TP 2 : programmation côté serveur
À rendre pour le mardi 6 mai 2008

Introduction

Dans ce TP, vous allez réaliser 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 principales tâches à réaliser sont les suivantes :

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

Pour ce TP, vous allez utiliser l'EDI NetBeans installé dans les salles de TP Windows. Cet outil vous permet de réaliser différents types de projets Java. NetBeans intègre un serveur Tomcat (serveur web Apache + container de servlets Catalina), et peut être connecté à un serveur d'applications plus complexe. Vous trouverez quelques détails sur l'utilisation de NetBeans en annexe.

Remarque : il y a plusieurs versions de NetBeans installées sur les machines, dont certaines sont des versions bêta instables. Vous ne devez utiliser que les versions 4.1 ou 5.0

Pour commencer, les différents composants de base de cette application sont décrits dans la partie suivante. Un certain nombre d'amélioration sont proposées dans la dernière partie.

Description de l'application

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, et envoyé à une servlet appelée **Debut**.

Figure 1. La page index.html

La servlet **Debut** effectue les opérations suivantes :

- Récupération du pseudo de l'utilisateur (le nom du champ texte passé en paramètre par le formulaire précédent est « pseudo »),
- Stockage de ce pseudo dans un attribut de l'objet `HttpSession` associé à l'utilisateur (voir annexe),

- Redirection de l'utilisateur par l'envoi au client d'un code de réponse HTTP `Send-Redirect` (méthode de l'objet `HttpServletResponse`), vers une page HTML nommée **frames.html**.

La page **frames.html** 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). Cette page n'étant pas d'un intérêt particulier pour ce TP, son code vous est donné ci-dessous.

```
<html>
  <head>
    <title>Mon Appli de chat</title>
  </head>
  <frameset rows="*, 200px">
   <frame name="haut" id="haut" src="Affiche" />
   <frame name="bas" id="bas" src="saisie.jsp" />
  </frameset>
</html>
```

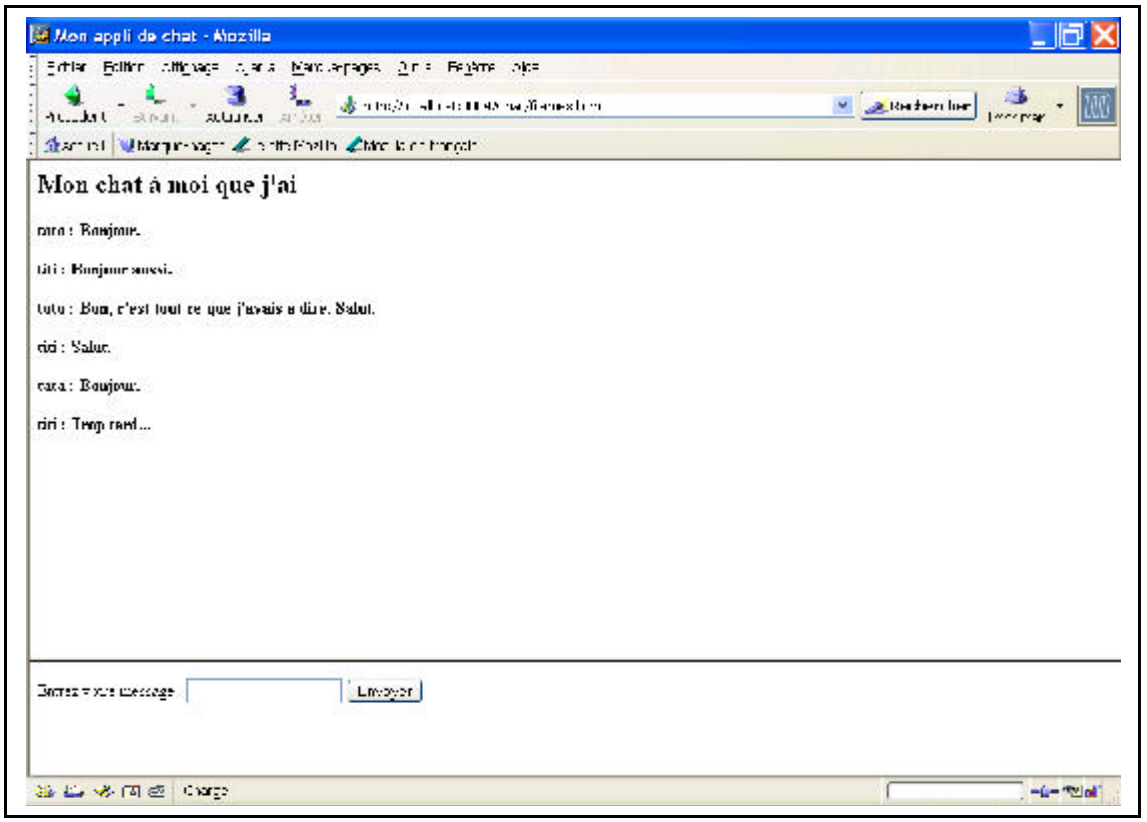

Figure 2. Les cadres « haut » et « bas », positionnés par frames.html.

