JEE et AJAX

Achref El Mouelhi

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

elmouelhi.achref@gmail.com

Plan

AJAX

AJAX : **A**synchronous **J**avaScript **A**nd **X**ML

- Inventé par **Jesse James Garrett** en 2005
- Ce n'est pas un langage de programmation, mais plutôt une technologie
- Utilisant l'objet non standard `XMLHttpRequest` pour échanger avec des scripts situés coté serveur
- Permettant d'échanger des informations sous format :
  - **XML**
  - **HTML**
  - **JSON**
  - Textuel

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**...

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` et **X** d'**AJAX** désignent le format d'échange même si de nos jours on s'oriente plutôt vers le format **JSON** : **J**ava**S**cript **O**bject **N**otation.

AJAX

AJAX avec **jQuery**

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

AJAX

Étapes à suivre

- Créer une requête
- Préciser ce qu'on fait à la réception d'une réponse
- Ouvrir la requête
- Envoyer

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();`

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;
}
```

AJAX

Pour créer un objet `HTTP`, il suffit d'appeler la fonction `getXMLHttpRequest`

```
xhr = getXMLHttpRequest();
```

AJAX

Pour créer un objet `HTTP`, il suffit d'appeler la fonction `getXMLHttpRequest`

```
xhr = getXMLHttpRequest();
```

Dans la suite, nous utiliserons l'écriture suivante étant donné qu'elle est supportée par tous les navigateurs modernes

```
xhr = new XMLHttpRequest();
```

AJAX

Préciser au serveur la callback à exécuter à la réception d'une réponse

```
xhr = new XMLHttpRequest();

xhr.onreadystatechange = nomFonction;
```

AJAX

Préciser au serveur la callback à exécuter à la réception d'une réponse

```
xhr = new XMLHttpRequest();

xhr.onreadystatechange = nomFonction;
```

ou bien aussi en utilisant une fonction anonyme

```
xhr = new XMLHttpRequest();

xhr.onreadystatechange = function() {
    // instructions
};
```

AJAX

Première étape : ouverture

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

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**, **TEXT**...
  - 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)

AJAX

Exemple

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

AJAX

Exemple

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

ou en utilisant l'URL d'une servlet Java ou un contrôleur Spring

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

AJAX

Si vous souhaitez modifier l'entête : par exemple, lorsque vous envoyez des données au format `JSON` avec les méthodes `POST` ou `PUT`

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

AJAX

Deuxième étape : envoi

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

AJAX

Deuxième étape : envoi

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

Et si on a des données à envoyer (ici au format `JSON`)

```
let obj = JSON.stringify({"id": 1, "nom": "Doe", "prenom": "John"});
xhr.send(obj);
```

AJAX

Pour envoyer des paramètres dans la requête avec `POST`

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

AJAX

Pour envoyer des paramètres dans la requête avec `POST`

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

Avec GET :

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

AJAX

Pour envoyer des paramètres dans la requête avec `POST`

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

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();
```

AJAX

Nous devons indiquer le nom d'une fonction à exécuter au changement de l'état de la requête

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

AJAX

Nous devons indiquer le nom d'une fonction à exécuter au changement de l'état de la requête

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


Explication

Tout objet `XMLHttpRequest` a un attribut appelé `readyState` qui contient une valeur représentative de l'état de la requête.

AJAX

Nous devons indiquer le nom d'une fonction à exécuter au changement de l'état de la requête

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


Explication

Tout objet `XMLHttpRequest` a un attribut appelé `readyState` qui contient une valeur représentative de l'état de la requête.

Pour vérifier l'état de la requête (`readyState`)

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

AJAX

Valeur de `readyState`

- 0 (`UNSENT`) : objet créé et requête non initialisée (`open()` n'a pas encore été appelé)
- 1 (`OPENED`) : requête initialisée mais pas encore envoyée (`open` a été appelé)
- 2 (`HEADERS_RECEIVED`) : requête reçue (`send()` a été appelé
- 3 (`LOADING`) : requête encours de traitement
- 4 (`DONE`) : traitement de requête terminé et réponse prête (résultat dans `responseText`))

AJAX

Valeur de `readyState`

- 0 (`UNSENT`) : objet créé et requête non initialisée (`open()` n'a pas encore été appelé)
- 1 (`OPENED`) : requête initialisée mais pas encore envoyée (`open` a été appelé)
- 2 (`HEADERS_RECEIVED`) : requête reçue (`send()` a été appelé
- 3 (`LOADING`) : requête encours de traitement
- 4 (`DONE`) : traitement de requête terminé et réponse prête (résultat dans `responseText`))

