Cours HTML / CSS / PHP / MySQL

Pierre-Nicolas Clauss

Laboratoire Lorrain de Recherche en Informatique et ses Applications

12 mars 2008

Plan

Plan

Plan

Généralités sur l'HTML

- Language descriptif
  - Pas de séquences de contrôle
  - Description de la sémantique du document
- Balises
  - Balise ouvrante : <TAG>
  - Balise fermante : </TAG>
  - Les deux en une : <TAG />
- Standardisé
  - W3C : http ://www.w3c.org
  - Dernière version : HTML 4.01
    - Strict
    - Transitional
    - Frameset
  - Validation automatique : http ://validator.w3.org
  - Balise DOCTYPE, sur la première ligne du fichier
    - <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http ://www.w3.org/TR/html4/loose.dtd">

Généralités sur l'HTML

- Encapsulation de type "pile"

(Mal)formation

```
<A> <B> </A> </B>   incorrect

<A> <B> </B> </A>   correct
```

- Structure arborescente

Arbre minimal

```
<HTML>
  <HEAD>
    <TITLE />
  </HEAD>
  <BODY />
</HTML>
```

Généralités sur l'HTML

- Commentaires entre <!-- et -->
- HTML + XML => XHTML
  - Balises toujours fermées
  - Transformation en d'autres formats
- HTML donne un découpage selon la sémantique du document
- Mise en forme visuelle
  - en HTML : moins lisible, plus de code
  - en CSS : plus lisible, regroupement et généralisation

Plan

Espaces

- Tous les espaces blancs (y compris \t et \n) sont ignorés
- Cas particulier : l'espace seul => reproduit tel quel

Exemple

```
Mon texte    avec des espaces
         et plusieurs lignes
n'apparait pas comme ceci
```

Résultat

```
Mon texte avec des espaces et plusieurs lignes n'apparait pas comme ceci
```

- Il faut utiliser des balises

Texte

Mise en forme au niveau paragraphe

-
 : aller à la ligne
-   : insérer un espace
- <P> : commencer un nouveau paragraphe
- entre <PRE> et </PRE>, tout est reproduit tel quel

Mise en forme au niveau phrase

- <EM> à </EM> : Mise en valeur (italique)
- <STRONG> à </STRONG> : Mise en valeur forte (gras)
- <CITE> à </CITE> : Citation courte (gras italique)
- <Q> à </Q> : Citation courte (entre guillements)
- <BLOCKQUOTE à </BLOCKQUOTE> : Citation longue (gras)

Texte

Mise en forme au niveau phrase

- <CODE> à </CODE> : Extrait de code source (gras italique petit)
- <ABBR> à </ABBR> : Abbréviation (gras italique)
- <ACRONYM> à </ACRONYM> : Acronyme (gras italique)
- ^à : Mettre en exposant
- _à : Mettre en indice
- <BIG> à </BIG> : Plus gros
- <SMALL> à </SMALL> : Plus petit
- Beaucoup d'autres : <DFN>, <SAMP>, <KBD>, <VAR>, <INS>, <DEL>, ...

Texte

Liens hypertexte

- Entre <A> et </A>
- Choisir le(s) bon(s) mot(s) pour servir de lien
- Liens internes (<A NAME=...>) ou externes (<A HREF=...>)
- Possibilité de lien sur une image
- Lier les pages de manière cohérente

Plan

Listes non-ordonnées

- <UL> marque le début de la liste et </UL> sa fin
- <LI> permet de commencer un nouvel item dans la liste

Exemple

```
<UL>
  <LI>1<SUP>er</SUP> element
  <LI>2<SUP>eme</SUP> element
  <LI>3<SUP>eme</SUP> element
</UL>
```

Listes ordonnées

1. <OL> marque le début de la liste et </OL> sa fin
2. <LI> permet de commencer un nouvel item dans la liste

Exemple

```
<OL>
  <LI>1<SUP>er</SUP> element
  <LI>2<SUP>eme</SUP> element
  <LI>3<SUP>eme</SUP> element
</OL>
```

Listes de définitions

- <DL> marque le début de la liste et </DL> sa fin
- <DT> permet de définir un titre pour une définition
- <DD> permet de définir le contenu d'une définition

Exemple

```
<DL>
   <DT><STRONG>Cout</STRONG>
   <DD>Prix de revient
   <DT><STRONG>Benefice</STRONG>
   <DD>Gain realise par une personne ou une collectivite
</DL>
```

