

## 2. Les bases de PHP

**Guillaume Kulakowski**

Expert technique en solutions Open-source (CGI)



# Agenda

1. Présentation du langage
2. Installation & configuration
3. Syntaxe
4. TD n°1
5. TP n°1
6. Programmation Orientée Objet (POO)
7. Normes & PSR
8. Les outils pour travailler avec PHP
9. TP n°2

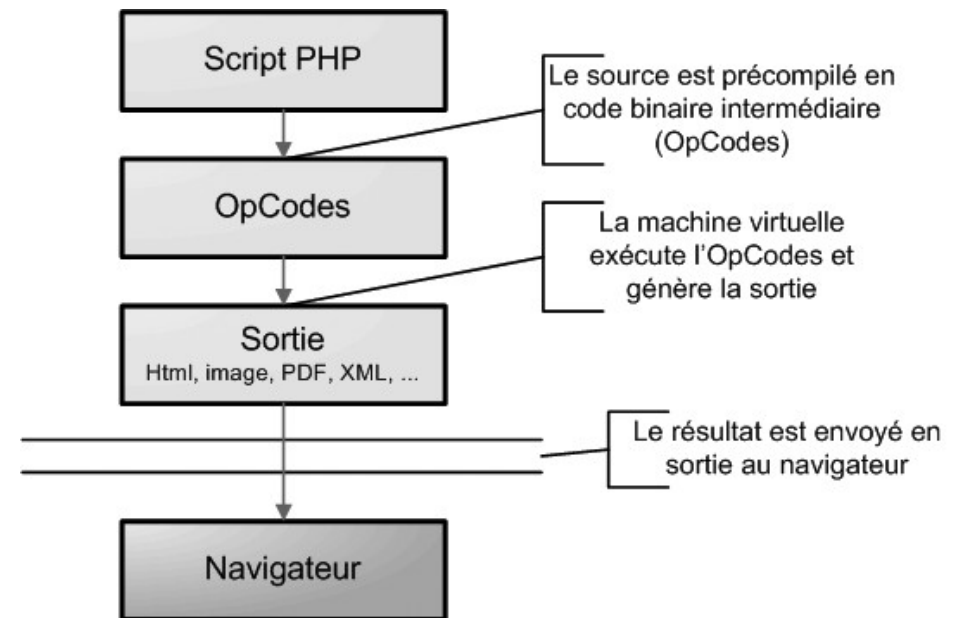
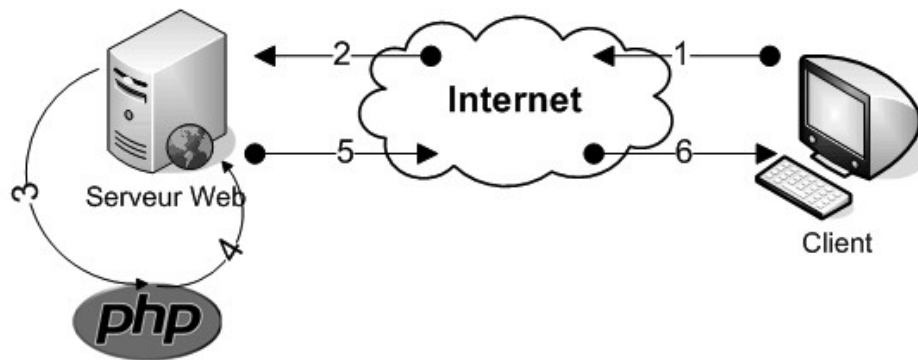
# 1 - Présentation du langage

- Les fondamentaux
- Le fonctionnement
- L'histoire
- PHP 6

# PHP en quelques mots

- Un langage Open Source : la Licence PHP [est approuvée par la FSF mais pas GPL compatible](#) , pas de frais de licence,
- Exécution via « *byte code* » recréé à chaque exécution (sauf si cache OPCode) pas de compilation,
- Langage à la prise en main rapide,
- Langage extensible (PECL, PEAR, PEAR2, Composer),
- Langage pour :
  - Applications serveur (sites web),
  - Scripts,
  - Web services,
  - Applications graphiques (php GTK, mais à déconseiller !),
- Langage à typage faible mais typable (de + en + typable avec PHP 7),
- Performant et fiable n'en déplaie à ses détracteurs. Le problème de php est d'être trop simple et de permettre à des personnes incompetentes de distribuer du code à la qualité / sécurité discutable !

# Fonctionnement de PHP



© PHP 5 avancé..

