Technologies du web

4 - PHP (suite)

PHP

Langage de script interprété côté serveur

Couplé au serveur Apache (via mod_php ou PHP-FPM)

Utile pour générer du HTML, traiter des formulaires, accéder aux bases de données

Afficher la date

```
<html>
    <head>
    </head>
    <body>
        <h1>La date du serveur est:</h1>
        <p>
           <?php   echo date("d/m/y")  ?>
        </p>
     </body>
</html>
```

Syntaxe

- Code entre balises **<?php … ?>**
  - Commentaires
  - // sur une ligne
  - /* sur plusieurs lignes */

- Chaque instruction se termine par un **;**

Fonction echo

echo : affiche une **chaîne de caractères** (string)

```
<?php
	echo '<p>ceci est du html</p>';
	echo date("d/m/y");
?>
```

Variables

Son nom commence par un **$**

Ex : $i, $objet, $categories

Sensible à la casse

Ex : $A != $a

Non typées : on peut changer le type de données stockées par une variable

```
$a = 10;
$a = 'pouet';
```

L’assignation

```
$var = 10;

$var2 = $var;    // copie

$var = 20;

echo ($var, $var2);
```

Types

Chaines de caractères (string)

Entiers (integer, int)

Nombre décimaux (float, double)

- Avec un point : 3.14159

Les booléens (boolean, bool)

- TRUE / FALSE

Types

Pas de valeur : **null**

Les tableaux : **array**

Les objets : **object**

…

https://secure.php.net/manual/fr/language.types.php

Manipulations de types

```
$total = (int) $var;    // conversion de type (casting)

$a = '0' + $total;      // conversion automatique
```

(int), (integer) : modification en integer
(bool), (boolean) : modification en boolean
(float), (double), (real) : modification en float
(string) : modification en string
(array) : modification en array
(object) : modification en object
(unset) : modification en NULL

Les tableaux

Un array simple, une liste

$a = [1, 2, 3, 4, 42];

Un tableau associatif, clé-valeur

$a = ['Jimi' => 27, 'Kurt' => 27, 'Amy' => 27, 'Michael' => 50];

Manipulation d’array

```
$tab = [];                  $tab2 = array();            // tableaux vides

// ajout à la fin
$tab[] =  'coucou';

// ajout par index
$tab[0] = 'coucou';         $tab[42] = 'serviette';

// ajout par clé
$tab['kikoo'] =  'coucou';
```

Manipulation d’array

```
// accès aux valeurs
$a = $tab['kikoo'];                                    // a vaut 'coucou'


// suppression d’une valeur
unset($tab['kikoo']);
```

Manipulation de chaines

```
$age = 42;
$login = 'Serge';

// concaténation
$a = 'Joyeux ' . $age . 'ème anniversaire ' . $login . ' !';

// substitution
$a = "Joyeux $login ème anniversaire $login !";

// substitution forcée
$a = "Joyeux {$login}ème anniversaire $login !";
```

Boucles

```
for ($i = 0; $i < 42; $i++) {
  echo $i;
}

// iteration clé-valeur d'un tableau
foreach ($tab as $cle => $val) {
  echo $cle . ":" . $val;
}
```

Boucle while

Boucle conditionnelle

```
$a = 0;
while ($a < 10) {
  echo $a;
  $a++;
}
```

Fonctions utiles

count($tableau); // renvoie le nbre d'elements
strlen($chaine); // renvoie le nbre de caractère dans la chaine
trim($chaine); // Renvoie la chaine sans les espaces de debut et de fin

explode(",", $chaine); // Transforme une chaine en tableau en separant selon le caractère spécifié (split)
implode(",", $tableau); // operation inverse (join)
print_r($tableau); // Affiche le contenu d'un tableau

Fonctions utiles

strtolower($chaine); // renvoie la chaine en minuscule
strtoupper($chaine); // renvoie la chaine en majuscule

substr($chaine, $index, $size); // Renvoie la sous chaine commencant à $index et de taille $size
str_replace($old, $new, $chaine); // Remplace $old par $new dans $chaine
strpos($chaine, $souschaine); // Renvoie l'index de $souschaine dans $chaine

Fonctions utiles

sort($tab) // trie le tableau par ses valeurs
array_pop($tab) // récupère et supprime le dernier élément
array_push($tab, $val) // ajoute l'élément à la fin du tableau
array_shift(tab) // récupère et supprime le premier élément
array_unshift(tab, $val) // ajoute l'élément au début du tableau
array_merge($array1, $array2, ...) // fusionne les x tableaux
in_array($needle,$array) // recherche $needle dans les valeurs de $array
array_key_exists($key, $array) // retourne true si $key est une clé de $array
array_flip($array) // inverse les clés et les valeurs

Définir une fonction

```php
<?php
function calcul($a, $b) { // déclaration de fonction
   $resultat = $a + $b ;

   // termine la fonction avec une valeur de retour
  return $resultat;
}
// appel de la fonction
echo("J'ai calculé " . calcul(2, 3) . " et " . calcul(2,2));
```

Fonctions - à savoir

Portée des variables : les variables dans la fonction sont isolées de celles en dehors. Seules les variables passées en paramètres sont accessibles.

