

PHP

CM4-2 : PHP « avancé »

Mickaël Martin Nevot

V5.0.0



Cette œuvre de [Mickaël Martin Nevot](#) est mise à disposition selon les termes de la [licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Partage à l'Identique 3.0 non transposé](#).

PHP

- I. Présentation
- II. PHP I
- III. XML
- IV. Regexp
- V. PHP II
- VI. MySQL
- VII. POO
- VIII. PDO
- IX. Hacking
- X. [PHP « avancé »](#)

Affichage

- Affiche **une** chaîne de caractères : `print $var`
- Affiche **une ou plusieurs** chaîne de caractères :
 - `echo $var`
 - `echo $var1, $var2`
- Affiche la valeur d'une **variable** : `print_r($var, ...)`
- Affiche **valeur et type** d'une variable : `var_dump($var, ...)`
 - Tous les attributs (publiques, protégés, et privés) d'un objet sont affichés, sauf si méthode `__debugInfo()`
- **Code PHP** (valide) d'une variable : `var_export($var, ...)`

Vérification de type

- Contrôle de type :
 - `is_bool($var)`
 - `is_int($var)`
 - `is_float($var)`
 - `is_string($var)`
 - `is_array($var)`
 - `is_object($var)`
- Filtre une variable avec un filtre spécifique :
`filter_var($var, ...)`

Le contrôle (y compris le format) des données et des types doit être effectué trois fois, aux niveaux client, serveur et base de données

Cookies

- **Petit fichier texte** (limité à 4 ko) côté client
- Permet de **suivre** les visiteurs ← Préférer les sessions pour l'authentification
- Les **cookies expirent** (et ne sont plus envoyés par le navigateur au serveur) sous certaines conditions :
 - A la fermeture du navigateur si le cookie n'est pas persistant
 - Une date d'expiration a été définie, et a été atteinte
 - La date d'expiration du cookie est changée en une date du passé
 - Le navigateur supprime le cookie à la demande de l'utilisateur

En cas d'acceptation du cookie, le navigateur le transmettra automatiquement à chaque requête HTTP

Date évaluée côté client

Cookies

- Envoyer un *cookie* : ← Pas d'affichage avant

```
setcookie($name, $value, $expire, $path = 0, $domain, $secure = false, $httponly = false)
```

- Accessible avec `$_COOKIE` :

```
setcookie('TestCookie', 'val', time() + 3600); // Expiration dans une heure.
```

```
// Afficher un cookie.
```

```
echo $_COOKIE['TestCookie'];
```

```
// Afficher tous les cookies.
```

```
print_r($_COOKIE);
```

Sécurité : config. serveur
avec HttpOnly et Secure



La CNIL vérifie (depuis 10/2014) que les sites Web français utilisant des cookies publicitaires ou de mesure d'audience invitent les visiteurs à leur consentement préalable



Déréférencement

- Tableau :

```
echo [1, 2, 3][0];
```

- Chaîne de caractères :

```
echo 'PHP'[0];
```

- Méthode :

```
class Tire
{
    public $brand = 'michelin';
}
class Car
{
    function tire()
    {
        return new \Tire();
    }
}
...
$car = new \Car();
echo 'Tires: ' . $car->tire()->$brand;
```

Fonction anonyme

- Usage :
 - Fonction de rappel
 - Assignment à une variable
 - Utilisation de `use` pour importer des variables depuis la portée où la fonction est définie
 - Etc.

```
echo preg_replace_callback('~-([a-z])~', function ($match) {  
    return strtoupper($match[1]);  
}, 'hello');
```



Fonctions utiles

- Vérifie si une clef existe dans un tableau :
 - `if (array_key_exists('first', $arr)) { ... }`
 - Teste la **définition d'une variable** en prenant en compte les **valeurs NULL** : `array_key_exists('login', $_GET)`
- Constante avec des valeurs complexes :

```
define('FRUITS', serialize(array('apple', 'cherry', 'banana')));  
$my_fruits = unserialize(FRUITS);
```
- Opérateur `instanceOf` : ←

