TP Réseau social

Giuseppe Lipari

11 mars 2018

Table des matières

1 Description

Dans ce TP final vous devez implémenter un mini réseau social qui s'appelle *MiageBook*. Le but de cet exercice est d'apprendre à coder avec le servlet, avec l'interface Jersey, et avec la méthode *Ajax*. L'utilisation d'une base de donnée et conseillé mais pais requise. Il est nécessaire de découper l'application en couches : la couche persistance, la couche modèle et contrôle, et la couche présentation.

2 Liste des exigences

2.1 Inscription

Pour accéder à *MiageBook*, un utilisateur doit d'abord s'inscrire. Le site fournit un "form" où l'utilisateur peut choisir son pseudo, son mot de passe, son adresse e-mail, son nom et son prénom. Après avoir rempli les information, il clique sur "Submit" pour envoyer la requête au serveur.

Si le pseudo est libre (non encore utilisé par un autre utilisateur), l'utilisateur est inscrit et dorénavant il peut se connecter à l'application avec son pseudo et sa mot de passe. Si le pseudo est déjà utilisé, le serveur renvoie la même "form" avec un message d'erreur.

2.1.1 Bonus

Pendant l'inscription, vérifier si le pseudo existe avant que l'utilisateur clique sur *Submit*. Utiliser un approache Ajax + REST pour verifier si le pseudo existe : quand l'utilisateur saisit le champ "pseudo" dans la form, une fonction Ajax interroge le serveur pour vérifier si le pseudo existe, et le cas échéant, il affiche un message d'erreur à coté du champ pseudo.

2.2 Login

L'utilisateur peut se connecter avec une page de login. Si l'identifiant et le mot de passe sont corrects, l'utilisateur peut accéder à l'application.

Pour l'authentification, *FriendCar* utilise les cookies : lors qu'un utilisateur est correctement identifié, le serveur génère un cookie de manière aléatoire. Le cookie reste valide pendant que la fenêtre du browser reste ouverte et donc l'utilisateur pourra accéder à tous les services.

2.3 Liste des utilisateurs

L'utilisatrice Alice peut regarder la liste des tous les utilisateurs de *MiageBook*. Pour chaque utilisateur, *MiageBook* montre son pseudo, s'il est déjà un amis d'Alice, s'il est en ligne ou pas, et la date de la dernière connexion. Pour chaque utilisateur de la liste, elle peut décider de l'ajouter à la liste de ses amis.

2.4 Profil utilisateur

Alice peut regarder son profil utilisateur, qui contient son pseudo, son nom, son prenom, et la liste de ses amis. Pour chacun de ses amis, elle peut décider de le retirer de la liste.

2.5 Status

Alice peut publier des "status".
Un status contient :
— un identifiant unique,
— un titre,
— un texte,
— une image (optionnel),
— la date de publication,
— un propriétaire (celui qui écris le status).

Un status est visible par le propriétaire et par tous ses amis. Le texte peut contenir de liens http, dans la forme *http :/. . . /*. L'image est en format `jpg`, `png` ou `gif`.

2.6 Commentaires

Un utilisateur peux ajouter un commentaire à un des status de ses amis. Un commentaire contient :
— un identifiant unique
— l'identifiant du status
— la date
— un propriétaire (celui qui écris le status)
— un texte.
Le texte peut contenir de liens http, dans la forme *http :/. . . /*.

2.7 Visualisation

Un status est visualisé de manière très simple :
— D'abord le titre, l'auteur et la date ;
— le texte ;
— l'image optionnel ;
— la liste de commentaires.

2.8 Feed

Un *feed* est la liste des status publié par un utilisateur. Par exemple, la liste des status de Alice est le *feed* d'Alice. Un utilisateur peux visualiser à tous moment son feed.

2.9 Home Page

Dans sa page principale, Alice voit la liste des dernier 10 status publiés par elle et par ses amis.
Les status sont visualisés en ordre de date de publication, le plus récent en tête de page.

