Haute École Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

2020 - 2021
WEBR4-DIN

Développement Internet

Langage PHP

PHP est un langage de programmation par scripts principalement utilisé pour produire des pages Web dynamiques.

1 Présentation du langage

PHP, à l'origine "Personal Home Page" mais depuis renommé en "PHP : Hypertext Preprocessor" [1] est un langage de scripts. Il est multi-usage, mais une de ses utilités principale est la possibilité d'intégrer des scripts PHP au sein de templates HTML afin de produire facilement des pages Web de façon dynamique.

Contrairement à JavaScript, où les pages Web sont **modifiées** de façon dynamique du côté du client (c'est-à-dire du navigateur Internet de l'utilisateur), PHP permet de **générer** des pages Web du côté du serveur. Le script PHP sera exécute au niveau du serveur lors d'une requête HTTP, et c'est la page créée à l'issue de cette exécution qui est envoyée au client en réponse à sa requête.

2 Template HTML et PHP

Un fichier PHP est un fichier texte, qui peut contenir de l'hypertexte de façon similaire à un fichier HTML.

Cependant, en outre, il peut contenir du code PHP, place au sein de balises `<?php` (ouvrante) et `?>` (fermante). Lorsque le serveur charge la page, le code situé entre ces balises est exécuté, et ses valeurs de retours (s'il y en a) sont insérées dans la page à l'endroit où la balise PHP était placée.

Prenons par exemple le template suivant :

```
<html>
<head> <title>test</title></head>
<body>
<?php echo "Bonjour le monde ! <br/> "; ?> </body>
</html>
```

Une fois que le serveur charge cette page, le code PHP est exécute. L'instruction `echo` indique que le texte qui suit est inséré dans la page, et il s'agit de la seule

1. https://www.php.net/manual/en/preface.php

Version du document : 24 mars 2021

instruction ici. Par conséquent, une fois le chargement terminé, le fichier HTML suivant est généré :

```
<html>
    <head>
        <title>test</title>
    </head>
    <body>
        Bonjour le monde ! <br />
    </body>
</html>
```

Notez qu'il est aussi possible d'écrire un fichier PHP qui ne contient que du code PHP, sans HTML autour. Dans ces cas-là, il faut quand même ouvrir la balise `<?php`, mais on peut omettre la balise fermante `?>`.

3 Éléments de syntaxe PHP

La syntaxe de PHP est inspirée de langages tels que C, C++ et Java ; vous pourrez sans doute la prendre en main sans trop de difficulté. Nous verrons ici les particularités du langage les plus utiles, mais nous vous encourageons comme toujours de consulter la documentation du langage pour plus d'information : https://www.php.net/docs.php

Notons pour commencer qu'à l'instar de Java, toute instruction en PHP se termine par un point-virgule `;`

3.1 Variables

PHP est un langage non typé, à l'instar de JavaScript. Cela signifie que le type des variables n'est pas déclaré lorsqu'une variable est créé. En outre, PHP offre certaines fonctionnalités par rapport aux variables qui permettront de rendre la génération de pages dynamiques particulièrement pratique.

PHP offre trois niveaux de visibilité pour ses variables :

- Les variables **locales** sont visibles uniquement dans le bloc de code où elles sont déclarées. Le nom d'une variable locale commence par le symbole `$`, par exemple `$var`.
- Les variables **globales** sont visibles partout dans le code. Celles-ci peuvent être définies de deux façons : soit avec le mot-clé `global`, par exemple `global $var`, soit en l'ajoutant au tableau `$GLOBALS` via `$GLOBALS['var']`.
- Les variables **superglobales** sont des variables visibles partout dans le code, déjà prédéfinies par PHP, avec des rôles bien définis. Il s'agit également de tableaux, qui pourront recevoir des champs nommés. On notera en particulier :
  - La variable `$_GET`, qui contient les données passées au serveur durant une requête HTTP GET.
  - La variable `$_POST`, qui contient les données passées au serveur durant une requête HTTP POST.

— La variable de fichiers `$_FILES`, qui contient les fichiers passés par une requête HTTP POST.

— Les cookies `$_COOKIE` qui permet d'accéder aux cookies HTTP.

