PHP

PHP: Hypertext Preprocessor

Historique & Chiffres

Historique

1994 : Création par Rasmus Lerdorf pour gérer son CV en ligne
1995 : Rasmus Lerdorf open source le projet
1997 : Sortie de PHP 3 suite à la réécriture du coeur de PHP par 2 étudiants (Zeev et Andi)
1999 : Sortie de PHP 4 suite à l'introduction du ZEND Engine, connu pour avoir une configuration par défaut peu sécurisée (corrigé en 4.2)
2004 : Sortie de PHP 5, allant plus loin dans l'implémentation du paradigme de Programmation Orienté Objet
2010 : Abandon de PHP 6

Historique

2012 : Sortie de PHP 5.4
2013 : Sortie de PHP 5.5
2014 : Sortie de PHP 5.6
2015 : Sortie de PHP 7.0
2016 : Sortie de PHP 7.1
2017 : Sortie de PHP 7.2 : En fin de vie
2018 : Sortie de PHP 7.3 : Activement supporté jusqu'en 2021
2019 : Sortie de PHP 7.4 : Activement supporté jusqu'en 2022
2020 : Sortie de PHP 8

source : https://www.php.net/supported-versions.php

Chiffres

En 2013, 244 millions de sites utilisent PHP

En 2016, PHP a 82% de part de marché coté serveur

source : wikipedia

15, c'est le nombre d'antennes de l'AFUP en France.
Antenne Bordelaise : https://www.meetup.com/fr-FR/Bordeaux-PHP-Meetup/

10, c'est le nombre de langues dans laquelle la documentation est traduite.
La documentation est complète, à jour et contient beaucoup d'exemples.
C'est votre principale source d'information : https://www.php.net/docs.php

Installation WAMP + cmder (Windows)

Télécharger WampServer 64 bits ici : http://www.wampserver.com/

Lancer le fichier et suivre les instructions d'installations.

Pour avoir un terminal ressemblant à ceux sur Unix, installer cmder ici : https://cmder.net

Download Mini devrait suffire.

Principaux Sites, applications et framework

Sites, applications et framework importants utilisant PHP :

- Wikipedia / Facebook / Yahoo / Flickr / Tumblr
- TF1 / France2 / M6
- Wordpress / Prestashop / Magento
- Symfony / Laravel

Installation

- Installation WAMP + cmder (Windows)
- Installation MAMP (Mac)
- Installation Ubuntu (Linux)
- Installation PhpStorm
- Installation Netbeans
- Serveur de développement

Installation MAMP (Mac)

Télécharger MAMP ici : https://www.mamp.info/en/downloads/

Suivre les instructions d'installation.

Une fois l'installation terminée, lancer un terminal :

```
php -v
```

Installation Ubuntu (Linux)

Dans un terminal : `sudo apt install php`

Mettez votre mot de passe.

Pour vérifier que l'installation a réussi :

```
php -v
```

Installation PHPStorm

De loin l'IDE le plus performant avec PHP.

Il est payant mais offre une version d'essai de 1 mois.

Vous pouvez le télécharger ici : https://www.jetbrains.com/phpstorm/download

Nécessite peut-être d'avoir java d'installé

Installation Netbeans

Gratuit et vient avec le support de PHP intégré.

Vous pouvez le télécharger ici :
https://netbeans.apache.org/download/index.html

Nécessite probablement d'avoir java d'installé

Serveur de développement

Pour lancer un serveur de développement, se placer dans le dossier de votre projet php et lancer :

```
php -S localhost:8000
```

Depuis un navigateur, vous pouvez aller sur la page `http://localhost:8000`
Par défaut, cela essaiera d'ouvrir un fichier `index.php`.

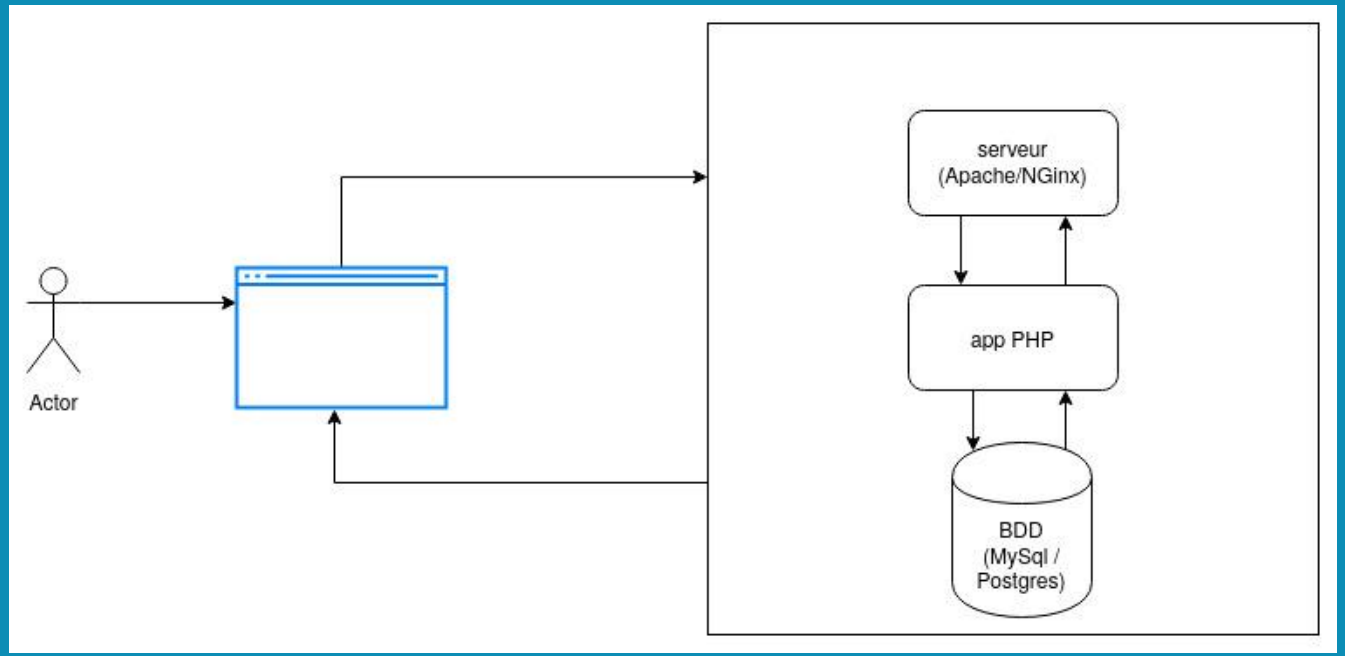
Vous pouvez aussi viser un fichier en indiquant son nom :

```
http://localhost:8000/mon_fichier.php
```

Bonnes Pratiques

- Back-End language
- PHP-FIG
  - PSR-1: Basic Coding Standard
  - PSR-12: Extended Coding Style

Back-End Language



PHP-FIG

- PHP-FIG
  - PSR (PHP Standard Recommandation)
- Définit un ensemble de bonnes pratiques dans toutes la communauté PHP

-> Règles pour écrire le code (coding style)
-> Règles pour namespaces
-> Règles pour autoload de classe
-> ...