2.10 Ajouter un status

Pour ajouter un status, on utilise une form :
— l'utilisateur peut écrire le texte de son status ;
— il a la possibilité de sélectionner une image se son ordinateur.
Le status est publié lors du click sur le bouton "submit".

2.11 Bonus

Les status et les commentaires sont mis à jour en temps réel : tout les deux seconds, un script *Ajax* récupère les nouveaux status et commentaires, et met à jour le contenu de la page.

3 Implementation

3.1 Base de donnée

Vous pouvez utiliser un serveur BD en locale, ou le serveur MySQL sur WebTP (Intranet FIL).
Avant de créer la BD, je vous conseil de concevoir le couche "modèle" de votre application :
— les classes ;
— leur relations.
Une fois le modèle bien conçu et testé, vous ajouterez la couche de persistance pour mémoriser le données sur la BD.

3.2 Première étape : Inscription et login

On démarre la conception et l'implémentation de notre application par la phase d'inscription et de login.

1. Concevoir la classe `ProfilUtilisateur` pour stocker les information concernant un utilisateur.
2. Concevoir un form HTML pour l'inscription.
3. Concevoir une *servlet* dédié au traitement de l'inscription ;
   — si la procédure se termine correctement, la servlet renvoie à la page de login ;
   — si la procédure ne se termine pas correctement, la servlet renvoie à un page d'erreur.

Vous pouvez coder le servlet "à la main" en Java, ou utiliser des pages JSP. À vous de choisir !
La page login permet de se logger sur l'application. Si le login est correct, la servlet prépare un *session* (ou directement un cookie) et renvoie sur la page principale qui contient le menu des services :

```
1. Montrer mon profile
2. Lister les utilisateurs
3. Logout
```

Le menu est suivi par la liste des status (voire 2.9). Pour l'instant on n'a pas de status, donc on ne montre rien.

3.3 Deuxième étape : Profil utilisateur et liste d'amis

Dans un deuxième temps, on s'occupe de la notion de amis, c'est à dire la relation entre les profils utilisateurs.

1. Concevoir un page pour lister tous les utilisateurs, et un page pour lister le profil utilisateur avec la liste des amis.
   Pour construire les pages, vous pouvez utiliser un simple JSP ou une servlet.
2. À coté de chaque amis, on ajoute des liens pour permettre de ajouter/retirer des amis.
   Exemple :

   ```
   Liste utilisateurs

   1. Alice   est un amis   <retirer des amis>
   2. Bob                   <ajouter aux amis>
   3. Charlie est un amix   <retiser des amis>
   4. David                 <ajouter aux amis>
   ...
   ```

   En cliquant sur le lien, une fonction JavaScript est exécuté pour ajouter/retirer l'amis. La fonction Javascript

— appelle une fonction REST pour ajouter/retirer l'amis du modèle
— si la fonction REST se termine correctement, met à jour la page en changeant le lien.

Attention : toutes les fonctions vérifient que l'utilisateur est bien connecté avant d'exécuter l'opération, et elles renvoient un erreur (ou directement à la page de login) s'il n'est pas le cas.

3.4 Troisième étape : status

1. Concevoir les classes `Status` et `Commentaire`.
2. Préparer un form pour permettre de créer un status. En cliquant sur submit, il faudra appeler un servlet qui crée le status et il l'ajoute à la liste des status.
3. Modifier la page principale d'un utilisateur pour montrer la liste de status.
   — coder une méthode REST pour récupérer la liste des status en format JSON
   — coder une fonction Javascript qui appelle la fonction REST et modifie la page pour permettre de visualiser les status.
4. Finalement, préparer une form pour permettre d'ajouter un commentaire à un status.

Bonus : vous pouvez mettre à jour la visualisation des status en temps réel, en appelant la fonction javascript correspondante tous le 2 seconds.