Secteur Tertiaire Informatique
Filière Production Exploitation Administration

Développer une application Web

PROGRAMMER DES PAGES CLIENT RICHE AJAX

Accueil

Apprentissage

Période en entreprise

Evaluation

Code barre

SOMMAIRE

Création Octobre 2006 (Marc FAYOLLE)
Actualisation et adaptation PHP novembre 2013 (Benoit Hézard)

Objectifs

Ce petit document a comme objectif de donner un éclairage sur l'utilisation d'AJAX. Le fonctionnement d'Ajax et l'utilisation de l'objet XmlHttpRequest sont présentés ; quelques cas d'utilisation standard sont détaillés et pourront être réalisés à titre d'exercices. Les codes sources associés sont donnés.

Ajax ou pas Ajax ?

Le but recherché initialement est de pouvoir *rafraîchir une partie seulement de la page Web déjà affichée par un navigateur.* Mais le principe des conversations entre clients et serveurs Web (HTTP/HTML) est basé sur *l'envoi d'une page HTML complète à chaque requête HTTP* (et certaines pages peuvent être longues...).

HTML propose pourtant « depuis toujours » des moyens de faire varier une partie seulement de l'écran affiché par un navigateur : le découpage en « ***frames*** » ou l'insertion de « ***iframe*** » offrent des moyens *d'afficher simultanément plusieurs pages HTML complètes* dans des portions d'écran. Notons que ces techniques permettent *d'afficher des pages en provenances de plusieurs sites*, ce que ne permet pas (aujourd'hui) Ajax ; ainsi, un site touristique peut « héberger » des pages de Météo France dans une portion d'écran sans le rediriger complètement vers le site Météo France ni ouvrir une nouvelle fenêtre (donc sans trop de risque de « perdre » l'utilisateur).

Au-delà de cette capacité à rafraîchir une portion de la page Web affichée par un navigateur, Ajax offre des fonctionnalités diverses liées à ses principes :

- requêtes HTTP asynchrones (le client Web envoie la requête et passe à autre chose en attendant la réponse) ;
- transfert de données en formats non forcément HTML (XML, JSON) mieux adaptés aux traitements des serveurs ;
- traitements côté client écrits en JavaScript.

Enfin, la prolifération de bibliothèques et frameworks Ajax (voir 1.8) offrant des fonctionnalités toujours plus complètes et simples à mettre en œuvre, font d'Ajax une technique incontournable pour la réalisation de sites professionnels, dynamiques et ergonomiques.

AJAX

I.1 Présentation d'Ajax

Le fonctionnement des applications Web est basé sur *l'envoi au serveur Web, par le navigateur, d'une requête HTTP* ; le traitement du *serveur adresse alors une réponse HTTP contenant la page HTML* rétrouvée sur disque ou générée à la volée. Lors d'un rafraîchissement de page demandé par le navigateur, c'est donc toute la page qui est reconstituée et renvoyée au navigateur.

Ceci n'est pas très performant, en terme de temps de réponse et de trafic réseau. En effet, une simple modification d'une petite partie de notre page web se traduira par la reconstitution et l'envoi de *l'ensemble* de la page.

Ajax permet de s'affranchir de ce concept de requête-réponse *en mode page*, et permet à tout moment d'aller chercher des informations sur le serveur, via requêtes et réponses HTTP, pour les *ajouter dans la page en cours*, ou *modifier des parties de la page courante*, et tout *cela sans avoir à recharger complètement la page*.

Ajax signifie **Asynchronous Javascript And XML**, et permet donc de faire des appels *asynchrones* au serveur depuis le client, et de récupérer un message au format XML renvoyé par le serveur.

Ajax est basé sur l'utilisation d'un composant particulier : l'objet **XmlHttpRequest** . Cet objet va permettre des échanges en format Xml entre le client et le serveur via des requêtes HTTP, qui seront adressées au serveur par les navigateurs via des scripts Javascript.

Attention: Comme les requêtes sont envoyées par Javascript, le support du Javascript ne doit pas être désactivé sur les postes clients….

I.2 L'objet XmlHttpRequest

Cet objet est au cœur du fonctionnement d'Ajax. On va *l'instancier* dans chaque script nécessitant des échanges Ajax. Cet objet va pouvoir *envoyer une requête*, grâce à sa méthode ***open*** (en mode synchrone ou asynchrone, en utilisant la méthode HTTP get ou post, et en passant ou non des paramètres). On va aussi pouvoir *intercepter les changements d'état de cet objet* (propriété ***onreadystatechange***) et l'on pourra ainsi *exécuter le traitement que l'on veut lorsque le traitement serveur sera terminé* (propriété ***readyState = 4***).

A l'origine, la création de l'objet XmlHttpRequest se faisait comme suit car Internet Explorer n'avait pas adopté le même objet que ses concurrents :

```
// on crée tout d'abord l'objet XMLHttpRequest
var xhr=null;
if (window.XMLHttpRequest) // Firefox et autres
        xhr = new XMLHttpRequest();
else if (window.ActiveXObject) // Internet Explorer
{
        try
        {    xhr = new ActiveXObject("Msxml2.XMLHTTP");     } //IE version <5
       catch (e)
       {   xhr = new ActiveXObject("Microsoft.XMLHTTP");  } // IE version >=5
```

```
}
```

Depuis Internet Explorer version 7, Microsoft s'est rallié au standard de fait et ne nécessite plus l'instanciation de son objet ActiveX spécifique. Le code peut devenir alors, pour tous navigateurs :

```
// on crée tout d'abord l'objet XMLHttpRequest
var xhr=null;

try
{   xhr = new XMLHttpRequest();} //tout navigateur" recent"
catch (e)
{… gestion de l'erreur … } // anciens navigateurs : soit refus, soit traitements dégradés sans Ajax
```

Les différentes méthodes et propriétés de l'objet XmlHttpRequest sont les suivantes :

Propriété/Méthode	Description
Méthode ***open(méthode, url, indic)***	Cette méthode initialise une requête au serveur. • Le paramètre ***méthode*** vaut "get" ou "post" : c'est la méthode HTTP d'envoi de la requête. Si l'on est en méthode "get", les paramètres seront passés directement dans l'url. Attention : si l'on est en méthode "post", il faut aussi exécuter la méthode suivante de l'objet XmlHttpRequest : ***setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');*** • Le paramètre ***url*** contient l'adresse du script serveur que l'on veut exécuter (c'est un script jsp, aspx, php…). Ce pourra être par exemple le script serveur qui va générer le 'bout de page HTML' qui va être rafraîchi. Si l'on est en méthode "get", les paramètres doivent être passés dans cette url. • Le paramètre ***indic*** sert à préciser si l'on est en mode synchrone (indic=false) ou asynchrone (indic=true). En mode synchrone, le navigateur attend la réponse du serveur avant de rendre la main à l'internaute. Il est donc préférable d'utiliser le mode ***asynchrone***.
Méthode ***send(paramètres)***	Cette méthode permet l'envoi effectif de la requête au serveur. Si la méthode d'envoi est "get", la chaîne de paramètres passée vaut null. Si l'on est en méthode "post", la chaîne paramètres contient les paramètres sous la forme *nom=valeur;*
Evenement ***onreadystatechange***	Evenement qui se déclenche lors de chaque changement d'état de l'objet XmlHttpRequest. Les différents états possibles sont donnés par la propriété suivante ***readyState***. Cet événement sera notamment utilisé pour exécuter le traitement désiré lorsque le traitement serveur sera terminé (mode asynchrone oblige). Ceci se fera en appelant une fonction spécifique comme suit : ***xhr.onreadystatechange = function()***
Propriété ***readyState***	Cette propriété permet de connaître l'état du traitement de la requête. Les différentes valeurs possibles sont : • ***1*** : la méthode open vient de s'exécuter. La requête est préparée. • ***2*** : la méthode send vient de s'exécuter. La requête est envoyée

	• ***3*** : le traitement serveur s'exécute • ***4*** : Le traitement serveur est terminé, et les données ont été retournées. C'est cet état qui nous intéressera en général pour déclencher le rafraîchissement du 'bout de page HTML'.
Propriétés ***responseText*** ou ***responseXML***	Propriété dans laquelle on va récupérer la réponse du serveur, soit au format Texte, soit au format XML.
Propriétés ***status*** et ***statusText***	Permet de récupérer le status de la réponse HTTP du serveur (200 = OK).

