AJAX

March 13, 2023

AJAX : une solution pour le Web 2.0

AJAX=

Exploitation d'un ensemble de technologies pour réaliser des applications RIA

RIA=

- Rich Internet Application
- Application en ligne rendant les mêmes services qu'une application installée sur un ordinateur
- Exemples :
  - Webmail
  - Moodle (cf : Plubel)
  - Google Maps
  - Amazon

Problématique

- web dynamique
- interactivité
- réactivité
- protocole HTTP
- format html

Point faible

Échanges client serveur soumis à un rechargement de la page

Autres alternatives pour le Web 2.0

Il existe d'autres technologies pour le web 2.0

- Flash couplé avec Flex
- Java déployé à l'aide de Java Web Start
- Corba
- web services (WSDL/SOAP/XML)
- ...

Les technologies d'AJAX

AJAX est l'exploitation de

- CSS (Cascading Style Sheets)
- DOM (Document Object Model)
- JavaScript et son objet XMLHttpRequest
- XML (eXtended Markup Language)
- **AJAX = Asynchronous JavaScript and XML**

Avantages par rapport aux autres technologies

Ne nécessite pas l'installation de plug-in
Ne nécessite pas le rechargement de la page

Pourquoi AJAX

- Effectuer des échanges Client/serveur sans devoir recharger la page
- Afficher le résultat d'un traitement coté serveur sans bloquer l'utilisateur durant ce temps.
- Comparer/tester des éléments stockés en BDD de manière transparente pour l'utilisateur
- Stocker des informations en bdd au fur et à mesure de la saisie

Attention à l'UX

Les traitements en tâche de fond permis par Ajax doivent prendre en compte l'expérience utilisateur qui s'attend à valider un formulaire et obtenir un affichage indiquant la prise en compte de sa demande.

A quoi sert AJAX

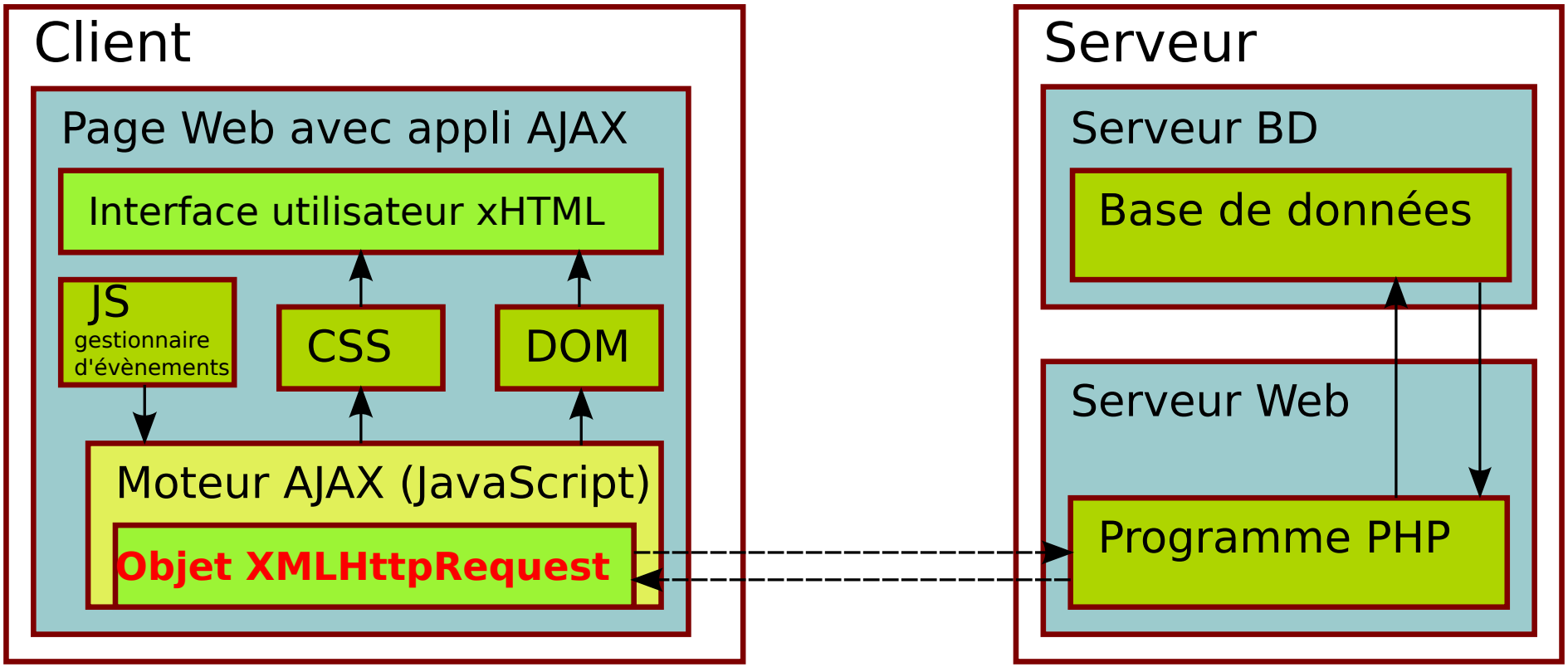
- Actualisation d'information en tâche de fond
  Exemple : Actualisation de la liste des visiteurs connectés au site sans rechargement de la page
- Complétion automatique
  Exemple : La saisie des premières lettres génère une liste déroulante contenant des mots clés provenant d'une base de données.
- Contrôle en temps réel les données d'un formulaire
  Exemple : les données saisies sont vérifiées immédiatement et avant l'envoi du formulaire

