Web - TP n^o 4
G. Subrenat, R. Zrour

Licence Informatique L3 - 2020-2021
UFR SFA - Université de Poitiers

Survol de PHP

Table des matières

1 Résumé

Dans ce TP nous verrons quelques fonctionnalités du langage PHP. Il s'agit uniquement d'un survol ; vous vous familiariserez avec le langage au fil des TP.

Pour chaque exercice ou question d'exercice, faites une page web à part pour être sûr que vos divers codes n'interfèrent pas.

Le site http://php.net contient, entre autre, la description des fonctions de PHP ; vous aurez besoin de vous y référer tout au long du TP.

Il vous est fourni un serveur web correctement configuré. Vous pouvez travailler sur un serveur personnel, mais c'est alors de votre responsabilité de le configurer correctement.

Rectificatif confinement au cas où ce dernier persisterait : le serveur de l'université n'étant pas accessible de l'extérieur, il vous est fourni plusieurs pistes pour installer votre propre serveur.

2 Serveurs web

Point de cours

Un navigateur web interprète uniquement les langages suivants :

- HTML
- CSS
- JavaScript

Et donc notamment il ne connaît pas PHP. Si un navigateur reçoit du code PHP, le comportement est indéterminé (mais le PHP ne sera pas exécuté).

Un serveur basé sur PHP ne connaît pas les 3 langages sus-cités. Son but est de fournir, au navigateur, du code contenant ces 3 langages. Généralement le code n'est pas figé mais fabriqué à la volée par le serveur et c'est le rôle du langage supporté par le serveur (PHP dans notre cas).
Le rôle de PHP :

- recevoir la requête du navigateur : une URL accompagnée éventuellement de données supplémentaires (un formulaire renseigné par exemple)
- analyser l'URL et les données
- interroger la base de données
- générer du code HTML (plus rarement du CSS ou du JavaScript [a])
- l'envoyer au navigateur

a. dans le sens où les fichiers *.css* et *.js* sont transmis directement et non pas fabriqués dynamiquement.

Même si tous les sites que nous allons faire fonctionnent certainement avec la version 8 de PHP, il est peut-être préférable d'utiliser la version 7.
Il faut au minimum la version 7.1 pour les TP ultérieurs ; la version 7.3 ou 7.4 est recommandée.

Il est conseillé d'installer le serveur web (et plus particulièrement PHP) avant les autres logiciels dont on aura ultérieurement besoin comme *composer* et *symfony*.

2.1 Serveur fourni

Le serveur de pages web se nomme *tpweb* (192.168.49.9). Vous devez faire une connexion *ssh* [1] (avec les identifiants de votre compte SEL) pour créer votre répertoire d'accueil :

```
ssh votrelogin@tpweb
```

ou plus simplement :

```
ssh tpweb
```

1. ou *sftp*.

Ce serveur est inaccessible en dehors du réseau de l'université.

ATTENTION : un fichier *.my.cnf* et un répertoire *.my.cnf.d* sont également créés pour l'accès aux bases de données. NE les modifiez PAS, pas plus que le mot de passe d'accès à la BD.
Et au cas où ce ne serait pas clair, il NE faut PAS modifier ce fichier ou ce répertoire [2].

Ensuite, vous pouvez télécharger vos fichiers via *ssh* ou *sftp* (par exemple vous pouvez utiliser le logiciel *FileZilla*).
Les fichiers sont à déposer dans le répertoire *public_html* ou un de ses sous-répertoires.
Votre site est alors accessible à l'adresse (remplacer *login* par votre login) :

```
http://login.projet.sfasp2mi.univ-poitiers.fr
```

2.2 Serveur distant

Si vous avez un serveur chez un hébergeur distant, vous pouvez l'utiliser.

2.3 LAMP (Linux type debian)

2.3.1 Installation service par service

Il faut installer quelques paquets : *apache2*, *mysql-server* (ou *mariadb-server*), *php*, *libapache2-mod-php*, *php-mysql*.

On peut aussi installer *phpmyadmin* (qui installe par dépendances plusieurs autres paquets utiles).

Il peut y avoir des configurations à faire (pour *apache* en général, ou *mysql* ou *phpmyadmin*).

Voici quelques liens :

- https://doc.ubuntu-fr.org/lamp
- https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-20-04
- https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-20-04-fr

2.3.2 Installation tout en un

XAMPP est une installation préconfigurée et installée dans un seul répertoire.
XAMPP est multi-plateforme.