Quelques précisions utiles concernant cet objet XmlHttpRequest :

- le fonctionnement de base est bien en *mode asynchrone*, c'est-à-dire que le navigateur Web exprime sa requête mais n'attend pas la réponse pour continuer son travail ; il reste toutefois possible de programmer des traitements Ajax en mode synchrone ;
- en conséquence, on devra *placer une fonction JavaScript sur le changement d'état de l'objet XmlHttpRequest* de manière à réagir quand le traitement serveur est terminé, c'est-à-dire lorsque l'on a reçu la réponse HTTP ;
- cette fonction est donc déclenchée à chaque changement d'état de l'objet XmlHttpRequest (5 états successifs, voir la doc de référence) ; *on déclenchera le rafraîchissement de l'affichage au moment où l'état vaut 4* ;
- en JavaScript, on peut définir une fonction, en extension, sans la nommer, « à la volée » comme dans le code :

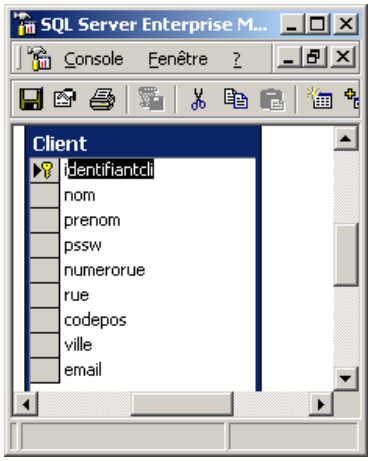
```
xhr.onreadystatechange = function() {
    if (xhr.readyState==4)
        document.getElementById('erreuremail').innerHTML = xhr.responseText;
}
```

-

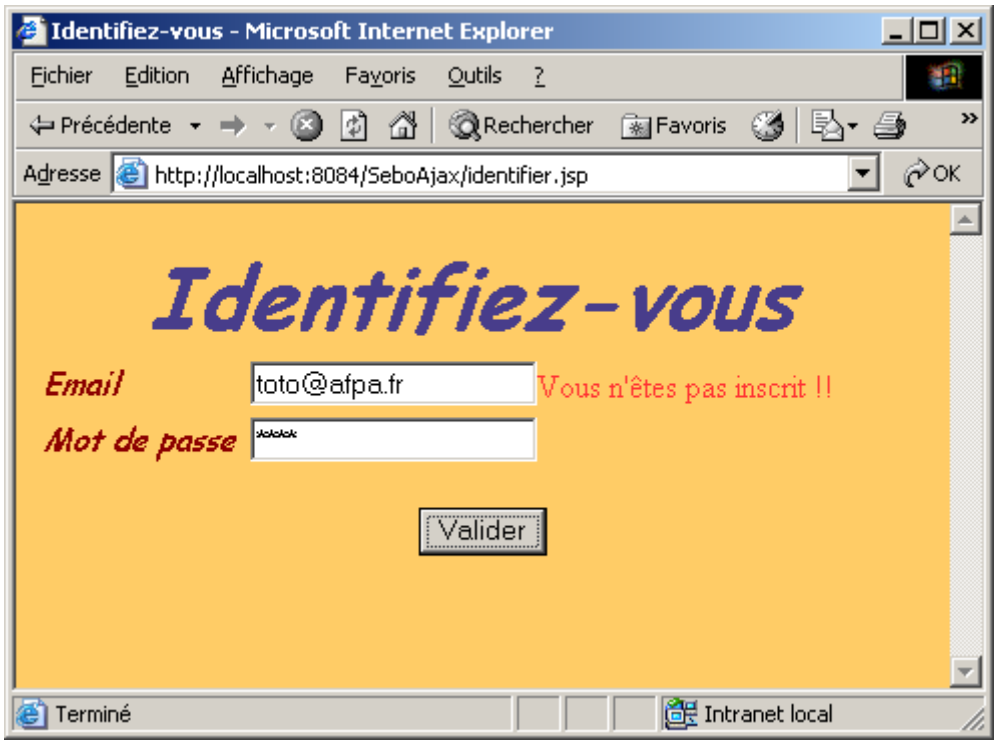
I.3 Utilisation : contrôle serveur avec affichage de messages d'erreurs

Dans une première utilisation, nous allons utiliser Ajax pour effectuer des contrôles serveurs et afficher un message en cas d'anomalie **sans avoir à rafraîchir toute la page**. Nous allons afficher les messages sous forme de labels inclus dans la page HTML, mais ils peuvent être aussi affichés à l'aide de pop-up (alert JavaScript).

Nous allons tester ceci sur l'identification d'un client dans un site de e-commerce. Le client doit s'identifier en saisissant son mail et son mot de passe. On doit vérifier (sur le serveur) que le client existe bien dans la table Client (donnée ci-dessous). Si l'on fait cette vérification avec Ajax, cela évite de rafraîchir l'ensemble de la page.



La saisie du mail et du mot de passe se fait dans la page ci-dessous, et l'envoi des données au serveur est réalisé lorsque l'on clique sur le bouton Valider ; si l'identification est incorrecte, un message d'erreur est affiché.



SOLUTION JSP (CODE NON ACTUALISE)

Nous allons travailler avec du JSP, mais on peut noter qu'Ajax peut être utilisé avec des scripts serveur en ASP.Net, ou en PHP.

Nous allons commencer par créer une page principale ***identifier.jsp.*** Cette page est volontairement structurée de manière 'basique', sans beans, ni tags, avec uniquement du JSP 'de base'. Tout le code nécessaire est dans cette page (sauf les css et le .js).

Appel du *.js* contenant le code Ajax

Emplacement du message d'erreur, qui sera rempli par innerHTML à partir de l'id *erreuremail*.

```
<html>
   <head>
      <title>Identifiez-vous</title>
      <link rel="stylesheet" type="text/css" href="Style.css">
      <script type="text/javascript" src="ajax.js"></script>
   </head>
   <body class=standard>
            <div class=Entete><p class=titregros>Identifiez-vous</p></div>
            <table border=0 >
               <tr><td class=titre>Email</td>
                  <td><INPUT Type="text" id="email" /><Span id="erreuremail" class="erreur" /></td>
               </tr>
               <tr><td class=titre>Mot de passe</td>
                  <td><INPUT Type="text" id="pssw" /><Span id="erreurpssw" class="erreur" /></td>
               </tr>
            </table>
            <div class=centre><BR><INPUT Type="button" value="Valider" onclick="controleIdentAjax();"
                                          language="javascript"/></div>
   </body>
</html>
```

Appel de la fonction Ajax de contrôle de l'identification, sur l'événement *onclick*

La fonction Ajax de contrôle de l'identification ***controleIdentAjax()*** sera définie comme suit (dans un premier temps, on travaille en mode Texte, et notre script serveur nous renvoie du texte) :

```
function controleIdentAjax()
{
   //Cette fonction sera appelée lors du click sur le bouton OK
   // elle sert à contrôler que l'identification est correcte
   // on crée tout d'abord l'objet XMLHttpRequest
   var xhr=null;
   if (window.XMLHttpRequest) // Firefox et autres
      xhr = new XMLHttpRequest();
   else if (window.ActiveXObject) // Internet Explorer
       {
       try
          {   xhr = new ActiveXObject("Msxml2.XMLHTTP");     }
       catch (e)
          {   xhr = new ActiveXObject("Microsoft.XMLHTTP");  }
       }

   //on définit l'appel de la fonction au retour serveur. Lorsque le serveur a terminé son traitement,
   // c'est cette fonction qui sera exécutée. Elle va renvoyer un message d'erreur
   xhr.onreadystatechange = function()
      {
      if (xhr.readyState==4)
          document.getElementById('erreuremail').innerHTML = xhr.responseText;
      }

   //on utilise la méthode GET pour passer les paramètres.
```