PHP-CS-fixer Permet de formater automatiquement le code.

PHP-FIG

```php
<?php

namespace Vendor\Package;

class ClassName
{
    const VERSION = '1.0';

    public $foo = null;
    protected static int $fooVar;

    public function aVeryLongMethodName(
        ClassTypeHint $argNumber1,
        ?int &$arg2,
        array $arg3 = [],
        &...$parts
    ): string {
        return 'foo';
    }
}
```

Bases de PHP

- "Bonjour Bordeaux"
- commentaires
- constantes & variables
- tableaux
- transtypage
- opérateurs
- structures de contrôles
- fonctions, closures
- création d'une page HTML
- formulaires : `$_GET` et `$_POST`
- Cookies
- Sessions
- `require` et `include`

"Bonjour Bordeaux"

Créer un dossier `workspace` où bon vous semble.

Dans ce dossier `workspace`, créer un fichier `index.php` et écrire à l'intérieur :

```
<?php
echo "Bonjour Bordeaux";
```

"Bonjour Bordeaux"

Dans un terminal, se placer dans le dossier `workspace` et lancer la commande
`php index.php`

Au même endroit dans le terminal, lancer la commande :
`php -S localhost:8000`
Ensuite, dans un navigateur, aller sur la page `http://localhost:8000`

Commentaires

Un commentaire est un ensemble de caractères qui sera ignoré par php.

```
// Ceci est un commentaire sur une seule ligne

/*
 * Ceci est un commentaire
 * multiligne
 */
```

Commentaires

Docblock de la PHPDoc : https://docs.phpdoc.org/latest/guides/docblocks.html

```
/**
 * Les 2 * indiquent qu'on est dans une docblock
 *
 * La docblock est une documentation interprétable par les
analyseurs syntaxiques,
 * mais qui n'influe en rien sur l'exécution du code
 *
 * @author Carlos Pereira De Amorim
 */
```

Constantes & Variables

Définissons une constante dans le fichier `index.php`

```
define('PI', 3.14);
var_dump(PI);
```

Une constante ne peut être redéfinie. Elle est donc... constante.

Constantes & Variables

Définissons une variable dans le fichier `index.php`

- Une variable doit commencer par un **$**
- Les espaces sont interdits
- le **-** est interdit
- le nom de la variable *ne* peut *pas* commencer par un **chiffre** (~~$2var~~)

```php
$season = 'hiver';
var_dump($season);
```

Généralement, les variables sont en camelCase (mais les PSR n'impose rien)

Constantes & Variables

Les variables ont un typage dynamique.

```php
// Type string : utilisation de ' conseillé
$firstName = 'Carlos';
var_dump($firstName);
// Type string : utilisation de " déconseillé
$lastName = "Pereira De Amorim";
var_dump($lastName);
// Type entier
$age = 37;
var_dump($age);
```

Constantes & Variables

```php
$size = 1.89; // Type float
var_dump($size);
$isTrainer = true; // Type booléen
var_dump($isTrainer);
$date = new DateTime(); // Objet de type DateTime
var_dump($date);
```

`var_dump` affiche le type de la variable.

Constantes & Variables

Les objets sont toujours passé par référence.

```
$today = new DateTime();
var_dump($today);
$tomorrow = $today;
$tomorrow->add(new DateInterval('P1D'));
var_dump($tomorrow);
var_dump($today); // $today a changé alors qu'il n'a pas été modifié directement
```

Constantes & Variables

Les types primitifs (entiers, réels, flottants, booléens, chaines de caractères, tableaux) ne sont pas passé par références par défaut...

```
$a = 1;
var_dump($a);
$b = $a;
$b = 2;
var_dump($b);
var_dump($a); // $a n'a pas changé
```

Constantes & Variables

... mais on peut forcer le passage par référence avec **&**

```
$c = 3;
var_dump($c);
$d = &$c;
$d = 4;
var_dump($d);
var_dump($c); // $c a changé
```

Constantes & Variables

On peut concaténer les variables.

```
var_dump($firstName.' '.$lastName);
// et identique à
var_dump("$firstName {$lastName}");

$name = $firstName; // cela ne changera pas $firstName
$name .= ' '.$lastName;
var_dump($name);
var_dump($firstName);
```

Constantes & Variables

Le transtypage est automatique.

```
// $age est automatiquement transtypé en string
var_dump('age : '.$age.' ans');
$n = '5'; // ceci est un string
var_dump($n);
var_dump($n + 1); // $n est transtypé en int
var_dump($n + 1.2); // float
var_dump($n * '100 ans de solitude'); // il y aura
quand-même une notice...
```

Constantes & Variables

Les opérations mathématiques possibles

```
$a = 5;
$b = 2;
$c = 0;
$c = $a + $b; // 7
var_dump($c);
$c = $a * $b; // 10
var_dump($c);
$c = $a / $b; // 2.5
var_dump($c);
```

Constantes & Variables

```
$c = $a - $b; // 3
var_dump($c);
$c = $a % $b; // modulo -> 1
var_dump($c);
$a++; // 6 post incrémenation
++$a; // 7 pré incrémentation (à préférer)
var_dump($a);
$b--; // 1 post décrémentation
--$b; // 0 pré décrémentation (à préférer)
var_dump($b);
```

Constantes & Variables

Variable dynamique

```
$name = 'Carlos Pereira De Amorim';
$var = 'name';
var_dump($$var); // affichera le contenu de $name
```

Constantes & Variables

Existence et destruction d'une variable

```php
var_dump(isset($newVar)); // false
$newVar = 'Hello';
var_dump(isset($newVar)); // true

unset($newVar); // destruction de $newVar
var_dump(isset($newVar)); // false
```

HEREDOC

La syntaxe HEREDOC permet de conserver les retours à la ligne.

```
$lorem = <<<EOT
Lorem Ipsum Dolor sit amet
 consectetur adipiscing elit
EOT; // l'identifiant de fin ne doit pas être indenté !

echo $lorem;
```

Tableaux

```php
// Type tableau : utilisation de [] conseillé
$hobby = ['climbing','football', '...'];
var_dump($hobby);
// Type tableau : utilisation de array déconseillé
$skill = array('agile', 'php', 'symfony', '...');
var_dump($skill);
```

Tableaux

Par défaut, un tableau est indexé numériquement

```
$citys = [
    'Nantes',
    'Bordeaux',
    ];
var_dump($citys); // [0 => Nantes, 1 => Bordeaux]
```

Il est possible (et conseillé) de laisser une virgule à la fin.

