Programmation Web côté serveur

CM4 : PHP « avancé »

Mickaël Martin Nevot

V2.4.0

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`

CNIL
Commission Nationale de l'Informatique et des Libertés

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
  - Assignation 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] =>
// )
```

- Insére
 automatiquement :

```
echo nl2br('Ceci\r\nest\n\rune\nchaîne\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**)

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

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

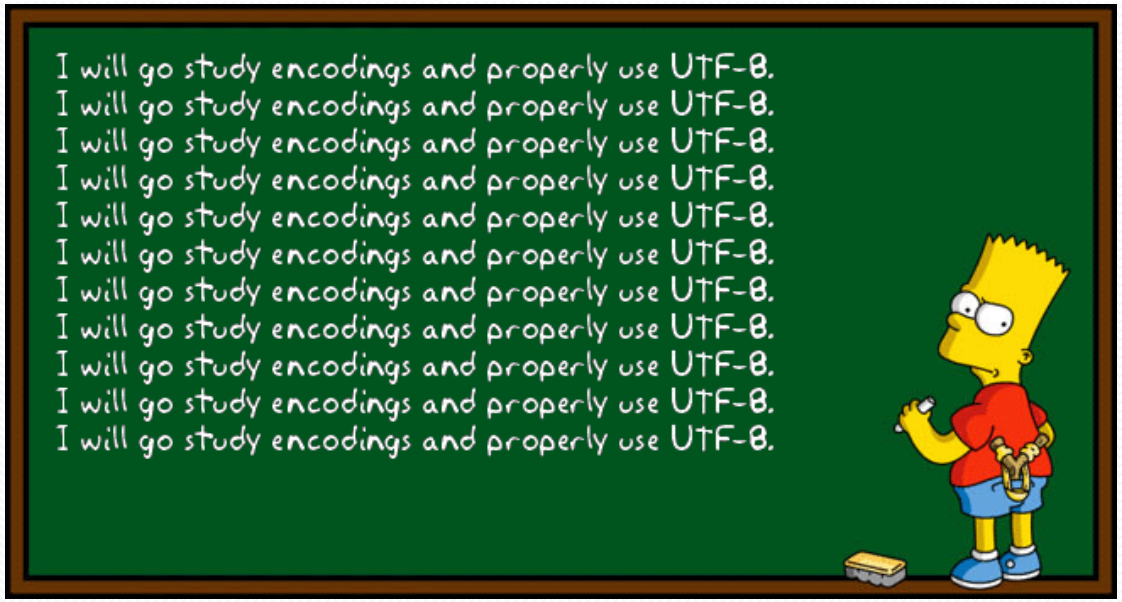
- PDO :

