

[IMAC S2] Programmation Back-end

TD 02 : Formulaires et PHP Objet

Travaux Dirigés

Pascale Ho - 2018 - Ingénieur IMAC

Avant de commencer

<https://www.php-fig.org/psr/>

Si vous avez lu les compléments du TD 01, vous savez qu'il existe des "règles" de syntaxe pour que le code soit plus lisible (PSR). À partir de maintenant **je vous demande de respecter la norme PSR** (ce sera donc pris en compte dans la notation). En vrai je ne vais pas vous demander de suivre toutes les règles, mais voici la liste des règles que vous allez devoir respecter (**PSR-1** et quelques normes **PSR-2**) :

PSR-1 : Standard de base

- On encapsule le code php **UNIQUEMENT** avec la balise <?php ?>. (en vrai le standard autorise aussi une autre balise mais je trouve ça sale)
- Les fichiers PHP **DOIVENT** être encodés UTF-8 (sans BOM).
- On **DEVRAIT** utiliser un fichier pour déclarer des classes/fonctions/constantes/..., un fichier pour les effets secondaires (rendu, configuration, ...), un même fichier **NE DEVRAIT PAS** faire les deux.
- Les noms de classe **DOIVENT** être en **StudlyCaps** (première lettre des termes en majuscule, pas d'espaces)
- Les constantes de classe **DOIVENT** être déclarées en majuscules avec des underscores _ comme séparateurs.
- Concernant les noms des attributs de classe, vous **POUVEZ** utiliser le **camelCase** (premier terme en minuscules, premières lettres des termes suivants en majuscule, pas d'espaces), ou avec l'utilisation des underscores. Mais vous **DEVEZ** garder la même syntaxe pour tous vos attributs.
- Les noms des méthodes **DOIVENT** être en **camelCase**.
- **PAS OBLIGÉ** : les namespaces et les classes DOIVENT suivre un PSR "d'autoloading" (PSR-4) :
<https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-4-autoloader.md>

PSR-2 : Guide de programmation

Ici il n'y a que les règles que je vous demande de suivre. Il en existe d'autres ici :

<https://www.php-fig.org/psr/psr-2/> mais je ne vous demande pas de les appliquer.

- Les mots-clés PHP (class, for, if, new, public, require, ...) et que true, false et null **DOIVENT** être en minuscules.
- Il **DOIT** y avoir un ligne vide après la déclaration des namespaces, et il **DOIT** y avoir une ligne vide après le bloc de la déclaration des use.
- On indique **TOUT LE TEMPS** la visibilité (public, private, protected) aux attributs et aux méthodes de classe.
- Il **NE DOIT PAS** y avoir d'espace entre le nom de la méthode et la parenthèse ouvrante et il **NE DOIT PAS** y avoir d'espace avant la parenthèse fermante.
- S'il y a plusieurs arguments de méthode de classe, il **DOIT** y avoir un espace après chaque virgule.
- Les arguments par défaut **DOIVENT** être à la fin de la liste.
- Les mots-clés abstract et final **DOIVENT** précéder la visibilité.
- Lorsqu'on appelle une méthode ou une fonction, il **NE DOIT PAS** y avoir d'espaces avant et après les parenthèses. Dans la liste des arguments, il **DOIT** y avoir un espace après chaque virgule.
- Les structures de contrôle (if, for, while, ...) **DOIVENT** avoir un espace après le mot-clé de la structure. Il **NE DOIT PAS** y avoir d'espaces avant et après les parenthèses.

Norme PHP-Hisu : Recommandations

Ouais non le PHP-Hisu, ça n'a jamais existé, c'est du bullshit. En fait je vous donne quelques recommandations pour une meilleure lisibilité dans votre code (c'est ma façon de coder) :

- Dans votre fichier PHP, commencez par faire les include/require, (puis les namespaces,) ensuite la déclaration de constantes, la déclaration des fonctions, la déclaration de classes. Comme indiqué dans le PSR-1, l'appel de ces constantes, fonctions et classes se font dans un autre fichier.
- Faîtes en sorte de n'utiliser très peu d'echo dans votre fichier (genre faîtes un echo sur une main variable qui gèrera l'affichage principal).
- Les fichiers déclarant les classes sont de la forme **nom.class.php** .
- Essayez de commenter au maximum.

#01 Formulaires

Table de multiplication

Écrivez un (seul) script qui saisit dans un champ texte un entier et qui affiche la table de multiplication de base 10 de cet entier lors de l'envoi (vérifiez que la valeur soit bien un entier)

Par exemple, si je mets 5 dans mon formulaire, cela devrait m'afficher la table de multiplication de 5 (5x1, 5x2, ... 5x10)

Choix Multiples

Dans un fichier formulaireFruits.html, créez un formulaire avec des checkbox (cases à cocher) de fruits.

```
<form method="POST" action="afficheFruits.php">
...
</form>
```

Dans un fichier afficheFruits.php, faites en sorte d'afficher les fruits qui ont été cochés dans le formulaire. Affichez un message de ce type :

```
J'adore les pommes, les oranges, les bananes et les fraises.
```

Évidemment faites un affichage différent s'il n'y a aucune case de cochée par exemple.

