Introduction à AJAX et interaction avec PHP

Sommaire

I. Introduction

I-A. Remerciements

I-B. Prémbule : qu'est-ce qu'AJAX ?

I-C. Fonctionnement

II. AJAX pas à pas

II-A. Sans A et sans X => Javascript

II-B. Juste sans le X => Asynchronous Javascript

II-C. Le vrai, l'unique AJAX

III. Les choses qui nous facilitent la vie

III-A. Interaction avec le serveur : AJAX + PHP

III-B. Attendre c'est bien, le savoir c'est mieux !

IV. Conclusion

IV-A. Épilogue

IV-B. Liens

I. Introduction

I-A. Remerciements

Un grand merci à Denis Cabasson et Guillaume Rossolini pour leurs conseils avisés, leur rapidité et leur patience pendant la relecture.

I-B. Prémbule : qu'est-ce qu'AJAX ?

On parle beaucoup d'AJAX en ce moment dans le buzzword Web 2.0. En effet la technologie Web entière est basée sur le modèle de l'aller retour : pour une requête serveur, vous avez un retour qui se traduit par un rafraîchissement des données (la page Web affichée).

Ce modèle de fonctionnement est fiable car existant depuis très longtemps mais il pose aussi des problèmes d'interaction homme machine et de performances.

D'un point de vue utilisateur, le rafraîchissement de toute la page au moindre clic est synonyme de temps d'attente et de scintillement qui n'est pas toujours du meilleur effet dans une application professionnelle.

Du point de vue des performances, à la moindre modification, vous rechargez une page entière avec toutes ses balises HTML, ce qui génère un trafic important.

La technologie AJAX n'est pas nouvelle en soi. A signifie Asynchronous (Asynchrone), JA pour Javascript et X pour XML. La nouveauté d'AJAX est bel et bien de tirer parti des trois outils pour faire des applications dynamiques.

Les puristes ne vont sûrement pas aimer ce que je vais dire, mais on peut tout à fait faire de l'AJAX au sens large sans forcément faire de l'asynchrone ni du XML. Nous allons détailler les différentes possibilités fournies par AJAX dans ce document puis nous verrons comment interfacer tout cela avec PHP et une base MySQL.

I-C. Fonctionnement

AJAX est basé sur l'objet XMLHttpRequest qui permet de faire une requête via Javascript à un serveur HTTP. Le but est donc, comme dans le "Web 1.0", de faire une requête au serveur et d'en attendre le retour. Cependant, dans notre cas, le navigateur du client n'est pas nécessairement rafraîchi et tout est transparent pour l'utilisateur.

II. AJAX pas à pas

II-A. Sans A et sans X => Javascript

Dans un premier temps, nous allons aborder AJAX sans parler de dialogue asynchrone ni de XML, on va laisser ces barbarismes pour plus tard dans l'article et donc, pour l'instant, nous allons parler d'AJAX synchrone en mode texte.

Synchrone signifie ici que nous attendons que le serveur réponde à notre requête avant de rendre la main à l'utilisateur. Évidemment, si le serveur met du temps à répondre, le navigateur va sembler figé : nous verrons plus tard comment contourner cela.

Je vous propose tout de suite un exemple sur la manière de procéder.

L'exemple se trouve ici

Nous avons trois fichier, un fichier .html qui contient notre code, un fichier .js qui contient notre code assurant la fonctionnalité AJAX et un fichier texte tout simple qui contient ce que nous allons recevoir du serveur.

index1.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr"
lang="fr">
<head>
<title>Exemple 1</title>
</head>
<body>