A utiliser avec parcimonie

```
if ($myObj instanceof MyClass) { ... }
```
- Classe d'un objet : `get_class($obj)`
- Classe parent d'un objet : `get_parent_class($obj)`
- Lecture et analyse CSV :
 - `fgetcsv($handle, $length = ..., ...)`

Fonctions utiles

- Découpage d'informations sur une URI :

```
print_r(pathinfo('/some/path/.test'));  
// Array  
// (  
//     [dirname] => /some/path  
//     [basename] => .test  
//     [extension] => test  
//     [filename] =>  
// )
```

- Insère `
` automatiquement :

```
echo nl2br('Ceci\r\nest\r\nune chaîne\r');  
// Ceci<br>  
// est<br>  
// une<br>  
// chaîne<br>
```

Résolution statique à la volée

- Aussi appelé **Late Static Bindings**
- Référencer la classe appelée lors d'un d'héritage statique

```
class Cat {  
    public static function deadOrAlive1()  
    {  
        self::getStatus();  
    }  
    public static function deadOrAlive2()  
    {  
        static::getStatus(); // Utilisation de static à la place de self.  
    }  
    protected static function getStatus() { echo 'Alive'; }  
}  
class SchrodingerCat extends Cat  
{  
    protected static function getStatus() { echo 'Dead'; }  
}  
SchrodingerCat::deadOrAlive1(); // Affiche : Alive.  
SchrodingerCat::deadOrAlive2(); // Affiche : Dead.
```

Permet de palier aux limitations de self

Garbage collector (PHP 5.3+)

- **Activation** du Garbage collector : `gc_enable()`
- **Désactivation** du Garbage collector : `gc_disable()`
- **Libération de la mémoire perdue** : `gc_collect_cycles()`

```
class Node {  
    public $parentNode;  
    public $childNodes = array();  
    public function Node() {  
        $this->nodeValue = str_repeat('0123456789', 128);  
    }  
}
```

```
function createRelationship() {  
    $parent = new \Node();  
    $child = new \Node();  
    $parent->childNodes[] = $child;  
    $child->parentNode = $parent;  
}
```

Espace de noms

Fully qualified name (FQN) : nom complet

- Permet de **regrouper** des classes, interfaces, fonctions ou constantes
- Permet d'éviter des alias ou des noms trop long (**lisibilité**)

```
namespace a\b
{
    function getNamespace()
    {
        echo __NAMESPACE__; // Constante magique : l'espace de noms courant.
    }
}
```

L'utilisation de \ renvoie au à l'espace de nom global, et est plus rapide lorsqu'il n'est pas nécessaire

```
namespace b
{
    b\getNamespace(); // Appel de manière relative.
    \a\b\getNamespace(); // Appel de manière absolue.
}
...
use a\b\c\d\e\f as nsF, g\h\i\j\k\l as nsL;
use a\b\c\d\e\f\{MyClass, MyOtherClass};
```

Sous-espace de noms possibles

Alias possibles

Trait

- Mot-clef : `trait` ← Ne peut pas être instancié
- Mécanismes de **réutilisation de code source**
- Réduit les limites de l'héritage simple
- Permet l'héritage de comportements horizontaux (avec `use`)

```
trait EZCReflectionReturnInfo {  
    function getReturnType() { ... }  
    function getReturnDescription() { ... }  
}  
  
class EZCReflectionMethod extends ReflectionMethod {  
    use EZCReflectionReturnInfo;  
    ...  
}  
  
class EZCReflectionFunction extends ReflectionFunction {  
    use EZCReflectionReturnInfo;  
    ...  
    $this->getReturnType();  
}
```

Trait

- Utilisation de plusieurs à la fois possible
- Prioritaire par rapport à la classe parente, mais pas par rapport à la classe courante
- Résolution de conflits : `insteadof` (exclusion)/`as` (alias)



Programmation modulaire

- Regroupement de codes sources en **unités de travail logiques**
- **Encapsulation**
- **Réutilisabilité** et partage du code facilités
- Facilitation de réalisation de bibliothèques
- **Généricité** possible



