Programmation Web

Thierry Hamon

Bureau H202
Institut Galilée - Université Paris 13
&
LIMSI-CNRS
hamon@limsi.fr
http://perso.limsi.fr/hamon/Teaching/ProgWeb-20132014/

Présentation du cours

- Objectifs de l'enseignement : Acquérir les notions de base en programmation Web
  - Manipulation de documents Web
  - Mise en place de formulaires et utilisation du Javascript/AJAX/JQuery
  - Programmation en PHP et utilisation d'une base de données

  Également : connaître les sources d'informations complémentaires
- Répartition des enseignements : 4 séances de cours/TP de 3h ou 4h30
- Devoir développé au cours des TPs autour de la gestion de la base de données Auto-Ecole

HTML

HyperText Markup Language

- Langage de structuration d'un texte à l'aide de balises (définition logique d'un document)
- Remarque : Tout document doit pouvoir s'afficher dans n'importe quel navigateur, mais l'affichage dépend de l'interprétation du navigateur
- Problèmes de syntaxe et validation
  - HTML 4.01 : syntaxe formelle et précise (permet une validation automatique)
    Doit être respectée par tout document HTML
  - En pratique, peu de documents valides (pas de respect de la norme, même par les éditeurs)
  - Navigateurs tolérants aux erreurs de syntaxe
- Evolution : HTML 5

XML

eXtensible Markup Language

- Langage de description de documents
- Description de la structure logique du document, indépendante de l'application
- Format universel des documents structurés et des données
- Utilisation de langage de mise en page suivant l'application
  Exemples :
  - XML $\longrightarrow$ XHTML (PHP)
  - XML $\longrightarrow$ autres formats (eXtented Stylesheet Language - XSL)

Pour aller plus loin :

`http://natalia.grabar.perso.sfr.fr/cours/xml.pdf`

XHTML

eXtensible HyperText Markup Language

- HTML compatible XML
- Transformation formelle respectant les règles syntaxique du XML :
  - Déclaration d'une déclaration de type de documents (DTD)
  - Chaque balise a une balise de fin
  - Chaque attribut a une valeur entre guillemets
  - etc.

Document Type Definition (DTD)

- Document XML respectant la syntaxe XML : document *bien formé*
- Mais besoin de décrire les contraintes propres à un format (structure d'une classe de documents) : définition d'une DTD
- Document XML respectant une DTD : document *valide* par rapport à une DTD donnée (outil pour la validation `xmllint`)

Document Type Definition (DTD)

DTD : $\longrightarrow$ Spécification d'une grammaire pour un langage

- DTD interne : placée au début du document, associée au document
- DTD externe : placée dans un fichier séparé, associée aux documents y faisant référence
- DTD mixte

Exemple XML (1)

- `etudiants.dtd`

```
<!ELEMENT etudiants (etudiant*)>
<!ELEMENT etudiant (nom,
                    prenom,
    INE)>

<!ELEMENT nom     (#PCDATA)>
<!ELEMENT prenom  (#PCDATA)>
<!ELEMENT INE     (#PCDATA)>
```

- `exemple1.xml` (DTD externe)

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE etudiants SYSTEM "etudiants.dtd">

<etudiants>
  <etudiant>
    <nom>Pierre</nom>
    <prenom>Dupont</prenom>
    <INE>1234567890</INE>
  </etudiant>
</etudiants>
```

Exemple XML (2)

- `exemple2.xml` (DTD interne)

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<!DOCTYPE etudiants [
  <!ELEMENT etudiants (etudiant*)>
  <!ELEMENT etudiant (nom,
                      prenom,
          INE)>
  <!ELEMENT nom     (#PCDATA)>
  <!ELEMENT prenom  (#PCDATA)>
  <!ELEMENT INE     (#PCDATA)>
]>

<etudiants>
  <etudiant>
    <nom>Pierre</nom>
    <prenom>Dupont</prenom>
    <INE>1234567890</INE>
  </etudiant>
</etudiants>
```

XSL

eXtended Stylesheet Language

- Complément indispensable au XML (interprétation et sélection des balises XML)
- Dérivé du XML (donc respect de la syntaxe XML)
- Langage des feuilles de styles XML
- 3 composants :
  - XSLT : permet la transformation d'un document XML en un autre format (XML, texte, XHTML, etc.)
  - XSL-FO (Formatting Objects) : permet le formatage d'un document XML pour un rendu optimisé à l'impression et la visualisation (PocketPC, etc.)
  - Xpath : permet d'adresser des parties d'un document XML