<script type="text/javascript" src="ajax1.js"></script>
```

```
<p>
<a href="javascript:ajax();">Cliquez-moi !</a>
</p>

</body>
</html>
```

reponse.txt

```
Bonjour Monde
```

ajax1.js

```
function ajax()
{
   var xhr=null;

   if (window.XMLHttpRequest) {
      xhr = new XMLHttpRequest();
   }
   else if (window.ActiveXObject)
   {
      xhr = new ActiveXObject("Microsoft.XMLHTTP");
   }
   //on appelle le fichier reponse.txt
   xhr.open("GET", "http://gael-donat.developpez.com/web/intro-ajax/reponse.txt", false);
   xhr.send(null);

   alert(xhr.responseText);
}
```

La création de l'objet xhr par la méthode "var xhr=null;" n'est pas recommendée, je l'utilise ici par souci de transparence de code. Vous trouverez en annexe un article de Denis Cabasson illustrant une méthode pour le réaliser proprement.

Lorsque vous cliquez sur le lien "Cliquez moi !", le code Javascript est exécuté.

Ce code fait appel via l'objet XMLHttpRequest à la page reponse.txt qui contient le texte affiché en message box.

Voilà, ce n'est pas plus compliqué que cela et nous venons de faire de l'AJAX de manière synchrone en mode texte.

II-B. Juste sans le X => Asynchronous Javascript

Nous allons voir maintenant AJAX de manière asynchrone toujours en mode texte, dans le but de ne pas trop compliquer les choses dans un premier temps.

Le choix entre synchrone et asynchrone se fait tout simplement dans l'appel à notre objet instancié XMLHttpRequest dans le dernier paramètre :

- **true** pour asynchrone,
- **false** pour synchrone.

Très bien, nous savons faire une requête au serveur mais asynchrone veut dire que nous ne maîtrisons pas la réponse du serveur, alors comment on récupère notre résultat ?

Eh bien tout simplement en déclarant une fonction qui sera appelée lors du déclenchement d'un évènement de l'objet instancié -> xhr.onreadystatechange

Il suffit donc de faire un code qui ressemble à :

exemple

```
Requete_http.onreadystatechange = function() {
   // mettre le code souhaité
};
```

Cette fonction sera appelée à chaque changement dans l'état de notre objet " xhr.readyState ".

Voici les états que peut prendre la valeur de readyState :

- 0 non initialisée
- 1 en chargement
- 2 chargée

- 3 en cours de traitement
- 4 terminée

Passons à l'application pratique de ce que nous venons de voir.

L'exemple se trouve ici

Nous avons toujours notre fichier reponse.txt contenant "Bonjour Monde"

Voici notre fichier index2.html :

index2.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<title>Exemple 2</title>
</head>
<body>

<script type="text/javascript" src="ajax2.js"></script>

<p>
<a href="javascript:ajax();">Cliquez-moi encore !</a>
</p>

</body>
</html>
```

Et notre ajax2.js qui change :

ajax2.js

```
function ajax()
{
   var xhr=null;

   if (window.XMLHttpRequest) {
      xhr = new XMLHttpRequest();
   }
   else if (window.ActiveXObject)
   {
      xhr = new ActiveXObject("Microsoft.XMLHTTP");
```

```
    }
    //on définit l'appel de la fonction au retour serveur
    xhr.onreadystatechange = function() { alert_ajax(xhr); };

    //on appelle le fichier reponse.txt
    xhr.open("GET", "http://gael-donat.developpez.com/web/intro-ajax/reponse.txt", true);
    xhr.send(null);
}

function alert_ajax(xhr)
{
alert(xhr.responseText);
}
```

Vous voyez maintenant que nous avons deux fonctions : l'une sert à l'appel vers le serveur, la seconde sera executée quand le serveur aura répondu.

Remarquez au passage qu'on a bien passé l'appel de la fonction xhr.open avec le dernier paramètre à true.

Le message apparaît plusieurs fois vide puis plusieurs fois avec notre réponse, c'est normal car, comme on l'a vu au-dessus, le readyState a plusieurs états et nous n'en avons pas tenu compte dans notre traitement. Nous verrons cela plus tard.

II-C. Le vrai, l'unique AJAX

Maintenant que nous avons vu les possibilités synchrone et asynchrone, nous allons voir la partie XML de l'AJAX.

Nous allons remplacer le Requete.responseText par Requete.responseXML qui va donc nous renvoyer un document XML à traiter.

