Programmation Web Serveur

M3104 - TP 1

Les exercices suivants sont les exercices du premier chapitre de cours de PHP. Pour tous les exercices, nous allons utiliser le serveur local. Étant donné la configuration des ordinateurs de l'IUT, le serveur local ne peut accéder qu'aux fichiers du répertoire `/home/VOTRE_NUMERO_ETUDIANT/public_html` . Pour accéder à ces documents en passant par le serveur local, il faut alors taper l'url :

http://localhost/~VOTRE_NUMERO_ETUDIANT/chemin/nom_du_fichier

où `chemin` correspond au chemin où se trouve votre fichier à partir du répertoire `public_html` . Par exemple, si votre numéro d'étudiant est le 001 et si vous créez un dossier `TP1` dans votre répertoire `public_html` et un fichier `exercice1.php` dans `TP1` , pour accéder à ce fichier, vous devez spécifier l'adresse Web :

http://localhost/~001/TP1/exercice1.php

Attention : Les données stockées dans le répertoire `public_html` sont supprimées à chaque redémarrage de l'ordinateur. Le moyen le plus simple pour que vos documents soient sauvegardés tout en étant accessibles via le serveur local est d'utiliser le task runner robo. Correctement configuré, ce task runner va vous permettre de synchroniser votre répertoire de travail (situé dans la partie sauvegardée de vos documents) avec le répertoire `/home/VOTRE_NUMERO_ETUDIANT/public_html` . Cette

synchronisation implique que chaque changement effectué dans votre répertoire de travail sera automatiquement appliqué dans le répertoire `public_html` .

Pour configurer facilement le task runner, effectuer les instructions suivantes :

1. Ouvrez un terminal.
2. Déplacez-vous dans le répertoire sauvegardé (`cd Bureau/Mes_Montages/VOTRE_NUMERO_ETUDIANT`).
3. Clonez ce dépôt Github (`git clone https://github.com/mathieuLacroix/M3104.git`).
4. Allez dans le répertoire cloné (`cd M3104`).
5. Lancez la synchronisation : `robo sync` . Le répertoire `web` se trouvant dans le répertoire `M3104` est maintenant synchronisé avec le répertoire `public_html` .
6. Vérifiez le fonctionnement. L'url http://localhost/~VOTRE_NUMERO_ETUDIANT/hello_world.php doit exécuter le script `/home/~VOTRE_NUMERO_ETUDIANT/hello_world.php` et afficher `hello world` . Modifiez ensuite le fichier `web/hello_world.php` , sauvegardez-le et vérifiez que le rafraîchissement de la page affiche bien les modifications !
7. À partir de maintenant, vous travaillerez systématiquement dans le répertoire `M3104/web` et lancerez la synchronisation via robo. Lisez le fichier `README` du dépôt Github pour plus d'information sur l'utilisation du task runner.

Remarque : Comme l'ensemble des réponses aux exercices contiennent du code PHP, il n'est pas possible d'ouvrir le fichier directement à l'aide du navigateur. Il est donc impératif de passer par le serveur local.

Exercice 1 : Premiers pas en PHP

Voici un code php.

```
<!doctype html>
<html>
<head>
<title> <?php echo 'Premiers pas en PHP'; ?> </title>
<meta charset="utf-8"/>
</head>
<body>
<h1> Mes premiers pas en PHP </h1>
<?php $tps=2; echo '<p> Je débute depuis '; echo $tps; echo ' heures... </p>';
<p> Mais cela a l'air intéressant <?php echo '!' ?> </p>
<?php echo '
<h2> Vive le PHP </h2>
<p> Les pages vont pouvoir être dynamiques! </p>
'; ?>
<p> Encore quelques paragraphes </p>
echo '<p> Avant dernier paragraphe </p>';
<p> Voilà, c'est terminé! </p>
</body>
</html>
```

Répondre aux questions suivantes :

1. Donner dans ce fichier les parties correspondant à du code PHP et à du code HTML.
2. Si ce fichier s'appelle `exo1.php`, et s'il est stocké sur le site Web `www.exemple.org` dans le répertoire `PremierTP`, comment "exécuter" ce fichier ? Que donne son exécution ? Ceci est-il valide ? Pourquoi ? Corriger alors le problème.
3. Remplacer les trois instructions `echo` de la ligne 9 par une seule.

