**Spring Framework**

**Introduction 🡪**

1. Spring is a framework
2. Spring framework is also framework of framework.
3. Spring framework is also called as dependency Injection framework.
4. Using sprig framework, we will be able develop loosely coupled applications.
5. Imp. Point 🡪 In spring framework we have IOC, which is responsible for doing dependency Injection.
6. Spring framework will have many more modules/ components like Spring JDBC, Spring MVC, Spring security, Spring ORM
7. Spring framework in Rod Johnson in 2003.
8. Before Spring, this framework was called as Interface21.
9. EJB, it was heavy weighted application, in order to overcome it Spring comes in picture.
10. Spring is named on season, as EJB was considered as winter so in order to overcome from winter Spring season plays role therefore Interface21 later named as Spring framework.

**Dependency Injection**.

It is nothing but it is a design pattern. Design pattern is somewhat helps us to design our application on proper format.

Class Teacher{

}

Class Student{

Teacher obj = new Teacher();

}

//traditional approach that we can create object by ourselves.

But using Spring, IOC will take the responsibility of creating objects for us.

Imp Point 🡪 Here in Spring framework, spring is taking the responsibility of creating object for our application and this mechanism is called IOC.

**What is IOC?**

Inversion of Control,

Here Spring is going to take control of creating objects and injecting as per requirement therefore this mechanism we can say as IOC.

**Where exactly it will be used?**

Specially it is used to developed advance Java or EE applications.

main

**UI layer**

Customer

**Business Logic Layer**

CustomerDao

**Data Access Layer**

Spring Modules:

Web module of Spring

**Data Access Layer**

Servlets

Web

Spring ORM

Spring JDBC

Web Socket

Portlet

OXM

JMS

Messaging

Instrumentation

ASPECT

AOP

Spring Core- Core Bean Context SpEl

Test

**Core**- Core is responsible for all fundamentals of Spring, core functionality like Dependency Injection, IOC is responsibility of Core

**Context** -Internationalization, propagation of functions, resource loading and any more, EJB will be provided by context, JMS, remoting. The Context module builds on the solid base provided by the Core and Beans modules and it is a medium to access any objects defined and configured. The ApplicationContext interface is the focal point of the Context module.

**Bean**- POJO classes, for which IOC will generate objects. The Bean module provides BeanFactory, which is a sophisticated implementation of the factory pattern.

**SpEl** – It is very useful for graphing the objects. Simple words we can say that it is use to manipulate values in object.

**AOP-** Aspect Oriented Programming, It helps us to implement method interceptors. We can also define point cut. It also helps is to decouple code.

**Interceptors** – it is nothing but it is set of instructions should be fired before or after any method.

**Instrumentation 🡪** If there will be requirement of some classes for our application then responsibility of loading those classes is of instrumentation.

**Messaging 🡪** If there will be requirement of messaging in our application, then messaging module of Spring will be helpful, Messaging also includes some annotations.

Data Access Layer 🡪

**Spring JDBC**- Spring JDBC is going to provide abstract layer for JDBC. In traditional JDBC we had to write long and tedious code for getting our application connect with Database but here using Spring JDBC it’s become very easy.

**ORM** - If our requirement is to implement ORM in our application then instead of using Hibernate we can also use Spring ORM.

**OXM**- Object XML mapping

**JMS**- for producing messages and consume messages.

**Web Module 🡪**

When we will have to implement web-based application then Spring web modules plays very important role. For implementing REST API, we can also use Spring web module.

Web

The Web layer consists of the Web, Web-MVC, Web-Socket, and Web-Portlet modules the details of which are as follows −

**The Web module** provides basic web-oriented integration features such as multipart file-upload functionality and the initialization of the IoC container using servlet listeners and a web-oriented application context.

**The Web-MVC** module contains Spring's Model-View-Controller (MVC) implementation for web applications.

**The Web-Socket** module provides support for WebSocket-based, two-way communication between the client and the server in web applications.

**The Web-Portlet** module provides the MVC implementation to be used in a portlet environment and mirrors the functionality of Web-Servlet module.

**Test** 🡪 This module will help us to let integrate the testing framework in our project to perform testing for our application. The Test module supports the testing of Spring components with JUnit or TestNG frameworks.

**Spring IOC Container** 🡪

Spring IOC is component which comes along with our Spring. IOC container is responsible for generating object and inject them as per requirement. In other words, I can say IOC is responsible for lifecycle of object.

Spring IOC is mainly doing three tasks.

1. **Create objects**
2. **Hold them in memory**
3. **Inject them as per requirement.**

For achieving this we have to follow some steps

1. We will have to specify Bean classes, Bean classes nothing but consider them as POJO classes.
2. Specify configuration file.

Java Application

Bean

Bean

Config.xml

For IOC container we got something called as

**ApplicationContext** -- > It is an interface. In spring container we also have BeanFactory interface but ApplicationContext is going to extends BeanFactory interface because of which all functionality of BeanFactory interface is present in ApplicationContext.

As it is an interface we can’t create the object of ApplicationContext. So for using this interface we have subclasses for ApplicationContext.

Subclasses for ApplicationContext are

1. **ClassPathXMLApplicationContext**
2. **AnnotationConfigApplicationContext**
3. **FileSystemXMLApplicationContext.**

**ClassPathXMLApplicationContext**-> It searches for XML configuration from java class path.

**AnnotationConfigApplicationContext** 🡪 If we have implemented annotations in Bean itself then this subclass is going to search for those Beans in our application.

**FileSystemXMLApplicationContext🡪**Whenever we will have to search for XML base configuration in file system then we will have to use this subclass.

**Dependency Injection🡪**

Types of Injection

1. Setter Injection (Properties injection)
2. Constructor Injection.

**Configuration File 🡪**

This config file is similar to config file in Hibernate.

So whatever will be Bean present in our application for which we will require objects need to specify in config file.

<beans>

<bean>

</bean>

</beans>

**Important Point, while doing the injection IOC will also checks the data types of Objects.**

Which datatypes IOC consider while injecting.

1. Primitive data type
   1. byte, int, float, short, double, long, Boolean, char
2. Collection Type – List, Set, Map
3. Reference Type or Non-primitive type. Classes

**Injecting Collection-**

<bean class=” ” name=” ”>

<property name=” Phones”>

<list>

<value>101</value>

<value>102</value>

</list>

</property>

<property name=” Addresss”>

<set>

<value>Mumbai</value>

<value>Chennai</value>

<value>Hyderabad</value>

</set>

</property>

<property name=”courses”>

<map>

<entry key=”Java” value=”2 months”>

<entry key=”C#” value=”1 months”>

</map>

</property>

</bean>

Class Employee

Name

List Phones

Set Addresses

Map courses