— La variable de requête `$_REQUEST`, qui combine par défaut `$_GET`, `$_POST` et `$_COOKIE`

— La variable de session `$_SESSION` qui persistent durant une session utilisateur donnée.

— La variable d'environnement `$_ENV` contient les informations de l'environnement dans lequel le programme s'exécute.

Notons aussi que `$GLOBALS` est en fait une variable superglobale qui reprend les variables globales définies dans le programme.

Les variables superglobales, en particuliers, sont celles qui vont nous permettre de créer des pages. Par exemple, lors de l'envoi d'un formulaire HTML :

```
<form action="login.php" method="POST">
    <input type="text" name="login" placeholder="Login" />
    <input type="submit" value="Se connecter" />
</form>
```

L'action de la balise `<form>` dicte l'URL qui est demandée au serveur. Dans ce cas-ci, il s'agit d'un fichier PHP ; le serveur va alors exécuter le script contenu dans ce fichier :

```
<html>
<head> <title>Login</title></head>
<body>
<?php
    $login = $_POST['login'];
    echo "Bienvenue $login";
?>
</body>
</html>
```

La variable superglobale `$_POST` récupère automatiquement les informations passées par le formulaire.

3.1.1 Contrôle des variables

Il est possible d'effectuer des vérifications sur les variables :

— Vérification de l'existence : via la fonction `isset()`

— Vérification de la vacuité (si la variable est vide) : via la fonction `empty()`

  — Une variable peut **exister** mais être **vide** : par exemple un champ laissé vide dans un formulaire !

— Destruction manuelle de variable : via la méthode `unset()`

3.1.2 Opérateur $ et variables dynamiques

En-dehors de la déclaration d'une variable, $ est un opérateur qui permet de retourner la valeur contenue dans la variable dont le nom est renseigné juste après le symbole $. Par exemple, dans `login.php` ci-dessus, dans l'expression `"Bienvenue $login";`, l'opérateur `$login` est évalué en premier, et la valeur contenue dans cette variable est concaténée à la chaîne adjacente.

Un usage particulier de cette capacité en PHP est l'emploi de **variables dynamiques** : si on souhaite à accéder à une variable dont le nom est lui-même contenu dans une autre variable, on pourra se servir de cet opérateur pour procéder rapidement :

```php
<?php
// appel d'une variable dynamique // affectation
$cours = "AIN";
$varx= "cours" ;
// appel de la variable dynamique
echo $$varx; # équivaut à echo $cours ;
?>
```

3.2 Constantes

La définition d'une constante se fait via la fonction `define(name, value)`. Par exemple :

```php
<?php
define ("WEB", "http://www.heb.be/esi" );
echo "Site internet : ".WEB. "<br>";
```

Notez que l'accès à la valeur de la constante se fait en notant son nom, **sans** le précéder par l'opérateur $

PHP possède en outre quelques constantes prédéfinies :

- `__FILE__` : nom du fichier actuellement exécuté
- `__LINE__` : n° de la ligne actuellement exécutée
- `PHP_VERSION` : chaîne de caractères donnant la version de PHP utilisée
- `PHP_OS` : om du système d'exploitation

3.3 Opérateurs

Les opérateurs de PHP sont, pour la plupart, similaires à ceux que vous aurez rencontrés dans d'autres langages comme Java. On notera par exemple :

- Les opérateurs arithmétiques (`+`, `-`, `*`, `/`, `%`)
- Les opérateurs d'incrémentation et de décrémentation (`++` et `--`)
- L'opérateur d'assignation (`$a=42;`)
- Les opérateurs de comparaison (`==`, `!=`, `>`, `>=`, `<`, `<=`)
- Les opérateurs logiques (ET : `&&` ou **`and`**, OU : `||` ou **`or`**). Notez que la version texte de ces opérateurs a une priorité moindre.
- L'opérateur de concaténation de chaînes, **attention**, est différent de Java : il s'écrit avec `.` Par exemple : `$a . $b` concatène les chaînes de caractères contenues dans les variables `$a` et `$b`.
- L'accès à un caractère dans une chaîne de caractère se fait via les crochets `[]`, par exemple `$str[0]` renvoie le premier caractère de la chaîne contenue dans `$str`

L'instruction **`echo`**, déjà mentionnée, place ce qui suit (typiquement une chaîne de caractère) dans la page HTML générée à la fin de l'exécution du script. La fonction **`print()`** existe aussi, et effectue la même opération, mais renvoie également une valeur de retour (1 en cas de réussite), et peut donc être appelée au sein d'une expression.