Voici quelques liens :

- https://doc.ubuntu-fr.org/xampp
- https://www.apachefriends.org/fr/index.html

2.4 Windows

2.4.1 WampServer

WampServer est un des classiques et est un logiciel tout-en-un. Il permet de changer de version de PHP (ou MySQL ou Apache) simplement. De même beaucoup de configurations se font via les menus.

WampServer écoute par défaut sur le port 80. Cela peut entrer en conflit avec d'autres logiciels [3].

- solution 1 : stopper les logiciels conflictuels
- solution 2 : changer le port d'écoute (en 8080 par exemple) en modifiant le fichier *httpd.conf*.

Voici quelque lien :

- https://www.wampserver.com

2. Sinon vous irez plaider votre cause auprès de l'ingénieur système.
3. comme Skype, du moins à une certaine époque

2.4.2 XAMPP

XAMPP, décrit dans la section sur Linux, s'installe aussi sur Windows.

2.5 MacOS

XAMPP, décrit dans la section sur Linux, s'installe aussi sur MacOS.

2.6 Exercice

Déposez le fichier ci-dessous sur le serveur et accédez-y via le navigateur.

bonjour.php

```php
<!DOCTYPE html>
<html lang="fr">
    <head>
        <meta charset="UTF-8">
        <title>Welcome</title>
    </head>
    <body>
        <div>
            <?php
                echo 'bonjour';
                phpinfo();
            ?>
        </div>
    </body>
</html>
```

3 Aperçu de PHP

Le but est de faire de la programmation classique et de faire des affichages simples avec *echo*.
Il n'est pas obligatoire de passer par le naviteur car seule la manipulation du langage nous intéresse.
Ceci dit, pour certaines questions (utilisation de *print_r*, ou de table HTML par exemple) il est intéressant de voir le résultat dans un navigateur.

Pour exécuter un fichier *.php* en ligne de commande :
`$ php -f <nom-fichier>`

Pour lancer un interpréteur : `$ php -a`

Problème des retour à ligne :

- en mode console : `echo "\n"` (il faut des guillemets et non des apostrophes)
- en mode HTML : `echo '<br />'`

a) opérateurs +

Écrivez une fonction qui prend deux entiers en paramètres et en renvoie la somme.

Dans un premier temps, déclarez deux variables x et y que vous initialisez directement à *123* et *654*, appelez la fonction et affichez le résultat. Vérifiez que tout est correct.

Pourquoi ne peut-on pas initialiser x et y avec des prompts ?

Initialisez x et y avec *"123"* et *"654"* (guillemets compris) pour en faire des chaînes. Quel est le résultat de la fonction ?

Et avec les valeurs *"123"* et *"aaa"* ?

Et avec les valeurs *"123"* et *"1aaa"* ?

Et avec les valeurs *"aaa1"* et *"123"* ?

D'ailleurs que se passe-t-il si on passe autre chose que des nombres ?

b) égalité ?

Testez le code suivant et dites si le résultat est logique :

```
if (1 == "miaou")
   echo "il y a un problème (1 == "miaou") ?<br />";
```

Testez le code suivant et dites si le résultat est logique :

```
if (0 == "miaou")
   echo "il y a un problème (0 == "miaou") ?<br />";
```

Écrivez une procédure *comparer* qui prend deux arguments et qui :

- affiche le type et la valeur de chacun des arguments,
- compare les deux paramètres avec l'opérateur == et affiche le résultat,
- compare les deux paramètres avec l'opérateur === et affiche le résultat.

Affectez les variables *a* et *b* et chaque fois appelez la procédure *comparer*. Utilisez les valeurs suivantes :

- 123 et 123
- 123 et "123"
- 123 et "miaou"
- 123 et "123miaou"
- 0 et "123miaou"
- 0 et "miaou"
- "123" et "123d0"
- "123" et "123e0" (celui-ci est troublant voire choquant)

Quelle est la différence entre == et ===?

c) existence de variables

Dans chacun des cas suivant, il s'agit de tester l'état d'une variable, c'est à dire :

- afficher si elle existe (fonction *isset*),
- afficher si elle est *null* (fonction *is_null*),
- regarder si sa valeur est *NULL* (avec == et ===),
- afficher sa valeur.

