TP 6

Thèmes du TP :

- Serveurs de type sessions
- Déploiement : une introduction
- httpunit : première utilisation

Préambule : Les classes de tests unitaires présentes dans votre projet Bluej, utilisent cette fois **httpunit** (http://www.httpunit.org).

Il est utile de parcourir cette présentation http://httpunit.sourceforge.net/doc/cookbook.html

Cet exemple d'utilisation de HttpUnit, intégré à votre environnement BlueJ, se trouve dans le paquetage de la question1, classe nommée TestAvecHttpUnit. Cette classe teste la validité d'un lien sur la page de l'unité. Essayez-la !

```
package question1;

import com.meterware.httpunit.*;


public class TestAvecHttpUnit extends junit.framework.TestCase{

  public void test_lien_accessible(){
    try{

      WebConversation conversation = new WebConversation();
      WebRequest request = new GetMethodWebRequest("http://www.esiee.fr/~bureaud/Unites/In4a21,
      WebResponse response = conversation.getResponse( request );
      assertTrue("jfod.cnam.fr inaccessible ?, peu probable...",response.getText().length() > (

      WebLink link = response.getLinkWith( "tp_http.jar" );
      assertNotNull("le lien du tp_http n'est pas actif, patience ...",link);

    }catch(Exception e){
           fail("exception inattendue ! " + e.getMessage());
    }
  }
}
```

Un serveur Web : la classe SimpleHttpd2 extrait du livre : Java Server and Servlets, by **Rossbach**.

question1-1) Lire attentivement ce source Java de serveur Web puis exécuter ce serveur sur votre PC, vérifier le bon fonctionnement, un client privilégié est votre navigateur, l'url est alors http://localhost:8100/index.html.(ou bien http://127.0.0.1:8100/index.html., si cela ne fonctionne pas, vérifiez les paramètres proxy de votre navigateur).

Attention ! A l'ESIEE dans le navigateur (uniquement !), il faut remplacer `localhost` par le nom du pc en minuscules (exemple: pc5103k).

Modifier ce serveur, en ajoutant sur la console pour chaque requête reçue, l'adresse IP de la machine du Client,

comme le suggère cette trace d'exécution

[147.215.228.59] -- Request: GET / HTTP/1.1
[147.215.228.59] -- Request: GET /index.html HTTP/1.1

[147.215.228.92] -- Request: GET /test.html HTTP/1.1
....

[*adresse IP du client*] -- Request: *méthode HTTP*

voir la méthode getInetAddress de la classe Socket et la classe InetAddress

question1-2) Exécuter maintenant l'application SimpleWebClient; vérifiez sur la console du client les éléments retournés par :

1. le serveur HTTP adressé... par défaut sur le port 80 comme www.esiee.fr, www.cnam.fr, www.google.fr,
2. et ensuite le serveur de la question précédente en utilisant dans le WebClient le même port (voir la ligne 165 de la classe SimpleHttpd2)

notez que si un proxy est nécessaire ces quelques lignes suffisent, à l'ESIEE, par exemple

```
public static void setHttpProxy(String proxyHost,int proxyPort){
  Properties prop = System.getProperties();
  prop.put("proxySet","true");
  prop.put("http.proxyHost",proxyHost);
  prop.put("http.proxyPort",Integer.toString(proxyPort));
}

setHttpProxy("cache.esiee.fr",3128);
```

1) Ajout de service au serveur SimpleHttpd2

Enrichir le serveur étudié à la question 1 d'un "service Web"

- Ce service retourne les propriétés du serveur, de la JVM qui héberge le serveur, sous la forme d'une table "HTML" accessible depuis n'importe quel client HTTP, votre navigateur par exemple

  La page HTML retournée par ce nouveau service contient une table à 2 colonnes, reflétant la structure de données d'une "Properties", il suffit de retourner cette page (une "String") à chaque demande.

