Utiliser la documentation PHP et standards de codage PHP

Table des matières

I. Contexte

Durée : 1 h

Environnement de travail : Repl.it

Pré-requis : Connaître les bases de PHP

Contexte

La documentation sera notre meilleure alliée lors de nos développements.

PHP est un langage très vaste dont il est difficile de connaître toutes les possibilités, c'est pourquoi il est important de bien savoir utiliser les ressources proposées.

Nous aborderons également les standards de programmation en PHP afin de produire un code répondant aux bonnes pratiques.

II. Utiliser la documentation PHP

Objectif

- Savoir utiliser la documentation PHP

Mise en situation

Lors de nos développements, nous aurons besoin de consulter la documentation PHP afin, par exemple, de savoir à quoi correspond une fonction dans le code ou de savoir quelle fonction utiliser pour répondre à notre besoin.

Il est important d'apprendre à bien l'utiliser.

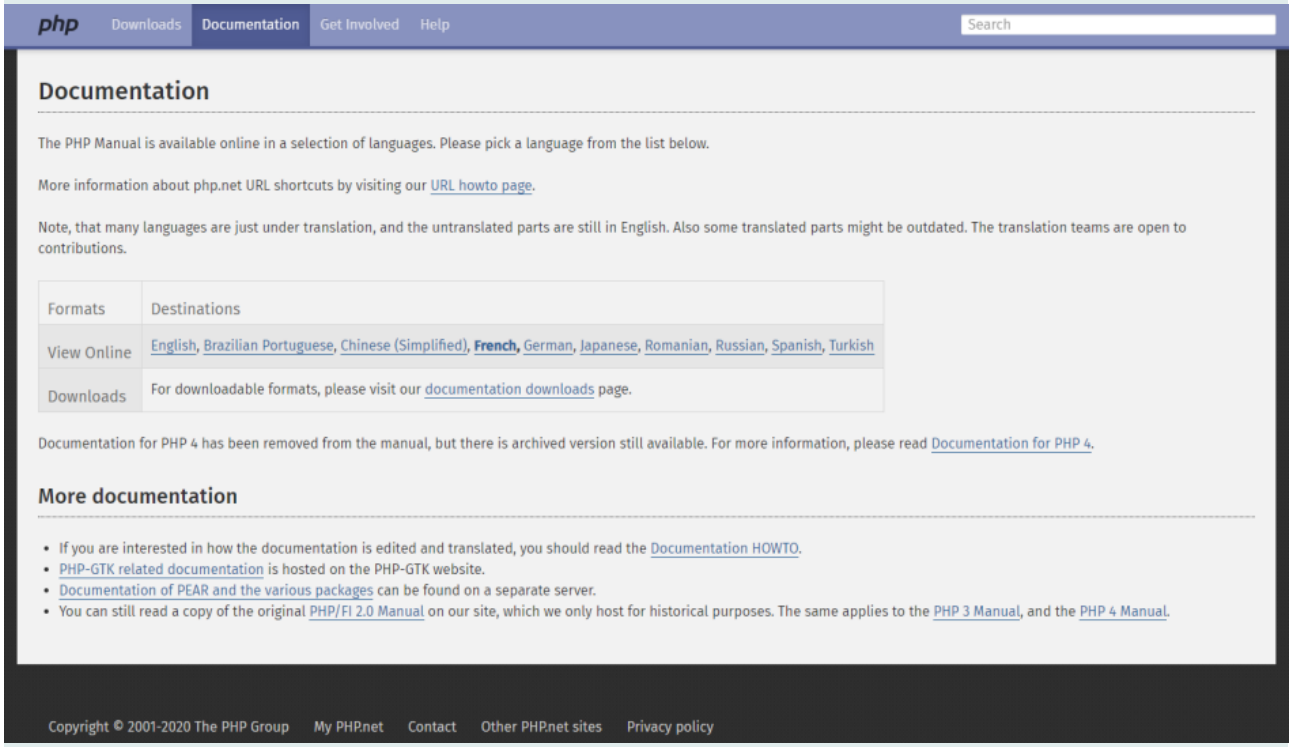
Fondamental

L'un des avantages de PHP réside dans l'importance de sa communauté et de la qualité de sa **documentation, très complète et traduite dans de nombreuses langues, dont le français**.[1]

PHP est un langage qui évolue régulièrement et sa documentation est donc mise à jour en fonction, ce qui est un gage de qualité.

