Programmation Web Avancée
AJAX

Thierry Hamon

Bureau H202 - Institut Galilée
Tél. : 33 1.48.38.35.53
Bureau 150 – LIM&BIO – EA 3969
Université Paris 13 - UFR Léonard de Vinci
74, rue Marcel Cachin, F-93017 Bobigny cedex
Tél. : 33 1.48.38.73.07, Fax. : 33 1.48.38.73.55
thierry.hamon@univ-paris13.fr
http://www-limbio.smbh.univ-paris13.fr/membres/hamon/

Asynchronous Javascript And XML (AJAX)

Introduction

- Introduit en 2005 par Jesse James Garrett
- Applications Web avec interface utilisateur
- Déportation d'une partie des traitements liés à l'interface du code coté client
  - Réduction des ressources utilisées coté serveur
  - Economie de bande passante
- Exemple d'application Web AJAX :
  - Google Mail, Maps, Earth, etc.
  - Liste de suggestions automatiques
  - Traitement de texte

Asynchronous Javascript And XML (AJAX)

Introduction

- Regroupe un ensemble de technologie Web utilisées conjointement (HTML, CSS, DOM, Javascript, XMLHttpRequest, XML)
- Permet la récupération de données sur le serveur de manière asynchrone, sans interférer avec les données dans la page courante (utilisation de l'objet XMLHttpRequest)
- Utilise comme format d'échange, XML, des fichiers textes et aussi JSON

Asynchronous Javascript And XML (AJAX)

Introduction

Deux composants (Application Web Classique) :

- Serveur (implémentation JAVA ou PHP par exemple)
  - Contrôle général de l'application
  - Propose des ressources statiques : Modèle du document, bibliothèque de scripts, feuilles de style
  - Traitement dynamique des données
  - Composition dynamique de l'interface
- Client (implémentation Javascript par exemple)
  - Gestion des évènements utilisateur
  - Composition dynamique de l'interface
- Dialogue : HTTP, (X)HTML

Asynchronous Javascript And XML (AJAX)

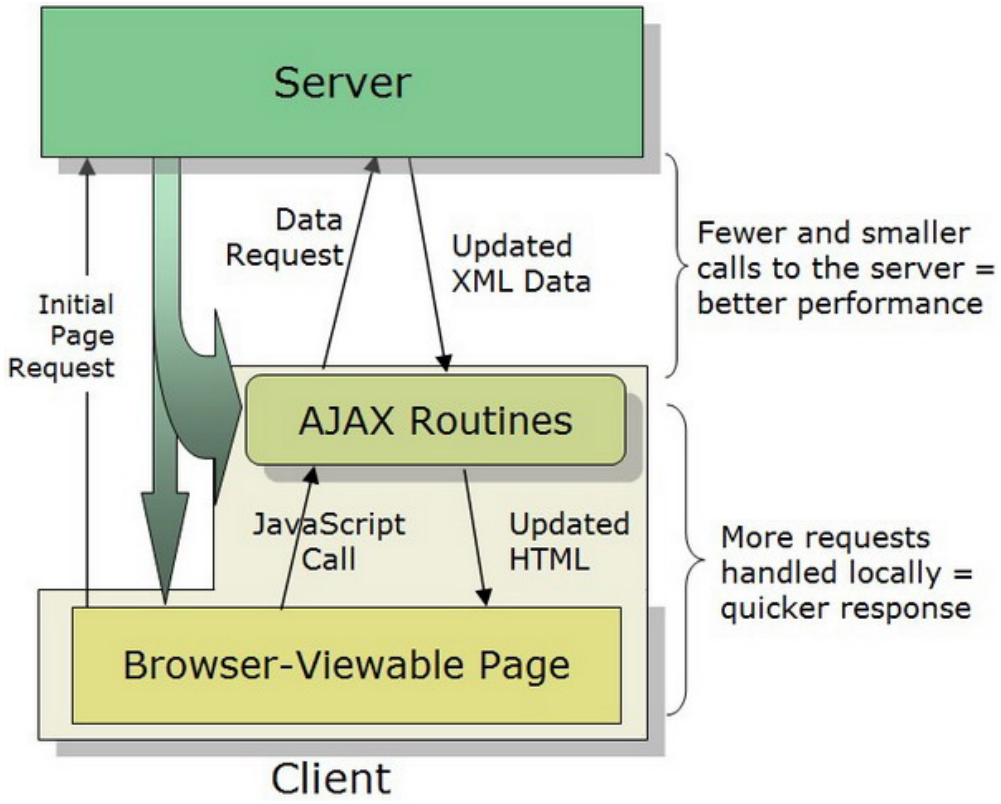
Introduction

Deux composants (Application Web AJAX) :

