Ecole Normale Supérieure de l'Enseignement Technique Mohammedia

Université Hassan II de Casablanca

ENSET

ا لمدرسة العليا لأساتذة التعليم التقنيالمحمدية

جامعة الحسن الثاني بالدار البيضاء

Département Mathématiques et Informatique

Cycle Ingénieur

« Ingénierie Informatique – Big Data et Cloud Computing »

COMPTE-RENDU:
Activité pratique N°1

.

Réalisé par: Asmaa ELASRI

Classe : II-BDCC2

Année universitaire : 2021 / 2022

Inversion de contrôle et injection de dépendances

1. Création de l'interface IDao

```java
package dao;

public interface IDao {
    public double getData();
}
```

2. Création d'une implémentation de cette interface

```java
package dao;

import org.springframework.stereotype.Component;

@Component("dao")
public class DaoImpl implements IDao{

    @Override
    public double getData() {
        System.out.println("Version base de donnees");
        double temp=Math.random()*40;
        return temp;
    }
}
```

3. Création de l'interface IMetier

```java
package metier;

public interface IMetier {
    public double calcul();
}
```

4. Création d'une implémentation de cette interface en utilisant le couplage faible

```java
package metier;

import dao.IDao;
import org.springframework.stereotype.Component;

@Component("metier")
public class MetierImpl implements IMetier{
 //Couplage faible
   private IDao dao;
    public MetierImpl(IDao dao) {
        this.dao = dao;
    }
    @Override
    public double calcul() {
        double tmp=dao.getData();
        double res=tmp*540/Math.cos(tmp*Math.PI);
        return res;
    }
/* Injecter dans la variable dao un objet d'une classe qui implemente l'injection IDao*/
    public void setDao(IDao dao) {
        this.dao = dao;
    }
}
```

5. L'injection des dépendances :

a. Par instanciation statique

```java
package pres;
import ext.DaoImpl2;
import metier.MetierImpl;

public class PresentationStatique {

    public static void main(String[] args)
    {
        /*
           Injection des dépendences par instanciation statique ==> new
         */
        DaoImpl2 dao = new DaoImpl2();
        MetierImpl metier = new MetierImpl();
        metier.setDao(dao);
        System.out.println(metier.calcul());
    }
}
```

Et voici l'implementation de l'interface utilisee (DaoImpl2) version capteur :

```java
package ext;

import dao.IDao;

public class DaoImpl2 implements IDao {
    @Override
    public double getData() {
        System.out.println("Version Capteurs");
        double temp=1000;
        return temp;
    }
}
```

b. Par instanciation dynamique

Dans cette classe on a utilisée le couplage faible et le fichier config.txt qui contient les classes des implémentations des deux interfaces utilisées

```java
package pres;

import ...
public class PresentationDynamique {
    public static void main(String[] args) throws Exception {

            Scanner scanner=new Scanner(new File( pathname: "config.txt"));

            String daoClassname=scanner.next();
            String metierClassName=scanner.next();
            Class cdao=Class.forName(daoClassname);
            IDao dao= (IDao) cdao.newInstance();

            Class cmetier=Class.forName(metierClassName);
            IMetier metier=(IMetier) cmetier.newInstance();

            Method meth=cmetier.getMethod( name: "setDao",IDao.class);

            meth.invoke(metier,dao);
            System.out.println(metier.calcul());
    }
}
```

Le fichier « config.txt »

config.txt

```
dao.DaoImpl
metier.MetierImpl
```

c. En utilisant le Framework Spring

❖ Version XML

➢ **On utilise le fichier pom.xml qui contient les dépendances**

spring-core

spring-context

spring-beans

```xml
<properties>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
</properties>
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>5.3.16</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.16</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-beans -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-beans</artifactId>
        <version>5.3.16</version>
    </dependency>
```

- applicationContext.xml :4

applicationContext.xml

Application context not configured for this file — Configure application c

```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"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/
<bean id="dao" class="ext.DaoImpl2"></bean>
<bean id="metier" class="metier.MetierImpl">
<!--    <property name="dao" ref="dao"></property>-->
<constructor-arg ref="dao"></constructor-arg>
</bean>

</beans>
```

-SpringXML :

```java
package pres;

import metier.IMetier;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class PresSpringXML {
    public static void main(String[] args) {
        ApplicationContext context=
                new ClassPathXmlApplicationContext( configLocation: "applicationContext.xml");
        IMetier metier=(IMetier) context.getBean( s: "metier");
        System.out.println("Resultat=>"+metier.calcul());
    }
}
```

❖ **Version annotations**

```java
package pres;

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

public class PresSpringAnnotations {
    public static void main(String[] args) {
        ApplicationContext context=new AnnotationConfigApplicationContext( ...basePackages: "dao","metier");
        IMetier metier=context.getBean(IMetier.class);
        System.out.println(metier.calcul());
    }
}
```