Le cadre du haut contient une servlet **Affiche**, destinée à l'affichage des messages envoyés par les différents utilisateurs. Lorsque cette servlet reçoit une méthode POST, celle-ci contient un message à rajouter à la liste ; lorsqu'elle reçoit une méthode GET, il s'agit d'une demande de réactualisation de la page, sans envoi de nouveau message. Cette servlet renvoie, dans tous les cas, une page HTML comportant les éléments suivants :

- un en-tête « Refresh » http, indiquant que la page devra être rechargée toutes les 5 secondes (pour actualiser les messages)
- une ou plusieurs lignes de titre et de présentation de l'application,

- l'ensemble des messages envoyés depuis le démarrage du serveur, à raison d'un message par ligne, et en faisant apparaître pour chacun d'eux le pseudo de son auteur et le texte.

Remarque :

Dans un premier temps, la gestion des messages pourra se faire dans une variable globale de type String, en interne de la servlet.

Le cadre du bas contient une page JSP, appelée **Saisie.jsp** contenant un formulaire permettant la saisie d'un message, envoyé par POST à « haut » (à l'aide de l'attribut « target » de la balise « form »). Cette page JSP récupère l'attribut de session contenant le pseudo, défini dans **Debut**, et le stocke dans un champ « hidden » du formulaire, de façon à le renvoyer au serveur en même temps que le texte du message.

Question subsidiaire

Quel est le principal défaut de cette application par rapport à un chat « normal » ? Comment pourrait-on y remédier ?

Amélioration de l'application

Gestion des messages

Vous allez créer une classe java appelée **GestionMessages**, capable de prendre en charge la réception de messages, leur stockage et leur envoi au format HTML, aux différents composants de l'application.

Cette classe communique avec la couche présentation de l'application (décrite dans la partie précédente) par des méthodes de type set et get :

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

Les messages sont gérés en interne de cette classe sous forme d'un arbre DOM et stockés dans un fichier XML (à l'aide des méthodes parse() et writeXmlToFile() de la classe DOMUtil, disponible dans le fichier http://www710.univ-lyon1.fr/~lmedini/SIMA/DOMUtil.zip).

L'utilisation de cette classe 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 en informe cette classe en utilisant la première fonction.
- la récupération de l'ensemble des messages (auteurs et textes tapés), **déjà formatés en XHTML**, se fait en appelant `getMessages()`.

Modifiez les composants de la question précédente pour leur permettre d'utiliser cette classe.

Rendu du TP

Ce projet est à rendre pour le mardi 2 mai 2008.

Vous devrez envoyer par mail à Lionel Médini un fichier zip contenant les éléments suivants :

- les sources du TP (fichiers .html, .java et .jsp),
- un rapport au format Word ou PDF, qui décrira votre application, présentera un diagramme de classes UML de votre application « améliorée », et répondra à la question subsidiaire posée plus haut.

Remarques :

- dans le diagramme de classes, une page JSP sera considérée comme une servlet,
- ce diagramme mettra également en évidence la séparation des trois couches présentation, métier et données,
- le code devra être suffisamment commenté.

Annexe : notes sur l'utilisation de l'EDI NetBeans

Création d'un projet

Pour ce TP, vous créerez un nouveau projet de type Web application.

Vous pouvez localiser le répertoire de votre projet sur un support amovible (clé USB).

Configuration de NetBeans et de Tomcat

Vous pouvez accéder à différentes informations de configuration dans l'onglet « Runtime » du cadre gauche de l'application.

En particulier, vous pouvez démarrer, arrêter ou configurer le serveur Tomcat fourni avec NetBeans, en utilisant le menu contextuel du menu « Servers », puis « Bundled Tomcat ». La configuration de Tomcat est stockée dans un fichier nommé « server.xml ».

Remarque : les fichiers de configuration spécifiques à un projet sont dans le sous-menu « Configuration files » du projet (onglet « Projects »). Dans Tomcat, l'essentiel des informations de configuration d'une application web est stocké dans un fichier nommé « web.xml ».

Gestion des projets

La plupart des opérations de gestion des projets se font dans le cadre « Project », à gauche de la fenêtre.

Par défaut, NetBeans crée une page d'accueil nommée « index.jsp », dont vous n'aurez pas besoin. Vous pouvez donc la supprimer. Pendant que vous y êtes, vous pouvez modifier le fichier « web.xml » pour que la page d'accueil du projet ne s'appelle plus « index.jsp », mais « index.html ».

Pour ajouter un élément à votre projet, le plus simple est de faire un clic droit sur le nom du projet et de sélectionner « New » suivi du type d'élément que vous souhaitez ajouter (« HTML… », « JSP… », « servlet… »)

Pour tester un élément dynamique (servlet ou JSP), vous devrez obligatoirement cliquer sur « Run file », dans le menu contextuel disponible sur le nom de l'élément, même si vous ne pouvez pas l'utiliser sans l'envoi de paramètres par un autre élément.

Remarque : lorsque vous lancez un projet web depuis NetBeans, le nom du serveur dans l'URL appelée est « localhost ». En raison de problèmes de configuration sur les machines, il faut remplacer ce nom de serveur par l'adresse IP de la boucle locale : « 127.0.0.1 ».