Tableaux

On peut choisir les clés

```
$citys = [
    'Éléphant' => 'Nantes',
    'Cannelé' => 'Bordeaux',
    ];
var_dump($citys); // [Éléphant => Nantes, Cannelé => Bordeaux]
```

Tableaux

Insertion de données dans un tableau existant

```
$citys[] = 'Paris'; // prendra le premier index numérique disponible
$citys['rose'] = 'Toulouse';
var_dump($citys); // [Éléphant => Nantes, Cannelé => Bordeaux, 0 => Paris, rose => Toulouse]
```

il existe aussi les méthodes particulières array_push et array_unshift

Tableaux

Il est possible de faire des tableaux multidimensionnel

```
$citys['nord'] = ['Lille', 'Dunkerque'];
var_dump($citys);
```

Tableaux

Récupération de la valeur d'un tableau

```
print_r($citys['Éléphant']);
```

NB : print_r est capable d'afficher un tableau, contrairement à echo.

Tableaux

On peut vérifier l'existence d'une clé ou d'une valeur

```
var_dump(isset($citys['rose'])); // true
var_dump(in_array('Toulouse', $citys));
```

Tableaux

On peut supprimer une entrée de tableau

```
unset($citys['rose']);
var_dump(isset($citys['rose'])); // false
```

il existe aussi les méthodes particulières array_pop et array_shift

Tableaux

Parcourir un tableau

```php
foreach ($citys as $city) { // on ignore les clés
    print_r($city);
}


foreach ($citys as $key => $city) { // on récupère les clés
    print_r($key);
    print_r($city);
}
```

Tableaux

Fun Fact

On peut partiellement traiter une chaine de caractères comme un tableau indexé

```
$ocean = 'Atlantique';
var_dump($ocean[0]); // A
```

Mais un foreach n'acceptera pas de itérer sur `$ocean`.

Tableaux

Il est très simple de connaitre la taille d'un tableau (i.e. le nombre de clés)

```
var_dump(count($city));
```

Il existe énormément de fonctions pour faire toutes sortes de choses, comme shuffle qui mélange les éléments d'un tableau.
Voici la liste ici

Transtypage

```php
var_dump((bool) 0); // false
var_dump((bool) 1); // true
var_dump((bool) 2); // true
var_dump((bool) -1); // true
var_dump((bool) ''); // false
var_dump((bool) null); // false
var_dump((bool) "string"); // true
var_dump((int) "string"); // 0
var_dump((int) "3.14 string"); // 3
var_dump((float) "3.14 string"); // 3.14
```

Opérateurs

la partie *Opérateurs* est un extrait de la documentation officielle sur les opérateurs : https://www.php.net/manual/fr/language.operators.php

Dans certains cas non intuitif, il peut être intéressant d'aller voir l'ordre de priorité des opérateurs : https://www.php.net/manual/fr/language.operators.precedence.php

Opérateurs arithmétiques

Exemple	Nom	Résultat
$a + $b	Addition	Somme de $a et $b.
$a - $b	Soustraction	Différence de $a et $b.
$a * $b	Multiplication	Produit de $a et $b.
$a / $b	Division	Quotient de $a et $b.
$a % $b	Modulus	Reste de $a divisé par $b.
$a ** $b	Exponentielle	Résultat de l'élévation de $a à la puissance $b.

Doc : https://www.php.net/manual/fr/language.operators.arithmetic.php

Opérateurs de comparaison

Exemple	Nom	Résultat
$a == $b	Egal	TRUE si $a est égal à $b après le **transtypage**.
$a === $b	Identique	TRUE si $a est égal à $b et qu'ils sont de même type.
$a != $b	Différent	TRUE si $a est différent de $b après le **transtypage**.
$a <> $b	Différent	TRUE si $a est différent de $b après le **transtypage**.

Opérateurs de comparaison

Exemple	Nom	Résultat
$a !== $b	Différent	TRUE si $a est différent de $b ou bien s'ils ne sont pas du même type.
$a < $b	Plus petit que	TRUE si $a est strictement plus petit que $b.
$a > $b	Plus grand	TRUE si $a est strictement plus grand que $b.

Opérateurs de comparaison

Exemple	Nom	Résultat
$a <= $b	Inférieur ou égal	TRUE si $a est plus petit ou égal à $b.
$a >= $b	Supérieur ou égal	TRUE si $a est plus grand ou égal à $b.
$a <=> $b	Combiné	Renvoie **-1**, **0** ou **1** lorsque $a est respectivement inférieur, égal, ou supérieur à $b.

Doc : https://www.php.net/manual/fr/language.operators.comparison.php

Opérateurs d'incrémentation et décrémentation

Exemple	Nom	Résultat
++$a	Pre-incrémente	Incrémente $a de 1, puis retourne $a.
$a++	Post-incrémente	Retourne $a, puis incrémente $a de 1.
--$a	Pré-décrémente	Décrémente $a de 1, puis retourne $a.
$a--	Post-décrémente	Retourne $a, puis décrémente $a de 1.