Codage de caractères

- Encodage des fichiers textuels (HTML, PHP, etc.)
 - Sans BOM (*byte order mark*)

- Page HTML :

```
<meta charset="utf-8">
```

- E-mail :

```
$message .= 'Content-Type: text/plain; charset=utf-8' . "\n\n";
```

- Page CSS (première ligne) :

```
@charset "utf-8";
```

- Base de données :

```
CREATE TABLE ( ... ) DEFAULT CHARSET=utf8;  
SET NAMES utf8;
```

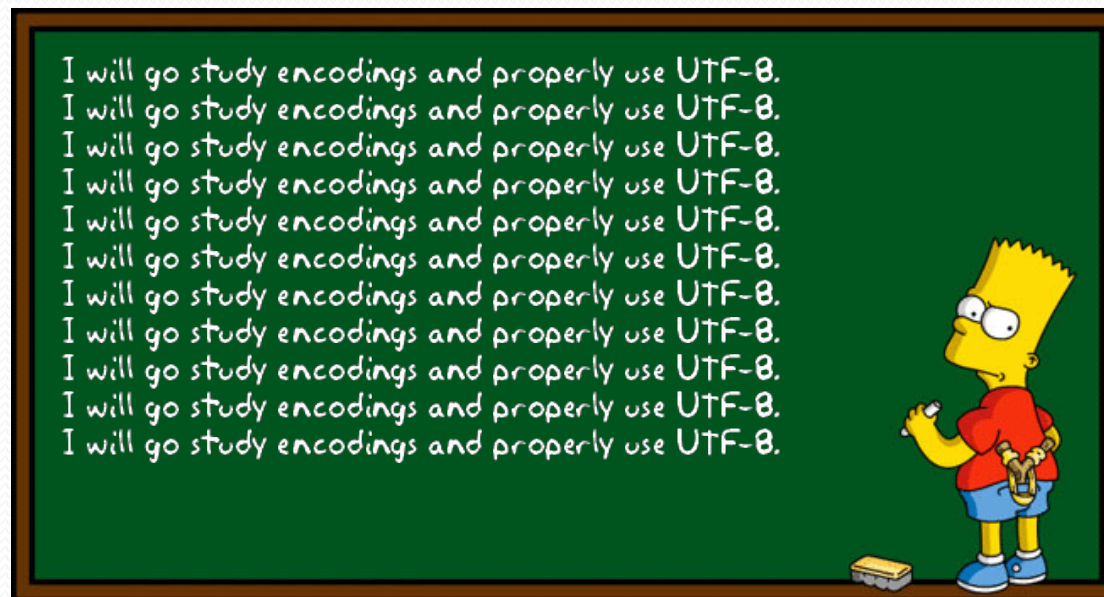
- PDO :

```
$dsn = 'mysql:host=localhost;dbname=my_dbname';  
$pdo = new \PDO($dsn, 'mysql_username', 'mysql_password');  
$pdo->exec('SET CHARACTER SET utf8');
```

