Gaurav Bhatt

WEEK6 ASSIGNMENT

QUESTION 1-

App.java

```
package com.espire.weekly_assignment_q1;
//importing required classes
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

//main class
public class App {
	public static void main(String[] args) {

//spring.xml file invoked
	 ApplicationContext context = new
ClassPathXmlApplicationContext("Spring.xml");
		BillCalculator bill = (BillCalculator) context.getBean("bill");
		System.out.printf("Total bill to pay is
$%.2f",bill.calculateBill());
```

BillCalculator.java

```
package com.espire.weekly_assignment_q1;

public class BillCalculator{
//fields
private int units;
private double tax;
private double unitPrice;

//parameterized constructor
public BillCalculator(int units, double tax, double unitPrice) {
	super();
	this.units = units;
	this.tax = tax;
	this.unitPrice = unitPrice;
}
//method
public Double calculateBill(){
	double Total = (units*unitPrice)+tax;
	return Total;}
}
```

Gaurav Bhatt

WEEK6 ASSIGNMENT

Spring.xml

```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="bill" class="com.espire.weekly_assignment_q1.BillCalculator">
<constructor-arg value="100"></constructor-arg>
<constructor-arg value="18"></constructor-arg>
<constructor-arg value="10"></constructor-arg>
</bean>
</beans>
```

Junit test

```java
package com.espire.weekly_assignment_q1;

import static org.junit.Assert.assertEquals;

import org.junit.jupiter.api.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
//Testing the BillCalculator class

class BillCalculatorTest {
	ApplicationContext context = new
ClassPathXmlApplicationContext("Spring.xml");
	BillCalculator bill = (BillCalculator) context.getBean("bill");
//testing calculate bill method
	@Test
	void testCalculateBill() {
//assertEquals(expected,actual,delta);
delta to deviate from expected by scale of 0.4 from actual
	assertEquals(1018.0,bill.calculateBill(),0.4);}}
```

Gaurav Bhatt

WEEK6 ASSIGNMENT

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.espire</groupId>
  <artifactId>weekly_assignment_q1</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>weekly_assignment_q1</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-core
-->
            <dependency>
                  <groupId>org.springframework</groupId>
                  <artifactId>spring-core</artifactId>
                  <version>5.3.23</version>
            </dependency>
            <!--
https://mvnrepository.com/artifact/org.springframework/spring-context-
support
Spring dependency -->
            <dependency>
                  <groupId>org.springframework</groupId>
                  <artifactId>spring-context-support</artifactId>
                  <version>5.3.23</version>
                  </dependency>
  </dependencies>
</project>
```

Gaurav Bhatt

WEEK6 ASSIGNMENT

Question 2-

BillCalculator.java

```java
package com.bill_calculator.bean;

public class BillCalculator {
//fields
	private int units;
	private double tax;
	private double unitPrice;
//parameterized constructor
	public BillCalculator(int units, double tax, double unitPrice) {
		super();
		this.units = units;
		this.tax = tax;
		this.unitPrice = unitPrice;
	}
//calculatBill method
	public Double calculateBill(){
		double Total = (units*unitPrice)+tax;
		return Total;	}

}
```

SpringConfig.java

```java
package com.bill_calculator.configuration;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

import com.bill_calculator.bean.BillCalculator;
//bean class package is invoked by the help of annotation
@Configuration
@ComponentScan(basePackages="com.bill_calculator.bean")
public class BillConfig {
	@Bean
	public BillCalculator bill() {
		int units=100;
		double tax=18;
		double unitPrice=10;
		return new BillCalculator(units,tax,unitPrice);
```

Gaurav Bhatt

WEEK6 ASSIGNMENT

App.java

```
package com.bill_calculator.ui;

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

import com.bill_calculator.bean.BillCalculator;
import com.bill_calculator.configuration.BillConfig;

//main class(user interface class)

public class App {
      public static void main(String[] args) {
//Configuration class is invoked
            ApplicationContext applicationContext = new
AnnotationConfigApplicationContext(BillConfig.class);
//Spring's Application Context initializes the beans by calling getBean
method
            BillCalculator bill =
applicationContext.getBean(BillCalculator.class);
            System.out.printf("Total bill to pay is
%.2f",bill.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.espire</groupId>
      <artifactId>weekly_assignment</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <packaging>jar</packaging>

