Département d'Informatique

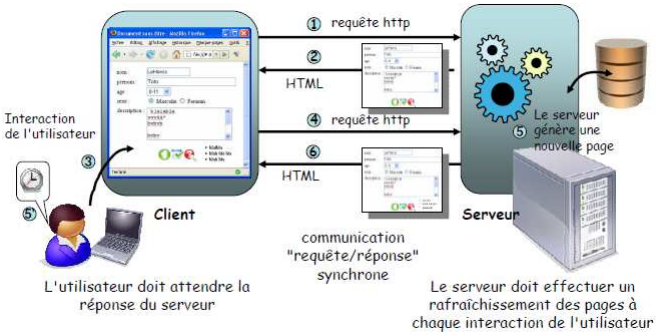
Introduction à AJAX

- Frédéric MOAL
- Année 2011/2012

1

Web : modèle classique 1.0

- Navigateur chargé du rendu de l'affichage seulement, pas d'intelligence
- Navigation sous forme d'enchainement « statique » déterminé par le serveur



2

Web : modèle classique 1.0

- Limitations importante d'ergonomie
- Widgets limités au HTML
- Pas de retour immédiat sur les actions utilisateur
- Requête serveur bloquante pour l'utilisateur
- Perte de contexte (position,…) à chaque requête/réponse

3

Comment remédier au Web 1.0

- Animation des écrans du côté client
  - Code exécuté sur le navigateur
    - Limite les échanges client/serveur
    - Meilleure interactivité et ergonomie
  - Optimisation des échanges client/serveur
    - Communications asynchrone : pas d'attente
    - Échange de données plutôt que de la présentation
- Technologies RIA (Rich Internet Applications)
- Web 2.0 vs Web 1.0
  - Eg Google maps, apps, …

4

Rich Internet Application

- De nombreuses technologies (pas toutes récentes)
  - Applets (1995) : Code java
  - DHTML (Dynamic HTML)
    - Javascript (1995) + DOM + CSS
  - Flash/FLEX (1996, 2004) Macromedia-Adobe
    - MXML + ActionScript
  - Silverligth (2006) Microsoft
    - XAML + ECMACScript
  - …
  - AJAX : Asynchronous Javascript And XML
    - AJAX = Javascript + XmlhttpRequest
      Code client écrit en Javascript
      Communications avec le serveur réalisées à l'aide de l'objet Javascript **XmlhttpRequest**

5

Technologies AJAX

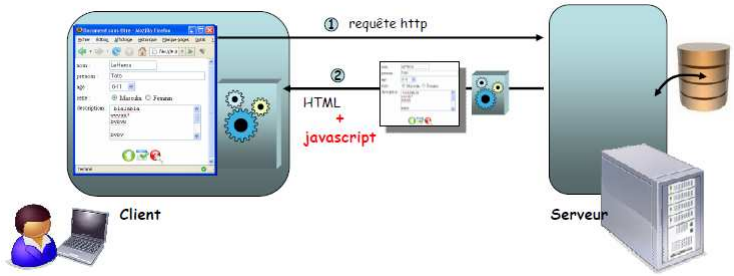
- **AJAX = DHTML (DOM + CSS + Javascript) + XmlhttpRequest**
  - Javascript
    - **Langage de script orienté objet et faiblement typé**
    - **Fonctions javascript invoquées lorsque intervient un événement sur la page**
    - **"Glue" pour tout le fonctionnement d'AJAX**
  - DOM (Document Object Model)
    - **API pour accéder à des documents structurés**
    - **Représente la structure de documents XML et HTML**
  - CSS (Cascading Style Sheets)
    - **Permet un séparation claire du contenu et de la forme de la présentation**
    - **Peut être modifié par le code Javascript**
  - XmlhttpRequest
    - **Objet Javascript qui assure une interaction asynchrone avec le serveur**

6

Appli Web AJAX : principe

- Une partie de l'intelligence fonctionnelle de l'application est déportée vers le navigateur

  1er échange similaire au web "classique" : le serveur envoie une page au client mais en y embarquant de l'intelligence (code javascript)



7

Appli Web AJAX : principe

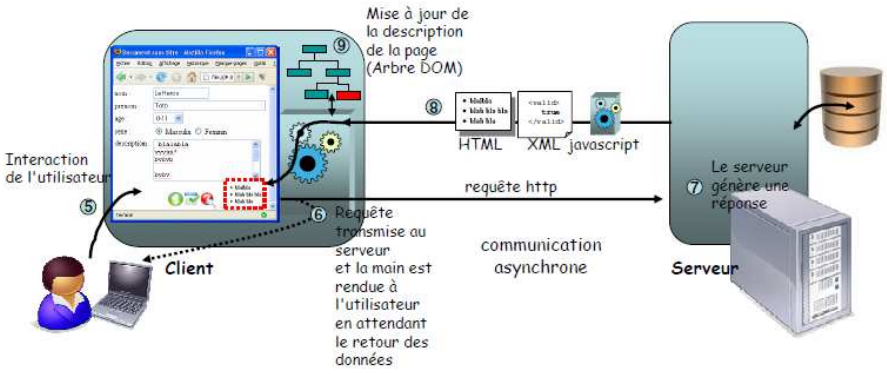
- Une partie de l'intelligence fonctionnelle de l'application est déportée vers le navigateur