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

Codage de caractères

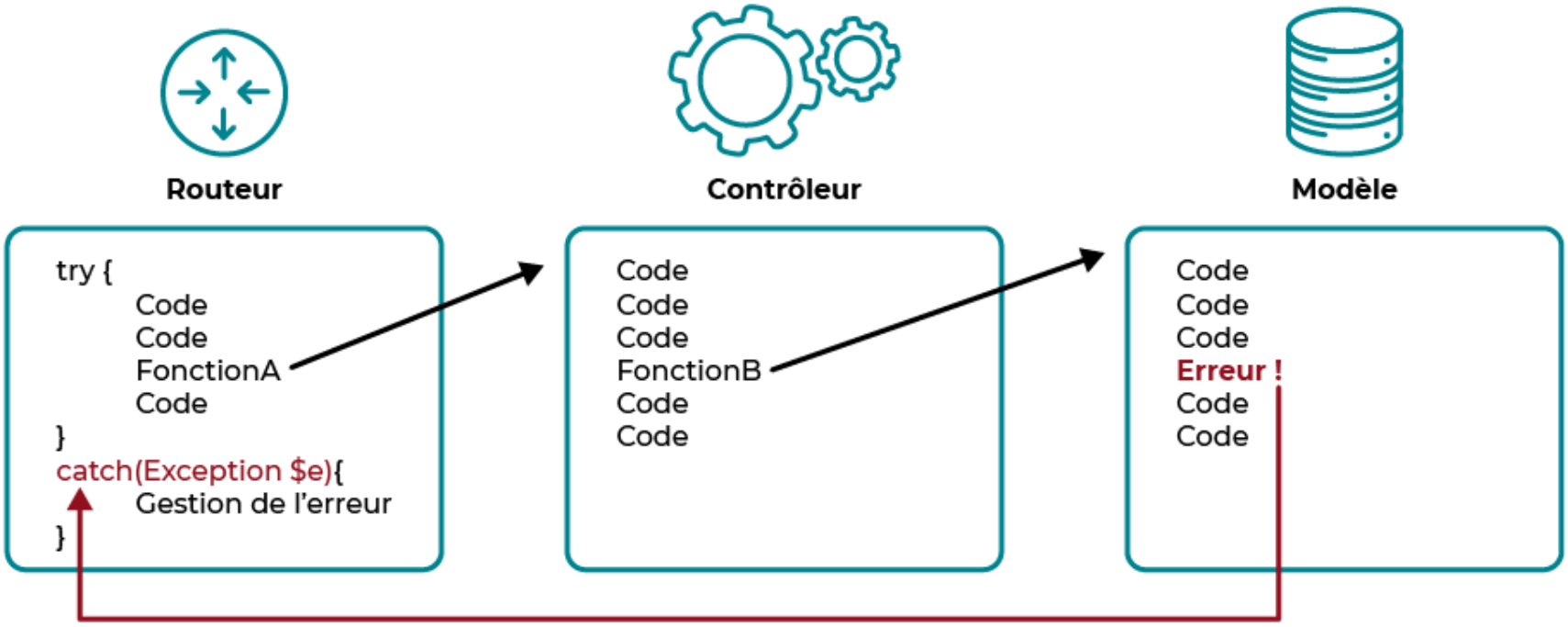
- *Shell* Unix (`~/.bashrc`, `~/.profile`, etc.) :

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

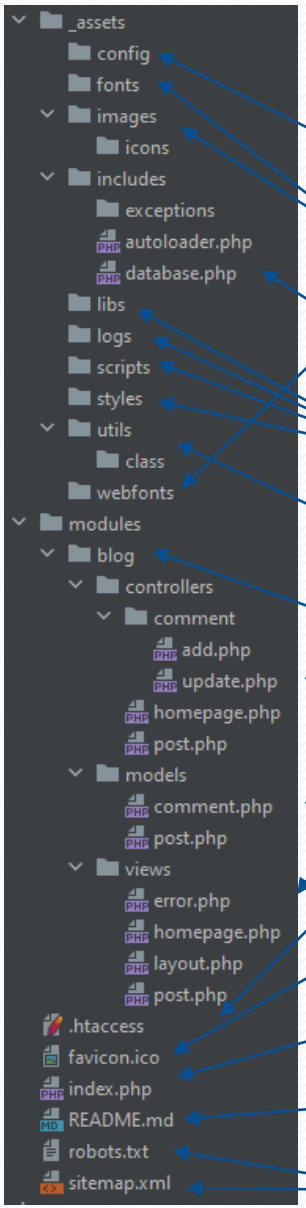


Encodage **identique** à tous niveaux (par exemple UTF-8)

MVC


Routeur
Contrôleur
Modèle
try {
Code
Code
FonctionA
Code
}
catch(Exception $e){
Gestion de l'erreur
}
Code
Code
Code
FonctionB
Code
Code
Code
Code
Code
Erreur !
Code
Code