clé	valeur

  Soit en HTML

```
<table border="3">
    <tr>
        <td><font size="6">clé</font></td>

        <td><font size="6">valeur</font></td>
    </tr>
    <tr>
        <td> </td>
        <td> </td>
    </tr>
</table>
```

  Le format HTML d'une table se trouve en: http://www.w3.org/TR/html4/struct/tables.html

  L'URL associée à ce service doit être http://localhost:8200/**getProperties/**

  - un exemple de ce qui est attendu est en ligne ici : http://jfod.cnam.fr/csiml/getProperties/

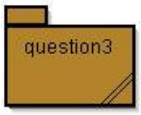
Notes :

1. Les propriétés d'une JVM sont accessibles par la méthode System.getProperties();
2. La méthode ci-dessous permet de retourner une String au client HTTP,
   celle-ci ajoutée à la classe SimpleHttpd2 peut vous être utile

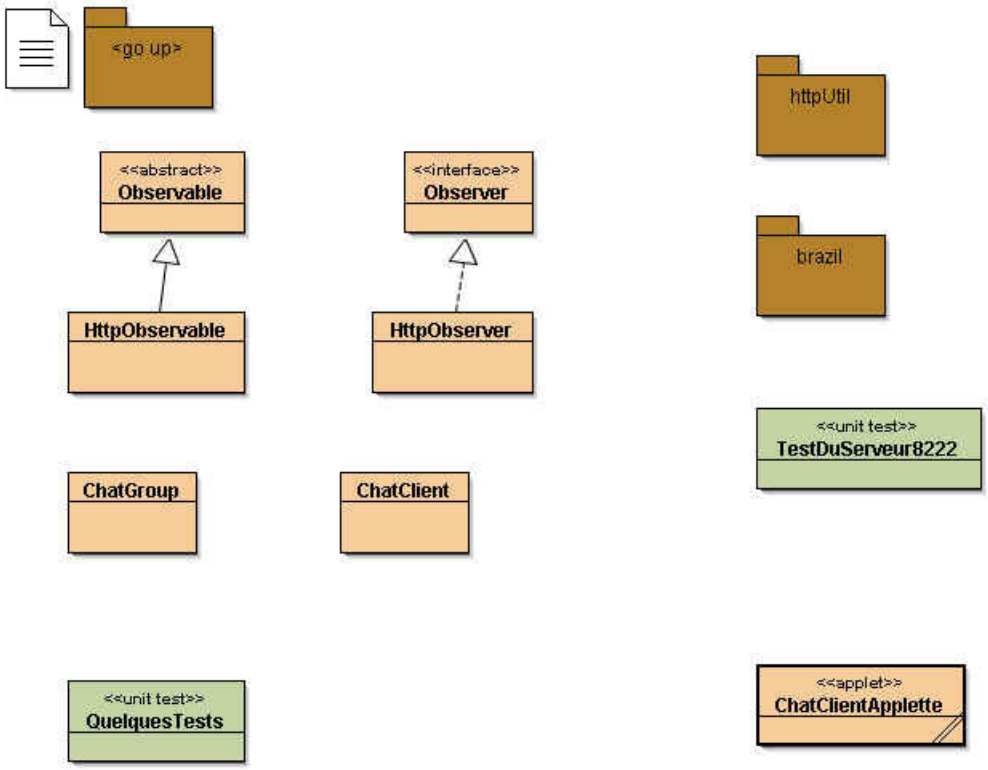
```
public void sendDocument(DataOutputStream os, String message) throws IOException{
  try{
   os.write("HTTP/1.0 200 OK\r\n".getBytes());
   os.write(new String("Content-Length: " + message.length() + "\r\n").getBytes());
   os.write("Content-Type: text/html\r\n\r\n".getBytes());
   os.write(message.getBytes());
   os.flush();
  }catch(Exception e){
    e.printStackTrace();
   }
}
```

Complétez la classe de test en vérifiant, par exemple, que le service Web fournit la version de la JVM : java.vm.version

question2_2) Echanger quelques requêtes avec vos voisins ... Vérifier les données obtenues, (notamment le type du processeur, la version du j2se installée...)



