Haute École Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

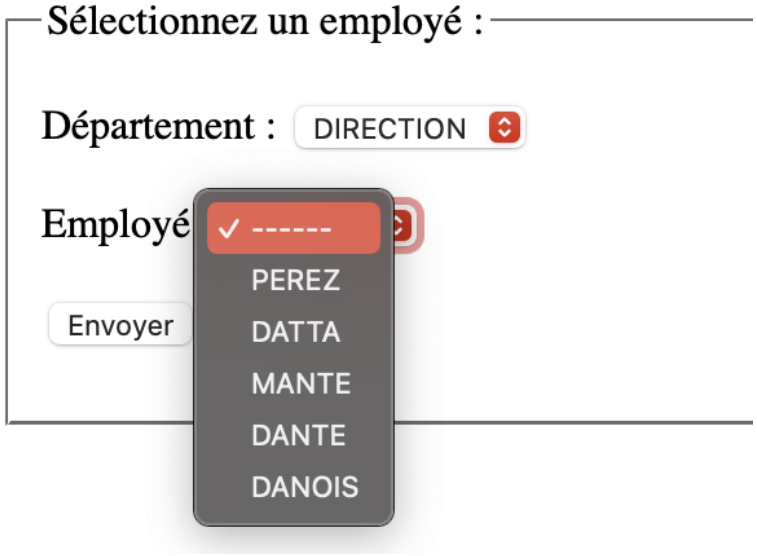
2020 - 2021
WEBR4-DIN

Développement Internet

AJAX

AJAX, nom donné à la mise en œuvre d'un ensemble de techniques, permet de dynamiser les pages html.

Avec une page permettant de sélectionner un employé comme ci-dessous :



Dans un site classique, à chaque fois que l'utilisateur choisit un autre département, une requête doit être faite au serveur afin de peupler la liste des employés avec ceux qui correspondent au département demandé. C'est peu efficace, surtout si l'utilisateur revient en arrière et doit dès lors demander une nouvelle liste.

Avec **AJAX**, nous pourrons lancer une requête unique au serveur demandant l'ensemble des employés du département sélectionné et les placer dans la liste déroulante des employés. Généralement la liste des employés nous sera retournée sous la forme d'un document XML ou JSON.

1 AJAX

AJAX signifie **Asynchronous JavaScript And Xml**. Il ne s'agit pas d'un langage ou d'une librairie, mais d'une démarche permettant de diminuer certaines limitations imposées aux applications Web par HTTP et HTML.

La lacune essentielle des applications Web était leur manque d'interactivité : toute modification de la page affichée par le navigateur imposait un ré-affichage complet.

Version du document : 2 mars 2021

Nous avons déjà aperçu que JavaScript permet d'assouplir ce fait, mais les actions apparaissant dans le navigateur sont réalisées sans aucun contact avec le serveur. Si, par exemple, nous voulons afficher des informations à la demande de l'utilisateur, soit l'ensemble des informations doit être chargé initialement (transfert prohibitif [cf. exemple ci-dessus : le client devrait recevoir tous les départements et tous les employés dès le premier chargement de la page !]) soit la page complète doit être rechargée à chaque action !

AJAX va nous permettre de nous libérer de ces contraintes. Attention, à nouveau, comme pour JavaScript, il s'agira de ne pas abuser de cette démarche pour ne pas rendre nos pages trop complexes à concevoir et surtout à maintenir. En particulier, **la logique métier doit rester sur le serveur !**

En reprenant l'acronyme complet, les caractéristiques d'AJAX sont :

1. Il est **Asynchrone** : Le client va pouvoir parler au serveur en mode asynchrone. Il demande des informations au serveur et continue son travail. Lorsque la réponse du serveur est prête, un code précédemment spécifié est appelé pour la traiter.
2. Il utilise **JavaScript** : Du côté client, c'est généralement ce langage qui est utilisé. JavaScript a longtemps été décrié car peu compatible d'un navigateur à l'autre, créant des problèmes de sécurité et finalement essentiellement utilisé pour de futiles effets graphiques. À tel point que les navigateurs permettent d'empêcher l'utilisation de JavaScript. Depuis, la compatibilité s'est accrue et, surtout, on s'est rendu compte, qu'au-delà des gadgets, il pouvait apporter un réel confort d'utilisation.
3. Il utilise aussi **XML**, en tant que format utilisé par le serveur pour transmettre des informations au client. C'est parfois également fait avec HTML (rarement, car limité) ou JSON (que nous reverrons au chapitre suivant).

2 Mise en œuvre

Nous ne réaliserons réellement ceci que plus tard quand nous programmerons une application web, car les exemples ci-dessus sont prévus pour le protocole *http(s)* et pas pour le protocole *file*.

2.1 En JavaScript

L'objet JavaScript `XMLHttpRequest` [1] permet d'exécuter des requêtes HTTP utilisées par Ajax :

```
http=new XMLHttpRequest();
```

Il faut noter qu'anciennement, chaque navigateur utilisait une syntaxe différente, et il fallait procéder via un `try catch` de JavaScript pour essayer les différentes syntaxes existantes. C'est toutefois du passé.