8

Appli Web AJAX : principe

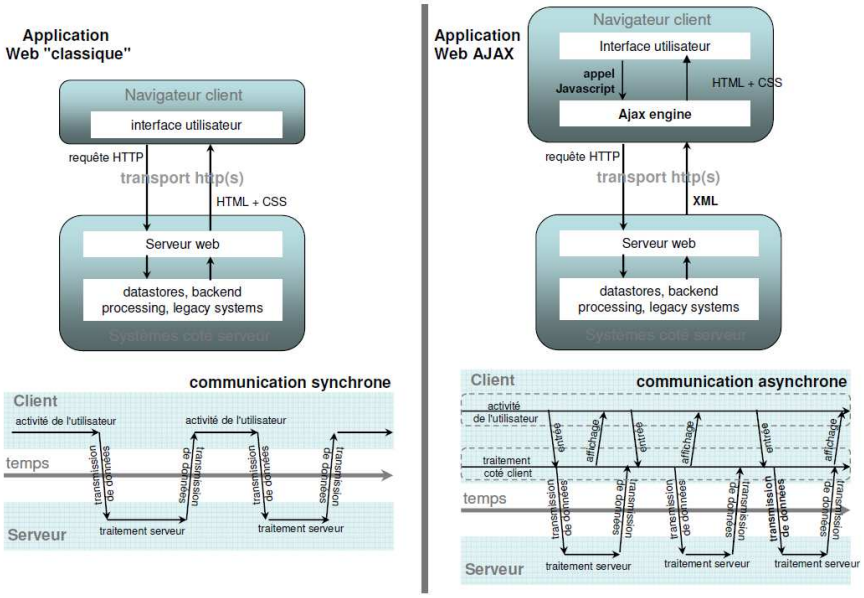
- Une partie de l'intelligence fonctionnelle de l'application est déportée vers le navigateur

D'autres requêtes nécessitent l'interrogation du serveur



9

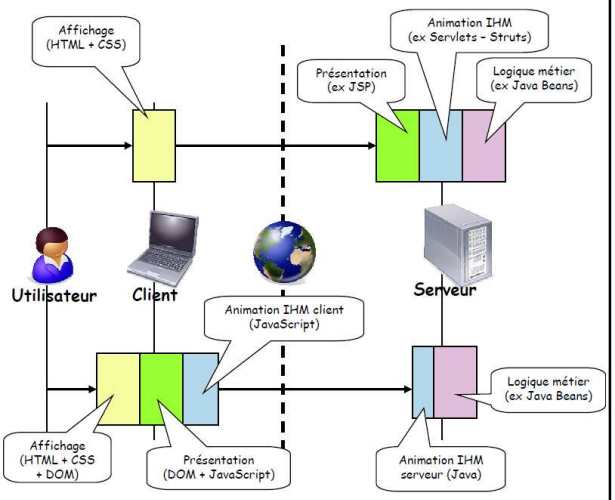
Web 1.0/2.0 : deux modèles



10

Différents modèles

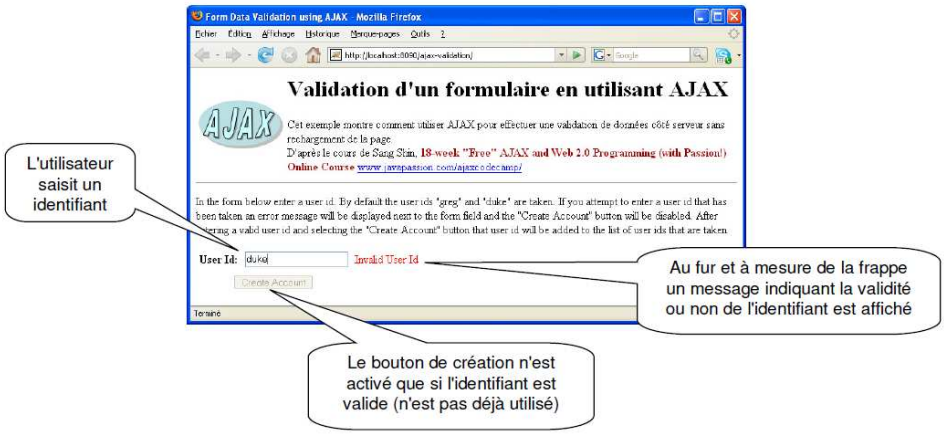
- Appli WEB 1.0
  - Peu de javascript
  - Charge serveur très importante
  - Ergonomie faible
- AJAX léger
  - Cohabitation entre technologie "classique" (ASP, PHP, JSP) et AJAX
  - Charge serveur importante
  - Ergonomie améliorée
- Full AJAX
  - Beaucoup de javascript
  - Charge serveur - importante
  - Ergonomie ++



