

DÉPARTEMENT MATHÉMATIQUES ET INFORMATIQUE

Compte rendu de l'Activité pratique N°1 :
Inversion de contrôle
et Injection des dépendances

Filière :
**« Ingénierie Informatique : Big Data et Cloud
Computing »**
II-BDCC2

Créé par : Mohamed Ait Lahcen

Introduction :

Dans ce compte rendu je vais mettre en détails la réalisation de l'activité pratique qui consiste à traiter le concept de l'inversion du contrôle et l'injection des dépendances, alors la réalisation de cette activité est passée par l'ensemble des parties suivantes :

1-La création de l'interface IDao.

2-L'implémentation de cette interface.

3-La création de l'interface IMetier.

4-La création d'une implémentation de cette interface en utilisant le couplage faible

5.L'injection des dépendances :

a. Par instanciation statique.

b. Par instanciation dynamique.

c. En utilisant le Framework Spring .

- Version XML.

- Version annotations.

1-L'interface IDao :

```
MetierImpl.java IDao.java ImplDao.java ImplMetier.java
1 package dao;
2
3 public interface IDao {
4 double getValue();
5 }
6
```

2-L'implémentation de l'interface IDao :

```
MetierImpl.java IDao.java ImplDao.java ImplMetier.java "11
1 package dao;
2
3 import org.springframework.stereotype.Component;
4
5 @Component
6 public class ImplDao implements IDao{
7
8     public double getValue() {
9
10         return Math.random()*10;
11     }
12
13 }
14
```

3-L'interface IMetier :

```
IDao.java ImplDao.java IMetier.java ImplMetier.java "11
1 package metier;
2
3 public interface IMetier {
4 public double calculer();
5 }
6
```

4- L'implémentation de l'interface IMetier:

```
MetierImpl.java  IDao.java  ImplDao.java  IMetier.java  IMetier.java  ImplMetier.java
1 package metier;
2
3 import dao.IDao;
4
5 public class MetierImpl implements IMetier{
6     private IDao dao;
7
8     @Override
9     public double calculer() {
10         double res= dao.getValue()*Math.random()*2;
11         return res;
12     }
13     public IDao getDao() {
14         return dao;
15     }
16     public void setDao(IDao dao) {
17         this.dao = dao;
18     }
19
20 }
21
```

5-L'injection des dépendances par instanciation statique :

```
MetierImpl.java  ImplMetier.java  presentation.java  *injection_stat  config.txt
1 package presentation;
2
3 import dao.DaoImpl;
4 import metier.MetierImpl;
5
6 public class injection_statique {
7     public static void main(String[] args) {
8         //the static method to inject dependency :
9         MetierImpl metier=new MetierImpl();
10        DaoImpl dao = new DaoImpl();
11        metier.setDao(dao);
12        System.out.println(metier.calculer());
13    }
14 }
15
```

a-L'injection des dépendances par instantiation dynamique:

```
MetierImpl.java IMetier.java ImplMetier.java *presentation.j *injection_stat config.txt »11
1 package presentation;
2
3 import java.io.File;
12
13 public class presentation {
14
15     public static void main(String[] args) throws FileNotFoundException, ClassNotFoundException {
16         //dynamic method to inject dependency :
17         File file = new File("config.txt");
18         Scanner sc = new Scanner(file);
19         String daoClassName = sc.nextLine();
20         String metierClassName = sc.nextLine();
21         Class cDao = Class.forName(daoClassName);
22         DaoImpl dao = (DaoImpl) cDao.newInstance();
23         Class cMetier = Class.forName(metierClassName);
24         MetierImpl metier = (MetierImpl) cMetier.newInstance();
25         Method meth = cMetier.getMethod("setDao", IDao.class);
26         meth.invoke(metier, dao);
27         System.out.println("la valeur est "+metier.calculer());
28
29     }
30 }
31
32 }
33
```

Le fichier config.txt qui contient les références des
classes à instancier dynamiquement :

```
MetierImpl.java config.txt IMetier.java
1 dao.DaoImpl
2 metier.MetierImpl
```

b-L'injection des dépendances en utilisant le framework spring version xml :

-Le fichier config.xml contient l'ensemble des classes à
instancier sous forme des beans:

```
MetierImpl.java  config.txt  *config.xml  ImplMetier.java  *presentation.j  *injection_stat
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://www.springframework.org/schema/beans
5     http://www.springframework.org/schema/beans/spring-beans.xsd">
6 <bean id="d" class="ext.ImplDao2"> </bean>
7 <bean id="metier" class="metier.ImplMetier">
8 <property name="dao" ref="d"></property>
9 </bean>
10
11 </beans>
12
```

Voilà l'utilisation du fichier xml pour avoir les instances
des classes déclarés dans ce fichier :