Exercice 2 : Inclusion d'en-tête et pied de page

Le langage PHP permet d'inclure des fichiers dans d'autres. Ceci permet alors de décomposer un code HTML ou PHP en plusieurs parties logiques et d'insérer ensuite ces différentes parties dans un fichier PHP.

1. Créer un fichier `debut_code_html.php` contenant tout le code HTML jusqu'à la balise `<body>` incluse.
2. Créer un fichier `fin_code_html.php` contenant les balises `</body></html>` .
3. Créer un fichier `code.php` contenant un titre et un paragraphe et incluant les fichiers `debut_code_html.php` et `fin_code_html.php` .
4. Modifier le fichier `debut_code_html.php` pour que le titre de l'onglet (balise `<title>`) dépende de la valeur de la variable `$title` définie dans le fichier `code.php` .

Exercice 3 : Initiation aux tableaux

Voici un tableau contenant les différents mois de l'année :

```
$mois = [
    "janvier",
    "février",
    "mars",
    "avril",
    "mai",
    "juin",
    "juillet",
    "août",
    "septembre",
    "octobre",
    "novembre",
```

```
    "décembre"
];
```

1. Quelles sont les clés ? Parcourir ce tableau pour afficher les mois de l'année sous la forme d'une liste ordonnée.

On considère maintenant le tableau suivant donnant pour chaque mois de l'année, le nombre de jours qu'il contient (On suppose que l'année n'est pas bissextile.) :

```php
$mois = [
    "janvier"   => 31,
    "février"   => 28,
    "mars"      => 31,
    "avril"     => 30,
    "mai"       => 31,
    "juin"      => 30,
    "juillet"   => 31,
    "août"      => 31,
    "septembre" => 30,
    "octobre"   => 31,
    "novembre"  => 30,
    "décembre"  => 31
];
```

2. Quelles sont les clés du tableau ? Les valeurs ? Parcourir ce tableau pour afficher les mois de l'année sous la forme d'une liste ordonnée (comme dans la question précédente).

3. Modifier le code pour afficher dans la liste après le nom de chaque mois le nombre de jours qu'il contient.

Exercice 4 : Tableau à deux dimensions

On définit le tableau suivant :

```php
$personnes = [
  'mdupond' => ['Prénom' => 'Martin', 'Nom' => 'Dupond', 'Age' => 25, 'Ville' =
  'jm'      => ['Prénom' => 'Jean'  , 'Nom' => 'Martin', 'Age' => 20, 'Ville' =
  'toto'    => ['Prénom' => 'Tom'   , 'Nom' => 'Tonge' , 'Age' => 18, 'Ville' =
  'arn'     => ['Prénom' => 'Arnaud', 'Nom' => 'Dupond', 'Age' => 33, 'Ville' =
  'email'   => ['Prénom' => 'Emilie', 'Nom' => 'Ailta' , 'Age' => 46, 'Ville' =
  'dask'    => ['Prénom' => 'Damien', 'Nom' => 'Askier', 'Age' => 7 , 'Ville' =
];
```

1. Quelles sont les clés du tableau `$personnes` et leur type ? De quel type sont les valeurs de ce tableau ? Quelle est la valeur associée à *'toto'* ?

2. Comment accéder à la valeur *33* dans le tableau ? À la valeur *'Epinay'* ? Au tableau contenant les valeurs *'Damien'*, *'Askier'*, *7*, *'Villetaneuse'* ?