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

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
<?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
<?php
namespace Blog\View; // 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 ($this->posts as $post) { /* PSR-12: opening brace next line */ ?>
    <div class="news">
        <h3><?= htmlspecialchars($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
        }
        (new \Blog\Views\Layout('Le meilleur blog', ob_get_clean()))->show();
    }
}
```

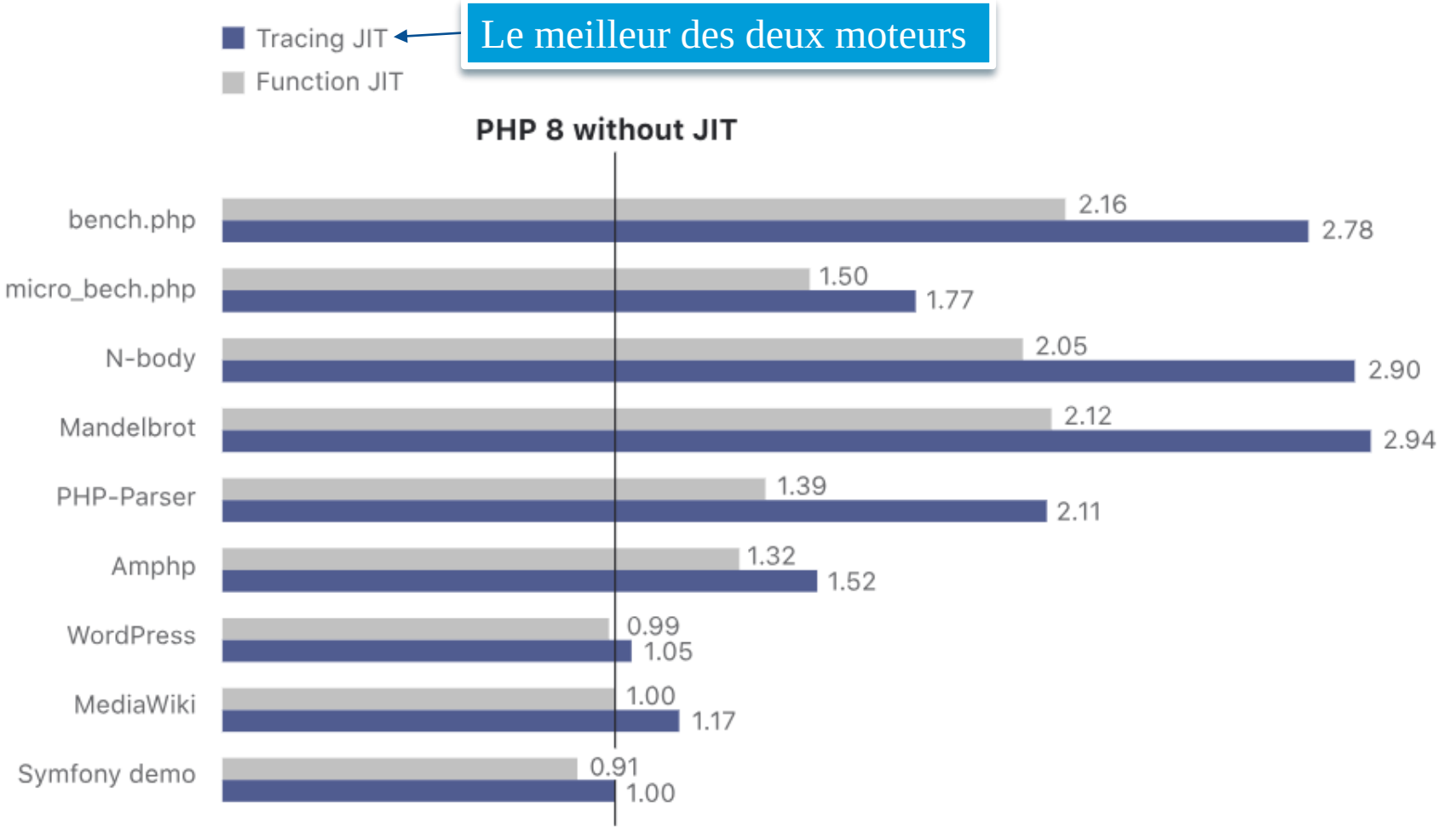
Exemple de disposition (*layout*)

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

class Layout { // 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

Index

- Structures de données, physiquement et logiquement indépendantes des données stockées dans la base
- Permet un accès direct (**rapide**) aux enregistrements
- Permet l'**optimisation de requêtes**
- Peut être composite (multi-attributs)
- Bonne utilisation :
  - Trouver le meilleur compromis entre :
    - Efficacité des requêtes
    - Coût d'exécution des mises à jour
    - Espace de stockage nécessaire

Cas d'utilisation d'index

- Attributs utilisés dans des **conditions de sélection** simples (c'est-à-dire sans `!=`, `IS NULL`, `NOT IN`, `LIKE`, `||`, une fonction de calcul horizontal ou vertical)
- Attributs utilisés pour des **jointures** (**clefs étrangères**, etc.)
- **Index composites** plutôt que plusieurs index simples
- Tous les attributs ayant une **forte cardinalité**
- Pas pour un attribut **volatile**
  (avec une fréquence de mise à jour des données élevée)
- Taille des données indexées importante :
  privilégier les attributs de type entier
- Attention à l'ordre de spécification des attributs indexés !

Création d'index

- Types d'index :
  - `PRIMARY KEY` : **index à valeur unique non nulle**
  - `KEY` : synonyme de `INDEX`
  - `INDEX` (simple ou composite) :
    - `INDEX index (name, firstname)`
  - `UNIQUE` : **index à valeur unique**

Un seul par table

Pouvant être nulle

- Création/destruction :
  - `CREATE INDEX` :