L'exemple se trouve ici

Pour notre exemple, nous allons prendre le fichier "reponse.xml" qui contiendra :

reponse.xml

```
<?xml version="1.0"?>
<exemple>
```

```
<donnee>Bonjour</donnee>
<donnee>Monde</donnee>
</exemple>
```

Notre fichier index3.html :

index3.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr"
lang="fr">

<head>
<title>Exemple 3</title>
</head>
<body>

<script type="text/javascript" src="ajax3.js"></script>

<p>
<a href="javascript:ajax();">Cliquez-moi toujours !</a>
</p>

</body>
</html>
```

Et notre fichier ajax3.js :

ajax3.js

```
function ajax()
{
   var xhr=null;

   if (window.XMLHttpRequest) {
      xhr = new XMLHttpRequest();
   }
   else if (window.ActiveXObject)
   {
      xhr = new ActiveXObject("Microsoft.XMLHTTP");
   }
   //on définit l'appel de la fonction au retour serveur
   xhr.onreadystatechange = function() { alert_ajax(xhr); };

   //on appelle le fichier reponse.txt
```

```
    xhr.open("GET", "http://gael-donat.developpez.com/web/intro-ajax/reponse.xml", true);
    xhr.send(null);
}

function alert_ajax(xhr)
{
var docXML= xhr.responseXML;
var items = docXML.getElementsByTagName("donnee")
//on fait juste une boucle sur chaque élément "donnee" trouvé
for (i=0;i<items.length;i++)
{
        alert (items.item(i).firstChild.data);
}
}
```

Le code qui change là c'est l'assignation du résultat à une variable qu'on va traiter par node.

Là on va juste lire en boucle tout les éléments de type "donnee" de notre fichier XML.

III. Les choses qui nous facilitent la vie

III-A. Interaction avec le serveur : AJAX + PHP

Maintenant que nous savons traiter un fichier XML, nous allons voir comment générer un fichier XML avec PHP et MySQL.

L'exemple se trouve ici

Nous prendrons une table :

```
CREATE TABLE `temp` (
 `text` VARCHAR( 250 ) NOT NULL
) TYPE = MYISAM ;

INSERT INTO `temp` ( `text` ) VALUES ('Bonjour'), ('Monde');
```

Ainsi qu'un fichier reponse.php qui va simplement lire dans la base de donnée et transformer le résultat en XML sur le même modèle que notre fichier XML de l'exemple précédent :

reponse.php

```php
<?php
header('Content-Type: text/xml');
echo "<?xml version=\"1.0\"?>\n";
echo "<exemple>\n";

//on connecte a la BDD
$dbhost="localhost";
$dbuser="gael";
$dbpass="donat ";

$dblink=mysql_connect($dbhost,$dbuser,$dbpass);
mysql_select_db("gael",$dblink);

//on lance la requete
$query = "SELECT * FROM temp";
$result = mysql_query($query,$dblink) or die (mysql_error($dblink));

//On boucle sur le resultat
while ($row = mysql_fetch_array($result, MYSQL_NUM))
{
echo "<donnee>" . $row[0] . "</donnee>\n";
}
echo "</exemple>\n";

?>
```

Notez la ligne header('Content-Type: text/xml'); très importante c'est ce qui permet à PHP de dire au navigateur le format du fichier retourné.

Notre fichier index4.html :

index4.html

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">

<head>
<title>Exemple 4</title>
</head>
<body>
```

```
<script type="text/javascript" src="ajax4.js"></script>

<p>
<a href="javascript:ajax();">Vous avez le droit de me cliquer !</a>
</p>

</body>
</html>
```

Notre fichier ajax4.js est sensiblement identique au ajax3.js car la structure du XML est la même. On change juste le reponse.xml par le reponse.php :

ajax4.js

```
function ajax()
{
   var xhr=null;

   if (window.XMLHttpRequest) {
      xhr = new XMLHttpRequest();
   }
   else if (window.ActiveXObject)
   {
      xhr = new ActiveXObject("Microsoft.XMLHTTP");
   }
   //on définit l'appel de la fonction au retour serveur
   xhr.onreadystatechange = function() { alert_ajax(xhr); };

   //on appelle le fichier reponse.txt
   xhr.open("GET", "http://gael-donat.developpez.com/web/intro-ajax/reponse.php", true);
   xhr.send(null);
}

function alert_ajax(xhr)
{
var docXML= xhr.responseXML;
var items = docXML.getElementsByTagName("donnee")
//on fait juste une boucle sur chaque element "donnee" trouvé
for (i=0;i<items.length;i++)
{
        alert (items.item(i).firstChild.data);
}
}
```

Voilà.