11

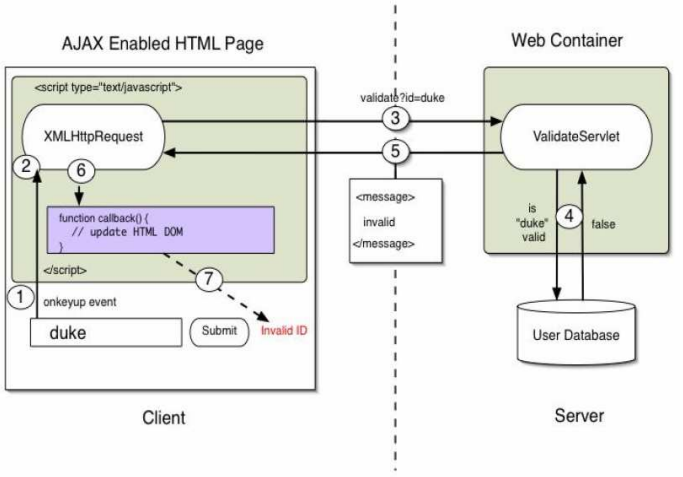
Exemple AJAX

- Exemple d'après : *AJAX Basics and Development Tools de Sang Shin* (sang.shin@sun.com, Sun Microsystems) www.javapassion.com/ajaxcodecamp



12

Fonctionnement



① Événement sur le client → appel d'une fonction javascript

② Création et configuration d'un objet XMLHttpRequest

③ L' objet XMLHttpRequest fait une requête asynchrone

④ Le servlet valide l'identifiant soumis

⑤ Le servlet retourne un document XML contenant le résultat de la validation

⑥ L'objet XMLHttpRequest appelle la fonction callback() et traite ce résultat

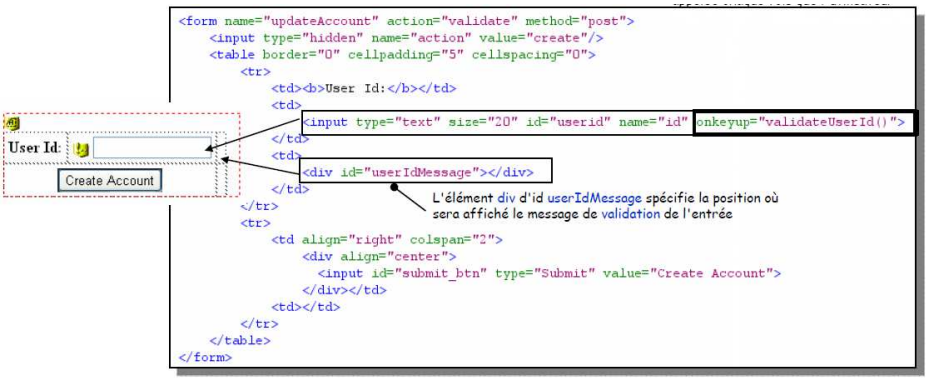
⑦ Mise à jour de la page HTML (DOM)

13

Fonctionnement : formulaire

- **Gestion des événements dans le formulaire HTML**

La fonction Javascript validateUserId
est associée au champ de saisie de texte "userid" pour la gestion des événements de type onkeyup :
validateUserId est appelée chaque fois que l'utilisateur tape une lettre dans le champ de saisie.



14

Fonctionnement : Javascript

Coté client :
la fonction JavaScript invoqué à chaque événement "keyup"sur le champ de saisie

```
var req;

function validateUserId() {

    if (window.XMLHttpRequest) {
          req = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
          isIE = true;
          req = new ActiveXObject("Microsoft.XMLHTTP");
    }

    req.onreadystatechange = processRequest;

    if (!target)
        target = document.getElementById("userid");
    var url = "validate?id=" + escape(target.value);

    req.open("GET", url, true);

    req.send(null);
}
```

2 **Création et configuration d'un objet `XMLHttpRequest`**

Selon le navigateur l'objet XMLHttpRequest est crée différemment

fonction callback : fonction Javascript (voir plus loin) qui sera invoquée lorsque le serveur aura fini de traiter la requête :
En Javascript les fonctions sont des objets et peuvent être manipulées en tant que tels

Récupération de la valeur userid tapée par l'utilisateur (via API DOM) et construction de l'url du composant serveur qui sera invoqué

