

DEPARTEMENT MATHÉMATIQUES ET INFORMATIQUE

Filière :
« Génie du Logiciel et des Systèmes Informatiques Distribués »
GLSID

Compte rendu:
Activité 2 du JEE

Année Universitaire : 2021-2022

Réalisé par :

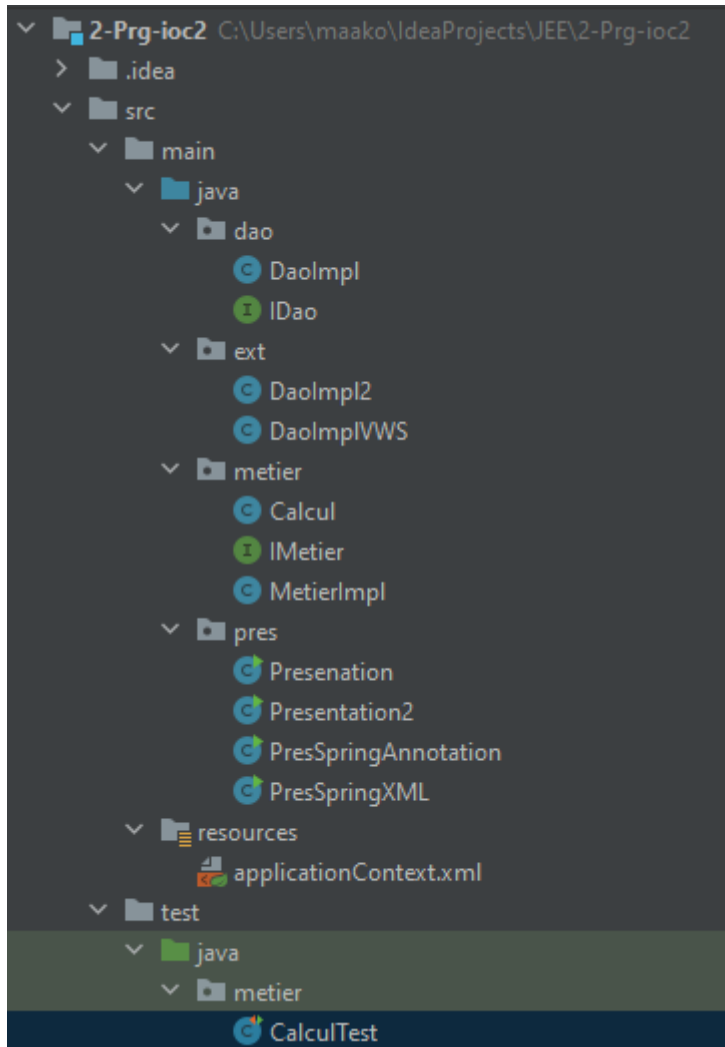
Amina MAAKOUL

Encadré par :

M. Mohamed YOUSSEFI

Activité 2 : Injection des dépendances par instanciation dynamique avec Spring

La structure du projet :



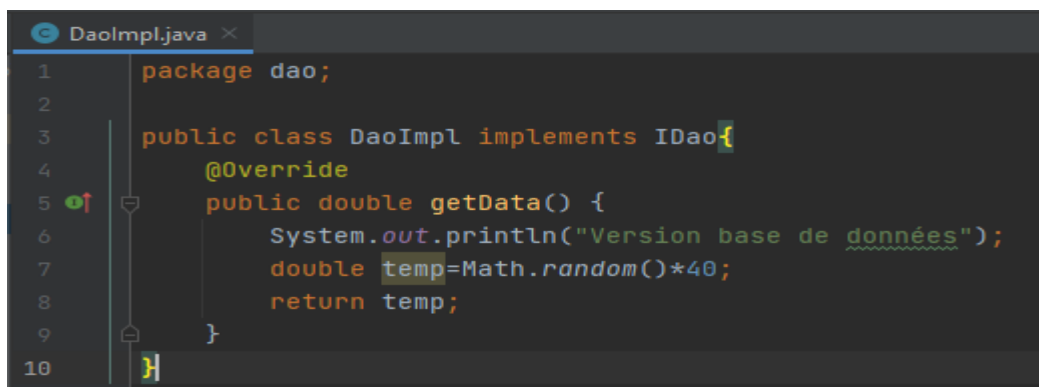
I. Le package « dao »

L'interface « IDao »

```
IDao.java x
1 package dao;
2
3 public interface IDao {
4     double getData();
5 }
```

La classe « DaoImpl »