On peut définir une valeur par défaut pour les variables passées en paramètres.

```
function foo($a, $b, $c = 10) {
    echo $a + $b + $c;
}

foo(1, 2, 3);    // affiche 6
foo(1, 2);       // affiche 13
```

Import de scripts PHP

On peut importer le contenu d'un script PHP dans un autre, notamment pour réutiliser facilement du code.

include(*chemin*); // insère le contenu du fichier php donné en paramètre

require(*chemin*); // pareil, mais lance une exception si le fichier n'est pas trouvé

lib.php

```
<?php
  // définition d’une variable avec du texte
  $str = "<p>ceci est du html</p>";

  function ma_fonction($a, $b) {
    echo  "bonjour " . $a . " et " . $b;
  }
```

main.php

```
<?php
  include('lib.php');  // inclusion d’un autre fichier PHP

  // $str et ma_fonction sont définies dans lib.php
  echo $str;

  $b = ma_function('jean', 'jacques');

  echo '<p>' . $b . '</p>';
```

Gérer des données

Le script PHP est évalué à chaque requête, le contenu des variables est réinitialisé entre chaque appel.

Comment conserver des données ?

- **Session** : données isolées, seulement disponibles pour l'utilisateur (ex: panier d'une boutique, filtres de recherche, etc.)
- **Fichier / Base de données** : données potentiellement partageables, stockées à long terme (articles, commentaires, notes et avis, etc.)

Sessions

PHP permet de définir et de gérer un ensemble de **données stockées durant toute la durée de la session**, c'est à dire accessibles après la fin de l'exécution du script initial.

- Création d'un identifiant unique de session par PHP
- Stockage de l'identifiant dans un cookie coté client
- Stockage des données de session sur le serveur

À chaque requête le cookie est envoyé et PHP peut ainsi retrouver les données de la session

session_start();

Ajouter des données en session

```php
<?php
session_start();

$_SESSION['login'] = 'JohnW';
// session_register(login, 'JohnW');    // alternative


print_r($_SESSION);
```

Vérifier les données en session

```
<?php
session_start();

if (isset($_SESSION['pagesVues'])) {
// if (session_is_registered('pagesVues')) {   // alternative
   $_SESSION['pagesVues']++;
} else {
   $_SESSION['pagesVues'] = 1;
}

echo 'Pages vues : ' . $_SESSION['pagesVues'];
```

Détruire les données en session

```php
<?php

session_start();

unset($_SESSION['pagesVues']);
```

Fichiers

```
// lecture
if (file_exists('fichier.txt')) {
    $contenu = file_get_contents('fichier.txt');
}

// écriture
$texte = 'du texte';
file_put_contents('autre-fichier.txt', $texte);
```

Limites du stockage fichiers

Problèmes de structuration de la donnée, pas de modèle relationnel

Les gros volumes de données chargés en mémoire

Persistance de données sérialisées

```
$fichier = "/home/user/data.db";

// écriture de données sérialisées
$data = [...];
file_put_contents($fichier, serialize($data));



// lecture
$data = unserialize(file_get_contents($fichier));
```

Problèmes et alternatives

Données conçues pour être seulement utilisées avec des scripts PHP

Difficile de modifier les données et leur structure

On peut en revanche utiliser un des standards conçus exprès pour structurer des données :

JSON
XML

JSON

Javascript Object Notation

Tableau associatif avec un syntaxe provenant du javascript, et parfaitement compatible

Types de données possibles :

- chaînes de caractères
- nombres
- listes
- sous tableaux associatifs (tableaux imbriqués)
- ...

JSON Exemple

```
{
  "nom" : "perret",
  "prenom" : "pierre",
  "age" : 1000,
  "telephones" : ["0648151623", "0836656565"],
  "adresse" : {
    "rue" : "rue de la gerbe",
    "code" : 69002,
    "ville" : "lyon"
  }
}
```

Lire du JSON

```php
<?php
$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}'; // string JSON
// lecture depuis un fichier
// $json = file_get_contents("mon_fichier.json");

$var = json_decode($json, true);

// var contiendra un tableau associatif, ou false si le JSON
est mal formé
```

Écrire du JSON

```php
$ages = [
    'Thierry' => 42,
    'Bobby'   => 23,
];

$json = json_encode($ages);

file_put_contents("mon_fichier.json", $json);

// contiendra la chaîne :
// { "Thierry": "42", "Bobby": "23" }
```

Pourquoi JSON

Syntaxe simple, proche de la syntaxe des objets en JS
Facile à analyser, lire, écrire, en PHP
Et pas seulement PHP, tout le monde en fait
Pratique pour transmettre de la donnée structure via HTTP (web services, AJAX)

XML

Structuré par des balises

Les balises sont libres

Difficile à analyser, lire et écrire

XML Exemple

```
<observations>
     <observation>
          <auteur>Jean Paul</auteur>
          <date>18-07-2012</date>
          <taxon>ophrys apifera</taxon>
     </observation>
     <observation>
          <auteur>Jacque Remi</auteur>
          <date>19-07-2012</date>
          <taxon>ophrys fuciflora</taxon>
     </observation>
</observations>
```

Lire un XML

```
$dom = new DomDocument();
$dom->loadXML($chaineXML);


$listeObs = $dom->getElementsByTagName('observation');


foreach($listeObs as $obs) {
    echo $obs->firstChild->nodeValue . "<br />";
}
```

Pour les motivé•e•s https://secure.php.net/manual/fr/refs.xml.php

Base de données

Pour stocker de larges volumes de données

Des langages permettent d’accéder à une portion de données = requêtes

Ça implique d’utiliser un moteur de base de données spécifique

Types de moteurs

- Type SQL (relationnel)
  - MySQL
  - SQLite
  - PostgreSQL
  - Oracle

- Type NoSQL (clé-valeur, documents, graph, …)
  - Redis
  - MongoDB
  - Cassandra
  - Neo4j

Base de données relationnelle

Gère des tables

Les champs sont représentés sous forme de colonnes

Chaque ligne est une entrée différente

Les relations