A quoi sert AJAX

- Lecture d'un flux RSS
  Exemple : récupération du flux RSS au format XML et affichage dans le navigateur
- Sauvegarde de documents éditables
  Exemple : édition d'un document en ligne et sauvegarde directe sans passer par un formulaire
- Widget = petites applications ne nécessitant pas la présence d'un navigateur pour fonctionner
  Exemple : affichage de la météo / Talk de Google

Exemple d'application utilisant AJAX

- Google
  construction d'une liste de suggestions à fur et à mesure de la saisie
- Gmail
- Google Maps
- Netvibes
  portail personnalisable
- Mise à jour du statut d'actions chronophages côté serveur sur l'interface client


Client
Page Web avec appli AJAX
Interface utilisateur xHTML
JS
gestionnaire
d'évènements
CSS
DOM
Moteur AJAX (JavaScript)
Objet XMLHttpRequest
Serveur
Serveur BD
Base de données
Serveur Web
Programme PHP

HTTP

Revoir le rappel sur les requêtes HTTP (GET and POST)

Communication avec le serveur

- Basée sur l'utilisation de l'objet JavaScript XMLHttpRequest
- Permet d'exécuter des requêtes HTTP
  - du navigateur vers le Serveur
  - **en mode asynchrone** : réponse différée sans blocage de l'application
  - **sans avoir à recharger la page**

Remarques :

- Existe depuis 1998 (sous IE sous forme de contrôle ActiveX)
- Implémenté sur la plupart des navigateurs
- fait l'objet d'une spécification du W3C

Remarque sécurité

- Les appels AJAX ne peuvent être réalisés que vers le domaine de l'application AJAX
- contrôlé par les navigateurs

Remarque

- Cela pose un problème pour accéder à des serveurs externes (flux RSS notamment)
- possibilité de mettre en oeuvre un serveur mandataire (proxy)

Propriétés de l'objet XMLHttpRequest

Propriété	Description
Onreadystatechange	désigne la fonction à appeler à chaque changement d'état d'une requête asynchrone
readyState	Etat d'une requête asynchrone (0 à 4)
responseText	Contient le résultat sous forme de texte
responseXml	Contient le résultat sous forme XML
status	Contient le code de statut HTTP (en lecture seul)
statusText	Contient le message de statut HTTP (en lecture seul)

État et code de statut

Valeur	État	Signification
0	Uninitialized	objet pas initialisé (méthode open() pas encore appelée)
1	Loading	Objet initialisé mais requête pas envoyée (méthode send() pas encore appelée)
2	Loaded	requête envoyée via la méthode send()
3	Interactive	La méthode est en cours de réception
4	Completed	réponse reçue, les données sont disponibles : • dans responseText si donnée texte • dans responseXML si donnée XML

Méthodes de l'objet XMLHttpRequest

abort()	Annule la requête en cours.
getAllResponseHeaders()	• retourne les entêtes HTTP de la réponse du serveur • est utilisable que dans l'état de code 3 ou 4 (Interactive ou Completed)
getAllResponseHeaders(nom_entete)	• retourne l'entête HTTP dont le nom est passé en paramètre • est utilisable que dans l'état de code 3 ou 4 (Interactive ou Completed)

