|  |
| --- |
| **Travaux Pratiques**  **FI BDCC S4 : Architecture distribuée J2EE** TP : Inversion de Contrôle |

# Introduction :

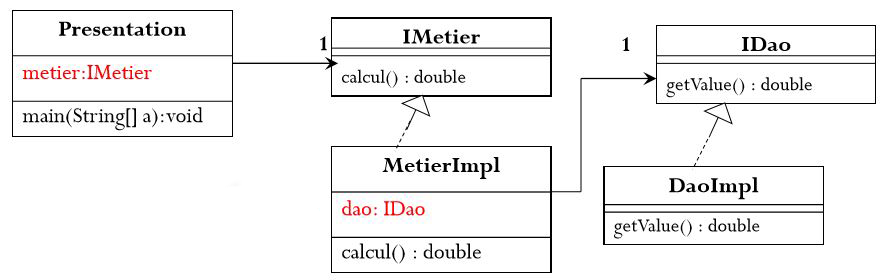
L’inversion de contrôle (inversion of control, IoC) est un patron d'architecture commun à tous les frameworks (ou cadre de développement et d'exécution). Il fonctionne selon le principe que le flot d'exécution d'un logiciel n'est plus sous le contrôle direct de l'application elle-même mais du framework ou de la couche logicielle sous-jacente.

L’inversion de contrôle est un terme générique. Selon le problème, il existe différentes formes, ou représentation d'IoC, le plus connu étant l'injection de dépendances (dependency injection) qui est un patron de conception permettant, en programmation orientée objet, de découpler les dépendances entre objets.

# Enoncé :

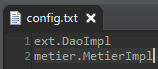
On sait le principe de l’inversion de contrôle et la forme le plus connue de ce dernier : l’injection des dépendances, d’abord on va voir un exemple qui va nous illustrer un peu ce principe, en utilisant le principe du couplage faible entre les classes.

# Architecture et conception :



# Code source :

* + Injection des dépendances sans aucun Framework :



|  |  |
| --- | --- |
| Public interface Idao {  public double getValue();  } | public class DaoImpl implements IDao {  public double getValue() {  return 5;  }  } |
| public interface IMetier {  public double calcul();  } | public class MetierImpl implements IMetier {  private IDao dao;  public double calcul() {  double nb=dao.getValue();  return 2\*nb;  }  public void setDao(IDao dao) {  this.dao = dao;  }  } |
| /\*  \* Injection des dépendances par instanciation statique  \*/  DaoImpl dao=new DaoImpl();  MetierImpl metier=new MetierImpl();  metier.setDao(dao);  System.***out***.println(metier.calcul()); | /\*  \* Injection des dépendances par instanciation dynamique  \*/  Scanner scanner = new Scanner(new File("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("setDao",IDao.class);  meth.invoke(metier,dao);  System.***out***.println(metier.calcul()); |

* + Injection des dépendances avec Spring :

|  |  |
| --- | --- |
| <?**xml** version=*"1.0"* encoding=*"UTF-8"*?>  <!**DOCTYPE** beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN" "http://www.springframework.org/dtd/spring-beans-2.0.dtd" >  <**beans**>  <**bean** id=*"d"* class=*"dao.DaoImpl"*></**bean**>  <**bean** id=*"metier"* class=*"metier.MetierImpl"*>  <**property** name=*"dao"* ref=*"d"*></**property**>  </**bean**>  </**beans**> | /\*Sans les annotations\*/  ApplicationContext context=new ClassPathXmlApplicationContext("spring-ioc.xml");  IMetier metier=context.getBean("metier");  System.***out***.println(metier.calcul());    /\*Avec les annotations\*/  ApplicationContext ctx=new AnnotationConfigApplicationContext("dao","metier");  IMetier metier=ctx.getBean(IMetier.class);  System.***out***.println(metier.calcul()); |

Les annotations que nous avons utilisées est les suivantes :

* ***@Component*** : est utilisé pour désigner une classe en tant que composant. Cela signifie que le framework Spring détectera automatiquement ces classes pour l'injection de dépendances lorsque la configuration basée sur les annotations et l'analyse du chemin de classe sont utilisées.
* ***@Autowired***: injecte automatiquement les beans dépendants dans les références associées d'une classe.

# Conclusion :

On peut déduire que l’utilisation d’un framework qui s’occupe du code technique (l’inversion du contrôle) est une stratégie performante qui va nous permettre de gagner beaucoup du temps, et l’utilisation d’un framework développé par une grande communié est plus efficace qu’utiliser votre code technique qui peut avoir plusieurs bugs.

Table de matière

[**Travaux Pratiques** 1](#_Toc33614380)

[**FI BDCC S4: Architecture distribuée J2EE** 1](#_Toc33614381)

[TP : Inversion de Contrôle 1](#_Toc33614382)

[ Introduction : 1](#_Toc33614383)

[ Enoncé : 1](#_Toc33614384)

[ Architecture et conception : 1](#_Toc33614385)

[ Code source : 1](#_Toc33614386)

[ Conclusion : 2](#_Toc33614387)