Projet MediaWeb

Le projet Bibliotheque réalisé en AppServ – période B - a donné satisfaction et il vous est désormais demandé de réaliser des éléments d'une application JavaEE sur le même thème. La bibliothèque devient une médiathèque (livres, DVDs, CDs) et l'application web sera utilisée principalement par les abonnés pour emprunter/rendre des documents et les bibliothécaires pour ajouter des documents. *La réservation de documents depuis chez soi reste envisageable mais elle est en option* (à la base, demander une réservation devrait envoyer un message « Cette fonctionnalité sera bientôt opérationnelle »).

Déploiement - mise en œuvre dans la médiathèque

Le serveur JavaEE support applicatif de votre application web sera installé au sein du réseau local de la médiathèque (intranet) et les postes de la médiathèque seuls y auront accès via une adresse IP locale - si ça n'était pas le cas, un abonné pourrait emprunter ou retourner un document depuis n'importe quel poste équipé d'un navigateur web, par exemple chez lui, ce qui serait absurde. *Pour traiter les réservations dans un contexte réaliste, il faut avoir la liste des adresses IP des abonnés afin de vérifier que la réservation provient bien du poste d'un abonné référencé.*

Abonné/bibliothécaire : les utilisateurs

L'accès aux données de la médiathèque suppose une authentification (login, mot de passe). A partir de là, un utilisateur *abonné* pourra emprunter ou rendre un document, un utilisateur *bibliothécaire* pourra ajouter un document. Ces 2 catégories seront les ***utilisateurs*** de l'application web. L'ajout/modification/suppression d'utilisateurs par votre application n'est pas à envisager.

Découplage et (non-)dépendance : services – mediatheque -persistance

L'architecture de votre application est imposée afin de mettre en œuvre un découplage strict entre 3 modules :

- les services : le package ***services*** contient toutes les classes servant aux échanges http, donc toutes les servlets ;

- la médiathèque : le package ***mediatheque*** contient le domaine (Mediatheque, Document, Utilisateur) ainsi qu'une interface PersistantMediatheque qui permet de ne pas être couplé au modèle de persistance (technique d'inversion de dépendance vue en BPO-2);

- persistance : le package ***persistance***, comme son nom l'indique, représente les éléments de persistance des données ; on y trouvera donc le code JDBC et la classe « point d'entrée » de ce package, MediathequeData, implémentant l'interface mediatheque.PersistantMediatheque. On trouvera également ici les classes implémentant mediatheque.Document.

Le package ***mediatheque*** vous est fourni compilé et opérationnel sous forme d'un fichier *mediathek.jar*. Ce code compilé est donc stable et respecte bien le non-couplage du package ***mediatheque*** aux 2 autres.

Si A→B signifient que A est couplé à B

services → mediatheque ← persistance

De même, aucun couplage ne doit apparaître dans votre code entre ***services*** et ***persistance***. Vous écrirez donc les parties ***services*** et ***persistance***, sachant que *persistance.MediathequeData* doit s'auto-déclarer à *mediatheque.PersistantMediatheque* par injection de dépendance – ce qu'elle fait en appelant *setData* dans son bloc static. Un embryon de MediathequeData qui effectue cette injection vous est fourni.

A rendre

Le projet est à réaliser par binômes ou seul – au sein du bigroupe uniquement, aucun trinôme ne sera accepté ni binôme inter-bigroupes. Une soutenance/recette sera effectuée lors de la dernière séance de tp et un fichier zippé sera rendu par dépôt dans le puits en fin de semaine. Vous pouvez développer les services sous forme de servlets ou de JSP (dans ce cas, pas de package ***services****).*