Introduction à PHP : sites et webservices

Guillaume Allègre

Polytech Grenoble

2016

Le langage PHP

Rendre dynamique le HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Horloge</title>
</head>

<body>
<p>
Nous sommes le <?php
setlocale(LC_TIME, 'fr_FR');
echo strftime("%A %e %B %Y") ?>.<br />
Il est <?php  echo strftime("%H h %M") ?>.
</p>
</body>
</html>
```

PHP : qu'est-ce ?

Les raisons du succès

- langage spécialisé : embarqué web
- simple (mais inspiré de C et de Perl)
- accessible aux débutants

La maturité, enfin ?

- En évolution constante depuis sa création
  1994 : PHP 3 (première version publique)
  2000 : PHP 4 (ajout de la programmation objet)
  2004 : PHP 5 (programmation objet refaite !)
  2015 : PHP 7.0
- Programmation objet possible : depuis PHP 4 et surtout PHP 5
- Bibliothèques libres : email, images, PDF, base de données...

PHP : installation

Logiciels nécessaires

- Serveur web (apache)
- Extension PHP pour le serveur web
  - `mod_php` pour Apache
  - ou `php-fpm` (Apache, Nginx...)
- interpréteur PHP en ligne de commande (php-cli)

Options

- Base de données (MySQL...)
  - mysql-server
  - compléments optionnels : PHPMyAdmin...

PHP : installation - en pratique

Sous Linux (Debian/Ubuntu)

Installer les paquets nécessaires.

```
aptitude install apache2 libapache2-mod-php php-cli
aptitude install mysql-server php-mysql
```

Sous Windows (déconseillé)

Choisir un installateur complet : **WampServer**, EasyPHP, XAMPP...

Environnement de TP

PC UFR ou Polytech

- accès ssh `goedel.e.ujf-grenoble.fr`
- `http://im2ag-wiki.ujf-grenoble.fr/doku.php?id=environnements:php`

Machines personnelles - sous Linux

- installation préalable des paquets

Machines personnelles connectés à goedel (non recommandé)

- environnement VPN - Cisco AnyConnect
- `http://im2ag-wiki.ujf-grenoble.fr/doku.php?id=infra:reseau`
- recommandé `apt install openconnect`

Exemple, côté serveur

```
<html>
<head>
    <title>Exemple</title>
</head>
<body>
    <!-- un commentaire HTML -->
    <p>
        <?php echo "1789*19.6/100=" . (1789*19.6/100); ?>
    </p>
</body>
</html>
```

exemple.php, sur le serveur

Exemple, côté client (réponse à une requête HTTP)

```
<html>
<head>
    <title>Exemple</title>
</head>
<body>
    <!-- un commentaire HTML -->
    <p>
        1789*19.6/100 = 350.644
    </p>
</body>
</html>
```

exemple.php, dans le navigateur

PHP à grands traits

```php
<?php
if (2 > 1) {
    /* Le test ci-dessus devrait nous amener ici */
    echo "Tout va bien !";
} // fin du bloc du if
?>
```

PHP à grands traits

```php
<?php
if (2 > 1) {
    /* Le test ci-dessus devrait nous amener ici */
    echo "Tout va bien !";
} // fin du bloc du if
?>
```

- Le code PHP est toujours entre **<?php** et **?>**
  Éviter les "short tags" : <? ...?>
- Généralement sensible à la casse : $a ≠ $A
- instructions terminées par **;**
- blocs délimités par **{ ... }**
- commentaires avec **//** en fin de ligne ou **/* ... */**
- toujours se référer à la documentation officielle :
  http://fr.php.net/manual/fr/

Première expérimentation

Le but est de réaliser une multiplication par un formulaire web.

1. Créer une page avec un formulaire permettant de saisir 2 nombres. Les noms des champs du formulaire seront "a" et "b".
2. Cette page est-elle en HTML ou en PHP ? Pourquoi ?
3. Ce formulaire doit être envoyé sur une page `res.php`.
   Y afficher le contenu de `$_GET['a']` et `$_GET['b']`.
4. Sur la page `res.php`, afficher l'opération avec son résultat, par exemple : $13x14 = 182$.

Les environnements d'exécution

PHP pour le web

- couplé à un serveur web (Apache, Nginx...)
- serveur intégré `php --server` : prototypage

PHP en ligne de commande (php-cli)

- pour l'administration (installation, mises à jour)
  - plus de contraintes réseau (timeout...)
  - droits simplifiés
- pour le développement (prototypage, tests formels)

Code hybride

- bibliothèques, classes communes
- interface web "grand public"
- interface CLI administration
- possible dans tous les frameworks PHP

Premiers pas en PHP

Les données

Variables

Préfixées par **$**

