

AJAX et PHP

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com



- 1 Introduction
- 2 AJAX avec JavaScript
- 3 Exemple
- 4 AJAX et jQuery

PHP & AJAX

Définition et caractéristiques

- a été inventé par Jesse James Garrett en 2005
- n'est pas un langage de programmation, mais plutôt une technologie
- utilise l'objet non standard XMLHttpRequest pour échanger avec des scripts situés coté serveur
- permet d'échanger des informations sous format :
 - Textuel
 - XML
 - HTML
 - JSON

PHP & AJAX

Avantages

- faire des requêtes vers le serveur d'une façon discrète : effectue la requête de façon asynchrone (en arrière plan de la page) :
 - sans perturber le flux normal de celle-ci
 - sans avoir à recharger la page
- analyser et travailler avec des documents de plusieurs formats XML, JSON...

PHP & AJAX

Avantages

- faire des requêtes vers le serveur d'une façon discrète : effectue la requête de façon asynchrone (en arrière plan de la page) :
 - sans perturber le flux normal de celle-ci
 - sans avoir à recharger la page
- analyser et travailler avec des documents de plusieurs formats XML, JSON...

Initialement, XML du XMLHttpRequest ou X d'AJAX désignent le format d'échange mais de nos jours on s'oriente plutôt vers le format JavaScript Object Notation (JSON).

PHP & AJAX

jQuery a simplifié l'écriture d'**AJAX**

- plus besoin de XMLHttpRequest et ses problèmes d'incompatibilité (entre navigateurs) ou de sa complexité
- en utilisant `$.ajax()` ou `load()`

PHP & AJAX

Étape à suivre

- Créer une requête
- Préciser ce qu'on fait à la réception d'une réponse
- Lancer la requête
 - ouvrir
 - envoyer

PHP & AJAX

Un peu d'histoire

- Les navigateurs Internet Explorer dont la version est < 7 utilisent ActiveX, développé par Microsoft, pour échanger avec le serveur
 - première version :

```
var xhr = new  
ActiveXObject ("Microsoft.XMLHTTP");
```
 - deuxième version :

```
var xhr = new  
ActiveXObject ("Msxml2.XMLHTTP");
```
- Les autres navigateurs (Safari, Firefox, Chrome...) utilisent :
 - ```
var xhr = new XMLHttpRequest();
```

# PHP & AJAX

## Définir un objet compatible avec (tous) les navigateurs

```
function getXMLHttpRequest() {
 var xhr = null;
 if (window.XMLHttpRequest || window.ActiveXObject) {
 if (window.ActiveXObject) {
 try {
 xhr = new ActiveXObject("Msxml2.XMLHTTP");
 }
 catch(e) {
 xhr = new ActiveXObject("Microsoft.XMLHTTP");
 }
 }
 else {
 xhr = new XMLHttpRequest();
 }
 }
 else {
 alert("Navigateur incompatible avec XMLHttpRequest");
 return null;
 }
 return xhr;
}
```

# PHP & AJAX

Préciser au serveur le nom de la fonction à exécuter à la réception d'une réponse

```
// Create a request
xhr = getXMLHttpRequest();
// Specify the JavaScript function without () nor parameters
xhr.onreadystatechange = nomFonction;
```

ou bien aussi

```
// Create a request
xhr = getXMLHttpRequest();
// Specify the JavaScript function without () nor parameters
xhr.onreadystatechange = function() {
 // instructions of anonymous function
};
```

# PHP & AJAX

## Première étape : ouverture

```
xhr.open('method', 'URL', bool);
```

### Étapes à suivre

- method : nom de la méthode : GET, POST...
- URL : URL de la page demandée
  - page statique : XML, txt...
  - page dynamique : PHP, JSP, ASP...
- bool : TRUE pour dire que c'est asynchrone (l'exécution de la fonction JavaScript se poursuivra en attendant l'arrivée de la réponse du serveur)

# PHP & AJAX

Exemple :

```
xhr.open("GET", "page.php", true);
```

ou en utilisant l'url d'une servlet java

```
xhr.open("GET", "SearchPersonne", true);
```

# PHP & AJAX

**Cas particulier de la méthode POST** : il faut changer l'entête de la requête avant d'envoyer des données issues d'un formulaire

```
xhr.open("POST", "page.php", true);
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhr.send();
```

