Web dynamique (M2105)

Eugen Dedu

Maître de conférences
Univ. de Franche-Comté, IUT de Belfort-Montbéliard
Dépt. R&T, 1ère année
Montbéliard, France
mars 2016

http://eugen.dedu.free.fr
eugen.dedu@univ-fcomte.fr

Intégration dans la formation R&T

- Pré-requis (S1) :
  - initiation au développement Web
  - langages de programmation (bases + consolidation)
  - bases de données
- Prolongement : aucun

- FI/FA : 6h CM, 6h TD, 18h TP

- Objectif : savoir concevoir des pages Web simples mais dynamiques en utilisant les dernières technologies
  - exemple : page Web montrant l'utilisation du CPU et du disque dur d'un serveur
- Ce n'est pas faire des sites Web complexes

Exemples de besoins pour un administrateur réseau

- Cédric Michelot, administrateur réseau du pôle :
  - une page Web qui affiche l'espace disque utilisé et l'utilisation du CPU en graphique (en utilisant PHP et JavaScript) avec l'historique récent (utilise PHP et un script shell)
  - gestion des absences des étudiants, avec droits différents pour les enseignants (utilise PHP et MySQL)
  - gestion des comptes utilisateurs pour src-projets : l'étudiant introduit son user+pwd pour activer son compte src-projets (utilise PHP)

Plan du cours

- Scripts côté serveur (PHP)
  - sessions ("votre panier contient 2 éléments")
  - intégration avec une base de données (MySQL)
- Scripts côté client (JavaScript)
- Interactions asynchrones client-serveur (AJAX)
- Sécurisation des sites (https)
- (CMS : mediawiki, joomla, drupal etc.)
- (Divers : XML, XSD, XSLT)

Où les pages Web sont modifiées

- Au début du Web, les pages Web étaient statiques, mais de nos jours beaucoup (voire la plupart ?) des pages sont dynamiques
- La page (fichiers HTML et CSS, y compris les images et autres objets) peut être modifiée par le serveur, le réseau ou le client

- Exemples de pages dynamiques :
  - "l'heure courante est ...h ...m ...s" – il veut mieux que ce soit le client qui l'actualise en permanence
  - "les maisons à vendre dans la région choisie sont : ..." – il vaut mieux que ce soit le serveur qui l'exécute
  - "les maisons à vendre aujourd'hui à ...h ...m ...s sont : ..." – il vaut mieux utiliser le serveur et le client
- Une partie ou toute la page est écrite en un langage à exécuter, qui sort du HTML
- Le changement dépend de l'entrée de l'utilisateur, des conditions environnementales (par ex. la date courante, navigateur) etc.
- Voir plus tard une comparaison plus détaillée

Exemples de pages créées dynamiquement sur le serveur

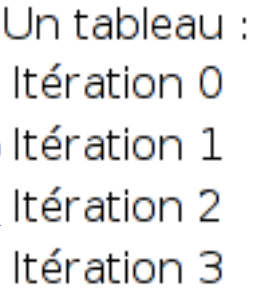
- formulaire fait en TP Web1
- google search
- recherche sur un site d'achats
- morpion, calculatrice, convertisseur devises
- page qui affiche toutes les images d'un répertoire du serveur
- Dans tous les cas, c'est la même page que le client demande (mais avec des paramètres différents en principe)
- Nous allons étudier PHP

Transformation d'une page Web dynamique avec un script côté serveur

Fichier exemple.php :

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Page simple</title>
</head>
<body>
  Un tableau :
  <table>
  <?php
    for ($i=0 ; $i < 4 ; $i ++)
      echo '<tr><td>Itération ' .
        $i . '</td></tr>';
  ?>
  </table>
</body>
</html>
```