Création de l'objet XmlHttpRequest

Remplissage du message d'erreur à partir du responseText de l'objet XmlHttpRequest

```
    xhr.open("GET", "controleIdent.jsp?email="+ document.getElementById('email').value+"&pssw="+
document.getElementById('pssw').value, true);
    xhr.send(null);


}
```

Envoi de la requête. On utilise la méthode get, asynchrone, et appel de *controleIdent.jsp* avec passage des paramètres *email et pssw*

Enfin, le script serveur, ***controleIdent.jsp***, aura le code suivant :

```
<%
//Cette page représente le traitement de l'identification
// elle écrit  un message d'erreur si le client n'est pas  identifié -->

// définir le pilote ODBC
class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
//définir la connection
java.sql.Connection con;
con=java.sql.DriverManager.getConnection("jdbc:odbc:mfcat","visiteur","visiteur");

//execution d'une requete select
java.sql.PreparedStatement comm=con.prepareStatement("Select prenom from client where email=? and
pssw=?");

//donner des valeurs pour l'exécution de la requete
comm.setString (1,request.getParameter("email"));
comm.setString (2,request.getParameter("pssw"));

java.sql.ResultSet rs=comm.executeQuery();

//balayage du ResultSet
if (rs.next())
    out.print("");   //client existant, pas d'erreur'
else
    out.print("Vous n'êtes pas inscrit !! ");

//fermeture de la commande
comm.close();
//fermer la connection
con.close();%>
```

SOLUTION PHP

Nous allons travailler avec du code PHP, mais on peut noter qu'Ajax peut être utilisé avec des scripts serveur en ASP.Net, ou en JSP.

Nous allons commencer par créer une page principale ***identifier.html.*** Cette page est volontairement structurée de manière 'basique', en XHTML. Tout le code nécessaire est dans cette page (sauf les css et le .js).

Appel du *.js* contenant le code Ajax

Emplacement du message d'erreur, qui sera rempli par innerHTML à partir de l'id *erreuremail*.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//Dtd XHTML 1.1//EN"
   "http://www.w3.org/tr/xhtml11/Dtd/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
      <head>
      <meta http-equiv="Content-Type" content="text/html; cha      utf-8" />
      <title>Identifiez-vous</title>
      <link rel="stylesheet" type="text/css" href="style.css" />
      <script type="text/javascript" src="ajax.js"></script>
   </head>
   <body class="standard">
      <div class="Entete"><p class="titregros">Identifiez-vous</p></div>
      <form id="frmLogin">
         <table border="0" >
         <tr><td class="titre">Email</td>
            <td><input type="text" id="email" name="email" /><span id="erreuremail" class="erreur"
/></span></td>
         </tr>
         <tr><td class="titre">Mot de passe</td>
```

```
            <td><input type="password" id="pssw" name="pssw" /><span id="erreurpssw" class="erreur"
/></span></td>
          </tr>
       </table>
       <div class="centre"><input type="button" value="Valider" onclick="controleIdentAjax();" /></div>
     </form>
   </body>
</html>
```

La fonction Ajax de contrôle de l'identification ***controleIdentAjax()*** sera définie comme suit (dans un premier temps, on travaille en mode Texte, et notre script serveur nous renvoie du texte) :

```
function controleIdentAjax()
{
   //Cette fonction sera appelée lors du click sur le bouton OK
   // elle sert à contrôler que l'identification est correcte
   // on crée tout d'abord l'objet XMLHttpRequest
   var xhr;
   try
   {
     xhr = new XMLHttpRequest();  // tout navigateur moderne
   }
   catch (e)
   {
        xhr = null;             // Ajax non supporté
       return ;
   }

   //on utilise la méthode GET pour passer les paramètres.
   xhr.open("GET", "controleIdent.php?email=" + document.getElementById("email").value + "&pssw=" +
document. + document.getElementById("pssw").value, true);
   xhr.send(null);
   //on définit l'appel de la fonction au retour serveur. Lorsque le serveur a terminé son traitement,
   // c'est cette fonction qui sera exécutée. Elle va renvoyer un message d'erreur
   xhr.onreadystatechange = function()
      {
         if (xhr.readyState==4) // on devrait aussi tester xhr.status == 200 soit réponse HTTP OK
         document.getElementById('erreuremail').innerHTML = xhr.responseText;
      };

}
```

Création de l'objet XmlHttpRequest

Envoi de la requête. On utilise la méthode get, asynchrone, et appel de *controleIdent.php* avec passage des paramètres *email et pssw*

Remplissage du message d'erreur à partir du responseText de l'objet XmlHttpRequest

Enfin, le script serveur, ***controleIdent.php***, aura le code suivant :

```
<?php
//Cette page représente le traitement de l'identification
// elle renvoie un message d'erreur si le client n'est pas identifié

// récup des paramètres reçus en querystring
 $email = $_GET["email"] ;
 $pssw = $_GET["pssw"] ;

//définir la connection à la table Client de la BDD Commerce
 $host = "localhost";
 $user="CDI";
 $password = "CDI";
$bdd = "commerce";

$link = mysqli_connect($host, $user, $password, $bdd) or die ("erreur de connection au serveur");
mysqli_set_charset($link,'utf8');

//execution d'une requete select sur la table user
 $query = "select prenom from client where email='" . $email . "' and pssw='" .  $pssw . "';";
 //echo $query; // pour tests
 $result = mysqli_query($link, $query) or die ("erreur sur requete a la table client");

// examen du ResultSet
 $row ;
```

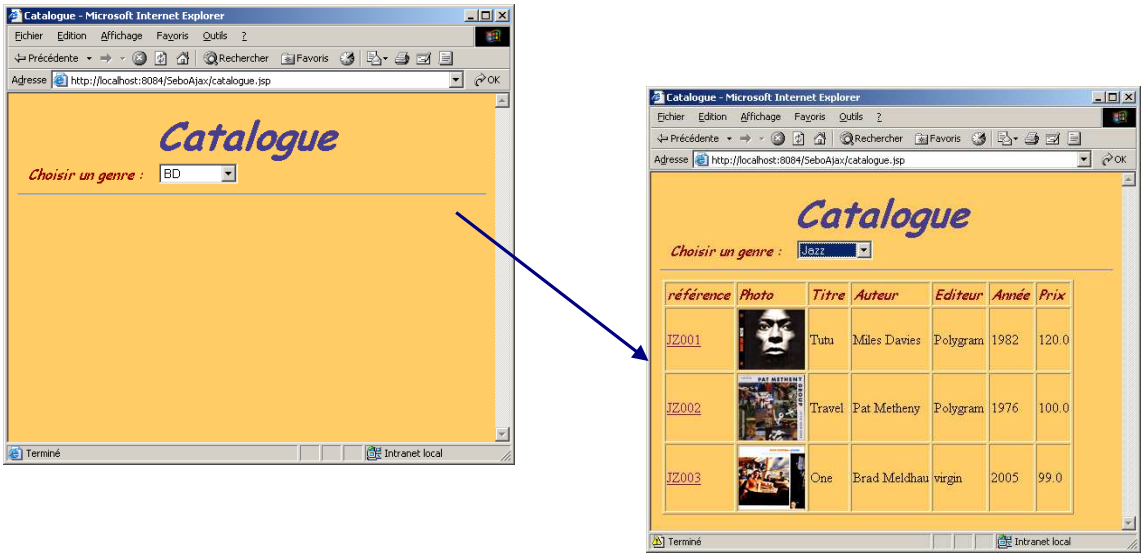
```
if ($row = mysqli_fetch_array($result))      // si lecture renvoie (au moins) une ligne
    echo "OK";   //client existant, pas d'erreur
else
    echo "Vous n'etes pas inscrit !!";  // si lecture n'a pas renvoyé de ligne