# PHP & AJAX

## Deuxième étape : envoi

```
xhr.send();
```

# PHP & AJAX

Pour envoyer des paramètres avec POST :

```
xhr.open("POST", "page.php", true);
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhr.send("nomVar1=val1&...&nomVarN=valN");
```

Pour envoyer des paramètres avec GET :

```
xhr.open("GET", "page.php?nomVar1=val1&...&nomVarN=valN", true);
xhr.send();
```

Protéger les caractères spéciaux avant envoi :

```
var var1 = encodeURIComponent("variable contenant espaces");
xhr.open("GET", "page.php?var1=" + var1, true);
xhr.send();
```

# PHP & AJAX

À la réception d'une réponse, on appelle une fonction pour la traiter

```
xhr.onreadystatechange = nomFonction;
```

Dans la fonction JavaScript, Il faut vérifier l'état de la requête :

```
if (httpRequest.readyState == value){...}
```

## Valeur de readyState

- 0 : requête non initialisée
- 1 : requête initialisée mais pas encore envoyée
- 2 : requête reçue
- 3 : requête encours de traitement
- 4 : traitement de requête terminé et réponse prête

# PHP & AJAX

Il faut aussi vérifier le code d'état de la réponse HTTP du serveur :

```
if (httpRequest.status == value) { . . . }
```

## Plusieurs valeurs possibles de status

- 200 : OK
- 400 : Bad Request
- 401 : Unauthorized
- 404 : not found
- 500 : Internal Server Error
- la liste complète :  
<https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>

# PHP & AJAX

## Les méthodes

- `responseText` : pour récupérer les données d'une réponse sous forme de chaîne de caractères
- `responseXML` : pour récupérer les données d'une réponse sous forme d'un objet XML que nous pouvons parcourir à l'aide des fonctions DOM de JavaScript (`getElementById()` ...)

# PHP & AJAX

## Exemple

- On veut indiquer à l'utilisateur si le nom qu'il vient de saisir est valide ou pas.
- En effet, pendant qu'il saisit une valeur dans le champ nom, on lui affiche un message pour lui dire si ce dernier existe (ou pas) dans la base de données.

# PHP & AJAX

## Contenu de page1.php

```
<!DOCTYPE html>
<html>
<head>
 <title>Première page</title>
</head>
<body>
 <div>
 Nom à chercher : <input type="text" id="txt" oninput=
 "findNom(this.value)">

 </div>
 <script src="script.js">
 </script>
</body>
</html>
```

# PHP & AJAX

## Contenu de script.js

```
// n'oublions pas de copier la fonction XMLHttpRequest

function findNom(str) {
 let resultat = document.getElementById("resultat");
 if (str.length == 0) {
 resultat.innerHTML = "";
 } else {
 let xhr = new XMLHttpRequest();
 xhr.onreadystatechange = function() {
 if (xhr.readyState == 4 && xhr.status == 200) {
 resultat.innerHTML = xhr.responseText;
 }
 }
 xhr.open("GET", "page2.php?name=" + str, true);
 xhr.send();
 }
}
```

# PHP & AJAX

## Contenu de page2.php

```
<?php
spl_autoload_register(function ($class) {
 @include "managers/" . $class . '.php';
});

$manager = new PersonneManager();
$name = $_REQUEST["name"];
$result = $manager->existsNom($name);
echo $result ? "non valide" : "valide";
```

# PHP & AJAX

**Code de la méthode findByNom qu'il faut ajouter dans**

PersonneManager

```
function existsNom(string $nom) : bool
{
 try {
 $req = $this->_db->prepare('SELECT * FROM
 personne where nom = :nom');
 $req->bindValue(':nom', $nom, PDO::PARAM_STR
);
 $req->execute();
 return $req->rowCount() == 0 ? false : true;
 } catch (PDOException $e) {
 echo "Erreur : " . $e->getMessage();
 }
 return false;
}
```

# PHP & AJAX

## Deux manières différentes

- la fonction `$.ajax()`
- la méthode `load()`

## Syntaxe de la fonction \$.ajax()

```
$.ajax({param1:value1,...,paramN:valueN})
```

## Syntaxe de la fonction `$.ajax()`

```
$.ajax({param1:value1,...,paramN:valueN})
```

### Les paramètres

- `async` : Par défaut `true` pour dire asynchrone, sinon `false`
- `type` : Par default : GET. Sinon POST
- `url` : Par default : la page en cours, sinon la page ciblée
- `complete` : Execute une fonction lorsque la requête est terminée
- `error` : Fonction à exécuter en cas d'erreur
- `beforeSend` : Fonction à exécuter avant l'envoi de la requête
- `success` : Fonction à exécuter en cas de réussite de la requête
- `contentType` : Par default : `application/x-www-form-urlencoded ; charset=UTF-8`
- `username`, `password`, `data` (les paramètres), `dataType` (`html`, `xml`), `timeout`, ...