```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Page simple</title>
</head>
<body>
  Un tableau :
  <table>
  <tr><td>Itération 0</td></tr>
  <tr><td>Itération 1</td></tr>
  <tr><td>Itération 2</td></tr>
  <tr><td>Itération 3</td></tr>
  </table>
</body>
</html>
```



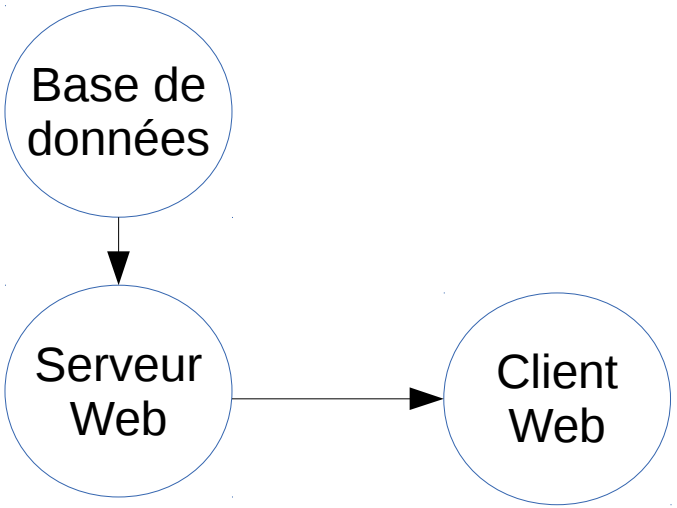
- Le script s'exécute sur le serveur Web
- Lors de la demande de la page, il s'exécute ***une seule fois***, pour générer la page, autrement dit une fois la page générée le script ne change plus la page (mais un script côté client peut continuer à la modifier)
- Le script sort du texte HTML

PHP

- Beaucoup de documentation sur PHP
- Voir cours séparé : http://www.cril.univ-artois.fr/~lecoutre/teaching/web/cours/slidesPHPFr.pdf
- http://www.lephpfacile.com/cours/
- http://cours-info.iut-bm.univ-fcomte.fr/docs/php/bigmanual.html
- http://public.iutenligne.net/informatique/langages/roose/php/general

PHP/MySQL

- Architecture 3-tier
- Quatre opérations principales SQL sur une base de données (BD) : select, insert, update, delete
- Trois méthodes (API) pour accéder à une BD MySQL :
  - extension MySQL de PHP – vieille, mais simple et suffisante pour nous, donc c'est celle que nous allons utiliser
  - extension mysqli de PHP
  - PHP Data Objects (PDO)
- Voir cours PHP/MySQL



Scripts côté client

- Exemple : page Web avec le temps qui s'affiche en permanence ?
- Le script s'exécute sur le client (navigateur)
- Interprété, pas compilé

  La date courante en php est : <?php echo time(); ?>.
  La date courante en php est : 10 mars 2015, 12h20'25".
- Langages : JavaScript (n'a rien à voir avec Java), ActionScript, VBScript etc.
- Nous allons étudier JavaScript, recommandé par W3C
- Bibliographie :
  - tout livre sur JavaScript
  - sites, par ex. http://fr.openclassrooms.com/informatique/cours/tout-sur-le-javascript

Insertion du code JavaScript

- Direct dans la page HTML
  - dans <script>, soit dans le body pour du code à s'exécuter au chargement de la page, soit dans head pour du code à s'exécuter plus tard
  - dans les balises même, par ex. pour les écouteurs (onmouseover d'un élément)
- Dans un fichier à part, avec <script src="scripts.js"></script>

Utilisation de JavaScript

- Exécution du code : au chargement de la page ou lors d'un événement utilisateur
- Interaction avec le navigateur (BOM) : afficher boîte de dialogue, lecture de l'historique, des paramètres de l'écran etc.
- Interaction avec la page Web (DOM)
  - vérification des champs de formulaires
  - mise en forme de la page
  - modification du contenu de la page

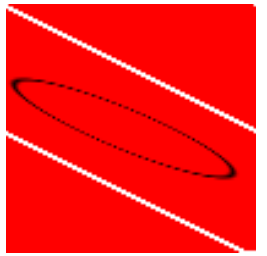
- Cours : http://www.cril.univ-artois.fr/~lecoutre/teaching/web/cours/slidesJavascriptFr.pdf

Canvas

- Le canvas est une région rectangulaire où on peut dessiner des graphiques et des images 2D dynamiquement par des scripts