Voici les cas :

1. sur *$v* qui n'existe pas
2. après avoir affecté *$v* à une chaîne quelconque
3. après avoir affecté *$v* à *NULL*
4. après avoir affecté *$v* à la chaîne vide
5. après avoir utilisé la fonction *unset* sur *$v*

d) boucles

Essayez le code suivant :

```
for ($i = 0; $i < 10; $i ++)
{
   echo "ligne " . $i . " : ";
   afficheLigne($i);
}

function afficheLigne($numLigne)
{
   for ($i = 0; $i < 10; $i ++)
      echo $numLigne * $i . " ";
   echo "<br />";
}
```

Tout ce passe-t-il correctement ? Comparez avec JavaScript si vous avez du courage.

Modifiez le code fourni pour afficher la table de multiplication dans une vraie table au sens HTML du terme.

Essayez le code suivant :

```
$i = "miaou";
affiche();
function affiche()
{
   echo $i . "<br />";
}
```

Corrigez le code pour qu'il fonctionne comme on l'attend, mais la fonction ne doit pas prendre de paramètre (ce qui serait pourtant la bonne solution).

e) tableaux indicés par des nombres

Note : toutes les cases ne sont pas obligatoirement du même type.

Écrivez une procédure qui affiche un tableau en utilisant la fonction *print_r* (le code de votre procédure ne doit faire qu'une instruction). Faites en sorte que l'affichage soit agréable à lire, c'est à dire une ligne par case du tableau.
Indices :

- étudiez la balise `<pre></pre>`
- regardez le deuxième paramètre de la fonction *print_r*

Écrivez une procédure qui affiche un tableau en utilisant un *for* classique, la taille du tableau devant être déterminée directement par la procédure (cf. fonction *count*). L'affichage doit se faire dans un tableau XHTML de deux colonnes, la première colonne contenant le numéro des cases, et la seconde la valeur des cases.

Écrivez une procédure qui affiche un tableau en utilisant *foreach*. L'affichage doit se faire dans un tableau XHTML de deux colonnes, la première colonne contenant la clé des cases (ici des numéros), et la seconde la valeur des cases.

Créez un tableau qui contient trois noms de capitales sans préciser le numéro des cases. Testez vos fonctions sur le tableau.

Ajoutez une case sans préciser la clé et mettez 123 dedans (i.e. `$t[] = 123;`).
Ajoutez de la même manière une case avec un booléen.
Affichez le tableau.

Supprimez, avec la fonction*unset* la case numéro 1 (la deuxième donc) et la dernière (normalement la numéro 4). Affichez le tableau avec les trois fonctions, en notant le comportement de la seconde.

Ajoutez à nouveau une case sans préciser le numéro. Quel numéro prend-elle ? Affichez le tableau.

Ajoutez une case en position 10 et affichez le tableau.

Appliquez la fonction *array_values* sur le tableau et affichez le résultat (ne modifiez pas le tableau initial). Faites de même avec la fonction *array_keys*.

f) tableau indicé par des chaînes : tableau associatif

Reprenez le tableau précédent et ajoutez deux cases indicées par des chaînes et affichez-le.
Le fait de mélanger les indices numériques et des indices chaînes pose-t-il un problème ?

Utilisez la fonction *array_values* et *array_keys* pour voir leurs comportements sur un tableau associatif.

Créez un nouveau tableau, en une seule instruction, qui comporte 3 cases dont les clés sont des pays et les valeurs des capitales.

Soit la chaîne : *"Pencroff et Ayrton procédèrent donc aux préparatifs du lancement"*.
Remplacez tous les espaces par des slashs et comptabilisez le nombre d'espaces :

— d'une part avec la méthode *str_replace*,

— d'autre part avec les méthodes *explode* et *implode*.

g) variables de variables

Créez trois variables nommées *capitale_france*, *capitale_espagne* et *capitale_italie* que vous initialisez avec des valeurs adéquates.

Créez une variable *choix* que vous initialisez, au choix, avec *france*, *italie* ou *espagne* (il faut supposer que cette variable vient d'un formulaire par exemple).

En une seule instruction, sans test, affichez la capitale du pays ainsi désigné.

Est-ce que cette manière de faire est algorithmiquement bien pratique ?

4 Fonctions

a) références

Note : avant chaque question, faites un *unset* sur les variables utilisées pour ne pas avoir d'effet de bord.

Créez une variable numérique *$a*, recopiez-la dans *$b*, modifiez *$b* et affichez *$a*. La modification s'est-elle répercutée sur *$a* ? Est-ce le comportement attendu ?

Même question, mais la "recopie" est faite par référence (symbole `&`).