III-B. Attendre c'est bien, le savoir c'est mieux !

Comme nous l'avons vu précédemment, le mode asynchrone permet de ne pas bloquer le navigateur client pendant le chargement de la page. Nous allons voir ici comment afficher un message d'attente pendant le traitement Javascript.

Cela va se faire en deux temps : d'une part afficher un message quelconque lors de l'appel initial, puis le retirer lorsque notre onreadystatechange passe à 4.

Le seul intérêt est d'avertir l'utilisateur qu'une mise à jour des données est en cours.

L'exemple se trouve ici

Voici donc notre fichier reponse.php modifié avec une pause forcée pour bien voir le message d'attente :

reponse5.php

```
<?php
header('Content-Type: text/xml');
echo "<?xml version=\"1.0\"?>\n";
echo "<exemple>\n";

//on connect
$dbhost="localhost";
$dbuser="donat";
$dbpass="donat";

$dblink=mysql_connect($dbhost,$dbuser,$dbpass);
mysql_select_db("gael",$dblink);

//on lance la requete
$query = "SELECT * FROM temp";
$result = mysql_query($query,$dblink) or die (mysql_error($dblink));
sleep(5);

//On boucle sur le resultat
while ($row = mysql_fetch_array($result, MYSQL_NUM))
{
 echo "<donnee>" ; $row[0] ; "</donnee>\n";
}
echo "</exemple>\n";
```

```
?>
```

On note donc le sleep(5) qui oblige PHP à stopper...

J'ajoute un petit fichier CSS qui permettra de gérer le style masqué/affiché du message d'attente :

exemple5.css

```
.tumevoispas
{
 visibility: hidden;
}

.tumevois
{
 visibility: visible;
 font-size : 200%;
 background-color: #DAEDFC;
 width: 300px;
 height: 40px;
}
```

Notre fichier index5.html contient le code de gestion et l'ajout d'un DIV contenant le message d'attente.

index5.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr"
lang="fr">
<head>
<title>Exemple 5</title>
<link rel="stylesheet" href="exemple5.css" type="text/css" />
</head>

<body>
<script type="text/javascript" src="ajax5.js"></script>

<p>
<a href="javascript:ajax();">Allez, une dernière fois !</a>
</p>
```

```
<div class="tumevoispas" id="message">Veuillez patienter...</div>

</body>
</html>
```

Notez l'apparition d'un DIV qui est caché lors du chargement de la page, c'est cet objet que nous allons afficher lorsque AJAX sera en train de s'exécuter.

Notre fichier Javascript ajax5.js est modifié pour gérer le message d'attente :

ajax5.js

```
function ajax()
{
   var xhr=null;

   if (window.XMLHttpRequest) {
      xhr = new XMLHttpRequest();
   }
   else if (window.ActiveXObject)
   {
      xhr = new ActiveXObject("Microsoft.XMLHTTP");
   }
   //on définit l'appel de la fonction au retour serveur
   xhr.onreadystatechange = function() { alert_ajax(xhr); };

   //on affiche le message d'acceuil
   document.getElementById("message").className="tumevois";

   //on appelle le fichier reponse.txt
   xhr.open("GET", "http://gael-donat.developpez.com/web/intro-
ajax/reponse5.php", true);
   xhr.send(null);
}

function alert_ajax(xhr)
{
   if (xhr.readyState==4)
   {
         var docXML= xhr.responseXML;
         var items = docXML.getElementsByTagName("donnee")
         //on fait juste une boucle sur chaque element "donnee" trouvé

document.getElementById("message").className="tumevoispas";
```

```
                for (i=0;i<items.length;i++)
                {
                        alert (items.item(i).firstChild.data);
                }
        }
}
```

Notez les deux lignes supplémentaires qui changent la classe de l'objet DIV du HTML.

IV. Conclusion

IV-A. Épilogue

Vous venez de faire vos premiers pas dans la création d'une application nouvelle génération qui, bien qu'elle ne soit pas normée par le W3C, fonctionne avec la plupart des navigateurs du marché et sur tous les systèmes d'exploitation.