TP 12: Envoi de données avec AJAX

Objectifs:

- Envoyer des informations (ex: un tweet) à une API du Web
- Effectuer une requête `POST` avec `XMLHttpRequest`

Slides du TP

Effectuer une requête `POST` avec `XMLHttpRequest`

Dans la partie précédente, nous avons utilisé la classe `XMLHttpRequest` pour envoyer des requêtes de type `HTTP GET`. Pour rappel, ce type de requête permet de récupérer des informations depuis un serveur.

Dans cette partie, nous allons voir comment utiliser cette même classe pour envoyer des requêtes de type `HTTP POST`. Même s'il permet aussi de recevoir une réponse du serveur à ces requêtes, ce type de requête permet d'**envoyer** des informations.

Envoi d'une chaine de caractères

Voici un exemple de requête `POST` effectuée en JavaScript/AJAX:

```
var xhr = new XMLHttpRequest();
xhr.open('POST', 'https://httpbin.org/post');
xhr.onreadystatechange = function() {
  if (xhr.readyState === 4) {
    alert(xhr.responseText);
  }
};
// envoi d'une chaine de caractères:
xhr.send('ceci est un exemple de données envoyées');
```

Le principe de fonctionnement est exactement le même que celui d'une requête `GET`, sauf que nous avons cette fois-ci **envoyé une chaine de caractères** via notre requête à l'adresse serveur `https://httpbin.org/post`.

Pour cela, nous avons:

- remplacé le paramètre `GET` par `POST`, dans l'appel à la méthode `open()`,
- et passé une chaine de caractères en paramètre de l'appel à la méthode `send()`.

Envoi d'un objet JavaScript / JSON

Pour envoyer un objet JavaScript / JSON dans une requête `POST`, il faut d'abord sérialiser l'objet (c'est à dire: le convertir) sous forme de chaine de caractères, à l'aide de la fonction `JSON.stringify()`.

Il suffit donc de modifier le paramètre passé à la méthode `send()`, tel que dans l'exemple suivant:

```
// ... ou envoi d'un objet JSON:
xhr.send(JSON.stringify({ message: 'bonjour!' }));
```

En effet, `JSON.stringify()` est la fonction inverse de `JSON.parse()`:

- alors que `JSON.parse()` permet de convertir une chaine de caractères en objet,
- `JSON.stringify()` permet de convertir un objet en chaine de caractères.

> **Attention**: veillez à ne pas appeler la méthode `send()` d'une même instance (ex: `xhr`, dans notre exemple) plus d'une seule fois !

Conseils pratiques pour diagnostiquer le fonctionnement de vos requêtes

- Comme d'habitude, n'oubliez pas de consulter la console de votre navigateur pour vérifier la présence éventuelle d'erreurs dans votre code et/ou dans vos requêtes AJAX. (ex: accès non autorisé à une API)
- Utilisez l'onglet "Réseau" (ou *Network*, en anglais) de Chrome Dev Tools pour suivre l'exécution de vos requêtes, et consulter à la fois leur contenu et celui de la réponse du serveur.
- Utilisez l'API `https://httpbin.org/post` pour tester le bon fonctionnement de vos requêtes POST. Cette API vous envoie en réponse le contenu que le serveur a reçu suite à votre requête.

Exercice 4: *Tweeter* en AJAX

Un serveur est mis à votre disposition à l'URL https://js-ajax-twitter.herokuapp.com. Comme Twitter, l'application Web exécutée sur ce serveur permet aux utilisateurs de publier des messages publics en temps réel.

Le but de cet exercice est de publier vos messages en utilisant une requête AJAX.

Sont fournis:

- une page web affichant en temps réel le *flux* des derniers messages publiés sur le serveur;
- une API HTTP permettant de publier des messages sur le serveur, accessible depuis l'adresse `/tweet`;
- et un client simple permettant de publier des messages à l'aide d'un formulaire HTML.

Documentation de l'API fournie

Pour publier un message sur ce serveur, il faut envoyer une requête `HTTP POST` à l'adresse `/tweet` (accessible depuis la racine du serveur), en transmettant un objet JSON.

L'objet JSON à envoyer comme contenu de la requête doit contenir deux propriétés:

- `message`: le texte à publier. (type: `string`)

- `token` : le jeton fourni après identification de l'utilisateur. (type: `string`)

Note: La valeur de la propriété `token` de l'objet à envoyer est générée automatiquement par le *client simple* fourni, dans la variable globale `window.token`.

À chaque requête HTTP valide reçue, le serveur répondra par un objet JSON ayant:

- soit une propriété `error` contenant un message d'erreur (type: `string`);
- soit une propriété `ok`, dans le cas où le message a été publié sans erreur.

> Remarque importante: le bouton d'identification de Google n'a été activé que depuis les domaines `jsbin.com`, `codepen.io` et `js-ajax-twitter.herokuapp.com`. Vous ne pourrez donc pas exécuter le *client simple* fourni (ni une version dérivée) depuis un autre domaine, ni depuis le système de fichiers de votre machine (protocole `file://`).

Étapes proposées

1. Tester le client fourni: https://js-ajax-twitter.herokuapp.com/client.html (après login)
2. Cloner/forker le client fourni: https://jsbin.com/bucilir/edit?html,js,output (ou codepen: https://codepen.io/anon/pen/GxBgYg)
3. Remplacer le formulaire par l'envoi d'une requête HTTP POST à chaque fois que l'utilisateur pressera `ENTRÉE` dans le champ de saisie (ou clic sur un bouton).
4. Dans le cas où la publication du message a fonctionné sans erreur, effacer le champ de saisie, afin de permettre la saisie immédiate d'un nouveau message.
5. Dans le cas contraire, afficher une description de l'erreur dans un `alert()`.

> Conseil: Pensez à utiliser la console JavaScript et l'onglet "Réseau" de Chrome Dev Tools pour diagnostiquer.

Bonus: améliorations

Quand vous aurez terminé toutes les étapes proposées ci-dessus, vous pourrez apporter les améliorations suivantes à votre client:

- intégrer le flux de tous les messages sur votre page,
- améliorer le design et l'expérience utilisateur de votre client.

<!--

> Solution: jsbin
> <! -- ajax-post-twitter.html. -- >
> -->