Créez une variable tableau *$a*, recopiez-la dans *$b*, modifiez une case de *$b* et affichez *$a*. La modification s'est-elle répercutée sur *$a* ? Est-ce le comportement attendu ? [4]

Même question, mais la "recopie" est faite par référence.

Créez une variable tableau *$a*, recopiez-la dans *$b* par référence, puis affectez une autre valeur à *$b* (i.e. un autre tableau, ou autre chose qu'un tableau d'ailleurs) et affichez *$a*. La modification s'est-elle répercutée sur *$a* ? Est-ce le comportement attendu ?

b) passage de paramètres

Écrivez une fonction qui prend un paramètre, a priori de type entier, et le modifie. Vérifiez que la modification n'est pas répercutée.

Faites la même chose avec un paramètre de type tableau et la fonction modifie une case du tableau. A-t-on le même comportement qu'en C ?

Maintenant, le paramètre doit être passé par référence. Vérifiez le bon comportement. Que se passe-t-il si l'on passe directement un nombre en paramètre.

c) valeurs par défaut

Les fonctions en PHP peuvent avoir des valeurs par défaut pour leurs paramètres ; les paramètres sans valeur par défaut doivent être déclarés en premier.

Écrivez la fonction *carre_decalage* qui élève au carré son premier paramètre et lui ajoute le second paramètre. Par défaut le décalage doit être de 0.

d) valeur de retour

Au fait, une fonction peut retourner une valeur !

On peut simuler le retour de plusieurs valeurs en retournant un tableau et en séparant les valeurs au retour grâce à la fonction *list*.
Écrivez une fonction qui prend un entier en paramètre (par défaut 0) et renvoie les 3 entiers le suivant.

e) récursivité

Les fonctions récursives sont possibles. Écrivez la fonction factorielle.

f) nom de fonction dans une variable

Il est possible de stocker le nom d'une fonction dans une variable, et d'appeler la fonction via la variable. On se sert du nom de la variable (avec le `$`) comme si c'était une fonction [5].

4. Spoil et avertissement : la modification n'est normalement pas répercutée ; si c'est malgré tout le cas, cherchez où est l'erreur, par exemple en lisant la note en début d'exercice.
5. On peut appliquer ce principe pour les méthodes des classes.

Appelez la fonction *factorielle* en passant par l'intermédiaire d'une variable.

g) deux fonctions particulières

La fonction *function_exists* permet de tester ... l'existence d'une fonction.

La fonction *get_defined_functions* permet d'avoir la liste de toutes les fonctions disponibles. Affichez le retour de cette fonction et regardez les deux catégories qu'elle distingue.

h) à étudier

- écrire une fonction dans une fonction
- nombre de paramètres variable
- fonctions anonymes

5 HTML, SQL, URL

a) passage à la ligne [6]

Utilisez la fonction *nl2br* pour insérer les balises *br* dans une chaine contenant des $\backslash n$.
Essayez une chaîne délimitée par des apostrophes, et une autre délimitées par des guillemets.

Vous remarquez que les $\backslash n$ ne sont pas remplacés par la fonction (il faut regarder le code source de la page pour s'en apercevoir) : les balises *br* sont insérées "avant" et non pas "à la place de". Comment faire un remplacement ?

Écrivez le code inverse, avec la fonction *str_replace*, qui supprime les `<br />`.

Écrivez le code inverse qui supprime indifféremment les `<br />`, `<br/>` et `<br>`. Faites tout d'abord 3 instructions avec la fonction *str_replace*, puis essayez de le faire en une seule instruction avec la fonction *preg_replace*.

b) requêtes SQL

Les apostrophes peuvent poser des soucis dans les requêtes SQL. Testez les fonctions *addslashes* et *stripslashes* sur une chaîne contenant des apostrophes, des guillemets et des backslashes.

c) Entités HTML

En utilisant *htmlspecialchars*, écrivez le code PHP permettant d'affichez dans la page, au sein d'un paragraphe, le texte :

```
Un aspect <strong>important</strong> de "ça" est le 'mors'
```

Dans la même page, affichez juste en dessous l'encodage qui doit être fait dans le code HTML pour que l'on puisse effectivement voir ce qui est demandé ; sur la page doivent donc s'afficher les deux lignes :

```
Un aspect <strong>important</strong> de "ça" est le 'mors'
Un aspect &lt;strong&gt;important&lt;/strong&gt; de &quot;ça&quot; est le &#039;mors&#039;
```