//fermeture de la connection
mysqli_free_result($result);
mysqli_close($link);
?>
```

On génère le message texte pour responseText de l'objet XmlHttpRequest comme un simple affichage : echo

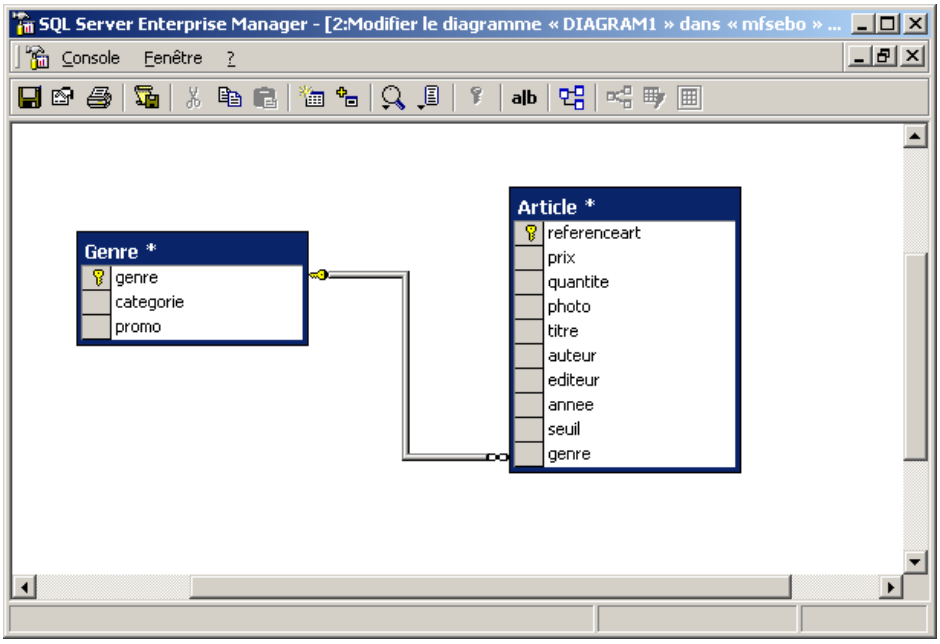
NB : attention de rien écrire en dehors de cette balise php, ni avant < ?php, ni après ?>, même pas un espace ni un retour chariot ; en effet, dans un script serveur php, tout caractère extérieur aux balises < ?php … ?> est envoyé tel quel au navigateur, ce qui perturberait ici le contenu de la propriété .responseText et aboutirait systématiquement au message « Vous n'êtes pas inscrit !! ».

I.4 Utilisation : affichage dynamique d'un tableau

Nous allons reprendre et modifier notre page JSP d'affichage du catalogue. On va trier les articles par genre. Cette page comprend une boite de liste, remplie avec les différents genres d'articles proposés, et un tableau contenant les articles correspondant au genre sélectionné. Chaque fois que l'on sélectionne un nouveau genre, on va rafraîchir uniquement le tableau d'articles en utilisant Ajax.



Les articles et les genres sont stockés dans la base de données mfcat selon le MLD suivant :



SOLUTION JSP (CODE NON ACTUALISE)

Nous allons commencer par créer une page principale ***catalogue.jsp.***

```
<html>
   <head>
      <title>Catalogue</title>
      <link rel="stylesheet" type="text/css" href="Style.css">
      <script type="text/javascript" src="ajax.js"></script>
   </head>
   <body class=standard>
   <div class=Entete><p class=titregros>Catalogue</p></div>

      <%
      // définir le pilote ODBC
      class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      //définir la connection
      java.sql.Connection con;
      con=java.sql.DriverManager.getConnection("jdbc:odbc:mfcat","visiteur","visiteur");

      //execution d'une requete select pour rempli la combo
         java.sql.PreparedStatement comm=con.prepareStatement("Select genre from Genre");
         java.sql.ResultSet rs=comm.executeQuery();
   %>


      <table border=0>
         <tr>
            <td width=150px valign=top align=center>
               <div ><p class=titre>Choisir un genre :</p></div>
            </td>
            <td>
               <!-- Sur l'événement onchange, on appelle la fonction ajax qui va rafraichir le catalogue -->
               <select id="genre" onchange="rafraichirCatAjax();" language="javascript" >
                <%while (rs.next())
               {%>
                <option><%=rs.getString("genre")%></option>
               <%}%>
               </select>
             </td>
         </tr>
      </table>
      <HR>
      <table>
         <tr>
            <td>
            <!--Ici on définit l'emplacement du catalogue, tel qu'il sera généré à chaque modif du genre
            on le remplira par innerHTML -->
            <div id="cat">
            </div>
            </td>
         </tr>
       </table>
      <%//fermer la connection
         con.close(); %>
   </body>
</html>
```

Inclusion du *.js* contenant le code Ajax

Appel de la fonction Ajax de rafraîchissement du catalogue, sur l'événement *onchange*

Emplacement du tableau contenant le catalogue, qui sera rempli par innerHTML à partir de l'id *cat*.

La fonction Ajax de rafraîchissement du catalogue ***rafraichirCatAjax()*** sera définie comme suit (dans un premier temps, on travaille en mode Texte, et notre script serveur nous renvoie du texte) :

```
function rafraichirCatAjax()
{
   //Cette fonction sera appelée lors du changement de genre dans la combo
   // elle sert à rafraîchir le catalogue en fonction du genre sélectionné
   // on crée tout d'abord l'objet XMLHttpRequest
   var xhr=null;
   if (window.XMLHttpRequest) // Firefox et autres
      xhr = new XMLHttpRequest();
```

Création de l'objet XmlHttpRequest

```
    else if (window.ActiveXObject) // Internet Explorer
        {
        try
           {   xhr = new ActiveXObject("Msxml2.XMLHTTP");     }
        catch (e)
           {   xhr = new ActiveXObject("Microsoft.XMLHTTP");  }
        }

    //on définit l'appel de la fonction au retour serveur. Lorsque le serveur a terminé son traitement,
    // c'est cette fonction qui sera exécutée. Elle va remplir le div qui s'appelle cat par le text retourné dans le
    //xhr
    xhr.onreadystatechange = function()
       {
        if (xhr.readyState==4)
          document.getElementById('cat').innerHTML = xhr.responseText;
       }

    //on ouvre le fichier cat.jsp (qui 'génère' le catalogue).
    xhr.open("POST", "cat.jsp", true);
    // mettre ça si on est en POST
    xhr.setRequestHeader('Content-Type','application/x-www-form-urlencoded');
    //on passe le genre en paramètre (on prend le genre sélectionné)
    sel=document.getElementById("genre");
    gr=sel.options[sel.selectedIndex].text;
    xhr.send("genre="+gr);

}
```

Remplissage du tableau cat à partir du responseText de l'objet XmlHttpRequest

Préparation de la requête : méthode post, asynchrone, et appel de *cat.jsp* avec passage d'un paramètre *genre*

Récupération du genre sélectionné

Enfin, le script serveur appelé, ***cat.jsp***, aura le code suivant :

```
<!--Cette page représente le 'petit bout' de page HTML (JSP) que l'on va rafraîchir suite à maj du genre -->
<%
// définir le pilote ODBC
class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
//définir la connection
java.sql.Connection con;
con=java.sql.DriverManager.getConnection("jdbc:odbc:mfcat","visiteur","visiteur");