- Serveur (implémentation JAVA ou PHP par exemple)
  - Contrôle général de l'application
  - Propose des ressources statiques : Modèle du document, bibliothèque de scripts, feuilles de style
  - Traitement dynamique des données
- Client (implémentation Javascript par exemple)
  - Contrôle délégués en fonction du type de vue
  - Gestion des évènements utilisateur
  - Composition dynamique de l'interface
  - Traitement des données reçues
- Dialogue : HTTP, XML, JSON

Fonctionnement

Schéma



Source: `http://www.codeproject.com/KB/showcase/FarPointAJAX.aspx`

Fonctionnement

Illustration

1. Requête asynchrone au serveur dans une fonction JavaScript, déclenchée par un événement
2. Transfert asynchrone de données en XML
3. Traitement dynamique du côté client pour affichage inclusion au document HTML, transformation XSLT, etc.
4. Requête asynchrone sur un document XML en utilisant un objet XMLHttpRequest (Mozilla) ou un contrôle ActiveX XMLHTTP (IE)
5. Puis communication AJAX

Fonctionnement

Communication AJAX

Client :

- Envoi de la requête :
  - Création de l'objet requête (XMLHttpRequest)
  - Spécification des éléments de la requête (URL, méthode, headers HTTP, paramètres)
  - Association d'un gestionnaire d'événements
  - Envoi de l'objet
- Réception de la réponse :
  - A chaque modification de l'état de la requête : tester si dans l'état *ready*
  - Traitement des données reçues (Ajout à l'interface, transformation XSL)

Fonctionnement

Communication AJAX

Serveur :

- Définition des actions à réaliser lors de la réception d'une requête asynchrone AJAX

Objet XMLHttpRequest

API utilisée

- par JavaScript et d'autres langages de scripts
- pour transférer des données au format XML, texte ou JSON entre le client (navigateur) et le serveur Web
- de manière asynchrone généralement. Mais possibilité d'utilisation synchrone

Création de l'objet XMLHttpRequest : méthode ActiveXObject (IE) et objet Javascript XMLHttpRequest

Exemples de code

création de l'objet requête

```
var req = null;
function getRequest()
{
   if (window.XMLHttpRequest)
   {
      req = new XMLHttpRequest();
   }
   else if (typeof ActiveXObject != "undefined")
   {
      req=new ActiveXObject("Microsoft.XMLHTTP");
   }
   return req;
}
```

Exemples de code

création de l'objet requête

```
function basicAjaxExample() {
  var xmlHttp;
  try {
    xmlHttp=new XMLHttpRequest(); // Firefox, Opera 8.0+, Safari
  } catch (e) {
    try {
      xmlHttp=new ActiveXObject("Msxml2.XMLHTTP"); // IE 6.0+
    } catch (e) {
      try {
        xmlHttp=new ActiveXObject("Microsoft.XMLHTTP"); // IE 5.5+
      } catch (e) {
        alert("Your browser does not support AJAX!");
        return false;
      }
    }
  }
}
```

Exemples de code

Chargement asychrone - simple

```
function GetDataUsingAJAX (req, elt)
{
  // elt : contenu d'un champs
  if(req != null)
  {
    // Association de la fonction de gestion de l'état
    var url="http://www.univ-paris13.fr/monscript.php?elt=" + elt;

     // méthode sans paramètre
     req.onreadystatechange = stateChange;
     req.open("GET", url, true);
     req.send(null);
   }
}

function stateChanged(req)
{
  if (req.readyState==4) {
    document.getElementById("txt").innerHTML=req.responseText;
  }
}
```

