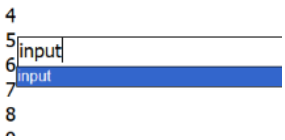
TD1 RIA

HTML 5, jQuery : Auto-Complete

PAUL-ANTOINE BISGAMBIGLIA
bisgambiglia@univ-corse.fr



L'objectif de ce TD est de réaliser un ensemble de pages pour simuler un code de type auto complete. Vous allez devoir utiliser HTML 5, CSS3, JavaScript, PHP 5, et jQuery pour développer vos pages.

I. La première étape va être de réaliser l'arborescence du site. Vous allez devoir créer un dossier racine nommé : **tdwebAC** (www/tdwebAC/)

Puis créer 5 dossiers :

1. **css** : pour sauvegarder les feuilles de style
2. **lib** : pour sauvegarder vos scripts et les librairies (jQuery, jeux.js, ajax.js, etc)
3. **ima** : pour sauvegarder vos images
4. **php** : pour sauvegarder les fichiers php
5. **data** : pour sauvegarder les données de votre base de données

Nous allons ensuite récupérer une base de données. Vous pouvez suivre le lien **3** pour récupérer la Liste des pays du monde au format qui vous conviendra : SQL, CSV ou XML.

Nous allons définir une page html simple qui demande à l'utilisateur de saisir les premières lettres d'un pays et lui propose une liste de pays associer.

II. Moteurs Ajax (*échauffement*)

Nous allons commencer avec notre propre moteur AJAX. L'objectif est d'initier une communication avec un fichier PHP qui va pré traiter la base de données. Dans cet exercice, nous allons utiliser le fichier php pour remplir une balise <select> dont les options seront les noms des pays.

Nous avons besoin de trois fichiers de code :

1. html (index.html)
2. php (select.php)
3. js (ajax.js)

Voici le fichier html :

```
<!DOCTYPE HTML>
<html lang="fr">
<head>
  <meta charset="utf-8">
  <title>Pays list</title>
  <link rel="stylesheet" href="css/style.css">
  <script src="lib/ajax.js"></script>

</head>
<body onload="process();">
        <!-- Debut: En-tete de la page Web -->
        <header>
             <nav align="right"></nav>
            <!-- Fin: Menu de la page Web -->
        </header>
        <!-- Fin: En-tete de la page Web -->
<form method="post" name="form" action="">
    <div>
        <p>
                <span id="select"></span>
        </p>
    </div>
</form>
<!-- Debut: Pied de la page Web -->
        <footer align="center">
                <p>Mon Site &copy; Tous droits reserves</p>
            <!-- Debut: Liens de naviagtion du pied de page -->
            <nav>
                <a href="mentions-legales.html">Mentions Legales</a>
                <a        href="conditions-generales.html">Conditions
Generales</a>
                <a href="a-propos.html">A propos</a>
            </nav>
            <!-- Fin: Liens de naviagtion du pied de page -->
        </footer>
        <!-- Fin: Pied de la page Web -->
</body>
</html>
```

Le fichier javascript :

```
var _xmlHttp = createXmlHttpRequestObject(); // récupère la référence
à l'objet XMLHttpRequest
function createXmlHttpRequestObject() { // crée l'objet XMLHttpRequest
var _xmlHttp; // objet XMLHttpRequest temporaire
  if(window.ActiveXObject){ // si sous Internet Explorer
```

```
    try{
      _xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    }catch (e) {
      _xmlHttp = false;
    }
  } else { // si c'est Mozilla ou un autre browser
    try {
      _xmlHttp = new XMLHttpRequest();
    }catch (e) {
      _xmlHttp = false;
    }
  }
  // affiche un message d'erreur ou l'objet créé
  if (!_xmlHttp) alert("Error creating the XMLHttpRequest object.");
  else return _xmlHttp;
}

function process(){   // utilise XMLHttpRequest pour requête HTTP
asynchrone
     if(_xmlHttp&&_xmlHttp.readyState!=0){
          _xmlHttp.abort()
     }
     _xmlHttp=createXmlHttpRequestObject();
     if(_xmlHttp){   // si pas occupé
    _xmlHttp.open("GET","php/select.php", true);
     // se prépare à exécuter monscript.php sur le serveur
    _xmlHttp.onreadystatechange = handleServerResponse;
     // défini l'action a exécuter lors de la réception de la réponse
    _xmlHttp.send(null); // lance la requête sur le serveur
  }  else {
          setTimeout('process()', 1000); // si occupé réessaie dans
une seconde
     }
}
function handleServerResponse() { // exécuté quand le serveur répond
  if (_xmlHttp.readyState == 4) { // si la transaction est finie
     if (_xmlHttp.status == 200) { // si fini comme il faut
      var Response = _xmlHttp.responseText; // extrait le code XML
reçu
      document.getElementById("select").innerHTML = Response;
      //setTimeout('process()', 1000); // relance la séquence
    }else {   // status différent de 200 = erreur
      alert("There was a problem accessing the server: " +
_xmlHttp.statusText);
    }
  }
}
```