```
<canvas id="dessin" width="100" height="100"></canvas>

<script>
var example = document.getElementById ("dessin");
var context = example.getContext ("2d");
context.fillStyle = "red";
context.fillRect (0, 0, 100, 100);
context.strokeStyle = "blue";
context.beginPath();
// x centre, y centre, rayon, start angle, fin angle
context.arc (50, 50, 20, 0, 2*Math.PI);
context.stroke ();
</script>
```



Canvas API

- fillStyle, strokeStyle
- createLinearGradient, createRadialGradient
- lineWidth
- fillRect, strokeRect
- beginPath, moveTo, lineTo, arc, stroke/fill
- fillText, strokeText
- drawImage

Exemple d'utilisation :
camembert sans utiliser des fichiers image

Comparaison entre scripts côté client et côté serveur

- morpion, calculatrice, convertisseur devises – est-il mieux de les écrire côté serveur ou côté client ou les deux ?
- Visibilité du code : si serveur, code invisible (par ex. mots de passe), si client, code visible
- Le côté serveur limite l'accès du client au serveur (sa BD, ses fichiers, ...)
- Le côté serveur consomme des ressources sur le serveur
- Le côté serveur a besoin d'un serveur Web, i.e. ces pages ne marchent pas en local
- Côté client – sécurité des clients (exécution en sandwich)
- Côté client – il faut faire attention aux incompatibilités
- JS est plus réactif aux changements dues à l'utilisateur, mais la page initiale est plus lourde ; PHP recharge toute la page, alors que JS charge/modifie juste une partie de la page (AJAX)

Technologies (langages) Web pour l'écriture des sites dynamiques

- Classification (parfois floue) selon l'endroit où la page est modifiée :
  - sur le serveur : CGI, PHP, JSP, ASP.NET, Node.js, ...
    - bibliothèques/frameworks PHP : symfony, zend, cakephp etc.
  - sur le client : JavaScript, ...
    - frameworks JS : jQuery, ...
  - combinaison des deux :
    - AJAX
    - Bootstrap – le développeur compile son HTML/CSS et y ajoute du JS haut niveau
- Classification selon la spécialisation :
  - jQuery – général
  - Bootstrap – pour affichage et RWD
  - d'autres – pour communication, pour graphisme (e.g. 3D) etc.

Interactions asynchrones client-serveur (AJAX)

- AJAX, Asynchronous JavaScript and XML, est une manière d'utiliser JavaScript dans une page Web pour faire des requêtes au serveur et d'utiliser la réponse dans la page même
- Exemples :
  - affichage en live du score des matchs à Wimbledon, sofascore
  - télécharger et afficher les items d'un menu en fonction du choix de l'utilisateur
  - moteur de recherche qui affiche des complétions au fur et à mesure que l'utilisateur écrit dans la zone de texte
- Le format des données : XML, JSON, HTML, voire texte tout simplement

AJAX, exemple

Script dans la page Web, exécuté sur le client :

```
var req = new XMLHttpRequest ();

req.onreadystatechange = function () {
    // 1=ouvert, 2=en-tête HTTP reçu, 3=contenu commencé, 4=contenu transféré
    if (this.readyState == 4) {
        if (this.status == 200)  // 200 = code HTTP pour transfert avec succès
            alert (this.responseText);  // 'This is the returned text'
        else
            alert ('Erreur : ' + this.status);  // An error occurred during the request
    }
};

req.open ('get', 'ajax.txt');  // extension txt
req.send ();
```

Fichier ajax.txt, sur le serveur :