```
$nombre = 174;
```

Déclaration

La déclaration n'est pas obligatoire, mais recommandée.

Une variable est créée à sa première utilisation. Par défaut, elle vaut toujours `NULL`.

Certaines variables spéciales sont créées par PHP avant l'exécution du script : `$_GET`, par exemple.

Fonctions utiles

- `isset()` Teste l'existence
- `unset()` Supprime une variable
- `empty()` Teste si une variable est vide (inexistante, 0, "", etc.)

Types de données

Langage *faiblement typé* (types non déclarés).
Conversion automatique des scalaires : `echo 5` $\Longleftrightarrow$ `echo "5"`

PHP connaît en interne différents types de données :

- scalaire entier : `-157`
- scalaire flottant : `15.7`
- scalaire chaîne : `"Ceci est un texte"`
- scalaire booléen : `TRUE, FALSE`
- tableaux
- objet

Types de données

Langage *faiblement typé* (types non déclarés).
Conversion automatique des scalaires : `echo 5` $\Longleftrightarrow$ `echo "5"`

PHP connaît en interne différents types de données :

- scalaire entier : `-157`
- scalaire flottant : `15.7`
- scalaire chaîne : `"Ceci est un texte"`
- scalaire booléen : `TRUE, FALSE`
- tableaux
- objet

Fonctions utiles

- `var_dump($var)` : détaille le contenu de `$var`
- `gettype($var)`
- `is_integer($var)`, `is_string($var)`, etc.

Les scalaires : les nombres

Opérateurs numériques

Identiques à ceux du C.

Affectation : `=`
Calcul : `+` `-` `*` `/`
Combiné : `+=` `-=` `*=` `/=`
Modulo : `%`
Comparaison : `==` `<` `<=` `>` `>=` `!=`
Incrémentation/décrémentation : `++` `--`

Exemple :

```
$a = 1;
$a++;
$a += 4;
$a *= 2; // au final : 12
```

Les scalaires : les chaînes de caractères

Interpolation

- Sans interpolation

```
$t = 'Salut!';                  Salut!
$t = 'Salut\n a \\toi';         Salut\n␣a␣\toi
$t = 'j\' arrive ';             j'arrive
```

Tous les caractères sont conservés, sauf `\\` et `\'`.

Les scalaires : les chaînes de caractères

Interpolation

- Sans interpolation

```
$t = 'Salut!';                  Salut!
$t = 'Salut\n a \\toi';         Salut\n␣a␣\toi
$t = 'j\' arrive ';             j'arrive
```

  Tous les caractères sont conservés, sauf \\ et \'.

- Avec interpolation

```
$t = "Salut\n a \\toi";         Salut
                                ␣a␣\toi