La documentation aborde tout les points de PHP : l'histoire du langage[2], son installation[3], les fonctions disponibles[4], les aspects de sécurité[5], etc.

1 https://www.php.net/manual/fr/
2 https://www.php.net/manual/fr/history.php
3 https://www.php.net/manual/fr/install.php
4 https://www.php.net/manual/fr/funcref.php
5 https://www.php.net/manual/fr/security.php



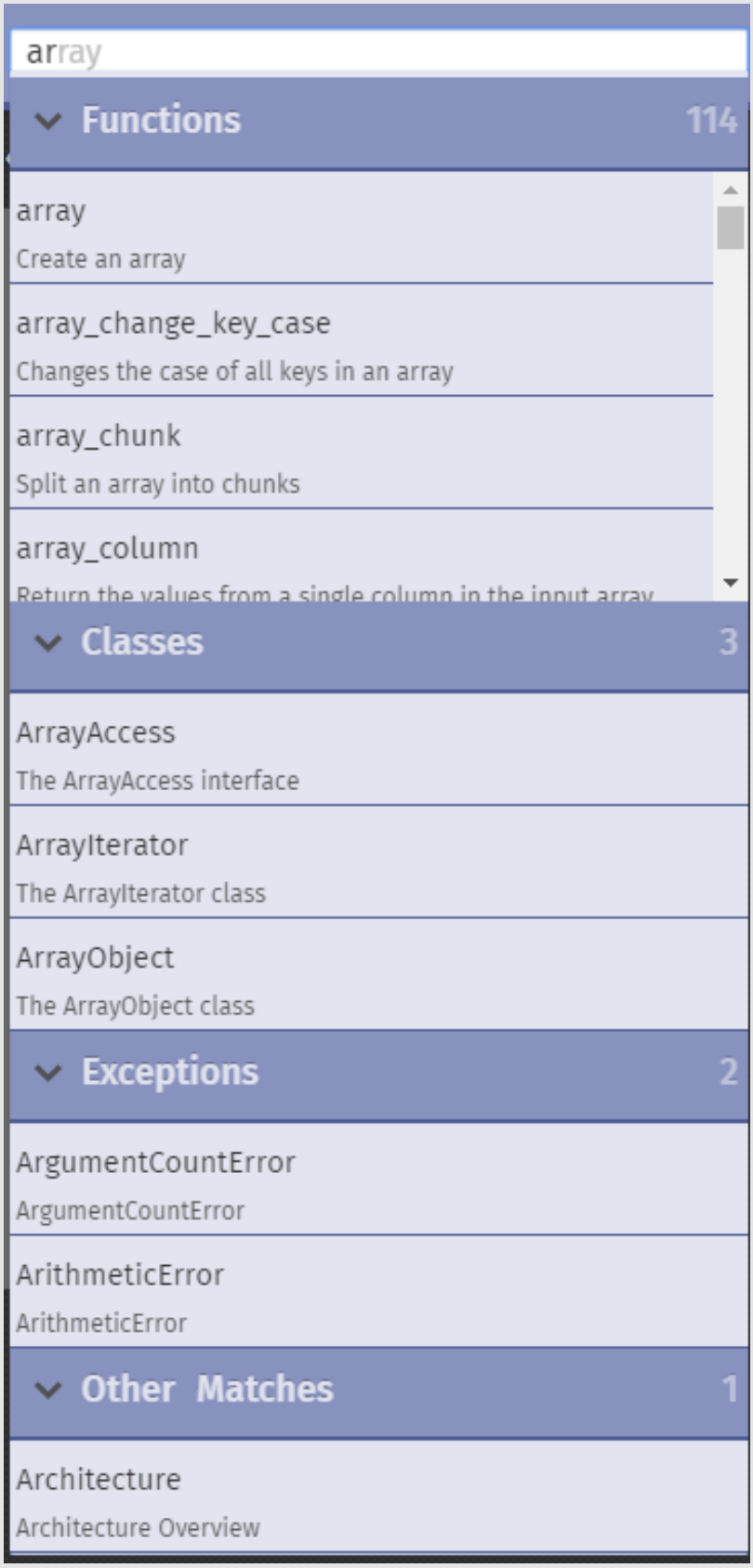
Capture écran de la documentation PHP

Méthode **Rechercher dans la documentation**

Il n'est pas toujours facile de s'y retrouver dans les différents menus de navigation.

