Programmation Internet

Cours 6

kn@lri.fr
http://www.lri.fr/~kn

Plan

Avant PHP

Constat: besoin de pages Web dynamiques (contenu généré au chargement de la page)
Première solution: scripts (ou programmes) CGI

```
http://www.example.com/foo.cgi?sort=alpha
```

1. Programme écrit dans n'importe quel langage et exécuté par le serveur Web
2. Le serveur passe les infos au programme par des variables d'environnement et l'entrée standard
3. Le programme génère une page Web par affichage sur la sortie standard

Inconvénients

1. Communication difficile entre le serveur Web et le programme
2. Les langages généralistes ne sont pas faits pour le Web (pas de support d'HTML nativement par exemple)

PHP (Avantages)

- **Langage utilisé côté serveur : le navigateur ne « voit » jamais de PHP, uniquement du HTML (comportement indépendant du navigateur)**
- **Langage interprété : on ne génère pas de binaire, déploiment facile (on copie les fichiers sur le serveur)**
- **Intégration avec HTML : PHP supporte des *templates* (modèles/patrons) HTML contenant du PHP**

Exemple: fichier `heure.php` **:**

```
<html>
  <head><title>Heure</title></head>
  <body>
    <h1>L'heure</h1>
    <p>Il est <?php echo date('H:i:s'); ?> </p>
  </body>
</html>
```

PHP

- On compte 140,000,000 de noms de domaines enregistrés
- On estime à environ 20,000,000 le nombre de sites faits en PHP
- Quelques gros sites:
  - Facebook
  - Portail Yahoo!
  - Flickr
  - Digg
  - Wikipedia

Principe

- Le serveur Web crée un fichier temporaire. Il copie le contenu de heure.php jusqu'à trouver la balise <?php
- Le code se trouvant entre <?php et ?> est exécuté.
- Tout ce que le code écrit (instruction echo) est aussi copié dans le fichier temporaire
- Le serveur web reprend la copie du fichier après ?>
- Le serveur renvoie le contenu du fichier temporaire comme page au client

```
<html>
  <head><title>Heure</title></head>
  <body>
    <h1>L'heure</h1>
    <p>Il est 15:53:00 </p>
  </body>
</html>
```

PHP (inconvénients)

1. Génère du HTML via echo (débuggage difficile)
2. Interprété : problème de performances si beaucoup de clients
3. NON TYPÉ : C'est HORRIBLE

Devinette : qu'affiche l'instruction suivante ? (on ne connaît pas encore PHP mais on peut utiliser son intuition)

```
echo 013 + "013 c'est en fait 11 en base 8";
```

- 013
- 13
- 013 c'est en fait 11 en base 8
- 013013 c'est en fait 11 en base 8
- 13013 c'est en fait 11 en base 8
- 24 ✔✔✔
- une erreur
- autre chose

Plan

Entiers (integer)

Les entiers ont une taille fixe (généralement 32bits) :

Notation décimale :	10, 3444, -25, 42, ...
Notation binaire :	0b10, -0b10001010, ...
Notation octale :	0755, -01234567, ...
Notation hexadécimale :	0x12b, -0xb00b5, 0xd34db33f, ...

Opérateurs arithmétiques :

- : « Moins » unaire
+, -, *, % : addition, soustraction, produit, modulo
/ : Division. Si x et y sont des entiers et que y divise x alors x/y renvoie un **entier** sinon x/y renvoie un **flottant**!

Booléens (`boolean`)

TRUE/FALSE : vrai/faux (en majuscules)

Opérateurs logiques :

! : négation (unaire)

&&, || : « et » logique, « ou » logique

Flottants (`float`)

Notation scientifique : 1.3, 0.99, 00.34e102, -2313.2313E-23, ...

Opérateurs arithmétiques :

- :	« Moins » unaire
+, -, *, /, % :	opérations standard
abs, sin, cos, sqrt, pow, ... :	fonctions mathématiques pré-définies

Variables, affectations

- Un nom de variable commence toujours par $
- Pas besoin de donner le type, l'affectation détermine le type

Exemples :

```
$foo = 123;
$bar = 1323e99;
$_toto = $bar;
```

Chaînes de caractères (string)

Simples quotes : `'foo', 'c\'est moi', 'Un antislash : \\', ...`

Pas d'autre séquence d'échappement

Doubles quotes : `"foo", "c'est moi", "Un retour chariot: \n", "La variable \$toto contient: $toto"`

Les séquences d'échapement sont: \n, \t, \\, \", \$. Les variables (sous-chaînes commençant par un $) sont remplacées par leur valeur.