- Spring et XML

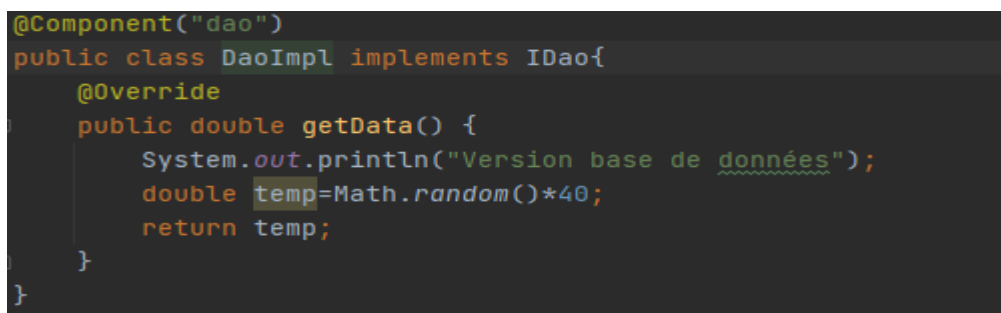


```

1 package dao;
2
3 public class DaoImpl implements IDao{
4     @Override
5     public double getData() {
6         System.out.println("Version base de données");
7         double temp=Math.random()*40;
8         return temp;
9     }
10 }

```

- Spring et Annotation



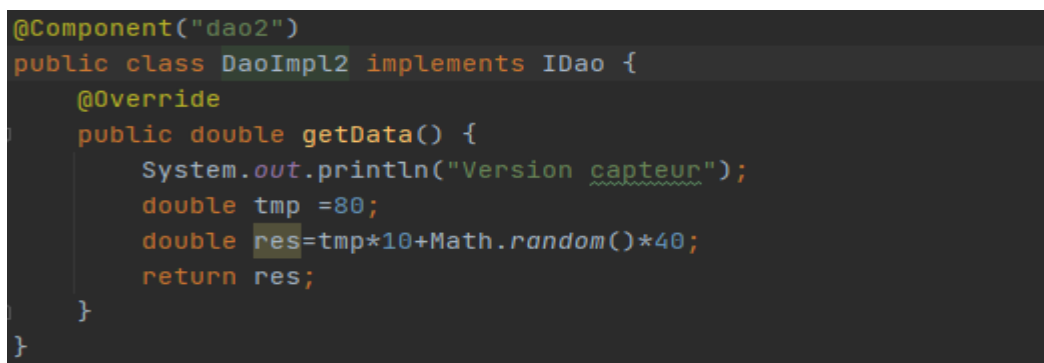
```

@Component("dao")
public class DaoImpl implements IDao{
    @Override
    public double getData() {
        System.out.println("Version base de données");
        double temp=Math.random()*40;
        return temp;
    }
}

```

II. Le package « ext »

La classe (Extension) « DaoImpl2 »



```

@Component("dao2")
public class DaoImpl2 implements IDao {
    @Override
    public double getData() {
        System.out.println("Version capteur");
        double tmp =80;
        double res=tmp*10+Math.random()*40;
        return res;
    }
}

```

La classe (Extension) « DaoImplVWS »



```

@Component("dao3")
public class DaoImplVWS implements IDao {
    @Override
    public double getData() {
        System.out.println("Version web service");
        return 90;
    }
}

```

III. Le package « métier »

L'interface « IMetier »

```
package metier;

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

La classe « MetierImpl »

- Version Spring at XML

```
public class MetierImpl implements IMetier {
    private IDao dao=null;
    @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 implémente l'interface IDao
     */
    public void setDao(IDao dao) { this.dao = dao; }
}
```

- Version Spring et Annotation

```
@Component
public class MetierImpl implements IMetier {
    @Autowired
    @Qualifier("dao")
    private IDao dao=null;
    @Override
    public double calcul() {
        double tmp=dao.getData();
        double res=tmp*540/Math.cos(tmp*Math.PI);
        return res;
    }
}
```

IV. Le package « pres »

- Injection des dépendances par instanciation dynamique avec Spring et XML

Le fichier de configuration « applicationContext.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/spring-beans.xsd">
    <bean id="d" class="dao.DaoImpl"></bean>
    <bean id="metier" class="metier.MetierImpl">
        <property name="dao" ref="d"></property>
    </bean>
</beans>
```

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

L'exécution :

```
PresSpringXML x
"C:\Program Files\Java\jdk1.8.0_171\bin\java.exe" ...
Version base de données
5444.749014739917
```

- Injection des dépendances par instanciation dynamique avec Spring et Annotation

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

L'exécution :

```
PresSpringAnnotation x
"C:\Program Files\Java\jdk1.8.0_171\bin\java.exe" ...
Version base de données
102387.6261143437
```

V. Le test unitaire

D'abord nous avons ajouter une classe « calcule » dans le package « metier »

```

Calcul.java x
1 package metier;
2
3 public class Calcul {
4     double somme(double a, double b) { return a+b; }
7 }

```

Puis dans le répertoire test\java\metier nous avons créer un test unitaire du classe « calcul.java »

```

public class CalculTest {
    private Calcul calcul;
    @Test
    public void testSomme(){
        calcul=new Calcul();
        double a=5; double b=9;
        double expected=14;
        double res=calcul.somme(a,b);
        Assert.assertTrue( condition: res==expected);
        System.out.println("ok");
    }
}

```

L'exécution

```

v ✓ CalculTest (metier) 2 ms "C:\Program Files\Java\jdk1.8.0_171\bin\java.exe" ...
  ✓ testSomme 2 ms ok
Process finished with exit code 0

```

Le test avec maven

```

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 5.128 s
[INFO] Finished at: 2022-02-21T18:08:15Z
[INFO] -----

Process finished with exit code 0

```