Plan

Forme du tableau

- Un tableau commence par <TABLE> et finit par </TABLE>
- Une ligne dans un tableau commence par <TR> et finit par </TR>
- Une cellule dans une ligne commence par <TD> et finit par </TD>
- Une cellule d'en-tête dans une ligne commence par <TH> et finit par </TH>

Exemple : 2 lignes x 3 colonnes

```
<TABLE>
  <TR>
    <TD>Cellule 1,1</TD>
    <TD>Cellule 1,2</TD>
    <TD>Cellule 1,3</TD>
  </TR>
  <TR>
    <TD>Cellule 2,1</TD>
    <TD>Cellule 2,2</TD>
    <TD>Cellule 2,3</TD>
  </TR>
</TABLE>
```

Cellule 1,1	Cellule 1,2	Cellule 1,3
Cellule 2,1	Cellule 2,2	Cellule 2,3

Forme du tableau

- On peut donner un "titre" au tableau entre <CAPTION> et </CAPTION>, en dehors des définitions de lignes ou de cellules
- On peut ranger les lignes entre <THEAD> et </THEAD> pour indiquer qu'elles font partie de l'en-tête
- On peut ranger les lignes entre <TFOOT> et </TFOOT> pour indiquer qu'elles font partie du pied du tableau
- On peut ranger les lignes entre <TBODY> et </TBODY> pour indiquer qu'elles font partie des données
- Attention : <THEAD> et <TFOOT> doivent apparaître avant <TBODY>
- Il peut y avoir plusieurs <TBODY>

Forme du tableau

- On peut donner des indications de taille sur les colonnes entre <COLGROUP> et </COLGROUP>
- Entre ces balises, on indique avec <COL width=...> la taille d'une colonne
- On peut aussi utiliser <COLGROUP span=XX width=YY> pour spécifier XX colonnes de taille YY

- Les balises <TD> et <TH> ont les attributes :
  - rowspan pour indiquer sur combien de lignes s'étant la cellule
  - colspan pour indiquer sur combien de colonnes s'étant la cellule
  - align pour indiquer l'alignement horizontal dans la cellule (left, right ou center)
  - valign pour indiquer l'alignement vertical dans la cellule (top, middle, bottom)

Bordures du tableau

<TABLE> a les attributs :

- border pour indiquer l'épaisseur des bordures
- rules pour indique le type de bordure entre les cellules :
  - none : pas de bordure
  - groups : bordures entre les groupes de lignes (<THEAD>, <TFOOT>, <TBODY> et les groupes de colonnes (<COLGROUP>, <COL>)
  - rows : bordures entre les lignes uniquement
  - cols : bordures entre les colonnes uniquement
  - all : bordures autour de toutes les cellules

Plan

Plan

Mise en forme

- Séparation de la forme et du fond
  - HTML décrit le fond
  - CSS décrit la forme
- Centralisation de l'aspect visuel
- On insère du CSS entre :

```
<STYLE type="text/css">
<!--
  du CSS ici
-->
</STYLE>
```

- Ou on lie un fichier CSS avec :

```
<LINK rel="stylesheet" type="text/css" href="...">
```

- Ou on ajoute du CSS à une balise :

```
<BALISE style="...">
```

Plan

Format d'une classe CSS

- Le CSS est formé d'un ensemble de **classes**.
- Une classe s'écrit de cette façon :

```
nom {
   attribut:valeur;
   ...
}
```

- **nom** peut être :
  - Un nom de balise : les attributs s'appliquent à toutes ces balises
  - Un nom générique (commençant par un point) : les attributs s'appliquent aux balises utilisant `class="nom"` (sans le point)
  - Un mélange des deux, séparés par des virgules : les attributs s'appliquent suivant les deux points précédents
- **attribut** désigne quel élément visuel est modifié (couleur, bordure, fond, marges ...).
- **valeur** désigne par quelle valeur est remplacée l'attribut désigné

Exemple de CSS

style.css

```
body {
  font-family:Arial;
}
p {
  background-color:#F0C0C0;
  border: thin solid black;
}
.titre {
  color: yellow;
}
```

index.html

```
<HTML>
  <HEAD>
    <LINK rel="stylesheet" type="text/css" href="style.css">
    <TITLE>Exemple CSS</TITLE>
  </HEAD>
  <BODY>
    <H1 class="titre">Exemple</H1>
    <P>Un paragraphe avec bordure et couleur de font
  </BODY>
</HTML>
```

Plan

Plan

Pages dynamiques

Exemple statique

```
<HTML>
  <HEAD><TITLE>Page statique</TITLE></HEAD>
  <BODY>
    Nous sommes le 28/03/2007
  </BODY>
</HTML>
```

- Problème : Afficher une page différente en fonction de l'utilisateur, de l'environnement, ...
- Solution : Utiliser un langage de programmation évolué, par exemple PHP.

