

# Notions fondamentales (sans Spring Boot)

## Objectifs

- ¥ Comprendre l'inversion de contr<sup>TM</sup>le (IoC)
- ¥ Comprendre l'injection de d<sup>TM</sup>pendances (DI)
- ¥ D<sup>TM</sup>finir des beans Spring en XML
- ¥ D<sup>TM</sup>marrer un contexte Spring
- ¥ Organiser une application simple en couches

## Inversion de contr<sup>TM</sup>le (IoC)

- ¥ Principe : l'application ne cr<sup>TM</sup>te plus directement ses objets.
- ¥ C<sup>TM</sup>est le contexte Spring qui instancie et connecte les composants ^ notre place.
- ¥ Avantage : d<sup>TM</sup>couplage du code, testabilit<sup>TM</sup>, maintenabilit<sup>TM</sup>.

## Inversion de contr<sup>TM</sup>le (IoC)

*Exemple (sans Spring)*

```
Service service = new Service(new Repository());
```

*Avec Spring (IoC)*

```
Service service = context.getBean("service", Service.class);
```

## Injection de d<sup>TM</sup>pendances (DI)

- ¥ Spring fournit automatiquement les d<sup>TM</sup>pendances n<sup>TM</sup>cessaires ^ un bean.
- ¥ Plusieurs types d'injection : par constructeur, par setter, par champ (champ = d<sup>TM</sup>conseill<sup>TM</sup> ici).
- ¥ Dans ce TP : injection via setter dans le fichier XML.

## Le fichier applicationContext.xml

- ¥ C<sup>TM</sup>est le c<sup>TM</sup>ur de la configuration dans ce TP.
- ¥ On y d<sup>TM</sup>clare les objets (beans) et leurs d<sup>TM</sup>pendances.

```
<bean id="..." class="...">  
    <property name="..." ref="..." />  
</bean>
```

Montrez qu'on peut gérer les dépendances sans annotation, tout est déclaré dans ce fichier XML.

## Structure classique d'une application

- ¥ Entity : objets représentant les données manipulées
- ¥ Repository : gère la "persistance" ou l'accès aux données
- ¥ Service : logique métier, utilise les repositories
- ¥ Main / Lanceur : initialise le contexte et lance l'application

## Démarrage de Spring

Chargement du contexte via la classe `ClassPathXmlApplicationContext` :

```
ApplicationContext context =  
    new ClassPathXmlApplicationContext("applicationContext.xml");
```

Récupération d'un bean Spring :

```
Service service = context.getBean("service", Service.class);
```

## Bonnes pratiques

- ¥ Garder une responsabilité claire pour chaque classe.
- ¥ Éviter que les services connaissent les détails d'implémentation des repositories.
- ¥ Penser en termes d'interfaces et d'abstraction.

## Ce que vous devez savoir faire

- ¥ Créer une classe simple avec des getters/setters.
- ¥ Créer un repository avec une méthode "save".
- ¥ Créer un service qui dépend d'un repository.
- ¥ Définir les beans dans un fichier XML.
- ¥ Démarrer un contexte et appeler un service.

# Attention

- ¥ Pas d'annotations dans ce TP
- ¥ Pas de Spring Boot
- ¥ Pas de base de données : repository simplifié
- ¥ Uniquement de la configuration XML