3. Écrire une fonction prenant en paramètre un tableau (avec une structure similaire à `$personnes`) et l'affichant sous forme d'une table HTML. Le code HTML obtenu avec le tableau `$personnes` doit être :

```html
<table>
    <tr>
        <td>Martin</td>
        <td>Dupond</td>
        <td>25</td>
        <td>Paris</td>
    </tr>
    <tr>
        <td>Jean</td>
        <td>Martin</td>
        <td>20</td>
        <td>Villetaneuse</td>
    </tr>
    <tr>
        <td>Tom</td>
        <td>Tonge</td>
        <td>18</td>
        <td>Epinay</td>
    </tr>
    <tr>
```

```
            <td>Arnaud</td>
            <td>Dupond</td>
            <td>33</td>
            <td>Paris</td>
        </tr>
        <tr>
            <td>Emilie</td>
            <td>Ailta</td>
            <td>46</td>
            <td>Villetaneuse</td>
        </tr>
        <tr>
            <td>Damien</td>
            <td>Askier</td>
            <td>7</td>
            <td>Villetaneuse</td>
        </tr>
    </table>
```

4. Modifier la fonction précédente pour ajouter une ligne d'en-têtes. Les en-têtes seront les clés du tableau contenu dans la première case. Dans le cas de `$personnes`, les en-têtes seront *Prénom, Nom, Age, Ville*.
   Indice : Utiliser `array_keys($personnes)[0]` pour récupérer la première clé du tableau `$personnes`.
5. Vérifier que l'affichage obtenu est correct lorsque la fonction prend en paramètre le tableau suivant :

```php
$scores = [
  ['Joueur' => 'Camille'  , 'score' => 156 ],
  ['Joueur' => 'Guillaume', 'score' => 254 ],
  ['Joueur' => 'Julien'   , 'score' => 192 ],
  ['Joueur' => 'Nabila'   , 'score' => 317 ],
  ['Joueur' => 'Lorianne' , 'score' => 235 ],
  ['Joueur' => 'Tom'      , 'score' => 83  ],
  ['Joueur' => 'Michael'  , 'score' => 325 ],
  ['Joueur' => 'Eddy'     , 'score' => 299 ]
];
```

Exercice 5 : Tableau à deux dimensions

On considère les deux tableaux suivants :

```php
$tabMagazines = [
  'le monde'              => ['frequence' => 'quotidien', 'type' => 'actualité
  'le point'              => ['frequence' => 'hebdo'    , 'type' => 'actualité
  'causette'              => ['frequence' => 'mensuel'  , 'type' => 'féministe
  'politis'               => ['frequence' => 'hebdo'    , 'type' => 'opinion'
  'le monde diplomatique' => ['frequence' => 'mensuel'  , 'type' => 'analyse'
];

$tabMagazinesAbonne = ['le monde', 'le monde diplomatique'];
```

1. Afficher sur une ligne le nom de tous les magazines triés par ordre alphabétique et séparés par des virgules, sans faire de boucle. Vous utiliserez des fonctions déjà existantes telles que *implode*, *sort* et *array_keys* que vous trouverez dans le manuel php.

2. Afficher les magazines exactement de la façon suivante en supposant qu'il peut y avoir beaucoup de magazines et beaucoup de propriétés associées :
   - le monde (quotidien, actualité, 220)
   - le point (hebdo, actualité, 80)
   - causette (mensuel, féministe, 180)
   - politis (hebdo, opinion, 100)
   - le monde diplomatique (mensuel, analyse, 60)

3. En utilisant le tableau tabMagazinesAbonne contenant le nom des magazines d'un abonné, calculer le prix total de son abonnement.

Exercice 6 : Objets

L'objectif de cet exercice est de développer une classe permettant la gestion d'une TODO liste (liste de tâches à réaliser).