Présentation

- Langage récent (crée en 1994)
- Versions utilisée :
  - 4.3 (plus répandue)
  - 5.0 (avec une couche objet)
- Langage de script
  - Langage interprété
  - Présence d'un interpréteur côté serveur
- Intégré au code HTML
- Syntaxe proche du C et du Java
- Interface simple avec beaucoup de SGBD

Modèle d'exécution

1. Le client demande une page PHP
2. Le serveur web exécute le code de la page
   1. Lancement de l'interpréteur
   2. Exécution du code
3. Le serveur web renvoie le résultat de l'exécution
4. Le client affiche le résultat

- Pour le client, il est impossible de voir le code PHP
- Seul le résultat de l'exécution est récupéré par le client

Premier exemple

Code côté serveur

```
<HTML>
  <HEAD><TITLE>Page dynamique</TITLE></HEAD>
  <BODY>
  <?php
    echo("Nous sommes le ");
    echo(date("j/m/Y"));
  ?>
  </BODY>
</HTML>
```

Résultat côté client

```
<HTML>

  <HEAD><TITLE>Page statique</TITLE></HEAD>

  <BODY>
    Nous sommes le 12 mars 2008
  </BODY>
</HTML>
```

Plan

Mélange HTML/PHP

- PHP s'intègre dans l'HTML entre `<?php` et `?>`
- Les instructions se finissent par ;
- Les commentaires sont soit entre /*et */, soit après //
- Manuel complet en français : http ://www.php.net/manual/fr

Les variables

- Les variables sont préfixées par $
- Leur nom suit les règles classiques
- Exemple : `$my_var_03`
- Les noms sont sensibles à la casse : `$var` $\neq$ `$Var`

- Pas de déclaration, typage implicite
- Exemple :

```
$my_var_03 = 54;      // Maintenant, c'est un entier
$my_var_03 = "pif";   // Maintenant, c'est une chaine
```

- Attention aux fautes de frappes dans les noms de variables

Les types

- Entiers : `54`
- Flottants : `54.3`
- Chaînes : `"54"` ou `'54'`
- Booléens : `false` ou `true`
- Tableaux

- Fonctions de test :
  - `isset($var)` : renvoie `true` si `$var` existe
  - `unset($var)` : détruit `$var`
  - `is_integer($var)`, `is_string($var)`, ... : renvoie `true` si `$var` est un entier, une chaîne, ...

Les constantes et l'affichage

Constantes

- On les définit à l'aide de la commande `define`
- Exemples : `define("PI", 3.14)`
- On les utilise directement (sans $) : `echo(PI)`

Affichage

- On peut afficher avec la commande `echo` (avec ou sans parenthèses)
- `print` est équivalente à `echo`
- On peut faire un affichage comme en C avec `printf`

Opérateurs

- Arithmétiques : + − * / % ++ −−
- Affectation : = .= += −= *= /= %=
- Comparaison : == < != > === <= !== >=
- Logiques : and && or || xor !
- Conditionnel : ... ? ... : ...

Conditionnelles et boucles

```php
if(cond) {
...
}
else if(cond) {
...
}
else {
...
}
```

```php
switch(expr) {
  case VALEUR_1:
    ...
    break;
  case VALEUR_2:
    ...
    break;
  default:
    ...
    break;
}
```

```php
for(init; cond; modif) {
...
}
```

```php
while(cond) {
...
}
```

```php
do {
...
}while(cond);
```

Les tableaux

- Chaque élément du tableau a une **clé** et une **valeur**
- Pas de déclaration du tableau
- Les valeurs des éléments ne sont pas forcément toutes du même type
- Exemple de remplissage à la volée :

```
$tab[0] = 54;
$tab[1] = "pif";
$tab["paf"] = false;
```