Méthodes de l'objet XMLHttpRequest

open(methode,url,async)	Initialise l'objet en précisant : • la méthode utilisée (GET ou POST) • l'URL du script coté serveur • le type de communication : async = true ou false
send(contenu)	envoie une requête • utilisable qu'après appel de open (statut = 1) • contenu = null si méthode GET • contenu = chaîne si méthode POST

Instanciation de XMLHttpRequest

Enfin standardisée :

pour Mozilla, Opera,...

```
objetXHR = new XMLHttpRequest() ;
```

mais avant

pour IE > 5.0

```
objetXHR = new ActiveXObject("Msxml2.XMLHTTP") ;
```

pour IE 5.0

```
objetXHR = new ActiveXObject("Microsoft.XMLHTTP") ;
```

```
function createXHR(){
var resultat = null ;
try { // test pour opera, Mozilla,...
resultat = new XMLHttpRequest() ;
    }
   catch (Error) {
        try {  //test pour IE > 5.0
                resultat = new ActiveXObject("Msxml2.
                   XMLHTTP") ;
             }
            catch (Error){
            try{ // test pour IE 5.0
                          resultat = new ActiveXObject("
                             Microsoft.XMLHTTP") ;
                       }
                       catch (Error){
                              resultat = null ;
                           }
            } }
        return resultat ;
} }
```

Requête synchrone sans paramètre (GET ou POST)

```
// création d'une instance XMLHttpRequest
var objetXHR= new XMLHttpRequest() ;

// configuration de la méthode
objetXHR.open("get","script.php",false);// ou post

// envoi de la requête
objetXHR.send(null) ;

// récupération de la réponse du serveur
var resultat = objetXHR.responseText ;

// exploitation à faire par le suite
...
```

Coté serveur

```php
<?php

// on spécifie que la réponse envoyée sera de type texte
header("Content-Type: text/plain");

// la réponse est construite en écrivant sur le flux de
    sortie
echo "coucou" ;

?>
```

▸ Exemple

Requête asynchrone sans paramètre

Idem mode synchrone, mais en plus, il faut :

- déclarer la function qui va traiter les changements d'état de la requête,
  - elle sera appelée à chaque changement d'état
  - il faut tester l'état de la requête avant d'effectuer les traitements
- la référencer dans l'objet XMLHttpRequest

Requête asynchrone sans paramètre

Idem mode synchrone, mais en plus, il faut :

- déclarer la function qui va traiter les changements d'état de la requête,
  - elle sera appelée à chaque changement d'état
  - il faut tester l'état de la requête avant d'effectuer les traitements
- la référencer dans l'objet XMLHttpRequest

Requête asynchrone sans paramètre

Idem mode synchrone, mais en plus, il faut :

- déclarer la function qui va traiter les changements d'état de la requête,
  - elle sera appelée à chaque changement d'état
  - il faut tester l'état de la requête avant d'effectuer les traitements
- la référencer dans l'objet XMLHttpRequest

Requête asynchrone sans paramètre

Idem mode synchrone, mais en plus, il faut :

- déclarer la function qui va traiter les changements d'état de la requête,
  - elle sera appelée à chaque changement d'état
  - il faut tester l'état de la requête avant d'effectuer les traitements
- la référencer dans l'objet XMLHttpRequest

Requête asynchrone sans paramètre (GET ou POST)

```
// Création d'une instance XMLHttpRequest
var objetXHR= new XMLHttpRequest() ;

// Déclaration de la fonction de rappel
function traitementResultat() {
        if (objetXHR.readyState==4 { // la reponse a été
            envoyée
        var reponse = objetXHR.responseTest ;
            // ici on fait ce qu'on veut du résultat
           }}
// Désignation de la fonction de rappel
objetXHR.onreadystatechange = traitementResultat ;

// configuration de la méthode
objetXHR.open("get","script.php",true); // ou post

// envoi de la requête
objetXHR.send(null) ;
```

Traitement des erreurs

Sources de problèmes

- pas de script coté de serveur
- serveur non disponible ...

```
...
function traitementResultat() {
 if (objetXHR.readyState==4 { // la reponse a été envoyée

 // on teste le statut de la requête HTTP
  if (objetXHR.status == 200) { //idem code précédent
    } else {
    alert("Erreur HTTP N"+objetXHR.status);
    }
  }
 }
```