//exécution d'une requête select pour rempli la combo
//cette requête est paramétrée et récupère le paramètre passé en méthode POST
java.sql.PreparedStatement comm=con.prepareStatement("Select referenceart, photo, titre, auteur, editeur,
                                annee, prix from Article where genre='"+request.getParameter("genre")+"'");
java.sql.ResultSet rs=comm.executeQuery();
%>
<table border=1>
            <tr>
                <td>r&eacute;f&eacute;rence</td><td>Photo</td><td>Titre</td>
                <td>Auteur</td><td>Editeur</td><td>Ann&eacute;e</td><td>Prix</td>
              </tr>
                <%
                while (rs.next())
                {%>
               <tr>
                 <td><A HREF="#" ><%=rs.getString("referenceart")%></A></td>
                   <td><IMG SRC="images/<%=rs.getString("photo")%>"></td>
                    <td><%=rs.getString("titre")%></td>
                    <td><%=rs.getString("auteur")%></td>
                    <td><%=rs.getString("editeur")%></td>
                     <td><%=rs.getInt("annee")%></td>
                     <td><%=rs.getFloat("prix")%></td>
                </tr>
                  <%}
                  %>
            </table>
 <%//fermer la connection
 con.close(); %>
```

SOLUTION PHP

Nous allons commencer par créer une page principale ***catalogue.php*** ; cette fois, il s'agit bien d'une page php car il faut alimenter la boite de liste avec les différents genres d'articles recherchés dans la base de données.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//Dtd XHTML 1.1//EN"
   "http://www.w3.org/tr/xhtml11/Dtd/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
      <title>Catalogue</title>
      <link rel="stylesheet" type="text/css" href="Style.css" />
      <script type="text/javascript" src="ajax.js"></script>
   </head>
   <body class="standard">
            <div class="Entete"><p class="titregros">Catalogue</p></div>
<?php
//définir la connection      à la table Genre de la BDD Commerce
$host = 'localhost';
$user='CDI';
$password = 'CDI';
$bdd = 'commerce';

$link = mysqli_connect($host, $user, $password, $bdd) or die ("erreur de connection au serveur");
mysqli_set_charset($link,'utf8');

//execution d'une requete select sur la table user
$query = 'Select * from genre order by categorie';
$result = mysqli_query($link, $query) or die ("erreur sur requete à la table genre");
$row;    // ligne d'enregistrement lu
?>
         <form id="frmGenre">
          <table border="0">
              <tr>
                 <td width="150px" valign="top" align="center">
                    <div ><p class="titre">Choisir un genre :</p></div>
                 </td>
                 <td>
                   <!-- Sur l'événement onchange, on appelle la fonction ajax qui va rafraichir le catalogue -->
                   <select id="genre" onchange="rafraichirCatAjax();" >
                    <!-- première option invitant à changer  -->
                    <option value="erreur">&lt;Choisir&gt;</option>
<?php while ($row = mysqli_fetch_assoc($result)){?>
                    <option><?php echo $row["genre"] ; ?></option>
<?php }?>
                   </select>
                 </td>
              </tr>
          </table>
         </form>
         <hr />
         <table>
          <tr>
             <td>
              <!--Ici on définit l'emplacement du catalogue, tel qu'il sera généré à chaque modif du genre
 on le remplira par innerHTML -->
               <div id="cat">
               </div>
              </td>
          </tr>
         </table>
<?php //fermer la connection
mysqli_free_result($result);
mysqli_close($link);
?>
   </body>
</html>
```

Liaison avec le .js contenant le code Ajax

Appel de la fonction Ajax de rafraîchissement du catalogue, sur l'événement *onchange*

Emplacement du tableau HTML contenant le catalogue, qui sera rempli par innerHTML à partir de l'id *cat*.

La fonction Ajax de rafraîchissement du catalogue ***rafraichirCatAjax()*** sera définie comme suit (dans un premier temps, on travaille en mode Texte, et notre script serveur nous renvoie du texte) :

```
function rafraichirCatAjax()
{
   //Cette fonction sera appelée lors du changement de genre dans la combo
   // elle sert à rafraîchir le catalogue en fonction du genre sélectionné
   // on crée tout d'abord l'objet XMLHttpRequest
   var xhr;
   try
   {
                  xhr = new XMLHttpRequest();  // tout navigateur moderne
         }
   catch (e)
   {
            xhr = null;         // Ajax non supporté
            return ;
         }

   //on définit l'appel de la fonction au retour serveur. Lorsque le serveur a terminé son traitement,
   // c'est cette fonction qui sera exécutée. Elle va remplir le div qui s'appelle cat par le text retourné dans le
   //xhr
   xhr.onreadystatechange = function()
      {
        if (xhr.readyState==4)
          document.getElementById('cat').innerHTML = xhr.responseText;
      }

   //on ouvre le fichier catalogueGenre.php (qui 'génère' l'extrait de catalogue).
   xhr.open("POST", "catalogueGenre.php", true);
   // mettre ça si on est en POST
   xhr.setRequestHeader('Content-Type','application/x-www-form-urlencoded');
   //on passe le genre en paramètre (on prend le genre sélectionné)
   sel=document.getElementById("genre");
   gr= sel.value;  // ou sel.options[sel.selectedIndex].text;
   xhr.send("genre="+gr);
}
```

Création de l'objet XmlHttpRequest

Remplissage du tableau cat à partir du responseText de l'objet XmlHttpRequest

Préparation de la requête : méthode post, asynchrone, et appel de *catalogueGenre.php* avec passage d'un paramètre

Récupération et transmission du genre sélectionné

Enfin, le script serveur ***catalogueGenre.php***, appelé par JavaScript à chaque changement dans la boite de liste, aura le code suivant :

```
<?php

// cette page génère le contenu du tableau HTML listant les films du genre sélectionné
// récup du paramètre reçu en querystring
 $genre = $_POST["genre"]          ;

//définir la connection      à la table Article de la BDD Commerce
 $host = "localhost";
 $user="CDI";
 $password = "CDI";
$bdd = "commerce";

$link = mysqli_connect($host, $user, $password, $bdd) or die ("erreur de connection au serveur");
mysqli_set_charset($link,'utf8');

//execution d'une requete select sur la table user
 $query = "select * from article where genre='" . $genre . "' order by titre ;";
 //echo $query; // pour test
 $result = mysqli_query($link, $query) or die ("erreur sur requete a la table client");
// code HTML pour affichage début du tableau
 ?>
<table border="1">
     <tr>
<th>Reference</th><th>Photo</th><th>Titre</th><th>Auteur</th><th>Editeur</th><th>Annee</th><t
h>Prix</th>
     </tr>
```

Accès à la base de données

Pour commencer, entête du tableau HTML

```
<?php
// parcours du ResultSet
while($ligne = mysqli_fetch_array($result))
{
   extract($ligne); // découpage auto des champs
?>
    <tr>
      <td><a href="detail.php?refart='<?php echo $referenceart; ?>'"><?php echo
$referenceart;?></a></td>
      <td><img src="images/<?php echo $photo; ?>" /></td>
      <td><?php echo $titre;?></td>
      <td><?php echo $auteur;?></td>
      <td><?php echo $editeur;?></td>
      <td><?php echo $annee;?></td>
      <td><?php echo $prix;?></td>
    </tr>
<?php
} // fin de boucle while
?>
</table>
<?php
// fermer la connection
mysqli_free_result($result);
mysqli_close($link);
?>
```

Pour chaque ligne lue dans la base de données, génération d'une ligne de tableau HTML

I.5 Utilisation : affichage d'un tableau en utilisant une réponse XML

SOLUTION JSP (CODE NON ACTUALISE)

Nous allons reprendre l'exercice précédent, mais en utilisant cette fois la réponse XML (*le 'X' de Ajax*). Nous allons renvoyer le catalogue (après modification du genre) au format xml, et nous lui appliquerons une feuille de style xslt pour l'afficher.

