Sommaire

TP 0- installation de JSF

TP 0- installation de JSF

0.a) Introduction au projet Bankonet

Objectif :
Réaliser un mini-projet mettant en œuvre les différents concepts du framework JSF.

Présentation du mini-projet :
Nous réaliserons une application de consultation et de virement de comptes bancaires en-ligne.

Point de départ :
Nous utiliserons un ensemble de classes bancaires « métier » :

- Client
- Compte
  - CompteCourant
  - CompteEpargne
- Virement

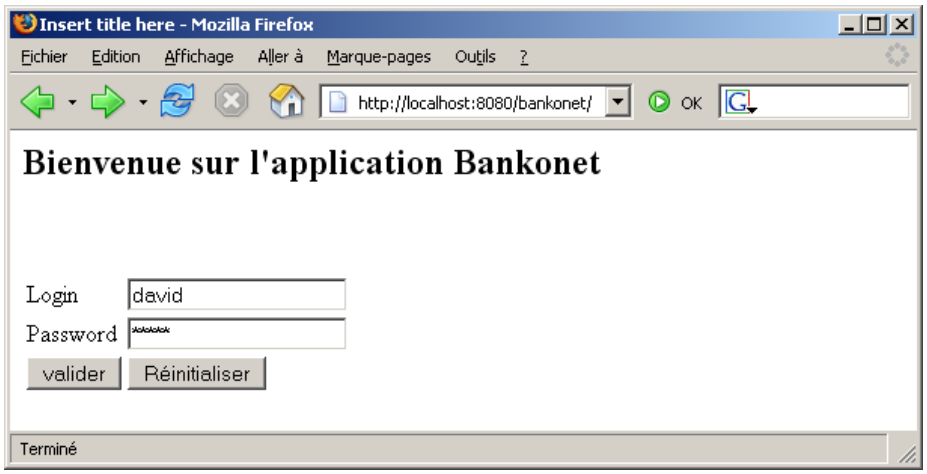
Les informations client et comptes sont stockées en base de données.
Les paramètres de connexion des clients sont :

Login	Password
david	David
sandie	Sandie
fabrice	Fabrice

TP 1- Implémentation de la page de login

Objectif :

Le but de ce TP est d'établir un premier contact avec JSF via la mise en place d'une simple page de login (suivre l'exemple du cours)



1.a) Création du bean `ClientBean`

Note : tous les packages devront être placés dans le dossier `WEB-INF/src`.

⇨ Créer la classe `com.bankonet.bean.ClientBean`

- Ce `bean` doit contenir deux propriétés : `login` et `password`. (créer des méthodes getter/setter si nécessaire).

⇨ Déclarer `clientBean` dans le `faces-config.xml`

1.b) Créer la page `login.jsp` selon l'impression d'écran ci-dessus.


Tester l'affichage de la page de login.

TP 2- Navigation et contextes

Objectifs :

Manipuler les concepts de navigation.
Utiliser les `backing beans`.

2.a) Méthode d'authentification

⇨ dans `ClientBean`, créer la méthode `traiterLogin()`. Cette méthode transmet l'authentification aux classes du package `com.bankonet.service` comme suit :

```
Client cl = new BanqueService().findClient(login, password);
```

⇨ Après récupération du client, le stocker comme attribut de session.

```
FacesContext.getCurrentInstance().getExternalContext().getSessionMap
().put("client", cl);
```

Note : cette syntaxe sera détaillée par la suite.

⇨ Ecrire l'appel à cette méthode dans la page `login.jsp`

2.b) pagePrincipale.jsp

⇨ Créer une page `pagePrincipale.jsp`. Cette page affichera un message de bienvenue à l'utilisateur si son login et son password sont exacts.

2.c) Déclarer les règles de navigation dans le `faces-config.xml`

Note : en cas d'erreur d'authentification, l'utilisateur sera redirigé vers la page de login sans message d'erreur. Un message d'erreur sera rajouté par la suite.

 Tester le login.

2.d) Page de déconnexion

⇨ Créer la page `fin.jsp` qui invalide la session de l'utilisateur en cours. Afficher le message « Merci d'avoir utilisé Bankonet ».

 *Pour invalider une session : session.invalidate() ;*

⇨ Dans `pagePrincipale.jsp`, positionner un lien de déconnexion.

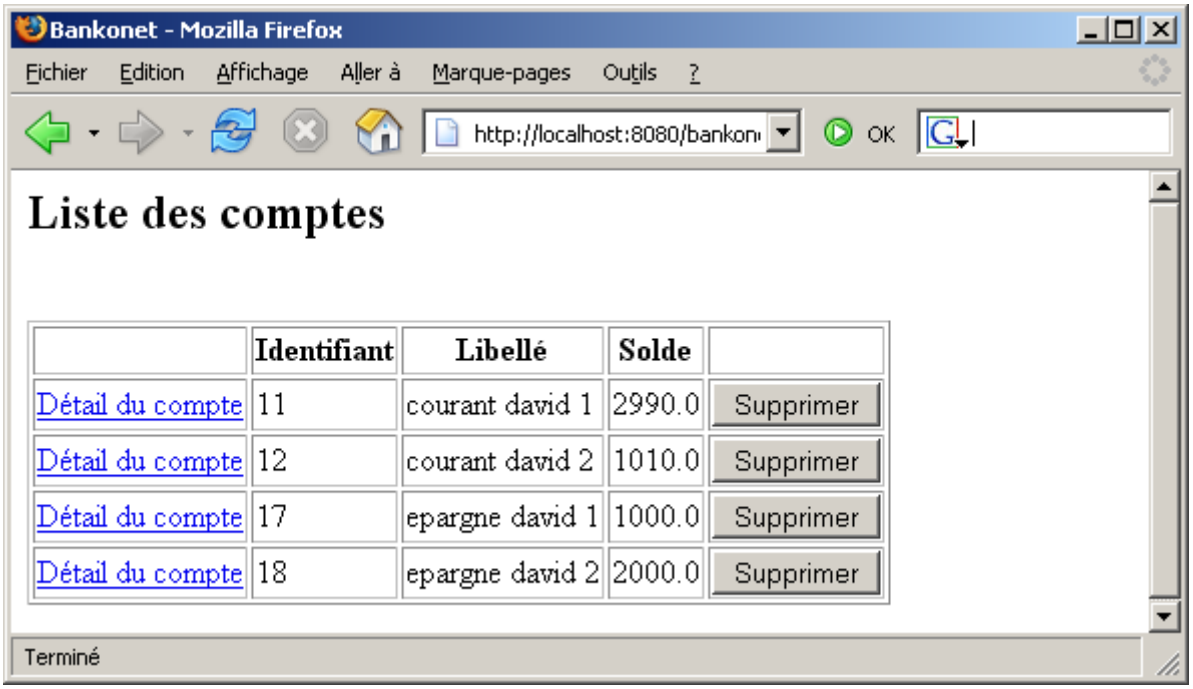
⇨ Impacter le fichier `faces-config.xml`.

