Compte Rendu :
Inversion de contrôle & Injection de dépendance

DE

2 ÈME ANNÉE CYCLE D'INGÉNIEUR

GENIE INFORMATIQUE

AKRAM LKIHAL

Encadrée par :

Mme. BADRI TIJANE

Établissement :

EMSI

(École Marocaine des Sciences de L'Ingénieur)

Année universitaire 2023-2024

Introduction :

Pour créer un projet informatique dans les normes il faut respecter les exigences fonctionnelles (métier) càd les fonctionnalités de l'application , techniques (gestion de transactions, sécurité, journalisation, gestion des exceptions et accès au base de données…) et financières (le cout du logiciel doit respecter les contraintes budgétaires).

L'une des exigences techniques c'est la performance càd d'éviter le problème de montée en charge qui signifie c'est quand le nombre d'utilisateurs simultanés qui accèdent à l'application est énorme, le serveur effectue beaucoup de traitement en même temps, il gère beaucoup de requêtes ce qui signifie cpu utilisé avec un taux trop élevé. Alors pour régler ce problème, il faut aller vers la scalabilité verticale ou horizontale.

- Scalabilité verticale : c'est ajouter capacité disque, mémoire dans le serveur.
- Scalabilité horizontale : c'est démarrer l'application dans plusieurs machines serveurs.

L'application doit être facile à maintenir càd elle doit être fermée à la modification et ouverte à l'extension pour évoluer dans le temps.

Récupérer une application pour appliquer une modification au niveau du code ça ne se fait pas au marché. La question qui se pose comment implémenter ça dans un projet ?

Pour répondre à cette question il faut savoir la différence entre le couplage faible et le couplage fort.

- Le couplage fort : c'est quand on a deux classes liées directement, quand une classe est liée à une autre classe par une association forte càd quand une classe dépend d'une autre classe directement.
- Le couplage faible : c'est quand une classe dépend que des interfaces, puisque l'interface peut être implémentée par plusieurs classes, on a le droit d'ajouter plusieurs extensions.

D'un côté je peux bénéficier de plusieurs classes (fonctionnalités) et de ce côté je ne peux pas toucher le code, ceci résume ne pas toucher le code c'est avoir la flexibilité au niveau de l'application.

L'inversion de contrôle :

L'inversion de contrôle : c'est séparer l'aspect technique et l'aspect fonctionnel (métier) . Elle permet au développeur de se concentrer sur l'aspect métier càd le code métier de l'application alors que c'est le Framework qui va s'occuper du code technique, l'inversion de contrôle est basé sur la programmation orienté aspect AOP .

Pour s'assurer que le principe de l'inversion de contrôle fonctionne concrètement sur un projet, on s'occupe du code métier et le Framework s'occupe du code technique par l'injection de dépendances.

Un Framework ou nous demandons de s'occuper du code technique il a besoin de librairies, un Framework possède plusieurs modules (un ensemble de fonctionnalités), à chaque fois ou on attaque un module on a besoin d'injecter des dépendances nécessaires au code technique de ce module.

Sans utiliser Spring :

- Si je suis dans un projet simple java sans le Framework Spring on fait l'injection de dépendances de manière basique à travers l'instanciation statique càd l'utilisation des setters, le constructeur, l'interface.
- L'instanciation dynamique à travers un fichier de configuration, c'est la manière assez complexe, généralement utilisé par les développeurs des Framework, on spécifie et on déclare les classes qu'on veut utiliser dans ce fichier de configuration.

Avec utilisation de Spring :

- A travers le fichier XML on communique avec le Framework à travers le code, il faut que je lui spécifie quelle partie technique que je souhaite avoir.
- Un moyen beaucoup plus facile c'est les annotations, il suffit de mettre une annotation dans une classe indirectement on a parlé avec Spring et on lui spécifie

ce qu'il doit faire. Spring scanne toutes les classes pour tirer les annotations. Il tire l'annotation et il sait le travail qu'il doit faire.