XSLT

- Transformation de documents XML en XML, (X)HTML, texte, etc.
- Possibilité de définir plusieurs formats de publication pour un même contenu
- Vision d'un document comme un arbre
- Parcours de l'arbre et application de règles de transformation
- Production d'un document en sortie
- Utilisation de feuille de style XSL

exemple de processeur XSLT : `xsltproc`

XPATH

- Expression de chemin dans un arbre et sélection de nœuds
- Utilisation par XSL, Xpointer et XQuery
- Chemin absolu (`/document/section`) ou relatif (`../section`)

XSD

XML Schema Definition

DTD :

- types pauvres et peu de contraintes sur le contenu
- Pas de gestion d'espace de nom
- Pas au format XML

XSD :

- définition de types et de contraintes sur les contenus
- Définition précisément le nombre d'apparitions d'un élément
- Espaces de noms
- Format XML

Exemple XSD (1)

- etudiants.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified"
           targetNamespace="http://foo.bar/etudiants/"
           xmlns:etudiants="http://foo.bar/etudiants/">

<xs:element name="etudiants" >
    <xs:complexType>
      <xs:sequence>
        <xs:element  minOccurs="0" maxOccurs="unbounded" ref="etudiants:etudiant"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="etudiant">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nom" type="xs:string"/>
      <xs:element name="prenom" type="xs:string"/>
      <xs:element name="INE" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Exemple XSD (2)