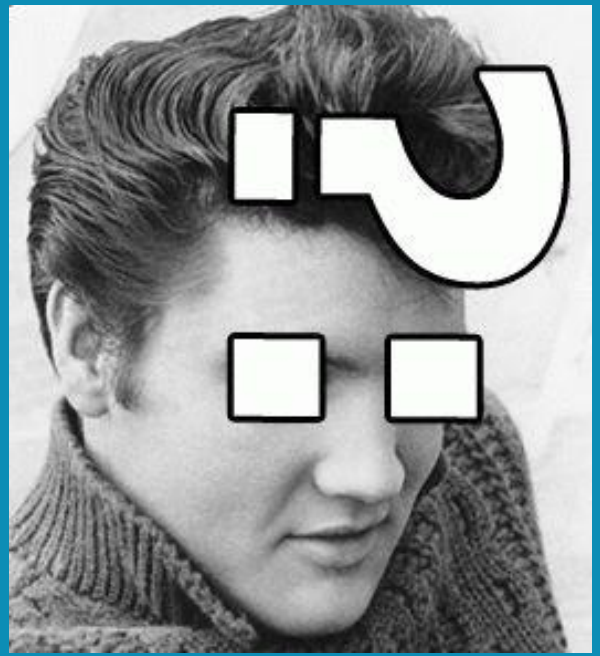
Doc : https://www.php.net/manual/fr/language.operators.increment.php

Opérateurs logiques

Exemple	Nom	Résultat
!$a	Not (Non)	TRUE si $a n'est pas TRUE.
$a && $b	And (Et)	TRUE si $a ET $b sont TRUE.
$a || $b	Or (Ou)	TRUE si $a OU $b est TRUE.

Doc : https://www.php.net/manual/fr/language.operators.logical.php

Opérateur Elvis



aussi appelé opérateur ternaire

Opérateur ternaire

```
$d = $a ? $b : $c; // est évaluée à $b si $a est évaluée à TRUE, et sinon $c si $a est évaluée à FALSE.
if ($a) {
    $d = $b;
} else {
    $d = $c;
}
```

Opérateur ternaire

On peut faire sans `$b`

```
$d = $a ?: $c; // est évaluée à $a si $a est évaluée à
TRUE, et sinon $c si $a est évaluée à FALSE.
if ($a) {
    $d = $a;
} else {
    $d = $c;
}
```

Opérateur de fusion Null (The null coalescing operator)

```php
$d = $a ?? $c; // est évaluée à $c si $a est NULL, et sinon $a.
if (is_null($a)) {
    $d = $c;
} else {
    $d = $a;
}

$d = 0 ?? 'Carlos'; // 0
$d = 0 ?: 'Carlos'; // 'Carlos'
```

Opérateurs de tableaux

Exemple	Nom	Résultat
$a + $b	Union	Union de $a et $b.
$a == $b	**Egalité**	TRUE si $a et $b contiennent les mêmes paires clés/valeurs.
$a === $b	**Identique**	TRUE si $a et $b contiennent les mêmes paires clés/valeurs dans le même ordre et du même type.

Opérateurs de tableaux

Exemple	Nom	Résultat
$a != $b	Inégalité	TRUE si $a n'est pas **égal** à $b.
$a <> $b	Inégalité	TRUE si $a n'est pas **égal** à $b.
$a !== $b	Non-identique	TRUE si $a n'est pas **identique** à $b.

Doc : https://www.php.net/manual/fr/language.operators.array.php

Structures de contrôles

Les principales structures de contrôles sont :

- `if`, `elseif`, `else`
- `switch`, `case`
- `while`, `do while`
- `for`
- `foreach`
- `try`, `catch`, `finally`

NB : PSR-2 donne une règle pour les structures de contrôles

Structures de contrôles : `if`, `elseif`, `else`

```php
$nbHabitantNantes = 309346;
$nbHabitantBordeaux = 254436;
if ($nbHabitantNantes > $nbHabitantBordeaux)
    echo 'Nantes a plus d\'habitants que Bordeaux';
```

Il n'y a pas d'accolades parce qu'il n'y a qu'une seule instruction

Structures de contrôles : `if`, `elseif`, `else`

```php
$nbHabitantNantes = 309346;
$nbHabitantBordeaux = 254436;
if ($nbHabitantNantes > $nbHabitantBordeaux) {
    echo 'Nantes a plus d\'habitants que Bordeaux';
}
```

Version "complète" avec accolades

Structures de contrôles : `if`, `elseif`, `else`

```php
$nbHabitantNantes = 309346;
$nbHabitantBordeaux = 254436;
if ($nbHabitantNantes > $nbHabitantBordeaux) {
    echo 'Nantes a plus d\'habitants que Bordeaux';
} else {
    // $nbHabitantNantes <= $nbHabitantBordeaux
    echo 'Bordeaux a au moins autant d\'habitants que Nantes';
}
```

Structures de contrôles : `if`, `elseif`, `else`

```php
$nbHabitantNantes = 309346;
$nbHabitantBordeaux = 254436;
if ($nbHabitantNantes > $nbHabitantBordeaux) {
    echo 'Nantes a plus d\'habitants que Bordeaux';
} elseif ($nbHabitantNantes < $nbHabitantBordeaux) {
    echo 'Bordeaux a plus d\'habitants que Nantes';
}
```

Structures de contrôles : `if`, `elseif`, `else`

```
$nbHabitantNantes = 309346;
$nbHabitantBordeaux = 254436;
if ($nbHabitantNantes > $nbHabitantBordeaux) {
    echo 'Nantes a plus d\'habitants que Bordeaux';
} elseif ($nbHabitantBordeaux > 0 && $nbHabitantNantes <
$nbHabitantBordeaux) {
    echo 'Bordeaux a plus d\'habitants que Nantes';
} else {
    echo 'Bordeaux et Nantes ont autant d\'habitants';
}
```

Structures de contrôles : `switch`, `case`

```php
$city = 'Nantes';
switch ($city) {
    case 'Nantes':
        echo 'Allons voir l\'éléphant'; // no break
    case 'Rennes':
        echo 'Allons boire du cidre';
        break; // empêche d'exécuter default
    default:
        echo 'Allons boire et manger';
        break;
}
```

Structures de contrôles : `while`, `do while`

```php
$kmNantesBordeaux = 347;
$kmFait = 0;

while ($kmFait == 0 || $kmFait < $kmNantesBordeaux) {
    echo sprintf('km parcouru sont de %s km', $kmFait);
    ++$kmFait; // ne pas oublier cette ligne, sinon vous aurez une boucle infini
}
```

Structures de contrôles : `while`, `do while`

```
while ($kmFait == 0 || $kmFait < $kmNantesBordeaux) {
    if (!$kmFait) {
        ++$kmFait; // ne pas oublier cette ligne
        continue; // on passe à l'itération suivante'
    }
    if ($kmFait > 200) {
        break; // on ne va pas plus loin
    }
    echo sprintf('km parcourus sont de %s km', $kmFait);
    ++$kmFait; // ne pas oublier cette ligne
}
```

Structures de contrôles : `while`, `do while`

```php
$kmNantesBordeaux = 347;
$kmFait = 0;

do {
    echo sprintf('km parcourus sont de %s km', $kmFait);
    ++$kmFait; // ne pas oublier cette ligne
} while ($kmFait < $kmNantesBordeaux);
```

Comme le `while` sauf que le test est effectué *après* la première exécution.