- Exemple de remplissage direct :

```
$tab = array(54, "pif");
$tab = array("paf" => false);
```

Parcours de tableaux

- Parcours "classique" avec `for`
- Parcours spécifique :

```
foreach($tab as $value) {
...
}
```

```
foreach($tab as $key => $value) {
...
}
```

Les chaînes de caractères

- Délimitées par ' : contenu non interprété
- Délimitées par " : contenu interprété
- Les unes peuvent contenir les autres
- Concaténation avec .
- Exemple :

```
$pif = "toto";                  // Contient "toto"
$paf = "␣comme␣$pif";           // Contient " comme toto"
$pouf = 'pas␣comme␣$pif';       // Contient "pas comme $pif"
$bim = $pif.$paf;               // Contient "toto comme toto"
```

- Accès à un caractère : `$bim[0]`
- `strlen($str)` : longueur de `$str`
- `substr($str, start[, len])` : sous-chaîne de `$str` commençant au caractère start, et faisant éventuellement len caractères de long
- Comparaison avec ==, === ou `strcmp`

Les fonctions

```
function ma_fonc($param1, $param2, ...) {
   ...
   return ...;
}
```

- Pas de type pour les paramètres ou la valeur de retour
- Nombre fixé de paramètres
- Le nom ne commence pas par $
- Le nom est insensible à la casse
- Le résultat est renvoyé avec la commande `return`
  - Une seule valeur de retour
- Passage des paramètres par valeur (par défaut)
  - Passage par référence : &$param

- Les variables utilisées à l'intérieur d'une fonctions sont détruites à la fin, sauf :
  - si on les définit avec static
  - si on les définit avec global

```
function ma_fonc() {
  static $appels = 0;
  $appels++;
  echo("J'ai ete appellee $appels fois");
}

function ma_fonc2() {
  global $var;
  $var = 54;
}

$var = 0;
ma_fonc2();
echo($var);
```

Inclusion de fichiers

- On utilise `require(" fichier ")`, `include(" fichier ")`, `require_once(" fichier ")`, `include_once(" fichier ")`
- Les variantes `include` provoquent des warnings au lieu d'erreurs en cas de problème
- Les variantes `_once` n'incluent le fichier que si celui n'a pas déjà été inclu

Plan

Utilisation des formulaires

On peut "dialoguer" avec le visiteur en utilisant les formulaires

- En méthode GET : données encodées dans l'URL

  index.php?var=value&var2=value2...

- En méthode POST : données cachées mais pas de navigation avec Précédent/Suivant

Exemple de formulaire

```
<FORM action="traitement.php" method="post">
   prenom : <INPUT type="text" name="prenom">
   age : <INPUT type="text" name="age">
  <INPUT type="submit" value="Envoyer">
</FORM>
```

Eléments de formulaire

Il existe différents type pour les balises INPUT :

- text : une zone de texte sur une seule ligne
- password : idem, mais avec affichage d'étoiles
- file : permet la selection d'un fichier
- checkbox : une case à cocher
- button : un bouton simple (pas d'action sans javascript)
- hidden : un champ "texte" caché
- radio : un bouton d'option
- reset : un bouton de remise à zéro
- submit : un bouton de soumission

PHP avec formulaires

- Le script de traitement des formulaires reçoit un tableau pré-rempli
- $_GET pour la méthode GET
- $_POST pour la méthode POST
- Il contient les données du formulaire

Exemple de traitement

```
<?php
   $prenom = $_POST["prenom"];
   $age = $_POST["age"];
   echo("Bonjour $prenom, vous avez $age ans");
?>
```

Persistence des données

- On veut parfois garder de l'information entre plusieurs pages :
  - Login / Password
  - Préférences de navigation
  - Sélection de produits à acheter (panier, ...)
- On utilise donc les sessions PHP.
  - Les sessions permettent de stocker des informations côté serveur
  - Elles sont identifiées par un numéro qui reste valide tant que le visiteur reste connecté
  - Le numéro est transmis au serveur soit dans l'URL, soit dans un cookie
- Les données se placent et se récupèrent dans `$_SESSION`, comme pour les formulaires

