the spring IOC Container as a source of bean definitions. The @Bean annotation tells spring that a method annotated with @Bean will return an object that should be registered as a bean in the spring application Context. [when we can avoid configuring bean in the xml file, & write that bean in the configuration class by using @Bean in the method.

(8) @ComponentScan

@ComponentScan which is used along with the @Configuration annotation to specify the packages that we want to be scanned.

It tells spring in which packages you have annotated classes which should be managed by spring. Spring needs to know which packages contain spring beans, otherwise you would have to register each bean individually in (xml file). This is the use of @ComponentScan.

9 @Qualifier :-

We Can eliminate the issue of which bean needs to be injected.

```
public class FoodService{
    @Autowired
    @Qualifier("foodFormatter");
    private Formatter formatter;
```

→ @Qualifier annotation is used to resolve the autowiring Conflict, when there are multiple beans of same type. The @Qualifier annotation Can be used on any class annotated with @Component or methods annotated with @Bean. This annotation Can also applied on Constructor arguments or method parameters.

(10) @Primary :-

It is used when a particular implementation requires a higher preference over other beans. If exactly one 'primary' bean exists among the candidates, it will be the Default autowired value.

(11) @ value

which is used to assign default values to variables and method arguments.

(12) what is BeanFactory ?

→ BeanFactory is an Interface present in "org.springframework.beans.factory.BeanFactory."

→ It is having an Implementation class XmlBeanFactory.

→ It is a Simple Container which provides basic support for dependency injection.

→ To Create XmlBeanFactory object we require Classpath resource class instance.

→ Where in classpathResource we have to provide the xml file name along with extension.

→ Then we have to pass the object reference to ~~the~~ XmlBeanFactory (" classpath reference ref") to instantiate BeanFactory.

(13) what is ApplicationContent ?

→ ApplicationContent is a Interface which implements BeanFactory and present in org.springframework.context.ApplicationContent.

→ Generally ApplicationContent is more use Compare to BeanFactory.

$$\begin{tikzcd} \text{Beanfactor (I)} \\ \text{ApplicationContent (I)} \arrow[u] \end{tikzcd}$$

→ It Contain Somany Implementation class For the ApplicationContent like →

1) classPathxmlApplicationContent

2) AnnotationConfigApplicationContent.

→ based on requirement we choose Implementation Contegt.

BeanFactory	ApplicationContent
→ BeanFactory is an Interface	→ It is a Sub Interface of BeanFactory.
→ Support lazy loading.	→ Support fast loading
→ Does not support annotation based dependency injection.	→ Support annotation based dependency injection.
→	

(14) What is the different bean Scope in Spring.

Singleton → The bean instance will ~~Create~~ be only once and same instance will be returned by the IOC Container, it is the default scope. BF ⇒ Lazy creation
AC → Eager Creation

prototype → The bean instance will Create everytime we a request for that specific bean
BF → Lazy creation
AC → Lazy Creation

→ If objecttype is prototype → destroy method won't work.

(15) What is the use of destroy-method attribute of bean tag in spring.

→ The destroy-method is called before the bean is removed from the Container.

→ we can define destroy-method attribute inside a bean tag of xml file and initialize with a method name.

→ The method which initialised will get executed before bean instance is removed from Container.

Ex →

```
Class A {
    public void m1() {
    }
}
```

in xml

```
<bean id="a" class = "com.ty.A" destroy-method="m1">
</bean>.
```

(16) what is the use of init-method attribute of bean tag in spring.

→ The init-method attribute used for specify a method that is to be called immediately upon instantiate in bean

→ we can declare a init-attribute inside bean tag of the xml file of spring.

→ We have to pass method name as value to the init-method.

Ex:—

```
class A {
    public void m1()
    }
}
```

in xml

```
<bean id = "a" , class= "com.ty" init-method="m1">
</bean>
```