FaceMash

L'application

L'application à réaliser durant ce TP est tirée du film The Social Network de David Fincher. Pour ceux qui n'auraient pas eu l'occasion de le voir, ce film relate la naissance du site Facebook. Une des première séquence de ce film relate la création d'une application permettant de voter pour la plus "jolie" fille de la fac d'Harvard. Cette application, nommée FaceMash est extrêmement simple et a été créée en quelques heures dans une chambre d'étudiant. Prêt à relever le défi ?

La vidéo tirée du film est disponible ici : http://dl.dropbox.com/u/3121809/TSN.mp4

Architecture générale

La présentation du site est très simple. Deux vues principales, la page de vote et la page de scores. La page de votre est composée de deux photos "cliquables". Un clic sur une photo correspond à un vote pour la photo en question, dans ce cas là, la photo reste inchangée, seule la seconde change. La page de score est simplement la liste des photos disponibles triée par nombre de votes (ou par score).

L'application est composée de la façon suivante:

- Un modèle de base de donnée, Face, représentant une photo. Ce composant est une entité JPA comportant un nombre de vote et l'URL de la photo
- Un contrôleur principal permettant d'afficher les photos, de voter pour l'une d'entre elles, d'afficher les scores, etc …
- De modèles contenant les informations à afficher dans les vues * *Un modèle pour les informations standard (photos à afficher, liste des photos, etc …)* * Un modèle pour l'affichage des erreurs
- D'un composant de "bootstrap" permettant de peupler la base de données par rapport aux photos contenues dans le dossier img
- De différentes vues JSF2 au format .xhtml

Organisation du projet

arborescence du projet une fois fini :

```
.....
|
├── src
|   ├── conf
|   |   ├── MANIFEST.MF
|   |   └── persistence.xml
|   └── java
|       └── com
|           └── acme
|               └── facemash
|                   ├── ApplicationBootstrap.java
|                   ├── controllers
|                   |   ├── ApplicationController.java
```

```
|                   |   ├── ApplicationModel.java
|                   |   └── ErrorModel.java
|                   ├── models
|                   |   └── Face.java
|                   └── util
|                       ├── ApplicationBoundary.java
|                       ├── ContextServlet.java
|                       ├── Elo.java
|                       ├── Functionnal.java
|                       └── ProducerUtils.java
└── web
    ├── WEB-INF
    |   ├── beans.xml
    |   └── web.xml
    ├── bootstrap.css
    ├── error.xhtml
    ├── head.xhtml
    ├── img
    |   ├── cat01.jpg
    |   ├── cat02.jpg
    |   ├── cat03.jpg
    |   ├── cat04.jpg
    |   ├── cat05.jpg
    |   ├── cat06.jpg
    |   ├── cat07.jpg
    |   ├── cat08.jpg
    |   ├── cat09.jpg
    |   ├── cat10.jpg
    |   ├── cat11.jpg
    |   └── cat12.jpg
    ├── index.xhtml
    ├── jquery-1.7.min.js
    ├── jquery.jcarousel.min.js
    ├── next-horizontal.png
    ├── next-vertical.png
    ├── prev-horizontal.png
    ├── prev-vertical.png
    ├── skin.css
    └── stats.xhtml
```

Etapes

Base de données

Une base de données est nécessaire pour l'application FaceMash afin de retenir les photos disponibles et le nombre de votes pour chaque photo.

Dans un premier temps, il va être nécessaire de créer le fichier persistence.xml. NetBeans propose un assistant permettant de générer proprement le fichier en question. Lors de la génération du fichier, choisir la datasource jdbc/__sample qui est une base de données de test disponible dans GlassFish.

Il va ensuite être nécessaire de créer l'entité Face. Cette classe comporte deux champs, l'url de la photo et le nombre de vote pour chaque photo. N'oubliez pas de compléter les accesseurs pour chaque

champs. Vous pouvez également ajouter les méthodes métier relative à l'entité Face (récupérer une face aléatoire, récupérer toutes les entités, compter toutes les entités, voter, etc …). Vous pouvez également ajouter ces méthodes dans un bean à part.

Il va également être nécessaire de compléter la classe ProducerUtils.java pour que le persistenceUnit utilisé soit le même que celui généré dans le fichier persistence.xml.

Il va également être nécessaire de modifier la classe ApplicationBoostrap (méthode findAndRegisterImages) afin que celle-ci scanne le dossier img et enregistre les faces correspondantes à chaque image trouvée. Cette méthode n'est appelée qu'une fois et la classe possède le chemin du dossier img nommé 'imgPath'. Pour chaque image touvée, enregistrer une instance de Face en base de données avec le nombre de vote à zéro et une url de type '/img/nomDeLimage.jpg'.

Contrôleur et modèles

Le contrôler ApplicationController va être responsable des interactions avec l'utilisateur. Chaque méthode retourne une chaine de caractère correspondant à la vue vers laquelle on veut aller. Chaque méthode est censée calculer les données nécessaire à l'affichage de la vue et les mettre dans le modèle de la vue afin que ces données soient accessibles depuis la vue. Attention, pour que cette classe soit accessible depuis la vue, n'oubliez pas de l'annoter @Named. De plus, un contrôleur étant stateless par nature, il peut-être nécessaire de lui ajouter un scope de type requête (@RequestScoped).

Les modèles sont simplement des beans accessibles depuis la vue (ne pas oublier le @Named) contenant les données de la vue courante. Le modèle de l'application est quasiment complet. Il va par contre être nécessaire de concevoir le modèle d'erreur.

Vues

Les vues sont des fichiers .xhtml contenant des composants JSF2. Ajouter le code nécessaire pour compléter les vues.

Score Elo

La classe Elo.java contient une méthode permettant de calculer le score d'une photo par rapport à une autre (http://fr.wikipedia.org/wiki/Classement_Elo). Ajouter le code nécessaire dans votre application afin de gérer le score de chaque photo.

Pour les plus rapides

Exposition REST

Exposer les différentes fonctions majeures de l'application via des services de type REST envoyant des données sous forme de JSON.

Client HTML/JS

Créer un petit client HTML/JS pour consommer les services REST de l'application.