

AJAX

Concepts et Technologies XML

Master Pro ILI

Année 2007-08

AJAX : Asynchronous Java And Xml

- ▶ client riche : une meilleure répartition de la charge de travail entre le serveur, le réseau et le client
- ▶ recharger uniquement les parties nécessaires d'une page en fonction des événements

- ▶ ensemble de techniques qui utilisent
 - ▶ HTML, CSS
 - ▶ DOM
 - ▶ HTTP
 - ▶ JavaScript
 - ▶ l'objet XMLHttpRequest
 - ▶ et éventuellement un autre langage sur le serveur

L'objet XMLHttpRequest

Pour effectuer une requête HTTP

```
interface XMLHttpRequest {  
  // pour établir la connexion  
  void open(méthode DOMString, uri DOMString,  
            boolean async);  
  
  void setRequestHeader(en-tête DOMString,  
                        valeur DOMString) raises(DOMException);  
  // pour envoyer la requête (params. uniqu. pour POST)  
  void send(donnée DOMString);  
  void send(donnée Document);  
  void abort();  
};
```

Pour traiter la réponse du serveur

```
interface XMLHttpRequest {  
  DOMString getAllResponseHeaders() ;  
  DOMString getResponseHeader(en-tête DOMString) ;  
  attribut DOMString responseText ;  
  attribut Document responseXML ;  
  // 200 : OK, 404 : File Not Found, ...  
  attribut unsigned short status ;  
  attribut DOMString      statusText ;  
};
```

L'objet XMLHttpRequest

Les différents états de l'objet requête :

```
interface XMLHttpRequest {  
  // change de valeur, de 0 à 4  
  readonly attribute unsigned short readyState;  
  // on lui associe une fonction : activée à chaque  
  // changement de valeur pour readyState  
  attribut EventListener onreadystatechange;  
  
  // les états possibles  
  const unsigned short UNSENT = 0;  
  const unsigned short OPEN = 1;  
  const unsigned short SENT = 2;  
  const unsigned short LOADING = 3;  
  const unsigned short DONE = 4;  
};
```

Lorsqu'un événement sur la page justifie d'en rafraîchir une partie, on appelle une fonction qui :

- ▶ créé un objet XMLHttpRequest
- ▶ associe à sa propriété `onreadystatechange` une fonction qui traitera le retour de la requête
- ▶ créé une connexion HTTP
- ▶ envoie la requête

Un premier exemple

```
<script language="JavaScript">
function submitForm(){
    var req = null;
    try {
        req = new XMLHttpRequest();
    }
    catch (e) { // spécial I.E.
        req = new ActiveXObject(Microsoft.XMLHTTP);
    }
    req.onreadystatechange = function() {
        // beaucoup de travail
    };
    req.open("GET", "data.xml", true);
    req.send(null);
}
```

Schéma général :

- ▶ si le statut est différent de 4 (**DONE**) :
 - ▶ ne rien faire ! ou informer l'internaute
- ▶ sinon, il faut étudier le retour de la requête
 - ▶ si son statut est 200 :
récupérer le résultat (**responseText** ou **responseXml**)
 - ▶ sinon, traiter l'échec.

Exemple de fonction de retour

```
req.onreadystatechange = function(){
    document.getElementById("info").value="Wait server...";
    if(req.readyState == 4) {
        if(req.status == 200) {
            document.getElementById("retour").value=
                req.responseText ;
        }
        else {
            document.getElementById("retour").value=
                "Error : returned status code " + req.status
                + " " + req.statusText ;
        }
    }
};
```

```
<body>
  <form name="ajax" method="POST" action="">
    <input type="button"
      value="Submit"
      onClick="submitForm()" />
    <input type="text"
      id="retour"
      size="32" value="" />
  </form>
</body>
```

Et les informations demandées ?

Le fichier `data.xml` peut simplement contenir :

Un message de bienvenue !

mais il peut aussi contenir un document XML :

```
<h3>Un message de bienvenue !</h3>
```

Dans ce cas on récupèrera le résultat de la requête avec :

```
attribut Document responseXML ;
```

Avec les méthodes de DOM! :

```
// on récupère le document XML
result = req.responseXML;
// on accède à sa racine
racine = result.documentElement;
// on fait une copie du noeud
// dans le document courant
nd = document.importNode(racine,true);
// on l'insère dans le document courant
document.body.appendChild(nd);
```

Mais `importNode()` n'est pas supporté par tous les navigateurs...
Une autre proposition :

```
// on récupère le document XML
// sous forme texte
result = req.responseText ;
// si on veut ajouter à la fin...
// on crée un élément
boite = document.createElement("div") ;
// on insère la réponse dans cette boite
boite.innerHTML = result ;
// et on ajoute la boite dans le document courant
document.body.appendChild(boite) ;
```