Heureusement, **le moteur de recherche situé en haut à droite est extrêmement bien fait et performant**, c'est pourquoi nous vous encourageons à l'utiliser.

En effet, celui-ci propose de l'auto-complétion dès que nous commençons à taper quelques caractères, en regroupant les résultats par catégories (*Functions, Classes, Exceptions, Extensions, Other Matches*, etc.) : nous pouvons alors cliquer sur le résultat de notre choix ou continuer de taper pour affiner notre recherche.

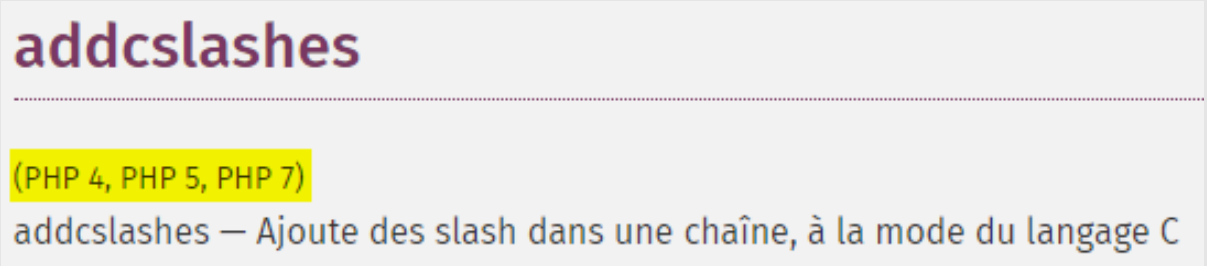


Exemple de recherche dans la documentation PHP

Méthode **Vérifier la compatibilité**

Lorsque nous consultons la documentation, il y a des points sur lesquels il faut être vigilant. Il faut notamment vérifier la version de PHP et la validité d'une fonction.

Par exemple, au-dessous du nom de la fonction `addcslashes`, nous retrouvons les **versions de PHP compatibles**, ce qui nous permet de savoir si nous pouvons l'utiliser sur notre projet.

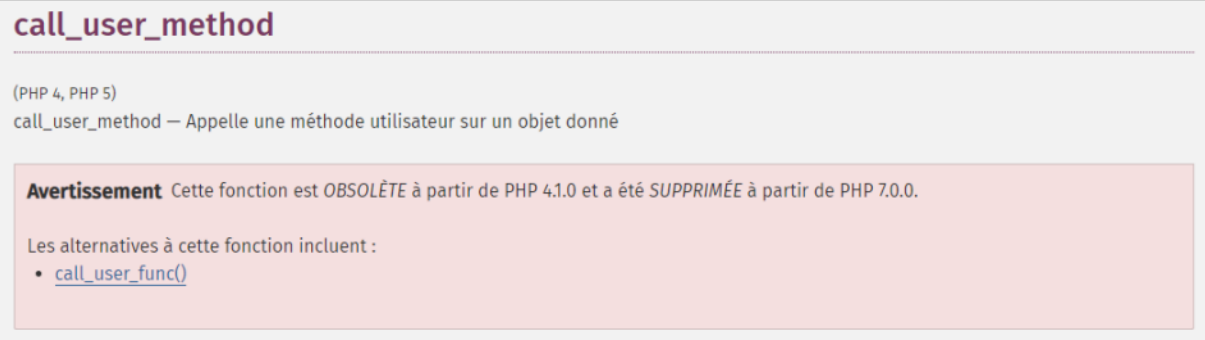


Exemple d'affichage des versions de PHP compatibles

Conseil **Repérer les éléments dépréciés**

Concernant la validité d'une fonction, il est clairement indiqué dans la documentation PHP **si celle-ci est obsolète** (encart rouge) et à partir de quelle version de PHP c'est le cas.