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

      <properties>
            <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
            <maven.compiler.target>1.</maven.compiler.target>
            <maven.compiler.source>1.8</maven.compiler.source>
      </properties>

      <dependencies>
            <dependency>
                  <groupId>junit</groupId>
```

```
                <artifactId>junit</artifactId>
                <version>3.8.1</version>
                <scope>test</scope>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>5.3.18</version>
            </dependency>
        </dependencies>
</project>
```

Question 3-

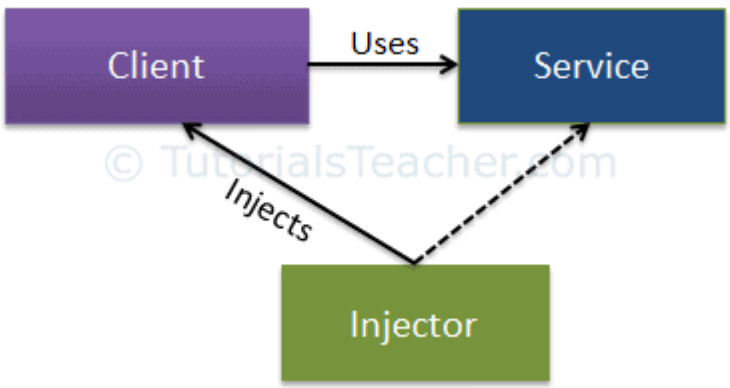
Dependency Injection:

Dependency Injection (DI) is a design pattern used to implement IoC. It allows the creation of dependent objects outside of a class and provides those objects to a class through different ways. Using DI, we move the creation and binding of the dependent objects outside of the class that depends on them.

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

The following figure illustrates the relationship between these classes:



Dependency Injection

Gaurav Bhatt

WEEK6 ASSIGNMENT

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="project" class="app.myspringapp.Project">
        <constructor-arg value="MIS"></constructor-arg>
        <constructor-arg value="San Diago"></constructor-arg>
</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:

Gaurav Bhatt

WEEK6 ASSIGNMENT

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,

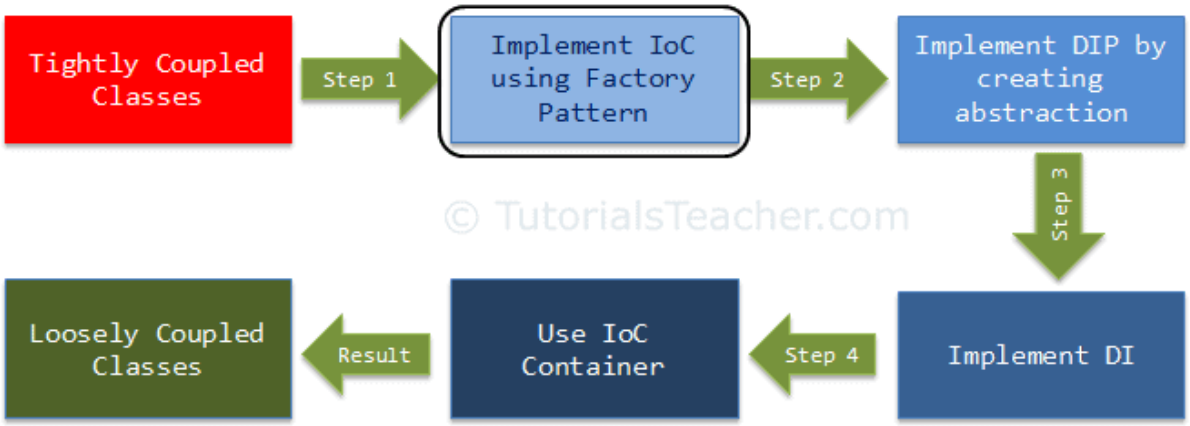
- Creating Object for us.
- Managing our objects.
- Helping our application to be configurable.
- Managing dependencies.

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();
    }
}
```

And now if you want to use the Airtel sim so you have to change only inside the **beans.xml** file. The main method is going to be the same.

```
<bean id="sim" class="Airtel"></bean>
```



Steps of Ioc Containers

Gaurav Bhatt

WEEK6 ASSIGNMENT