- exemple3.xml

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>

<etudiants xmlns="http://foo.bar/etudiants/">
  <etudiant>
    <nom>Pierre</nom>
    <prenom>Dupont</prenom>
    <INE>1234567890</INE>
  </etudiant>
</etudiants>
```

Vers le web dynamique et interactif

- HTML : pas d'interactivité avec l'utilisateur. Les pages sont "statiques"
- Pages dynamiques : dépendantes des manipulations l'utilisateur
  Génération dynamique en fonction de paramètres de l'utilisateur
  Principaux types d'Interactions :
  - Coté client
  - Coté serveur

  $\longrightarrow$ On parle de programmation Web

Programmation Web côté client

- Exécution réalisée sur le client
- Navigateur capable de réaliser l'exécution
- Transfert du code `embarqués` dans la page HTML, depuis le serveur vers le client (*HTML-embedded scripting*)
- Exemples
  - Scripts : Javascript, Vbscript
  - Applets : Java, ActiveX (nécessite une machine virtuelle)
  - Plugins propriétaires

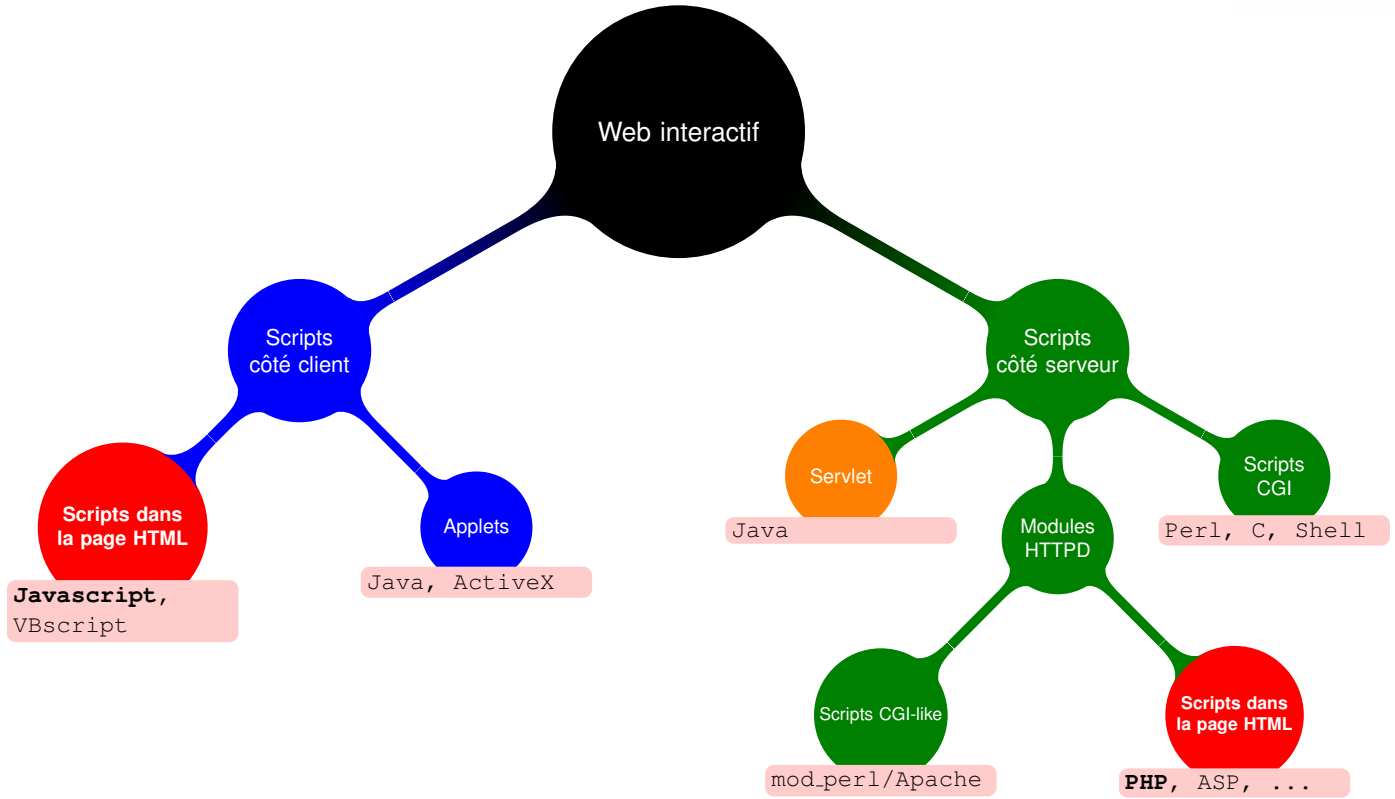
Programmation Web côté serveur

- Exécution coté serveur
- Résultat de l'exécution : page HTML envoyée par le serveur au navigateur
- Exemples :
  - CGI (*Common Gateway Interface*) – interface définissant le format d'échange entre le client et le serveur HTTP
    - Rôle du serveur : transmission des paramètres à un programme qui traite la requête et produit une page HTML (dynamique)

→ *Obsolète mais à la base de toutes les interactions client/serveur*

  - Interpréteurs intégrés au serveur HTTP (scripts embarqués dans la page HTML)
    - Modules du serveur httpd (PHP, ASP), scripts à la CGI (`mod_perl` d'apache)

Résumé



D'après `http://www710.univ-lyon1.fr/~hbriceno/teaching/cs/04_CS_http-web.pdf`

NB : *on peut/doit aussi mélanger les deux cotés*

Formulaires HTML

- Possibilité d'interactivité avec l'utilisateur
- Proposition de zone de dialogue
- Traitement des choix de l'utilisateur à l'aide de code CGI, PHP, ...
- Exemples :
  - Requêtes dans un moteur de recherche
  - Interrogation d'une base de données

Formulaires HTML – description

- Description des champs de saisies à l'aide de balises HTML
- Identification des zones par un nom symbolique l'utilisateur lui associe une valeur
- Soumission du formulaire : transmission des couples `nom/valeur` dans la requête HTTP
- Sans traitement sur les données, on parle de *client passif*
- Possibilité d'associer un traitement (sur le client) grâce à un évènement Javascript
  On parle de *client actif*

Eléments d'un formulaire

1. Champs de saisie de texte et boutons : `input`
   - Zone de texte (type par défaut) : `type="text"`
   - Zone de texte caché : `type="password"`
   - Cases à cocher : `type="checkbox"`
   - Boutons radio, minimum 2, un seul sélectionnable : `type="radio"` :
   - Soumission du formulaire `type="submit"`
   - Bouton de remise à zéro des champs : `type="reset"`
   - Bouton associé à du code JavaScript : `type="button"`
   - Bouton caché : `type="hidden"`
2. Menus déroulants, listes à ascenceurs : `select`
   - Liste avec 1 seul élément sélectionnable (pop list) : `size="1"`
   - Liste à choix multiples : `size="n"` avec $n > 1$
3. Zone de saisie d'un texte "long" : `textarea`

Exemple de formulaire