Structures de contrôles : `for`

```php
$kmNantesBordeaux = 347;

for ($i = 0; $i < $kmNantesBordeaux; ++$i) {
    if (!$i) {
        continue; // on passe à l'itération suivante
    }
    echo sprintf('km parcourus sont de %s km', $i);
}
```

Structures de contrôles : foreach

```php
$citys = [
    'Éléphant' => 'Nantes',
    'Cannelé' => 'Bordeaux',
    ];
foreach ($citys as $key => $city) {
    if (empty($city)) {
        break; // on sort de la boucle
    }
    print_r($city);
}
```

Structures de contrôles : `try`, `catch`, `finally`

```php
try {
    // on envoie une exception
    throw new LogicException('où est la logique ?');
} catch (LogicException $e) {
    echo 'Oups ! une erreur';
    echo $e->getMessage();
} catch (Exception $e) {
    echo $e->getMessage();
} finally {
    echo 'ici, c\'est finally !';
}
```

Fonctions et closures

La fonction ne peut être appelée qu'après avoir été déclarée

```
function stripAccents($stripAccents)
{
    echo iconv('UTF-8', 'ASCII//TRANSLIT//IGNORE',
$stripAccents);
}

stripAccents('un éléphant');
```

Fonctions et closures

On peut forcer les types des paramètres. Pas de transtypage dans ce cas.

```
function stripAccents(string $stripAccents) // on accepte
uniquement les string non null
{
    echo iconv('UTF-8', 'ASCII//TRANSLIT//IGNORE',
$stripAccents);
}

stripAccents('un éléphant');
```

Fonctions et closures

que On peut retourner une valeur

```
function stripAccents(?string $stripAccents) // on accepte
les string et NULL
{
    // on n'affiche rien. On retourne une valeur à la place
    return iconv('UTF-8', 'ASCII//TRANSLIT//IGNORE',
$stripAccents);
}

echo stripAccents('un éléphant');
```

Fonctions et closures

On peut forcer le type de la valeur retournée

```
function stripAccents(?string $stripAccents): string //
renvoie forcément un string non NULL
{
    // on n'affiche rien. On retourne une valeur à la place
    return iconv('UTF-8', 'ASCII//TRANSLIT//IGNORE',
$stripAccents);
}

echo stripAccents('un éléphant');
```

Fonctions et closures

Une closure (fonction anonyme) est une fonction sans nom dans une variable.

```
$where = 'Bordeaux';
$closure = function ($who) use ($where) {
    echo sprintf('%s is in %s', $who, $where);
};
$closure('Carlos'); // Carlos is in Bordeaux
```

Splat Operator (...)

splat operator (...) permet d'avoir un nombre infini d'argument d'une fonction.

```
function stripAccents(string ...$stripAccents)
{
    foreach ($stripAccents as $strip) {
        echo iconv('UTF-8', 'ASCII//TRANSLIT//IGNORE',
$strip);
    }
}

echo stripAccents('un éléphant', 'dans la forêt');
```

Création d'une page HTML

créer un fichier `page.php`

```
<html>
    <head>
        <title>Formation PHP</title>
    </head>
    <body>
    </body>
</html>
```

Création d'une page HTML

Dans le body, ajouter du code php

```
<?php
    $city = 'Bordeaux';
    echo "<p>Bonjour $city</p>";
?>
```

Ne pas oublier de fermer la balise php avec `?>`

Lancer `php -S localhost:8000`
et aller sur `http://localhost:8000/page.php`

Formulaires : `$_GET` et `$_POST`

Remplacer le code php par

```php
<?php
    // on récupère la valeur passée en GET
    $city = $_GET['city'];
    echo "<p>Bonjour $city</p>";
?>
```

et aller sur `http://localhost:8000/page.php?city=Bordeaux`

Formulaires : `$_GET` et `$_POST`

On va remplacer le code php par du code html + php qui envoie la donnée en POST.

Pour rappel, une donnée GET est passée dans l'url.
Une donnée POST est passée dans les paramètres des l'appel.

Pour tester, remplacer le code php précédent.

Formulaires : $_GET et $_POST

```
<form action="page.php" method="post">
    <input type="text" name="city" id="city" required>
    <input type="submit" value="Go !">
</form>
<?php
    if (isset($_POST['city'])) {
        $city = $_POST['city'];
        echo "<p>Bonjour $city</p>";
    }
?>
```

Formulaires : `$_GET` et `$_POST`

et aller sur `http://localhost:8000/page.php`

Vous pourrez soumettre le formulaire.

Aller sur "Examiner l'élément" pour voir ce qu'il se passe.

Cookies

Un cookie est une donnée stockée sur le navigateur par le site web.
En php, le code pour déposer un cookie doit être AVANT le code HTML.

Tout en haut de page.php, ajouter :

```
<?php setcookie('city', 'Bordeaux', ['expires' => time() + 365*24*3600, 'httponly' => true]); ?>
```

Pour afficher le cookie, remplacer le code php précédent.

Cookies

```php
<?php
    if (isset($_COOKIE['city'])) {
        $city = $_COOKIE['city'];
        echo "<p>Bonjour $city</p>";
    } else {
        echo 'pas de cookie, pas de ville';
    }
?>
```

Sessions

Les sessions sont des données stockées généralement sur le serveur, et qui permettent de reconnaitre les utilisateurs, notamment ceux qui sont connectés.

Selon votre configuration, il se peut qu'un cookie soit aussi généré pour stocker la session coté navigateur.