1. Créer la classe `TODOList` contenant un attribut privé `$to_dos`, correspondant à un tableau contenant la liste des tâches à réaliser (chaque case correspond à une tâche qui est une chaîne de caractères). Les clés correspondront aux indices. Cette classe sera définie dans le fichier `TODOlist.php`.
2. Définir un constructeur ne prenant pas d'argument et initialisant une TODO liste vide.
3. Définir la méthode `add_to_do` prenant en paramètre une chaîne de caractères (correspondant à une tâche). Si cette chaîne n'est pas vide et n'est pas constituée uniquement d'espaces, la méthode ajoute cette tâche à la fin de la TODO liste.
4. Définir la méthode `remove_to_do` prenant en paramètre un indice et supprimant la tâche associée à cet indice dans la TODO liste.
5. Définir la méthode `is_empty` retournant `true` s'il n'y a aucune tâche, et `false` sinon.
6. Définir la méthode `get_html` retournant une chaîne de caractères correspondant au code HTML d'une liste non ordonnée où chaque item correspond à une tâche. Dans le cas où la TO DO liste est vide, `get_html` doit renvoyer le paragraphe "Aucune tâche à faire !".
7. Créer un script `testTDL.php` créant une TODO liste avec 4 tâches, l'affichant, supprimant une des tâches puis l'affichant à nouveau. Ajouter des tests pour vérifier que la méthode `is_empty` fonctionne correctement et qu'une tâche vide ou constituée uniquement d'espaces n'est pas ajoutée à la liste.

Exercice 7 : Paramètres dans l'url

1. Appeler une page PHP en passant dans l'url un paramètre de nom `pseudo`. Le script devra vérifier si la valeur du paramètre correspond à un pseudonyme (c'est-à-dire une clé) du tableau `$personnes` défini dans l'exercice 4. Si c'est le cas, le script doit afficher le pseudo et les informations associées contenues dans le tableau `$personnes`. Il doit sinon afficher `Désolé, votre pseudonyme n'apparaît pas dans la liste`.

2. Créer un formulaire permettant à l'utilisateur de saisir le pseudonyme à rechercher afin de faciliter la saisie pour l'utilisateur. Mettre ensuite directement le formulaire dans le script PHP créé précédemment afin de pouvoir effectuer facilement plusieurs recherches. Faire en sorte que le champ de saisie du pseudonyme contienne la dernière valeur saisie.

3. Créer un deuxième formulaire demandant un pseudonyme, un prénom, un nom, un âge et une ville, et ajoutant dans le tableau `$personnes` une nouvelle personne dont les valeurs sont celles données par le formulaire. Ajouter plusieurs personnes et expliquer alors le problème. (Afficher le tableau en entier pour mieux voir le problème.) Comment remédier à ceci (réponse sans code car les connaissances nécessaires à la réponse n'ont pas encore été vues en cours) ?

Exercice 8 : Contenu HTML protégé par mot de passe

Le but de cet exercice est de créer du contenu protégé par un mot de passe (cet exercice provient du site *openclassroom*).

1. On suppose dans un premier temps qu'il n'y a qu'une seule page Web dont le contenu est protégé. Pour cela, choisir un mot de passe (par exemple : kangourou). Créer un formulaire permettant de saisir un mot de passe. Le contenu protégé doit alors s'afficher uniquement si le mot de passe est correct. Dans le cas contraire, le formulaire doit de nouveau s'afficher.

2. On suppose maintenant qu'il y a plusieurs pages (site Web) dont le contenu doit être protégé par un mot de passe. Télécharger l'archive. Les 3 fichiers de cette archive doivent être protégés par un mot de passe. De plus, l'utilisateur doit se connecter une seule fois. Dès qu'il est connecté, il peut avoir accès à toutes les pages.

   Indication : Il est nécessaire d'utiliser les sessions. Créer le fichier `securite.php` et l'inclure en début de chaque fichier à protéger. Le fichier `securite.php` doit activer les sessions. Il doit ensuite tester si la variable `$_SESSION` contient une clé `connecte`. Si ce n'est pas le cas, le formulaire doit être affiché puis le script terminé. Dans le cas contraire, la page doit s'afficher normalement. Le fichier `securite.php` doit également tester si le formulaire a été soumis avec le bon mot de passe pour mettre à jour la variable `$_SESSION`.

   Notes :

   - Pour terminer un script, il est possible d'utiliser la fonction `exit`.
   - La variable `$_SERVER["PHP_SELF"]` contient le nom du fichier appelé dans l'url. Il peut être utilisé comme valeur du paramètre `action` de la balise `form` pour que le même fichier soit appelé lors de la soumission du formulaire.