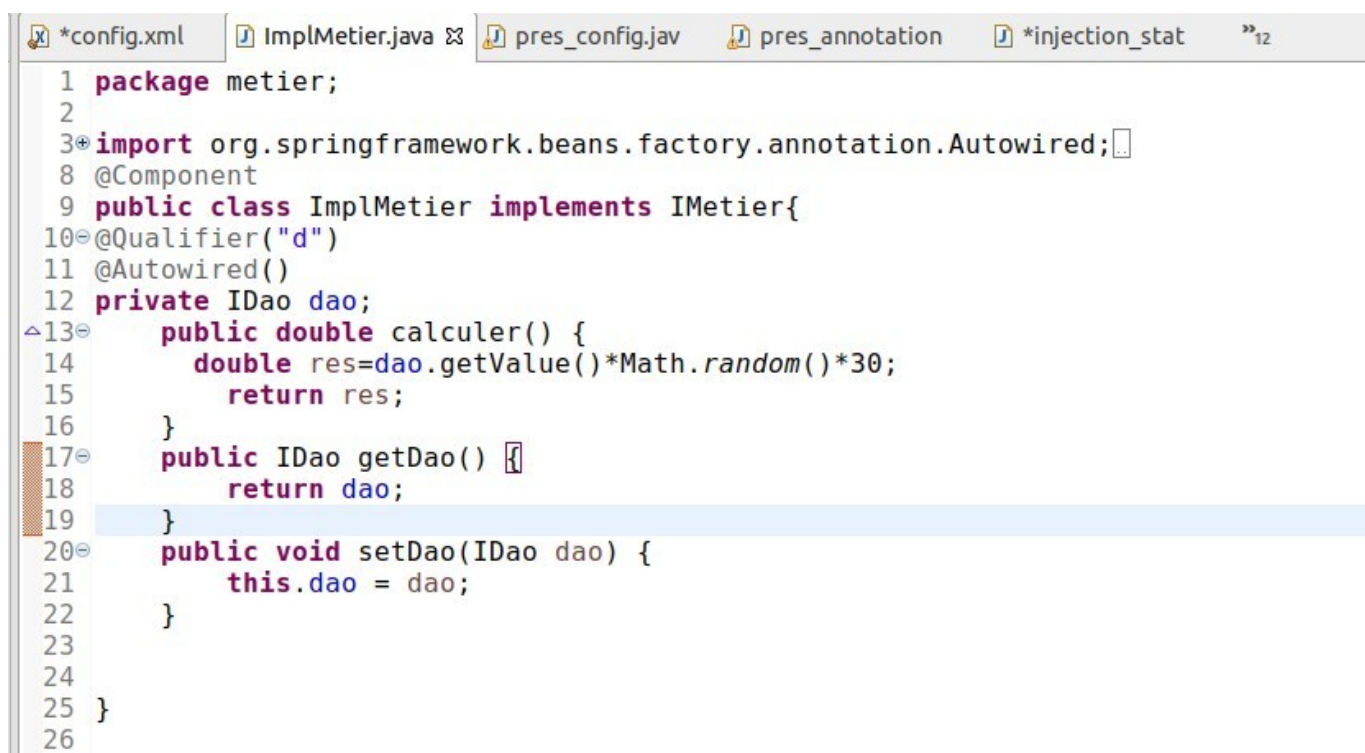
```
MetierImpl.java  config.txt  *config.xml  *presentation.j  pres_config.jav  *injection_stat
1 package presentation;
2
3 import java.applet.AppletContext;
11
12 public class pres_config {
13 public static void main(String[] args) {
14     ApplicationContext context=new ClassPathXmlApplicationContext("config.xml");
15     IMetier metier=(IMetier) context.getBean("metier");
16     System.out.println(metier.calculer());
17 }
18 }
19
```


c-L'injection des dépendances en utilisant le framework spring version annotation:

cette version consiste à ajouter l'annotation

@Component aux classes qu'on va instancier au moment de leurs déclarations et l'annotation **@Autowired** aux interfaces qui vont prendre comme valeur une implémentation déclaré avec l'annotation **@Component** :

voila l'exemple dans lequel nous avons utilisé les annotations pour déclarer les implémentions des interfaces IDao et IMetier :

A screenshot of an IDE window showing the implementation of the IMetier interface. The code is in a file named ImplMetier.java. It includes imports for Spring's @Component and @Autowired annotations. The class ImplMetier implements IMetier and has a private IDao dao field. It has a calculer() method that uses dao.getValue() and a getDao() method that returns dao. The setDao() method is also present. The code is as follows:

```
1 package metier;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5 @Component
6 public class ImplMetier implements IMetier{
7     @Qualifier("d")
8     @Autowired()
9     private IDao dao;
10
11     public double calculer() {
12         double res=dao.getValue()*Math.random()*30;
13         return res;
14     }
15
16     public IDao getDao() {
17         return dao;
18     }
19
20     public void setDao(IDao dao) {
21         this.dao = dao;
22     }
23
24 }
25
26 }
```

```
ImplDao.java IMetier.java ImplMetier.java pres_config.java pres_a

1 package dao;
2
3 import org.springframework.stereotype.Component;
4
5 @Component
6 public class ImplDao implements IDao{
7
8     public double getValue() {
9
10         return Math.random()*10;
11     }
12
13 }
14
```

Dans la classe ci-dessous j'ai créé une instance de la classe qui implémente l'interface IMetier en utilisant les annotations .

```
ImplDao.java IMetier.java ImplMetier.java pres_config.java *pres_annotatio 12

1 package presentation;
2
3 import org.springframework.context.ApplicationContext;
4
5 public class pres_annotation {
6
7     public static void main(String[] args) {
8         ApplicationContext context =
9             new AnnotationConfigApplicationContext("ext","metier","dao");
10         IMetier metier = context.getBean(IMetier.class);
11         System.out.println(metier.calculer());
12     }
13 }
14
15
16
17
18
19
20
```