 Tester la déconnexion.

TP 3- Une étude complète

Objectifs :

Manipuler différents composants UI, naviguer entre les écrans et introduction à la notion de modèle de données



3.a) page `compteListe.jsp`

⇨ Créer la classe `com.bankonet.bean.CompteBean`.

- Créer l'attribut `compteListe` et les méthodes get/set correspondantes.
- Créer la méthode `initListe()`. Dans cette méthode, récupérer la liste des comptes à partir du client stocké dans la session. Affecter cette liste dans l'attribut `compteListe` du bean `CompteBean`.
- Déclarer le bean `compteBean` dans le fichier `faces-config.xml`.

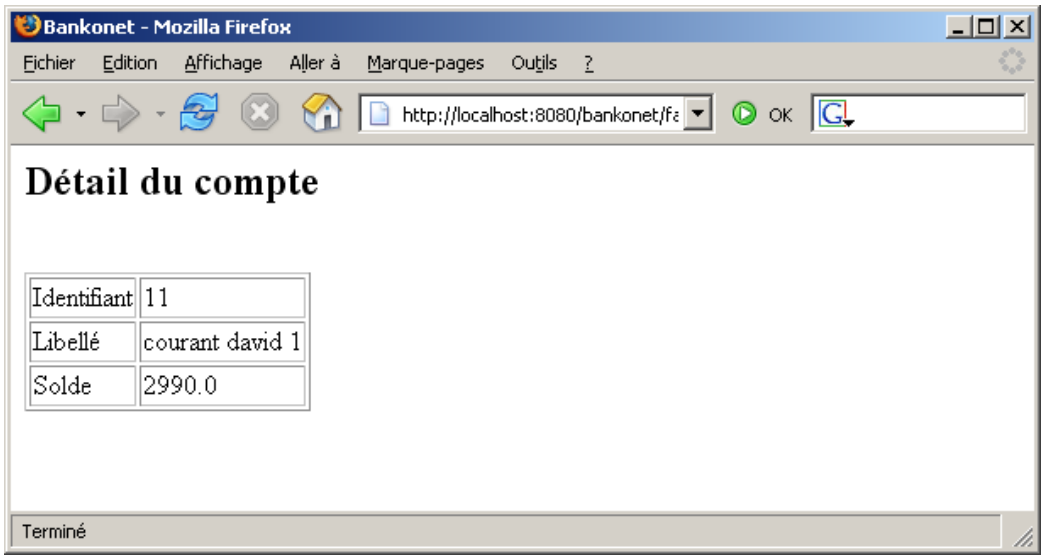
⇨ Concevoir la page `compteListe.jsp` en utilisant le tag `h:dataTable`

- Les informations du tableau doivent être récupérées à partir du `compteBean`.
- Les liens et boutons doivent être inactifs (ils seront implémentés par la suite).

⇨ Ajouter un lien vers la liste des comptes dans `pagePrincipale.jsp`. En préalable de l'affichage de la liste des comptes, la méthode `compteBean.initListe` sera invoquée.


Tester l'affichage de la liste des comptes.

3.b) Détail du compte : sélection d'un élément dans une liste



⇨ Le bean `CompteBean` a été créé dans la question précédente. Y ajouter la propriété `detailCompte` (type `Compte`) représentant le compte sélectionné pour l'affichage du détail

⇨ Egalement dans `CompteBean`, créer la méthode `initDetail()`.

- Cette méthode récupère le compte sélectionné et l'affecte à la propriété `CompteBean.detailCompte`.

⇨ Compléter les liens de détail de compte `compteListe.jsp`, ainsi que le fichier `faces-config.xml`.

⇨ Créer la page `compteDetail.jsp` d'après l'impression d'écran ci-dessus.

 Tester l'affichage du détail de compte.

3.c) Suppression d'un compte.

⇨ Dans la classe `CompteBean`, ajouter la méthode `traiterSuppression()` qui sera invoquée lors de la suppression d'un compte.

- Dans cette méthode, après avoir récupéré le compte à supprimer, faire un appel à la couche service comme suit :

```
new BanqueService().deleteCompte(…);
```

(ne pas oublier de mettre à jour l'attribut `CompteBean.compteListe`).

⇨ Compléter les boutons de suppression dans `compteListe.jsp`, ainsi que le fichier `faces-config.xml`.

⇨ Après suppression, la navigation revient vers `compteListe.jsp`.

 Tester la suppression d'un compte.