Réalisation d'un "chat" au protocole HTTP, utilisation du patron Observateur



Une implémentation du pattern Observateur :

Observable: cette classe maintient une liste d'observateurs, à chaque notification,tous les observateurs sont prévenus
HttpObservable : l'observable, le modèle est ici une URL

Observer: cette interface contient la déclaration de la mise à jour, déclenchée à chaque

notification
HttpObserver : l'observateur, la mise à jour (méthode update) déclenche une requête HTTP

ChatGroup, le groupe de discussion du chat**, à compléter**
ChatClient, le client abonné au groupe de discussion, **à compléter**

package httpUtil, quelques utilitaires au protocole HTTP**, serveurWeb, requêtes ...**
implémentation du pattern Command : voir httpUtil.Command

package brazil, utilitaires extraits du projet Brazil de Sun
http://www.experimentalstuff.com

question3_Mise_en_oeuvre) Méthode **GET** et ses paramètres

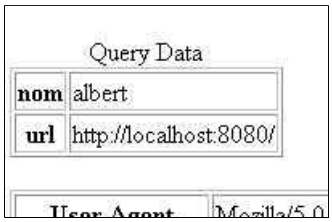
A cette URL **http://jfod.cnam.fr/csiml/reflect/** se trouve un service web, qui se contente de retourner les paramètres envoyés par le client, paramètres présents dans l'URL

exemples

- http://jfod.cnam.fr/csiml/reflect/?nom=albert&url=http://localhost:8080/

  nom et url représentent ci-dessus 2 paramètres

- http://jfod.cnam.fr/csiml/reflect/?cmd=parler&phrase=bonjour&nom=paul
- http://jfod.cnam.fr/csiml/reflect/?cmd=entendre&phrase=bonjour&nom=bill
- http://jfod.cnam.fr/csiml/reflect/?cmd=participer&url=http://163.173.228.147:8101/chat/&nom=paul
- etc ....



Les paramètres de l'URL sont dans cette table

Reproduisez les mêmes requêtes en Java, en utilisant la méthode **httpUtil.HttpRequest.executeGET,** afficher les résultats obtenus sur la console suffira

un extrait de source possible :

```
Properties params = new Properties();
params.setProperty("nom","paul");
params.setProperty("url","http://163.173.228.147:8101/chat/");
params.setProperty("cmd","participer");
String res = HttpRequest.executeGET("http://jfod.cnam.fr/csiml/reflect
System.out.println("res : " + res);
```

question3_Mise_en_oeuvre) Tests de la classe **question3.httpUtil.ServeurWeb**, en créant un serveur Web, et en vérifiant depuis votre navigateur le bon fonctionnement

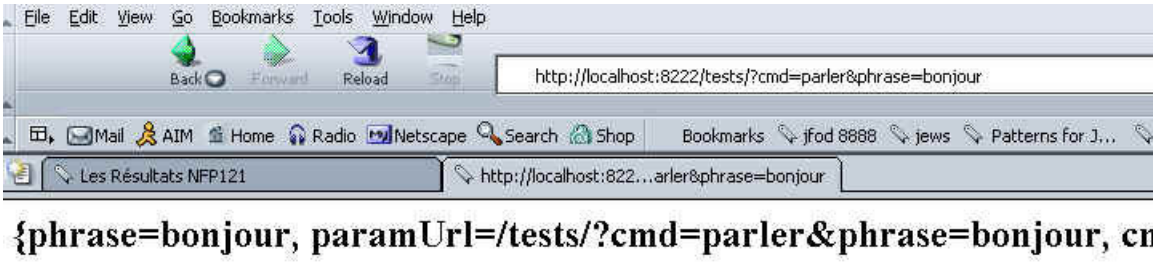
Cette classe utilise l'interface Command, il ne vous reste alors qu'à implémenter la méthode de cette interface
voir question3.httpUtil.HttpRequest.Command

exemple : un serveur Web en 8222, qui se contente de retourner les paramètres de l'URL installés lors de la requête

```
WebServer server = new WebServer( "tests",8222, new Command() {
                                        public String requestToSatisfy(Properties
                                          return "<big>" + parameters + "</big>";
                                        }
                                      } );
```

