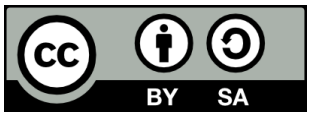
Les WebSockets



Contenu

- **Les WebSoquoi?**
- **Avant les WebSockets**
- **Créer un serveur WebSocket en PHP**
- **Utiliser WebSocket côté client en JavaScript**

Les WebSoquoi?

Les WebSoquoi?

- **WebSocket est un standard du Web comprenant:**
  - Un protocole réseau de la couche application, implanté par-dessus TCP
  - Une interface de programmation disponible dans les navigateurs (accessible en JavaScript)
- **Contrairement au protocole HTTP, WebSocket permet les communications bidirectionnelles entre un client et un serveur**

Pourquoi WebSocket?

- **Lorsqu'on développe une application Web interactive, il peut être utile de permettre au serveur de pousser (*push*) des données directement au client**
  - Ex: dans une application de clavardage, transmettre les nouveaux messages au client dès qu'ils arrivent
- **Le protocole HTTP n'offre pas cette possibilité directement**
  - HTTP est un protocole de type « requête-réponse »

Sécurité

- **Comme HTTP avec HTTPS, le protocole WebSocket peut être sécurisé**
- **L'URL d'une connexion WebSocket commence par:**
  - « ws:// » si la connexion n'est pas sécurisée
  - « wss:// » si elle est sécurisée

Avant les WebSockets

Avant les WebSockets

- **Les applications Web interactives sont apparues avant WebSocket**
- **Deux techniques permettent d'utiliser le protocole HTTP pour donner l'illusion d'un comportement semblable à WebSocket**
  - Le *polling*
  - Le *long polling*

Le *polling*

- **En *polling*, le client envoie constamment des requêtes au serveur pour vérifier l'arrivée de nouvelles données**
  - Ex: Appeler « GET api/messages » toutes les secondes

Le *long polling*

- **En *polling*, le client envoie une requête et le serveur lui répond immédiatement (utilisation normale du protocole HTTP)**
- **En *long polling*, on « détourne » l'utilisation normale du protocole HTTP pour donner l'illusion d'une transmission en mode « push »**

Le *long polling*

- Le client effectue une requête au serveur
- **Si des nouvelles données sont déjà disponibles:**
  - Le serveur répond à la requête immédiatement
  - Le client effectue une nouvelle requête
- **Sinon:**
  - Le serveur attend que de nouvelles données soient disponibles avant de répondre
  - La requête demeure ouverte jusqu'à la réception d'une réponse par le client

Le *long polling*

- **Le *long polling*, c'est un peu comme effectuer un appel téléphonique, puis être placé en attente!**
  - On fait l'appel, mais on n'a pas une réponse immédiatement!

Photo par James Sutton sur Unsplash

Créer un serveur WebSocket en PHP

Créer un serveur WebSocket en PHP

- **PHP ne supporte pas WebSocket directement**
- **Des librairies externes permettent d'ajouter ce support**
- **La librairie *Ratchet* est la plus populaire**
  - Documentation et exemple ici: https://github.com/ratchetphp/Ratchet

Principes côté serveur

- **Peu importe le langage et la librairie utilisés côté serveur, le fonctionnement sera sensiblement le même:**
  - On maintient une liste de clients connectés;
  - On fournit une fonction/méthode qui sera appelée lorsqu'un nouveau client se connecte, et une autre qui sera appelée lorsqu'un message est reçu;
  - Une fonction/méthode nous est fournie pour envoyer des messages à des clients connectés.

Utiliser WebSocket côté client en JavaScript

Utiliser WebSocket côté client en JavaScript

```
const socket = new WebSocket("wss://example.org/websocket");


socket.onopen = () => {
    console.log("Connexion établie");
    console.log("Envoi d’un message au serveur");
    socket.send("Bonjour serveur!");
}


socket.onmessage = (event) => console.log(event.data);
socket.onclose = () => console.log("Connexion interrompue");
socket.onerror = (error) => console.log(`Erreur: ${error.message}`);
```

Fin de la présentation

Des questions?

Image par GrammarGirl (CC BY-NC 2.0)