$a = 'PHP';
$b = "Le site de $a";           Le␣site␣de␣PHP

$j = 40;
$t = "Les $j voleurs";          Les␣40␣voleurs
```

Les scalaires : les chaînes de caractères

Affichage

`echo` / `print` / `printf()`

Exemple : `echo '<p>La variable "$a"</p>';`

Opérateurs

Concaténation : `.`

Comparaison : `==` `<` `<=` `>` `>=` `!=`

Attention :
les opérateurs peuvent induire des conversions implicites.

Les scalaires : les chaînes de caractères

Affichage

`echo` / `print` / `printf()`

Exemple : `echo '<p>La variable "$a"</p>';`

Opérateurs

Concaténation : `.`

Comparaison : `==` `<` `<=` `>` `>=` `!=`

Attention :

les opérateurs peuvent induire des conversions implicites.

Les comparaisons suivantes valent-elles TRUE ou FALSE ?

- "précédent" $<$ "suivant"
- 8 $<$ "13"
- "21" $<$ "9"
- 9 $<$ "dix-huit"

Les scalaires : les chaînes de caractères

Affichage

`echo` / `print` / `printf()`

Exemple : `echo '<p>La variable "$a"</p>';`

Opérateurs

Concaténation : `.`

Comparaison : `==` `<` `<=` `>` `>=` `!=`

Attention :

les opérateurs peuvent induire des conversions implicites.

Si on n'est pas sûr d'avoir une chaîne non numérique, utiliser `strcmp()` au lieu de `<`.

En résumé : variables

- Une variable peut contenir des nombres...
  `$a = 15; $prix = 2.50;`
- ou du texte...
  `$txt = 'La monnaie est le $';`
  Interpolation : `$msgPrix = "Vendu pour $prix euros";`
- ou un booléen...
  `$vrai = true;`
- ou un tableau, un objet... (à suivre)
- Comparaisons et opérations
  `if ($a < 2) { $pluriel = true; }`

Boucles

PHP utilise la syntaxe du C :

```
// boucle pour $i allant de 0 a 99
for ($i=0 ; $i<100 ; $i++) {
    ...
}
// en PHP, on utilise rarement "for"


// on boucle tant que $i!=0
while ($i != 0) {
    ...
}
```

Compléments

- `continue` saute une itération
- `break` termine la boucle

Tests - 1/2

PHP utilise la syntaxe du C :

```
// le plus simple des tests
if ($a > 10) {
    ...
} else {
    ...
}


// operateur ternaire (if sous forme de fonction)
$res = ($a > 0 ? 'positif' : 'negatif');

$res2 = ($a > 0 ? 'positif' : ($a < 0 ? 'negatif' : 'nul') );
```

Tests - 2/2

PHP utilise la syntaxe du C :

```
// idem serie de "if/else if/else"
switch ($var) {
    case 'a': ...
        break;
    default: ...
        break;
}
```

Exercices

1. Créer une page PHP affichant “Hello world !” en utilisant 2 variables PHP. Proposer des variantes.
2. Qu'affichent les instructions suivantes ?

```php
$a=1;
$b="2+$a";
$a=2;
echo $b . $a;
```

3. Que se passe-t-il quand on incrémente une variable inexistante ? Quand on l'affiche ? Expérimenter.
   Moralité : mieux vaut donner une valeur initiale à ses variables.
4. Calculer et afficher les tables de multiplication dans un tableau HTML.
   Commencer avec une ligne statique, puis la faire varier.

Les tableaux

Pourquoi utiliser des tableaux ?

```
$val1 = "Moi";
$val2 = "Toi";
$val3 = "Lui";
echo $val1;
```

Quand on a plusieurs valeurs reliées, mieux vaux les mettre dans un tableau.

```
$val = array("Moi", "Toi", "Lui");
echo $val[0];
```

Les tableaux numériques

- **array** permet de créer un tableau PHP.
- Par défaut, les tableaux sont numérotés à partir de 0.
- Pour accèder à un élément : `$val[...]`
- Pour ajouter à la fin du tableau :

Les tableaux en PHP

- plus précisément : **tableaux associatifs ordonnés**
- paires **clé / valeur** (dictionnaire) :

```php
$tab = array(
  "yes" => "oui",
  "no"  => "non",
);
```

- **accès aux valeurs** :

```php
// Lecture
echo "Traduction de yes : " . $tab["yes"];
// Ecriture
$tab["never"] = "jamais";
```

- multitude de fonctions pour les tableaux :
  `http://fr.php.net/manual/fr/function.array.php`

Parcours d'un tableau

Pour ne lire que le contenu des cases

```php
<?php
foreach ($tableau as $valeur) {
    echo "$valeur\n";
}
```

Pour lire à la fois la clé de la case et sa valeur

```php
<?php
foreach ($tableau as $cle => $valeur) {
    echo "$cle -> $valeur\n";
}
```

***foreach* est la boucle la plus fréquente en PHP !**

Exercices

1. Peut-on interpoler un tableau dans une chaîne, c'est-à-dire écrire `echo "$tab[1]"` ou `echo "$tab['clef']"` ?
2. Que donne `echo $tab` et `echo "$tab"` ? Afficher en HTML tout le contenu d'un tableau PHP.
3. Expérimenter le code suivant. Qu'en déduire ?

