ADEDJOUMON Adam
GAMEIRO COSTA Bruce

M1 Informatique
2020 / 2021

ChatOS : Manuel d'utilisation

Présentation

Ce projet est basé sur le protocole **ChatOS** et permet à des clients de discuter entre eux via un serveur.

Il est livré avec deux **`.jar`** exécutables : un pour le serveur et un pour le client. Les deux exécutables sont dans le dossier `/bin`.

Lancement

Pour utiliser correctement le projet, il faut premièrement **démarrer le serveur** en précisant son port d'écoute :

```
$ java --enable-preview -jar bin/ServerChatOS.jar 7777
```

Par la suite, vous pourrez démarrer autant de clients que vous souhaitez en indiquant l'adresse du serveur, son port d'écoute ainsi que l'espace de travail (c'est-à-dire là où il enregistrera les fichiers et où les autres pourront lire ses fichiers) :

```
$ java --enable-preview -jar bin/ClientChatOS.jar [adresse] [port] [dossier]
```

Vous voilà maintenant avec un client connecté au serveur !

Utilisation

Maintenant que vous êtes connecté au serveur, vous devez choisir un **pseudo**. Si ce pseudo est déjà utilisé par quelqu'un d'autre, vous devrez choisir un autre pseudo.

Une fois votre pseudo validé par le serveur les autres utilisateurs recevront un message indiquant votre connexion et vous aurez accès aux actions suivantes pour communiquer avec les autres utilisateurs :

- **`@pseudo message`** : vous permettra d'envoyer un message privé à `pseudo`.
- **`/pseudo ressource`** : vous permettra d'établir une connexion privée avec `pseudo`.

Le client demandé pourra accepter ou refuser la demande.

S'il l'accepte vous recevrez alors la ressource demandée qui sera soit affichée dans le terminal (dans le cas d'un fichier `.txt`) ou sauvegardé dans le dossier indiqué au démarrage.

Tout autre message sera interprété comme un message **général** et sera donc transmit à tout le monde.

NOTE :

Il est possible de **commencer un message par '@' ou '/'** sans qu'il soit interprété comme un message privé ou une connexion privée. Pour cela précéder le symbole d'un '\'. *(backslash)*

ATTENTION

Les fichiers récupéré via une connexion privée sont concaténé avec les fichiers du même nom déjà existant.

Lors de votre déconnexion les autres utilisateurs recevront un message indiquant votre déconnexion.

Test complet

Si vous voulez tester toutes les fonctionnalités proposées, nous vous invitons à suivre les étapes suivantes :

- Ouvrez 4 terminaux à la racine du projet (dans le dossier `/prog_reseau_projet`).
  Par la suite les terminaux seront identifiés par une lettre.

- Dans un premier terminal **A**, lancez le serveur avec la commande :

```
$A> java --enable-preview -jar bin/ServerChatOS.jar 7777
```

- Dans le terminal **B**, lancez un client ayant pour espace de travail le dossier `/bruce` dans `/assets` :

```
$B> java --enable-preview -jar bin/ClientChatOS.jar localhost 7777 assets/bruce
```

- Faites la même chose pour les terminaux **C** et **D** dans les dossiers `assets/adam` et `assets/michel` :

```
$C> java --enable-preview -jar bin/ClientChatOS.jar localhost 7777 assets/adam
```

```
$D> java --enable-preview -jar bin/ClientChatOS.jar localhost 7777 assets/michel
```

- Maintenant que vous avez vos 3 clients connectés, il faut les authentifier. Donc dans le terminal **B** entrez un pseudo (ici ce sera `bruce`), puis dans le terminal **C** entrez le même pseudo.

```
$B> bruce
$C> bruce
```

Vous devriez voir un message d'erreur ainsi que la répétition de la demande de pseudo.

- Dans **C** entrez un pseudo différent de celui pris par le terminal **B** pour vous authentifier (ici ce sera `adam`).

```
$C> adam
```

Vous devriez voir dans le terminal **B** un message indiquant qu'un nouveau client s'est connecté.

- Une fois vos deux clients connectés, essayez d'envoyer des messages généraux et des messages privés.

```
$B> Bonjour
$C> Bonjour :)
$C> @bruce Comment allez-vous ?
$B> @adam Bien et vous?
$C> @bruce Ma foi je n'ai pas à me plaindre.
```

- Essayez maintenant d'envoyer un message privé à un utilisateur non connecté ou à vous même :

```
$B> @bruce Moi-même.
$B> @inconnu Je ne vous connais pas.
```

- Une fois ces tests finis, vous pouvez essayer d'établir des connexions privées. Premièrement en la refusant :

```
$B> /adam hello.txt
$C> n
```

Le client **B** devrait recevoir un message d'erreur indiquant que la connexion a été refusée.
Puis en l'acceptant :

```
$B> /adam hello.txt
$C> y
```

Vous devriez voir chez les deux clients un token apparaitre. Et le client **B** recevra en plus le contenu du fichier `hello.txt`.

- Maintenant qu'une connexion est établie entre les deux clients vous pouvez authentifier le dernier client (il n'est pas nécessaire d'attendre maintenant pour l'authentifier, si vous l'avez fait avant ça ne changera rien) :

```
$D> michel
```

- Essayez maintenant connecter le client **D** aux deux autres clients. Tout comme pour la première connexion privée établie, vous recevrez un token pour chaque nouvelle connexion (donc 2 par client) :

```
$C> /michel hello.txt
$D> y
$D> /bruce 1Mio.dat
$B> y
```

- Rien ne devrait s'afficher (autre que le token) pour la dernière requête. C'est normal le fichier étant un `.dat` et non un `.txt`, il a été sauvegardé dans l'espace de travail (`assets/michel`).

- Vous pouvez tester de demander plusieurs ressources sur la même connexion pour vérifier qu'elle n'est bien faite qu'une seule fois :

```
$D> /bruce hello.txt
$B> /michel hello.txt
```

- Essayez aussi de demander des fichiers qui n'existent pas pour voir le message d'erreur :

```
$B> /adam unknown.dat
```

- Une fois tous vos tests finis, fermez un des clients. Vous devriez voir chez les autres un message de déconnexion ainsi qu'un message indiquant la fermeture de la connexion privée.

- Maintenant fermez le serveur pour voir que les clients s'arrêtent correctement (sans erreur).

- Vous pouvez aussi relancer le serveur avec un client et fermer le client avant l'authentification pour voir que le serveur fonctionne toujours.