R4.A.10 Complément Web Partie 3

Passage au WebService

Pour le moment, la source de données que vous utilisez est un fichier **`JSON`**, **`countries.json`**, que vous avez :

- soit chargé par code et affecté à une variable,
- soit modifié en faisant l'affectation de la variable directement dans le fichier (sans doute aussi renommé en **`.js`**) et que vous chargez dans votre page HTML, juste avant le chargement de votre fichier de script **`JS`**.

Cela dit, peu importe la méthode utilisée, vous allez, dans cette troisième partie, remplacer ce mécanisme par un appel à un *WebService*, en **`AJAX`**.

Sur la base de **`countries_v5.html`**, créez une page **`countries_final.html`** dans laquelle vous apporterez cette modification.

L'URL du WebService est : **`https://restcountries.com/v2/all`**

C'est tout…

Problème de CORS ?

Il est fort probable que vous rencontriez des soucis en essayant d'accéder à l'API **`restcountries.com`** depuis un script **`JS`** provenant de votre serveur Web. Voici une explication du problème et une solution.

Quand, dans votre navigateur, vous appelez directement une URL sur **`restcountries.com`** (ou n'importe quel autre site d'ailleurs), vous êtes à l'initiative de cette demande, c'est vous qui avez tapé l'URL dans votre navigateur.

Si maintenant vous chargez une page Web qui contient un lien vers un script **`JS`** qui, lui-même, contient du code chargeant une URL sur **`restcountries.com`**, cette fois-ci ce n'est pas un chargement à votre initiative, enfin, pas directement en tout cas. Vous êtes à l'initiative du chargement de la page Web, mais pas des scripts que cette page peut contenir (vous connaissez tous les scripts chargés par la page **`index`** de Google ?). Potentiellement, et c'est bien le cas dans ce projet, ces scripts peuvent donc contenir du code qui va charger d'autres URLs, mais cela va être vu par le serveur cible (**`restcountries.com`** dans notre exemple) comme des requêtes faites par votre navigateur, donc par vous, et qui semblent donc faites… à votre initiative ! Ce qui n'est

pas vrai, vous l'avez compris, ce n'est pas la même chose que de saisir ces URLs vous-mêmes dans le navigateur.

Tous les navigateurs récents intègrent un mécanisme qui vérifie cette situation.

Quand cette situation survient, le navigateur ne prend pas la décision tout seul. Il commence par "questionner" le serveur cible (**`restcountries.com`** dans notre cas) pour savoir si ce serveur est d'accord qu'une requête puisse lui être envoyée de façon indirecte, c'est-à-dire sans être à l'initiative directe de l'utilisateur. Seul le serveur cible sait si l'information qu'il va délivrer dans sa réponse revêt un caractère confidentiel ou sécuritaire par exemple ou si l'information est totalement publique. Le serveur cible peut alors répondre qu'il accepte de telles requêtes ou qu'il les refuse. Le navigateur utilise la réponse de cette pré-requête pour savoir s'il peut finalement faire la requête qui récupérera les données ou s'il renvoie une erreur au script qui a émis la requête.

Ce mécanisme s'appelle **`CORS`**, pour ***C**ross **O**rigin **R**esource **S**haring*. En cas d'erreur, vous obtiendrez alors un message du style *XMLHttpRequest cannot load XXX due to access control checks* ou encore *Blocage d'une requête multi-origine*

Dans votre projet, vous ne pouvez évidemment pas agir sur l'autorisation ou le refus des requêtes **`CORS`** sur **`restcountries.com`**, seul le propriétaire du site peut le faire.

Comment régler cette situation ?

Une solution peut être de désactiver le mécanisme **`CORS`** dans votre navigateur. Cette solution est valable pour tester rapidement en mode développement, mais ça reste un pansement sur une jambe de bois ! On ne peut pas demander aux utilisateurs de procéder ainsi, et puis cette désactivation est une mauvaise idée car vous risquez de l'oublier et de la laisser définitivement, or la sécurité **`CORS`** a sa raison d'être ! Donc, sauf pour un petit test très court, on oublie cette idée.

La solution qu'on vous propose de mettre en place est de créer un script (en **`PHP`**) qui va servir de passerelle vers **`restcountries.com`**. Le mécanisme **`CORS`** est intégré aux navigateurs, mais un script **`PHP`** sur un serveur Web n'a pas cette contrainte. De plus, si votre script **`JS`** (par un appel **`AJAX`**) charge une URL sur le même domaine que celui d'où a été chargé le script **`JS`**, pas de **`CORS`**, pas de *Cross Origin*, c'est un *Same Origin* dans ce cas, donc pas d'interdiction.

Créez donc un script **`PHP`** (nommé **`gateway_to_restcountries.php`** par exemple), avec ce contenu, très simple :

```php
<?php
    header('Content-Type: application/json');
    echo file_get_contents('https://restcountries.com/v2/all');
?>
```

Maintenant, dans l'appel **`ajax`**, changez l'URL **`https://restcountries.com/v2/all`** par **`http://localhost:6789/gateway_to_restcountries.php`**

Exécutez un mini serveur Web local en lançant, depuis le dossier contenant votre script, un **`php -S localhost:6789`** par exemple.

Enfin, n'ouvrez plus votre script par l'explorateur de fichier mais utilisez maintenant l'URL **`http://localhost:6789/countries_final.html`**

Notez que, puisque le script **`gateway_to_restcountries.php`** est appelé par la page **`countries_final.html`**, et qu'ils sont tous deux sur la même source "serveur", vous pouvez simplement omettre le domaine **`http://localhost:6789`** dans l'appel **`AJAX`** et utiliser une URL avec uniquement **`/gateway_to_restcountries.php`**. Le navigateur saura compléter l'URL en préfixant avec le même domaine que celui d'où est venue la page Web.