**INTERNITY FOUNDATION**

**TASK-16**

**Submitted By:**

**Amisha Singhal**

**Java Batch**

**Spring - Dependency Injection**

**Ans-** Dependency Injection is the main functionality provided by [Spring](https://www.geeksforgeeks.org/introduction-to-spring-framework/) IOC(Inversion of Control). The Spring-Core module is responsible for injecting dependencies through either Constructor or Setter methods. The design principle of Inversion of Control emphasizes keeping the Java classes independent of each other and the container frees them from object creation and maintenance. These classes, managed by [Spring](https://www.geeksforgeeks.org/introduction-to-spring-framework/), must adhere to the standard definition of Java-Bean. Dependency Injection in [Spring](https://www.geeksforgeeks.org/introduction-to-spring-framework/) also ensures loose-coupling between the classes.

**Example-**

**Employee.java**

It is a simple class containing two fields id and name. There are four constructors and one method in this class.

**package** com.Amisha;

**public** **class** Employee {

**private** **int** id;

**private** String name;

**public** Employee() {System.out.println("def cons");}

**public** Employee(**int** id) {**this**.id = id;}

**public** Employee(String name) {  **this**.name = name;}

**public** Employee(**int** id, String name) {

**this**.id = id;

**this**.name = name;

}

**void** show(){

    System.out.println(id+" "+name);

}

}

**applicationContext.xml**

We are providing the information into the bean by this file. The constructor-arg element invokes the constructor. In such case, parameterized constructor of int type will be invoked. The value attribute of constructor-arg element will assign the specified value. The type attribute specifies that int parameter constructor will be invoked.

<?xml version="1.0" encoding="UTF-8"?>

<beans

    xmlns="http://www.springframework.org/schema/beans"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xmlns:p="http://www.springframework.org/schema/p"

    xsi:schemaLocation="http://www.springframework.org/schema/beans

                http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

<bean id="e" **class**="com.Amisha.Employee">

<constructor-arg value="10" type="int"></constructor-arg>

</bean>

</beans>

**Test.java**

This class gets the bean from the applicationContext.xml file and calls the show method.

**package** com.Amisha;

**import** org.springframework.beans.factory.BeanFactory;

**import** org.springframework.beans.factory.xml.XmlBeanFactory;

**import** org.springframework.core.io.\*;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

        Resource r=**new** ClassPathResource("applicationContext.xml");

        BeanFactory factory=**new** XmlBeanFactory(r);

        Employee s=(Employee)factory.getBean("e");

        s.show();

    }

}

**Output: 10 null**

**Spring - Injecting Inner Beans**

**Ans -** Injecting Inner Beans in the Spring means beans that are defined within the scope of another bean whenever a bean is used for only one particular property. Inner bean can be configured both as **setter dependency using property tag and as constructor dependency using constructor-arg.**

Defining Person bean as an inner bean of Employee bean will mean that definition of Person bean is restricted within the scope of Employee bean.

**Example-**

***Person Class***

public class Person {

private String name;

private int age;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}

***XML Configuration***

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-4.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd">

<bean id="employeeBean" class="org.netjs.exp.Spring\_Example.Employee">

<property name="person">

<bean class="org.netjs.exp.Spring\_Example.Person">

<property name="name" value="Amisha" />

<property name="age" value="21" />

</bean>

</property>

</bean>

</beans>

***Java class to run application***

public class App {

public static void main( String[] args ){

ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext

("appcontext.xml");

Employee employee = (Employee) context.getBean("employeeBean");

System.out.println("Employee " + employee);

context.close();

}

}

***Output***

**Employee Name - AmishaAge – 21**

**Spring - Injecting Collection**

**Ans-** We can inject collection values by constructor in spring framework. Spring framework provides four type of collection (<list>, <set>, <map> and <props>) element to configure the collections.

* **<list>**

This element is used to wire a list of values and is used for List and Arrays type of objects in java. Since list and arrays allows duplicates, so the duplicates are supported in <list> tag also. This tag can be used for both simple types and beans.

### ****<set>****

This element is used to wire a list of values and is used for Set type in java. Since set does allows duplicates, so the duplicates are not supported in <set> tag also. This tag can be used for both simple types and beans.

* **<map>**

This element is used to wire a key-value pair and used for java.util.Map type in java.

In Maps, key and values can be of simple types and can be of object types so <map> tag also support both simple type and beans at both key and value level.

**Elements supported by <map> tag are**

* <key> - to define a key of simple type
* <key-ref> - to define a key referring to a  bean
* <value> - to define simple type values
* <value-ref> - to define values referring to a bean

### ****<props> -****

### This element is used to define a java.util.Properties to wire a key-value pair similar to map with the only difference is that both key and value can be of type String only.

### Example-

### package com.Amisha;

### import java.util.Iterator;

### import java.util.List;

### public class Department {

### private String dept;

### private int deptId;

### private List<String> subjects;

### public Department(String dept,int deptId, List<String> subjects){

### this.dept = dept;

### this.deptId = deptId;

### this.subjects = subjects;

### }

### public void displayDetails(){

### System.out.println("Department: "+ dept);

### System.out.println("Department Id: "+ deptId);

### Iterator<String> deptItr = subjects.iterator();

### System.out.println("Subjects to study:");

### while(deptItr.hasNext()){

### System.out.println(deptItr.next());

### }

### }

### }

### XML Configuration - applicationContext.xml

### <?xml version="1.0" encoding="UTF-8"?>

### <beans

### xmlns="http://www.springframework.org/schema/beans"

### xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

### xmlns:p="http://www.springframework.org/schema/p"

### xsi:schemaLocation="http://www.springframework.org/schema/beans

### http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

### <bean id="departmentBean" class="com.Amisha.Department">

### <constructor-arg value="Computer Science Engineering"></constructor-arg>

### <constructor-arg value="1004"></constructor-arg>

### <constructor-arg>

### <list>

### <value>Java</value>

### <value>Spring</value>

### <value>C</value>

### <value>Python</value>

### </list>

### </constructor-arg>

### </bean>

### </beans>

### Test Class

### package com.Amisha;

### import org.springframework.context.ApplicationContext;

### import org.springframework.context.support.ClassPathXmlApplicationContext;

### public class TestDepartment {

### public static void main(String[] args) {

### ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

### Department subBean = (Department) context.getBean("departmentBean");

### subBean.displayDetails();

### }

### }

### OUTPUT

### Department: Computer Science Engineering

### Department Id: 1004

### Subjects to study:

### Java

### Spring

### C

### Python