A vous de faire le php. Pour vous aider, voici le squelette :

```
<meta charset="utf-8">
<?php
/*Ouverture du fichier en lecture seule*/
/*Si on a réussi à ouvrir le fichier*/
     /* création du select */
     /*Tant que l'on n'est pas à la fin du fichier*/
     /*On lit la ligne courante*/
          /*On crée un tableau avec chaque ligne */
          /* création des options*/
          // Vérification de la taille du tableau
          // Suppression des ""
     /*On ferme le fichier*/
     /* on ferme le select */
     /* on renvoie le résultat */
/* Sinon en renvoie une erreur */
?>
```

III. jQuery

Exemple de code :

```
$("#more_come").click(function(){

    $.ajax({
       url : 'more_com.php', // La ressource ciblée
       type : 'GET', // Le type de la requête http GET ou POST
       data : 'utilisateur=' + nom_user; //les données envoyées
       dataType : 'html' // Le type de données à recevoir, ici, du
HTML.
       success : function(code_html, statut){}, // code_html contient
le HTML renvoyé
       error : function(resultat, statut, erreur){},
       complete : function(resultat, statut){}
    });

});
```

On peut remplacer *$.ajax … type* pour $.get() ou $.post().

Vous allez devoir faire la même chose en remplaçant votre moteur ajax par jQuery.

Il va falloir supprimer le onload du fichier html, et inclure jQuery

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"></script>
```

Ensuite, vous remplacez le code de votre moteur ajax par :

```
$(document).ready(function($){
           $.get( "php/select.php", function( data ) {
                 $("#select" ).html( data );
                 //alert( "Load was performed.");
});
});
```

Une constatation, ça semble plus long, enfaite non dans le premier exemple on crée les options avec l'événement onload, alors qu'en jQuery on utilise la fonction ready qui attend que tout le DOM soit chargé donc après onload.

IV. Exemple d'autocomplete avec jQuery (utilisation du module jQuery UI)

Maintenant, nous allons créer un formulaire avec un champ input de recherche et il va falloir proposer des pays lorsque l'utilisateur tape des lettres.

On peut utiliser le module autocomplete de jQuery comme ça : $('#recherche').autocomplete();

Recherche étant l'id de l'input.

Il va falloir inclure le module jQuery UI après l'avoir téléchargé.

```
<script src="lib/jquery-ui-1.11.2/jquery-ui.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"></script>
```

Voici un exemple de code :

```
$(function() {
var liste = [
    "France",
    "Finlande",
    "Fidji"
];

$('#recherche').autocomplete({
    source : liste, // les données
     minLength : 2, // on indique qu'il faut taper au moins 3
caractères pour afficher l'autocomplétion
     /*
     position : { // la liste va se placer au-dessus et à l'extérieur
du champ de texte.
        my : 'bottom', // qui définit la position de la liste autour
du champ de texte
```

```
        at : 'bottom' // définit la position de la liste dans le champ
de texte
            //  top, bottom, left ou right
      }
      */
});
});
```

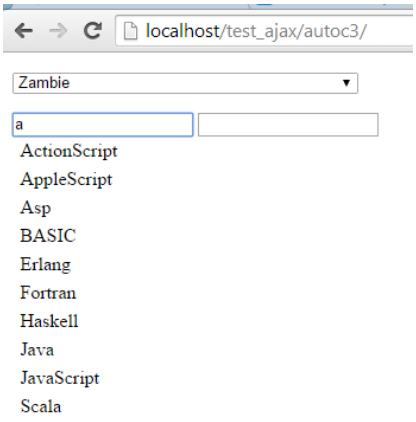


Figure 1: exemple

Explications :

$("#article").autocomplete, initialise l'auto-complétion pour l'objet input d'id recherche.

source: "page.php" ou function … s : cript qui renvoie la liste des données trouvées

minLength: 2 : permet d'attendre d'avoir deux caractères avant de lancer la recherche.

select: function(event, ui) {traitement} : le traitement à effectuer quand la sélection a été faite.

Essayez le code.

Maintenant il va falloir récupérer des données depuis votre page php, et remplir la liste avec. Pour cela, nous allons modifier la source avec une fonction anonyme.

