Les applications web

Vers l'Internet de demain

Nathanaël Cottin

contact@ncottin.net
http://www.ncottin.net

version 0.0.4 – 2011

Partie 1 – Présentation des applications web

Afficher

Partie 2 – Interfaces graphiques du web

Afficher

Partie 3 – Principales technologies

Afficher

Partie 4 – Intégration de services web

▸ Afficher

Partie 5 – L'avenir des applications web

▸ Afficher

Partie I

Présentation des applications web

Principes fondateurs
Évolution des interfaces

À l'origine

À ses débuts :

- ARPANET (1969)
- Besoin de diffuser l'information (scientifique)
- Statique, immuable
- Utilisation d'un protocole d'échange normalisé : `HTTP`

De nos jours. . .

Maintenant :

- Internet = outil quotidien
- Contenus riches, dynamiques
- Vitrine d'entreprise
- Moteur de l'économie
- Publication d'informations personnelles ⇒ Image de soi

Versionning du web

Web 1.0, 2.0 ? 3.0 ?. . .

Introduction

Terminologies galvaudées, utilisées par les entreprises pour donner un sentiment de «nouvelles technologies». . .
Qu'en est-il exactement ?
Quand porter l'appellation web 2.0 ?

Bénéfices

- Mutualisation des ressources
- Application cliente unique (navigateur)
- Aucune (ou peu de) maintenance côté client
- Évolutivité des applications
- Interfaces proches des applications «traditionnelles»
- Respect du MVC
- Nombreux frameworks

Limites

- Hétérogénéité du support des standards W3C (M$)
- Difficulté de reproduire certains comportements (et composants graphiques)
- Accès au système de fichiers
- Conception = patchwork (client)
- JavaScript = langage non typé, programmation peu conventionnelle (tableaux associatifs)
- Modèle "pull" (`HTTP`)
- Requêtes au serveur ⇒ lenteurs

Programmation JavaScript...

Tout est tableau associatif (`Object`)
JavaScript est donc non typé...

```
var myVar = new Object();

myVar.myAttribute = 0;
myVar.myAttribute = "string";

myVar = {att1: 0, att2: "string"};
alert(myVar.att1);

myVar["integer"] = 0;
myVar["string"] = "value";
alert(myVar["string"]);
```


Serveur web : définition

Serveur web

«Simple» logiciel permettant d'interpréter des requêtes `HTTP`, généralement sur le port 80 et de fournir une réponse avec ce même protocole.

– Apache Foundation

Exemple : Apache, IIS

Serveur web : fonctionnalités

- Service de pages statiques (originel)
- Pages dynamiques : langages interprétés (CGI, PHP, python, ASP)
- Mélange présentation / logique métier

⇒ Sites web, applications de faible complexité

Serveur d'applications : définition

Serveur d'applications

Serveur hébergeant les applications destinées à être utilisées dans un réseau distribué. En comparaison au serveur de fichier qui abrite les données destinées à être téléchargées et traitées par le poste client, le serveur d'applications assume une partie du traitement.

– Le Journal du Net

Exemples : Websphere, WebLogic, JBoss, JoNaS, Tomcat, Glassfish
Les plus utilisés : Eclipse/Tomcat et NetBeans/Glassfish

Serveur d'applications : fonctionnalités

- Pages dynamiques : L4G (Java)
- Paradigme multi-tiers
- MVC

⇒ Applications complexes

Choix du type de serveur

Critère	Serveur web	S. d'applications
Site web dynamique		
Application simple		
Application complexe		
MVC		
L4G		
Frameworks		
Multi-tiers		
Support de charge		

Moteurs technologiques de l'évolution

- Nouveaux formats de contenus
- Évolution des standards
- Nouveaux modèles, architectures et paradigmes
- Évolution du matériel

Applications web 1.0

- Nombreuses pages web
- Chargement intégral (navigation)
- Dynamicité limitée
- Échanges synchrones
- Interfaces propres au web (hyperliens)
- HTML souvent mal rédigé (moteurs d'affichage complexes)