#02 Formulaires et PHP Objet

Une classe PHP

1. Créez un fichier Personne.class.php avec le contenu suivant :

```
<?php
class Personne {

    private $prenom;
    private $nom;
    private $age;
    private $ville;

    // constructeur
```

```
public function __construct() {
    // -- à compléter
}

// getters et setters -- à compléter

// méthode d'affichage -- à compléter
}
```

2. Complétez le constructeur en renseignant comme paramètre un prénom, un nom, un âge et une ville. (N'oubliez pas de faire un cast)

Bonus : Faîtes en sorte que la première lettre des prénoms et des noms soit en majuscule, et pas les autres. Attention aux noms composés. Ces fonctions peuvent vous aider : **strtolower, ucfirst, ucwords**

3. Créez des getters et setters pour le prénom, le nom, l'âge et la ville.

Bonus :

Vous avez remarqué mais déjà en faisant 4 getters et setters cela commence à devenir fastidieux.

Si vous n'avez pas d'IDE qui permet l'intégration automatique des getters/setters, vous pouvez créer des getters génériques **get(\$nomAttribut)** qui renvoient l'attribut de nom \$nomAttribut. Pour accéder à un attribut de nom \$nomAttribut de l'objet \$objet, il suffit de faire **\$objet->\$nomAttribut**. Faites aussi des setters génériques **set(\$nomAttribut, \$valeur)**.

4. Créez une méthode afficher() qui permet d'afficher une présentation de la personne comme ceci :

```
Bonsoire, je m'appelle Pascale Ho, je viens de Schlag-sur-Marne et j'ai  
25 ans.
```

Bonus : Personnalisez encore plus le message, genre en mettant "Bonjoure" en journée et "Bonsoire" en soirée (il suffit de récupérer l'heure).

Utilisation de la classe Personne

1. Créez un fichier **testPersonne.php** qui contiendra un require_once de Personne.class.php et le squelette HTML classique (<html>, <head>, <body>)

2. Créez une variable \$personne1 qui sera une instance de classe Personne. Affichez cette personne à l'aide de sa méthode afficher.

3. Faîtes de même avec une liste de personnes (un tableau d'objets Personne).

Interaction avec un formulaire

1. Créez un fichier **formulairePersonne.html** qui va contenir un formulaire avec un champ pour le prénom, le nom et l'âge et un bouton “**Envoyer**”.

```
<form method="GET" action="affichePersonne.php">  
...  
</form>
```

2. Créez un fichier **afficherPersonne.php** qui va créer un objet de classe Personne et l'afficher.

- Vérifiez d'abord qu'il y a bien toutes les informations qui ont été renseignées dans le formulaire (vérifiez que `$_GET` n'est pas vide).
- N'oubliez pas de vérifier que les informations renseignées sont bonnes (par exemple que l'âge soit correct).

#03 Fil Rouge

Rappel : Ce TD n'est pas obligatoire. Plus vous m'envoyez vos travaux plus vous pouvez avoir des **bonus sur votre note finale**. Vous avez donc plusieurs façons de me fournir vos TDs :

- soit en m'envoyant un fichier compressé (.zip, ...) contenant un répertoire avec tous les fichiers nécessaires à l'exercice,
- soit en m'envoyant un repo Git des TDs

À m'envoyer par mail (hp.pascale.ho@gmail.com) ou via messagerie 2.0 du digital des réseaux sociaux

Date limite : 05 mars à 8h30 pétantes

Le style n'est pas important ! Et si vous avez des questions, n'hésitez pas à me les poser ou à vos camarades !

Sujet

Tout d'abord j'ai modifié le fichier **data.movies.php** (correction de quelques fautes, nouvelles variables et j'ai rajouté encore plus de films, eh ouais). Prenez ce fichier et regardez le contenu. Je vous demande de **NE PAS LE MODIFIER** thanks 😊

Cette fois-ci avec nos données on va construire un formulaire de recherche. Concernant la création de classes, on s'en occupera lorsqu'on va manipuler les BDDs avec PHP.

1. Dans un fichier **search.php** construisez un formulaire de recherche avec des champs de recherche pour le titre, la date de sortie du film, le genre et une personne du cast. Le formulaire utilisera la méthode **GET** et comme url d'action **movies.php**. (Il y a une liste de genres dans data.movies.php, mais oups la liste n'est pas ordonnée alphabétiquement)

2. Avant de s'occuper de movies.php, créez un autre script php qui contiendra une fonction **searchMovies()** qui prend comme paramètre la liste des informations envoyées par le formulaire. Cette fonction doit retourner une liste de films correspondant aux critères choisies par l'utilisateur. Vérifiez que les critères de recherche sont valides.

Attention, cette fois-ci on va comparer des dates entières (jour, mois, année). Pour comparer deux dates vous pouvez sous servir de la classe **DateTime**.

3. Faîtes le rendu dans **movies.php** (vous pouvez réutiliser le travail du TD précédent) des films correspondant à la recherche. Testez la recherche avec l'URL sans passer par le formulaire. N'oubliez pas de faire un affichage quand la recherche a échoué.

4. Bonus :

- Au lieu de rechercher qu'une seule date de sortie, on va proposer dans le formulaire le choix de faire une fourchette de dates.
- Si ce n'est pas été fait, utilisez des checkbox pour le choix du genre de film.
- Plutôt que de chercher exactement le terme exact du titre ou du cast, faîtes en sorte que la fonction de recherche vérifie que la valeur recherchée est bien une portion du titre du film ou du réalisateur/réalisatrice.

5. Exercice de déclaration de classe : dans un fichier Movie.class.php, créez une classe Movie qui a comme attributs (privées) un numéro d'identifiant, un titre, une date de sortie, un genre et un nom de réalisateur/réalisatrice. Ajoutez un constructeur, des getters/setters, et une fonction renderHTML() qui retourne le rendu HTML d'un film. Testez avec un nouveau script **movie.php** (et non movies.php). Lorsqu'on lance l'url **movie.php?id=x** depuis le navigateur, avec id un numéro d'identifiant, le script doit afficher les informations du film à l'identifiant x.