Exemple

```
if (xhr.readyState == 4){...}
```

AJAX

Valeur de `readyState`

- 0 (`UNSENT`) : objet créé et requête non initialisée (`open()` n'a pas encore été appelé)
- 1 (`OPENED`) : requête initialisée mais pas encore envoyée (`open` a été appelé)
- 2 (`HEADERS_RECEIVED`) : requête reçue (`send()` a été appelé
- 3 (`LOADING`) : requête encours de traitement
- 4 (`DONE`) : traitement de requête terminé et réponse prête (résultat dans `responseText`))

Exemple

```
if (xhr.readyState == 4){...}
```

Ou

```
if (xhr.readyState == XMLHttpRequest.DONE){...}
```

AJAX

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

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

AJAX

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

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

Plusieurs valeurs possibles de status

- 200 : OK
- 201 : Created
- 400 : Bad Request
- 401 : Unauthorized
- 404 : Not Found
- 500 : Internal Server Error
- la liste complète :
  `https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html`

AJAX

Si nous devions exécuter une fonction uniquement à la réception de la réponse et pas à chaque changement d'état

```
xhr.onload = nomFonction;
```

AJAX

Où trouver la réponse ?

- `xhr.responseText` : pour récupérer les données d'une réponse sous forme de chaîne de caractères
- `xhr.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()`...)

AJAX

Considérons la Servlet `PersonneServlet`

```
public class PersonneServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;
    public PersonneServlet() {
        super();
    }

    protected void doGet(HttpServletRequest request,
       HttpServletResponse response) throws ServletException,
       IOException {
        this.getServletContext().getRequestDispatcher("/WEB-INF/Vue.jsp
           ").forward(request, response);
    }

    protected void doPost(HttpServletRequest request,
       HttpServletResponse response) throws ServletException,
       IOException {
        doGet(request, response);
    }
}
```

AJAX

Contenu du formulaire de la page `Vue.jsp`

```
<form action="AddPersonne" method="post">
  <div>
    <label for="nom">Nom</label>
    <input type ="text" id="nom" name="nom" onkeyup= "valider()"/>
    <span id="validationMessage"></span>
  </div>
  <div>
    <label for="prenom">Prenom</label>
    <input type ="text" id="prenom" name="prenom"/>
  </div>
  <button type="submit" >Ajouter</button>
</form>
```

AJAX

Contenu du formulaire de la page `Vue.jsp`

```
<form action="AddPersonne" method="post">
  <div>
    <label for="nom">Nom</label>
    <input type ="text" id="nom" name="nom" onkeyup= "valider()"/>
    <span id="validationMessage"></span>
  </div>
  <div>
    <label for="prenom">Prenom</label>
    <input type ="text" id="prenom" name="prenom"/>
  </div>
  <button type="submit" >Ajouter</button>
</form>
```


La fonction `valider()`

```
requete = "";
function valider() {
  var nom = document.getElementById("nom");
  var url = "SearchPersonne?nom=" + escape(nom.value);
  requete = new XMLHttpRequest();
  requete.open("GET", url, true);
  requete.onreadystatechange = displayMessage;
  requete.send();
}
```

AJAX

La fonction `displayMessage()`

```
function displayMessage() {
  var message = "";
  if ((requete.readyState == 4) && (requete.status == 200)) {
    var messageTag = requete.responseXML.getElementsByTagName("message")[0];
    message = messageTag.childNodes[0].nodeValue;
    mdiv = document.getElementById("validationMessage");
    mdiv.innerHTML = message;
  }
}
```

AJAX

La fonction `displayMessage()`

```
function displayMessage() {
  var message = "";
  if ((requete.readyState == 4) && (requete.status == 200)) {
    var messageTag = requete.responseXML.getElementsByTagName("message")[0];
    message = messageTag.childNodes[0].nodeValue;
    mdiv = document.getElementById("validationMessage");
    mdiv.innerHTML = message;
  }
}
```

Il faut

- créer un nouveau dossier `js` dans `WebContent` (ou `webapp` pour les projets **maven**)
- créer un fichier `script.js` dans `js`
- faire référence à ce fichier dans la page JSP avec la balise `<script src=js/script.js></script>`
- placer les 3 fonctions **JavaScript** (`valider`, `displayMessage` et `getXMLHttpRequest`) dans `script.js`