# L'histoire de PHP 1/3

- **La genèse de PHP :**
  - Créé en 1994 par Rasmus Lerdorf,
  - Bibliothèque en Perl,
  - Connaitre la consultation du CV de l'auteur,
  - **Personnal Home Page Tools.**
- **PHP/FI 1.0 :**
  - **Personal Home Page / Form Interpreter,**
  - Libération du code en 1995,
  - Migration de Perl en C,
  - 1997 : 1% des noms de domaines.
- **PHP/FI 2.0 :**
  - Novembre 1997.

# L'histoire de PHP 2/3

- **PHP 3.0 :**
  - Libération en 1998,
  - Refonte de PHP/FI plus qu'une suite,
  - Refonte qui fait suite au développement d'une application eCommerce qui rencontrait des problèmes de performances,
  - Par **Andi** Gustman et **Zeev** Suraski (fondateurs de Zend),
  - Première vraie version suffisamment stable pour la production,
  - Rasmus rejoint PHP qui devient donc la suite de PHP/FI,
  - **PHP: Hypertext Preprocess**,
  - Ajout du support des extensions,
  - 10 % du parc serveur.

# L'histoire de PHP 3/3

- **PHP 4.0 :**

- Libération en 2000,
- Réécriture totale du moteur pour améliorer les performances,
- Naissance du Zend Engine :
  - Analyseur syntaxique,
  - Procède en deux étapes : analyse puis exécution.

- **PHP 5.0 :**

- Libération en 2004,
- Simplification & professionnalisation,
- Zend Engine II avec intégration complète de la POO,
- Refonte du moteur XML,
- SQLite,
- PDO.



# PHP 6, un rendez-vous manqué

- Support de l'unicode,
- Besoin de refondre toutes les extensions,
- Divergence forte entre la branche 5 et 6 (trunk) : les améliorations apportées à la branche de développement de php 5.3 pas toutes mergées dans PHP 6,
- Division des équipes et des forces en présence,
- Décision en 2010 d'arrêter le développement de PHP 6 :
  - Le trunk devient branche php6 (PHP était encore sous SVN à l'époque),
  - Branche php5.3 devient trunk.

# L'après PHP 6, PHP 5.3+

- **PHP 5.3 :**

- Support des espaces de nom,
- Phar,
- Closure,
- GC.

- **PHP 5.4 :**

- Trait,
- Serveur HTTP.

- **PHP 5.5 :**

- Libération de Zend OPcache,
- Support mysql obsolète préférer mysqlnd ou mysqli.

- **PHP 5.6 :**

- Nouveau déboguer.

# L'après PHP 6, PHP 7

- **PHP next a.k.a PHP 7 :**
  - Amélioration des performances (x2).
  - Conversion d'un grand nombre de « fatal errors » en « Exceptions ».
  - Typage des retours.
  - Support des scalaires dans les déclarations.
  - Classes anonymes.

# 2 – Installation & configuration

- LAMP
- WAMP
- MAMP

# Sous Linux

- Plusieurs serveurs Web peuvent exécuter du PHP :
  - **Apache**,
  - LightHTTP,
  - Nginx,
  - PHP.
  - Etc...
- Plusieurs versions de php :
  - **mod\_php** pour apache,
  - php\_cgi,
  - php\_fcgi,
  - **php\_fpm**.
- Compatible avec plusieurs SGBD :
  - MySQL / **Maria**,
  - SQLite,
  - PostgreSQL,
  - Oracle,
  - Etc...
- Via le gestionnaire de paquets (yum, apt, etc...).

# Sous Windows

- Pour le DEV et uniquement pour le DEV !
  - **WAMP**,
  - **EasyPHP**,
  - **XAMPP**,
  - Etc...
- Pour la production :
  - Utiliser un serveur Linux si possible...
  - Sinon, installation / configuration à la main.

# Sous Mac

- Pour le DEV et uniquement pour le DEV !
  - [MAMP](#).
- Pour la production :
  - Utiliser un serveur Linux si possible...
  - Sinon, installation / configuration à la main.

# En développement

- Configurer PHP pour afficher les erreurs :

```
display_errors = On  
display_startup_errors = On  
error_reporting = E_ALL  
html_errors = On
```

- Activer et user et abuser d'[Xdebug](#) :

```
xdebug.default_enable = 1  
xdebug.max_nesting_level = 200
```



# 3 – Syntaxe

- Les bases
- PHP au sein d'HTML
- Hello World
- Commentaires
- Variables

# Les bases

- Fichier .php (.php4 ou .php5 possible),
- Commence par `<?php` (Syntaxe `<%` et `<?` à proscrire),
- Se termine par `?>` ou par rien si en fin de fichier,
- Ouverture / fermeture à répétition possibles, conjointement à HTML par exemple,
- « ; » après chaque instruction,
- Si erreur de syntaxe : « *Parse error* ».
- S'exécute en ligne de commande ou à travers le navigateur (<http://localhost>).