```
mysql> CREATE INDEX index ON table (col1, col2);
```

  - `DROP INDEX` :

```
mysql> DROP INDEX index ON table;
mysql> ALTER TABLE table DROP INDEX index;
```

Moteur/type de tables MySQL

MyISAM

Le plus utilisé

- Application :
  - Table en **lecture seule**, « log »
  - Recherche **plein texte**
- Avantages :
  - Moteur **rapide**
  - Gain de place sur disque
- Inconvénients :
  - Pas de gestion des contraintes de clefs étrangères
  - Pas de gestion de transactions (pas de `COMMIT/ROLLBACK` possibles)

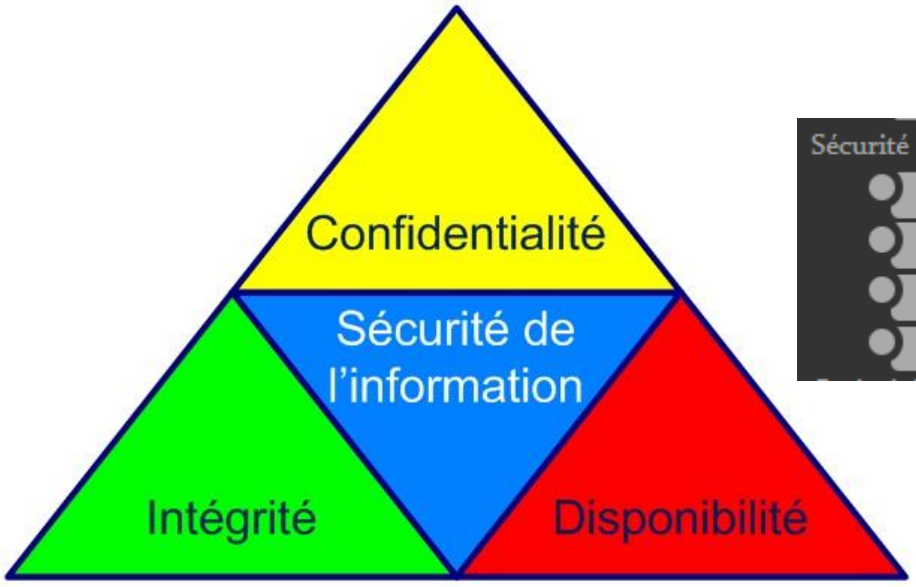
InnoDB

Par défaut à partir de MySQL 5.5

- Application :
  - Gestion des **transactions**
  - Fiabilité de l'information
- Avantages :
  - Gestion des **clefs étrangères**
  - Gère les gros volumes de données
- Inconvénients :
  - Lenteur de certaines opérations telles que `SELECT COUNT(*) FROM myTable`
  - `TRUNCATE` est synonyme de `DELETE`

Sécurité Web

- **Disponibilité** : maintenir le bon fonctionnement du système