```
$('#recherche').autocomplete({
    source : function(){ // la fonction anonyme permet de maintenir
une requête AJAX directement dans le plugin
     $.ajax({
            url : 'php/select.php', // l'adresse
            dataType : 'json', // on spécifie bien que le type de
données est en JSON ou un xml
             data : {
                donneeUser : $('#recherche').val(), // on donne la
chaîne de caractère tapée dans le champ de recherche
                maxRows : 15
            },
            success : function(donnee){…}
        });
    }
});
```

Dans ce cas, il faut modifier le code php pour renvoyer un fichier json, pour cela on peut utiliser la fonction json_encode qui prend un tableau en paramètre (voir la doc php).

Il faut aussi utiliser les données envoyées pour filtrer le fichier json en php. Coté serveur vous pouvez récupérer les données du formulaire avec $_GET['donneeUser'], ensuite, il va falloir créer un fichier JSON avec les pays composés des caractères saisis par l'utilisateur. Par exemple, vous pouvez regarder al fonction php *stripos* (voir la doc php).

Il y a une autre méthode, encore plus simple :

```
$('#recherche').autocomplete({
    source : 'select.php'
});
```

Il est également possible de lui indiquer directement le nom d'un fichier. La fonction va lire automatiquement, et récupérer les données. Dans ce cas, il est possible d'utiliser coté serveur une variable superglobale de type GET pour récupérer le champ saisi par l'utilisateur. La variable s'appelle 'term' ($term = $_GET['term'];). Grace à ça, il est possible de pré traiter le fichier php.

A vous de coder.

V. HTML 5

Nous allons maintenant faire la méthode chose avec la nouvelle balise HTML 5 <datalist> (voir lien 5).

Exemple de code :

```
<input list="pays" type="text" id="choix_pays">
<datalist id="pays">
  <option value="Italie">
  <option value="France">
  <option value="Amerique">
  <option value="Bresil">
</datalist>
```

Au lieu de générer un fichier JSON, comme dans la partie 1, nous allons directement renvoyer de l'HTML et construire la balise <datalist> en bleu.

Voici le code HTML à ajouter :

```
<span id="malist" />
```

Voici la fonction javascript :

```
$(function(){
	$("#recherche").keypress(function() { // id de l'input
		var msg = $("#recherche").val(); // recupération du text
		$.ajax({
			url : 'php/selectjson.php',
			type : 'GET', // Le type de la requête HTTP
			data : 'term='+msg, // On fait passer nos variables
			dataType : 'html',
			success : function(code_html, statut){ // code_html
contient le HTML renvoyé
				$("#malist" ).html( code_html );
			},
			error : function(resultat, statut, erreur){
				alert(resultat)
			}
		});
		});
	});
```

Et le php :

```
<meta charset="utf-8">
<?php
if (!empty($_GET['term']))
{
	$term = $_GET['term'];
	/*Ouverture du fichier en lecture seule*/
	$file = fopen('../data/sql-pays.csv', 'r');
	/*Si on a réussi à ouvrir le fichier*/
	if ($file)
	{
		/*Tant que l'on est pas à la fin du fichier*/
		$array = '<datalist id="pays">';
		while (!feof($file))
		{
			/*On lit la ligne courante*/
			$buffer = fgets($file);
			/*On l'affiche echo $buffer; */
			$key = explode(",", $buffer);
			/* création des options*/
			if (count($key)>4) // verification de la taille du
tableau
			{

				$str = $key[4]; // suppression des ""
				$str = substr($str,1,strlen($str)-2);
```

```
                              //echo $str;
                              if (stripos(' ' . $str,$term)) // Recherche
la position de la première occurrence dans une chaîne, sans tenir
compte de la casse
                              {
                                    $array .= "<option>";
                                    $array .= $str; // et on crée nos
options
                                    $array .= "</option>";
                              }
                        }
                        }
            /*On ferme le fichier*/
            fclose($file);
            $array .= "</datalist>";
            echo $array; // il n'y a plus qu'à envoyer
      }
      else
      {
            echo "erreur";
      }
}
else
{
      echo "erreur";
}
?>
```

VI. Chalenges

Refaire le même travail mais avec un fichier de données au format XML !

Intégrer ses (nouvelles) connaissances dans le TD du MVC

VII. Liens

1. http://openclassrooms.com/courses/dynamisez-vos-sites-web-avec-javascript/deboguer-votre-code-partie-3-3
2. http://contrib.spip.net/Realiser-un-champ-de-formulaire-avec-autocompletion
3. http://sql.sh/categorie/bases-donnees-gratuites
4. http://openclassrooms.com/courses/decouvrez-la-puissance-de-jquery-ui/l-autocompletion-1
5. http://www.alsacreations.com/article/lire/1334-html5-element-datalist.html
6. http://php.net/manual/fr
7. http://api.jquery.com/keypress/