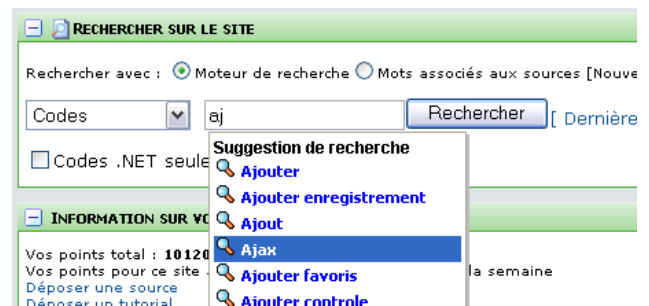


Conception d'une AjaxTextbox pas à pas

I. Introduction

Nous allons voir dans cet article comment construire pas à pas une AjaxTextbox. Ce que j'appelle AjaxTextbox est un champ texte avec une liste de suggestions comme celle présente sur Google suggest : <http://www.google.com/webhp?complete=1> ou sur la recherche de www.CodeS-SourceS.com



II. Le principe

Avant de commencer à analyser le code, voici le principe de fonctionnement : l'utilisateur tape une lettre dans la textbox, JavaScript effectue alors une requête Ajax sur une page de recherche et affichera le résultat dans un div en dessous de la textbox. Lorsque l'utilisateur appuie sur une nouvelle touche, une nouvelle requête Ajax s'exécute et ainsi de suite.

III. La partie HTML

Pour commencer nous allons avoir besoin d'un simple formulaire avec une textbox :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title>Fabrication d'une AjaxTextbox</title>
</head>
<body>

<form id="maform" action="AjaxTextbox.htm">
<label for="textboxSearch">Rechercher </label> <input type="text" id="textboxSearch"
name="textboxSearch" />
<input type="submit" id="buttonSubmit" value="Rechercher" />
</form>

</body>
</html>
```

Code : La partie HTML

IV. La partie Javascript

Pour faire fonctionner notre ajaxTextbox nous allons devoir nous servir de JavaScript. Commençons par créer un fichier AjaxTextbox.js ; n'oublions pas de l'inclure dans la page :

```
<script type="text/javascript" src="js/ajaxTextbox.js"></script>
```

A. Création et initialisation de l'objet AjaxTextbox

Déclarons un nouvel objet qui prendra en paramètre l'id de la textbox et l'url de la requête.

```
var AjaxTextbox = function(elmID, url)
{
// Les différentes fonctions seront écrites ici

// Le code d'initialisation sera écrit ici
}
```

Si vous n'êtes pas à l'aise avec les notions d'objets en JavaScript j'ai écrit quelques tutoriels à cette adresse : <http://blogs.developpeur.org/cyril/archive/category/1037.aspx>

On commence par récupérer la `textbox` à partir de l'id passé en paramètre du constructeur, on définit sa variable `autocomplete` à « off » pour désactiver l'auto-complétion natif de certains navigateurs. On crée ensuite un objet `XmlHttpRequest`.

```
var elmt = document.getElementById(elmtID);
elmt.autocomplete = "off";

var xhr_object = createXHRObject();
```

Voici la fonction `createXHRObject` qui va retourner un nouvel objet `XMLHttpRequest`

```
var createXHRObject = function()
{
    var tempXHR = null;

    if(window.XMLHttpRequest)
        tempXHR = new XMLHttpRequest(); // Firefox & co
    else if(window.ActiveXObject)
        tempXHR = new ActiveXObject("Microsoft.XMLHTTP"); // Internet Explorer
    else
        return null;

    return tempXHR;
}
```

Si le navigateur ne sait pas utiliser AJAX alors inutile d'aller plus loin

```
if (!xhr_object)
    return;
```

On déclare une variable qui contient le « *div* » de résultat.

```
var divResult = writeDivResult();
```

Voici la fonction `writeDivResult` qui va créer un nouvel élément « *div* » que l'on positionne juste en dessous de notre `textbox`.

```
var writeDivResult = function()
{
    var tempDiv = document.createElement('div');
    tempDiv.className = 'AjaxTextBoxResult';
    tempDiv.style.position = 'absolute';
    tempDiv.style.top = ( elmt.offsetTop + elmt.offsetHeight ) + 'px';
    tempDiv.style.left = elmt.offsetLeft + 'px';
    tempDiv.style.width = elmt.offsetWidth + 'px';
    tempDiv.style.display = 'none';

    document.getElementsByTagName('body')[0].appendChild(tempDiv);

    return tempDiv;
}
```

On déclare une variable qui nous permettra de savoir s'il y a une requête AJAX en cours

```
var isRequestActive = false;
```

B. Abonnements aux différents événements

On va maintenant s'abonner à l'événement « *onkeyup* » qui est déclenché lorsqu'une touche est relâchée.