- **Intégrité** : garantir que les données sont celles voulues
- **Confidentialité** : information inintelligible en dehors des acteurs de la transaction



Ressources dans le recueil des cours

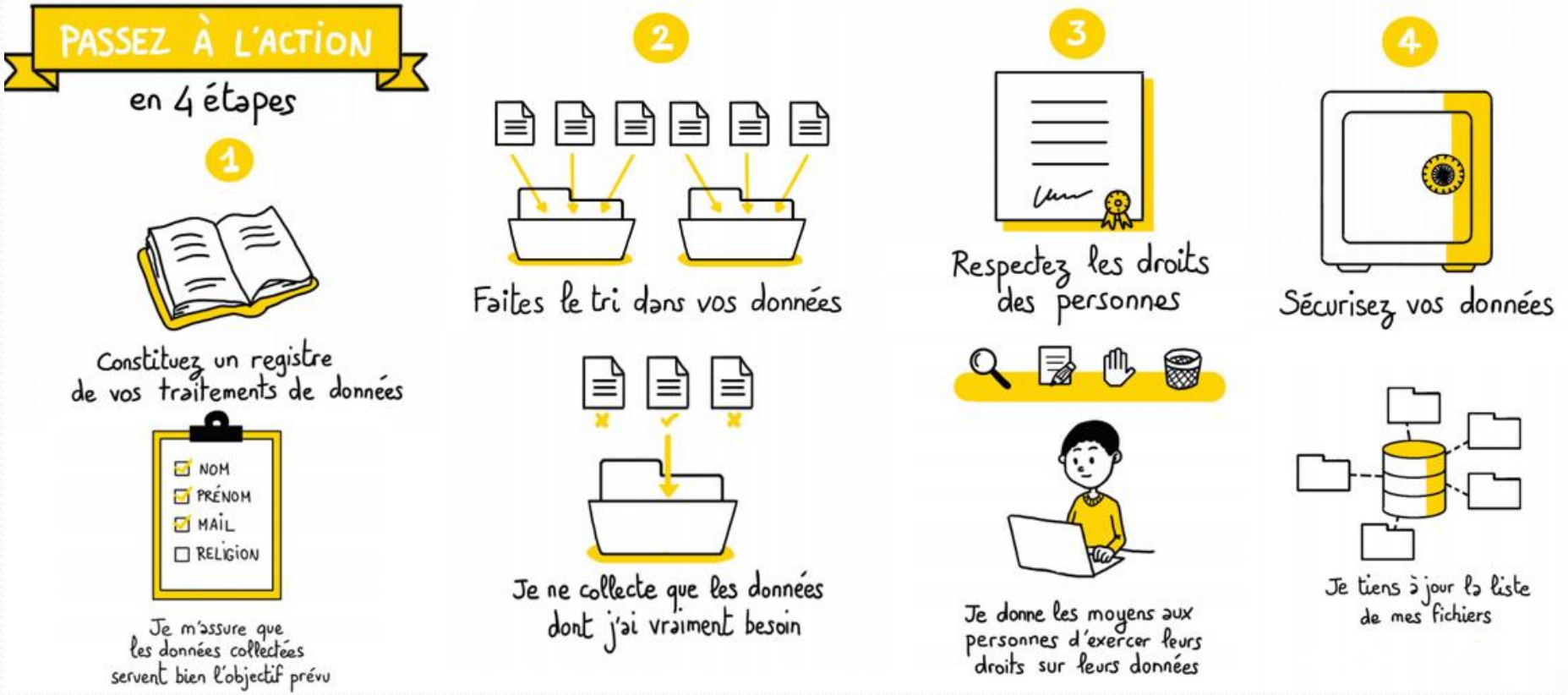
Sécurité
- Open Web Application Security Project (OWASP) : Top Ten
- Open Web Application Security Project (OWASP) : Cheat Sheet
- CVE (MITRE)
- Guide de sélection d'algorithmes cryptographiques (ANSSI)

RGPD

Depuis 2016

Par l'Union européenne, mais repris dans le monde entier

- Règlement général sur la protection des données



OWASP

OWASP
Open Web Application Security Project

1. **Broken Access Control** : droits d'accès
2. **Cryptographic Failures** : usurpations d'identité, CB, etc.
3. **Injection** : SQL, JavaScript, etc.
4. **Insecure Design** :
5. **Security Misconfiguration** : conf. serv. Web / *frameworks*
6. **Vulnerable / Outdated Components** : comp. tiers