Depuis votre navigateur, et le serveur actif

Obtenez un résultat analogue à cette copie d'écran



Compléter la classe de tests unitaires "TestDuServeur8222" en utilisant httpunit.

question3_Mise_en_oeuvre) Tester les classes HttpObservable et HttpObserver : une implémentation HTTP du pattern Observateur

un extrait de source minimaliste :

```
Observable model = new HttpObservable("http://localhost:8222") // l'ob

model.addObserver(new HttpObserver("http://www.esiee.fr") // ); // l'e
model.addObserver(new HttpObserver("http://www.cnam.fr"));
String res = model.notifyObservers(Observable.ARGS_EMPTY);
```

Note : les méthodes de ces classes pourront être adaptées afin de répondre aux spécificités induites par le fonctionnement d'un logiciel de causerie (chat...)

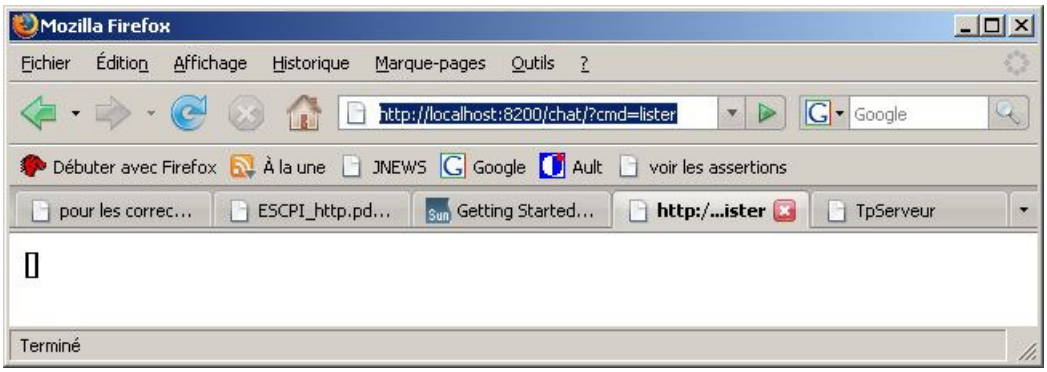
question3_1) Compléter les classes ChatGroup et ChatClient, afin de proposer un chat *simplifié* au protocole HTTP

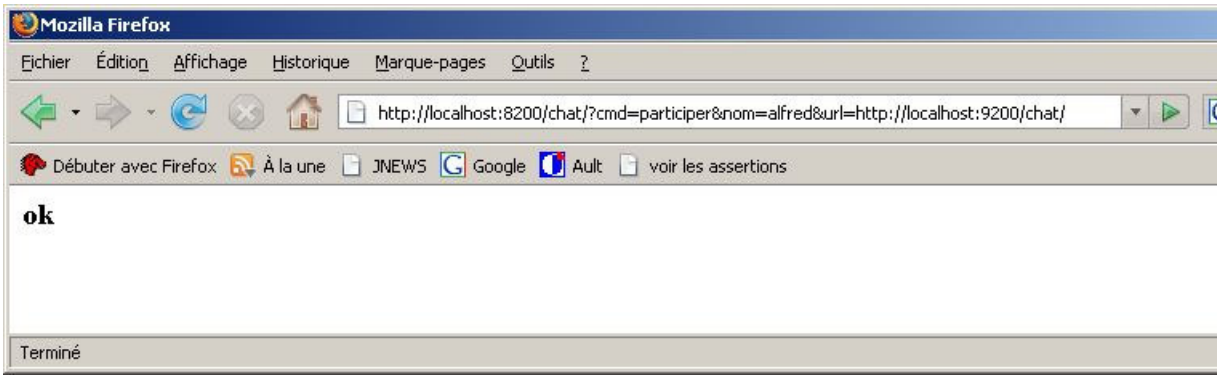
ChatGroup : permettre aux clients du "chat" de participer, parler, chuchoter, quitter et lister.