La page ***catalogueXml.jsp*** sera la même que la précédente. Seul l'endroit ou l'on va afficher le catalogue contiendra une balise div supplémentaire (id= catalogue), afin d'assurer un fonctionnement correct sous FireFox

```
……
 <!--Ici on définit le fichier XMl qui servira à remplir la table catalogue -->
<div id="cat">
  <div id="catalogue"></div>
</div>
……
```

La fonction Ajax de rafraîchissement du catalogue ***rafraichirCatAjax()*** sera définie comme suit (on va utiliser le retour XML). Le fichier xml récupéré sera transformé à l'aide d'une feuille de style xslt.

```
function rafraichirCatXmlAjax()
{
   //Cette fonction sera appelée lors du changement de genre dans la combo
   // elle sert à rafraîchir le catalogue en fonction du genre sélectionné
   // on crée tout d'abord l'objet XMLHttpRequest
   var xhr=null;
   if (window.XMLHttpRequest) // Firefox et autres
      xhr = new XMLHttpRequest();
   else if (window.ActiveXObject) // Internet Explorer
       {
       try
          {   xhr = new ActiveXObject("Msxml2.XMLHTTP");     }
       catch (e)
          {   xhr = new ActiveXObject("Microsoft.XMLHTTP");  }
       }

   //on définit l'appel de la fonction au retour serveur. Lorsque le serveur a terminé son traitement,
```

```
   // c'est cette fonction qui sera exécutée. Elle va remplir le div qui s'appelle cat à partir du xml retourné dans
  //le xhr
     xhr.onreadystatechange = function()
       {
       if (xhr.readyState==4)
          {

             //sous Internet Explorer
             if (window.ActiveXObject)
             {
                 //on va appliquer une feuille de style XSLT au document XML récupéré
                 var style=new ActiveXObject("Microsoft.xmldom");
                 style.async=false;
                 style.load("cat.xsl");
                 document.getElementById("cat").innerHTML =
                                        xhr.responseXML.transformNode(style.documentElement);
             }

             //sous FireFox
             else if (window.XSLTProcessor)
             {
                 //Chargement de la feuille de style
                 var style = document.implementation.createDocument('', '', null);
                 style.async=false;
                 style.load("cat.xsl");
                 //transormation du xml à l'aide de la feuille de style xsl
                 var xslt = new XSLTProcessor();
                 xslt.importStylesheet(style);
                 var frag = xslt.transformToFragment(xhr.responseXML, document);
                 // Ajouter le fragment au document courant au niveau de la balise spécifiée (ici cat)
                 document.getElementById("cat").removeChild(document.getElementById("catalogue"));
                 document.getElementById("cat").appendChild(frag);
             }
          }
       }

   //on ouvre le fichier catXml.jsp (qui 'génère' le catalogue).
   //se mettre en méthode POST pour passer les paramètres. Sinon il fau
   xhr.open("POST", "catXml.jsp", true);
   // mettre ça si on est en POST
   xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
   //on passe le genre en paramètre (on prend le genre sélectionné)
   sel=document.getElementById("genre");
   gr=sel.options[sel.selectedIndex].text;
   xhr.send("genre="+gr);


}
```

On applique la feuille de style cat.xsl.
Attention, le code est différent selon le navigateur utilisé !!

Noter le code de transformation avec FireFox…

Avec FireFox, on utilise appendChild, et l'on doit donc supprimer l'élément avant le rajout (d'où l'utilité de la balise div id=catalogue….)

Le script serveur appelé, ***catXml.jsp***, aura le code suivant :

```
<%
//donner le format xml au fichier renvoyé
response.setContentType("text/xml");
// définir le pilote ODBC
class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
//définir la connection
java.sql.Connection con;
con=java.sql.DriverManager.getConnection("jdbc:odbc:mfcat","visiteur","visiteur");

//exécution d'une requête select
//cette requete est paramétrée et récupère le paramètre passé en méthode POST
java.sql.PreparedStatement comm=con.prepareStatement("Select
referenceart,photo,titre,auteur,editeur,annee,prix from Article where
genre='"+request.getParameter("genre")+"'");
java.sql.ResultSet rs=comm.executeQuery();

out.print("<?xml version='1.0' encoding='ISO-8859-1' ?>");
   out.println ("<cat>");
```

Indique que l'on renvoie un document xml

On fabrique ici le document xml renvoyé.
Attention, il doit être bien-formé !!

```
   while (rs.next())
   {
      out.println ("<ligne>");
      out.println ("<referenceart>"+rs.getString("referenceart")+"</referenceart>");
      out.println ("<photo>"+rs.getString("photo")+"</photo>");
      out.println ("<titre>"+rs.getString("titre")+"</titre>");
      out.println ("<auteur>"+rs.getString("auteur")+"</auteur>");
      out.println ("<editeur>"+rs.getString("editeur")+"</editeur>");
      out.println ("<annee></annee>");
      out.println ("<prix></prix>");
      out.println ("</ligne>");
   }
   out.println ("</cat>");
 con.close(); %>
```

Et enfin, la feuille xslt ***cat.xsl***, sera comme suit :

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

<xsl:template match="/">
        <div id="catalogue">
                <hr></hr>
                <br></br>
                <table border="1">
                <tr>
                        <td class="titre">Reference</td>
                        <td class="titre">Photo</td>
                        <td class="titre">Titre </td>
                        <td class="titre">Auteur</td>
                        <td class="titre">Editeur</td>
                </tr>
                <xsl:apply-templates />
                </table>
      </div>
</xsl:template>

<xsl:template match="ligne">
        <tr>
                <td><a href="#" ><xsl:value-of select="./referenceart" /></a></td>
                <td><img>
                        <xsl:attribute name="src">
                                images/<xsl:value-of select="./photo" />
                        </xsl:attribute>
                </img></td>
                <td><xsl:value-of select="./titre" /></td>
                <td><xsl:value-of select="./auteur" /> </td>
                <td><xsl:value-of select="./editeur" /></td>
        </tr>
        </xsl:template>
</xsl:stylesheet>
```

I.6 Utilisation : affichage d'un tableau en utilisant une réponse JSON

Nous allons reprendre l'exercice précédent, mais en utilisant cette fois une réponse JSON.

Nous allons renvoyer un fichier JSON composé d'un tableau de ligne, chaque ligne contenant la référence de l'article, la photo, le titre, etc…. ce fichier aura le format suivant :

SOLUTION JSP (CODE NON ACTUALISE)

```
{'ligne':
  [
        {       'referenceart':'JZ001 ',
                'photo':'davies.gif ',
                'titre':'Tutu ',
                'auteur':'Miles Davies ',
                'editeur':'Polygram ',
                'prix':'120'
        },

        {       'referenceart':'JZ002 ',
                'photo':'metheny.jpg ',
                'titre':'Travel ',
                'auteur':'Pat Metheny ',
                'editeur':'Polygram ',
                'prix':'100'
        },

        {       'referenceart':'JZ003 ',
                'photo':'meldhau.gif ',
                'titre':'One ',
                'auteur':'Brad Meldhau ',
                'editeur':'virgin ',
                'prix':'99'
        }
  ]
}
```

et nous récupérerons de fichier JSON sur le poste client et l'afficherons directement depuis le javascript.

La page ***catalogueJson.jsp*** sera la même que la précédente.

```
    ……
<!--Ici on définit l'emplacement ou sera affiché le catalogue -->
  <div id="cat" style="left: 40px; position: absolute; top: 200px">
  </div>
  ……
```

La fonction Ajax de rafraîchissement du catalogue ***rafraichirCatJsonAjax()*** sera définie comme suit :

```
function rafraichirCatJsonAjax()
{
  //Cette fonction sera appelée lors du changement de genre dans la combo
  // On utilise ici un retour au format JSON
  var xhr=null;
  if (window.XMLHttpRequest) // Firefox et autres
     xhr = new XMLHttpRequest();
  else if (window.ActiveXObject) // Internet Explorer
     {
     try
       {   xhr = new ActiveXObject("Msxml2.XMLHTTP");     }
     catch (e)
       {   xhr = new ActiveXObject("Microsoft.XMLHTTP");  }
     }
```