Non technique : utilisé par les professionnels

Applications web 2.0

- Unique page web (“Single Page Interface” - SPI)
- Chargement partiel (absence de navigation)
- Boutons «Reculer» et «Avancer» du navigateur non utilisés
- Contenu hautement dynamique (JavaScript)
- Échanges asynchrones (+ synchrones)
- Interfaces proches des interfaces «traditionnelles»
- HTML → XHTML, DOM (“Document Object Model”) XML
- Syndication de contenu (RSS)

Non technique : associé aux réseaux sociaux ⇒ partage par les internautes

Partie II

Interfaces graphiques du web

Interfaces web 1.0 et web 2.0

Interfaces du web 1.0 : composants graphiques

Éléments de formulaires :

- Éléments `input` (dont le type `hidden`)
- Éléments `select`
- Éléments `textarea`

Barre de progression remplacée par GIF animée
⇒ Transposé aux applications traditionnelles

Interfaces du web 1.0 : navigation

- Externe (entre pages)
- Ancres (balise `a`)
- Boutons (`submit` et `button`)
- Autres éléments supportant les actions souris

Interfaces du web 1.0 : mise en page

- Conteneurs : `frameset`, `frame` et `iframe`
- Panneaux : `layer` (Netscape) et `div` (M$)
- Éléments `table`, `p`, `br`, `hr`, ...
- Boîtes de dialogue modales du navigateur (alert, confirm, prompt)
- Fenêtres JavaScript avec l'attribut `target` : pollution

⇒ Gérée par le HTML : tableaux imbriqués, ...

- Astuces diverses : `p` vide, image transparente de `1px`, ...
- CSS uniquement pour apparence des éléments
- Présentation figée, non évolutive

Interfaces du web 2.0 : composants graphiques

- Nouveaux composants : onglets, menus déroulants, infos bulles, accordéons, arborescences, calendriers, sélecteurs de couleur, ...
- Amélioration des composants existants : boutons

Nombreuses bibliothèques de composants et d'effets :

- jQuery, plugins et dérivés
- MooTools, plugins et dérivés : MochaUI, jxLib, ...
- raphael, prototype, ...

Interfaces du web 2.0 : navigation

- Interne à la page web (index)
- Ancres supplantées par autres composants
- Facilité de gestion des événements : souris, clavier, focus, blur, ...
- Par JavaScript (en majorité)

Interfaces du web 2.0 : mise en page

- Éléments `frameset` et `frame` proscrits
- Conteneur `div` supplante `layer`
- Éléments `table` et `iframe` déconseillés
- Boîtes de dialogue du navigateur dépréciées
- Fenêtres JavaScript (`window.open()`) bannies

⇒ Gérée par CSS (souvent externe)

- Thèmes graphiques
- CSS pour apparence des éléments et positionnement
- Présentation plus apte à évoluer : séparation contenu / mise en page

Partie III

Principales technologies

Technologies anciennes et actuelles
Côté client et côté serveur
Évolution des technologies

HTML

Contient toutes les données directes et indirectes (URL)

Organisation :

- En-tête (`head`) : scripts, feuilles de style
- Corps (`body`) : contenu à afficher
- Pied de page : scripts (peu employé)

Mise en page :

- `div`
- `p`
- `span`

CSS

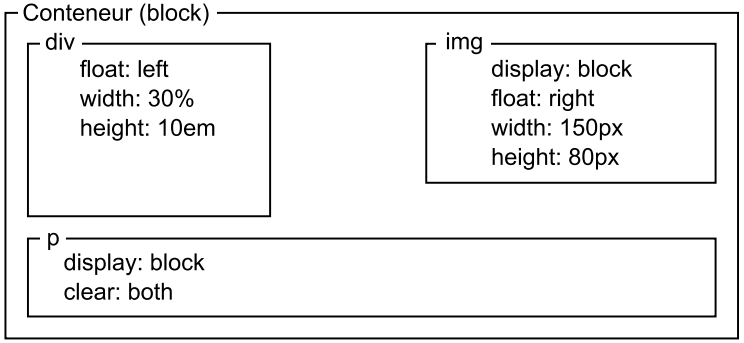
Rôle :

- Disposition des conteneurs dans la page :
  - Dans le flux : `position: relative` (défaut)
  - Hors du flux : `position: absolute`
- Attributs des éléments graphiques et de mise en page

