Examen - LI 328

18 mai 2015

Les documents sont autorisés.

Contexte général

Nous nous intéressons au développement d'un site de paris en ligne, dont les spécifications principales à respecter sont les suivantes :

- Seuls les opérateurs du site peuvent inscrire de nouveaux utilisateurs via une application particulière. L'inscription d'un utilisateur produit un identifiant d'utilisateur utilisé pour toute interaction future de l'utilisateur (via une application cliente) avec le serveur de réservation.
- Chaque utilisateur possède un id, un prenom, un nom, un login et un solde. Le solde peut être crédité par un service de paiement en ligne ; Il ne peut en aucun cas être négatif.
- Sur le site, n'importe quel utilisateur identifié peut proposer un nouveau pari, correspondant à une question à choix multiples.
- Un utilisateur peut miser sur une ou plusieurs réponses d'un pari. Il ne peut pas rejouer sur un pari auquel il a déjà participé.
- Lorsqu'un utilisateur se connecte au site (impossible de voir quoi que ce soit en mode non connecté), la liste des paris en cours est affichée à l'écran.
  - Chaque pari prend la forme d'une question associée à une liste de réponses possibles.
  - Chaque réponse est associée à une cote (calculée automatiquement par le serveur en fonction des mises effectuées sur ce pari) ainsi qu'à une zone de texte où l'on peut entrer la somme que l'on souhaite miser sur cette réponse.
  - Si l'utilisateur connecté a déjà misé sur un des paris, ses mises effectuées sont affichées à la place des zones de saisie des mises et le bouton de validation est caché.
  - Si l'utilisateur connecté est le propriétaire du pari, un bouton de clôture est associé au pari afin de pouvoir clôturer le pari.
- Le propriétaire d'un pari peut le fermer en donnant la bonne reponse parmi les choix qui étaient proposés. Les comptes de chaque utilisateur ayant parié sur cet item avec la bonne réponse sont alors crédités (en fonction de la cote de la reponse fournie : recompense = cote * mise).
- Le login et le solde de la personne connectée sont affichés en haut à droite de l'écran.

Exercice I - Spécifications des Web Services (3 points)

Pour chaque service, vous spécifierez les entrées du service, et donnerez un exemple de sortie au format JSON. Par exemple :

```
== Service Inscription
== Entrée:
        nom: nom de l'utilisateur
```

```
          prenom: prenom de l'utilisateur
          login: login a utiliser
          password : le mot de passe (en clair...) de l'utilisateur
  == Description:
   Le service permet d'inscrire un nouvel utilisateur au service
  == Erreurs possibles: * ce nom + prenom existe déjà
                        * ce login existe deja
  == Exemple de sortie:
- si l'inscription s'est faite (renvoie le numero associé à l'utilisateur) :
   {id = "0BAE569F3AC"}
- si le nom/prenom existent deja
   {error = "utilisateur deja inscrit"}
```

Question 1
Service *A*jouterPari permettant la creation d'un pari.

Question 2
Service *F*ermerPari permettant la fermeture d'un pari.

Question 3
Service *L*isteParis permettant d'afficher la liste des paris en cours (non encore fermés).

Question 4
Service *M*iser permettant de valider les mises saisies pour un pari.

Exercice II - Bases de Données (4 points)

Les informations permettant de connaitre les paris en cours et les utilisateurs seront stockées dans une base MySQL. La base MongoDB servira à stocker les mises effectuées par les différents utilisateurs.

Question 1
Qu'est ce qui justifie ce choix de bases de données ?

Question 2
Quelles tables allez vous définir en SQL ? Décrivez ces tables (nom, attributs, et type des attributs)

Question 3
Ecrivez la méthode JAVA qui permet de récupérer la liste des paris en cours.

Question 4
Décrivez sous forme de JSON le format de stockage dans la base MongoDB.