AJAX

La méthode `doGet()` de la servlet `SearchPersonneServlet` avec l'url `/SearchPersonne`

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
   ServletException, IOException {
  String resultat = "valide";
  String nom = request.getParameter("nom");
  PersonneDaoImpl daoImpl = new PersonneDaoImpl();
  boolean result = daoImpl.findByNom(nom);
  response.setContentType("text/xml");
  if (result) {
    resultat = "invalide";
  }
  response.getWriter().write("<message>"+resultat+"</message>");
}
```

AJAX

La méthode `findByNom()` du DAO

```
public boolean findByNom(String nom) {
  Connection c = MyConnection.getConnection();
  if (c !=null) {
    try {
      PreparedStatement ps =  c.prepareStatement("select * from personne where nom = ?
         ;");
      ps.setString(1, nom);
      ResultSet r =ps.executeQuery();
      if (r.next())
        return true;
      else return false;
    } catch (SQLException e) {
      e.printStackTrace();
      return false;
    }
  }
  return false;
}
```

AJAX

La fonction `valider()`

```
function valider(){
  var nom = document.getElementById("nom");
  var url = "SearchPersonne?nom=" + escape(nom.value);
  ajax(url)
    .then((res) => displayMessage(res))
    .catch((err) => console.log("error"))
}
```

La fonction `ajax()`

```
function ajax(url) {
  return new Promise(function(resolve, reject) {
    let xhr = getXMLHttpRequest();
    xhr.open("GET", url, false);
    xhr.send();
    if ((xhr.readyState == 4) && (xhr.status == 200))
      resolve(xhr);
    else
      reject(xhr);
  });
}
```

AJAX

La fonction `displayMessage()`

```
function displayMessage(res) {
  var message = "";
  var messageTag = res.responseXML.getElementsByTagName("
     message")[0];
  message = messageTag.childNodes[0].nodeValue;
  mdiv = document.getElementById("validationMessage");
  mdiv.innerHTML = message;
}
```

AJAX

Deux manières différentes

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

AJAX

Syntaxe :

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

AJAX

Syntaxe :

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

Les paramètres

- `async` : Par défaut `true` pour dire asynchrone, sinon false
- `type` : Par défaut `GET`, sinon `POST`
- `url` : Par défaut : la page en cours, sinon la page ciblée
- `complete` : Exécute 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 défaut : `application/x-www-form-urlencoded; charset=UTF-8`
- `username, password, data (les paramètres), dataType (html, xml), timeout, ...`

AJAX

L'exemple précédent avec jQuery (une réponse de type html)

```
$('#nom').keyup(function(e){
    $.ajax({
        type: "GET",
        url: "SearchPersonne",
        data : {'nom' : $("#nom").val()},
        dataType: "html",
        success: function(response) {
            $("#validationMessage").html(response);
        }
    });
});
```

AJAX

N'oublions pas de mettre à jour `Vue.js`

```
...
  <script  src="https://code.jquery.com/jquery-3.3.1.js"></
     script>
  <script  src="js/script.js"></script>
</body>
```

Il faut aussi modifier le type de réponse de la servlet

```
response.setContentType("text/html");
```

et aussi supprimer l'évènement `onkeyup` **de la zone texte** `nom`

```
<input type ="text" id="nom" name="nom"/>
```

AJAX

Deux méthodes raccourcies de `$.ajax()`

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

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

AJAX

Syntaxe :

```
// Attention a l'ordre des parametres
$.get(
    {
        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.
}
```

AJAX

Exemple :

```
$('#nom').keyup(function(){
    $.get(
        'SearchPersonne',
        {'nom' : $("#nom").val()},
        displayMessage,
        'html'
    );
    function displayMessage (res) {
        $("#validationMessage").html(res);
    }
});
```

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
- ...

AJAX

Exemple :

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

AJAX

Syntaxe

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

AJAX

Syntaxe

```
$(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`

AJAX

Exemple 1 :

```
$('#nom').keyup(function(){
    var param = $("#nom").val();
    $('#validationMessage').load('SearchPersonne', {nom:param});
});
```

AJAX

Exemple 1 :

```
$('#nom').keyup(function(){
    var param = $("#nom").val();
    $('#validationMessage').load('SearchPersonne', {nom:param});
});
```

Exemple 2 :

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

AJAX

Exemple 1 :

```
$('#nom').keyup(function(){
    var param = $("#nom").val();
    $('#validationMessage').load('SearchPersonne', {nom:param});
});
```

Exemple 2 :

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

Exemple 3 :

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