Conception de sites web dynamique

HTML – CSS – JAVASCRIPT – PHP
BASE DE DONNÉES

IF04U050

David Bouchet

david.bouchet.paris5@gmail.com

Conception de sites web dynamique

PHP
1re partie

Pages dynamiques

Javascript :

- Adapter la mise en page
- Ouvrir de nouvelles fenêtres
- Réagir à des actions de l'utilisateur
- Traiter des infos saisies par l'utilisateur

localement, sur le client

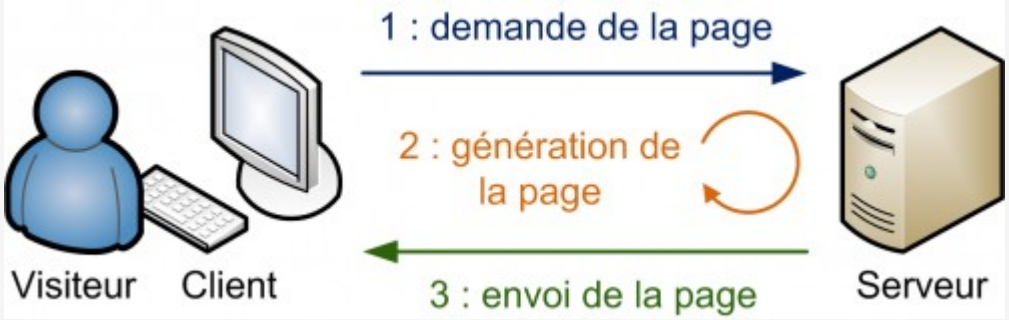
PHP : construire une page selon

- le profil de l'utilisateur
- les données (variables) du serveur
- les données envoyées par l'utilisateur

sur le serveur

Web dynamiques

Programmation côté serveur -> Pages web dynamiques



- Le serveur web interprète le code PHP et génère le code HTML
- Le code HTML est transmis au client

Pages dynamiques

Exemple statique

```
<!DOCTYPE html>
<html>
        <head>
                <title>Date - Heure</title>
                <meta charset="utf-8" />
        </head>
        <body>
                <p>Date de consultation du site :</p>
                <p>24/02/2016 12:10:09</p>
        </body>
</html>
```

Problème : comment afficher une page différente en fonction de l'utilisateur, de l'environnement, des données . . . ?

Affichage

Date de consultation du site :
24/02/2016 12:10:09

Exemple JavaScript

Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>Date - Heure</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <p>Date de consultation du site :</p>
    <p id="date"></p>
    <script>
      var d  = new Date();
      var options = { day: "2-digit", year: "numeric", month: "2-digit",
                hour: "2-digit", minute: "2-digit", second: "2-digit" };
      date.innerHTML = d.toLocaleString("fr-FR", options);
    </script>
  </body>
</html>
```

Affichage

Date de consultation du site :
24/02/2016 12:10:09

Exemple PHP

Code sur le serveur :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Date - Heure</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <p>Date de consultation du site :</p>
    <p><?php echo date("d/m/Y H:i:s"); ?></p>
  </body>
</html>
```

Code envoyé au client :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Date - Heure</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <p>Date de consultation du site :</p>
    <p>24/02/2016 12:10:09</p>
  </body>
</html>
```

Affichage :

Date de consultation du site :
24/02/2016 12:10:09

PHP

- Le client demande une page PHP (fichier.php)
- Le serveur web exécute le code de la page
  - Lancement de l'interpréteur
  - Exécution du code
- Le serveur web renvoie le résultat de l'exécution = page HTML (avec evt CSS et javascript)
- 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

Exemple classique : Hello World

Pseudo-balise PHP

```
<html>
   <head>
      <title>Ma premiere page</title>
   </head>
   <body>
      <?php
         echo "<h1>Hello World</h1>";
      ?>
   </body>
</html>
```

Le serveur interprète le code PHP en HTML :