```
elmt.onkeyup = function()
{
    if (isRequestActive)
        xhr_object.abort();

    makeRequest(elmt.value);
}
```

S'il y a une requête active alors on l'annule puis on crée une nouvelle requête en passant la valeur de la `textbox` en argument.

Lorsque la `textbox` perd le focus, on appelle la fonction qui va cacher le « *div* » de résultat.

```
elmt.onblur = function()
{
```

```
hideResult();  
}
```

Analysons maintenant la fonction makeRequest

```
var makeRequest = function(query)  
{  
    xhr_object.open('post', url, true);  
  
    xhr_object.onreadystatechange = function()  
    {  
        if(xhr_object.readyState == 4)  
        {  
            isRequestActive = false;  
            showResult();  
        }  
    }  
  
    xhr_object.setRequestHeader("Content-type", "application/x-www-form-urlencoded");  
    xhr_object.send('Query=' + escape(query));  
    isRequestActive = true;  
}
```

Il nous faut tout d'abord initialiser l'objet XMLHttpRequest avec sa fonction open. Cette fonction prend 3 paramètres : la méthode (post ou get), l'url de la requête et un booléen indiquant si l'on fait la requête en mode asynchrone ou non.

Faire une requête en asynchrone veut dire que l'exécution du script ne sera pas bloqué pendant l'exécution de la requête AJAX. Nous devons alors réagir à l'événement onreadystatechange pour être avertis lorsque la requête change d'état. Dans ce cas on vérifie que la requête est bien finie en regardant la propriété readyState. La valeur 4 correspond à l'état « Complete ». Si elle est finie on le spécifie à notre variable isRequestActive et on affiche les résultats.

Puis on va rajouter un en-tête http à notre requête via la méthode setRequestHeader, cet en-tête définit le Content-type de la requête.

Enfin on envoie la requête en passant le contenu de la textBox en paramètre via la méthode post.

Regardons maintenant la fonction showResult

```
var showResult = function()  
{  
  
    // On vérifie qu'il y ait des résultats et qu'il n'y ait pas d'erreur  
    if ( (xhr_object.responseText == '') || (xhr_object.status != 200) )  
    {  
        hideResult();  
        return;  
    }  
  
    divResult.style.display = 'block';  
    divResult.innerHTML = xhr_object.responseText;  
  
    // on récupère tous les li de la réponse puis on boucle dessus  
    var items = divResult.getElementsByTagName('ul')[0].childNodes;  
    for (var i = 0; i < items.length; i++)  
    {  
        // au passage de la souris, on change le className  
        items[i].onmouseover = function()  
        {  
            this.className = 'Hover';  
        }  
        // lorsque la souris quitte, on enlève le className  
        items[i].onmouseout = function()  
        {  
            this.className = '';  
        }  
        // quand on clique dessus, on met le contenu du li dans le textbox  
        items[i].onmousedown = function()  
        {  
            elmt.value = this.innerHTML;  
        }  
    }  
}
```

```
}
```

Tout d'abord on vérifie que la requête nous a bien retourné au moins un résultat et sans erreur.

Ensuite on met le contenu de la réponse dans le « *div* » de résultat. On récupère les différents items du résultat. Puis on boucle sur les ceux-ci pour s'abonner aux différents événements « *onmouseover* », « *onmouseout* » et « *onmousedown* » pour gérer le passage de la souris sur les items.

Ceci implique que la page de réponse doit contenir une liste de type « *ul* ».

Voici enfin la très simple fonction `hideResult` qui cachera le « *div* » de résultat

```
var hideResult = function()
{
    divResult.style.display = 'none';
}
```

La partie JavaScript est presque finie, il nous reste plus qu'à instancier l'objet avec les bons paramètres.

```
<script type="text/javascript">
<!--
    window.onload = function()
    {
        new AjaxTextbox('textboxSearch', '../AjaxResult/Default.aspx');
    }
-->
</script>
```

V. La partie serveur

Coté serveur il nous faut une page capable de retourner les résultats de cette façon :

```
<ul>
  <li>dscyxxjor</li>
  <li>dscrsgr</li>
  <li>dsa</li>
  <li>dsbya</li>
  <li>dsijo</li>
  <li>dswgyfhkgmupajdzk</li>
  <li>dsvmrgjyvilm</li>
  <li>dsynpmiegwtshfigt</li>
  <li>dsqoxvwlqkqi</li>
  <li>dsvl</li>
</ul>
```

Pour récupérer cette variable dans la page de résultat, il suffira de procéder de la même façon que pour récupérer une variable transmise par la méthode `post` :

En `asp.net` : `Request.Form("Query")` et en `php` : `$_POST['Query']`

S'il n'y a aucun résultat la page ne devra rien retourner pour que cela fonctionne correctement.

Vous pouvez évidemment écrire cette page avec la technologie que vous voulez : `asp.net`, `php`, ...

VI. Affinons la présentation avec CSS

Si vous avez testé ce que nous avons fait pour l'instant, le résultat n'est pas très joli. Voici donc quelques lignes de CSS afin d'égayer tout ça :