3.4 Tableaux

Les tableaux en PHP combinent deux types de tableaux :

- Les tableaux **scalaires**, où les éléments sont associés à des indices (nombres entiers)
- Les tableaux **associatifs**, où les éléments sont associés à des clés (typiquement des chaînes de caractères)

Ces deux types peuvent être combinés au sein d'un même tableau. On notera en particulier que les variables superglobales sont toutes des tableaux.

```php
<?php
// tableau scalaire à 1 dimension (vecteur)
$heb= array("esi","isti","defré");
// tableau scalaire à 2 dimensions (matrice)
$mat[0][1]= 16 ;

// tableau associatif à 1 dimension (vecteur)
$tab["quatrieme"]= "ghi";
// tableau associatif à plusieurs dimensions
$user["admin"]["nom"]= "Simpson";
$user["admin"]["prenom"]="Bart";

$mat [1]["nom"]= "Manfroid"; // combinaison d'indices associatif et numérique
?>
```

3.5 Structures de contrôle et répétitives

De même, les structures de contrôle et les boucles sont similaires à celles de Java :

— La condition **if**

```php
<?php
$a = 34;
$b= 136;
if ($a == $b) {
    echo "<p>\$a ($a) est égal à \$b ($b)";
    $c=23;
}
elseif ($a < $b)
    echo "<p>\$a ($a) est plus petit que \$b ($b)";
else
    echo "<p>\$a ($a) est plus grand que \$b ($b)";
?>
```

— L'alternative à choix multiples **switch**

```php
<?php
$i = 2; switch ( $i ) {
case 0 :
print '$i = 0';
break; case 1 :
print '$i = 1';
break; case 2 :
print '$i = 2';
break; default :
print '$i est très étrange';
}
?>
```

— La boucle "pour" **for**

```php
<?php
for ($i = 0; $i != 10; $i++){
    print $i;
}
?>
```

— La boucle "pour chaque" **foreach**

```php
<?php
$arr = array(1, 2, 3, 4);
foreach ($arr as &$value) {
    $value = $value * 2;
}
?>
```

— La boucle "tant que" **while**

```php
<?php
$i = 0;
while ($i != 10) {
    print $i." <br>";
    $i++;
}
?>
```

— La boucle "faire - tant que" **do while**

```php
<?php
$i=0;
do {
    print $i." <br>";
} while ($i != 0)
?>
```

Certaines instructions permettent d'influencer le déroulement de conditions ou de boucles :

— L'instruction **break;** permet de sortir d'une condition ou d'une boucle.

— L'instruction **continue;** permet

3.6 Inclusion d'un fichier

L'inclusion **include** permet de charger et d'exécuter d'autres fichiers PHP. Par exemple :

```php
<?php
$files = array('prem.php','deux.php', 'trois.php');
for ($i=0 ; $i < count ($files); $i++){
    include $files[$i];
}
?>
```

Ce fichier va charger et exécuter le contenu des fichiers **prem.php**, **deux.php** et **trois.php**. Notez qu'en particulier, cela signifie que tous les **echo** effectués dans ces fichiers seront appliqués, et toutes les définitions de fonctions ou classes dans ces fichiers seront désormais accessibles.

3.7 Fonctions

La définition et l'appel de fonctions utilise une syntaxe similaire à celle de JavaScript : le mot-clé **function** permet de définir une nouvelle fonction, et l'appel se fait via le nom de la fonction suivi de parenthèses (contenant éventuellement les arguments de la fonction).

```php
<?php
// DEFINITIONS DES FONCTIONS
function entete($titre) {
    echo "<html>
    <head><title> $titre </title></head>
    <body>";
}
function pied_de_page() {
    echo "<hr>
    </body>
    </html>";
}
// APPELS
entete("introduction à php");
echo "<h1>Utilisation de fonctions</h1>";
pied_de_page();
?>
```

