

## Département Mathématiques et Informatique

Filière :

**Ingénierie Informatique, Big Data et Cloud Computing  
(II-BDCC)**

# Activité Pratique N°1

**Inversion de contrôle et Injection des  
dépendances**

**Par : BOUZINE Ahmed houssam**

# Partie 1 :

## 1. Créer l'interface IDao avec une méthode getDate

```
package com.ahmedhoussambouazine.dao;

import java.util.Date;

12 usages 3 implementations new *
public interface IDao {
    1 usage 3 implementations new *
    public Date getDate();
}
```

## 2. Créer une implémentation de cette interface

```
package com.ahmedhoussambouazine.dao;

import java.util.Date;

4 usages new *
public class DaoImpl implements IDao{
    1 usage new *
    @Override
    public Date getDate() {
        System.out.println("Version DB");
        return new Date();
    }
}
```

## 3. Créer l'interface IMetier avec une méthode calcul

```
package com.ahmedhoussambouazine.metier;

import java.util.Date;

7 usages 1 implementation new *
public interface IMetier {
    3 usages 1 implementation new *
    Date calcul();
}
```

#### 4. Créer une implémentation de cette interface en utilisant le couplage faible

```
package com.ahmedhoussambouzone.metier;

import com.ahmedhoussambouzone.dao.IDao;
import java.util.Date;

5 usages new *
public class MetierImpl implements IMetier {
    // couplage faible
    2 usages
    private IDao dao;
    3 usages new *
    @Override
    public Date calcul() {

        return dao.getDate();
    }

    // injecter dans la variable dao un objet d'une classe qui implemente l'interface IDao
    new *
    > public void setDao(IDao dao) { this.dao = dao; }
}
```

#### 5. Faire l'injection des dépendances :

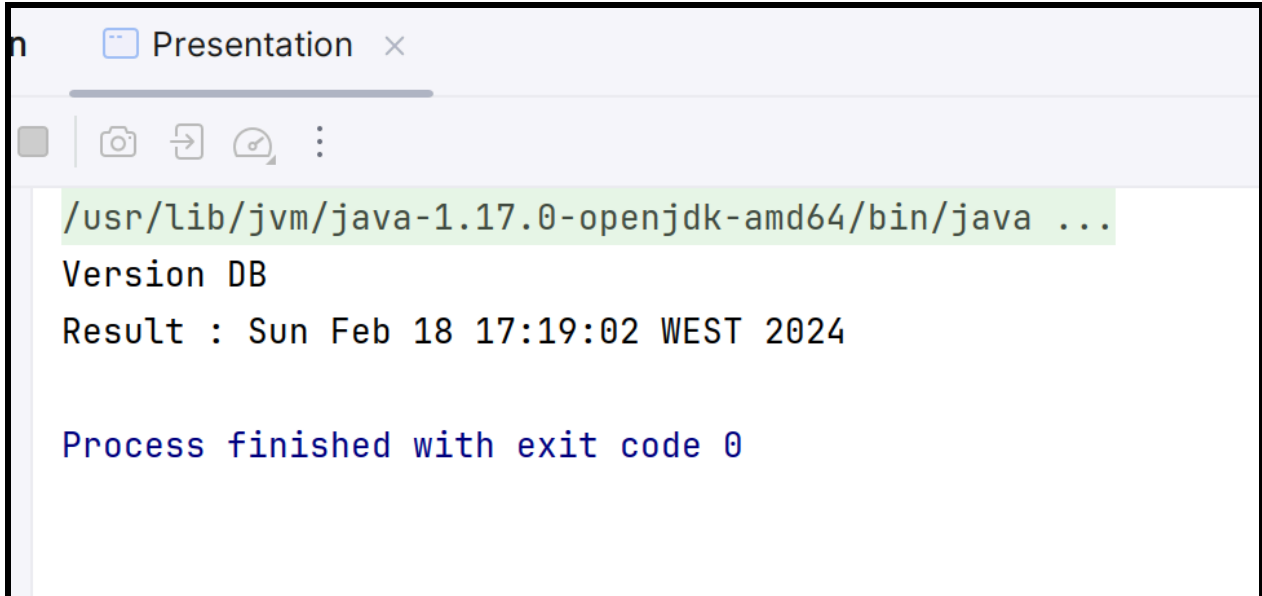
##### a. Par instanciation statique

```
package com.ahmedhoussambouzone.presentation;

import com.ahmedhoussambouzone.dao.DaoImpl;
import com.ahmedhoussambouzone.metier.MetierImpl;

new *
public class Presentation {
    new *
    public static void main(String[] args) {
        /*
        * Injection de Dependances par instanciation static => new => couplage fort
        * */
        DaoImpl dao = new DaoImpl();
        MetierImpl metier = new MetierImpl();
        metier.setDao(dao);
        System.out.println("Result : " + metier.calcul());
    }
}
```