```
<html>
   <head>
      <title>Ma premiere page</title>
   </head>
   <body>
      <h1>Hello World</h1>
   </body>
</html>
```

Code visible par le client

Autres pseudo-balises PHP (à éviter)

Éviter d'utiliser ces balises pour des raisons de portabilité (configuration du serveur nécessaire pour leur activation)

- Format short

```
<?
    echo “this is short style”;
?>
```

- Style JavaScript:

```
<SCRIPT LANGUAGE=“php”>
    echo “this is script style”;
</SCRIPT>
```

- Style ASP :

```
<%
    echo “this is ASP style”;
%>
```

Commentaires

- Similaire à beaucoup de langages

```php
<?php

// ligne commentée

# ligne commentée

/* commentaire
sur plusieurs
lignes echo $a */

?>
```

- Les commentaires sont ignorés par le serveur *php*

Les types de données (1)

8 types de base :

- boolean : booléen → TRUE ou FALSE (insensible à la casse)
- Integer / int : entier
- float / double : nombre à virgule flottante
- string : chaîne de caractères
- array : tableau
- object : objet
- resource : référence vers une ressource externe
- NULL : variable sans valeur

Les types de données (2)

Typage dynamique

- Pas besoin d'affecter un type à une variable avant de l'utiliser
- La même variable peut changer de type en cours de script
  - Note : les variables issues de l'envoi des données d'un formulaire sont du type *string*

Variables (1)

- PHP est **faiblement typé**
- Pas nécessaire de déclarer le type d'une variable
- Variables précédées du symbole « **$** » (dollar)

```
<body>
    <?php
        $toto = 5;
        echo $toto;
    ?>
</body>
```

- Noms des variables sensibles à la casse :
  $Toto ≠ $toto

Variables (2)

```php
<?php

   $x = FALSE;                          // boolean
   $x = TRUE;                           // boolean

   $x = 10;                             // integer
   $x = 1.45;                           // float
   $x = 0x1A;                           // integer (hexadecimal)
   $x = 0b100;                          // integer (binaire)

   $prenom = "Nicolas";                 // string
   $nom = 'DUPONT';                     // string
   $full_name = $prenom." ".$nom;       // string

   $tab[1] = 10;                         // array
?>
```

Chaînes de caractères

```php
// Guillemets simples
$s = 'Hello World';

// Guillements doubles
$s = "Hello World";

// Concaténation :
$s1 = "Hello";        // s1 = "Hello"
$s2 = "World";        // s2 = "World"
$s3 = $s1." ".$s2;    // s3 = "Hello World"

// Afficher une variable :
echo "La variable \$x vaut $x";  // Affiche : La variable $x vaut 10
echo "La variable $x vaut $x";   // Affiche : La variable 10 vaut 10
echo "La variable \$x vaut ".$x; // Affiche : La variable $x vaut 10
```

Caractères échappés

Séquence	Signification
\n	Fin de ligne (LF ou 0x0A (10) en ASCII)
\r	Retour à la ligne (CR ou 0x0D (13) en ASCII)
\t	Tabulation horizontale (HT or 0x09 (9) en ASCII)
\v	Tabulation verticale (VT ou 0x0B (11) en ASCII) (depuis PHP 5.2.5)
\e	échappement (ESC or 0x1B (27) en ASCII) (depuis PHP 5.4.4)
\f	Saut de page (FF ou 0x0C (12) en ASCII) (depuis PHP 5.2.5)
\\	Antislash
\$	Signe dollar
\"	Guillemet double

Différences pour les guillemets simples

Les guillemets simples, ne prennent pas en compte les variables ni certains caractères échappés :

```php
// Affiche : Arnold a dit : "I'll be back"
echo 'Arnold a dit : "I\'ll be back"';

// Affiche : Voulez-vous supprimer C:\*.*?
echo 'Voulez-vous supprimer C:\\*.*?';

// Affiche : Voulez-vous supprimer C:\*.*?
echo 'Voulez-vous supprimer C:\*.*?';

