**Spring Framework:**

**Why do we use spring?**

* It is light weight & open source.
* It is complete end (applicable for all layers in application) & modular framework (applicable for particular layer in application).
* We can achieve loose coupling.
* Non-invasive framework (doesn’t force to extend or implement any base class or interface).
* We can develop easy to test kind of applications.
* Which server Spring has by-default?
* Spring doesn’t have server, we need to add it externally.

**List out the modules of spring?**

There are total 6 modules in spring-

* Spring Core
* Spring Context/J2EE
* Spring DAO/JDBC
* Spring ORM
* Spring AOP
* Spring Web MVC

**What is IOC in spring?**

* IOC stands for Inversion of Control.
* It is heart of spring framework.
* It converts object from tight coupling to loose coupling which is achieved by dependency injection.
* (Tight Coupling -> Change in one class forces to change in other (dependent) class Loose Coupling -> Change in one class doesn’t forces to change in other (dependent) class)
* What are the types of Dependency?
* Primitive, Secondary and Collection.
* What is Dependency and what is Dependency Injection?
* Data member or properties are dependency and assigning values to those data member is called as dependency injection.

**What are the types of dependency injection (DI)?**

* Setter based DI & Constructor based DI

**Difference between Setter based and Constructor based?**

|  |  |
| --- | --- |
| Setter Based DI | Constructor Based DI |
| It allows partial injection. | It does not allow partial injection. |
| It overrides constructor based DI. | It doesn’t override setter based DI. |
| It is mutable. | It is immutable. |
| It increases line of code. | It decreases line of code. |

**Inject Secondary reference injection bean file?**

<bean id="e" class="com.demo.beans.Engine">

<property name="engineNumber" value="A123GAAG" />

<property name="modelYear" value="2018" />

</bean>

<bean id="c" class="com.demo.beans.Car">

<!-- passing reference of Engine class -->

<property name="engine" ref="e" />

<property name="carName" value="Honda" />

</bean>

**Inject Primary and Secondary array injection bean file?**

public class Engine {

private String modelYear; // primitive string

// generate getters and setters.

}

public class Car {

private String[] carNames; // primitive string array private Engine[] engines; // secondary string array

// generate setters.

}

In bean.xml file,

<bean id="c" class="com.demo.beans.Car">

<property name="carNames">

<list>

</list>

</property>

<value>Hindustan Moters</value>

<value>Tata Moters</value>

<value>Ashoka Leyland</value>

<property name="engines">

<list>

<ref bean="e1" />

<ref bean="e2" />

<ref bean="e3" />

</list>

</property>

</bean>

**How to inject values using List, Set and Map?**

<bean id = "javaCollection" class = "com.example.JavaCollection">

<property name = "addressList">

<list>

<value>India</value>

<value>Maharashtra</value>

<value>Pune</value>

</list>

</property>

<property name = "addressSet">

<set>

<value>India</value>

<value>Gujrat</value>

<value>Surat</value>

</set>

</property>

<property name = "addressMap">

<map>

<entry key = "1" value = "India"/>

<entry key = "2" value = "Goa"/>

<entry key = "3" value = "Panaji"/>

</map>

</property>

</bean>

**How to inject Vector from List hierarchy (write bean.xml)?**

<property name="empName">

<util:list list-class="java.util.Vector">

<value>ABC</value>

<value>XYZ</value>

</util:list>

</property>

How to inject Hashset from Set hierarchy (write bean.xml)?

<property name="empId">

<util:set set-class="java.util.HashSet ">

<value>SO11</value>

<value>SO12</value>

</util:set>

</property>

**How to inject Hashmap from Map hierarchy (write bean.xml)?**

<property name="empIdName">

<util:map map-class="java.util. HashMap ">

<entry key="WO1" value="ABC" />

<entry key="WO2" value="GHI" />

<entry key="WO3" value="MNO" />

</util:map>

</property>

**What are the types of Container?**

* Bean factory (Core) Container & Application Context Container.

**Difference between Bean Factory Container & Application Context Container?**

|  |  |
| --- | --- |
| **Bean Factory** | **Application Context** |
| It is core or legacy container. | It is advance container, it extends bean factory & have additional advance  properties. |
| It is used to develop desktop based  applications. | It is used to develop web based &  desktop based applications. |
| It is lazy loading. | It is eager loading. |
| It doesn’t support annotations. | It supports annotations. |

**How to make Application Context Lazy loading?**

* By using scope as Prototype, we can make application context to act as a lazy loading.

**What are bean scopes?**

There are five bean scopes.

* Singleton (same instance per IOC container)
* Prototype (any number of instances per IOC container)
* Request (Valid for Spring based applications, used for httprequest)
* Session (Valid for Spring based applications, used for httpsession)
* Global Session (Valid for Spring based applications, used for global httpsession)

**What is autowiring?**

* For injecting secondary type of dependency, we use autowiring.

**How to enable Autowiring?**

Add Spring context and <context:annotation-config /> in bean configuration file.

Include ‘AutowiredAnnotationBeanPostProcessor’ directly in bean configuration file.

<bean class = "org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcess or"/>

**What does autowire supports by-default?**

* By-default autowire supports byType.

**What is front controller?**

* It is also known as Dispatcher Servlet.
* It manages entire process.
* It find appropriate class as per request.

**What are Stereotype annotations? Explain them?**

* @Controller: Identifies class as a controller class & marks it as a bean.
* @Service: Identifies class as a service class & tells it has business logic.
* @Repository: Identifies class as a dao layer class & tells it has a database connection.

**Can we write @Repository for business logic class and @Service for dao logic class?**

* Yes we can shuffle @Repository & @Service.
* It is just to understand other programmers to identify business logic class and dao layer class.
* But we can’t replace @Controller class.

**What are annotations used in spring?**

@RequestMapping:

It is used at class level or method level.

It is used to map URL.

@RequestParam:

It brings single variable from client side.

E.g. Username, Password get from client side.

@ModalAttribute:

It brings complete POJO class from client side.

E.g. Registration form data from client side.

**What is server side validations or spring validations?**

Some annotations we can use as validations.

Annotations like-

* @NotEmptyValidation
* @SizeAnnotation
* @EmailAnnotation
* @NotNullAnnotation
* @Valid => Data from client side using request, then this checks data is as per validations or not.