Sessions

Pour démarrer une session, mettre tout en haut de `page.php`

```
session_start();
```

Sessions

remplacer le code pour récupérer un cookie par :

```php
<?php
        $_SESSION['city'] = 'Bordeaux';
        $city = $_SESSION['city'];
        echo "<p>Bonjour $city</p>";
        echo session_id(); // on affiche l'identifiant
?>
```

et aller sur `http://localhost:8000/page.php`

Sessions

Maintenant, retirer l'affection de `$city` dans la session.

```
<?php
        $city = $_SESSION['city'];
        echo "<p>Bonjour $city</p>";
?>
```

et aller sur `http://localhost:8000/page.php`

Sessions

Ouvrir une fenêtre de navigation privée et aller sur `http://localhost:8000/page.php`

La ville n'apparait plus car c'est une nouvelle session.

Sessions

`session_destroy()` permet de *détruire* une session mais ne supprime pas tout de suite les données de la session courante.

Mettre `session_destroy()` juste avant l'affichage de `$city`.

Sessions

```php
<?php
        var_dump($_SESSION);
        $_SESSION['city'] = 'Bordeaux';
        $city = $_SESSION['city'];
        echo "<p>Bonjour $city</p>";
        session_destroy(); // cela va supprimer la session
        var_dump($_SESSION); // toujours là
?>
```

et aller sur `http://localhost:8000/page.php`

Sessions

Pour supprimer tout de suite les données de la session courante, vider `$_SESSION`

Sessions

```php
<?php
        $_SESSION['city'] = 'Bordeaux';
        $city = $_SESSION['city'];
        echo "<p>Bonjour $city</p>";
        session_destroy(); // cela va supprimer la session
        $_SESSION = []; // on vide les données de session
        var_dump($_SESSION); // les données ne sont plus là
?>
```

et aller sur `http://localhost:8000/page.php`

`require` et `include`

Il est pratique de séparer en plusieurs fichiers sont code php pour maintenir la lisibilité.

`include file.php` va insérer `file.php` dans le fichier php où vous vous trouvez comme si le code avait été écrit à cet endroit.

Un problème dans le nom du fichier provoquera un warning.

require et include

créer le fichier `page.inc.php` et mettre

```php
<?php
        $_SESSION['city'] = 'Bordeaux';
        $city = $_SESSION['city'];
        echo "<p>Bonjour $city</p>";
        session_destroy(); // cela va supprimer la session
        $_SESSION = []; // on vide les données de session
        var_dump($_SESSION); // les données ne sont plus là
?>
```

require et include

Dans <body> de page.php, mettre

```
<form action="page.php" method="post">
    <input type="text" name="city" id="city" required>
    <input type="submit" value="Go !">
</form>
<?php include 'page.inc.php'; ?>
```

et aller sur http://localhost:8000/page.php

`require` et `include`

`require` est exactement comme include, mais un problème dans le nom du fichier provoquera une erreur.

`require_once` (à préférer à `require`) s'assure qu'un fichier n'est inséré qu'une seule fois.

require et include

Dans <body> de page.php, mettre

```
<form action="page.php" method="post">
    <input type="text" name="city" id="city" required>
    <input type="submit" value="Go !">
</form>
<?php require_once 'page.inc.php'; ?>
```

et aller sur http://localhost:8000/page.php

TP *Fibonacci*

TP *Le chiffrement de César*

TP *Les blagues du Joker*

La POO

Programmation Orientée Objet

- Glossaire
- Encapsulation
- Héritage
- Polymorphisme
- Interfaces

Glossaire

- *Objet* : Structure de données valuées qui répond à un ensemble de messages. Un objet est représenté par une *Classe* ou une *Interface*
- *Attribut* : Donnée ou Objet qui représente l'objet
- *Méthode* : Action de l'objet
- *Constructeur* : Méthode appelé lorsque l'objet est créé
- *Type* : Texte qui représente l'objet
- *Instance* : Représentation concrète d'un objet

Encapsulation

Permet de définir la visibilité des attributs et des méthodes de l'objets

- *public* : Tout le monde peut voir l'attribut ou la méthode
- *private* : Seul l'objet connait l'attribut ou la méthode
- *protected* : Seul l'objet et son/ses héritier/s connaissent l'attribut ou la méthode
- *package* : Tous les objets du "package" connaissent l'attribut ou la méthode (très peu utilisé)

Encapsulation

public

Pour un objet ***Homme*** :

- ***la couleur des cheveux*** est un attribut public
- ***manger()*** est une méthode public

Encapsulation

private

Pour un objet ***Femme*** :

- ***date de naissance*** est un attribut privé
- ***donnerNaissance()*** est une méthode privée

Encapsulation

protected

Pour un objet ***Homme*** :

- ***nombre de dents*** est un attribut protected hérité de l'objet ***Humain***
- ***marcherDebout()*** est une méthode protected héritée de l'objet ***Humain***

Héritage - Définition

Un objet dit "père" peut transmettre ses caractéristiques *public* et *protected* à son(ses) objet(s) dit "fils".
On dit que l'objet dit "fils" *hérite* de l'objet dit "père".

Dans la littérature, on dira aussi de façon synonymique que :

- le "fils" *étend* (`extends`) le "père"
- le "fils" est une *spécialisation* du "père"
- le "fils" *dérive* du "père"

Cela est aussi appelé *Polymorphisme par héritage*.

Héritage - Exemple

Exemple :
Un objet ***Femme*** héritera des caractéristiques d'un objet ***Humain***.
Un objet ***Homme*** héritera aussi des caractéristiques d'un objet ***Humain***.

Mais chaque objet ***Femme*** et ***Homme*** a des attributs et méthode spécifiques.

Polymorphisme

Le *polymorphisme* est la possibilité d'un attribut, d'une méthode ou d'un objet de prendre plusieurs formes.

Il existe plusieurs types de polymorphisme. Voici les 2 essentiels :

- *Polymorphisme par héritage* permet dans une classe dite "fille" de redéfinir une méthode héritée
- *Polymorphisme paramétrique* permet d'avoir le même nom de méthode mais avec des paramètres différents, que ce soit en nombre ou en type. *Cela n'est pas possible en **PHP***

Polymorphisme

Polymorphisme par héritage

Exemple :
Si la classe ***Homme*** hérite de la méthode **parler()**, alors la classe pourra redéfinir la méthode **parler()** pour y introduire la notion de la mue de la voix.

Classe, Classe Abstraite et Interface

Un objet dont on veut avoir une ou plusieurs instance sera défini en tant que *Classe*.