```
$tab = array( 1, 2, 3 );
foreach ($tab as $ele) { $ele++; }
```

4. Placer des images dans un répertoire `"pics"` du serveur PHP, puis utiliser la fonction `glob('pics/*')` pour les afficher toutes dans une page HTML.
5. Écrire un tableau PHP de liste de titres de disques. L'afficher sous forme de liste HTML.
6. Transformer le tableau ci-dessus en liste de titres avec leur auteur/compositeur. L'afficher sous forme de <TABLE>.
7. Insérer un élément en queue de liste. Au début de la liste.
8. Supprimer le premier élément de la liste. Le dernier.

Débogage basique

- `print_r($variable)` : afficher le contenu de $variable
- `var_dump($variable)` : idem, plus détaillé
- `die("message")` : arrête l'exécution du script en affichant ce message
- `error_log("message")` : insère un message dans le log du serveur http

Activer les messages d'erreur

Dans le fichier `php.ini`

- `error_reporting = E_ALL`
- `display_errors = On`

Sinon, au début de chaque fichier pendant le développement

```
error_reporting(E_ALL);
```

Formulaires HTML et données HTTP

Éléments des formulaires HTML

- Ligne de saisie de texte

```
<input type="text" name="x" value="" />
```

- Zone de saisie multi-lignes

```
<textarea rows="5" cols="80" name="x"></textarea>
```

- Liste déroulante

```
<select name="x">
 <option value="id1">A</option><option>B</option>
</select>
```

- Liste déroulée

```
<select name="x" size="5" multiple="multiple">...
```

- Boutton poussoir envoyant le formulaire

```
<input type="submit" name="x" value="Envoyer" />
```

ou

```
<button type="submit" name="x" >Envoyer</button>
```

http://www.w3.org/TR/html401/interact/forms.html

Formulaires HTML

```
<form action="http://localhost/formulaire.php"
  method="get">
<p>
    <input type="text" name="login" value="" />
    <button type="submit">OK</button>
</p>
</form>
```

Poster ce formulaire (rempli) se traduira par un accès à l'URL :
`http://localhost/formulaire.php?login=Albert`

Comment accéder à ces informations en PHP ?

Formulaires HTML

```
<form action="http://localhost/formulaire.php"
  method="get">
<p>
    <input type="text" name="login" value="" />
    <button type="submit">OK</button>
</p>
</form>
```

Poster ce formulaire (rempli) se traduira par un accès à l'URL :
`http://localhost/formulaire.php?login=Albert`

Comment accéder à ces informations en PHP ?

`$_GET` : tableau associatif des paramètres GET
Pour ce formulaire : `$_GET['login']`

Formulaires HTML et variables PHP superglobales

Variables superglobales (toujours accessibles)

- créées en interne par PHP (et non par le code utilisateur)
- débutant par `$_`

3 superglobales utiles pour les formulaires :

- `$_GET` : tableau des paramètres GET (reçus par l'URL)
- `$_POST` : tableau des paramètres POST (reçus par le champ POST de l'en-tête HTTP)
- `$_REQUEST` : fusion de ces 2 tableaux

Donc `$_REQUEST` convient que le formulaire utilise *method='GET'* ou *method='POST'*.

Exercices

1. Écrire une page d'authentification avec mot de passe caché et fixé dans le code.
2. Écrire un formulaire de saisie de disque. L'incorporer au projet précédent pour pouvoir ajouter un élément à la liste.
3. Valider le formulaire ci-dessus avant d'ajouter le disque, en demandant de compléter toute case vide.
4. Ajouter un choix de genre musical unique.
5. Comment faire si le disque appartient à plusieurs genres ? Quelle syntaxe HTML et PHP adopter ?
6. Utiliser la superglobale `$_SERVER` pour contrôler si le client est dans une liste d'IP valides.

Fonctions

Fonctions

Déclaration

```php
function add( $param1, $param2 ) {
    // ...
    return $param1 + $param2;
}
```

Utilisation (appel)

```php
echo add(88,-14) . add(3,4);
```

En détails

- Renvoi (facultatif) d'une unique valeur (scalaire, tableau, etc.)
- Paramètres
  - en nombre fixe
  - sauf en cas de **valeur par défaut** avec la déclaration `function add($p1, $p2=1) { ...`

Portée des variables

Les variables ont une zone d'action, dite **portée**.