Codage de caractères

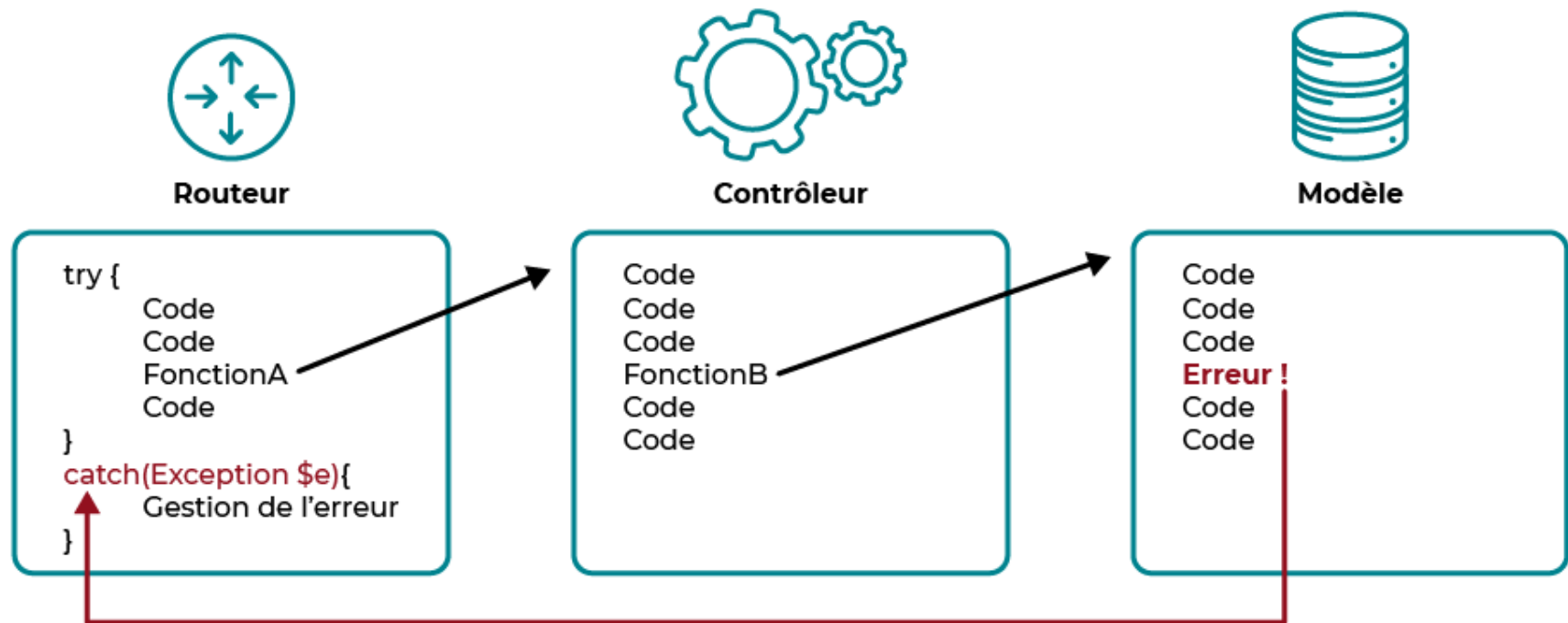
- *Shell* Unix (~/.bashrc, ~/.profile, etc.) :

```
export LC_ALL=en_US.UTF-8  
export LANG=en_US.UTF-8  
export LANGUAGE=en_US.UTF-8
```

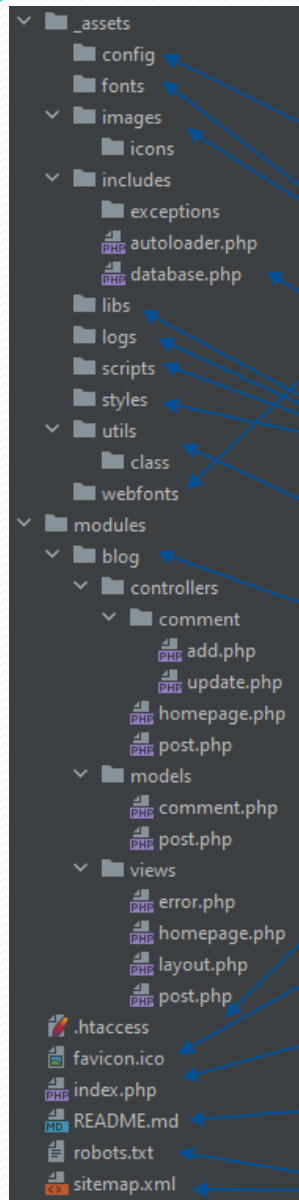


Encodage **identique** à tous niveaux (par exemple UTF-8)

MVC



Exemple d'architecture



Fichiers de configuration PHP personnalisés

Polices, images (et icônes) et polices Web (Comme Font Awesome)

Classes spécifiques et mutualisées (exceptions, bd, *autoloader*)

Bibliothèques externes, logs, scripts (JavaScript), styles (CSS)