7. **Ident. / Auth. Failures** : authentification, session, etc.
8. **Software / Data Integrity Failures** : intégrité des données
9. **Security Loggin / Monitoring Failures** : surveillance
10. **Server-Side Request Forgery (SSRF)** : DoS, RCE

Failles Web : analyse de site

- Partie visible :
  - **Statique** ou **dynamique** ? (y a-t-il une **URL *rewriting*** ?)
  - Les variables utilisées ? (méthode **GET** ou **POST**)
  - Les champs des formulaires ? Des **champs cachés** ?
  - Existence de ***cookies*** ?
  - Dossiers d'images, vidéos, etc. ?
  - Existence de **JavaScript** ?
  - Existence de **répertoires accessibles** ?
- Partie cachée :
  - Intercepter les échanges entre navigateur et serveur Web
  - Envoi massif de requêtes avec un ***fuzzer***

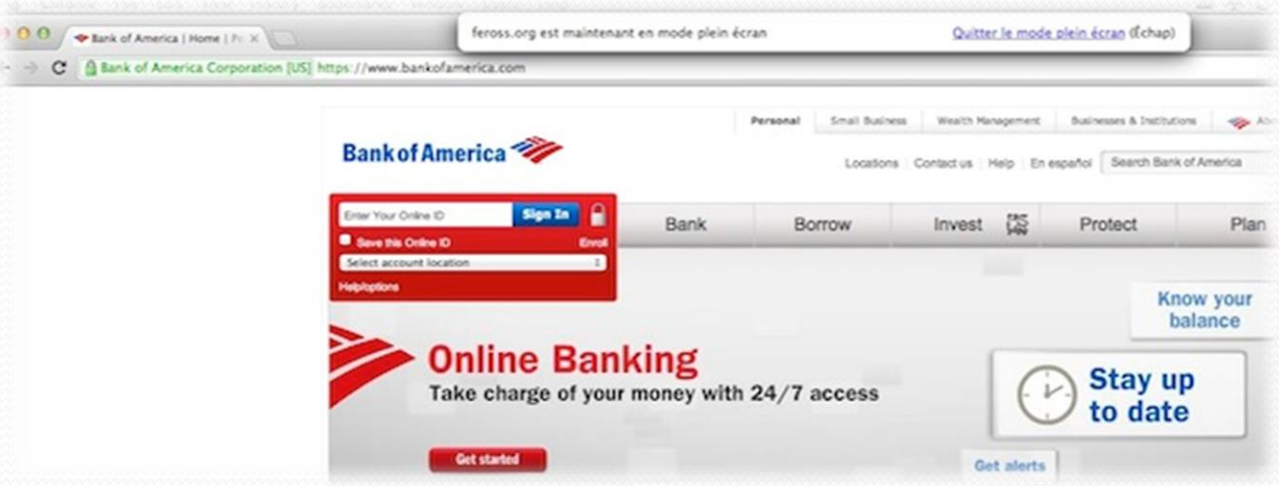
Failles Web : attaques

- Modification de chemins et URL :
  - Changement d'inclusion de fichier
  - **Injection de code JavaScript**