```
<html><head><title>Form example</title>
<script language="javascript">
function displayInfo() {
  document.getElementById("infofirstname").innerHTML = document.formulaire.firstname.value;
  document.getElementById("infolastname").innerHTML  = document.formulaire.lastname.value;
}
function resetInfo() {
  document.getElementById("infofirstname").innerHTML = "Not available";
  document.getElementById("infolastname").innerHTML  = "Not available";
}
</script></head>
  <body>
<div>
    <form name="formulaire" action="javascript:displayInfo()">
      <div>
<div>Firstname: <input type="text" name="firstname"/></div>
<div>Lastname: <input type="password" name="lastname"></div>
      </div><br/>
      <div style="text-align:center">
<input type="submit" value=" Ok " />
<input type="reset" value="Cancel" onclick="javascript:resetInfo();"/>
      </div>
    </form>
</div>
Information :
   <ul>
<li id="infofirstname">Not available</li>
<li id="infolastname">Not available</li>
   </ul>
</html>
```

Méthodes GET et POST

- Méthode GET :
  - Transmission des données relatives aux champs du formulaire dans l'URL
  - Récupération des données grâce à une variable (d'environnement, un tableau, suivant les langages)
- Méthode POST :
  - Transmission des données relatives aux champs du formulaire dans le corps de la requête HTTP
  - Positionnement de Content-type et Content-length
  - Récupération des données sur l'entrée standard

NB : Il est nécessaire d'analyser la requête (*parsing*, pouvant être réalisé par une fonction du langage de programmation utilisé)

Méthode GET

- Méthode standard pour
  - la récupération d'un document
  - l'activation d'un script ou d'une commande coté serveur (avec transmission des données)
- Contenu de la requête toujours vide
- Réponse du serveur :
  - une ligne décrivant l'état de la requête
  - une entête
  - le contenu demandé

  - En cas d'échec, contenu de la réponse décrivant la raison de l'échec

Utilisation de la méthode GET

- Données du formulaire transmis par l'intermédiaire de l'URL (après `?`, champs séparés par `&`)
- Exemple :
  `GET /index.php?login=titi&passwd=titi HTTP/1.1`
  (deux champs – login et passwd – en clair ...)
- Conservation dans un *bookmark* des données saisies et transmises par le formulaire
- Taille maximum de l'URL : 4ko

Utilisation de la méthode POST

- Transmission des données au serveur dans le corps de la requête
- Exemple

```
POST index.php HTTP/1.1
User-Agent: Mozilla/4.0 (compatible;MSIE 6.0;Windows
Host: localhost

Accept : */*

Content-type: application/x-www-form-urlencoded

Content-length: 36

login=titi&password=titi
```

NB : Mot de passe toujours en clair ...

Utilisation de la méthode HEAD

- Similaire à la méthode GET mais offre également la possibilité de récupérer l'entête relative à un document :
  - date de dernière modification (utile pour les caches et Javascript)
  - taille du document
  - type du document (filtrage en fonction du type par le client)
  - type du serveur (utilisation de requêtes spécifiques)
- NB : ces informations ne sont pas forcément fournies par le serveur

Analyse des données du formulaire

- Récupération des couples `nom`/`valeur` associés aux champs du formulaire
- Exemples de parsers existants :
  - Perl : (cgi-lib.pl) - `http://cgi-lib.berkeley.edu`
  - Perl : module CGI (voir exemples `http://www.validome.org/doc/HTML_fr/cgiperl/modules/cgi.htm`)
  - C : `http://www.boutell.com/cgic/`, `http://libcgi.sourceforge.net`, etc.
  - PHP : voir les tableaux associatifs `_POST` et `_GET`

Sessions (1)

- Transmission d'informations d'une page à l'autre (sur un site donné)
- Stockage coté serveur (sous forme de fichier)
- Evite donc la de perte d'informations ou de redemander une information (identifiant, mot de passe, panier, etc.)
- Fin de la session lorsque
  - l'utilisateur quitte le navigateur
  - l'utilisateur quitte le site
  - la date d'expiration est atteinte
- Attention : par défaut, sous apache/php5, confusion/remplacement par des cookies

Exemple de session

```
<?php
session_start();

