YE Fangyuan Lisa 202
MILLO Chelsey 202

Projet JavaEE : Médiathèque 2.0

Présentation du projet

Le projet que nous avons eu à réaliser est une application web destinée aux employés des médiathèques. Cette application ne sera accessible qu'aux bibliothécaires locaux disposant d'un compte comportant un identifiant ainsi qu'un mot de passe.

Ces derniers ne disposent que de trois actions :

- Ajout de documents
- Suppression de documents
- Recherche de documents

Pour accéder à la base de données, nous avons utilisé le compte system et le mot de passe 2F3F63447F. Pour lancer l'application, il faut soit changer le mot de passe de l'application qui est présent dans le Classe Connexion du package persistant, soit changer le mot de passe du compte system.

Structuration de code

L'architecture de l'application a été imposée afin de mettre en oeuvre un découplage strict entre 3 packages:

- "**médiatek2021**" fournie par le professeur et contient les interfaces *Utilisateur*, *Document* et *PersistentMediatek* (qui permet de découpler la "**persistantdata**" et le "**service**"), les classes *NewDocException*, *SuppressException*, et *Mediatek* (qui appelle les fonctions de la classe *MediatekData*).

- "**persistantdata**" contient la classe *MediatekData* qui impose l'interface *PersistentMediatek* et regroupe toutes les fonctions comme getUser, catalogue, getDocument, newDocument et suppressDoc (tous les fonctions vont connecter à la base de donnée et réaliser les requêtes et/ou retourne l' élément demandant); la classe *Connexion* assure la connection de la base de donnée; la classe *Bibliothécaire* implémente l'interface *Utilisateur* et les classes *CD*, *DVD* et *Livre* implémentent l'interface *Document*.

- Nous n'avions pas fait le package "**service**" (qui sert aux échanges http si on utilise les servlets) car nous avions opté pour une gestion JSP des services.

  Nous avions réalisé:
  La page index.jsp qui sert à vérifier les logins et passwords des bibliothécaires.
  La page bibliothe.jsp qui sert à réaliser les actions suivantes:
  - Supprimer les document à partir d'un numéro
  - Rechercher les documents à partir d'un numéro

- Affiche le catalogue selon le type choisi sur la page catalogue.jsp
- Ajouter le document sur la page AjoutDocument.jsp
La page OutLogin.jsp qui est appelé pour déconnecter

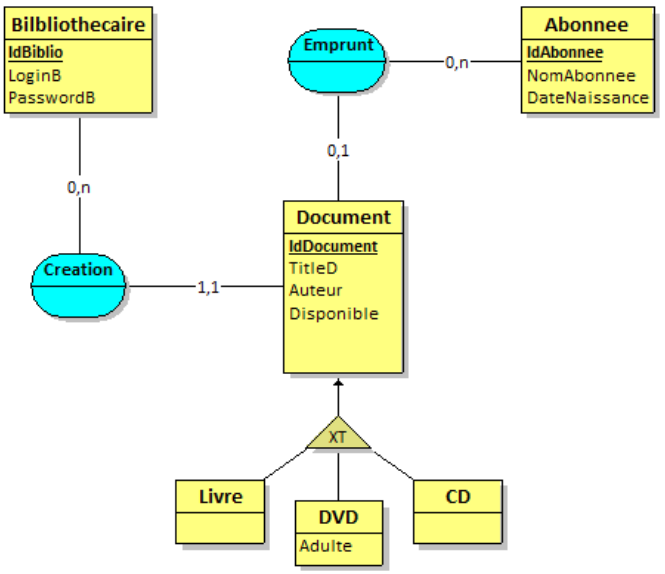
Dans chaque page jsp, nous avons importé le package mediatek2021 pour qu'on puisse utiliser les interfaces présentent dans ce package.

Transformation objet-relationnel

Cette application necessite d'une base de données c'est la raison pour laquelle nous avons dû implanter un modèle relationnel dans celle-ci.

Nous avons créé 3 tableaux:

- une table **Bibliothecaire** pour stocker les idBibiothecaire, le loginBibiothacaire et le passwordBibiothecaire
- une table **Abonne** pour stocker les idAbonne, les nomAbonne et le dateDeNaissanceAbonne
- un table **Document** qui contient les attributs IdDocument, Titre, Auteur, Type, Disponible (qui vérifie si le document est actuellement emprunter), IdBiblio (qui crée le document) et les attributs optionnel Adulte (qui vérifie si le DVD est pour adulte)



Nous avons employé la méthode ascendante, ce qui signifie que les attributs des sous-tables (DVD, CD et Livre) sont tous présentés dans le table Document. Cela évite qu'on crée des requêtes redondantes pour chaque table au moment de la réalisation des fonctions, mais cela peut créer des difficultés dans l'avenir si nous décidons d'ajouter un nouveau type de document.

Efficacité des requêtes d'accès à la base de données

Pour pouvoir le faire tourner nous nous sommes donc procuré un JDBC et avons intégré le script de base de données dans SQLPlus.

Avant de réaliser les requêtes de chaque fonction, nous devons accéder à la base de données et pour simplifier le code, nous avons réalisé un class Connexion qui assure la connexion et déconnexion avec la base de données.

Pour rendre l'exécution des requêtes plus efficace, nous avons utilisé le Prestatement au lieu de Statement, car l'interface Presetatement étend l'interface Statement, mais il contient l'instruction SQL déjà compilée. De plus, les instructions SQL des instances de PreparedStatement contiennent un ou plusieurs paramètres d'entrée, représenté par des points interrogatifs. Nous devrons spécifier ces paramètres avec le setXXX() avant l'exécution.

Variables sessions

Lorsque l'utilisateur réussit à se connecter, nous créons une session pour enregistrer les données de cet utilisateur. Après avoir récupéré les données de l'utilisateur connecté, nous avons enregistré ces données dans la session et défini la durée de la session qui est de 20 min.

Nous avons vérifié l'existence de session en début de chaque page jsp, pour s'assurer que la session existe pour chaque page web. Il nous permet d'utiliser des données contenant dans le session, par exemple, nous devrons récupérer l'id de l'utilisateur pour la création de nouveau document.

La fermeture de la session est réalisée dans le page outLogin.jsp, ce dernier est appelé lorsqu'on clique sur le bouton Déconnexion.

Concurrence

Dans cette application bibliothèque, les ressources critiques sont les données de Documents, car il peut y avoir plusieurs threads qui réalisent en même temps les différents types des actions sur ces données. Pour éviter qu' au moment où un utilisateur recherche un document, et qu'un autre utilisateur supprime ou ajoute un document, nous avons synchronized les fonctions newDocument et suppressDoc, afin de s'assurer qu'au moment de l'ajout de document ou suppression de document, les autres utilisateur ne peuvent pas avoir accès le base de données.

Conclusion

Nous avons réussi à faire fonctionner les fonctions qui servent à lister le catalogue des documents avec le type choisi; rechercher le document avec le numéro entré; supprimer le document avec le numéro entre et ajouter un document.

Les cas exceptionnels lors d’ajout et suppression de données vont afficher les messages erreur sur la page web.

Nous avons sécurisé la base de données lors de l'ajout et suppression de données.

Les variables sessions enregistrent les données de l’utilisateur. Toutefois, si on ouvre deux comptes sur une même adresse ip, ces sessions vont se recouvrir l' un sur l'autre.

Les principaux éléments à améliorer sont la structure de code, nous avons utilisé beaucoup de variable constante. Cela complique l’extension/ le développement de l’application à l'avenir.