# La syntaxe dans HTML

```
<html>
<head>
    <title><?php echo 'Titre'; ?></title>
</head>
<body>
    <h1>Titre niveau 1</h1>
    <?php
    echo "<p>ceci est du code PHP</p>";
    echo '<p>Là aussi !</p>';
    ?>
</body>
</html>
```

# Hello world

```
<?php
// forme la plus simple, recommandée
echo 'Hello World';
?>
```

```
<?php
echo 'Hello ', 'World';
?>
```

```
<?='Hello World';
```

```
<?php echo 'Hello World'; echo 'Comment allez-
vous ?'; echo 'Il fait beau non ?' ?>
```

# Les commentaires

- « // » ou « # » (déprécié)

// Commentaire sur une ligne

- « /\* \*/ »

/\* Commentaire  
Sur plusieurs lignes  
\*/

/\*\*  
 \* Commentaire multi lignes avec phpDoc  
 \*  
 \* @param \$username String: username.  
 \* @return bool  
 \*/

# Les variables

Correct	Incorrect	Explications
<code>\$variable1</code>	<code>\$Variable 1</code>	Contient des espaces
<code>\$variable</code>	<code>variable</code>	Une variable commence toujours par \$
<code>\$variable_double</code>	<code>\$variable-double</code>	Le signe - est interdit
<code>\$variable_email</code>	<code>\$test@yahoo.fr</code>	Les caractères @ et . sont interdits.
<code>\$test2</code>	<code>\$2test</code>	Une variable ne commence pas par un chiffre.

```
<?php
```

```
$variableFirst = 'Ma première variable, je suis sensible  
à la casse & je ne suis pas typée ! Ma portée est  
locale !' ;
```

# Jouer avec les variables

- Une variable est par défaut locale sauf si :
  - `$GLOBALS['maVariable']`
  - `global $maVariable`
- On peut tester si une variable est définie avec *isset()* et détruire avec *unset()* :

```
$toTest = 'maVariable';  
if (isset($$toTest) === true) {  
    unset($maVariable) ;  
}
```

# Les constantes

- PHP possède de vraies constantes :

```
define('MA_CONSTANTE', 'En générale les  
constantes sont toutes en majuscule avec des  
« _ »') ;
```

- On peut tester la définition d'une constante :

```
if (!defined('ROOT_PATH')) {  
    define('ROOT_PATH', dirname(__FILE__)) ;  
}
```



# Les tableaux

- Peut être défini via `array()` :

```
$myArray = array(1, 2, 3, 4, 5);
```

```
$mySugarArray = [1, 2, 3, 4, 5];
```

- Même fonction pour les tableaux et les tableaux associatifs :

```
array('nom' => 'Kulakowski', 'prenom' => 'Guillaume') ;
```

- Peut-être initié puis affecté au besoin :

```
$myArray = array() ;
```

```
$myArray[] = 'Guillaume' ;
```

```
$myArray['prenom'] = 'Guillaume' ;
```

- Peuvent-être manipulés avec [tout un tas de fonctions](#).

# Les opérateurs d'affectation, d'arithmétiques et de concaténations

Opérateur	Opération	Exemple
=	Affectation	<code>\$i = 1</code>
+	Addition	<code>echo \$a + \$b</code>
-	Soustraction	<code>echo \$a - \$b</code>
*	Multiplication	<code>echo \$a * \$b</code>
/	Division	<code>echo \$a / \$b</code>
%		<code>echo \$a % \$b</code>
++	Incrémentation	<code>\$i++</code> ou <code>++\$i</code>
--	Décrémentation	<code>\$i--</code> ou <code>--\$i</code>
X=	Opération puis affectation	<code>\$a = 10 ; \$a *= 5 ; // 50</code>
.	Concaténation	<code>echo 'Ici' . ' et' . ' là' ;</code>
.=	Concaténation et assignation	<code>\$a = 'Ici'; \$a .= ' et'; \$a .= ' là';</code>

