Spring-Core

**What is Spring?**

Spring is a Dependency Injection framework to make java application loosely coupled

Spring is framework of java

Spring provide us IOC (Inversion of control container) from which helps we do dependency injection

Spring framework makes the easy development of javaEE application

It was developed by Rod johnson in 2003

**What is Dependency Injection?**

It is a Design pattern

Example:

|  |  |
| --- | --- |
| Class Abhay  {  Priya p;    Public void doWork(){  } } | Class Priya  {  Public void doWork(){  }  } |

**Spring and JEE Layers**

UI Layer (Spring MVC) ProductController

Business/Services Layer(Security) ProductService (Business Logic)

Data Access Layer ProductDao

Data Base Layer (spring jdbc,Spring ORM)

**Note:** If we want to call ProductDao from ProductService then we have to create the object of ProductDao inside the ProductService (ProductDao dao = new ProductDao()😉 but it is not recommended in spring framework because here we have multiple annotation which will inject our object with different classes that’s make our application as loosely coupled.

**Spring Modules.**

1. **Spring-Core**

. Core,Beans,Context,spEL(Spring expression language)

Note: above four module is a fundamental of Spring framework

AOP(**Aspect-Oriented Programming** ), Aspect, Instrumentation, Messaging

**2.Data Access/Integration**

JDBC, ORM(Object relational Mapping), JMS(Java Message Service), OXM(Object xml mapping)

**3. Web:**

Web, Servlet, Portlet, WebSocket

**4.Test**

Junit test

**What is Spring IoC Container?**

It is a predefine program and we get this with spring framework and it is responsible for Creating the objects, Holding them in a memory, Injecting them in another object as required

Note:- Beans, Config(xml), spring containers maintain the application through Bean(pojo-class) and with xml configuration

**ApplicationContext:** **(interface)**

In spring we have one ApplicationContext (interface) which represents the ioc container

**Classes:**

**ClasspathXMLApplicationContext:** it seraches the xml configuration from the java class path

**AnnotationConfigApplicationContext:**  It searches the annotatiom based configuration from the java class path

**FileSystemXMLApllicationContext:** It searches the config file from the filesystem

**DependencyInjection:**

Dependency Injection is a fundamental aspect of the Spring framework, through which the Spring container “injects” objects into other objects

Dependency Injection can be done in 2 ways

1.Setter Injection

2.Constructor Injection

**SetterInjection:** (sometimes we call setters injection as property injection)

Class Student{ Class Address(){

Id,name,Adress adress; street, city, state

setId(id){} setStreet(street){}

setName(name){} setCity(city){}

SetAddress(address){} setState(state){}  
} }

**Constructor Injection**

Class Student{ Class Address{

String id,name; String street,city,state;

Address address; Address(street,city,state){

Student(id,name,address){

}

}

}

**Configuration-File:**

Where we declare beans and its dependency

**Data Types(Dependencies)**

1. Primitive Data Types:(Byte,short,char,int,float,double,long,boolean)
2. Collection Type: (List, Set, Map, and Properties)
3. Reference Type: User define object reference

**Softwares:**

1. Eclipse/Netbeans/Intellij
2. TomcatServer
3. Mysql for database
4. Sqlyog, workbench or phpmyadmin for mysql gui

**How to Create the Project:**

1. Create a maven project
2. Adding dependencies:-spring core, spring context
3. Creating beans
4. Creating configuration file:-config.xml
5. Setter injection
6. Main class : which can pull the object and use

Open Eclipse: File->New->Maven-Project->arctype-quickstart->next->type project name and click on finish.

**Dependencies in Pom.xml:**

Search maven-repository on google and then search spring-core and spring-context

Note: Same version context and same version for core

**Injecting Reference Type:**

In real time application it used a lot now a days

**Constructor Injection: <constructor-args />**

**Ambiguity (Constructor):**

**Note:** When we assign any value in xml first it will treat as string

<!-- Ambiguity constructor Addition Class -->

<bean class="com.spring.constructor.Addition" name="addition">

<constructor-arg value="12" type="int" index="1"/>

<constructor-arg value="13" type="int" index="0"/>

</bean>

**Life Cycle Methods Of Spring Bean:**

Spring provide two important methods to every bean

1. Public void init()
2. Public void destroy()

Note: We can change the name of these method but signature must be same

Initialization code loading config, Connecting with DB, Webservice etc.

**Configure Technique:**

We can configure in three ways

1. Xml
2. Spring Interface
3. Annotation

**Implementing Life Cycle methods of spring bean using XML**

Note: To call init() and destro() method we have to call the registerShutdownHook() method and this method is in AbstractApplicationContext.

**Implementing Life Cycle methods of spring bean using Interfaces**

1. Initializing Bean (init())
2. Disposable Bean (Destroy ())

**Implementing Life Cycle methods of spring bean using Annotation**

1. @PostConstruct (init())
2. @PreDestroy (destroy())

Note: From Java 9 these two dependency is deprecated if you want to use it then add below dependency in your pom.xml file

<**dependency**>

<**groupId**>javax.annotation</**groupId**>

<**artifactId**>javax.annotation-api</**artifactId**>

<**version**>1.3.2</**version**>

</**dependency**>

Note: To enable Annotation tag we have to write <context:annotation-config/> in config.xml file

Note: If you want to enable only single annotation then you have to write <bean class=" org.springframework.context.annotation.CommonAnnotationBeanPostProcessor "/>

**Autowiring in Spring:**

* Feature of spring Framework in which spring container inject the dependencies automatically.
* Autowiring can't be used to inject primitive and string values. It only works with reference only.

When we link one object with another object is called wiring

We can do autowiring in two ways

1. Xml (byName, byType, constructor)
2. Annotations (@Autowired)

**Advantage:**

* Automatic
* Less code

**Disadvantage:**

* No control of programmer
* It can't be used for primitive and String values

**@Autowire:**

We can use on the object reference , on the setter methods and on the constructor.

**Note:** and in config.xml file we have to mentioned <context:annotation-config />

**@Qualifier Annotation:**

Note: if we have multiple beans in xml file then we can mention to our class below the @Autowired @Qualifier("temp2")

**Spring Stand Alone Collection:**

For stand-alone collection we are using util schema

<util:list list-class="java.util.LinkedList" id="myfriend">

<value>Kareem</value>

<value>Junaid</value>

<value>Atif</value>

<value>Ansar</value>

</util:list>

<bean class="com.spring.standalone.collection.Person" name="person">

<property name="friends" ref="myfriend"/>

</bean>