```
This is the returned text
```

Évolutions possibles :

- le serveur modifie ajax.txt en temps réel et les clients le lisent régulièrement
- utiliser ajax.php et l'appeler avec ?param=..., où param est un paramètre écrit par l'utilisateur

Sécurité dans PHP

- Ne jamais faire confiance aux valeurs que l'utilisateur introduit, notamment dans les formulaires, car il peut les changer comme bon lui semble
  - vérifier toujours ces valeurs avant utilisation, par ex. entier entre 2 et 6
  - pour afficher une valeur, utiliser strip_tags pour enlever les balises, htmlspecialchars et htmlentities pour remplacer les caractères spéciaux
- error_reporting à 0, expose_php à off
- php.ini : disable_functions = system, exec et d'autres

Sécurité MySQL

- Injection SQL
  - sur une page on a une zone d'édition appelée produit pour le produit à chercher
  - dans le code php sur le serveur : $sql = "select * from produits where nom = $_GET['produit']";
  - si l'utilisateur met comme produit :
    - 20'; DROP produits; SELECT 'TOTO
  - résultat : la table produit est effacée !!
- Solution : en php, au lieu de $produit mettre mysql_real_escape_string ($produit)
- Autre solution : utiliser plusieurs utilisateurs, avec des droits spécifiques

Sécurité JavaScript

- XSS (Cross-Site Scripting)
  - injecter des scripts côté client introduits par un utilisateur dans la page vue par d'autres utilisateurs
- Exemple :
  - un site de News, avec des comptes, où les utilisateurs peuvent ajouter des news
  - M remarque que s'il met comme commentaire **J'aime les fleurs !<script src="...">**, le texte s'affiche et le script **s'exécute**
  - chaque utilisateur qui regarde la page verra le texte, et le script s'exécutera, et de plus cela passe inaperçu
  - le script récupère le cookie d'authentification et l'envoie au serveur de M, permettant donc à M de se faire passer pour l'utilisateur
- Problème : le navigateur/serveur exécute le **code** d'un utilisateur dans le contexte d'un **autre** utilisateur
- Solution : « escape » les caractères spéciaux : <, >, ", &, ', mais aussi d'autres, dans toutes les entrées des utilisateurs (y compris l'URL)

Sécurité des pages Web

- Tout le problème vient des entrées de l'utilisateur, car le programme, lui, n'est pas maléfique... Qu'est-ce que l'utilisateur peut introduire de bizarre ?
- Où peut-il introduire cela ?
  - directement : l'URL (via formulaires, voire écriture manuelle de l'URL)
    - écriture manuelle de l'URL : même si un combobox est utilisé, l'utilisateur peut modifier lui-même l'URL, par ex. : ...?name=Jean
  - indirectement : BD du serveur, par ex. l'utilisateur a spécifié qu'il s'appelle Jean
, et ce nom est retourné à tous les autres utilisateurs
- Dans tous les cas, c'est le développeur de la page Web qui est « coupable »
- Où le problème peut se manifester ?
  - autres utilisateurs (capture de leurs mots de passe)
  - serveur (effacement BD)

Complexité de la sécurisation des sites

- La sécurisation, d'un mot de passe par ex., apparaît à plusieurs endroits :
  - **l'utilisateur** prend soin de son mot de passe : choix d'un mot de passe complexe, ne pas l'écrire sur un papier, ne pas le taper quand un autre est à côté etc.
  - **le client** : le navigateur ne stocke pas les mots de passe en clair dans un endroit publique, pas de virus qui enregistre le clavier etc.
  - **le serveur** ne divulgue pas le mot de passe : ne pas tester la validité du le mot de passe dans la page (par ex. <script if pwd == "xyz" ...>), mais sur le serveur
    - aussi, le serveur doit stocker les mots de passe cryptés, sinon un intrus dans le serveur peut voir tous les mots de passe (imaginez la gravité de la situation pour le site d'une banque)
  - **le réseau** : si un utilisateur introduit son mot de passe sur un site non sécurisé, il circule en clair sur le réseau (navigateur -> serveur), donc un administrateur réseau peut le voir ; solution : crypter la communication, https
- Nous allons présenter le cryptage de la communication (https) seulement

HTTPS

- HTTP Secure
- Fournit cryptage et authentification du serveur
- HTTPS = HTTP sur TLS (Transport Layer Security) ou sur l'obsolète SSL
- Configurer le serveur pour utiliser HTTPS : par ex. http://www.onlamp.com/2008/03/04/step-by-step-configuring-ssl-under-apache.html

HTTPS – TLS

- TLS se met entre protocole de transport et application
- TLS utilise :
  - clé privée-publique
  - certificats X.509 ; infrastructure de clés publiques, une autorité de certification (CA) vérifie que le certificat (clé publique) correspond à son possesseur
- TLS est aussi utilisé par des programmes de courriel, de visioconférence, IM etc.
- Utilise le port 443 par défaut : le client est configuré pour demander au serveur d'utiliser https dans ce cas : le serveur envoie son certificat, le client le vérifie auprès de sa CA et si le nom du serveur correspond au nom dans le certificat, ensuite les deux cryptent la communication
- Voir http://en.wikipedia.org/wiki/Transport_Layer_Security pour plus d'informations

CMS

- Un CMS (Content Management System) permet d'éditer de manière simple des pages Web à partir d'une interface (Web) au lieu d'un programme
  - exemple avec wikipedia
- Couramment utilisés dans les blogs, serveur de news, vente en ligne etc.
- Caractéristiques usuelles : gestion des utilisateurs, rechercher un mot dans le site, stocker les versions anciennes (revision control), multilangue etc.
- Les pages Web sont généralement stockées dans une BD
- Exemples : WordPress, Joomla, Drupal, MediaWiki