3. Donner la possibilité à l'utilisateur de se déconnecter sur n'importe quelle page. Pour cela, créer un fichier `deconnexion.php`. Ce script affchera un lien hypertexte sur le fichier appelé avec le paramètre `action` de valeur `deconnexion`. Inclure ce fichier dans `page1.php`, `page2.php` et `page3.php`. Dans le fichier `securite.php`, faire en sorte que lors d'un clic sur ce lien hypertexte, l'utilisateur soit déconnecté (il doit saisir à nouveau son mot de passe).

Exercice 9 : TODO liste

L'objectif de cet exercice est de programmer une page Web (`tdl.php`) permettant de gérer une TODO liste (liste de tâches à réaliser). Chaque utilisateur se connectant sur la page Web doit pouvoir gérer sa propre TODO liste : créer des tâches, en supprimer et les sauvegarder. L'ajout d'une tâche se fera à l'aide d'un formulaire et la suppression d'une tâche se fera grâce à un lien hypertexte. La sauvegarde de la TODO liste se fera grâce aux cookies. La TODO liste doit être sauvegardée dans les sessions pour être accessible à chaque appel du script. L'instance de la classe `TODOList` sera stockée dans la variable `$_SESSION["tdl"]` .

Important : Cet exercice utilise la classe `TODOList` définie dans l'exercice 6. Il faut donc avoir terminé cet exercice avant de commencer celui-ci.

Important : Comme l'objet `TODOList` est sauvegardé dans la session, il est impératif d'inclure la définition de la classe avant d'activer les sessions. Le fichier `tdl.php` doit commencer par :

```php
require_once "TODOList.php"; // avant session_start car on stocke un objet TODO
session_start();
```

1. Créer le script `tdl.php` . Celui-ci doit :
   - vérifier que la TODO liste est existe dans la session. Si ce n'est pas le cas, il initialise une TODO liste vide qu'il sauvegarde dans la session ;
   - vérifier s'il existe dans l'url un paramètre `task` . Si ce paramètre existe, alors le script ajoute à la TODO liste une tâche correspondant à la valeur du paramètre `task` ;
   - afficher la TODO liste.

2. Créer un formulaire facilitant l'ajout d'une tâche dans la TODO liste.

3. Modifier la méthode `get_html` de la classe TODO liste pour que chaque item soit un hyperlien sur le script `tdl.php` avec le numéro de la tâche comme valeur du paramètre `rm` .

4. Modifier le script `tdl.php` pour permettre la suppression des tâches dans la TODO liste lors d'un clic sur un hyperlien d'une tâche.

5. Définir les méthodes `get_representation` et `set_representation` dans la classe `TODOList` . La première méthode retournera une chaîne de caractères contenant les différentes tâches séparées par `"///"` (on suppose qu'aucune tâche contient `"///"`). La deuxième méthode chargera la liste des tâches (ie, modifiera le tableau `to_dos`) en fonction de la représentation passée en paramètre.
6. Modifier le script `tdl.php` pour permettre de sauvegarder la représentation de la TODO liste dans un cookie lors d'un clic sur un hyperlien (à ajouter dans le script). Modifier l'initialisation de la TODO liste (lorsqu'elle n'est pas dans la session) pour que celle-ci charge la représentation dans le cookie si celui-ci existe. Ajouter un lien supprimant `$_SESSION["tdl"]` pour vérifier que la sauvegarde via le cookie fonctionne.

Exercice 10 : Questionnaire à choix multiples

1. Le but de cet exercice est de faire une première page Web contenant des énigmes ou questions avec plusieurs réponses possibles (une seule d'entre elles étant correcte). L'utilisateur doit alors répondre aux différentes questions et le site affiche alors le score obtenu par l'utilisateur.

2. Ajouter sur la page d'accueil un formulaire pour que l'utilisateur puisse saisir son nom. Cette donnée sera utilisée par la suite pour personnaliser l'affichage du site.

3. Stocker, à l’aide d’un cookie le meilleur score obtenu par l’utilisateur. Dans la page du score, afficher ce meilleur score en plus du score obtenu à l’instant par l’utilisateur.