# PHP & AJAX

## Exemple

```
$('#txt').keyup(function(e) {
 $.ajax({
 type: "GET",
 url: "page2.php",
 data : {'name' : $('#txt').val()},
 dataType: "html",
 success: function(data) {
 $("#resultat").html(data);
 }
 });
});
```

# PHP & AJAX

N'oublions pas de modifier page1.php

```
<!DOCTYPE html>
<html>
<head>
 <title>Première page</title>
</head>
<body>
 <div>
 Nom à chercher : <input type="text" id="txt">

 </div>
 <script src="https://code.jquery.com/jquery-3.4.1.js">
 </script>
 <script src="script.js">
 </script>
</body>
</html>
```

# PHP & AJAX

Deux méthodes raccourcies de `$.ajax()`

- `$.get()`
- `$.post()`

Elles font appel à `$.ajax()` mais de façon simplifiée

# PHP & AJAX

## Syntaxe :

```
// Attention a l'ordre des parametres
$.post(
 'page2.php', // La page cible
 {
 param1 : value1, // data
 ...
 paramN : valueN
 },
 nomFonction, // Fonction de retour
 'text' // Format des donnees de retour
);
function nomFonction(texteRetour) {
 // Traiter le retour de l'appel AJAX.
}
```

# PHP & AJAX

## Exemple

```
$('#txt').keyup(function(e) {
 $.post(
 'page2.php',
 {'name' : $('#txt').val()},
 affiche,
 'text'
);
 function affiche (res) {
 $("#resultat").html(res);
 }
});
```

# PHP & AJAX

## Surveiller les requêtes AJAX

- `ajaxStart()` : précise la fonction à exécuter lorsqu'une première requête AJAX commence
- `ajaxStop()` : précise la fonction à exécuter lorsque toutes les requêtes AJAX sont terminées
- `ajaxComplete()` : précise la fonction à exécuter lorsqu'une requête AJAX est terminée
- `ajaxSuccess()` : précise la fonction à exécuter lorsqu'une requête AJAX s'est terminée avec succès
- `ajaxSend()` : précise la fonction à exécuter avant qu'une requête AJAX soit envoyée
- `ajaxError()` : précise la fonction à exécuter lorsqu'une requête AJAX s'est terminée avec erreur
- ...

# PHP & AJAX

## Exemple

```
// afficher une image de chargement initialement
// cachee
$(document).ajaxStart(function() {
 $('#loading').show();
});
```

# PHP & AJAX

## Syntaxe de la méthode load

```
\$(selecteur).load(url, data, function(reponse, status, xhr));
```

# PHP & AJAX

## Syntaxe de la méthode `load`

```
$(selecteur).load(url, data, function(reponse, status, xhr));
```

### Les paramètres

- `url` : La page ciblée (**obligatoire**)
- `data` : Les paramètres à envoyer
- `function(response, status, xhr)` : Fonction à exécuter quand la méthode est terminée
  - `response` : contient les données résultant de la demande
  - `status` : contient l'état de la demande ('success', 'error'...)
  - `xhr` : contient l'objet XMLHttpRequest

# PHP & AJAX

## Exemple 1

```
$('#txt').keyup(function() {
 var param = $('#txt').val();
 $('#resultat').load('page2.php', {
 name : param
 });
});
```

# PHP & AJAX

## Exemple 1

```
$('#txt').keyup(function() {
 var param = $('#txt').val();
 $('#resultat').load('page2.php', {
 name : param
 });
});
```

## Exemple 2

```
// charger la page2.php en entier dans mon #container
$('#container').load("page2.php");
```

# PHP & AJAX

## Exemple 1

```
$('#txt').keyup(function() {
 var param = $('#txt').val();
 $('#resultat').load('page2.php', {
 name : param
 });
});
```

## Exemple 2

```
// charger la page2.php en entier dans mon #container
$("#container").load("page2.php");
```

## Exemple 3

```
// charger l'element avec id partie1 de la page2.php dans mon #
// container
$("#container").load("page2.php #partie1");
```