Question 5
Ecrivez la méthode JAVA qui permet d'ajouter les mises d'un joueur sur un pari (toutes les infos utiles sont passées en paramètres). A quel moment cette procédure sera appelée ?

Exercice III - Servlets (6 points)

Question 1
Ecrivez une servlet (et méthodes nécessaires) permettant d'implémenter le service *M*iser décrit à l'exercice 1. Vous commencerez en donnant le code "haut niveau" du service, et implémenterez les méthodes plus spécifiques par la suite.

Question 2

Ecrivez une servlet (et méthodes nécessaires) permettant d'implémenter le service *L*isteParis décrit à l'exercice 1. Vous commencerez en donnant le code "haut niveau" du service, et implémenterez les méthodes plus spécifiques par la suite. On considèrera que l'on dispose d'une fonction getOdds(p) retournant la liste des cotes des reponses (une ArrayList contenant en indice i-1 la cote de la reponse i) pour un pari p passé en paramètre.

Exercice IV - Map/Reduce (3 points)

Question 1

Ecrivez la fonction java (basée sur map reduce) permettant de calculer la cote des réponses d'un pari en cours. Si personne n'a encore parié, toutes les cotes sont égales à 1. Sinon, la cote d'une réponse i est égale à la somme des mises sur toutes les reponses du pari concerné, divisée par la somme des mises effectuée sur cette reponse i :

$$cote(p,r) = \frac{\sum_{i=1}^{n_p} \sum_{u \in U} mise(p,i,u)}{\sum_{u \in U} mise(p,r,u)} \quad (1)$$

avec p le pari et r le numero de la reponse concernés par le calcul de cote, U l'ensemble des utilisateurs du site, n_p le nombre de reponses possibles pour le pari p et $mise(p,i,u)$ une fonction retournant la mise effectuée par un utilisateur u pour la reponse i du pari p.

Exercice V - Développement d'un client (6 points)

On souhaite développer une application client permettant d'afficher les paris en cours et donnant la possibilité de miser sur des réponses proposées.

Question 1

Donner le code HTML / CSS d'une boîte générique de type block contenant 2 paris (penser aux différents éléments que doit comporter la boîte d'un pari : thème, reponses possibles, zones de saisie des mises, bouton de validation, éventuellement bouton de clôture, etc...).

Question 2

En considérant que les communications doivent se faire en asynchrone par le biais d'AJAX et que la servlet correspondant au service ListeParis est stockée sur le même domaine que le code client, écrire le code javascript permettant de récupérer la liste des paris en cours (uniquement le code de l'appel AJAX). En cas de succès de la communication, on appelera une fonction traiteReponseListeParis (à écrire dans les questions suivantes) qui attend la réponse au format texte. En cas d'echec on affichera un message "Echec de la communication avec le serveur".

Question 3

Ecrire les constructeurs d'objets Javascript suivants :

- Utilisateur contenant l'identifiant et le login d'un utilisateur ;
- Pari contenant les informations correspondant à un pari : propriétaire (qui est un Utilisateur), thème, reponses possibles associées à leurs cotes et aux mises éventuellement effectuées par l'utilisateur connecté ;
- ListeParis contenant la liste des paris en cours.

Pour les types d'objets Pari et ListeParis, vous implémenterez la méthoge **getHtml()** correspondante. Vous ferez en sorte de respecter les indications donné dans le contexte général du sujet.

Question 4

Écrire une fonction ListeParis.revival qui définit la manière de reconstruire un objet ListeParis à partir d'un texte au format JSON envoyé par le service ListeParis définit à l'exercice 1. Cette fonction sera utilisée dans la fonction traiteReponseListeParis de la manière suivante :

```
var liste=JSON.parse(json_texte,ListeParis.revival);
```

avec json_texte correspondant au texte retourné par la servlet ListeParis

Question 5

Écrire la fonction traiteReponseListParis permettant d'afficher la liste de paris en cours retournée par le serveur dans l'interface client.