Master CCI – Cours de systèmes d'information répartis

Examen

Durée : 45 minutes – documents autorisés – calculatrices et téléphones portables interdits. Les deux premières pages constituent une description de l'application à réaliser ; les questions sont en page trois.

Description de l'application

On souhaite faire une application de « chat » en Java, avec les techniques étudiées en cours, à l'aide d'un serveur web et d'un moteur de servlets et de JSP. Les tâches à réaliser sont les suivantes :

- Permettre à chaque utilisateur (client web) de saisir un pseudonyme.
- Permettre à chaque utilisateur de saisir un message textuel.
- Mémoriser les interventions de tous les utilisateurs, précédées de leurs pseudos.
- Les renvoyer aux clients.

Description et fonctionnement des composants de la couche présentation :

La page d'accueil (« index.html », voir figure 1) contient un formulaire demandant un pseudo à l'utilisateur. Le pseudo est saisi dans un champ texte, qui est envoyé par la méthode POST à une servlet, appelée «Init ». Cette servlet effectue un traitement de la requête (voir plus bas), et redirige l'utilisateur sur une page HTML appelée « frames.html » qui divise l'écran en deux cadres (frames), le premier en haut de l'écran, appelé «haut », et le second en bas, appelé « bas » (voir figure 2). Le cadre du haut contiendra les messages envoyés par les différents utilisateurs (voir question 2) et celui du bas une page HTML simple, appelée «Saisie.html » avec un formulaire permettant la saisie d'un message, également envoyé par POST à « haut ».

Remarques :

- la gestion des cadres est réalisée dans frames.html, et ne nous intéresse pas ici.
- On ne se préoccupe pas non plus de la sauvegarde des messages. Leur durée de vie est la même que celle du serveur.

Exemples d'interface

Figure 1. La page index.html

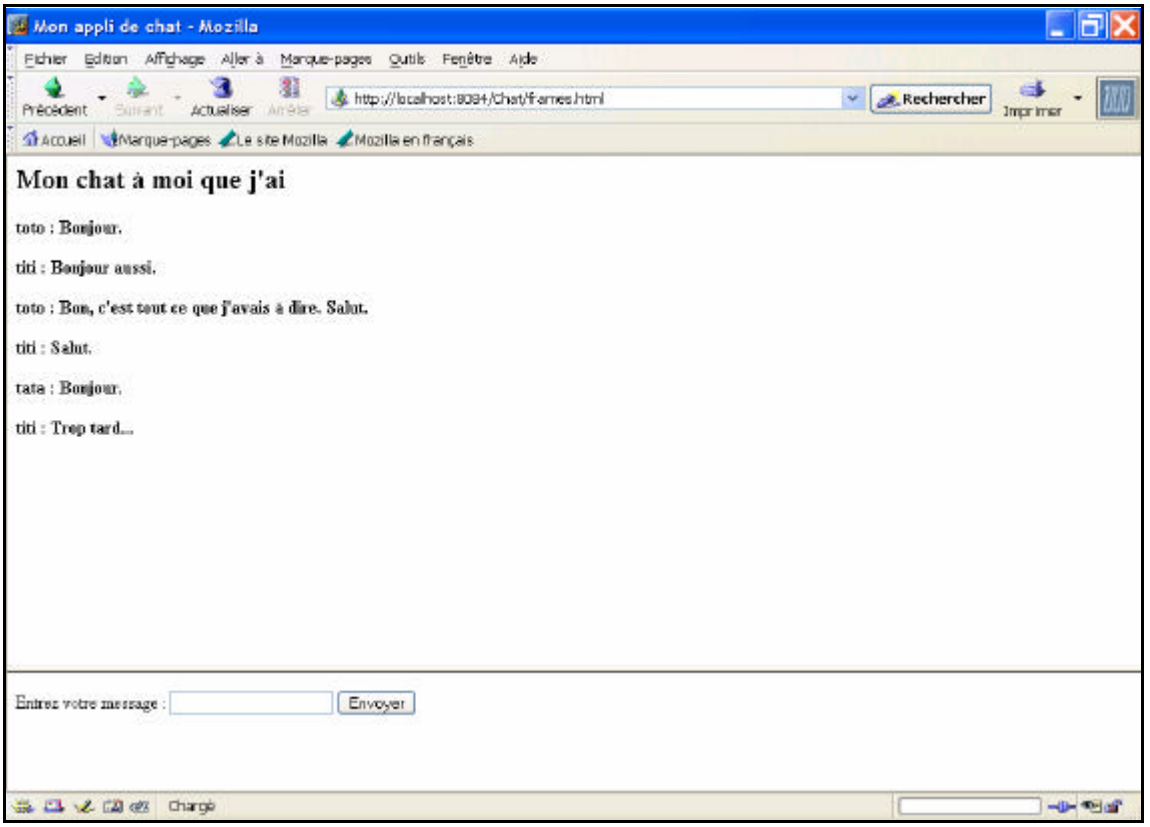

Figure 2. Les cadres « haut » et « bas » positionnés par frames.html.

Interface de la couche « métier » :

L'historique des messages est géré par un composant JavaBean appelé `GestionMessages`, qui communique avec la couche présentation de l'application par les méthodes :

```
public void setMessage (String[] message)
public String getMessages ()
```

L'utilisation de ce composant est la suivante :

- à la réception d'une requête contenant un message, la couche présentation met l'auteur et le message (dans cet ordre) dans un tableau de `String`, et l'envoie au Bean en utilisant la première fonction.
- la récupération de l'ensemble des messages (auteurs et textes tapés), **déjà formatés en HTML**, se fait en appelant `getMessages()`.

Questions

Remarque : ne vous préoccupez pas de l'appel du JavaBean. Vous pouvez utiliser une variable `gm`, représentant la même instance de `GestionMessages` pour toute l'application.

Question 1

Écrivez la servlet init, qui effectue les opérations suivantes :

- Récupération du pseudo de l'utilisateur (le nom du champ texte du formulaire est « pseudo »)
- Stockage dans un attribut de l'objet `HttpSession` associé à l'utilisateur (voir annexe).
- Génération de la réponse au client par l'envoi d'un entête HTTP `Send-Redirect` vers la page « frames.html ».

Question 2

- Écrivez une servlet ou une JSP (au choix), qui reçoit une requête contenant un message, ajoute ce message à l'historique et génère une page HTML basique contenant l'ensemble des messages.
- Quel est le principal défaut de cette application par rapport à un chat « normal » ? Comment pourrait-on y remédier ? Dans ce cas, suggérez une amélioration de la servlet/JSP précédente, en termes de charge du serveur et d'occupation du réseau.

Annexe

Rappel : une session HTTP contient l'ensemble des informations liées à la session d'un utilisateur particulier. Cet objet est accessible depuis l'objet requête et géré par le container de servlets.

Récupération d'une session HTTP à partir d'une instance de l'objet `HttpRequest` :

`HttpSession getSession(boolean create)`

Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session (En gros, l'argument doit être « `true` »).

Méthodes de l'objet `HttpSession` :

`void setAttribute(java.lang.String name, java.lang.Object value)`

Binds an object to this session, using the name specified (permet de lier un objet de nom `value` à un attribut de la session nommé `name`).

`java.lang.Object getAttribute(java.lang.String name)`

Returns the object bound with the specified name in this session, or null if no object is bound under the name (permet de récupérer la valeur de cet attribut).

Rappel : String hérite de java.lang.Object