Il est même parfois indiqué **si celle-ci a été supprimée** et à partir de quelle version de PHP elle l'a été.



Exemple de fonction PHP obsolète et supprimée

Syntaxe **À retenir**

- Ne pas hésiter à user et abuser de la documentation PHP.
- Utiliser le moteur de recherche pour gagner du temps.
- Attention à bien vérifier la version de PHP compatible par rapport à notre version.
- Vérifier qu'il ne s'agit pas d'une fonctionnalité obsolète, voire supprimée.

Complément

PHP Documentation[1]

1 https://www.php.net/docs.php

III. Exercice : Appliquez la notion

Question 1

[solution n°1 p.19]

À l'aide de la documentation de PHP, déterminez depuis quelle version de PHP la fonction `money_format` est obsolète.

Question 2

[solution n°2 p.19]

Toujours d'après la documentation, déterminez comment cette fonction peut être remplacée.

Question 3

[solution n°3 p.19]

Enfin, identifiez quels étaient les pré-requis système pour utiliser cette fonction.

IV. PSR-0, PSR-1 et PSR-12 : syntaxe

Objectif

- Savoir ce qu'est PSR

Mise en situation

Lors d'un développement, il est important de s'assurer de produire la meilleure qualité de code possible, que ce soit dans sa construction ou dans le respect des bonnes pratiques et des standards. Pour cela, des recommandations existent et sont regroupées sous le sigle PSR.

Connaître et maîtriser les recommandations PSR va nous aider à atteindre cet objectif.

Définition **Les recommandations PSR**

PSR signifie **PHP Standards Recommendations** : il s'agit de recommandations émises pour la standardisation des concepts de programmation en PHP.

Chaque recommandation est proposée par les membres du **FIG** (*Framework Interoperability Group*) et votée selon un protocole établi.

Pour simplifier, le but du FIG est de permettre l'interopérabilité des différents frameworks en fournissant une base technique commune.

En suivant les PSR lors de nos développements, nous utilisons des pratiques et conventions communes et connues. Cela permet également à d'autres développeurs connaissant les PSR de plus facilement comprendre et reprendre le code. Ainsi, celui-ci sera plus facilement maintenable et pérenne.

Il est possible de consulter et suivre l'état des PSR sur le site du FIG : https://www.php-fig.org/psr[1]/.

Complément **La numérotation PSR**

La numérotation correspond simplement à l'ordre dans lequel sont soumises les recommandations.

Comme les recommandations passent par plusieurs étapes (brouillon, revue, acceptée, abandonnée, etc.) et que certaines recommandations sont plus complexes que d'autres, il est parfaitement possible d'avoir une PSR acceptée ayant un numéro supérieur à une PSR toujours à l'état de brouillon.

Nous allons à présent étudier certaines de ces recommandations.

1 https://www.php-fig.org/psr/

Méthode **PSR-0 / PSR-4 : Autoloading**

La PSR-0[1] aborde l'*autoloading*, mais cette PSR a été dépréciée au profit de la PSR-4[2], qui définit donc une nouvelle approche.

Cette PSR décrit une spécification pour l'auto-chargement de classes à partir de chemins de fichiers. Il est également décrit où placer les fichiers qui seront auto-chargés.

Il s'agit donc notamment de respecter une syntaxe, une convention de nommage et un chemin lorsque nous créons nos classes PHP.

Par exemple, un projet comme Composer existe grâce à des conventions de nommage.

FULLY QUALIFIED CLASS NAME	NAMESPACE PREFIX	BASE DIRECTORY	RESULTING FILE PATH
\Acme\Log\Writer\File_Writer	Acme\Log\Writer	./acme-log-writer/lib/	./acme-log-writer/lib/File_Writer.php
\Aura\Web\Response\Status	Aura\Web	/path/to/aura-web/src/	/path/to/aura-web/src/Response/Status.php
\Symfony\Core\Request	Symfony\Core	./vendor/Symfony/Core/	./vendor/Symfony/Core/Request.php
\Zend\Acl	Zend	/usr/includes/Zend/	/usr/includes/Zend/Acl.php