En PHP, cela donnera :

```
class Homme {
}
```

Classe, Classe Abstraite et Interface

Un objet dont on ne veut pas avoir d'instance sera défini en tant que *Classe Abstraite*.

En PHP, cela donnera :

```
abstract class AbstractHumain {
}


class Homme extends AbstractHumain {
}
```

Classe, Classe Abstraite et Interface

Une Interface est un ensemble de signatures de méthodes publiques d'un objet.

En PHP, cela donnera :

```
interface IMammal {
    public function respirer();
}


abstract class AbstractHumain implements IMammal {
}
```

Static

static est un type particulier qui peut être attribué aux *attributs* et *méthodes* d'un *objet*.

Un attribut ou une méthode *static* sera accessible même si l'objet n'est pas instancié.

Static

```php
class Homme {
    public static $couleurCheveux = "noir";
    public static function marcher() { /* ...*/ }
}

// Le code suivant fonctionnera dans n'importe quelle méthode

Homme::$couleurCheveux; // pas d'erreur car static
Homme::marcher(); // pas d'erreur car static
```

Création et Utilisation d'une classe en php

Créer le fichier `Rectangle.php` et mettre la classe `Rectangle` à l'intérieur.

```php
<?php
class Rectangle
{

}
```

Création et Utilisation d'une classe en php

Ajouter des attributs de classe correspondant à la taille de chaque coté.

```
private float $side1;
private float $side2;
```

Création et Utilisation d'une classe en php

Ajouter un constructeur pour définir la taille de chaque coté.

```
public function __construct(float $side1, float $side2)
{
    $this->side1 = $side1;
    $this->side2 = $side2;
}
```

Création et Utilisation d'une classe en php

Ajouter une méthode pour connaitre l'aire.

```
public function area(): float
{
    return $this->side1 * $this->side2;
}
```

Création et Utilisation d'une classe en php

Créer le fichier `quadri.php` pour utiliser la classe `Rectangle`.

```
require_once 'Rectangle.php';
$rectangle = new Rectangle(3,4);
echo $rectangle->area();
```

et lancer `php quadri.php`

Création et Utilisation d'une classe abstraite en php

Créer le fichier `Quadrilateral.php` et mettre la classe abstraite `Quadrilateral` à l'intérieur.

```php
<?php
abstract class Quadrilateral
{

}
```

Création et Utilisation d'une classe abstraite en php

Ajouter des attributs de classe correspondant à la taille de chaque coté.

```
protected float $side1;
protected float $side2;
protected float $side3;
protected float $side4;
```

Création et Utilisation d'une classe abstraite en php

Ajouter un constructeur pour définir la taille de chaque coté.

```
    public function __construct(float $side1, float $side2,
float $side3, float $side4)
    {
        $this->side1 = $side1;
        $this->side2 = $side2;
        $this->side3 = $side3;
        $this->side4 = $side4;
    }
```

Création et Utilisation d'une classe abstraite en php

Ajouter la fonction de calcul du périmètre.

```
    public function perimeter(): float
    {
        return $this->side1 + $this->side2 + $this->side3
+$this->side4;
    }
```

Création et Utilisation d'une classe abstraite en php

Ajouter la fonction abstraite de calcul d'aire.

```
public abstract function area(): float;
```

Création et Utilisation d'une classe abstraite en php

Dire à `Rectangle` d'étendre `Quadrilateral`.

```
require_once 'Quadrilateral.php';

class Rectangle extends Quadrilateral {
    //...
}
```

Création et Utilisation d'une classe abstraite en php

Supprimer les attributs de classe et adapter le constructeur.

```
    public function __construct(float $side1, float $side2)
    {
        parent::__construct($side1, $side2, $side1,
$side2);
    }
```

Création et Utilisation d'une classe abstraite en php

Dans `quadri.php`, afficher le périmètre en plus de l'aire.

```
echo $rectangle->perimeter();
```

et lancer `php quadri.php`

Création et Utilisation d'une interface en php

Créer le fichier `Polygone.php` et mettre l'interface `Polygone` à l'intérieur.

```php
<?php
interface Polygone
{

}
```

Création et Utilisation d'une interface en php

Ajouter les fonctions `perimeter`, `area` et `type` qui indique le type de polygone.

```
public function perimeter(): float;
public function area(): float;
public function type(): string;
```

Création et Utilisation d'une interface en php

Dire à `Quadrilateral` d'implémenter `Polygone`.

```php
require_once 'Polygone.php';

abstract class Quadrilateral implements Polygone
{
    //...
}
```

Création et Utilisation d'une interface en php

Implémenter la fonction `getType` dans `Rectangle`.

```
public function type(): string
{
    return get_class();
}
```

Création et Utilisation d'une interface en php

Faire afficher le type dans `quadri.php`

```
echo $rectangle->type();
```

TP *Les FoodTrucks de Bordeaux*

PHP et les BDD

- Installation MySql
- Connexion MySql
- Insertion de données
- Récupération de données

Installation MySql avec WAMP et MAMP

Avec WAMP et MAMP, normalement, c'est déjà fait !

Installation MySql avec Ubuntu

```
sudo apt install mysql-client mysql-server php-mysql
```

Ne pas oublier `php-mysql` qui permet à php de se connecter à mysql.

Connexion MySql

Pré-requis :

- Connaitre le *username* et le *mot de passe* d'un accès à mysql.
- Créer un schema **formation_php**
- Créer une table **polygone** avec 5 colonnes :
  - **id** Primary Key, int, auto increment
  - **type** varchar(45)
  - **side1** float
  - **side2** float
  - **side3** float
  - **side4** float

Connexion MySql

Créer un fichier `db.php` pour gérer la connexion à la Base De Données.

Pour simplifier les choses, la connexion à la BDD est souvent faite à un seul endroit.

Connexion MySql

```php
<?php
try {
    $conn = new
PDO("mysql:host=localhost;dbname=formation_php",
'username', 'password');
} catch(PDOException $e) {
    echo "Connection failed: " . $e->getMessage();
} finally {
    $conn = null;
}
```

Insertion de données

Pour insérer des données, ajouter juste après la connexion :

```
    $sql = "INSERT INTO polygone (type, side1, side2,
side3, side4)
    VALUES ('rectangle', 3, 4, 3, 4)";
    $conn->exec($sql);
```

et lancer `php db.php`
Les données ont été ajoutées en BDD.

Insertion de données

Pour plus de sécurité, on va variabiliser les données.