# Les opérateurs de comparaison

Opérateur	Opération	Exemple
==	Égal à	<code>if (\$i == 1)</code>
<	Inférieur à	<code>if (\$i &lt; 1)</code>
>	Supérieur à	<code>if (\$i &gt; 1)</code>
<=	Inférieur ou égal à	<code>if (\$i &lt;= 1)</code>
>=	Supérieur ou égal à	<code>if (\$i &gt;= 1)</code>
!=	Différent de	<code>if (\$i != 1)</code>
===	Égal à en type et valeur	<code>\$a = 10 ; \$b = '10'; \$a == \$b ; // true \$a === \$b ; // false</code>
!==	Différent de en type ou en valeur	

# Les opérateurs logiques

Opérateur	Opération	Exemple
!	Négation	!\$a
&&	« Et » logique	if ( (\$a == \$b) && (\$c == \$d) )
	« Ou » logique	if ( (\$a == \$b)    (\$c == \$d) )
AND	« Et » logique	if ( (\$a == \$b) AND (\$c == \$d) )
OR	« Ou » logique	if ( (\$a == \$b) OR (\$c == \$d) )
XOR	« Ou » exclusif	if ( (\$a == \$b) XOR (\$c == \$d) )

# Les conditions

```
<?php
if ($a > $b) {
    echo "a est plus grand que b";
} elseif ($a == $b) {
    echo "a est égal à b";
} else {
    echo "a est plus petit que b";
}
```

# Switch

```
switch ($i) {  
    case 0:  
        echo "i égal 0";  
        break;  
    case 1:  
        echo "i égal 1";  
        break;  
    case 2:  
        echo "i égal 2";  
        break;  
}
```

# Syntaxe alternative

- Syntaxe utilisée conjointement au HTML (if, while, for, foreach et switch) :

```
<?php if ($a == 5): ?>
```

A égal 5

```
<?php endif; ?>
```

- Syntaxe rapide

```
$action = (empty($_POST['action'])) ? 'default' :  
$_POST['action'];
```

# Les boucles

```
$arr = array(1, 2, 3, 4);  
foreach ($arr as &$value) {  
    $value = $value * 2;  
}  
  
for ($i = 1; $i <= 10; $i++) {  
    echo $i;  
}
```

```
$i = 1;  
while ($i <= 10):  
    echo $i;  
    $i++;  
Endwhile;
```

```
$i = 0;  
do {  
    echo $i;  
} while ($i > 0);
```



# 4 – TD n°1

Voir « TD n°1.zip »

- Hello World

# TD n°1 : Hello World

- Créez un répertoire « *public\_html* » dans votre répertoire utilisateur. Ce répertoire, aussi représenté par `~/public_html`, sera votre racine apache.
- Décompressez « TD n°1.zip » dedans.
- Rendez-vous à l'adresse <http://localhost/~user/TD>.
- Si tout ce passe bien vous devriez voir un « Hello world » de trois façons différentes :
  - Procédurale,
  - Fonctionnelle,
  - Objet.

# 5 – TP n°1

Voir « TP n°1.zip »

- Acronyme
- Conjecture de Syracuse
- Lecture de fichier

# TP n°1 : Les bases

- Décompressez « TP n°1.zip » dans votre racine apache.
- Rendez-vous à l'adresse [http://localhost/~user/TP\\_1](http://localhost/~user/TP_1)
- Laissez-vous guider par ce qui s'affiche dans le index.php.

# 6 – POO

- Classes
- Héritage
- Design Pattern
- PDO

# Les classes

```
<?php
class SimpleClass
{
    // déclaration d'une propriété
    public $var = 'une valeur par défaut';

    // Déclaration d'une constante de classe
    const CONSTANT = 'valeur constante';

    // déclaration des méthodes
    public function displayVar() {
        echo $this->var;
    }
}
```

# Création d'une instance

```
<?php
$instance = new SimpleClass();

// Ceci peut également être réalisé avec une
variable :
$className = 'Foo';
$instance = new $className(); // Foo()

?>
```

# Héritage

- Comme tout langage objet, PHP permet l'héritage, l'implémentation d'interface, etc...

```
<?php
class ExtendClass extends SimpleClass
{
    // Redéfinition de la méthode parente
    function displayVar()
    {
        echo "Classe étendue\n";
        parent::displayVar();
    }
}
```

```
$extended = new ExtendClass();
$extended->displayVar();
```



# Constructeur et destructeur

- L'approche objet de php permet les `__construct()` et `__destruct()` :

```
<?php
class BaseClass {
    function __construct() {
        print "In BaseClass constructor\n";
    }
}

class SubClass extends BaseClass {
    function __construct() {
        parent::__construct();
        print "In SubClass constructor\n";
    }
}
```