- **Injection SQL** (par l'intermédiaire des formulaires)
- Modification des **entêtes** (pirater une authentification)
- Modification des ***cookies***
- **Dépôt de fichiers** malicieux

Exemple : un virus à la place d'une image devant servir d'avatar sur un forum

Hameçonnage (Fishing)

- **Faux site Web** (usurpation d'identité)
- Utilisation *d'e-mails* et ***cross-site scripting*** **(XSS)**
- Cécité au changement :
  - Représentation lacunaire faite d'observations partielles

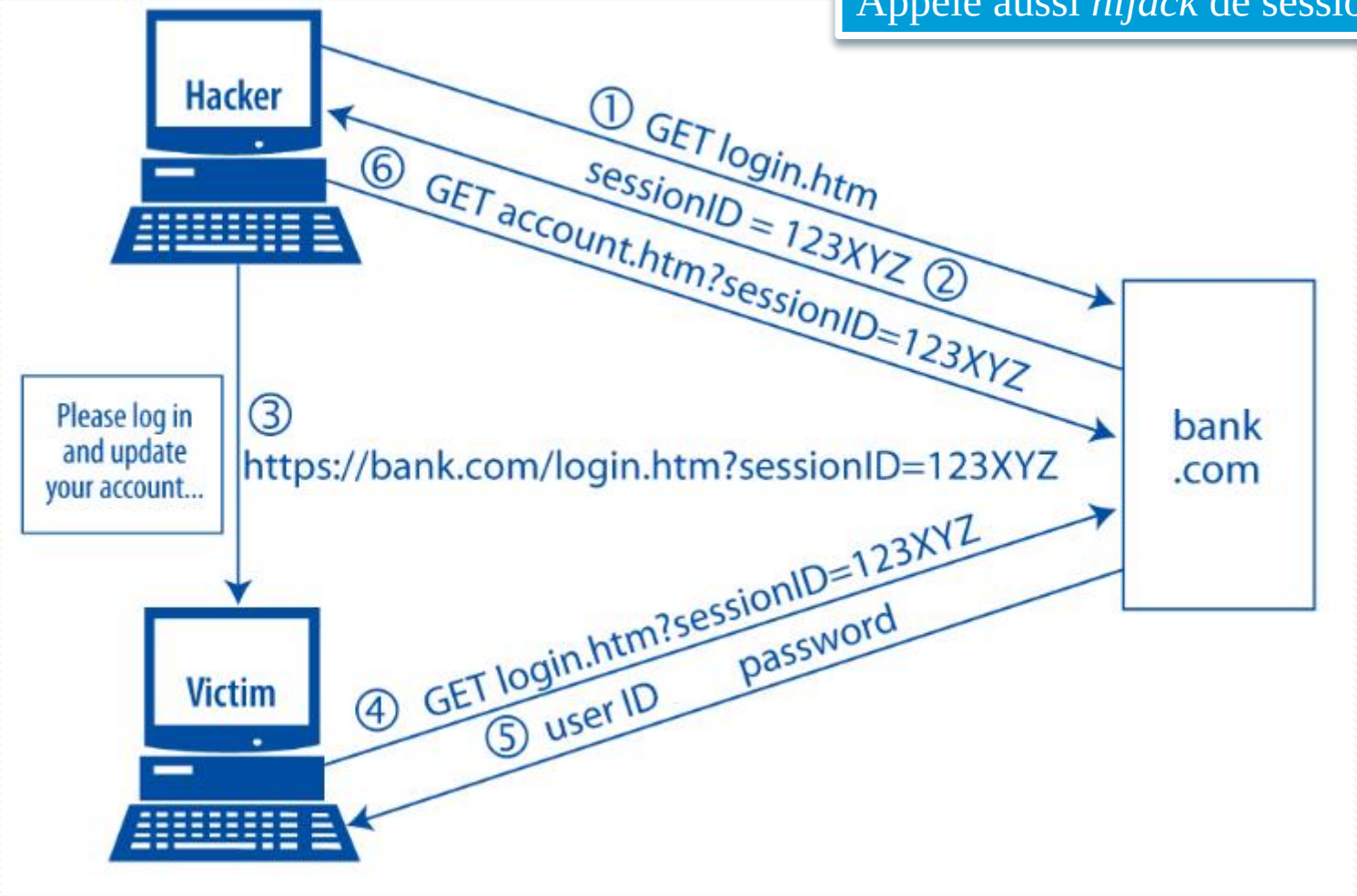


- http://feross.org/html5-fullscreen-api-attack/

Failles Web : CSRF

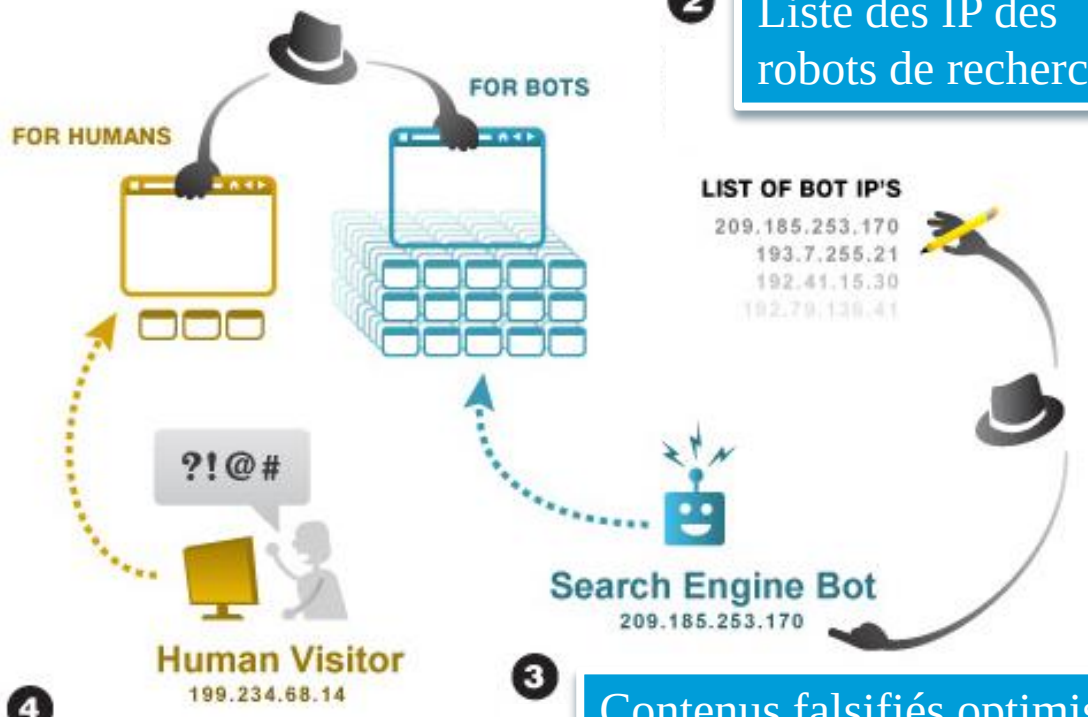
Cross-site request forgery

Appelé aussi *hijack* de session



Failles Web : Cloaking

1. Contenu différent pour un internaute ou un robot (filtrage par IP)



2. Liste des IP des robots de recherche

3. Contenus falsifiés optimisés pour le référencement

4. Contenu en décalage avec le référencement

Contrôle anti-injection

Sans PDO

