Weekly Assignment – 6

Muktanand Gautam

Question no. – 1

App.Java

**package** company.demo\_noel;

**import** java.text.DecimalFormat;

**import** org.springframework.context.ApplicationContext;

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

**import** com.hamarabank.bean.BillCalculator;

**public** **class** App

{

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

{

//spring.xml file is invoked

ApplicationContext context=**new** ClassPathXmlApplicationContext("Spring.xml");

//down casting from object to String

BillCalculator billcal = (BillCalculator)context.getBean("billcal");

String pattern="\*\*.0";

DecimalFormat df=**new** DecimalFormat(pattern);

//calling method of POJO class

System.***out***.println("Total bill to pay is $"+(df.format(billcal.calculateBill())));

}

}

BillCalculator.Java

**public** **class** BillCalculator {

//data member

**public** **int** unit;

**public** **double** tax;

**public** **double** unitPrice;

//default constructor

**public** BillCalculator() {

**this**(0,0.0d,0.0d);

}

//parameterized constructor

**public** BillCalculator(**int** unit,**double** tax,**double** unitPrice) {

**this**.unit = unit;

**this**.tax = tax;

**this**.unitPrice = unitPrice;

}

//calculateBill method

**public** **double** calculateBill() {

**return** unitPrice+(unitPrice\*tax)\*unit;

}

}

Spring.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=*"billcal"* class=*"com.company.demo\_noel.BillCalculator"*>

<constructor-arg name = *"unit"* value=*"50.7"*></constructor-arg>

<constructor-arg name = *"tax"* value=*"7.8"*></constructor-arg>

<constructor-arg name = *"unitPrice"* value=*"10"*></constructor-arg>

</bean>

</beans>

Junit Test

package com.hamarabank.bean;

import static org.junit.Assert.assertEquals;

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.Test;

class TestBillCalculator {

//initialization by constructor

BillCalculator bilcal = new BillCalculator(50,7.8,10.0);

//calculateBill tested

@Test

void testBillCalculator() {

//fail("Not yet implemented");

assertEquals(578,bilcal.calculateBill(),0.4);

}

}

Pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>company</groupId>

<artifactId>demo\_noel</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>demo\_noel</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.23</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context-support -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>5.3.23</version>

</dependency>

</dependencies>

</project>

Question no.-2

**package** com.hamarabank.bean;

**public** **class** BillCalculator {

//data member

**public** **int** unit;

**public** **double** tax;

**public** **double** unitPrice;

//default constructor

**public** BillCalculator() {

**this**(0,0.0d,0.0d);

}

//parameterized constructor

**public** BillCalculator(**int** unit,**double** tax,**double** unitPrice) {

**this**.unit = unit;

**this**.tax = tax;

**this**.unitPrice = unitPrice;

}

//calculateBill method

**public** **double** calculateBill() {

**return** 10.0+(10.0\*7.8)\*50.7;

}

}

Spring Configuration.java

**package** com.hamarabank.config;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.Configuration;

@Configuration

@ComponentScan(basePackages = "com.hamarabank.bean")

**public** **class** SpringConfig {

}

App.java

**package** com.hamarabank.ui;

**import** java.text.DecimalFormat;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.annotation.AnnotationConfigApplicationContext;

**import** com.hamarabank.bean.BillCalculator;

**import** com.mybank.configuration.SpringConfiguration;

**public** **class** App {

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

// **TODO** Auto-generated method stub

ApplicationContext applicationContext = **new** AnnotationConfigApplicationContext(SpringConfiguration.**class**);

//Spring's Application Context initializes the beans by calling getBean method

BillCalculator bilcal = (BillCalculator) applicationContext.getBean(BillCalculator.**class**);

String pattern="\*\*.0";

DecimalFormat df=**new** DecimalFormat(pattern);

//Bean class(BillCalculator) method is called

System.***out***.println("Total bill to pay is $"+(df.format(bilcal.calculateBill())));

}

}

Pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.hamarabank</groupId>

<artifactId>demo\_noel</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>demo\_noel</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.23</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context-support -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>5.3.23</version>

</dependency>

</dependencies>

</project>

Question no. – 3

Dependency Injection –

The Dependency Injection is a design pattern that removes the dependency of the programs. In such case we provide the information from the external source such as XML file. It makes our code loosely coupled and easier for testing.

The Dependency Injection pattern involves 3 types of classes-

1. Client Class: The client class (dependent class) is a class which depends on the service class.

2. Service Class: The service class (dependency) is a class that provides service to the client class.

3. Injector Class: The injector class injects the service class object into the client class.

Ex.

1. **class** Employee{
2. Address address;
4. Employee(Address address){
5. **this**.address=address;
6. }
7. **public** **void** setAddress(Address address){
8. **this**.address=address;
9. }
10. }

Types of Dependency Injection:

As you have seen above, the injector class injects the service (dependency) to the client (dependent). The injector class injects dependencies broadly in three ways: through a constructor, through a property, or through a method.

1.Constructor Injection

2.Property Injection

3.Method Injection

1.Constructor Injection: In the constructor injection, the injector supplies the service (dependency) through the client class constructor.

2.Property Injection: In the property injection (aka the Setter Injection), the injector supplies the dependency through a public property of the client class.

3.Method Injection: In this type of injection, the client class implements an interface which declares the method(s) to supply the dependency and the injector uses this interface to supply the dependency to the client class.

Example of Constructor Injection:

<bean id = “projet” class=”app.myspringapp.Project”>

<constructor-args value = “Hello” ></constructor-args>

<contructor-args value = “Java”></constructor-args>

</bean>

Inversion of Control : Spring Ioc (Inversion of Control) Container is the core of Spring Framework. It creates the objects, configures and assembles their dependencies, manages their entire life cycle. The Container uses Dependency Injection(DI) to manage the components that make up the application. It gets the information about the objects from a configuration file(XML) or Java Code or Java Annotations and Java POJO class. These objects are called Beans. Since the Controlling of Java objects and their lifecycle is not done by the developers, hence the name Inversion of Control.

There are 2 types of Ioc containers:

1.Bean Factory

2.Application Context

That means if you want to use an IoC container in spring whether we need to use a BeanFactory or ApplicationContext. The BeanFactory is the most basic version of IoC containers, and the ApplicationContext extends the features of BeanFactory. The followings are some of the main features of Spring IoC,

• Creating Object for us.

• Managing our objects.

• Helping our application to be configurable.

• Managing dependencies.

Example of Ioc Containers:

public class Mobile {

public static void main(String[] args) {

// Using ApplicationContext tom implement Spring IoC

ApplicationContext applicationContext = new ClassPathXmlApplicationContext("beans.xml");

// Get the bean

Sim sim = applicationContext.getBean("sim", Sim.class);

Calling the methods

sim.calling();

sim.data();

}

}