```
   //on définit l'appel de la fonction au retour serveur. Lorsque le serveur a terminé son traitement,
   // c'est cette fonction qui sera exécutée. Elle va remplir le div qui s'appelle cat par le text retourné dans le
xhr
   xhr.onreadystatechange = function()
   {
   if (xhr.readyState==4){
      if (xhr.status == 200)
      {
        //récupération du fichier JSON
        var cat = eval("(" + xhr.responseText + ")");
        var res
        //on formate l'affichage HTML directement sur le poste client
        res = "<table border=1><thead><tr><th class='titre'>referenceart
            </th><th class='titre'>photo</th><th class='titre'>titre</th>
            <th class='titre'>auteur</th><th class='titre'>editeur</th>
            <th class='titre'>prix</th></tr>";
        res = res + "</thead><tbody>";
        for(c in cat.ligne)
        {
           res = res+"<tr><td><a href='#' >" + cat.ligne[c].referenceart
                + "</a></td><td><img src='images/" + cat.ligne[c].photo
                +"' /></td><td>" + cat.ligne[c].titre +"</td>"+
                "<td>" + cat.ligne[c].auteur +"</td><td>" +
                cat.ligne[c].editeur +"</td><td>" + cat.ligne[c].prix +
                "</td></tr>";
         }
         res = res +  "</tbody></table>";
         document.getElementById("cat").innerHTML=res;
        }
      }
      }

   //on ouvre le fichier catJson.aspx (qui 'génere' le catalogue).
   //passage des parametres en méthode Post
   xhr.open("POST", "catJson.jsp", true);
   // mettre ça si on est en POST
   xhr.setRequestHeader('Content-Type','application/x-www-form-
                                          urlencoded');
   //on passe le genre en parametre (on prend le genre séléctionné)
   sel=document.getElementById("genre");
   gr=sel.options[sel.selectedIndex].text;
   xhr.send("genre="+gr);
}
```

On récupère le fichier JSON avec la fonction eval

Récupération des données JSON par balayage des lignes

Le script serveur appelé, ***catJson.jsp***, aura le code suivant :

```
<%
// définir le pilote ODBC
class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
//définir la connection (définir le lien ODBC correspondant)
java.sql.Connection con;
con=java.sql.DriverManager.getConnection("jdbc:odbc:mfcat","visiteur","visiteur");

//execution d'une requete select pour rempli la combo
//cette requete est paramétrée et récupere le parametre passé en méthode POST
java.sql.PreparedStatement comm=con.prepareStatement("Select
referenceart,photo,titre,auteur,editeur,annee,prix from Article where
genre='"+request.getParameter("genre")+"'");
java.sql.ResultSet rs=comm.executeQuery();

StringBuffer resJson;
resJson=new StringBuffer();
resJson.append("{'ligne':[ ");
boolean suite;
suite=rs.next();

while (suite)
{
```

Création du fichier JSON

```
        resJson.append("{'referenceart':'" + rs.getString("referenceart") + "',");
        resJson.append("'photo':'" + rs.getString("photo") + "',");
        resJson.append("'titre':'" + rs.getString("titre") + "',");
        resJson.append("'auteur':'" + rs.getString("auteur") + "',");
        resJson.append("'editeur':'" + rs.getString("editeur") + "',");
        resJson.append("'prix':'" + rs.getString("prix") + "'}");
        suite = rs.next();
        if (suite)
            resJson.append(",");
}
 resJson.append("]}");

 out.println(resJson);
//fermer la connection
 con.close(); %>
```

On ne rajoute pas de virgule à la fin

Retour du fichier JSON

I.7 Utilisation : AEP Ajax Extensible Page

Cette utilisation d'Ajax permet de créer des pages dynamiques dont le contenu se développe ou se réduit lorsque l'utilisateur clic à un endroit de la page (sur un bouton, un lien, une image…).
A.E.P. fait partie du Web 2.0.

La partie dynamique de la page (le contenu qui se développe ou se réduit) pourra être soit incluse dans la page, soit être stockée dans un fichier texte indépendant.

Les avantages d'un chargement dynamique d'une partie de page sont multiples :

- On peut rendre la page adaptable au lecteur (un utilisateur averti lira la page réduite, un novice lira la page développée)
- Cela permet de réutiliser un texte dans plusieurs pages
- Cela permet de gérer du contenu dynamique : le contenu qui change souvent sera stocké dans un fichier qui sera inséré dans la page

Il est d'usage de prévenir le lecteur que toutes les données de la page ne sont pas affichées, et que la page peut être développée.

L'intérêt des pages extensible réside dans le fait que l'on peut charger dynamiquement des données accessible depuis le serveur (données d'une base de données, données contenues dans un fichier texte….).

L'exemple qui suit est inspiré librement du tutoriel de Denis Sureau, disponible à l'adresse http://www.xul.fr/ajax/extensible/ et que je vous invite à consulter.

Le principe de fonctionnement est le suivant :

Dans la page jsp, sont présentes :
Une balise div nommée *texte*, affichée, qui contient le texte affiché, et sur laquelle on va cliquer, ce qui déclenchera une fonction *inverser().* Cette balise possède un attribut title, qui nous permettra de savoir si le texte est déployé (*title = 'affiche'*) ou résumé (*title='cache'*).
Et une seconde balise div nommée *sauve* qui est hidden, et qui contient le texte affiché lorsque la page est résumée..
Lorsque l'on clique sur le span, on appelle la fonction *inverser(),* qui va tester si la page est déployée ou résumée (par le biais du title), et selon le cas, va appeler l'objet XmlHttpRequest qui va renvoyer les informations à afficher, ou afficher simplement le texte de base présent dans la balise *sauve*.

Nous allons développer une page extensible suivante :

Qui lorsqu'on la développe, devient :

SOLUTION JSP (CODE NON ACTUALISE)

Nous allons commencer par créer une page principale ***jazz.jsp***

```
<html>
 <head>
      <title>Jazz</title>
      <link rel="stylesheet" type="text/css" href="Style.css">
      <script type="text/javascript" src="ajax.js"></script>
  </head>
<body class=standard>
         <div class=Entete><p class=titregros>Miles Davis</p></div>
         <p class=titre>Trompettiste et compositeur de jazz</p>
         <HR>
         Né à Alton, Illinois le 25 mai 1926<BR>
         Décédé à Santa Monica, Californie le 28 septembre 1991<hr>
          <span class=titremoyen>« Pourquoi jouer tant de notes alors qu'il suffit de jouer les meilleures ? »</span>
         <span class=titre>Miles Davis</span>
         <p>
         <div class=titre id="texte" title="cache" onclick="inverser();" language="javascript">Voir la discographie + </div>
         <div id="sauve" style="display:none">Voir la discographie + </div>    <p>
 <br>
  C'est dans une famille bourgeoise et mélomane que Miles Davis apprend la trompette. En 1945 il réalise son rêve et rejoint la Julliard School of Music à New York. Il rencontre, entre autres grands noms du jazz, Thelonious Monk et Dizzie Gillespie.
  A 19 ans il enregistre déjà avec le saxophoniste Charlie Parker. En 1949, il participe au festival international de jazz à Paris. A son retour, enregistrements et performances scéniques se succèdent, notamment grâce au quintette qu'il forme avec John Coltrane. Ses succès, 'Miles Ahead' et 'Porgy and Bess' datent de cette époque. Quand, en 1957, il enregistre à Paris la musique du film 'Ascenseur pour l'échafaud' il a alors atteint une renommée qui dépasse le cercle des amateurs de jazz. Mais des soucis de santé et les excès d'une vie par trop sulfureuse l'éloignent de la scène jusque dans les années 1980 où il fait un retour sous le signe du jazz rock. Il continue ainsi de traverser, toujours avec talent, les styles de la musique afro-américaine. On retrouve, en effet, du cool jazz à la fusion en passant par le hard bop, la sonorité singulière de sa trompette et une utilisation unique du silence. Ce parcours incroyable fait de lui un des musiciens de jazz les plus emblématiques.
```

Inclusion du *.js* contenant le code Ajax

Appel de la fonction d'inversion

```
  </body>
</html>
```

La fonction Ajax d'affichage de la discographie ***inverser()*** sera définie comme suit :