// Affiche : Ceci n'affichera pas \n de nouvelle ligne
echo 'Ceci n\'affichera pas \n de nouvelle ligne';

// Affiche : Les variables ne seront pas $traitees $ici
echo 'Les variables ne seront pas $traitees $ici';
```

Déterminer un type de donnée

- **Détermination du type** de donnée
  - gettype()
  - is_int(), is_double(), is_string(), is_array(), is_object(), is_bool()

Exemple

```php
$a = 1;             echo gettype($a);   // Affiche : integer
$b = 1.;            echo gettype($b);   // Affiche : double
$c = NULL;          echo gettype($c);   // Affiche : NULL
$d = new stdClass; echo gettype($d);   // Affiche : object
$e = 'foo';         echo gettype($e);   // Affiche : string

is_int($a);   // Renvoie TRUE
is_int($b);   // Renvoie FALSE
```

Modification de type (1)

Transtypage = conversion de type

Via la fonction *settype()*

```php
$nbr = 10;
echo gettype($nbr);          // Affiche : integer
settype($nbr, "double");
echo gettype($nbr);          // Affiche : double
```

Via un *cast*

```php
$str = "10 maisons";
$nbr = (int)$str;
echo "\$nbr = $nbr ; son type est : ".gettype($nbr);
// Affiche : $nbr = 10 ; son type est : integer
```

Modification de type (2)

Certaines conversions sont automatiques :