L'appel sera asynchrone

3 **L' objet `XMLHttpRequest` effectue une requête asynchrone**

15

Fonctionnement : serveur

Coté Serveur :
la servlet traitant la requête GET émise par la fonction JavaScript `validateUuserId`

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse  response)
        throws IOException, ServletException {

    String targetId = request.getParameter("id");

    if ((targetId != null) &&
          !LoginManager.validateUserId(targetId.trim())) {

        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-cache");
        response.getWriter().write("<valid>true</valid>");
    } else {

        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-cache");
        response.getWriter().write("<valid>false</valid>");
    }
}
```

4 **Le servlet valide l'identifiant soumis**

S'agit-il d'un identifiant déjà utilisé ?

5 **Le servlet retourne un document XML contenant le résultat de la validation**

```
<valid>
   true
</valid>
```

```
<valid>
   false
</valid>
```

16

Fonctionnement : retour JavaScript

6 L'objet `XMLHttpRequest` appelle la fonction callback() et traite ce résultat

```
function processRequest() {
   if (req.readyState == 4) {
      if (req.status == 200) {

         var message = req.responseXML.
               getElementsByTagName("valid")[0].
               childNodes[0].nodeValue;

         setMessageUsingDOM(message);

          var submitBtn = document.getElementById("submit_btn");
          if (message == "false") {
               submitBtn.disabled = true;
          } else {
               submitBtn.disabled = false;
          }
      }
   }
}
```

Cette fonction est invoquée chaque fois que le champ readyState de l'objet XMLHttpRequest est modifié

readyState == 4 et status = 200 indiquent que la réponse a été correctement reçue par le client

Extraction de la valeur true ou false des données retournées par le serveur

7 Mise à jour de la page HTML (DOM)

Affiche dans la zone prévue à cet effet la validité ou non de l'identificateur fourni

Active ou désactive le bouton de soumission du formulaire selon validité de l'identificateur fourni

17

Fonctionnement : retour JavaScript

7 Mise à jour de la page HTML (DOM)

```
function setMessageUsingDOM(message) {

   var userMessageElement = document.getElementById("userIdMessage");

   var messageText;
   if (message == "false") {
      userMessageElement.style.color = "red";
      messageText = "Invalid User Id";
   } else {
      userMessageElement.style.color = "green";
      messageText = "Valid User Id";
   }

   var messageBody = document.createTextNode(messageText);

   if (userMessageElement.childNodes[0]) {
       userMessageElement.replaceChild(
             messageBody, userMessageElement.childNodes[0]);
   } else {
        userMessageElement.appendChild(messageBody);
   }
}
```

Récupération de l'objet DOM correspondant à la zone du message grâce à l'id inséré dans le code HTML

Préparation du message

Création du message
Si il existe déjà un message le remplace par le nouveau, sinon le rajoute

18

Comment faire de l'AJAX ?

- Une multitude de Framework pour aider à l'utilisation de AJAX

Source : http://www.ajaxprojects.com/

.NET Frameworks (35)
Ajax Web Widgets, zumiPage, Ajaxium, ASP.NET Atlas, CAPXOUS.AutoComplete,...
Java Frameworks (71)
Direct Web Remoting, BQUoyURkscOO, SmartClient AJAX GUI System, Jaxcent, ...
PHP Frameworks (31)
aSSL Secure Connection, DutchPIPE, AJASON, Cajax, ProtoJax, ...
Ruby Frameworks (8)
Try Ruby, Pikipimp.com, Ruby-GNOME2, Ruby on Rails, jQuery and Flex, Radiant, ...
Other Frameworks (60)
Rico, Porcupine, mxGraph, XAP, Bold text Rialto, 4D Ajax Framework, PHPRPC, ...
Ajax Tools (23)
AJAX file upload, Prototype Window Class, mootools, Nitobi Grid V3, ...
JavaScript frameworks (32)
Bindows Benchmarking, JSONER, Script.aculo.us, Ample SDK, DHTMLX Toolkit, ...

19

Comment faire de l'AJAX ?

- L'un des gros problèmes d'AJAX est le développement Javascript !
  - Test et débogage limités
  - Portabilité difficile, il faut différentes versions des fonctions selon les navigateurs

⇨ Il faut limiter au maximum l'écriture de code javascript !

- 1ère solution : utilisation des bibliothèques javascript
  - Prototype, Script.aculo.us, DOJO, Yahoo UI
- 2ème solution : Utilisation des bibliothèque de tags, composants Struts ou JSF "ajaxifiés"
  - Les composants génèrent du javascript (AjaxTags, AjaxAnywhereRichFace, ajax4JSF, IceFaces…)
- 3ème solution : traduction d'un langage en Javascript
  - GWT (Google Web Toolkit) java javascript

20