Dans l'appel de la fonction *htmlspecialchars*, examinez la différence de résultat selon que l'on utilise, ou pas, le flag *ENT_QUOTES*. Qu'est-ce que cela donne si on remplace *htmlspecialchars* par *htmlentities* ? Dans le même esprit, montrez ce que donnent les deux fonctions *htmlspecialchars_decode* et *htmlentities_decode*.

d) URL

Dans une URL, beaucoup de caractères sont interdits, notamment les espaces et les accents [7]. Testez les fonctions *urlencode*, *rawurlencode*, *urldecode* et *rawurldecode*.

e) hashage

Testez les fonctions *md5*, *crypt*, *sha1* et *password_hash*. Pour cette dernière, testez les algorithmes *PASSWORD_DEFAULT*, *PASSWORD_BCRYPT*, *PASSWORD_ARGON2I* et *PASSWORD_ARGON2ID*.

6. la quasi-totalité des réponses sont dans la documentation de *nl2br* sur le site *php.net*.
7. si la norme n'a pas évolué ces dernières années

Cherchez les fonctions inverses (sans y passer trop de temps).

6 print_r (facultatif et hors programme)

Programmez votre propre fonction *print_r* selon le prototype suivant :

```
mixed myprint_r(mixed $expr, bool $ret = false, $withPre = true, $indent = 4)
```

La valeur de retour et les deux premiers paramètres sont les mêmes que la fonction fournie par PHP. Le paramètre `$withPre` indique si la production est entourée par les balises `<pre>` et `</pre>`. Enfin le paramètre `$indent` indique l'indentation pour chaque niveau de profondeur.

Les méthodes *is_array*, *str_repeat* pourraient être utiles.
Note : on ne demande pas de gérer les objets.
Note : la gestion de l'indentation est assez délicate ; comparez le résultat de votre fonction avec celui de *print_r*.
Note : affichez *true* ou *false* pour les booléens.

Testez, avec votre méthode et avec *print_r*, les variables avec les valeurs suivantes :
- *true*
- *false*
- 123
- un tableau avec numérotation numérique et contenu numérique.
- un tableau associatif avec des contenus de différents types simples.
- un tableau donc les cases peuvent contenir des tableaux avec ainsi une profondeur de 4 :

```
$e = array (
      "nom" => "tartempion",
      "ville" => array (
            "code"  => 86000,
            "nom"   => "Poitiers",
            "staff" => array (
                  "maire"    => "Terminator",
                  "adjoints" => array ("Bisounours", "Freddy"),
                  "associes" => "Dora",
                  "successeurs" => array("Batman", "Robin"),
               ),
         ),
      "metier" => "employé",
   );
```

7 XML (facultatif et hors programme)

L'idée de l'exercice est de transformer un élément PHP (formaté spécialement) en un fichier XML et réciproquement.

Pour simplifier, on suppose que :
- il n'y a que des balises doubles (pas de balise auto-fermante),
- les signes < et > ne sont utilisés que pour délimiter les balises,
- les " ne servent qu'à délimiter les attributs,
- on suppose que le fichier XML (ou le tableau selon le sens de transformation) est correct,
- au top level, il n'y a qu'un seul élément (une balise vraisemblablement).

Un élément est :
- soit une chaîne de caractères (feuille texte de l'arbre XML),
- soit un élément-balise (i.e. la balise et tous ses fils).

Un élément-balise est un tableau associatif :
- la première case (obligatoire) a pour clé la chaîne `"name"` et comme valeur une chaîne indiquant le nom de la balise,
- la deuxième case (obligatoire) a pour clé la chaîne `"attributs"` et comme valeur un tableau associatif de paires (clé, chaîne) ; la clé est le nom d'un attribut, et la chaîne sa valeur.
- les autres cases (facultatives) ont pour clés les nombres 0, 1, 2, ... Et la valeur de chaque case est un élément.

Référez-vous à Updago pour voir un exemple complet de correspondance entre un tableau PHP et un fichier XML.

Écrire un programme qui transforme un tableau en fichier XML (avec *fopen*, *fprintf*, ...).
Le fichier doit s'écrire au fur et à mesure de la lecture du tableau (on ne passe pas par des structures intermédiaires).
Pour vérifier votre résultat, ouvrez le fichier généré avec votre navigateur (et aussi un éditeur de texte).

Écrire un programme qui lit un fichier au format XML pour le transformer en un tableau PHP (que vous pourrez afficher avec la fonction de l'exercice ci-dessus).
Il est plus simple de lire le fichier dans une chaîne (cf. *file_get_contents*) et d'analyser la chaîne.
Cette partie est délicate.