Rappel des codes du statut HTTP

Code de statut	signification	Exemple/Message
200 à 299	Succès de l'opération	200/OK : Requête accomplie avec Succès
300 à 399	Redirection	301/Moved Permanently : Redirection permanente
400 à 499	Erreur client	404/ Not Found : Document non trouvé
500 à 599	Erreur serveur	503 Service Unavailable : Service non disponible

Requête asynchrone avec un paramètre GET

Il suffit d'ajouter le paramètre à l'URL lors de l'initialisation de la requête avec la méthode **open**.
Exemple :

```
objetXHR.open("get","script.php?nom=toto",true);
```

Coté serveur

Dans le script on récupère la valeur de la variable à l'aide de la variable superglobale $_REQUEST :

```
<?php
header("Content-Type: text/plain");

if(isset ($_REQUEST['nom']))
$nom=$_REQUEST['nom'] ;
else // traitement valeur nom non récupérée
...
?>
```

Remarque

- On peut utiliser aussi la variable superglobale $_GET.
- $_REQUEST contient les variables de $_GET, $_POST et $_COOKIE

▸ Exemple avec entête HTTP en direct

Coté serveur

Dans le script on récupère la valeur de la variable à l'aide de la variable superglobale $_REQUEST :

```
<?php
header("Content-Type: text/plain");

if(isset ($_REQUEST['nom']))
$nom=$_REQUEST['nom'] ;
else // traitement valeur nom non récupérée
...
?>
```

Remarque

- On peut utiliser aussi la variable superglobale $_GET.
- $_REQUEST contient les variables de $_GET, $_POST et $_COOKIE

▸ Exemple avec entête HTTP en direct

Requête asynchrone avec un paramètre POST

Remarques

- généralement utilisée lorsque les données des paramètres sont > 512 octets
- pas de moyen direct pour tester le script php du serveur (il faut faire un formulaire POST de test)
- ATTENTION il faut spécifier le type de données envoyées à l'aide de la méthode setRequestHeader

Requête asynchrone avec un paramètre POST

Remarques

- généralement utilisée lorsque les données des paramètres sont > 512 octets
- pas de moyen direct pour tester le script php du serveur (il faut faire un formulaire POST de test)
- ATTENTION il faut spécifier le type de données envoyées à l'aide de la méthode setRequestHeader

Requête asynchrone avec un paramètre POST

Remarques

- généralement utilisée lorsque les données des paramètres sont > 512 octets
- pas de moyen direct pour tester le script php du serveur (il faut faire un formulaire POST de test)
- ATTENTION il faut spécifier le type de données envoyées à l'aide de la méthode setRequestHeader

Requête asynchrone avec un paramètre POST

- Le principe reste le même que pour une requête asynchrone de type GET
- Mais les paramètres ne sont pas passés par l'URL

```
// Création d'une instance XMLHttpRequest
var objetXHR= new XMLHttpRequest() ;
// Declaration de la fonction de rappel
function traitementResultat() {...          }
// Désignation de la fonction de rappel
objetXHR.onreadystatechange = traitementResultat ;
// configuration de la méthode methode post et asynchrone
objetXHR.open("post","script.php",true);
// affectation du type d'encodage de la requête
objetXHR.setRequestHeader("Content-Type","application/x-
    www-form-urlencode")

// envoi de la requête avec un parametre
objetXHR.send("mon_parametre=2&param2=toto") ;
```

Déclenchement de la requête via le gestionnaire d'événement

```
<script  type="text/javascript">
// gestionnaire d'evenements
window.onclick = affichemessage ;
// déclaration (globale) de l'objet XHR
var objetXHR ;
// declaration de la fonction de traitement
function traitementResultat(){
 if (objetXHR.readyState==4)// utilise l'objet global
        }
// declaration de la fonction d'appel du moteur AJAX
function affichemessage(){
// Creation
objetXHR = new XMLHttpRequest() ; // ATTENTION ne pas dé
    clarer une variable globale avec var
...
}
</script>
```

▸ Exemple avec mode synchrone vs asynchorne

Résultat texte ou XML ?

Pour savoir si le résultat est en texte ou en XML on interroge l'entête HTTP

```
if (objetXHR.status == 200) {
  if(objetXHR.getResponseHeader("Content-Type") == "text/
      xml"){
    alert(objetXHR.responseXML)
  }else{
    alert(objetXHR.responseText)
  }
}
```

Résultat en JSON

Coté serveur en PHP

```php
$retour=... ;
echo json_encode($retour);
```

Coté client en JS

```
...
//On récupère dans la reponseText
var myObj = JSON.parse(objetXHR.responseText);
...
```