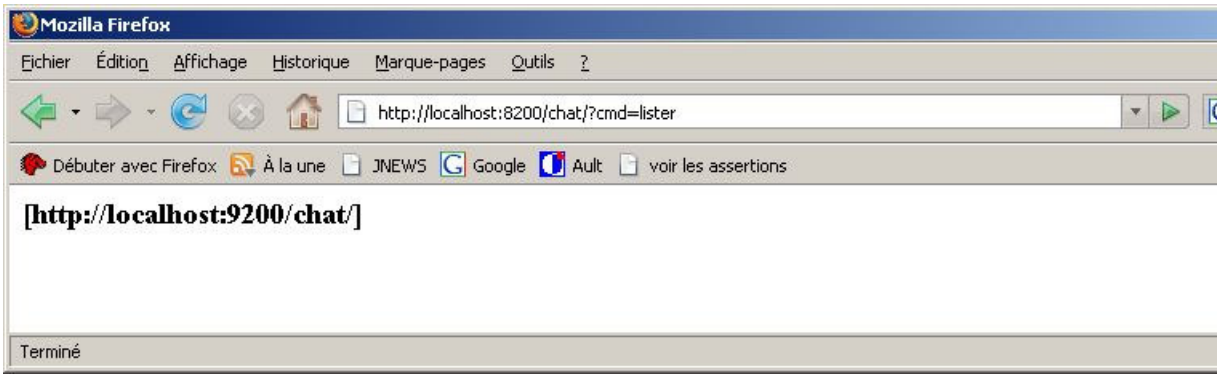
Exemples d'URL

- **http://localhost:8200/chat/?cmd=participer&nom=alfred&url=http://localhost:9200/chat/**
- **http://localhost:8200/chat/?cmd=parler&nom=alfred&phrase=test**
- **http://localhost:8200/chat/?cmd=chuchoter&nom=alfred&phrase=test&dest=paul**
- **http://localhost:8200/chat/?cmd=quitter&nom=alfred&url=http://localhost:9200/chat/**
- **http://site_du_groupe:8100/chat/?cmd=lister**

Un exemple de tests à l'aide de votre navigateur sur cette machine site_du_groupe=localhost







Sur la machine pc5357c, pendant la durée du TP, une instance de ChatGroup sur le port 8111 est installée

http://pc5357c.esiee.fr:8111/chat/?cmd=lister

Vérifiez les différentes commandes possibles et les résultats retournés avant de soumettre avec JNEWS

ChatClient : chaque client inscrit au groupe doit entendre, peut parler au groupe, et également chuchoter à un autre abonné.

- **http://localhost:9100/chat/?cmd=entendre&phrase=bonjour&nom=paul**

Notes :

- **N'oubliez pas de prendre en compte les clients quittant le groupe de discussion inopinément...**
- **Une page HTML pour vos tests, à compléter, pourrait vous être utile**

Développer une classe de tests unitaires de ce Chat.

question3_2) Le serveur ChatGroup doit maintenant être capable de délivrer une applette de type "ChatClient", dont le comportement est analogue à l'application ChatClient. Vouas autoriserez ainsi tout navigateur muni de sa JVM et relié à internet comme client de ce chat....

