Formation Java JEE - Marseille, 11/10/2019 - Achref El Mouelhi

TP : Développement d'une application Web avec JEE

Pré-requis

Concepts Java

- Classe et interface
- Collection
- Exception
- JDBC, Bean et DAO

Concepts (et composants) JEE

- Servlet
- JSP et JSTL
- Validation de formulaire
- Session et Filtre

Outils

Concepts Java

- SGBD : MySQL
- IDE : Eclipse

Script SQL pour créer la base de données

```
create database projet;

use projet;

create table personne(
id int primary key auto_increment,
nom varchar(30) not null unique,
prenom varchar(30) not null,
sexe varchar(1)
);

create table adresse(
id int primary key auto_increment,
rue varchar(30),
codePostal varchar(30),
ville varchar(30),
idPersonne int,
foreign key (idPersonne) references personne(id)
);
```

Objectif

Créer une application Web avec la plateforme JEE avec un accès restreint aux membres qui permet à un utilisateur connecté de gérer son compte, gérer ses adresses et gérer les affectations.

Démarche

1. Créer un **Dynamic Web Project** ou un **Maven Project** et préparer ce qu'il faut pour utiliser une base de données MySQL et la librairie JSTL.
2. Créer les deux beans `Personne` et `Adresse` en se basant sur le schéma relationnel décrit par le script SQL (Attention, la classe `Adresse` doit contenir un attribut `personne` de type `Personne`).
3. Créer une interface générique `Dao<T>` avec les cinq méthodes de base `findAll()`, `findById()`, `save()`, `remove()` et `update()`.
4. Créer deux classes `PersonneDao` et `AdresseDao` qui implémentent `Dao` et aussi les cinq méthodes abstraites `findAll()`, `findById()`, `save()`, `remove()` et `update()`.
5. Dans la classe `PersonneDao`, ajouter une méthode `findByNomAndPrenom(String nom, String prenom)` qui permet de retourner un objet de type `Personne` selon les `nom` et `prenom` passés en paramètre. S'il n'y a pas de correspondance, on retourne `null`.
6. Dans la classe `AdresseDao`, ajouter une méthode `findByIdPersonne(int id)` qui permet de retourner les adresses associées à la personne dont le numéro est passé en paramètre.
7. Dans la classe `AdresseDao`, ajouter une deuxième méthode `findByIdPersonneNotEqual(int id)` qui permet de retourner toutes les adresses non-associées à la personne dont le numéro est passé en paramètre.
8. Dans la classe `AdresseDao`, ajouter une troisième méthode `updatePersonne(Adresse adresse)` qui permet de modifier seulement la personne associée à cette adresse.
9. Dans `WebContent` (ou `webapp` pour un projet `Maven`), créer une vue `index.jsp` qui contient le formulaire d'authentification. L'utilisateur utilisera son `nom` et son `prénom` pour se connecter. S'ils sont enregistrés dans la base de données, l'utilisateur sera ajouté à la session et accédera à l'application, sinon un message d'erreur sera affiché tout en gardant les valeurs saisies dans le formulaire. L'utilisateur a toujours la possibilité de cliquer sur un lien pour aller sur une deuxième vue `inscription.jsp` pour créer un compte.
10. Dans `WEB-INF`, créer une vue `inscription.jsp` qui contient deux champs texte pour `nom` et `prénom` et une liste déroulante pour le `sexe`. En cliquant sur le bouton `Créer un compte`, un compte sera créé et l'utilisateur sera redirigé vers la page de connexion et il pourra désormais se connecter.
11. Dans `WEB-INF`, créer une vue `gestionCompte.jsp` qui permet à l'utilisateur de modifier les données de son compte (`nom`, `prenom` et `sexe` affiché comme dans `inscription.jsp`, les champs sont initialisées par les valeurs de la base de données) ou de le supprimer. Si l'utilisateur connecté décide de modifie ces données, il faut mettre à jour les données `session`. S'il décide de supprimer son compte, il faut vider la `session` et faire une redirection vers la page d'inscription.
12. Dans `WEB-INF`, créer une vue `gestionAffectation.jsp` composée de trois parties :
    - une première partie qui permet à l'utilisateur de créer une nouvelle adresse (en cliquant sur le bouton `Créer adresse`, l'adresse sera créée et associée à l'utilisateur connecté et la page sera actualisée)
    - une deuxième partie dans laquelle on affiche toutes les adresses associées à cet utilisateur, chacune avec un bouton `Dissocier`. En cliquant, la colonne `idPersonne` de la table `Adresse` de ce tuple sera mise à `null` et la page sera rechargée.

- une dernière partie dans laquelle on affiche toutes les adresses non-associées à cet utilisateur, chacune avec un bouton `Associer`. En cliquant, l'adresse sera associée à l'utilisateur connecté et la page sera rechargée.

13. Dans `WEB-INF`, créer une vue `gestionAdresse.jsp` qui affiche les adresses de l'utilisateur connecté : une adresse par ligne. Devant chaque adresse, on a deux boutons : un pour supprimer (en cliquant dessus, l'adresse sera supprimé et la page sera actualisée avec la nouvelle liste d'adresse) et un pour modifier, en cliquant dessus, les attributs de l'adresse ainsi que les valeurs seront affichés (dans des `input`) dans une vue `detailsAdresse.jsp`.
14. Dans toutes les vues, sauf `inscription.jsp` et `index.jsp`, on affiche
    - le titre de la page (par exemple, **Page d'affectation** dans `gestionAffectation.jsp`)
    - un message comme : `Bonjour (Monsieur ou Madame selon le sexe) + prénom`
    - un lien de déconnexion : en cliquant dessus, la session sera vidée et l'utilisateur sera redirigé vers la page de connexion
    - des liens vers les toutes les pages de gestion : `Gestion de compte`, `Gestion d'adresse` et `Gestion des affectations`
15. Toutes les saisies utilisateurs doivent être contrôlées
    - pour les personnes, les champs `nom` et `prenom` doivent contenir au moins deux caractères. De plus, la première lettre doit être en majuscule.
    - pour les adresses, les champs `rue` et `ville` doivent contenir au moins deux caractères. `codePostal` doit contenir exactement 5 caractères numériques.
16. Utiliser un filtre pour interdire un visiteur d'accéder aux pages de gestion sans s'authentifier.

Indices

Servlets

- `ConnexionServlet`
- `DeConnexionServlet`
- `GestionCompteServlet`
- `GestionAdresseServlet`
- `GestionAffectationServlet`
- `InscriptionServlet`
- `DetailsAdresseServlet`

JSP

- `index.jsp`
- `inscription.jsp`
- `gestionCompte.jsp`
- `gestionAdresse.jsp`
- `gestionAffectation.jsp`
- `detailsAdresse.jsp`