Contient des fichiers de classes et de fonctions utilitaires ou *helpers*

Modules

Fichiers de classes de contrôleurs, modèles et vues (avec *layout*)

Fichier de configuration distribué (pour Apache)

Favicon par défaut

Index et routeur

Lisez-moi (notamment pour GitHub)

Directives et plan du site pour les robots d'indexation (de Google)

Exemple de routeur (index)

```
<?php
```

```
require '_assets/includes/autoloader.php';
```

```
try {  
    if (filter_input(INPUT_GET, 'action')) {  
        if ($_GET['action'] === 'post') {  
            if (filter_input(INPUT_GET, 'id') && $_GET['id'] > 0) {  
                (new \Blog\Controllers\Post\Post())->execute($_GET['id']);  
            }  
            throw new ControllerException('Aucun identifiant de billet envoyé');  
        }  
        throw new ControllerException('La page que vous recherchez n\'existe pas');  
    }  
    (new \Blog\Controllers\Homepage\Homepage())->execute();  
} catch (ControllerException $e) {  
    (new \Blog\Views\Error($e->getMessage()))->show();  
}
```

+ .htaccess

Exemple de contrôleur

```
<?php
namespace Blog\Controllers\Homepage;

require '_assets/includes/autoloader.php';

use Includes\Database\DatabaseConnection, Blog\Models\Post\PostRepository;

class Homepage
{
    public function execute(): void
    {
        $postRepository = new PostRepository(DatabaseConnection::getInstance());
        $posts = $postRepository->getPosts();
        (new \Blog\Views\Post($posts))->show();
    }
}
```

Exemple de modèle

```
<?php
```

```
namespace Blog\Model\Post; // PSR-12: head blocks must be separated by a single blank line
class PostRepository { // PSR-12: opening brace next line
    public function __construct(private \Includes\Database\DatabaseConnection $connection) {}

    public function getPosts(): array
    {
        if (!$statement = $this->connection->getConnection()->query('SELECT id, title,
content, creation_date FROM posts ORDER BY creation_date DESC LIMIT 0, 5'))
        {
            throw new DatabaseException('Wrong query');
        }
        $posts = [];
        while ($row = $statement->fetch(PDO::FETCH_OBJ)) {
            $post = new Post($row->id, $row->title, $row->creation_date, $row->content);
            $posts[] = $post;
        }
        return $posts;
    }
}
```

Exemple de vue

```
<?php
```

```
namespace Blog\View\Homepage; // PSR-12: head blocks must be separated by a single blank line
```

```
class Homepage { // PSR-12: opening brace next line
```

```
    public function show(): void { // PSR-12: opening brace next line
```

```
        ob_start();
```

```
?><h1>Les derniers billets du blog</h1>
```

```
<?php foreach ($posts as $post) { ?>
```

```
    <div class="news">
```

```
        <h3><?= htmlspecialchars($post->getTitle()); ?><em>: <?= $post->getDate(); ?></em></h3>
```

```
        <p>
```

```
            <?= nl2br(htmlspecialchars($post->getContent())); ?><br>
```

```
            <em><a href="index.php?action=post&id=<?= urlencode($post->getId() ?>">+</a></em>
```

```
        </p>
```

```
    </div>
```

```
<?php
```

```
    }
```

```
    $content = ob_get_clean();
```

```
    (new \Blog\Views\Layout('Le meilleur blog', $content))->show();
```

```
    }
```

```
}
```


Exemple de disposition (*layout*)