```
$foo = "0";                          // $foo est une chaîne de caractères
$foo += 2;                           // $foo est maintenant un entier (2)
$foo = $foo + 1.3;                   // $foo est maintenant un nombre à
                                     // virgule flottante (3.3)
$foo = 5 + "10 Little Piggies"; // $foo est un entier (15)
$foo = 5 + "10 Small Pigs";     // $foo est un entier (15)
```

Constantes

- **Définition**
  - `define("UNIV", "Paris 5");`
  - `define("JAUNE", "#FFFF00");`
  - `define("PI", 3.14);`
- **Utilisation** d'une constante
  - `echo "Contenu de la constante :".UNIV;`
  - Attention: on accède au contenu sans le $

Constantes magiques

Nom	Description
`__LINE__`	La ligne courante dans le fichier.
`__FILE__`	Le chemin complet et le nom du fichier courant avec les liens symboliques résolus. Si utilisé pour une inclusion, le nom du fichier inclus est retourné.
`__DIR__`	Le dossier du fichier. Si utilisé dans une inclusion, le dossier du fichier inclus sera retourné. C'est l'équivalent de *dirname(__FILE__)*. Ce nom de dossier ne contiendra pas de slash final, sauf si c'est le dossier racine.
`__FUNCTION__`	Le nom de la fonction.
`__CLASS__`	Le nom de la classe courante. Le nom de la classe contient l'espace de nom dans lequel cette classe a été déclarée (i.e. *Foo\Bar*). Notez que depuis PHP 5.4, __CLASS__ fonctionne aussi en Traits. Lorsqu'une méthode Trait est utilisée, __CLASS__ est le nom de la classe que Trait utilise en interne.
`__TRAIT__`	Le nom Trait. Le nom Trait inclut l'espace de nom dans lequel il a été déclaré (e.g. *Foo\Bar*).
`__METHOD__`	Le nom de la méthode courante.
`__NAMESPACE__`	Le nom de l'espace de noms courant.

http://php.net/manual/fr/language.constants.predefined.php

Les variables prédéfinies

Variables prédéfinies :
Les variables d'environnement dépendant du client

Variable	Description
$_SERVER["HTTP_HOST"]	Nom d'hôte de la machine du client (associée à l'adresse IP)
$_SERVER["HTTP_REFERER"]	URL de la page qui a appelé le script PHP
$_SERVER["REMOTE_ADDR"]	L'adresse IP du client
$_SERVER["PHP_SELF"]	Nom du script PHP
$_SERVER["SERVER_ADDR"]	Adresse IP du serveur

L'instruction "echo phpinfo();" affiche toutes les variables prédéfinies.

Operateurs arithmétiques

Exemple	Nom	Résultat
-$a	Négation	Opposé de *$a*.
$a + $b	Addition	Somme de *$a* et *$b*.
$a - $b	Soustraction	Différence de *$a* et *$b*.
$a * $b	Multiplication	Produit de *$a* et *$b*.
$a / $b	Division	Quotient de *$a* et *$b*.
$a % $b	Modulo	Reste de *$a* divisé par *$b*.
$a ** $b	Exponentielle	Résultat de l'élévation de *$a* à la puissance *$b*. Introduit en PHP 5.6.

http://php.net/manual/fr/language.operators.arithmetic.php

Operateurs sur les bits

Exemple	Nom	Résultat
`$a & $b`	And (Et)	Les bits positionnés à 1 dans *$a* ET dans *$b* sont positionnés à 1.
`$a | $b`	Or (Ou)	Les bits positionnés à 1 dans *$a* OU *$b* sont positionnés à 1.
`$a ^ $b`	Xor (ou exclusif)	Les bits positionnés à 1 dans *$a* OU dans *$b* mais pas dans les deux sont positionnés à 1.
`~ $a`	Not (Non)	Les bits qui sont positionnés à 1 dans *$a* sont positionnés à 0, et vice-versa.
`$a << $b`	Décalage à gauche	Décale les bits de *$a*, *$b* fois sur la gauche (chaque décalage équivaut à une multiplication par 2).
`$a >> $b`	Décalage à droite	Décalage les bits de *$a*, *$b* fois par la droite (chaque décalage équivaut à une division par 2).

http://php.net/manual/fr/language.operators.bitwise.php

Opérateurs de comparaison

Exemple	Nom	Résultat
$a == $b	Egal	TRUE si *$a* est égal à *$b* après le transtypage.
$a === $b	Identique	TRUE si *$a* est égal à *$b* et qu'ils sont de même type.
$a != $b	Différent	TRUE si *$a* est différent de *$b* après le transtypage.
$a <> $b	Différent	TRUE si *$a* est différent de *$b* après le transtypage.
$a !== $b	Différent	TRUE si *$a* est différent de *$b* ou bien s'ils ne sont pas du même type.
$a < $b	Plus petit que	TRUE si *$a* est strictement plus petit que *$b*.
$a > $b	Plus grand	TRUE si *$a* est strictement plus grand que *$b*.
$a <= $b	Inférieur ou égal	TRUE si *$a* est plus petit ou égal à *$b*.
$a >= $b	Supérieur ou égal	TRUE si *$a* est plus grand ou égal à *$b*.

http://php.net/manual/fr/language.operators.comparison.php

Opérateurs logiques

Exemple	Nom	Résultat
$a and $b	And (Et)	TRUE si *$a* ET *$b* valent TRUE.
$a or $b	Or (Ou)	TRUE si *$a* OU *$b* valent TRUE.
$a xor $b	XOR	TRUE si *$a* OU *$b* est TRUE, mais pas les deux en même temps.
! $a	Not (Non)	TRUE si *$a* n'est pas TRUE.
$a && $b	And (Et)	TRUE si *$a* ET *$b* valent TRUE.
$a || $b	Or (Ou)	TRUE si *$a* OU *$b* est TRUE.

http://php.net/manual/fr/language.operators.logical.php

Opérateur ternaire

- Opérateur ternaire
  (condition) ? instruction si vrai : instruction si faux

Exemple : `abs_x = (x > 0) ? x : -x ;`

Expressions conditionnelles

- **if** statements

```
if (some test)
{
   // liste d'instructions T
}
elseif (some other test)
{
   // Liste d'instructions E
}
else
{
   // autres instructions
}
```

Switch