- **Proposer une applette client de ce chat, classe ChatClientApplette, vous pouvez vous inspirer de la classe question3.httpUtil.WebServerApplet**
- **Certains paramètres de l'URL destinés au groupe sont imposés http://localhost:8200/chat/?cmd=applette&nom=alfred&port=9200**
- **L'ébauche de solution proposée en commentaire (*classe ChatGroup, méthode requestToSatisfy*) consiste à retourner la balise <APPLET *habituelle* ..., ainsi à l'occurrence de cette requête, le navigateur télécharge cette applette et devient de facto un client du chat. L'applette contient un serveur au protocole HTTP, à l'identique de l'application ChatClient.**
- ***remarque : InetAddress.getLocalHost().getHostAddress() ne semble pas fonctionner pour une applette, voir en annexe 2 un extrait du web***

un exemple possible d'exécution, un groupe de discussion, quatre clients

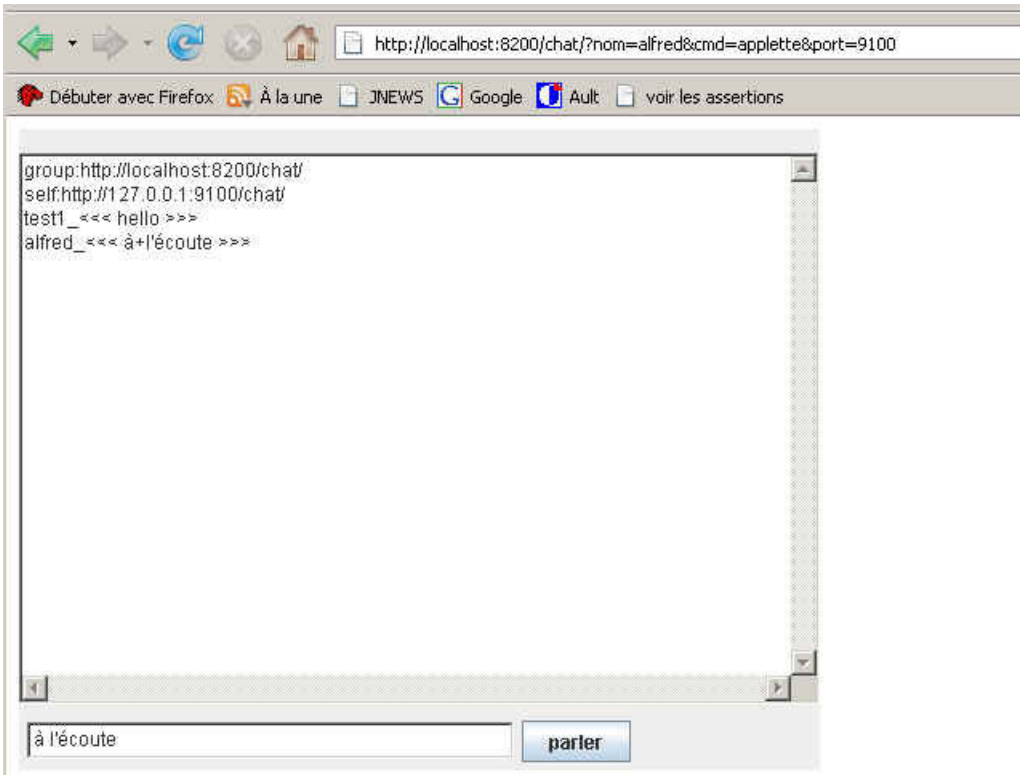
le "ChatGroup"