```
* {margin:0; padding:0;}

div.AjaxTextBoxResult
{
```

```

        background-color: #FFF;
    }
    div.AjaxTextBoxResult ul
    {
        border: solid 1px #333;
        margin: 0;
        padding: 0;
        list-style-type:none;
    }
    div.AjaxTextBoxResult ul li
    {
        line-height:1.3em;
        height:1.3em;
        background-color:#E6E6E6;
        padding:0 5px;
        cursor:pointer;
    }
    div.AjaxTextBoxResult ul liHover
    {
        background-color:#FFD9A0;
    }
}

```

Si vous avez suivi toutes les étapes vous devez alors tomber sur une page ressemblant à :

Rechercher :

- dscyzxjor
- dscrsgr
- dsa
- dsbya
- dsijo
- dswwgyfhkgrmpajzdk
- dsvmrgjywilm
- dsynpmiegwtshfigt
- dsqoxvwlqkqi
- dswl

VII. Allons plus loin

A. Optimisons notre résultat

Pour que cette AjaxTextbox soit vraiment utilisable il faudrait lui rajouter diverses fonctionnalités.

Dans notre exemple nous envoyons une requête Ajax à chaque fois que l'utilisateur tape une lettre. Pour éviter de surcharger le serveur, il faudrait rajouter un timer pour que les requêtes ne soient pas faites entre un intervalle t d'une demi-seconde par exemple.

Pour une meilleure expérience utilisateur il faudrait également rajouter la gestion des touches haut et bas pour se déplacer dans les résultats, etc.

J'ai complété l'exemple avec ces fonctionnalités, vous pouvez retrouver l'implémentation de celle-ci sur le CD joint au magazine.

B. Allons plus vite : utilisation du framework script.aculo.us

Ce que nous avons fait était très bien dans un but éducatif, mais si vous voulez aller plus vite et utiliser une AjaxTextbox beaucoup plus complète il est très intéressant d'utiliser des Framework qui possèdent la fonctionnalité de complétion. Je vous conseille script.aculo.us qui est très puissant. Voici comment arriver au même résultat avec ce Framework.

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

```

```

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>Utilisation de script.aculo.us</title>
  <link type="text/css" rel="stylesheet" href="css/styles.css" />
  <script src="js/prototype.js" type="text/javascript"></script>
  <script src="js/scriptaculous.js" type="text/javascript"></script>
  <script type="text/javascript">
<!--
    window.onload = function()
    {
      new Ajax.Autocompleter('textboxSearch', 'textboxSearchAutoComplete',
'../AjaxResult/Default.aspx')
    }
-->
  </script>
</head>
<body>
  <form id="maform" action="AjaxTextbox.htm">
    <label for="textboxSearch"> Rechercher :</label> <input type="text" id="textboxSearch"
name="textboxSearch" style="width:200px" />
    <div id="textboxSearchAutoComplete" class="AjaxTextBoxResult"></div>
    <input type="submit" id="buttonSubmit" value="Rechercher" />
  </form>
</body>
</html>

```

VIII. Ressources

A. Liens utiles

Tutoriels pour bien débuter avec JavaScript : <http://blogs.developpeur.org/cyril/archive/category/1037.aspx>

Un très bon tutorial sur les requêtes Ajax : http://robloche.free.fr/javascript/tuto_xhr/tuto_xhr.html

Le code complet des sources ainsi que le code des améliorations possible est disponible sur le CD du magazine.

B. Code au complet

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>Fabrication d'une AjaxTextbox</title>

  <link type="text/css" rel="stylesheet" href="css/styles.css" />

  <script type="text/javascript" src="js/ajaxTextbox.js"></script>
  <script type="text/javascript">
<!--
    window.onload = function()
    {
      new AjaxTextbox('textboxSearch', '../AjaxResult/Default.aspx');
    }
-->
  </script>
</head>
<body>

  <form id="maform" action="AjaxTextbox.htm">
    <label for="textboxSearch">Rechercher :</label> <input type="text" id="textboxSearch"
name="textboxSearch" />
    <input type="submit" id="buttonSubmit" value="Rechercher" />
  </form>

</body>
</html>

```

Fichier: [ExempleAjaxTextbox.htm](#)

