

Java EE

Java Enterprise Edition



Introduction aux CDI (Contexts and Dependency Injection)

Mohamed SALLAMI



Présentation

- Spécification CDI incluse dans Java EE 6
- Fournit des services pour améliorer les développements :
 - Mécanisme d'injection sûr (erreurs de typage détectées à la compilation)
 - Gestion du cycle de vie de beans
 - Intégration avec les autres parties de Java EE (en particulier EJB et JSF)
 - ...

Seules les bases seront abordées dans ce cours

Le fichier WEB-INF/beans.xml (même vide) est indispensable pour que CDI soit activé

DI: Définition

- L'injection de dépendance (DI pour Dependency Injection) : fournir une dépendance, une ressource, à un composant logiciel sans qu'il ne soit obligé de la créer ou de la chercher.
- Le composant indique qu'il a besoin d'une instance qui implémente une certaine interface et le contexte d'exécution lui fournit une telle instance.

DI: Exemple

- Un servlet veut utiliser un EJB pour accéder à une base de données
- Il n'a pas besoin de créer une instance de la classe de l'EJB ou de demander à un registre de lui fournir une instance
- Il suffit au développeur de dire que le servlet a besoin de cet EJB et le servlet le recevra quand le servlet sera créé
- Il peut le dire par une annotation ou dans un fichier de configuration

DI: Exemple de code

```
@WebServlet(name = "ModifSalaire", urlPatterns = {"/ModifSalaire"})
public class ModifSalaire extends HttpServlet {
    @EJB
    EmployeFacade employeFacade;
    ...
    Employe employe = employeFacade.find(id);
    ...
}
```

DI: Contexte d'exécution

Pas de miracle, DI nécessite un contexte d'exécution qui injecte les dépendances pour les composants qu'il gère

DI: Avantages

- Code plus simple
- Couplage plus souple (le type de la classe injectée n'est pas écrit en dur dans le code : un sous-type du type interface déclaré)
- Partage de données simplifié : une dépendance est automatiquement partagée par tous les clients qui s'exécutent dans la même portée (par exemple dans la session d'un utilisateur)

DI: L'injection dans Java EE

- Elle existe depuis Java EE 5
- **@Resource** par exemple pour injecter une source de données
- **@EJB** pour injecter un EJB
- CDI, introduit avec Java EE 6, étend l'utilisation de DI et lui donne bien plus de possibilités et de souplesse

CDI: Bean géré

- Concept central de CDI
- Un bean est dit géré quand son cycle de vie est géré par un container CDI

CDI: Gestion d'un bean géré par le container

- Le container
 - crée et supprime l'instance
 - injecte le bean déjà créé dans une autre instance
 - appelle d'éventuelles méthodes de callback aux diverses phases du cycle de vie
 - intercepte des méthodes du bean

CDI: Attributs d'un bean géré

- Un ensemble non vide de types de bean
- Un ensemble non vide de qualificateurs (qualifier)
- Une portée (scope)
- Optionnellement un nom EL (voir JSF)
- Une implémentation, la classe du bean

CDI: Type de bean

- Les types d'un bean sont les types visibles par les clients du bean (ceux qui vont l'injecter)

- Exemple :

```
public class A extends B implements C { ... }
```

Un bean de la classe A a 4 types : A, B, C et Object
(un bean a toujours le type Object)

Aucune déclaration/annotation nécessaire

@Inject

- Le type injecté doit être une interface Java
- Exemple:
 @Inject
 A a;
 - fournit une instance d'une classe qui implémente l'interface A
 - Si l'instance de A existait déjà dans la portée du bean, l'instance existante est fournie au code
 - Sinon, une nouvelle instance est créée par le container et fournie au code

@Named

- Cette annotation permet de donner un nom «EL» (Expression Language) de JSF au bean annoté
- Les pages JSF pourront ainsi utiliser une référence au bean

Qu'est ce qui peut être injecté?

- Presque toutes les classes Java (pas une classe interne non static) ; la classe doit avoir un constructeur sans paramètre (ou annoté par @Inject)

Conversation

- Portée utilisée quand on veut partager des informations pendant plus longtemps qu'une seule requête mais moins longtemps qu'une session entière
- La conversation convient bien aux traitements de type «wizard» où des informations concernant une action sont recueillies en présentant plusieurs pages à l'utilisateur

Exemple

```
@Named @ConversationScoped
public class ClientBean implements Serializable {
    @Inject
    private Conversation conversation;
    private Client client;
    public String voirDetails(Client client) {
        this.client = client;
        conversation.begin();
        return "DetailsClient?id=" +
            client.getClientId() + "&faces-redirect=true";
    }
    public String modifier() {
        client = clientManager.update(client);
        conversation.end();
        return "ListeClients?faces-redirect=true";
    }
}
```

Qualificateur

- Cas d'utilisation : le type d'une injection est une interface et plusieurs classes correspondent à ce type
- Un qualificateur est une annotation qui permet de différencier les sous-types pour préciser le type de l'instance injectée
- Très simple d'ajouter un nouveau qualificateur

Exemple

@Qualifier

@Retention(RUNTIME)

@Target({METHOD, FIELD, TYPE})

public @interface DixChiffres { }

Exemple

@HuitChiffres

```
public class Generateur  
implements GenerateurNombre {...}
```

@DixChiffres

```
public class Generateur  
implements GenerateurNombre {...}
```

Injection d'un générateur de nombres :

@Inject @HuitChiffres

```
private GenerateurNombre generateur;  
ou
```

@Inject @DixChiffres

```
private GenerateurNombre generateur;
```

Il y a encore beaucoup de choses à dire sur CDI mais ce cours n'est qu'une introduction et ce qui a été exposé suffit très souvent