Utilisation des sessions

- Les sessions utilisent les cookies : il faut donc ouvrir la session **avant** d'afficher quoi que ce soit (voir fonction `setcookie`)
- Note : Les valeurs des cookies sont dans le tableau pré-rempli `$_COOKIE`
- Le cookie utilisé (ou la variable dans `$_GET` à defaut) s'appelle PHPSESSID
- La session existe dès qu'elle est crée et jusqu'à ce qu'elle soit détruite
  - Création (et réouverture) : `session_start()`
  - Destruction : `session_destroy()`
  - Note : les session s'autodétruisent après un certain temps (généralement 30 min)

Plan

Plan

Définition

- SQL est un langage puissant de requête
- Il permet de faire une demande complexe à une base de données dans un langage proche de l'anglais
- On l'utilise pour récupérer, ajouter, supprimer et créer des données dans une base de données
- Les bases de données utilisent des tables :
  - chaque ligne est un enregistrement de champs avec des valeurs
  - les requêtes se font sur ces champs

Exemple

Table 'PersosXVI'

Nom	Prenom	Age	Activite
CARTIER	Jacques	44	Explorateur
CALVIN	Jean	26	Réformateur
CHASTEL	Jean	19	Assassin
PARE	Ambroise	44	Chirurgien

Requête interrogative

```
SELECT * FROM PersosXVI;
```

Requête sélective

```
SELECT Nom, Prenom FROM PersosXVI;
```

Requête restrictive

```
SELECT * FROM PersosXVI WHERE Prenom = 'Jean';
```

On peut complètement mélanger les types de requêtes

Exemple complexe

Table 'PersosXVI'

Nom	Prenom	Age	Activite	Roi
CARTIER	Jacques	44	Explorateur	3
CALVIN	Jean	26	Réformateur	1
CHASTEL	Jean	19	Assassin	4
PARE	Ambroise	44	Chirurgien	2

Table 'RoiXVI'

Id	Nom
1	Henri II
2	Charles IX
3	Francois Ier
4	Henri IV

Requête croisée

```
SELECT p.Nom, p.Activite FROM PersosXVI p, RoiXVI r WHERE p.Roi = r.Id AND r.Nom = 'Francois␣Ier'
ORDER BY p.Nom DESC;
```

Résultat

CARTIER	Explorateur

Exemple de fonctions

Table 'PersosXVI'

Nom	Prenom	Age	Activite
CARTIER	Jacques	44	Explorateur
CALVIN	Jean	26	Réformateur
CHASTEL	Jean	19	Assassin
PARE	Ambroise	44	Chirurgien

Décompte

```
SELECT count(*) FROM PersosXVI;
```

Somme

```
SELECT sum(Age) FROM PersosXVI;
```

Ce type de requête peut simplifier le code de traitement

Manipulation des données

Table 'PersosXVI'

Nom	Prenom	Age	Activite
CARTIER	Jacques	44	Explorateur
CALVIN	Jean	26	Réformateur
CHASTEL	Jean	19	Assassin

Requête d'insertion

```
INSERT INTO PersosXVI(Nom, Prenom, Age, Activite) VALUES('PARE', 'Ambroise', 44, 'Chirurgien');
```

Requête de mise à jour

```
UPDATE PersosXVI SET Age = 43 WHERE Nom = 'CARTIER';
```

Requête de suppression

```
DELETE FROM PersosXVI WHERE Prenom = 'Jean';
```

Plan

Accès au serveur

Connexion

- `$c = mysql_connect("localhost", "login", "mdp");`
- `$c` est `false` en cas d'erreur de connexion

Choix de la base

- `$s = mysql_select_db("nom", $c);`
- `$s` est `false` en cas d'erreur

Requêtes

Exécution

- `$res = mysql_query("une requete", $c);`
- `$res` est `false` en cas d'erreur

Récupération de(s) résultat(s)

- `$row = mysql_fetch_row($res);`
- `$tab = mysql_fetch_array($res);`
- `$obj = mysql_fetch_object($res);`
- `$row[0]; $row[1]; ...`
- `$tab["champ"]; ...` ou `$tab[0]; ...`
- `obj->champ; ...`

Fonctions annexes

- mysql_num_rows($res) : retourne le nombre de résultats d'un SELECT
- mysql_affected_rows($res) : retourne le nombre de ligne affectées par un INSERT, un UPDATE ou un DELETE
- mysql_insert_id($c) : retourne le dernier incrément d'un champ AUTO_INCREMENT

Attention aux injections SQL !