console>start java -cp . question3.ChatGroup

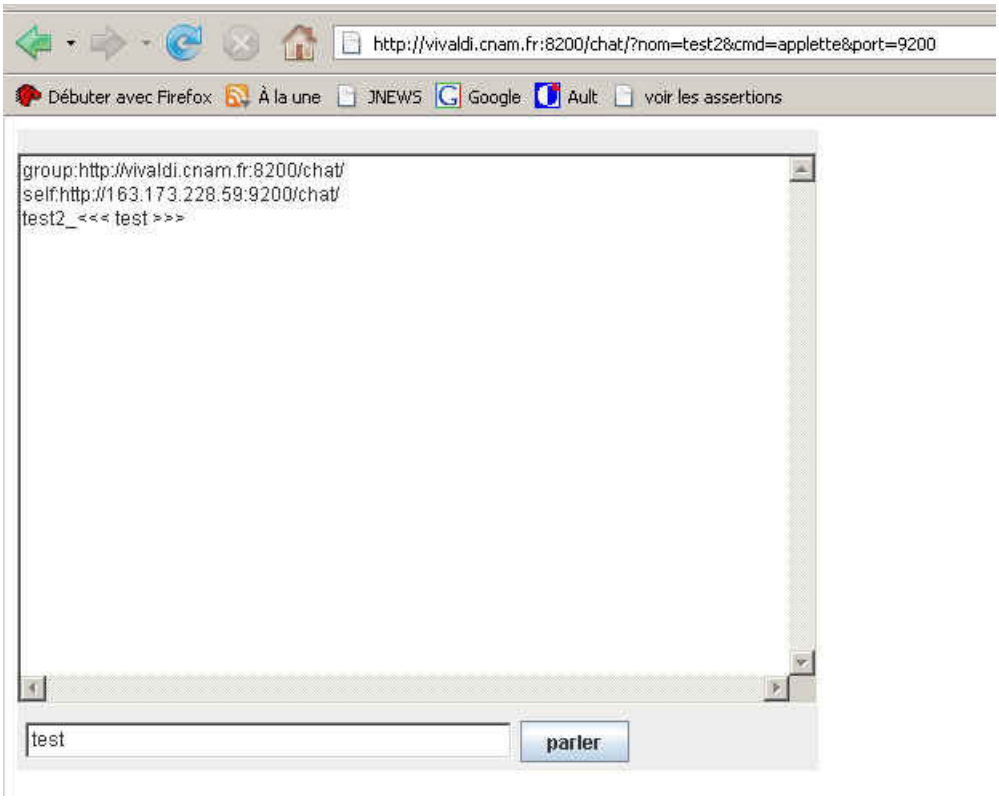
un "ChatClient" nommé test1, comme application

console>**java -cp . question3.ChatClient http://localhost:8200/chat/ test1 9000**

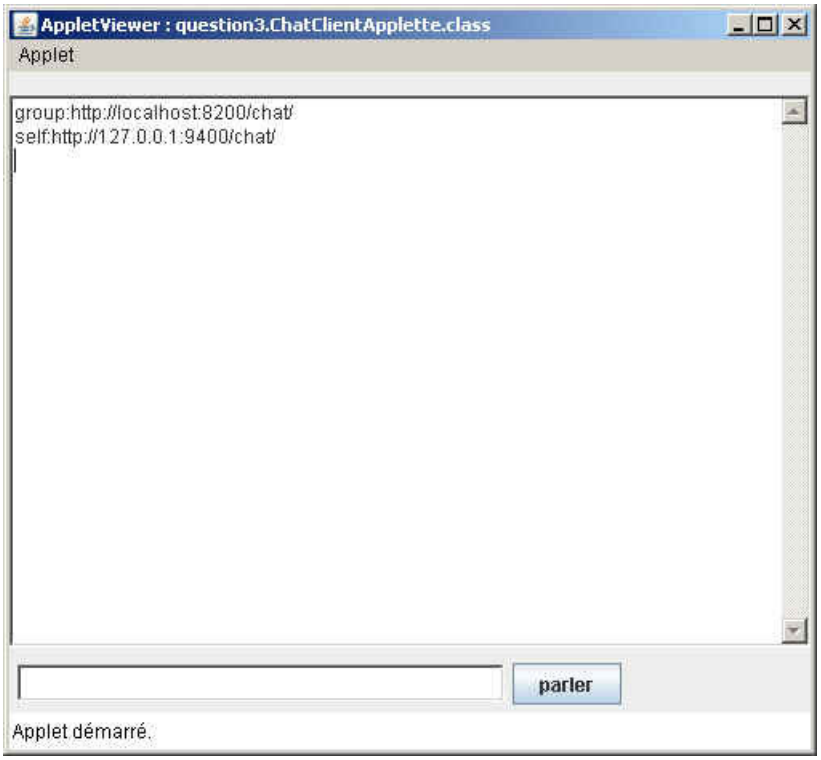
un "ChatClientApplette", depuis votre navigateur