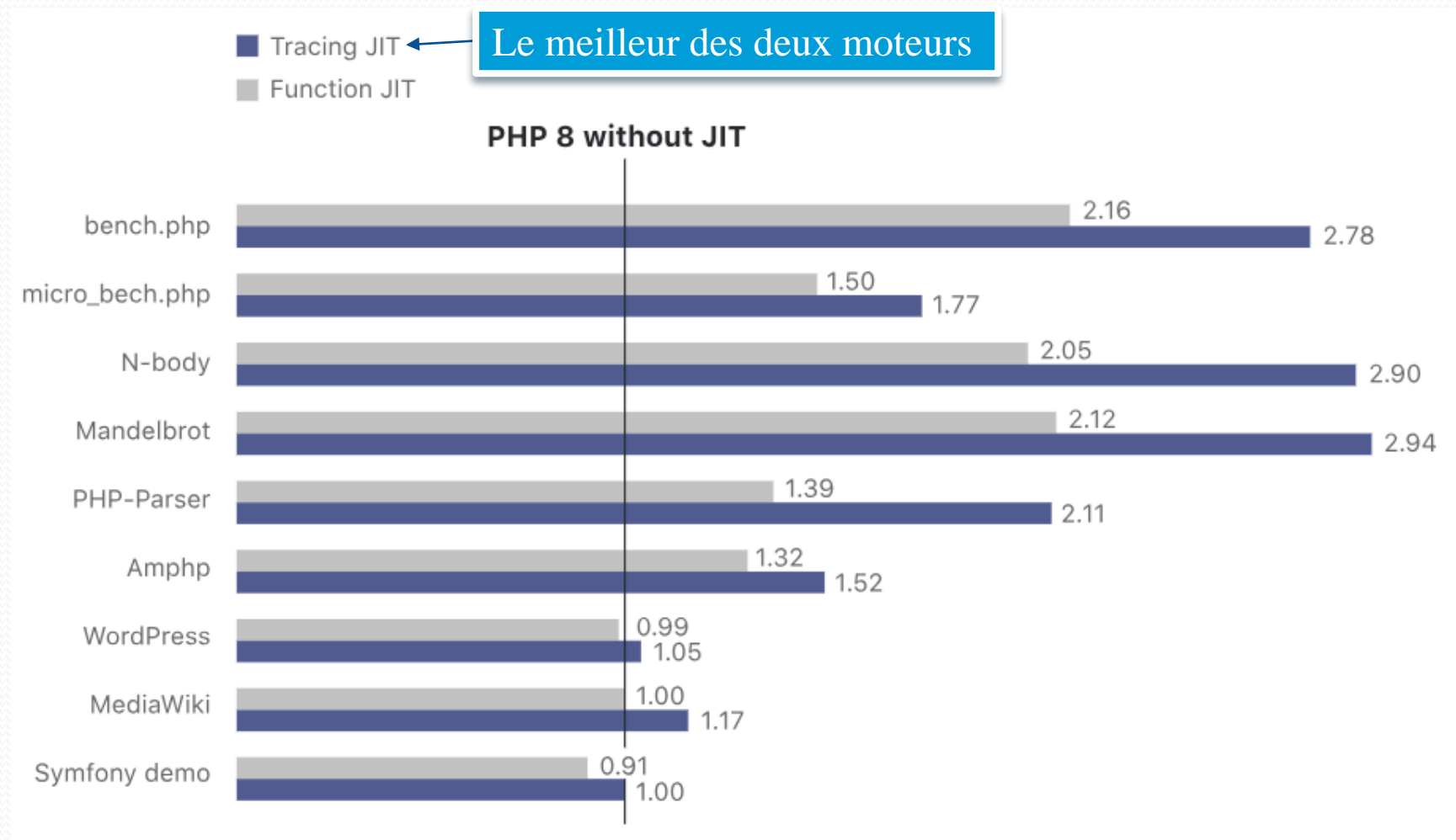
```
<?php
namespace Blog\View\Homepage;

class Homepage { // PSR-12: opening brace next line
    public function __construct(private string $title, private string $content) {}
    public function show(): void { // PSR-12: opening brace next line
?><!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
    <title><?= $this->title; ?></title>
    <link href="style.css" rel="stylesheet"/>
</head>
<body>
<?= $this->content; ?>
</body>
</html>

    <?php
    }
}
```

Avec un *layout*, plus vraiment besoin de blocs de page

Compilation



Utilisation de Tracing JIT

- `php.ini` :

```
opcache.enable = 1
opcache.enable_cli = 1
opcache.jit_buffer_size = 256M
opcache.jit = tracing
```