```

* {margin:0; padding:0;}

div.AjaxTextBoxResult
{
    background-color: #FFF;
}
div.AjaxTextBoxResult ul
{
    border: solid 1px #333;
    margin: 0;
    padding: 0;
    list-style-type:none;
}
div.AjaxTextBoxResult ul li
{
    line-height:1.3em;
    height:1.3em;
    background-color:#E6E6E6;
    padding:0 5px;
    cursor:pointer;
}
div.AjaxTextBoxResult ul li.Hover
{
    background-color:#FFD9A0;
}

```

Fichier: [css/styles.css](#)

```

var AjaxTextbox = function(elmtID, url)
{
    // créer et retourne un nouvelle objet XMLHttpRequest
    var createXHRObject = function()
    {
        var tempXHR = null;

        // Instantiation de notre objet XMLHttpRequest
        if(window.XMLHttpRequest)
            tempXHR = new XMLHttpRequest(); // Firefox
        else if(window.ActiveXObject)
            tempXHR = new ActiveXObject("Microsoft.XMLHTTP"); // Internet Explorer
        else
            return null; // XMLHttpRequest non supporté par le
navigateur

        return tempXHR;
    }

    // créer un nouveau noeud div et le positionne juste en dessous de la textbox
    var writeDivResult = function()
    {
        var tempDiv = document.createElement('div');
        tempDiv.className = 'AjaxTextBoxResult';
        tempDiv.style.position = 'absolute';
        tempDiv.style.top = ( elmt.offsetHeight + elmt.offsetTop ) + 'px';
        tempDiv.style.left = elmt.offsetLeft + 'px';
        tempDiv.style.width = elmt.offsetWidth + 'px';
        tempDiv.style.display = 'none';

        document.getElementsByTagName('body')[0].appendChild(tempDiv);

        return tempDiv;
    }

    // permet de faire une requete
    var makeRequest = function(query)
    {
        // initialise une nouvelle requete, avec la méthode post, sur l'url spécifié en mode asynchrone
        xhr_object.open('post', url, true);
    }
}

```

```

// lorsque la requete change d'etat
// on vérifie qu'elle est finit (readyStat ==4)
// si oui on indique que la requete est finit et on
// appelle la fonction qui va afficher les resultats
xhr_object.onreadystatechange = function()
{
    if(xhr_object.readyState == 4)
    {
        isRequestActive = false;
        showResult();
    }
}

// Sert pour l'encodage des paramètres de la requete
xhr_object.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
// Créer la requete en passant le paramètre encodé
xhr_object.send('Query=' + escape(query));
// on indique qu'il y a une requete d'active
isRequestActive = true;
}

// Cette fonction est appellé une fois la requete terminé
// elle permet d'afficher le div de suggestion
var showResult = function()
{
    // On vérifie qu'il y ait des resultats et qu'il n'y ait pas d'erreur
    if ( (xhr_object.responseText == '') || (xhr_object.status != 200) )
    {
        hideResult();
        return;
    }

    divResult.style.display = 'block';
    divResult.innerHTML = xhr_object.responseText;

    // on recupere tous les li de la réponse puis on boucle dessus
    var items = divResult.getElementsByTagName('ul')[0].childNodes;
    for (var i = 0; i < items.length; i++)
    {
        // au passage de la souris, on change le className
        items[i].onmouseover = function()
        {
            this.className = 'Hover';
        }
        // lorsque la souris quitte, on enleve le className
        items[i].onmouseout = function()
        {
            this.className = '';
        }
        // quand on clique dessus, on met le contenu du li dans le textbox
        items[i].onmousedown= function()
        {
            elmt.value = this.innerHTML;
        }
    }
}

// Permet de cacher le div de suggestion
var hideResult = function()
{
    divResult.style.display = 'none';
}

var elmt = document.getElementById(elmtID); // On recupere la textbox
elmt.autocomplete = "off"; // désactive l'autocomplétion de certains navigateurs

var xhr object = createXHRObject(); // déclaration de l'objet XMLHttpRequest
if (!xhr_object) // si le navigateur ne peut pas faire de l'ajax inutile
d'allerg plus loin
return;

```

```
var isRequestActive = false; // un simple boolean pour savoir s'il y a une requete en
cours
var divResult = writeDivResult(); // déclaration du div qui contiendra le résultat
elmt.onkeyup = function() // abonnement sur l'evenement "relachement de la touche"
{
    if (isRequestActive) // S'il y a une requete en attente on l'annule
        xhr_object.abort();
    makeRequest(elmt.value); // On fait notre requete en passant en paramètre le contenu
de la textbox
}
elmt.onblur = function() // abonnement sur l'evenement "perte du focus de la textbox"
{
    hideResult(); // On cache les resultats
}
}
```

Fichier: [js/AjaxTextbox.js](#)

Cyril DURAND

Membre de l'équipe de développement CodeS-SourceS
<http://blogs.developpeur.org/cyri/>