A chaque fois qu'on modifie le fichier pom.xml Maven reload project ça permet de dire à maven je viens de modifier le fichier pom.xml il recharge les dépendances maven, il va relire le fichier pom.xml et après il va ramener les dépendances.
Pour faire l'injection de dépendances on a besoin de trois modules Spring Core, Spring Context et Spring Beans.

A chaque fois qu'on a besoin d'ajouter une dépendance (une librairie) dans notre projet il suffit de la déclarer dans pom.xml, dans ce fichier on trouve les informations que nous venons de préciser à savoir le GroupId (Il est mieux d'utiliser nom de domaine inversé pour garantir l'unicité), ArtifactId (nom du module) et la version .

Dans le TP on a respecté le modèle en couches, on va trouver la classe DaoImpl la classe MetierImpl et la classe Présentation. Pourquoi on a opté à utiliser ce modèle ?

Quand je souhaite mettre des choses qui sont en relation avec la base de données je vais attaquer la classe dao, quand je vais mettre des choses qui sont associés au service et au traitement je vais attaquer la classe metier et des choses qui sont liés directement à la partie frontend je vais les mettre dans la classe presentation.

Maven c'est La plateforme d'utilisation de gestion de projet qui permet de faciliter et d'automatiser la gestion de projet . Il permet de délivrer trois types d'application Desktop, web et mobile. Maven a pris le succès en web et Gradle en mobile.

Maven c'est un projet java qui contient un fichier pom.xml.

A chaque fois qu'on modifie le fichier pom.xml Maven reload project ça permet de dire à maven je viens de modifier le fichier pom.xml il recharge les dépendances maven, il va relire le fichier pom.xml et après il va ramener les dépendances.
Pour faire l'injection de dépendances on a besoin de trois modules Spring Core, Spring Context et Spring Beans.

A chaque fois qu'on a besoin d'ajouter une dépendance (une librairie) dans notre projet il suffit de la déclarer dans pom.xml, dans ce fichier on trouve les informations que nous venons de préciser à savoir le GroupId (Il est mieux d'utiliser nom de domaine inversé pour garantir l'unicité), ArtifactId (nom du module) et la version ;

Pour travailler avec Spring, pour faire l'injection de dépendances on a utilisé deux versions : version xml et version annotation.

Version xml :

On demande à Spring au démarrage de nous créer un objet de la classe DaoImpl et de lui donner comme nom Dao ensuite on lui demande de créer un bean dont id c'est métier de la classe MetierImpl et quand il nous crée un objet de MetierImpl, il va appeler setter et il va faire l'injection de dépendances.
Si on veut faire l'injection via le setter on utilise property et on lui affecte comme valeur l'objet dao.

Pour faire l'injection de dépendances il va falloir démarrer Spring, on va utiliser un objet new ClassPathXmlApplicationContext pour lui demander d'aller lire le fichier .xml
Quand Spring va démarrer il va lire le fichier .xml, quand il va le lire il va trouver bean puis il va créer un objet DaoImpl, il va le placer dans une collection de type map de beans et il va lui donner comme clé c'est d par la suite il trouve l'objet MetierImpl, il va créer l'objet MetierImpl et il va lui donner le nom metier et après avec property il fait l'injection de dépendances il fait appel à setdao il lui transmet comme l'objet d.
Une fois on démarre spring il faut lui dire donne-moi un spring de type IMetier et après on fait appel à la méthode calcul.

Version annotation :

@Component dit à Spring au démarrage à chaque fois qu'il trouve une classe qui commence par l'annotation Component il va l'instancier, il va lui donner comme nom Dao la même chose pour métier, après pour faire l'injection de dépendances il faut utiliser l'annotation @Autowired demande à Spring au moment il va instancier la classe MetierImpl il doit chercher un objet de type IDao s'il le trouve, il doit l'injecter dans cette variable dao. Après il faut démarrer l'application avec un objet de ApplicationContext.
On va lui donner de scanner les classe qu'ils se trouvent dans le package dao et le package metier.