Exemples de code

Chargement asychrone - XML

```
function GetDataUsingAJAX (req, elt)
{
  // elt : element XML du document
  if(req != null)
  {
    // Association de la fonction de gestion de l'état

    var url="http://www.univ-paris13.fr/monscript.php?elt=" + elt;

     // méthode avec paramètres
     req.onreadystatechange = function() {stateChange(elt)};

     req.open("GET", url, true);
     // pour les requetes XML
     req.setRequestHeader("Accept", "application/xml");
     req.send(null);
   }
}
```

Exemples de code

Gestion de l'état - XML

```
function stateChange (elt)
{ // elt : element XML du document
   if(req.readyState == 4) { // READY_STATE_COMPLETE
      if (req.responseXML != null) {
         var docXML= req.responseXML;
      } else {
         var docXML= req.responseText;
         docXML=parseFromString(docXML);
      }
      var docXMLresult = traiteXML(docXML);
      var str = (new XMLSerializer()).serializeToString(docXMLresult);
      document.getElementById(elt).innerHTML += str;
   }
}
```

Exemples de code

Transformation XSLT

```
//Après chargement asynchrone des documents XML et XSLT
function transform XSLT (XMLDoc, XSLDoc, id)
{
  if(XMLDoc == null || XSLDoc == null) {return;}
  try {
     // Internet Explorer
     if (window.ActiveXObject)
     {
        var target = document.getElementById(id);
        target.innerHTML = xml.transformNode(xsl);
     } else if (window.XSLTProcessor) { // Safari / Mozilla
        var fragment;
        var xsltProcessor = new XSLTProcessor();
        xsltProcessor.importStylesheet(xsl);
        fragment = xsltProcessor.transformToFragment(xml, document);
        var target = document.getElementById(id);
        target.appendChild(fragment);
     }
  } catch (e) {
     return e;
  }
}
```

Propriétés de l'objet XMLHttpRequest

- Status
  Renvoie l'état de la requête
  - 200 : OK, page trouvée
  - 404 : page non trouvée
- `onreadystatechange`
  Association d'une fonction recevant et traitant les données retournées par le serveur après une requête
  Utilisation d'un *pointeur de fonction*

Propriétés de l'objet XMLHttpRequest

- `readyState`
  Gestion de l'état de la réponse du serveur
  A chaque changement d'état, la fonction associée à `onreadystatechange` est exécutée

  Valeurs possibles :

Etat	Description
0	Requête non initialisée
1	Connexion établie
2	Requête reçue
3	Réponse en cours/Traitement de la requête en cours
4	Réponse envoyée/Terminé

Propriétés de l'objet XMLHttpRequest

- `responseXML`
  Retourne un objet DOM du XML renvoyé par le serveur
- `responseText`
  Retourne une chaîne de caractères contenant les données chargées
  A utiliser si on ne souhaite pas traiter les données en Javascript mais uniquement les afficher (par exemple, données HTML)

Méthode de l'objet XMLHttpRequest

Utilisation de 2 méthodes pour l'envoi

- `open()` – préparation de la requête
  3 arguments :
  1. Méthode utilisée pour l'envoi de la requête (GET ou POST)
  2. URL du script coté server
  3. booléen indiquant si la requête doit être envoyée de manière asynchrone ou non
- `send()` – envoie de la requête au serveur
  1 argument :
  1. données à passer au script coté serveur
     - méthode GET : `null`
     - méthode POST : variable ou chaîne de caractères

JavaScript Object Notation (JSON)

- Format alternatif à XML
- Natif en Javascript
- Permet l'échange de données entre client et serveur sans analyse (contrairement au XML).
- JSON vs. XML :
  - JSON : facilité de lecture et simplicité de mise en oeuvre
  - XML : extensible et reconnu dans tous les langages de programmation

Syntaxe

Eléments :

- Objet : contient des
  - objets sous forme d'une liste de membres

    `{ nommembre1 : valmembre1, nommembre2: valmembre2, ...`
  - tableaux sous forme d'une liste de valeurs