# Typage

- Il est possible de typer la signature d'une fonction ou d'une méthode :

```
<?php
// Un exemple de classe
class MaClasse {
    public $var = 'Bonjour le monde!';
}

/**
 * Fonction de test
 *
 * Le premier paramètre doit être un objet de type MaClasse
 */
function maFonction(MaClasse $foo) {
    echo $foo->var;
}

$maclasse = new MaClasse;
maFonction($maclasse);
```

# Autoloading

- Introduit par PHP 5,
- Permet le chargement dynamique de fichier à partir du nom de la classe.

```
<?php  
function __autoload($class_name) {  
    include classes/$class_name . '.php';  
}
```

```
$obj  = new MaClasse1();  
$obj2 = new MaClasse2();
```

# Design pattern : Singleton

```
<?php
class Singleton {
    private static $_instance;

    /**
     * Empêche la création externe d'instances.
     */
    private function __construct () {}

    /**
     * Empêche la copie externe de l'instance.
     */
    private function __clone () {}

    /**
     * Renvoi de l'instance et initialisation si nécessaire.
     */
    public static function getInstance () {
        if (!(self::$_instance instanceof self))
            self::$_instance = new self();

        return self::$_instance;
    }

    /**
     * Méthodes dites métier
     */
    public function uneAction () {}
}

// Utilisation
Singleton::getInstance()->uneAction();
```

# Design pattern : Factory

- La méthode Factory vise donc à instancier des objets selon certains arguments, en dissimulant les détails de cette instantiation.

```
<?php
class MaClasse {
    public static function chargerDriver($driver) {
        if (include_once 'Drivers/' . $driver . '.php') {
            $nomDeClasse = 'Driver_' . $driver;
            return new $nomDeClasse;
        } else {
            throw new Exception ('Driver non trouvé');
        }
    }
}

$mysql = MaClasse::chargerDriver('MySQL');
$sqlite = MaClasse::chargerDriver('SQLite');
```

# PDO

- PHP Data Objects est une couche d'abstraction aux bases de données.
- Compatible avec plusieurs SGBD via des drivers : <http://fr2.php.net/manual/en/pdo.drivers.php>

```
<?php
```

```
// Connexion à la BDD
```

```
$connection = new PDO("mysql:host=$host;dbname=$dbName, $dbUser, $dbPwd);
```

```
// On va chercher tous les membres de la table qu'on trie par ordre croissant
```

```
$results = $connexion->query("SELECT membre
```

```
    FROM membres
```

```
    ORDER BY membre ASC");
```

```
$resultats->setFetchMode(PDO::FETCH_OBJ);
```

```
// On récupère la liste des membres
```

```
while( $ligne = $resultats->fetch() )
```

```
{
```

```
    echo 'Utilisateur : '.$ligne->membre.'<br />';
```

```
}
```

```
$resultats->closeCursor(); // on ferme le curseur des résultats
```

# 7 – Normes et PSR

- PSR-0
- PSR-1
- PSR-2
- PSR-3, 4 & next...

# Introduction

- PHP est un langage anarchique car il utilise plusieurs conventions :
  - Underslashed : *file\_get\_contents()*,
  - Tout attaché : *strtolower()*,
  - Camel case : *DateTime::setISODate()*.
- PHP est permissif,
- Bref, nous avons besoin de norme !
- Les PSR permettent l'interopérabilité entre les framework.
- Plus d'infos : <http://www.php-fig.org/>



# PSR-0

- Les classes et les espaces de noms entièrement qualifiés doivent disposer de la structure suivante `\<Nom du Vendor>\(<Espace de noms>\)*<Nom de la Classe>`.
- Chaque espace de noms doit avoir un espace de noms racine. ("Nom du Vendor").
- Chaque espace de noms peut avoir autant de sous-espaces de noms qu'il le souhaite.
- Chaque séparateur d'un espace de noms est converti en `DIRECTORY_SEPARATOR` lors du chargement à partir du système de fichiers.
- Chaque `"_"` dans le nom d'une CLASSE est converti en `DIRECTORY_SEPARATOR`. Le caractère `"_"` n'a pas de signification particulière dans un espace de noms.
- Les classes et espaces de noms complètement qualifiés sont suffixés avec `".php"` lors du chargement à partir du système de fichiers.
- Les caractères alphabétiques dans les noms de vendors, espaces de noms et noms de classes peuvent contenir n'importe quelle combinaison de minuscules et de majuscules.