PSR-4 Exemple / Source : php-fig.org

Méthode **PSR-1: Basic Coding Standard**

Cette PSR[3] spécifie ce qui doit être considéré comme les éléments de codage standards qui sont nécessaires pour assurer un niveau élevé d'interopérabilité technique entre les codes PHP partagés, par exemple :

- Tout fichier PHP doit contenir `<?php` ou `<?=`
- Les noms des classes doivent être en **PascalCase**, c'est-à-dire avoir une majuscule à chaque mot (`class AuthTokenController`)
- Les constantes de classe doivent être déclarées en majuscules avec des séparateurs de soulignement (`public const ROLE_DEFAULT = ['ROLE_READER'];`)
- Les noms de méthodes doivent être déclarés en **camelCase**, c'est-à-dire que la première lettre est en minuscule, puis il y a une majuscule à chaque mot (`public function findOrCreateUser()`)

Méthode **PSR-12 : Extended Coding Style**

La recommandation PSR-12[4] liste des règles et attentes communes sur la façon de formater le code PHP.

Il s'agit d'un ensemble de lignes directrices à utiliser pour tous nos projets, ce qui facilite également la collaboration avec d'autres développeurs.

Il est par exemple indiqué :

- De suivre la PSR-1
- De ne pas mettre de balise fermante `?>` si le fichier ne contient que du PHP
- Il ne doit pas y avoir d'espace blanc à la fin des lignes
- L'indentation du code doit être de 4 espaces pour chaque niveau, et la tabulation ne doit pas être utilisée
- Toute accolade de fermeture ne doit pas être suivie d'un commentaire ni d'une déclaration sur la même ligne

1 https://www.php-fig.org/psr/psr-0/
2 https://www.php-fig.org/psr/psr-4/
3 https://www.php-fig.org/psr/psr-1/
4 https://www.php-fig.org/psr/psr-12/

- Pour une classe, l'accolade d'ouverture doit aller sur sa propre ligne et l'accolade de fermeture pour la classe doit aller sur la ligne suivante, après le corps de la classe
- De façon générale, les accolades d'ouverture doivent être sur leur propre ligne et ne doivent pas être précédées ni suivies d'une ligne blanche
- De façon générale, les accolades de fermeture doivent être sur leur propre ligne et ne doivent pas être précédées d'une ligne blanche
- Les noms de méthodes et de fonctions ne doivent pas être déclarés avec un espace après le nom de la méthode, et il ne doit pas y avoir d'espace après la parenthèse ouvrante, ni avant la parenthèse fermante

Exemple **Quelques exemples issus de la PSR-12 / Source : php-fig.org**

```php
<?php

namespace Vendor\Package;

class ClassName
{
    public $foo = null;
    public static int $bar = 0;
}
```

```php
<?php

namespace Vendor\Package;

class ClassName
{
    public function fooBarBaz($arg1, &$arg2, $arg3 = [])
    {
        // method body
    }
}
```

```php
<?php

declare(strict_types=1);

namespace Vendor\Package;

class ReturnTypeVariations
{
    public function functionName(int $arg1, $arg2): string
    {
        return 'foo';
    }

    public function anotherFunction(
        string $foo,
        string $bar,
        int $baz
    ): string {
        return 'foo';
    }
}
```

Exemple **Structure de type if**

Attention au placement des parenthèses, des espaces et des accolades : nous remarquons aussi que `else` et `elseif` sont sur la même ligne que l'accolade de fermeture précédente.

```php
<?php

if ($expr1) {
    // if body
} elseif ($expr2) {
    // elseif body
} else {
    // else body;
}
```

Exemple **Structure de type switch**

Toujours veiller au bon emplacement des accolades et parenthèses.