```
switch (expression) {
      case valeur0:
        action;
        ...
        action ;
        break;
      case valeur1:
        ...
      case valeur2:
        ...
      default:
        action;
        ...
}
```

Exemple :

```
switch ($a) {
  case $b:
    echo "A est égal à B";
    break;
  case ($a>$b):
    echo "A est supérieur à B";
    break;
  default:
    echo "A est inférieur à B";
    break;
}
```

Boucles

- boucle **for**
  - **for** ($i=1; $i<6; $i++) { echo "$i
"; }
- boucle **while**
  - **while** (condition) {bloc d'instructions ;}
- boucle **do...while**
  - **do** {bloc d'instructions ;} **while** (condition) ;
- boucle **foreach**
  - **foreach** ($tableau **as** $valeur) {instructions utilisant $valeur ;}

Déterminer si une variable est définie

isset()

Détermine si une variable est définie et est différente de NULL.
Renvoie FALSE si :

- La variable est de valeur NULL.
- La variable n'est pas définie.
- La variable n'est plus définie (détruite par la fonction *unset()*).

```
if (isset($a))
    echo '$a est définie et n'est pas de type NULL';
else
    echo '$a n'est pas définie ou est de type NULL';
```

Déterminer si une variable est vide

empty()

Détermine si une variable est vide.
Ce qui suit est considéré comme étant vide :

- "" (une chaîne vide)
- 0 (0 en tant qu'entier)
- 0.0 (0 en tant que nombre à virgule flottante)
- "0" (0 en tant que chaîne de caractères)
- NULL
- FALSE
- array() (un tableau vide)
- $var; (une variable déclarée et sans valeur)
- Variable non déclarée

```
if (empty($a))
   echo '$a est vide ou n’est pas définie';
else
   echo "\$a existe et n’est pas vide";
```

Code imbriqué (1)

Exemple 1 :

```
<?php
     if($heure < 18)
     {
?>
     <b> Bonjour !</b>
<?php
     }
     else
     {
?>
     <b> Bonsoir !</b>
<?php
     }
?>
```

Exemple 2 :

```
<tr>
<?php
     for($i=0; i<count($tab); i++)
     {
?>
     <td class='toto'>
<?php
     echo $tab[$i] ;
?>
     </td>
<?php
     }
?>
</tr>
```

ILLISIBLE !!!
TECHNIQUE À ÉVITER !!!

Code imbriqué (2)

Lorsque l'on programme en PHP, il est très tentant de mélanger le code PHP au sein des balises HTML.

Le problème, c'est que le code devient très vite illisible.
Même avec de bons commentaires.

Pour une bonne pratique de programmation,
il faudra respecter les deux règles suivantes :

Pas de code PHP dans un fichier HTML
Que du code PHP dans un fichier PHP

Manipulation de chaînes de caractères

- **strstr**(ch1,ch2)
  → renvoie contenu de ch1 à partir de la première occurrence de ch2
  Si ch2 n'apparaît pas dans ch1, renvoie false
- **substr**(ch,deb,long)
  → renvoie la sous-chaîne de ch de longueur long à partir de deb
- **explode**(sep,ch)
  → découpe la chaîne selon le séparateur
  et renvoie un tableau avec les éléments
  ex : `$tab = explode(":","Rouge:Vert:Bleu");`
- **trim**() → suppression des blancs en début et fin de chaîne
- **ltrim**() → idem mais à gauche seulement
- **rtrim**() et **chop**() → à droite seulement
- **str_replace**(ch1,ch2,ch)
  → remplace dans ch toutes les occurrences de ch1 par ch2

http://php.net/manual/fr/ref.strings.php

Changement de la casse

```php
$s = "allons enfants de la patrie";

strtoupper($s) // ALLONS ENFANTS DE LA PATRIE
strtolower($s) // allons enfant de la patrie
ucfirst($s)    // Allons enfant de la patrie
ucwords($s)    // Allons Enfant De La Patrie
```