L'instruction **return** permet de renvoyer une valeur de retour. Si on souhaite renvoyer plusieurs valeurs, on passera par un tableau. Par exemple :

```php
<?php
function addition($nb1, $nb2) {
    $res= $nb1 + $nb2;
    return $res;
}
function add_mult($nb1, $nb2) {
    $res1 = $nb1 + $nb2;
    $res2 = $nb1 * $nb2;
    return array($res1, $res2);
}

$resultat = addition(8,6);
echo "la somme vaut : ".$resultat;

$resultats = add_mult(8,6);
echo "la somme vaut : ".$resultats[0] . "<br/> ";
echo "le produit vaut : ".$resultats[1] . "<br/> ";
?>
```

Il est aussi possible de renseigner des valeurs par défaut aux paramètres de vos fonctions. Faites attention à placer les arguments possédant des valeurs par défaut à la fin de vos en-têtes de fonction !

```php
<?php
// FONCTIONS avec valeurs par défaut pour les arguments
function entete($titre, $bgcolor= "red")
{
    echo "<html>
    <head><title> $titre </title></head>
    <body bgcolor='$bgcolor'>";
}
function pied_de_page($email= "info@he2b.be") {
    echo "<hr>Courriel : $email
    </body>
    </html>";
}
// APPELS
entete("introduction à php" );
echo "<h1>Utilisation de fonctions</h1>";
pied_de_page();
?>
```

3.8 Classes

PHP est un langage orienté objet : il est donc possible d'y définir des classes.

Les attributs sont définis comme des variables locales au sein d'une classe, et les méthodes comme des fonctins au sein de la classe également.

```php
<?php
class dossier
{
    // Attributs
    var $nom = null;
    var $fichiers = array( );
    // Méthodes
    function dossier ($valeur) {
        $this->nom = $valeur ;
    }
    function ajouter_fichier ($nom_fichier) {
        $this->fichiers[] = $nom_fichier;
    }
}
```

Le constructeur de classe se note comme toute autre fonction, mais a pour nom le nom de la classe. Notez aussi l'utilisation du mot-clé `$this` pour faire référence à l'objet lui-même au sein de fonction, et de l'opérateur flèche `->` pour accéder à un attribut ou appeler une méthode sur l'objet.

Pour créer un objet, on procèdera comme suit :

```php
<?php
$dossier = new dossier("maison ") ;
$dossier->ajouter_fichier("cuisine") ;
?>
```

4 Quelques fonctions utiles

4.1 Manipulation de chaînes de caractères

Documentation complète : https://www.php.net/manual/fr/book.strings.php.

- `strlen(string)` : fournit la longueur de la chaîne
- Gestion de la casse :
  - `strtolower(string)` : met les caractères en minuscules
  - `strtoupper(string)` : met les caractères en majuscules
  - `ucfirst(string)` : met le premier caractère en majuscule
- Gestion des sous-chaînes :
  - `substr(string, start, length)` : extrait une partie d'une chaîne de caractères
  - `substr_count(source, pattern)` : compte le nombre d'occurrences de la sous-chaîne `pattern` dans la chaîne `source`
  - `strrchr(string, pattern)` : extrait la fin de la chaîne `string`, à partir de la dernière occurrence de la sous-chaîne `pattern`
  - `str_replace(pattern, replacement, source)` : remplace toutes les occurrences de 'pattern' trouvées dans la 'source' par la chaîne 'replacement'
  - `trim(string)` : retourne un string sans les espaces blancs ("\n","\r","\t","\v", "\0", " ") en début et fin de chaîne.
  - `ltrim(string)` : retourne un string sans les espaces blancs de début de chaîne.
  - `rtrim(string)` : retourne un string sans les espaces blancs de fin de chaîne.
  - `explode(separator, string)` : scinde une chaîne sur base d'un séparateur et retourne un tableau contenant les éléments résultant de la division.
  - `implode(join, array)` : retourne une chaîne formée par la concaténation des chaînes présentes dans `array`, séparées par la sous-chaîne présente dans `join`
  - `split(pattern, join)` : scinde une chaîne aux occurrences de la sous-chaînes `pattern` et renvoie un tableau contenant les chaînes ainsi formées
- Suppression de texte indésirable :
  - `strip_tags(string [,allowable_tags])` : supprime les balises HTML et PHP dans une chaîne.
  - `stripslashes(string)` : supprime les caractères \dans une chaîne de caractères.