La variabilisation des données permet à PDO de s'assurer qu'aucune tentative d'injection sql est faite.

Remplacer le code précédent.

Insertion de données

```
    $sql = "INSERT INTO polygone (type, side1, side2,
side3, side4)
    VALUES (:type, :side1, :side2, :side1, :side2)";
    $prep = $conn->prepare($sql);
    $prep->execute([':type' => 'carré', ':side1' => 4,
':side2' => 4]);
```

et lancer `php db.php`
Les données ont été ajoutées en BDD.

Insertion de données

On utilise `bindValue` (ou `bindParam`) en général.

```
$prep->bindValue(':type', 'losange');
$prep->bindValue(':side1', 5);
$prep->bindValue(':side2', 6);
$prep->bindValue(':side3', 7);
$prep->bindValue(':side4', 8);
$prep->execute();
```

et lancer `php db.php` Les données ont été ajoutées en BDD.

Récupération de données

On va récupérer les valeurs

```
$sql = "SELECT * FROM polygone";
$prep = $conn->prepare($sql);
$prep->execute();
$polygones = $prep->fetchAll();
print_r($polygones);
```

lancer `php db.php`

Par défaut, c'est un tableau qui est renvoyé.

Récupération de données

On peut récupérer les données sous formes d'objet `stdClass`

```
$polygones = $prep->fetchAll(PDO::FETCH_OBJ);
```

lancer `php db.php`

Beaucoup d'autres possibilités pour récupérer les données.
Voir les docs ici, ça, là et là.

TP *CVThèque*

PHP et les dépendances

- Composer
- Packagist
- Utilisation de composer
- Installation d'un package
- Utilisation d'un package
- Mise à jour des packages

Composer

A Dependency Manager for PHP

Composer

https://getcomposer.org

- remplace PEAR
- contient un `composer.json` pour connaitre les dépendances du projet
- contient un `composer.lock` pour connaitre les dépendances installées par le projet (fixe le commit de chaque dépendance)
- **génère l'autoload qui sera chargé par le framework**
- on peut créer un projet avec `composer create-project symfony/skeleton`
- on peut exécuter des scripts à l'installation et à la mise à jour
- etc...

composer.json

Tous les détails dans la doc :
https://getcomposer.org/doc/04-schema.md

Cheat Sheet de JoliCode :
https://composer.json.jolicode.com/

composer.json

```
{
    "type": "project",
    "license": "proprietary",
    "require": {
        "php": "^7.1.3",
        "ext-ctype": "*",
        "ext-iconv": "*",
        "symfony/console": "4.3.*",
        "symfony/dotenv": "4.3.*",
        "symfony/flex": "^1.3.1",
        "symfony/framework-bundle": "4.3.*",
        "symfony/yaml": "4.3.*"
    },
    "require-dev": {
    },
    "config": {
        "preferred-install": {
            "*": "dist"
        },
        "sort-packages": true
    },
    "autoload": {
        "psr-4": {
            "App\\": "src/"
        }
    },
    "autoload-dev": {
        "psr-4": {
            "App\\Tests\\": "tests/"
        }
    },
    "replace": {
        "paragonie/random_compat": "2.*",
        "symfony/polyfill-ctype": "*",
        "symfony/polyfill-iconv": "*",
        "symfony/polyfill-php71": "*",
        "symfony/polyfill-php70": "*",
        "symfony/polyfill-php56": "*"
    },
    "scripts": {
        "auto-scripts": {
            "cache:clear": "symfony-cmd",
            "assets:install %PUBLIC_DIR%": "symfony-cmd"
        },
        "post-install-cmd": [
            "@auto-scripts"
        ],
        "post-update-cmd": [
            "@auto-scripts"
        ]
    },
    "conflict": {
        "symfony/symfony": "*"
    },
    "extra": {
        "symfony": {
            "allow-contrib": false,
            "require": "4.3.*"
        }
    }
}
```

Packagist

The PHP Package Repository

Packagist

https://packagist.org

- Recense toutes les librairies PHP
  - Une librairie est un ensemble fonctionnel
  - Une librairie peut avoir ses propres dépendances
- Recense tous les bundles Symfony
  - Un bundle Symfony est une librairie destinée à être utilisée dans Symfony
  - Certains bundle sont des bridges avec une librairie
  - Un bundle peut avoir ses propres dépendances
- Composer va chercher dans Packagist les dépendances à installer

Packagist

Exemple de librairie :

- https://packagist.org/packages/google/apiclient Librairie pour appeler les api google
- https://packagist.org/packages/guzzlehttp/guzzle Client HTTP

Utilisation de composer

Pour utiliser composer, on est obligé d'avoir un fichier `composer.json` à la racine du projet.

Utilisation de composer

```
{
    "require": {
        "php": "^7.4.0"
    },
    "autoload": {
        "psr-4": {
            "App\\": "./"
        }
    }
}
```

Utilisation de composer

Ajouter un namespace dans `Polygone`

```
namespace App;
```

Le namespace doit être tout en haut, et non indenté.

Utilisation de composer

Ajouter un `namespace` et un `use` dans `Quadrilateral`

```php
namespace App;

use App\Polygone;
```

Utilisation de composer

Ajouter un `namespace` et un `use` dans `Rectangle`

```
namespace App;

use App\Quadrilateral;
```

Utilisation de composer

Dans `quadri.php`, ajouter un `use`, et le `require` de l'autoload généré par composer.

```
use App\Rectangle;

require 'vendor/autoload.php';
```

et lancer `php quadri.php`

Utilisation de composer

Demander à composer de générer l'autoload.
Dans le terminal, à la racine du projet, exécuter :

```
composer install
```

Installation d'un package

Utiliser le package finder pour lister les fichiers dans le dossier.
Dans le terminal, à la racine du projet, exécuter :

```
composer require symfony/finder
```

Utilisation d'un package

Dans `quadri.php`, ajouter le `use` tout en haut

```
use Symfony\Component\Finder\Finder;
```

Utilisation d'un package

Ajouter le code utilisant `Finder`.

```
$finder = new Finder();
// find all files in the current directory
$finder->files()->in(__DIR__);
foreach ($finder as $file) {
    echo $absoluteFilePath = $file->getRealPath();
}
```

et lancer php `quadri.php`

Mise à jour des packages

Pour mettre à jour les packages, lancer dans le terminal :

```
composer update
```

TP *HTMLDown*