Ecole Normale Supérieure de l'Enseignement Technique Mohammedia

Université Hassan II de Casablanca

ENSET

ا لمدرسة العليا لأساتذة التعليم التقني المحمدية

جامعة الحسن الثاني بالدار البيضاء

Département Mathématiques et Informatique

Cycle Ingénieur

GLSID»

COMPTE-RENDU:
Activité pratique N°1

.

Réalisé par: TARMOUN AMAL

Classe : GLSID2

Année universitaire : 2021 / 2022

Inversion de contrôle et injection de dépendances

1. Création de l'interface IDao

```java
package dao;

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

2. Création d'une implémentation de cette interface

```java
package dao;

import org.springframework.stereotype.Component;

@Component("dao") // au demmarage a chaque fois qu'il trouve une classe qui commance par l'annotaion component
//il va l'instancier et lui donner comme nom dao
public class DaoImpl implements IDao{
    @Override
    public double getData() {
        /*
        se connetcter a BDD pour recuperer la temperature
        */
        System.out.println("versionn de base de donnes ");
        double temp=Math.random()*40;
        return temp;
    }
}
```

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

```java
package extention;

import ...

@Component("dao2") // avec l'injection autowired on va avoir an exception
public class DaoImpl2 implements IDao {
    @Override
    public double getData() {
        System.out.println("version capteur");
        double temp = 80;
        return temp;
    }
}
```

3. Création de l'interface IMetier

```java
package metier;

public interface IMetier {
    double calcul();

}
```

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

```java
package metier;

import ...

@Component("metier")
public class MetierImpl implements IMetier{
    // couplage faible
   //@Autowired // demander a spring au moment ou tu vas instancier la classe metierIml cherche moi
    // parmi les objets que tu a deja creer si tu trouve un   objet de type IDao tu va l'injecter dand cette variavle
  private IDao dao ;
    // injection en utilisant un constructeur avec parametres

// spring va faire l'injection via le constructeur
    public MetierImpl(IDao dao) { this.dao = dao; }

    @Override
    public double calcul() {
        double temp=dao.getData();
        double res = temp*2;
        return res;
    }
// injecter dans la variable dao un objet d'une classe qui impelemente l'interface IDao
    public void setDao(IDao dao) { this.dao = dao; }
}
```

5. L'injection des dépendances :

a. Par instanciation statique

```java
package presentation;

import ...

public class presentation {
    public static void main(String[] args) {
        /* injection des dependances par instanciation statique  c,a,d l'utilisation du new */
        DaoImpl2 dao  = new DaoImpl2();
        MetierImpl metier = new MetierImpl(dao);
        //metier.setDao(dao);
        System.out.println("Resultat"+metier.calcul());
    }
}
```

b. Par instanciation dynamique

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

```java
package presentation;

import ...

public class presentation2 {
    public static void main(String[] args) throws FileNotFoundException, ClassNotFoundException,
            InstantiationException,
            IllegalAccessException, NoSuchMethodException, InvocationTargetException {
        Scanner scanner = new Scanner(new File( pathname: "config.txt"));
        String daoClassName = scanner.nextLine();
        Class cDao = Class.forName(daoClassName);
        IDao dao  =(IDao)cDao.newInstance();
        System.out.println(dao.getData());


        String metierClassName = scanner.nextLine();
        Class cMetier = Class.forName(metierClassName);
        IMetier metier = (IMetier) cMetier.newInstance();

        Method method = cMetier.getMethod( name: "setDao",IDao.class);
        // metier .setDao(dao);
        method.invoke(metier,dao);
        System.out.println("Resultat=>"+metier.calcul());

    }
}
```

Le fichier « config.txt »

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

c. En utilisant le Framework Spring

❖ Version XML

-

applicationContext.xml :4

```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:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org
       /schema/beans/spring-beans.xsd http://www.springframework.org/schema/context https://www.springframework.org/schema/co
<!--    // creer moi un objet dao de type DaoImpl -->
    <bean id="dao" class =" dao.DaoImpl"></bean>

    <bean id="metier" class="metier.MetierImpl">
<!--        mainteant fait moi l'injection de depandence -->
<!--        injection via un setter -->
<!--        <property name="dao" ref="dao"></property>-->
<!--        injection via constructeur -->
<!--        quand tu vas instancier fait appelle a une constructeur avec parametre et lui transmet un objet de type dao -->
        <constructor-arg ref="dao"></constructor-arg>

    </bean>


</beans>
```

-SpringXML :

```java
package presentation;

import ...

public class presentationSpringXml {
    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 presentation;

import ...

public class PresentatioonSpringAnnotation {
    public static void main(String[] args) {
        // donnaer les package qui va scanner
        ApplicationContext context = new AnnotationConfigApplicationContext( ...basePackages: "dao" ,"metier");
        // donne moi un bean qui implemente l'interface IMetier
        IMetier metier =context.getBean(IMetier.class);
        System.out.println(metier.calcul());
    }
}
```