```
/*
 * Fonction qui protège la variable passée en paramètre des injections SQL
 * et des caractères spéciaux.
 *
 * @author Mickaël Martin Nevot
 */
function quote_smart($value) {
    $value = utf8_encode($value);

    // Protection concernant Stripslashes
    if (get_magic_quotes_gpc()) {
        $value = stripslashes($value);
    }
    // Protection si ce n'est pas une valeur numérique ou une chaîne numérique
    if (!is_numeric($value)) {
        $value = '\'' . mysqli_real_escape_string($value) . '\'';
    }
    return $value;
}
```

Injection SQL insolite

SOC


SOC
Supervision en temps réel
Découverte d'événement et priorisation des réponses
Mise en place de réponses automatiques aux événements
Actions ou recommandations de remédiation
Audits détaillés
Evaluation des risques
Investigation approfondie sur des incidents ou des événements

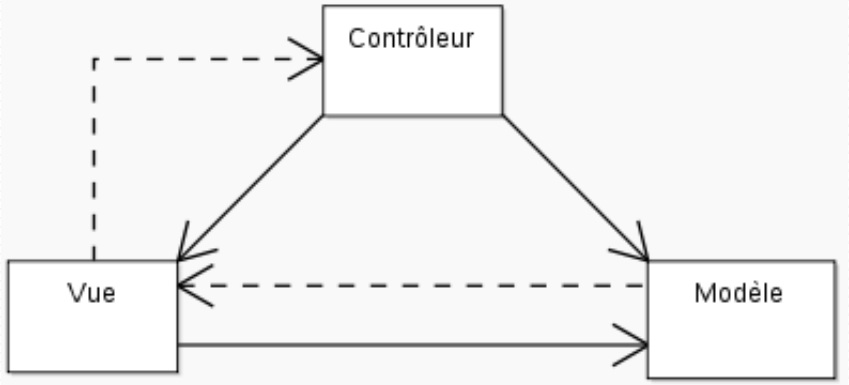
Frameworks

- 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**
  - PEAR
- **Avantages** :
  - Maintenance facilitée
  - Architecture logicielle propre
- **Inconvénients** :
  - Nécessite de fortes compétences (emploi de spécialistes / ingénieurs)
  - Performances diminuées

Aller plus loin

- PHPDoc
- Sérialisation/désérialisation
- Iterator
- API de réflexion
- Standard PHP Library (SPL)
- Alternatives (?) MVC : clean, onion, ddd, hexagonal, CQRS
- PHPMD
- Web 3.0 ?
  - Web sémantique
  - Web 3D
  - Web hors navigateur (applications de bureau, *smartphone*)

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