```
function inverser()
{
   //Cette fonction sera appelée lors du click sur le bouton 'discographie'
   // elle sert à afficher la discographie de Miles Davies
   // on crée tout d'abord l'objet XMLHttpRequest
   var xhr=null;
   if (window.XMLHttpRequest) // Firefox et autres
      xhr = new XMLHttpRequest();
   else if (window.ActiveXObject) // Internet Explorer
       {
       try
          {   xhr = new ActiveXObject("Msxml2.XMLHTTP");     }
       catch (e)
          {   xhr = new ActiveXObject("Microsoft.XMLHTTP");  }
       }

   //on définit l'appel de la fonction au retour serveur. Lorsque le serveur a terminé son traitement,
   // c'est cette fonction qui sera exécutée. Elle va tester si la page est déployée ou résumée. Ce test se
  // fait en utilisant l'attribut title de la balise div texte.

  xhr.onreadystatechange = function()
       {
       if (xhr.readyState==4)
          {
          // la discographie ne sera rafraichie que si on va l'afficher
          if(document.getElementById('texte').getAttribute("title") == "cache")
           {       document.getElementById('texte').innerHTML = xhr.responseText;
                   document.getElementById('texte').setAttribute("title","affiche");
           }
          else
           {
                 document.getElementById('texte').setAttribute("title","cache");
                 document.getElementById('texte').innerHTML   =   document.getElementById('sauve').innerHTML;

          }
          }
      }

    //on ouvre le fichier miles.jsp (qui 'génere' la discographie).
   xhr.open("GET", "miles.jsp", true);
   xhr.send(null);

}
```

Test de l'état de la page (déployée ou résumée)

Récupération du texte par défaut dans le champ caché *sauve*

Nous allons enfin créer la page ***miles.jsp*** qui va générer la discographie :

```
<%
// définir le pilote ODBC
class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
//définir la connection
java.sql.Connection con;
con=java.sql.DriverManager.getConnection("jdbc:odbc:mfcat","visiteur","visiteur");

//execution d'une requete select pour rempli la combo
//cette requete est paramétrée et récupere le parametre passé en méthode POST
java.sql.PreparedStatement comm=con.prepareStatement("Select titre, photo,editeur,annee from Article where
auteur='Miles Davies'");
java.sql.ResultSet rs=comm.executeQuery();
%>
<Div class="titre">Cacher la discographie +</div><BR>
<table border=1>
              <tr>
                  <td class="titre">Titre</td><td class="titre">Photo</td>
                   <td class="titre">Editeur</td><td class="titre">Ann&eacute;e</td>
               </tr>
```

```
                <%
                while (rs.next())
                {%>
               <tr>
                    <td><%=rs.getString("titre")%></td>
                    <td><IMG SRC="images/<%=rs.getString("photo")%>"></td>
                    <td><%=rs.getString("editeur")%></td>
                    <td><%=rs.getInt("annee")%></td>
                </tr>
                  <%}
                  %>
            </table>
<%//fermer la connection
con.close(); %>
```

I.8 Utilisation des frameworks AJAX

Ajax est souvent utilisé en combinaison avec des frameworks, qui masquent la mécanique Ajax et aident le développeur en fournissant des composants qui fonctionnent directement en Ajax.

Ces frameworks sont de deux types : **les frameworks clients** ou les **framework serveurs.**

- **Les frameworks client Javascript:**
  - ce sont des bibliothèques Javascript
  - contiennent des composants graphiques (drag and drop, tableaux, arbres…)
  - fournissent des fonctions qui simplifient l'appel d'Ajax
  - Se traduisent par le lien vers un fichier .js externe dans le code HTML
  - Exemples : **Rico, Dojo, Script.aculo.us**, **JQuery**… Ils sont tous basés sur le framework Javascript **Prototype**.

- **Les frameworks serveur:**
  - nécessitent une installation et une génération coté serveur
  - permettent d'envoyer directement un objet serveur et le rend accessible en Javascript
  - permettent d'utiliser des web services
  - Peuvent fournir des composants serveurs qui génèrent tout le code Javascript nécessaire
  - Exemples **: DWR, AjaxTags, ASP.Net Ajax (Atlas), Xajax, GWT (Google Web Toolkit)**

Notons que JQuery est actuellement très populaire dans le monde du développement Web (cela semble même un incontournable). Outre des fonctionnalités purement Ajax (masquant la complexité des variantes entre navigateurs par exemple), JQuery offre une multitude d'effets graphiques et interactifs qui permettent d'enrichir affichage et navigation à peu de frais (menus accordéons, effets de transparence, de fondus…).

MISE EN PRATIQUE DE GOOGLE MAPS (VERSION 3)

Le site web de Google (https://developers.google.com/maps/documentation/javascript/tutorial/) délivre toute information utile concernant la mise en œuvre de l'application Google Maps dans un site Web : tutoriaux, exemples de code, documentation de référence… Il est donc très facile de placer une image de géo-localisation dans une page Web quelconque.

Principes :

- toutes les données géographiques et les algorithmes sont hébergés sur les serveurs de Google (votre application dépend donc de ce prestataire…) ;
- les pages Web doivent simplement télécharger le code javascript minimal nécessaire pour faire le lien avec les services de Google et instancier les objets nécessaires ;
- une HTML doit être prévue pour que le service en ligne de Google y place l'image finale.

Partie <head> de la page HTML :

```
<!-- styles générés par Google Maps ; personnalisable... -->
<style type="text/css">
  html { height: 100% }
  body { height: 100%; margin: 0px; padding: 0px }
  #map_canvas { height: 100% }
</style>

<script type="text/javascript"
    src="http://maps.google.com/maps/api/js?sensor=false"> < !--appel du JS Google -->
</script>
```

puis le code à éxécuter côté client (ici, il s'agit d'une version personnalisée et commentée) :

```
<script type="text/javascript">
// variables globales
var map; // la carte google maps
var infowindow; // l'info bulle

// fonction principale appelée sur l'événement onload de la page
function initialize() {
// NB : google offre aussi un service de calcul de coordonnées à l'aide de l'objet Geocoder...
var latlng = new google.maps.LatLng(xxx, yyy) ; // le point central = adresse ABI
var myOptions = { // les options par défaut Google...
  zoom: 8,
  center: latlng,
  mapTypeId: google.maps.MapTypeId.ROADMAP
};
// instancier l'objet google maps pour affichage dans un div "map_canvas" en partie HTML
map = new google.maps.Map(document.getElementById("map_canvas"),
  myOptions);
// instancier un objet info bulle
infowindow = new google.maps.InfoWindow(
{
  content: "ABI : Nous sommes ici ! <br />12 rue des Fleurs <br />29400  Landivisiau",
  size: new google.maps.Size(50,50),
  position: latlng
});
// instancier un 'marker'
var marker = new google.maps.Marker({
  map: map,
  position:latlng
});
// ajouter l'appel de la fonction info() au click sur le marker
google.maps.event.addListener(marker, 'click', function(){info();});
// afficher tout de suite l'info bulle
info();
} // fin initialize()

// fonction d'affichage de l'info-bulle
// cette fonction est appelée par initialize() et au click sur le marker ==> 2 var map et infowindow
glabales
function info()
{
  infowindow.open(map);
} // fin info()
</script>
```

A l'aide de cet exemple et des informations en ligne sur Google Maps, réalisez votre propre page incluant une géo-localisation simple.

Notez bien que tout cela peut être encore enrichi, suivant les besoins, et qu'une carte Google Maps est interactive pour l'utilisateur : variantes d'affichage par boutons, curseur de zoom…

Pour compléter le service offert aux utilisateurs, il est aussi simple de faire établir un calcul d'itinéraire par un service annexe de Google et de l'afficher dans un autre div HTML.

Etablissement référent
DI NEUILLY

Equipe de conception
Marc FAYOLLE – *AFPA Grenoble Pont de Claix*
Actualisation PHP : Benoit HEZARD – Afpa Nice

Remerciements :

Date de mise à jour 21/5/14
afpa © Date de dépôt légal 11/2008

afpa / Direction de l'Ingénierie13 place du Générale de Gaulle / 93108 Montreuil Cedex
association nationale pour la formation professionnelle des adultes
Ministère des Affaires sociales du Travail et de la Solidarité