## Exécution :

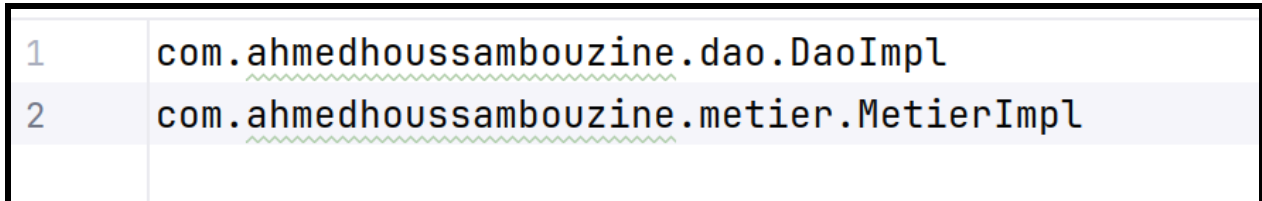


```
n  Presentation  x
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java ...
Version DB
Result : Sun Feb 18 17:19:02 WEST 2024

Process finished with exit code 0
```

### b. Par instanciation dynamique

Premièrement je vais créer un fichier **config.xml** contenant le path des classes d'implantation de l'interface.



```
1  com.ahmedhoussambouazine.dao.DaoImpl
2  com.ahmedhoussambouazine.metier.MetierImpl
```

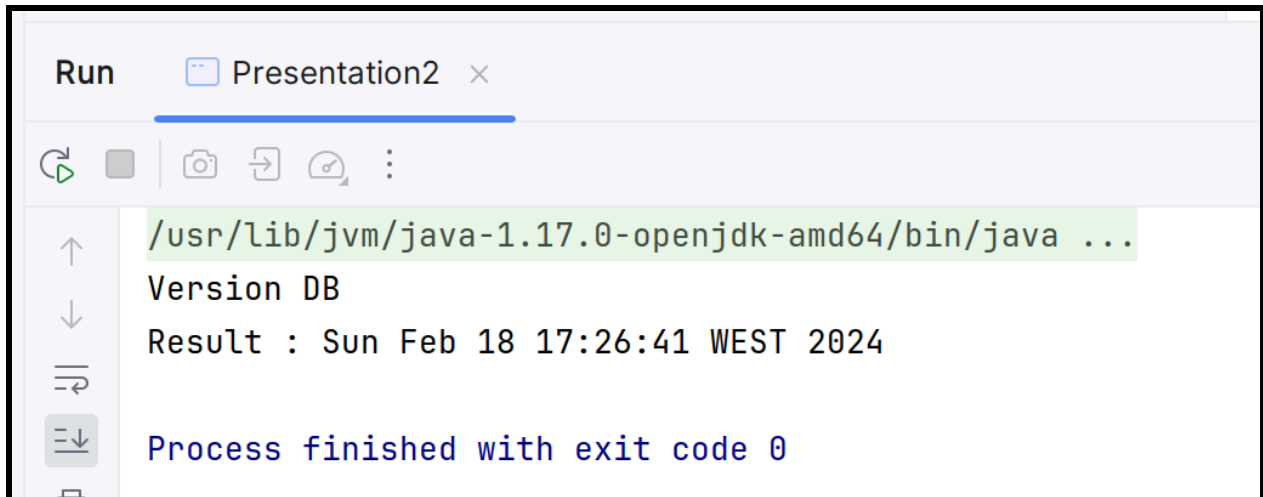
Puis je vais créer une nouvelle classe pour implémenter l'injection de dépendances d'une manière dynamique.

```

9  import java.util.Scanner;
10
11  new *
12  public class Presentation2 {
13      new *
14      public static void main(String[] args) {
15          try {
16              Scanner scanner = new Scanner(new File( pathname: "config.txt"));
17              String daoClassName = scanner.nextLine();
18              Class cDao = Class.forName(daoClassName);
19              // cDao.newInstance() is deprecated since Java 9
20              IDao dao = (IDao) cDao.getDeclaredConstructor().newInstance();
21              // System.out.println(dao.getDate());
22              String metierClassName = scanner.nextLine();
23              Class cMetier = Class.forName(metierClassName);
24              IMetier metier = (IMetier) cMetier.getDeclaredConstructor().newInstance();
25              // metier.setDao(dao); we don't do that we should do it with dynamic injection
26              Method method = cMetier.getMethod( name: "setDao", IDao.class);
27              method.invoke(metier, dao);
28              System.out.println("Result : " +metier.calcul());
29          } catch (Exception e) {
30              throw new RuntimeException(e);
31          }
32      }
33  }

