Design Pattern
TP Bonus : Synchronisation

Le but de cet exercice est de présenter un mécanisme simple de synchronisation entre le client et le serveur.

Synchronisation

Polling

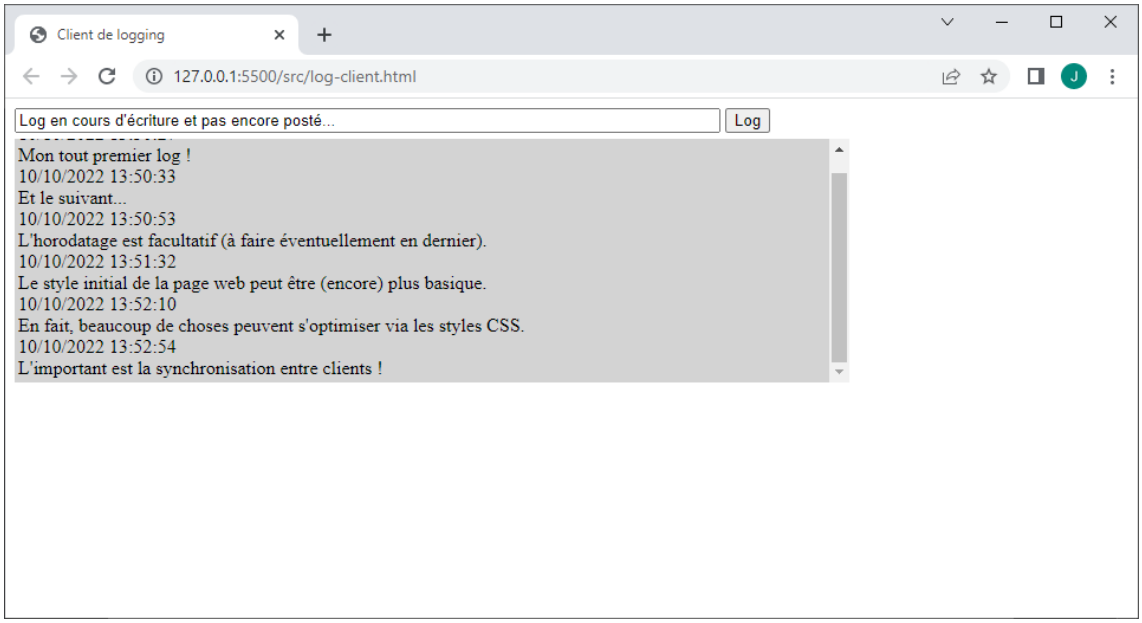
Une première façon de synchroniser le client avec les données du serveur est de réaliser des requêtes `GET` à intervalle régulier. Cette manière de faire a plusieurs inconvénients : les données ne sont pas récupérées en temps réel et le nombre de requêtes est important.

Long polling

Cette variante permet de supprimer les inconvénients de la méthode précédente. Le client envoie toujours une requête `GET` mais le serveur ne lui répond que lorsque c'est nécessaire. Concrètement, la requête contient un identifiant de version. Si le serveur a une version différente des données, il les envoie de suite. Dans le cas contraire, il met la requête en attente d'une modification avant de répondre. Il répond également au bout d'un certain délai même s'il n'y a pas eu de modification. De son côté, le client renverra une nouvelle requête dès qu'il a fini de traiter la réponse à la précédente.

Serveur de log

On cherche à développer une application Web où des clients peuvent laisser des messages de log au serveur. Tous les clients doivent pouvoir afficher tous les logs en temps réel. L'interface est simple.



Aide technique

Le client

Il devra connaître la version des données qu'il affiche actuellement.

Lorsqu'un log est posté, il envoie une requête `PUT`.

Lorsqu'il envoie une requête `GET` au serveur, il indique qu'il est prêt à attendre un certain temps qu'une mise à jour ait lieu. Il doit donc aussi indiquer sa version actuelle des données. Il faut pour cela utiliser les entêtes `If-None-Match` et `Prefer`. Par exemple, si le client est à la version `version` des données et est prêt à attendre `time` secondes, l'entête de la requête devra contenir :

```
If-None-Match: version
Prefer: wait=time
```

Lorqu'il reçoit une réponse à une requête `GET`, si son statut est :

200	Il met à jour son affichage avec les données reçues, sauvegarde le nouvel identifiant de version et relance une requête.
304	Il relance immédiatement une requête. Ce code indique qu'il n'y a pas eu de mise à jour durant le laps de temps prévu pour traiter la requête.
autre	C'est probablement un cas d'erreur à gérer.

Le serveur

Il dispose d'une version des données qu'il peut comparer avec celle du client.

Lorqu'il reçoit une requête `PUT`, il en stocke le contenu et met à jour son identifiant de version.

Lorqu'il reçoit une requête `GET`,

— Si aucune version n'est indiquée ou que la version ne correspond pas, il renvoie immédiatement les données courantes.

— Si la version est présente et correspond, il stocke la demande jusqu'à ce que le temps d'attente soit dépassé (prendre une valeur par défaut si non spécifié) ou qu'une modification ait lieu. Dans le premier cas de figure, il renvoie les données actualisées. Dans le second, une réponse sans corps avec le statut 304 (absence de changement) est envoyé.

L'identifiant actuel des données est transmis dans l'entête `ETag` de la réponse.