$_SESSION['prenom'] = 'Pierre';
$_SESSION['nom'] = 'Dupont';
$_SESSION['INE'] = '1234567890';
?>

<html>
    <head>
        <title>Exemple de session</title>
     </head>
     <body>

<h3>Information sur l'utilisateur</h3>
     <p>
       Nom : <?php echo $_SESSION['nom']; ?><br/>
       Prénom : <?php echo $_SESSION['prenom']; ?><br/>
       Numéro INE : <?php echo $_SESSION['INE']; ?><br/>
     </p>

     <p>
         <a href="session2.php">Page suivante</a><br />
     </p>

     </body>
</html>
```

Exemple de session

```
<?php
session_start();
?>

<html>
    <head>
        <title>Page suivante de l'exemple de session</title>
     </head>
     <body>

     <p>
<h3>Informations sur l'utilisateur de la page précédente</h3>
     <p>
        Nom : <?php echo $_SESSION['nom']; ?><br/>
        Prénom : <?php echo $_SESSION['prenom']; ?><br/>
        Numéro INE : <?php echo $_SESSION['INE']; ?><br/>
     </p>
     </body>
</html>
```

Cookies (1)

- Stockage d'informations par le serveur, chez le client
- Solution au caractère sans état de HTTP

Description :

- Une chaîne de caractères url-encodée de 4ko au maximum stockée sur le disque du client
- Informations associées à un ensemble d'URL, et envoyées lors de chaque requête vers une de ces URL

Cookies (2)

Exemples d'utilisation :

- Propagation d'un code d'authentification (permet d'éviter une authentificaton lors de chaque requête)
- Identification dans une base de données
- Envoi d'éléments statistique au serveur

Installation d'un cookie sur un client

$\longrightarrow$ Inclusion de la directive `Set-cookie` dans l'entête de la réponse HTTP (lors de la première connexion)

```
Set-Cookie: nom=valeur; expires=date;
path=chemin_acces; domain=nom_domaine; secure
```

- `nom/valeur` : contenu du cookie (champs obligatoire), sans caractère espace, `;` et `,`
- `date` : date à laquelle le cookie devient invalide
- `path=chemin_acces` : chemin contenu dans l'URL pour lequel le cookie est valable

  Exemple : `path=/home`
- `domain` : nom de domaine associé au serveur pour lequel le cookie est valable
- `secure` : le cookie et valable si la connexion est sécuriée

Possibilité d'insérer plusieurs cookies

Utilisation d'un cookie par un client

- A chaque requête du client, vérification si un cookie est associé à la requête (consultation de la liste des cookies)
- Si oui, utilisation de la directive cookie dans l'entête de la requête HTTP

```
Cookie: nom1=valeur1; nom2=valeur2; ...
```

Limitations (première spécification) :

- Stockage de 300 cookies au plus
- 20 cookies par domaine au plus
- Taille limite d'un cookie : 4 ko

Exemple

(dans du javascript)

```
var Cookies = {};

var allCookies = document.cookie.split('; ');
for (var i=0;i<allCookies.length;i++) {
   var cookiePair = allCookies[i].split('=');
   Cookies[cookiePair[0]] = cookiePair[1];
}

var x = Cookies['exempleCookie'];
if (x) {
   alert('Le cookie exempleCookie est encore actif\n Valeur du Cookie
                  (initialisé lors d\'une précédente visite) : ' + x);
}

function sauvegardeCookie(name) {
  var x = document.forms['cookieform'].login.value;
  if (!x) {
    alert('Indiquez un login');
  } else {
    var date = new Date();
    date.setTime(date.getTime()+(1*60*60*1000));
    var expires = "; expires="+date.toGMTString();
    document.cookie = name+"="+x+expires+"; path=/";
    Cookies[name] = x;
    alert('Cookie created');
  }
}
```

Exemple

(dans du javascript)

```
function lectureCookie(name) {
alert('La valeur du cookie est ' + Cookies[name]);
}

function suppressionCookie(name) {
    var date = new Date();
    date.setTime(date.getTime()+(-1*60*60*1000));
    var expires = "; expires="+date.toGMTString();
    document.cookie = name+"="+expires+"; path=/";
    Cookies[name] = undefined;
    alert('Cookie erased');
}
```

Conclusion

A voir :

- Javascript
- PHP
- Manipulation étendue des données : AJAX & JQuery