```

### Exécution :



```

Run  Presentation2 x
[Run icon] [Stop icon] [Screenshot icon] [Copy icon] [Paste icon] [More options icon]
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java ...
Version DB
Result : Sun Feb 18 17:26:41 WEST 2024
Process finished with exit code 0

```

### c. En utilisant le Framework Spring

Dans cette étape je vais créer un nouveau projet sur la base de maven puis je vais ajouter les dépendances suivantes : spring.core, spring.bean , spring.context dans le fichier **pom.xml** comme le suivant :

```

5     <groupId>com.ahmedhoussambouzzine</groupId>
6     <artifactId>enset-ioc-spring2</artifactId>
7     <version>1.0-SNAPSHOT</version>
8     <packaging>jar</packaging>
9
10    <name>enset-ioc-spring2</name>
11    <url>http://maven.apache.org</url>
12
13    <properties>
14        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15    </properties>
16
17    <dependencies>
18
19        <dependency>
20            <groupId>org.springframework</groupId>
21            <artifactId>spring-core</artifactId>
22            <version>6.1.4</version>
23        </dependency>
24        <dependency>
25            <groupId>org.springframework</groupId>
26            <artifactId>spring-context</artifactId>
27            <version>6.1.4</version>
28        </dependency>
29        <dependency>
30            <groupId>org.springframework</groupId>
31            <artifactId>spring-beans</artifactId>
32            <version>6.1.4</version>
33        </dependency>
34
35        <dependency>
36            <groupId>junit</groupId>
37            <artifactId>iunit</artifactId>

```

### - Version XML

Pour cette version il faut créer un 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/schem

    <bean id="dao" class="com.ahmedhoussambouzone.extension.DaoImpl2"></bean>
    <bean id="metier" class="com.ahmedhoussambouzone.metier.MetierImpl">
        <property name="dao" ref="dao"></property>
    </bean>
</beans>
```

Puis créer une classe de présentation pour l'exécution contenant le code suivant :

```
package com.ahmedhoussambouzone.presentation;

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

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

**Exécution :**

```
PresentationSpringXML x
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java ...
Version Capteurs
Result : Sun Feb 18 17:56:56 WEST 2024

Process finished with exit code 0
```

### - Version annotations

Dans cette version il faut ajouter dans l'implémentation des interfaces l'annotation **@Component**

```

@Component("metier")
public class MetierImpl implements IMetier {
    // couplage faible
    3 usages
    private IDao dao;
    new *
    public MetierImpl(IDao dao) {
        this.dao = dao;
    }
    4 usages new *
    @Override
    public Date calcul() {

        return dao.getDate();
    }
    // injecter dans la variable dao un objet d'une classe qui implemente l'interface IDao
    new *
    public void setDao(IDao dao) { this.dao = dao; }
}

```

La classe de test pour l'exécution :

```

1 package com.ahmedhoussambouzzine.presentation;
2
3 import com.ahmedhoussambouzzine.metier.IMetier;
4 import org.springframework.context.ApplicationContext;
5 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
6
7 new *
8 > public class PresentationSpringAnnotation {
9     new *
10    public static void main(String[] args) {
11        ApplicationContext context = new AnnotationConfigApplicationContext(...basePackages: "com.ahmedhoussambouzzine.dao", "com.ahmedhoussambouzzine.metier");
12        IMetier metier = (IMetier) context.getBean(IMetier.class);
13        System.out.println("Result : " + metier.calcul());
14    }
15 }

```