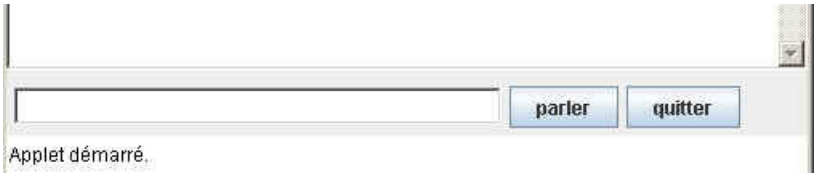
un autre client depuis une autre machine, si les pare-feux le permettent ...



encore un autre client avec appletviewer



n'oubliez la possibilité de quitter l'applette... (en arrêtant le serveur...)



Annexe 1 : Un test unitaire parmi d'autres ...

```
import com.meterware.httpunit.*;

public class TestsChatGroup extends junit.framework.TestCase{


    private ChatGroup chatGroup;

protected void setUp() throws java.lang.Exception{
chatGroup = new ChatGroup(8200);
}


protected void tearDown() // throws java.lang.Exception
{
chatGroup.stop();

}

WebConversation conversation = new WebConversation();
WebRequest request = new GetMethodWebRequest("http://localhost:8200/chat/?cmd=lister" );
WebResponse response = conversation.getResponse( request );

assertTrue(" liste vide : \"<big><b>[]</b></big>\" est attendue !!!, reçue: " +
response.getText(), response.getText().contains("<big><b>[]</b></big>"));

conversation = new WebConversation();
request = new GetMethodWebRequest("http://localhost:8200/chat/?
cmd=participer&nom=alfred&url=http://localhost:9200/chat/" );
response = conversation.getResponse( request );

assertTrue(" commande participer : \"<big><b>ok</b></big>\" est attendue !!!, reçue: " +
response.getText(), response.getText().contains("<big><b>ok</b></big>"));

conversation = new WebConversation();
request = new GetMethodWebRequest("http://localhost:8200/chat/?cmd=lister" );
response = conversation.getResponse( request );

assertTrue(" commande lister : \"<big><b>[http://site_du_client:port/chat/]</b></big>\"
est attendue !!!, reçue: " + response.getText(), response.getText().contains("<big><b>
[http://localhost:9200/chat/]</b></big>"));


}catch(Exception e){
fail("Exception inattendue ? " + e.getMessage());
}
}
```

Annexe 2 : InetAddress.getLocalHost().getAddress() ne semble pas fonctionner pour une Applette ... alors cet extrait du web peut vous être utile

```
/** Returns the IP address that the JVM is running in,
 *   InetAddress.getLocalHost().getHostAddress() ne semble pas fonctionner pour une applette ==> à vérifier
```

```
*
* extrait de http://www.jguru.com/faq/view.jsp?EID=790132
*/
       public String getIPAddress() {
               String strIPAddress = null;
               Socket local = null;
               try {
                 URL origin = this.getCodeBase();
                 local = new Socket(origin.getHost(), origin.getPort());
                 strIPAddress = local.getLocalAddress().getHostAddress();

               } catch (Exception ex) {
                 //ex.printStackTrace();
                 strIPAddress = "";
               } finally {
                 try {
                    local.close();
                 } catch (IOException ioe) {
                    //ioe.printStackTrace();
                 }
               }

               return strIPAddress;
       }
```