    `[ valeur1, valeur2, ...]`
- Variable scalaire de type `Number`, `String` ou `Boolean`
- Tableaux `[ valeur1, valeur2, ...]` (valeur : objet, tableau, etc.)
- Valeurs littérales : `null`, `false`, `true`, valeur numérique, chaîne de caractères (entre `"`)

Membre :

`"nom" : "valeur"`

Exemple de fichier JSON

```
{
  "menu": "Fichier",
  "commandes": [
      {
          "title": "Nouveau",
          "action":"CreateDoc"
      },
      {
          "title": "Ouvrir",
          "action": "OpenDoc"
      },
      {
          "title": "Fermer",
          "action": "CloseDoc"
      }
  ]
}
```

```
<?xml version="1.0" ?>
<root>
  <menu>Fichier</menu>
  <commands>
      <item>
          <title>Nouveau</value>
          <action>CreateDoc</action>
      </item>
      <item>
          <title>Ouvrir</value>
          <action>OpenDoc</action>
      </item>
      <item>
          <title>Fermer</value>
          <action>CloseDoc</action>
      </item>
  </commands>
</root>
```

Utilisation d'un fichier JSON

coté client

Récupération des données avec la méthode `eval()` et utilisation d'éléments et de la syntaxe Javascript :

```
req.open("GET", "fichier.json", true); // requête

var doc = eval('(' + req.responseText + ')'); // récupération

var nomMenu = document.getElementById('jsmenu'); // recherche
nomMenu.value = doc.menu.value;                  // assignation

doc.commands[0].title // lecture de  la valeur "title" dans le tableau
doc.commands[0].action // lecture de la valeur "action" dans le tableau
```

Fichier `fichier.json` :

```
{ "menu": "Fichier",
  "commandes": [
  { "title": "Nouveau",
    "action":"CreateDoc"
  },
  { "title": "Ouvrir",
    "action": "OpenDoc"
  },
  { "title": "Fermer",
    "action": "CloseDoc"
```

Utilisation d'un fichier JSON

coté serveur

- Utilisation de librairie propres à chaque langage (voir json.org) :
  Java : org.json.*
  Perl : JSON
  PHP : (interne en 5.2), json
  etc.

Avantages et inconvénients d'AJAX

- Avantages :
  - plus interactivité au niveau du client
  - réponse plus rapide
  - réduction des transactions client/serveur (récupération des scripts et des feuilles de style une fois pour toute)
  - séparation des méthodes pour la transmission de l'information et des formats utilisés pour représenter les informations
- Inconvénients :
  - Pas d'enregistrement dans l'historique du navigateur des pages modifiées dynamiquement
    Solution en modifiant la partie ancre (#) de l'URL
  - Difficulté à *bookmarker* l'état particulier d'une page
  - Pas d'indexation possible des pages par les moteurs de recherche
  - Si un navigateur ne supporte pas Javascript et AJAX, la page est inutilisable

Alternatives

- Flex et Flash : concurrents de AJAX
  Mais possibilité de combiner leur utilisation
  voir Goowy `http://www.goowy.com/` (Bureau virtuel)
- YAML (YAML Ain't Markup Language) : format d'échange basé sur l'utlisation de caractères spéciaux:
  `: & ! ? - --- [ ] *`, etc.
  Fichier JSON : contenu YAML valide (et non l'inverse), sauf les commentaires
  Format YAML moins lisible que JSON (?)

Pour aller plus loin

- Description d'AJAX :
  `https://developer.mozilla.org/fr/AJAX`
- Exemples et tutoriels du W3C:
  `http://www.w3schools.com/Ajax/ajax_examples.asp`
- Frameworks : voir `http://www.ajaxprojects.com/`
  - openAjax (IBM) : Dojo
  - Ruby / Ruby on Rails (RoR)
  - Plugins Eclipse : Rich Ajax Platform, Direct Web Remoting
  - PHP : `http://ajaxpatterns.org/PHP_Ajax_Frameworks`
- Bibliothèques :
  - SAJAX
  - Google Web Toolkit (AJAXSLT...)
- Article de Jesse James Garrett introduisant AJAX:
  `http://www.adaptivepath.com/ideas/essays/archives/000385.php`