1. https://www.w3schools.com/xml/xml_http.asp

Une fois l'objet créé, il peut être utilisé pour envoyer des requêtes à un serveur, à l'aide de plusieurs méthodes et propriétés :

1. `http.open("method","url")` prépare l'envoi d'une requête au serveur renseigné par l'URL mentionnée, avec la méthode renseignée (par exemple `"GET"` ou `"POST"`)
2. `http.responseType` précise le format dans lequel le serveur va répondre ; l'objet se prépare ainsi à recevoir et formater cette réponse.
3. `http.onreadystatechange` [2] est une propriété dans laquelle on peut stocker une fonction, qui sera appelée lorsque la requête a été envoyée et une réponse a été reçue.
4. `http.send()` envoie la requête au serveur.

Pour plus d'informations, un tutoriel est disponbible sur ce site.

Un exemple de code complet d'envoi d'une requête serait par exemple :

```
http = new XMLHttpRequest();
// fct appelée quand la réponse est prête
http.onreadystatechange = stateChanged;
// Prépare une requête GET asynchrone pour l'URL indiquée
http.open("GET", "url");
http.send();    // L'envoi au serveur

function stateChanged() {
    if (http.readyState == XMLHttpRequest.DONE) {
        // La réponse est revenue du serveur
        if (http.status == 200) {// Tout s'est bien passé
            http.responseText;          // contient la réponse
            http.responseXML        // pour récupérer du XML
        } else {
            // Gestion de l'erreur
        }
    }
}
```

Cet envoi de requête est bien asynchrone :

1. Pn spécifie les traitements à réaliser lorsque la réponse sera prête, en associant une fonction à la survenue de l'événement `onreadystatechange`. La classe XMLHttpRequest fournit également d'autres gestionnaires d'événements.
2. On prépare la requête avec la méthode `open()`, puis on l'envoie au serveur avec `send()`.
3. Quand la réponse est prête, cela déclenche l'exécution de la méthode spécifiée.

Si vous souhaitez tester ces méthodes sur des fichier locaux, vous pouvez utiliser le protocole *file* ; toutefois, notez que le code de réponse est alors 0 et non 200 comme pour une requête HTTP qui s'est déroulée avec succès.

2. Les valeurs que l'état peut prendre sont renseignées ici : https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/readyState

2.2 Avec jQuery

jQuery introduit une fonction `$.ajax()` [3] qui permet de définir un appel AJAX avec les paramètres suivants, passés via un objet JavaScript :

1. `url` : le document ciblé
2. `type` : la méthode d'appel
3. `data` : des données qui accompagnent la requête (par exemple via un formulaire)
4. `success` : la fonction JavaScript à exécuter en cas de réussite
5. `error` : la fonction JavaScript à exécuter en cas d'échec
6. `complete` : la fonction JavaScript à exécuter quand tout est terminé

Par exemple (pris du tutoriel de W3Schools) :

```
$("button").click(function () {
    $.ajax({
        url: "demo_test.txt", success: function (result) {
            $("#div1").html(result);
        }
    });
});
```

Pour des requêtes plus standard, des méthodes prédéfinies existent également :

1. `$(selector).load(url)` [4] charge le contenu de la ressource à l'URL donnée dans l'élément sélectionné.
2. `$.get(url[, data, function, dataType])` [5] envoie une requête avec la méthode GET à l'URL donnée.
3. `$.getJSON(url[, data, function])` [6] envoie une requête avec la méthode GET et attend des données spécifiquement au format JSON.
4. `$.post(url[, data, function, dataType])` [7] envoie une requête avec la méthode POST à l'URL donnée.

3. https://www.w3schools.com/jquery/ajax_ajax.asp
4. https://www.w3schools.com/jquery/ajax_load.asp
5. https://www.w3schools.com/jquery/ajax_get.asp
6. https://www.w3schools.com/jquery/ajax_getjson.asp
7. https://www.w3schools.com/jquery/ajax_post.asp

3 Avantages et Inconvénients

AJAX permet de fluidifier les échanges navigateur-serveur et de rendre l'interface utilisateur moins rigide (client « riche »). Il faut toutefois rester attentif à certains points :

1. Les robots ne référenceront pas les données obtenues au travers d'AJAX.
2. Attention à préserver la fonctionnalité du bouton 'Back' du navigateur (des solutions toutes faites existent)
3. Il faut assurer la sécurité des requêtes via AJAX aussi sérieusement que les requêtes « classiques ».
4. La notion de lien perd de son sens : fournir un lien à quelqu'un donne accès à la page mais pas nécessairement à l'info que l'on désire faire visualiser.

Remarque : Pour des raisons de sécurité, la plupart des navigateurs empêchent d'accéder, par le biais d'AJAX, à une ressource d'un autre serveur que celui qui a fourni la page courante. Ceci peut parfois être modifié, à vos risques et périls, dans les paramètres du navigateur.