```
$glob = "glob";
if (true) { $truc = "truc"; }
function test() {
    global $glob;
    $loc++;
    echo "$glob $truc  $loc\n";
}
test();
echo "$glob $truc  $loc\n";
test();
```

Affichera ?

Portée des variables

```
$glob = "glob";
if (true) { $truc = "truc"; }
function test() {
    global $glob;
    $loc++;
    echo "$glob $truc  $loc\n";
}
test();
echo "$glob $truc  $loc\n";
test();
```

Affiche

```
glob  1
glob truc
glob  1
```

Portée des variables

```php
$glob = "glob";
if (true) { $truc = "truc"; }
function test() {
    global $glob;
    $loc++;
    echo "$glob $truc  $loc\n";
}
test();
echo "$glob $truc  $loc\n";
test();
```

- Dans une fonction, 3 types de variables utilisateur :
  - locale (la variable est créée puis effacée en fin de fonction)
  - statique si elle est déclarée avec `static $var;`
  - globale si elle est déclarée avec `global $var;`
- Hors d'une fonction :
  - toute variable est globale
    $\Longrightarrow$ détruite si `unset($var)` ou en fin d'exécution du script

Fonctions utiles pour du texte

Fonctions très nombreuses :
http://fr.php.net/manual/fr/ref.strings.php

Une sélection

- `htmlspecialchars` / `htmlentities` : passe du texte brut au HTML en remplaçant certains caractères par leur entité HTML
- `implode(" ",$tab)` : fusionne un tableau en une chaîne
- `explode(" ",$tab)` : découpe une chaîne en un tableau
- `trim($texte)` : supprime les espaces de début et fin
- `strtolower` / `strtoupper` : change la casse
- `strlen` : donne la taille d'une chaîne
- `strpos` : trouve la position d'un caractère dans une chaîne
- `substr` : retourne un segment de chaîne

Inclusion de fichiers

Quand le code grossit, il faut le répartir dans plusieurs fichiers : `include("fichier.php")` : le contenu de ce fichier est inséré ici.

Variantes de `include` :

- `include_once` : le fichier est inclus que s'il ne l'a pas encore été (utile pour les bibliothèques de code)
- `require` : en cas d'absence du fichier, arrêter tout sur une erreur
- `require_once` : fusion des 2 précédents, le plus utilisé et le plus pratique

A retenir

`require_once` pour charger les fichiers contenant des déclarations de fonctions.

En résumé

- Il faut utiliser des fonctions :
  - code plus concis et plus lisible
  - moins de copier-coller $\Longrightarrow$ code maintenable
  - moins de risques de conflit de variables
- Les variables des fonctions n'existent que dans la fonction (variables **locales**).
- Pour accéder aux variables extérieures, utiliser `global`.
- Placer les déclarations de fonctions dans des fichiers séparés.
- Charger les fichiers de déclaration avec `require_once`.
  Syntaxe fréquente :
  ```
  require_once dirname(__FILE__) . '/lib.php';
  ```
- Chercher les fonctions dans la documentation officielle.

Exercices

1. Écrire une fonction renvoyant la somme des éléments du tableau reçu en argument. Et si on ne reçoit pas de paramètre ? ou pas un tableau ?
2. Avec la fonction PHP `date()`, écrire une fonction renvoyant la date courante en anglais. Passer au français. L'adapter pour qu'on puisse optionnellement décaler la date de X jours.
3. Reprendre le code qui affiche un tableau HTML de disques et en faire une fonction à un paramètre de type tableau PHP.
4. Écrire une fonction qui valide une adresse IP :
   `check_ip('88.77.66.1','127.0.0.1 192.168.');`
5. Écrire une fonction qui valide une adresse IP par rapport à une plage CIDR :
   `ip_cidr('88.77.66.1','192.168.1.1/24');`

Licence