— Encodage pour URL :
  — `urlencode(string)` : convertit une chaîne selon les règles d'encodage d'URL
  — `rawurlencode(string)` : convertit une chaîne en URL selon la norme IETF [RFC1738] (espace en '%20' au lieu de '+', utile lors d'utilisation d'URL avec le protocole FTP par exemple)
  — `urldecode(string)` : décode une chaîne selon les règles d'encodage d'URL
  — `htmlspecialchars(string)` : convertit les caractères spéciaux ('&',' " ','<','>') en équivalent HTML ('& ', '"' , '< , '>')
— Encodage pour HTML :
  — `nl2br(string)` : convertit les retours à la ligne en retour de chariot HTML `<br>`.
  — `htmlentities(string)` : convertit les caractères significatifs en HTML.
—

4.2 Manipulation de tableaux

Documentation complète : https://www.php.net/manual/fr/book.array.php

— `array_walk($var, fct)` : applique la fonction `fct` à chaque élément du tableau `$var`
— `list(var1, var2, ...)` : permet d'assigner une série de variables à partir d'un tableau : `list($drink, $color, $power) = $info;`
— `each(array)` : retourne chaque paire clé/valeur d'un tableau d'assignation
— `sizeof(array)` : retourne le nombre d'éléments du tableau.
— `is_array(var)` : teste si une variable est de type tableau.

4.3 Manipulation de répertoires et de fichiers

Documentation complète : https://www.php.net/manual/fr/refs.fileprocess.file.php

— Navigation
  — `opendir(dir)` : ouvre un répertoire et récupère un pointeur
  — `readdir(dir_handle)` : retourne le nom du fichier suivant dans le répertoire pointé par `dir_handle` (obtenu via `opendir`)
  — `rewinddir(dir_handle)` : se replace au début du répertoire `dir_handle`
  — `closedir(dir_handle)` : ferme le répertoire
  — `is_dir(filename)` : vérifie si la cible est un répertoire
  — `chdir()` : change le dossier courant de PHP.
— Manipulation de fichier
  — `copy("anc_fich","nouv_fich")` : copier 1 fichier
  — `unlink(filename)` : supprimer le fichier
  — `filesize(filename)` : retourne la taille d'un fichier en octets

- `file_exists(filename)` : teste si un fichier existe
- `tempnam(dir, prefix)` : crée un fichier et lui donne un nom unique
- `flock(file_handle, operation_type)` : permet un verrou sur un fichier en cas d'accès concurrents

— Ouverture et modification de fichier

- `fopen(filename, mode)` : ouvre un fichier et retourne un pointeur sur ce fichier.
- `fclose(file_handle)` : ferme le pointeur sur le fichier ouvert
- `fwrite(file_handle, string[,length])` : écrit dans un fichier
- `fread(file_handle, length)` : lit un fichier
- `fgets(file_handle, length)` : lit la ligne courante d'un fichier
- `file(filename)` : renvoie le contenu d'un fichier sous forme d'un tableau
- `readfile(filename)` : récupère le contenu d'un fichier et l'envoie sur la sortie standard
- `feof(file_handle)` : teste si la fin du fichier a été atteinte
- `tmpfile()` : crée un fichier temporaire et l'ouvre en mode écriture

4.4 Expressions régulières

Documentation complète : https://www.php.net/manual/fr/book.regex.php

Les expressions régulières permettent de représenter une catégorie de chaînes de caractère qui correspondent toutes au même modèle, afin de faire des manipulations plus poussées.

- `ereg(pattern, string[, regs])` : Cherche le modèle `pattern` dans la chaîne et place éventuellement le résultat de recherche dans le tableau `regs`
- `eregi(pattern, string[, regs])` : Idem, mais insensible à la casse.
- `ereg_replace(pattern, replacement, string])` : Cherche le modèle `pattern` dans la chaîne et les remplace par `replacement`
- `eregi_replace(pattern, replacement, string])` : Idem, mais insensible à la casse.