⇒ Rendu diffère selon le moteur d'affichage (Gecko, Webkit, IE, . . .)
⇒ Recours à des astuces CSS pour gérer la compatibilité

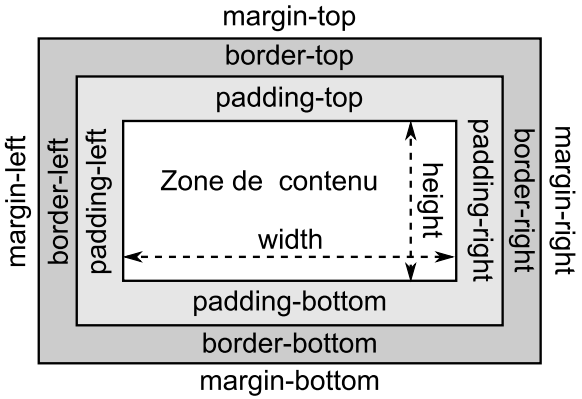
Disposition dans la page

- `display` : `block`, `inline`
- `float` : `left`, `right`, `inherit`
- `clear` : `left`, `right`, `both`



Apparence des éléments graphiques

- Espacement externe : `margin`
- Espacement interne : `padding`
- Alignement : `text-align` (`left`, `center`, `right`)



HTML, XHTML et DOM

- XHTML = instance XML (XMLSchema)
- DOCTYPE `transitional` (HTML) ou `strict` (XHTML)
- Accès aux éléments du DOM d'une page web via JavaScript
- Accès aux propriétés (attributs, valeurs) des éléments
- Fenêtres de dialogue = `div` + événements asynchrones

JavaScript

- Nécessaire aux interactions avec le serveur
- Gère la dynamicité de la page web

Intervient :

- Lors du chargement (`load`) : création d'éléments du DOM
- Lorsque la page est prête (`ready`)
- Lors de l'exécution automatique de fonctions (manipulation DOM)
- À la demande de l'utilisateur

Manipulation DOM traditionnelle JavaScript

Accès au DOM :

- `document.getElementById("...")`
- `document.getElementsByTagName("...")`

Modification partielle asynchrone de la page web :

- `XmlHttpRequest`
- `innerHTML`

Gestion uniforme des événements :

- `window.onDomReady("f")`
- `onclick=g`, `ondblclick=h`

Exemple : manipulation DOM avec Javascript

```
function setContent() {
  // Access DOM element
  var elt = document.getElementById("modify");

  // Modify element content
  // Dynamically append a paragraph child
  elt.innerHTML = "<p>Content</p>";
}

window.onDomReady(setContent);
```


Exemple : manipulation DOM avec MooTools

```
// DOM element creation
var elt = new Element("p");
elt.set("text", "Some text");

// DOM element access and modification
var dest = $("myFieldId");
dest.empty();
dest.appendChild(elt);
```


Exemple : événements MooTools

```
window.addEvent("load", function() {
  var container = $("container");
  new Element("p", {
    id: "myElement"
  }).addTo(container);
});

window.addEvent("domready", function() {
  $("myElement").set("text", "value");
});
```


Soumission de formulaire

Envoi en `HTTP GET` ou en `HTTP POST` (éventuellement `multipart`)

Cases à cocher

Les éléments `checkbox` ne sont transmis que si cochés (valeur “`on`”)...

⇒ Seuls les éléments du formulaire sont transmis au serveur

Description, principales opérations

Description :

- Objet JavaScript ou ActiveX (IE)
- API normalisée par le W3C

Principales opérations :

- `open(method, url, async[, uname, pwd])` : préparation de la requête
- `send(str)` : soumission de la requête. La chaîne `str` contient les données `POST` (`null` sinon)

Opérations secondaires

- `abort()` : annule la requête en cours
- `getAllResponseHeaders()` : récupère les en-tête `HTTP` de la réponse
- `setRequestHeader(name, value)` : ajoute un champ d'en-tête à la requête (2 chaînes de caractères)
- `getRequestHeader(name)` : récupère une valeur particulière d'en-tête de la requête
- `getResponseHeader(name)` : récupère une valeur particulière d'en-tête de la réponse