Notons également les indentations des `case` (1 niveau par rapport au `switch`) et des `break` (1 niveau par rapport au `case`). S'il n'y a pas de `break`, on le précise avec un commentaire : `// no break`.

```php
<?php

switch ($expr) {
    case 0:
        echo 'First case, with a break';
        break;
    case 1:
        echo 'Second case, which falls through';
        // no break
    case 2:
    case 3:
    case 4:
        echo 'Third case, return instead of break';
        return;
    default:
        echo 'Default case';
        break;
}
```

Exemple **Structure de type while, for, foreach**

Une attention particulière doit encore être portée sur l'emplacement des accolades et parenthèses.

```php
<?php

while ($expr) {
    // structure body
}

for ($i = 0; $i < 10; $i++) {
    // for body
}

foreach ($iterable as $key => $value) {
    // foreach body
}
```

Complément **Quelques spécificités**

Nous allons évoquer quelques spécificités hors PSR, que nous rencontrons souvent dans le milieu professionnel :

- Coder en anglais (noms des variables, fonctions, classes, fichiers, etc.)
- Toujours donner un nom clair aux variables/fonctions (`$password`, `public function createUser()`)
- Commenter son code (en anglais également)
- Toujours utiliser l'égalité/inégalité stricte, triple égal au lieu de double égal, et point d'exclamation et double égal au lieu de point d'exclamation et signe égal (`if ($isValid === true)` ou `if ($isValid !== true)`)
- Éviter d'utiliser des variables globales
- Pour les chaînes de caractères, il est préférable d'utiliser les simples quotes (`$message = 'Bienvenue sur notre site';`)

Conseil **Les PSR au quotidien**

Il est important de maîtriser au mieux les normes PSR afin de fournir un code d'une qualité optimale.

Néanmoins, l'Homme n'est pas une machine. Ces normes évoluent quotidiennement, et peuvent tout simplement être compliquées à retenir ou à appliquer de façon systématique, même si de nombreux automatismes arriveront très vite.

Au quotidien, certains outils seront des alliés essentiels afin d'appliquer au mieux ces normes :

- La plupart des éditeurs de code proposent des options de "re-formatage du code" afin de parer les oublis. Ces options sont puissantes et complètes, il ne faut pas hésiter à en faire usage.
- Il est de plus en plus courant de retrouver, sur un projet, la présence de scripts permettant l'analyse du respect des standards, ainsi que la correction automatique et systématique au besoin. Ces scripts permettent de conserver une qualité de code constante. L'un des plus connu est par exemple PHP-CS-Fixer[1], mais il en existe d'autres, la qualité du code étant une préoccupation importante dans beaucoup d'équipes.

Syntaxe **À retenir**

- **PSR signifie PHP Standards Recommendations** : il s'agit de recommandations émises pour la standardisation des concepts de programmation en PHP.
- Il est conseillé de les consulter régulièrement afin de rester informé des nouveautés ou mises à jour.
- Il est également important de développer en respectant les PSR, afin de respecter les standards et bonnes pratiques.

Complément

PHP Standards Recommendations[2]

PHP Standard Recommendation (Wikipédia)[3]

1 https://github.com/FriendsOfPHP/PHP-CS-Fixer
2 https://www.php-fig.org/psr/
3 https://en.wikipedia.org/wiki/PHP_Standard_Recommendation

V. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

1

Question

[solution n°4 p.19]

Vous disposez du code ci-dessous.

Celui-ci est fonctionnel, mais ne respecte aucune convention.

Modifiez ce code de sorte à ce qu'il prenne en compte les bonnes pratiques et conventions abordées dans le chapitre précédent, à savoir :

- Les différents éléments doivent être indentés correctement
- Le nom d'une classe est écrit en PascalCase
- Une constante doit être déclarée en majuscules
- Les mots réservés sont en minuscules
- Le nom d'une fonction doit respecter la casse camelCase, comme les noms de variables
- Les accolades sont sur une ligne à part
- On évite aussi les sauts de ligne inutiles

```php
<?php

class foo_bar
{
PUBLIC const my_const = 0;

PUBLIC function Sample_function ($SAMPLEVAR)
{
SWITCH ($SAMPLEVAR)

{

    CASE 1:
        echo 'Cas n°1';
        break;
    CASE 2:
        echo 'Cas n°2';
        break;
    DEFAULT:
        echo 'Cas par défaut';
        break;
}
}}

?>
```

1 https://repl.it/

VI. Auto-évaluation

A. Exercice final

Exercice 1

[solution n°5 p.20]

Exercice

Exercice

Sous quel acronyme regroupe-t-on couramment les recommandations et standards de code ?