Opérations sur les chaînes :

$foo[10] :	**accès au 11ème caractère**
$foo[10] = 'A'; :	**mise à jour**
. :	**concaténation**
strlen :	**longueur**

Tableaux (array)

Les tableaux sont des tableaux **associatifs** :

array() :	crée un tableau vide
array(k_1 => v_1, …, k_n => v_n **) :**	créé un nouveau tableau pour lequel l'entrée k_i est associée à la valeur v_i
array(v_1, …, v_n **) :**	créé un nouveau tableau pour lequel l'entier `i-1` valeur v_i

Quelques fonctions :

count :	taille du tableau (nombre d'éléments)
sort, rsort :	trie un tableau (`rsort` trie par ordre décroissant) par valeurs. Les clés sont supprimées et de nouvelles clées de 0 à longueur - 1 sont crées
ksort, krsort :	trie un tableau par clés
print_r :	affiche un tableau (ne pas utiliser `echo`)

Tableaux (exemples)

```php
$tab1 = array(); //tableau vide
$tab2 = array("zero", "un", "deux", "trois");
$tab3 = array("pi" => 3.14159, "e" => 2.71828; "phi" => 1.61803);
echo $tab2[0]; //affiche zero
echo $tab3["phi"]; //affiche 1.61803
$tab1["dix"] = 10; //affectation
sort($tab2);
echo $tab2[0];//affiche deux
sort($tab3);
echo $tab3[0]; //affiche 1.61803
echo $tab3[1]; //affiche 2.71828
echo $tab3[2]; //affiche 3.14159
echo count($tab2); //affiche 4
```

NULL

NULL est une constante spéciale, de type NULL. C'est la valeur d'une variable non déclarée ou d'un accès invalide dans un tableau.

```
$a = $b;  // $b n'est pas déclarée, $a reçoit NULL
$c = $tab["toto"]; // $tab existe mais n'a pas de valeur
                   // associée à la clé "toto", $c reçoit NULL
```

Conversions de types

☠☢☣☹ : les conversions se font **implicitement**, en fonction du contexte.

Booléen : 0, 0.0, "", "0", NULL, et le tableau vide sont convertis en FALSE, le reste en TRUE (en particulier "00" vaut TRUE ☹)

Entier : FALSE ⇝ 0, TRUE ⇝ 1, les flottants sont arrondis par partie entière (1.23242 ⇝ 1). Les chaînes **dont un préfixe** est un entier sont converties en cet entier, sinon en 0 ("123 bonjour" ⇝ 123)

Chaîne : La chaîne contient la représentation de la valeur (1 . "ABC" ⇝ "1ABC"). FALSE et NULL sont convertis en "", TRUE converti en "1"

La réponse à la devinette : 013 + "013 c'est 11 en octal" :

- 013 ⇝ notation octale pour l'entier décimal 11
- "013 c'est 11 en octal" chaîne de caractères utilisées dans une addition (contexte entier), le préfixe "013" est transformé en **décimal** 13
- 11 + 13 ⇝ 24 (ça au moins ça a du sens)

Comparaisons

Opérateurs de comparaisons

Opérateur	Description
`$a == $b`	Égal, après conversion de type
`$a != $b`	Différent, après conversion de type
`$a === $b`	Égal et de même type
`$a !== $b`	Différent ou de type différent
`$a < $b`	Strictement plus petit, après conversion de type
`$a > $b`	Strictement plus grand, après conversion de type
`$a <= $b`	Plus petit, après conversion de type
`$a >= $b`	Plus grand, après conversion de type

Affichage

On utilise l'instruction echo **pour écrire du texte dans la page HTML résultante :**

Code PHP	Code HTML	Affichage dans le navigateur
`echo "Hello";` `echo "World";`	**HelloWorld**	**HelloWorld**
`echo "Hello\n";` `echo "World";`	**Hello** **World**	**Hello World**
`echo "Hello ";` `echo "World";`	**Hello**` `**World**	**Hello** **World**

Plan

Conditionnelle: `if else`

```
if ( c ) {
        // cas then
 } else {
      // cas else
};
```

Les parenthèses autour de la condition `c` sont obligatoires. La branche `else { … }` est optionnelle. Les accolades sont optionnelles pour les blocs d'une seule instruction

Boucles

```
while ( c )  {
   //corps de la boucle while
};

do  {
   //corps de la boucle do
} while ( c );

for(init ; test ; incr)  {
   //corps de la boucle for
};

foreach($tab as $key => $val)  {
   //corps de la boucle foreach
   //$tab est un tableau, $key une clé et $val la valeur associée
};
```

Remarque: `ksort` **et** `krsort` **influencent l'ordre de parcours par une boucle** `foreach`

break et continue

`break :` **sort de la boucle immédiatement**

`continue :` **reprend à l'itération suivante**

```
for($i = 0; $i < 10; $i = $i + 1){
    if ($i == 2 || $i == 4) continue;
    if ($i == 7) break;
    echo $i . ' ';
}
```

Affiche 0 1 3 5 6

Plan

Formulaire HTML (version simple)

L'élément `<form>` permet de créer des formulaires HTML. Un formulaire est constitué d'un ensemble de widgets (zones de saisies de textes, boutons, listes déroulantes, cases à cocher, ...) et d'un bouton submit. Lorsque l'utilisateur appuie sur le bouton, les données du formulaires sont envoyées au serveur. Exemple, fichier `age.html` :

```
<html>
  <body>
    <form method="get" action="calcul.php">
      Entrez votre année de naissance: <input type="text" name="val_age"/>
      <input type="submit" />
    </form>
  </body>
</html>
```

Paramètres

Les paramètres envoyés au serveur web par la méthode `get`, sont accessibles en PHP dans la variable globale `$_GET`. C'est un tableau qui associe au nom d'un input sa valeur. Exemple : `calcul.php`

```
<html>
  <body>
    <?php
     echo "Vous êtes né il y a ";
     echo "<b>";
     echo date("Y") - $_GET["val_age"];
     echo "</b> ans";
     echo "<br/>";
     ?>
    </body>
</html>
```