<http://www.php-fig.org/psr/psr-0/>

# PSR-1 (La norme de codage de base)

- Les fichiers DOIVENT utiliser seulement les tags `<?php` et `<?='`.
- Les fichiers de code PHP DOIVENT être encodés uniquement en UTF-8 sans BOM.
- Les fichiers DEVRAIENT soit déclarer des symboles (classes, fonctions, constantes, etc.) soit causer des effets secondaires (par exemple, générer des sorties, modifier paramètres .ini), mais NE DOIVENT PAS faire les deux.
- Les espaces de noms et les classes DOIVENT suivre PSR-0.
- Les noms des classes DOIVENT être déclarés comme StudlyCaps.
- Les constantes de classe DOIVENT être déclarées en majuscules avec un sous-tiret en séparateurs.
- Les noms des méthodes DOIVENT être déclarés comme camelCase.

<http://www.php-fig.org/psr/psr-1/>

# PSR-2 (Guide pour le style d'écriture de code)

- Le code DOIT suivre les PSR-1.
- Le code DOIT utiliser 4 espaces pour l'indentation et aucune tabulation.
- Il NE DOIT PAS exister une limite stricte sur la longueur de la ligne, la limite acceptable DOIT être de 120 caractères; les lignes DEVRAIENT comprendre 80 caractères ou moins.
- Il DOIT y avoir une ligne vide après la déclaration de l'espace de noms, et il DOIT y avoir une ligne vide après le bloc de déclarations use.
- L'ouverture des accolades pour les classes DOIT figurer sur la ligne suivante, les accolades de fermeture DOIVENT figurer sur la ligne suivante après le corps de la classe.
- L'ouverture des accolades pour les méthodes DOIT figurer sur la ligne suivante, les accolades de fermeture DOIVENT figurer sur la ligne suivante après le corps de la méthode.
- La visibilité DOIT être déclarée sur toutes les propriétés et méthodes; abstraite et finale doivent être déclarés avant la visibilité; statique DOIT être déclaré après la visibilité.
- La structure des mots-clés de contrôle DOIT avoir un espace après eux, les méthodes et les appels de fonction NE DOIVENT PAS en avoir.
- L'ouverture des accolades pour les structures de contrôle DOIT figurer sur la même ligne, et la fermeture des accolades DOIT figurer sur la ligne suivante après le corps.
- l'ouverture des parenthèses pour les structures de contrôle NE DOIT PAS contenir d'espace après eux, la fermeture de parenthèses pour les structures de contrôle NE DOIT PAS contenir d'espace avant.

<http://www.php-fig.org/psr/psr-2/>

# PSR-3, 4 et next...

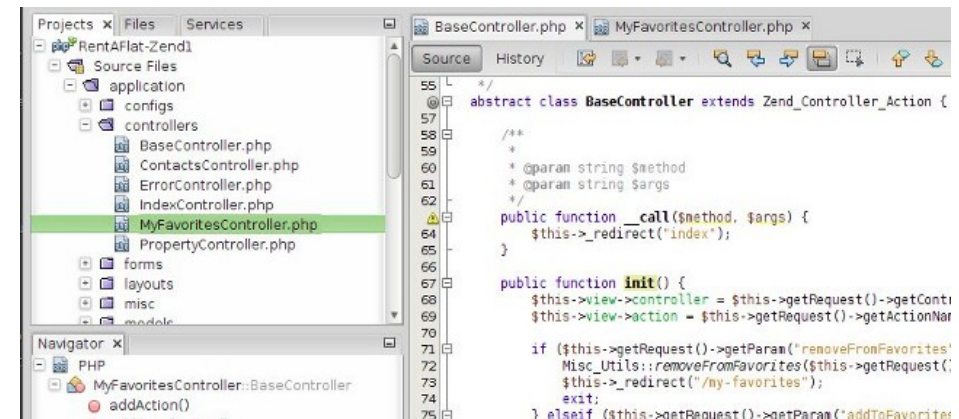
- Les PSR précédentes ont bien définies les standards de développements PHP.
- De nouvelles PSR ont été validées depuis :
  - PSR-3 : [Logger Interface](#).
  - PSR-4 : [Improved Autoloading](#).
  - PSR-7 : [HTTP message interfaces](#).
- Les prochaines (proposed) :
  - [Cache](#).
  - [Container](#).
  - [Extended Coding Style Guide](#).
  - [HTTP message interfaces](#).
  - [Etc...](#)