Exercice

Exercice

Exercice

Exercice

Exercice

Exercice

B. Exercice : Défi

Dans le but d'effectuer une montée de version PHP de sa version 5.6 sur l'une des applications dont vous avez la charge, vous êtes chargé d'effectuer des modifications au niveau du code pour utiliser les dernières fonctionnalités de PHP 7.4.

```php
<?php

session_start();

class user{

    // string
    public $name;

    // entier
    public $age;

    // string
    public $defaultRole;

    public $favoritePages = [];

    public function __construct($name, $age){
        $this->name = $name;
        $this->age = $age;

        $this->defaultRole = isset($_SESSION['DEFAULT_ROLE']) ? $_SESSION['DEFAULT_ROLE'] : 'ROLE_DEFAULT';
    }

    public function add_to_favorites($link){

        if (filter_var($link, FILTER_VALIDATE_URL) && !in_array($link, $this->favoritePages)) {
            array_push($this->favoritePages, $link);

            return true;
        }

        return false;}

    public function remove_from_favorites($link){
        if (filter_var($link, FILTER_VALIDATE_URL) && in_array($link, $this->favoritePages)) {
            unset($this->favoritePages[array_search($link, $this->favoritePages)]);

            return true;
        }

        return false;
    }
}

$user = new user('Eric', 45);

// true
echo $user->add_to_favorites('https://www.google.com/');
```

```php
// false
echo $user->add_to_favorites('https://www.google.com/');

// true
echo $user->remove_from_favorites('https://www.google.com/');

// ROLE_DEFAULT
echo $user->defaultRole;

$_SESSION['DEFAULT_ROLE'] = 'ROLE_USER';
$user2 = new user('Marie', 40);

// ROLE_USER
echo $user2->defaultRole;

unset($_SESSION['DEFAULT_ROLE']);

?>

```

Attention, avant d'effectuer ce défi sur repl.it, vérifiez la version de PHP grâce à la commande `php -v`.

Si vous ne disposez pas de la version 7.4, vous pouvez vous contenter de modifier le code pour qu'il soit compatible jusqu'en version 7.2, ou exécuter votre code sur un autre simulateur tel que http://sandbox.onlinephpfunctions.com/.

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

[1]

Question

[solution n°6 p.20]

À l'aide de la documentation de PHP, identifiez les dernières évolutions de PHP afin de les intégrer dans cette classe.

Profitez-en également pour vous assurer que toutes les règles PSR sont respectées et appliquez-les au besoin.

```

```

Indice :

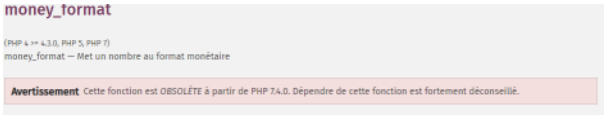
Avez-vous examiné les notes de montée de version[2] de PHP 5.6 à PHP 7 ?

Solutions des exercices

1 https://repl.it/
2 https://www.php.net/manual/fr/migration70.new-features.php

p. 7 Solution n°1

D'après la documentation[1], la fonction est obsolète depuis la version 7.4.0 de PHP.



p. 7 Solution n°2

La fonction peut être remplacée par l'utilisation de `NumberFormatter::formatCurrency`[2].

p. 7 Solution n°3

La fonction `money_format()` est uniquement définie si le système a les capacités `strfmon`.

p. 12 Solution n°4

```php
<?php
// Les différents éléments doivent être indentés correctement

// Le nom d'une classe est écrit en PascalCase
class FooBar
{
    // Une constante doit être déclarée en majuscules
    // Les mots réservés sont en minuscules
    public const MY_CONST = 0;

    // le nom d'une fonction doit respecter la casse camelCase, comme les noms de variables
    // Pas d'espace entre le nom d'une fonction et ses paramètres
    public function sampleFunction($sampleVar)
    {
        // Les accolades sont sur une ligne à part
        // On évite aussi les sauts de ligne inutiles
        switch ($sampleVar) {
            case 1:
                echo 'Cas n°1';
                break;
            case 2:
                echo 'Cas n°2';
```

1 https://www.php.net/manual/fr/function.money-format.php
2 https://www.php.net/manual/fr/numberformatter.formatcurrency.php

```
                break;
            default:
                echo 'Cas par défaut';
                break;
        }
    }
}

```

Exercice p. 13 Solution n°5

Exercice

Exercice

Sous quel acronyme regroupe-t-on couramment les recommandations et standards de code ?