Propriétés

- `readyState` : statut courant de la soumission :
  - 0 : requête non initialisée
  - 1 : connexion avec le serveur établie
  - 2 : requête reçue par le serveur
  - 3 : requête en cours de traitement
  - 4 : réponse prête
- `onreadystatechange` : exécute une fonction à chaque changement de statut de la soumission
- `status` : code `HTTP` de la réponse (e.g. 404)
- `statusText` : libellé du code `HTTP` (e.g. "Not found")
- `responseText` : contenu textuel de la réponse
- `responseXML` : réponse au format XML

Instanciation de `XmlHttpRequest`

```
function createXmlHttpRequest() {
  if (window.XMLHttpRequest)
    return new XMLHttpRequest();
  if (window.ActiveXObject) {
    var names = [
      "Msxml2.XMLHTTP.6.0", "Msxml2.XMLHTTP.3.0",
      "Msxml2.XMLHTTP", "Microsoft.XMLHTTP"];
    for(var i in names) {
      try {
        return new ActiveXObject(names[i]);
      } catch(e){}
    }
  }
  return null;
}
```

Exemple générique d'utilisation

```
var xhr = createXmlHttpRequest();

xhr.open("GET", "content.xml", true);
xhr.onreadystatechange = function() {
  if (xhr.readyState == 4) {
    var content = xhr.responseXML;
    // or xhr.responseText
    ...
  }
}

xhr.send(null);
```


Versions actuelles de XmlHttpRequest

Niveau 1 : seul domaine d'origine autorisé
Niveau 2 : ajout de communications inter-domaines

Support :

- Niveau 1 : tous navigateurs
- Niveau 2 : Firefox 3.5, Google Chrome, Opera et Safari 4.
  IE 8 implante son propre XDomainRequest...

Qu'est-ce que l'AJAX

- Bibliothèques proposant une surcouche à XmlHttpRequest et aux événements Javascript des pages web
- Prise en charge de la (non) compatibilité entre navigateurs
- Uniformisation de l'accès aux éléments du DOM : "$" et "$$"
- Interfaces uniformisées de capture et traitement des événements
- Codecs XML et JSON

Critique

Pour : asynchrone ⇒ pas de blocage de l'interface utilisateur

Contre : asynchrone ⇒ pas de blocage de l'interface utilisateur...

Exemple de requête POST HTTP avec MooTools

```
var request = new Request({
  method: 'post',
  url: 'servlet', data: "param1=val1&param2=val2",
  onSuccess: function(response) {
    // Do something with response data
    $("myFieldId").set("html", response);
  }
  onFailure: function(xhr) {
    // Do something when request fails
  }
});

request.send();
```


Téléchargement d'une image avec MooTools

```
var myRequest = new Request({
  url: "image.jpg",
  onProgress: function(event, xhr) {
    var ratio = event.loaded / event.total * 100;
    $("dowloadStatus").set("text", ratio + "");
  }
  ...
});

myRequest.send();
```


Exemple de requête JSON avec MooTools

```
var request = new Request.JSON({
  method: 'post',
  url: 'servlet', data: "param1=val1&param2=val2",
  onSuccess: function(response) {
    // Do something with response fields
    $("myFieldId").set("html", response.myValue);
  }
  onFailure: function(xhr) {
    // Do something when request fails
  }
});

request.send();
```

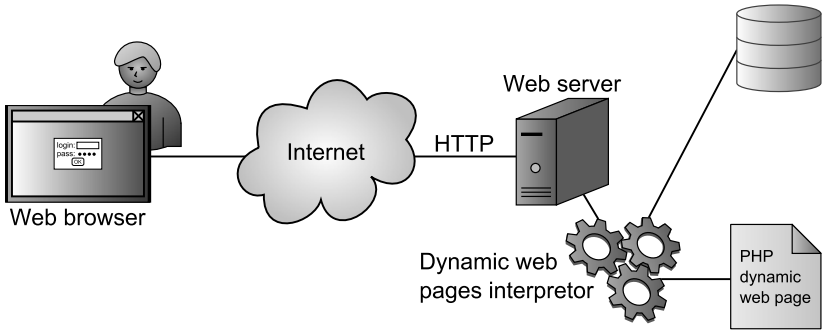

Éléments fondateurs

Modèle CGI :

- Perl
- PHP
- Python
- ...