# 8 – Les outils

- Netbeans
- Eclipse
- PHPMyAdmin
- MySQL Work Bench

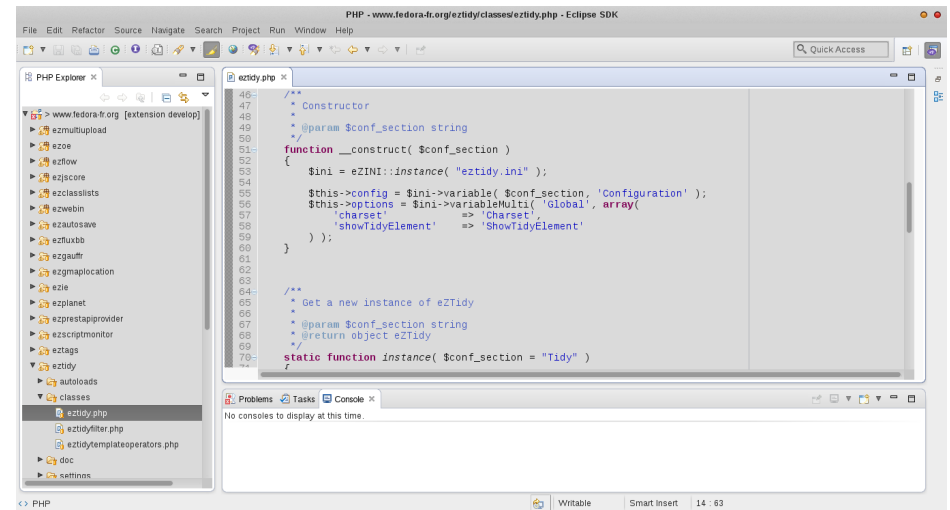
# EDI : Netbeans

- Site officiel de Netbeans :  
<http://netbeans.org>
- Support de PHP natif :  
<https://netbeans.org/features/php/>



# EDI : Eclipse

- Site officiel d'Eclipse :  
<http://eclipse.org/>
- PDT pour PHP Development Tools, l'extension qui apporte le support de PHP :  
<http://projects.eclipse.org/projects/tools.pdt>



# Autres EDI

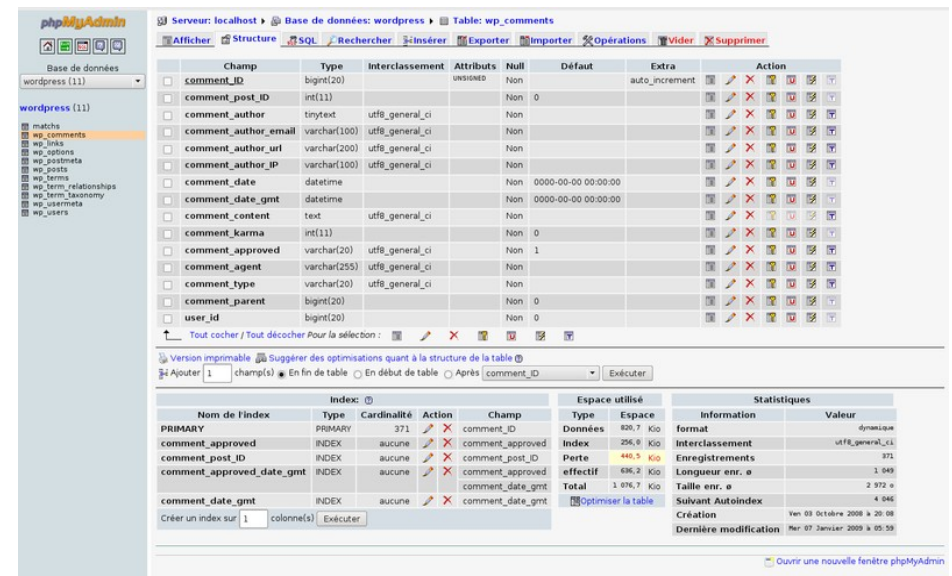
Bien entendu il existe moult autres Editeurs de texte / EDI :

- Notepad++
- Sublime Text
- Zend Studio
- Etc...



# PHPMyAdmin

- WebUI pour manipuler des bases de données MySQL / MariaDB,
- Open Source,
- Intégré à WAMP.