- Test :

```
var_dump(opcache_get_status()['jit']);
```

```
array:7 [
```

```
    "enabled" => true
```

```
    "on" => true
```

```
    "kind" => 5
```

```
    "opt_level" => 4
```

```
    "opt_flags" => 6
```

```
    "buffer_size" => 4080
```

```
    "buffer_free" => 0
```

```
]
```

Si enabled et on valent true, Tracing JIT fonctionne

Frameworks

- Espace de travail modulaire ← Également appelé **cadriciel**
- **Ensemble de bibliothèques, d'outils et de conventions permettant le développement d'applications**
- Permet de produire une application aboutie et **facile à maintenir**
- Composants organisés pour être utilisés en interaction les uns avec les autres
- Guide architectural
- Impose une rigueur de développement

Frameworks

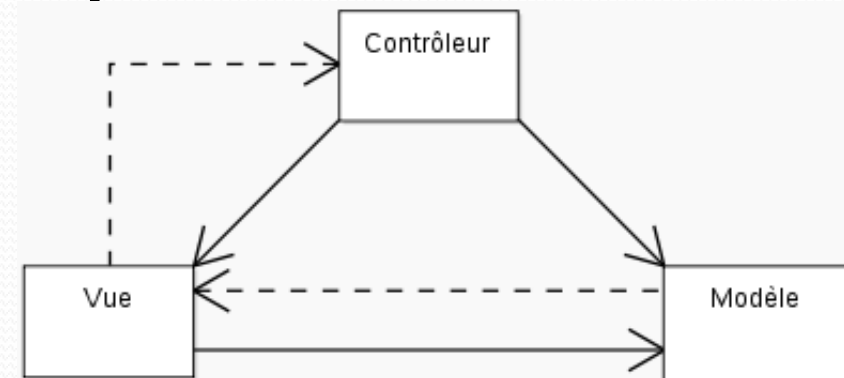
- Concepts fondamentaux :

- ORM** (*object-relational mapping*) : *mapping* objet-relationnel

Correspondance entre monde objet et monde relationnel

- Modèle **MVC** : Modèle-vue-contrôleur

- Architecture** et **méthode de conception** d'IHM
- Modèle** : données et leur manipulation
- Vue** : élément de l'interface graphique
- Contrôleur** : orchestre les actions, synchronise
- MVC 2** : un seul contrôleur



Frameworks

- Principaux *frameworks* PHP :
 - **Zend framework**
 - **Symfony**
 - **CakePHP**
- **Avantages :**
 - Maintenance facilitée
 - Architecture logicielle propre
- **Inconvénients :**
 - Nécessite de fortes compétences (emploi de spécialistes / ingénieurs)
 - Performances diminuées



Aller plus loin

- Sérialisation/désérialisation
- Iterator
- API de réflexion
- Standard PHP Library (SPL)
- Web 3.0 ?
 - Web sémantique
 - Web 3D
 - Web hors navigateur (applications de bureau, *smartphone*)
- Etc.

Liens

- Documents électroniques :
 - Tutoriels :
 - <http://apprendre-php.com>
 - Manuels :
 - <http://php.net/manual>
 - <http://fr.php.net/manual/fr/ref.mysql.php>
 - *Framework* :
 - <http://framework.zend.com>
 - <http://www.symfony-project.org>

Liens

- Documents classiques :

- Livres :

- Rasmus Lerdorf. *PHP : Précis et concis*.
 - Pascal Martin. *Développer une Extension PHP*.

- Articles :

- Alex Netkachov. *Optimize PHP memory usage: eliminate circular references*.
 - Dominique Albert. *PHP5 POO : déclarer une classe en PHP*.

Crédits

Auteur

Mickaël Martin Nevot

mmartin.nevot@gmail.com



Carte de visite électronique

Relecteur

- Christophe Delagarde
(christophe.delagarde@univ-amu.fr)
- Pierre-Alexis de Solminihac (pa@solminihac.fr)

Cours en ligne sur : www.mickael-martin-nevot.com

