Protocoles Internet
TP 3 : JSON

Juliusz Chroboczek

16 octobre 2022

Exercice 1. Avant de commencer, assurez-vous qu'il y a au moins quelques messages dans le *chat*.

1. À l'aide de la commande `curl -i`, examinez le contenu de l'URL `/chat/messages.json`.
2. Écrivez un programme qui affiche la liste des messages du *chat* sous un format lisible par un être humain en utilisant les données de l'URL `/chat/messages.json`. Vous pourrez vous servir du paquet `json` et de la structure suivante :

```
type jsonMessage struct {
    Id   string `json:"id"`
    Time int64  `json:"time"`
    Body string `json:"body"`
}
```

3. Combien de RTT votre programme attend-il les données ? Comparez avec la version REST du TP précédent.
4. Imaginez que le serveur soit modifié pour que les messages contiennent un champ supplémentaire, par exemple un champ nommé `from` qui contient le nom de l'utilisateur qui a posté le message. Votre programme continuera-t-il à fonctionner ?

Exercice 2.

1. À l'aide de la commande `curl -i`, examinez le contenu de l'URL `/chat/messages.json?count=4`
2. Modifiez votre programme pour qu'il affiche les derniers 4 messages au plus.
3. Modifiez votre programme pour qu'il vérifie toutes les 5 secondes si la liste de messages a changé et affiche les derniers 4 messages si c'est le cas. Servez-vous de l'entête de réponse `ETag`.
4. Optimisez votre programme en vous servant de l'entête de requête `If-None-Match`.
5. Le `ETag` retourné par mon serveur ne dépend pas de la valeur du paramètre `count`. Est-ce correct ?
6. Comment faire pour que votre programme n'affiche que les messages ajoutés depuis la dernière consultation ? Comparez avec l'approche REST.

Exercice 3 (S'il vous reste du temps)**.** Écrivez un client de *chat* complet. Faites mieux que *Telegram* et *Signal*. Devenez millionaire. Partagez avec votre enseignant.