Serveur web = serveur de pages HTML

Modèle architectural



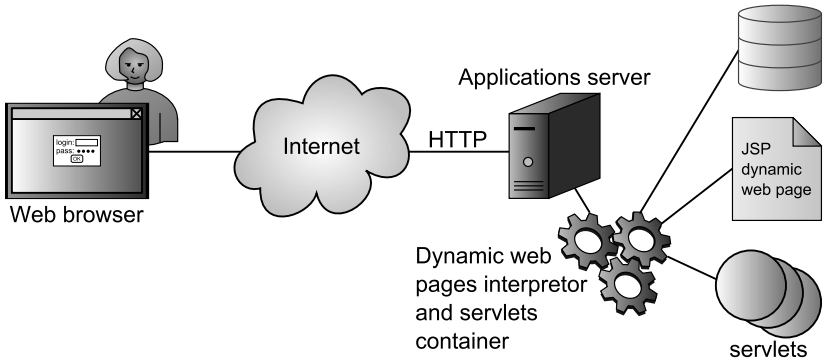
Éléments fondateurs

Modèle J2EE :

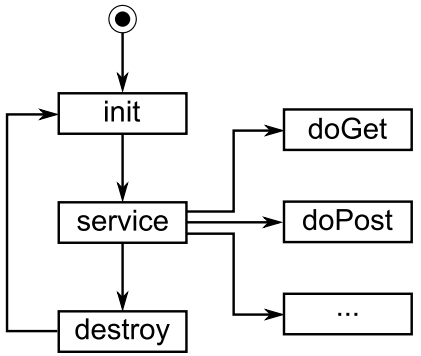
- Servlet
- JSP
- JavaBeans, EJB
- Frameworks : Struts, JSF, Hibernate, . . .
- GWT
- . . .

Serveur d'applications = conteneur de servlets

Modèle architectural



Cycle de vie d'une servlet



Canevas d'une servlet

```
public final class MyServlet extends HttpServlet {
  @Override
  protected void doGet(HttpServletRequest request,
      HttpServletResponse response)
  throws ServletException, IOException {
    ...
  }

  // Same signature for "doPost" operation

  @Override
  public String getServletInfo() {
    return "Servlet description";
  }
}
```


Partie IV

Intégration de services web

Les SOA
SOAP, WSDL, UDDI
Applications et services web
Évolution des technologies

Éléments fondateurs

- SOA
- SOAP (JSON, REST)
- WSDL
- UDDI

Partie V

L'avenir des applications web

Quelles applications
Évolution des technologies

Conception des applications web

- Contenus dynamiques, graphiques
- Différents types de support
- Outils évolués (frameworks)
- Sécurisation des transferts
- Conformité aux standards :
  - HTML : M$ contre le reste du monde (`marquee`)
  - CSS (1, 2, 2.1, 3) : support différent selon les moteurs
  - Accessibilité : WAI ("Web Accessibility Initiative")

Aujourd'hui

- Web très présent (individus et entreprises)
- Nouveaux paradigmes (réseaux sociaux, ...)
- "Desktop" $\rightarrow$ "Webtop"

$\Rightarrow$ Dimension économique et sociologique
$\Rightarrow$ Web participatif (humain)

Demain : web 3.0 ?

- Web omniprésent (navigateur web = organe central)
- Mutualisation des applications et des données
- Homogénéisation des technologies
- Collaboration dynamique des applications web (interopérabilité)
- Intégration du modèle P2P : «chacun stocke les données de tous»
- Mise à jour des standards existants (HTML 5)
- Apparition de nouveaux standards

$\Rightarrow$ Phénomène sociologique plus marqué
$\Rightarrow$ Web syntaxique $\rightarrow$ Web sémantique (humain + machines)

Pour en savoir plus

- Wikipedia : «Web 2.0»
- "What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software", Tim O'Reilly, 2005
- Extension «Greasemonkey» pour Firefox : `www.greasespot.net`