PSR

Exercice

Exercice

Exercice

Exercice

Exercice

Exercice

Exercice

p. 17 Solution n°6

Version compatible PHP 7.4 :

```php
<?php

session_start();

class User
{
    // Depuis PHP 7.4, il est possible de typer les propriétés d'une classe
    public string $name;

    public int $age;

    public string $defaultRole;

    public array $favoritePages = [];

    // Les paramètres d'entrée d'une fonction peuvent être typés avec les types string / int depuis PHP 7.0
    public function __construct(string $name, int $age)
    {
        $this->name = $name;
        $this->age = $age;

```

```php
        // Depuis PHP 7.0, ce nouvel opérateur permet d'éviter la répétition dont nous disposions auparavant
        $this->defaultRole = $_SESSION['DEFAULT_ROLE'] ?? 'ROLE_DEFAULT';
    }

    // Les paramètres de sortie d'une fonction peuvent être typés depuis PHP 7.0
    public function addToFavorites(string $link): bool
    {

        if (filter_var($link, FILTER_VALIDATE_URL) && !in_array($link, $this->favoritePages)) {
            array_push($this->favoritePages, $link);

            return true;
        }

        return false;
    }

    public function removeFromFavorites(string $link): bool
    {
        if (filter_var($link, FILTER_VALIDATE_URL) && in_array($link, $this->favoritePages)) {
            unset($this->favoritePages[array_search($link, $this->favoritePages)]);

            return true;
        }

        return false;
    }
}

$user = new User('Eric', 45);

// true
echo $user->addToFavorites('https://www.google.com/');
// false
echo $user->addToFavorites('https://www.google.com/');

// true
echo $user->removeFromFavorites('https://www.google.com/');

// ROLE_DEFAULT
echo $user->defaultRole;

$_SESSION['DEFAULT_ROLE'] = 'ROLE_USER';
$user2 = new User('Marie', 40);

// ROLE_USER
echo $user2->defaultRole;

unset($_SESSION['DEFAULT_ROLE']);

```

Version compatible PHP 7.2 :

```php
<?php

session_start();

class User
{
    public $name;

    public $age;

    public $defaultRole;

    public $favoritePages = [];

    // Les paramètres d'entrée d'une fonction peuvent être typés avec les types string / int depuis PHP 7.0
    public function __construct(string $name, int $age)
    {
        $this->name = $name;
        $this->age = $age;

        // Depuis PHP 7.0 ce nouvel opérateur permet d'éviter la répétition dont nous disposions auparavant :
        $this->defaultRole = $_SESSION['DEFAULT_ROLE'] ?? 'ROLE_DEFAULT';
    }

    // Les paramètres de sortie d'une fonction peuvent être typés depuis PHP 7.0
    public function addToFavorites(string $link): bool
    {

        if (filter_var($link, FILTER_VALIDATE_URL) && !in_array($link, $this->favoritePages)) {
            array_push($this->favoritePages, $link);

            return true;
        }

        return false;
    }

    public function removeFromFavorites(string $link): bool
    {
        if (filter_var($link, FILTER_VALIDATE_URL) && in_array($link, $this->favoritePages)) {
            unset($this->favoritePages[array_search($link, $this->favoritePages)]);

            return true;
        }

        return false;
    }
}

$user = new User('Eric', 45);

// true
echo $user->addToFavorites('https://www.google.com/');
// false
```

```php
echo $user->addToFavorites('https://www.google.com/');

// true
echo $user->removeFromFavorites('https://www.google.com/');

// ROLE_DEFAULT
echo $user->defaultRole;

$_SESSION['DEFAULT_ROLE'] = 'ROLE_USER';
$user2 = new User('Marie', 40);

// ROLE_USER
echo $user2->defaultRole;

unset($_SESSION['DEFAULT_ROLE']);

```