The screenshot displays the PHPMyAdmin web interface. On the left, a sidebar shows the database structure for 'wordpress (11)', with 'wp\_comments' selected. The main area shows the 'Structure' tab for the 'wp\_comments' table. It lists 14 columns with their respective types, attributes, and actions. Below this, there's a section for 'Index' and 'Espace utilisé' (Space used). The 'Index' table shows the primary key 'comment\_ID' and three other indexes. The 'Espace utilisé' table shows the size of the data, indexes, and total space. At the bottom, there's a 'Statistiques' (Statistics) section with various metrics like format, interclassement, enregistrements, longueur, taille, and création/modification dates.

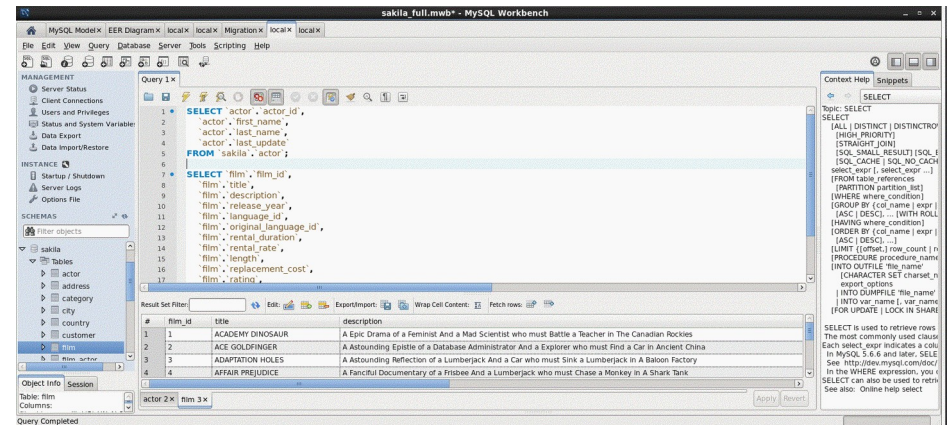
Champ	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
comment_ID	bigint(20)		unsigned	Non		auto_increment	
comment_post_ID	int(11)			Non	0		
comment_author	tinytext	utf8_general_ci		Non			
comment_author_email	varchar(100)	utf8_general_ci		Non			
comment_author_url	varchar(200)	utf8_general_ci		Non			
comment_author_ip	varchar(100)	utf8_general_ci		Non			
comment_date	datetime			Non	0000-00-00 00:00:00		
comment_date_gmt	datetime			Non	0000-00-00 00:00:00		
comment_content	text	utf8_general_ci		Non			
comment_karma	int(11)			Non	0		
comment_approved	varchar(20)	utf8_general_ci		Non	1		
comment_agent	varchar(255)	utf8_general_ci		Non			
comment_type	varchar(20)	utf8_general_ci		Non			
comment_parent	bigint(20)			Non	0		
user_id	bigint(20)			Non	0		

Index				Espace utilisé		Statistiques	
Nom de l'index	Type	Cardinalité	Action	Champ	Type	Espace	Information
PRIMARY	PRIMARY	371		comment_ID	Données	820,7 Kio	format
comment_approved	INDEX	aucune		comment_approved	Index	256,8 Kio	Interclassement
comment_post_ID	INDEX	aucune		comment_post_ID	Perte	440,5 Kio	Enregistrements
comment_approved_date_gmt	INDEX	aucune		comment_approved	effectif	696,2 Kio	Longueur enr. o
comment_date_gmt	INDEX	aucune		comment_date_gmt	Total	1 076,7 Kio	Taille enr. o

Statistiques	
Information	Valeur
format	utf8_general_ci
Interclassement	371
Enregistrements	1 049
Longueur enr. o	2 972 o
Taille enr. o	4 946
Suivant Autoindex	
Création	Ven 03 Octobre 2009 à 20:08
Dernière modification	Mer 07 Janvier 2010 à 05:58

# MySQL Work Bench

- Le client lourd (dans tous les sens du terme) pour manipuler des bases de données MySQL / MariaDB,
- Open Source.



# 9 – TP n°2

Voir « TP n°2.zip »

- Classe
- POO
- PDO

## TP n°2 : Plus loin...

- Décompressez « TP n°2.zip » dans votre racine apache.
- Rendez-vous à l'adresse [http://localhost/~user/TP\\_2](http://localhost/~user/TP_2)
- Laissez-vous guider par ce qui s'affiche dans le index.php.
- Concernant le db\_config.php, vous allez devoir vous créer un compte sur *Venus* pour avoir à disposition une base MySQL. Pour cela :
  - Aller sur Sif, Bloc « Mon Espace », menu « Mon compte ».
  - Tout en bas, allez dans la gestion des bases de données.
  - Voir les captures ci-dessous.



[Accueil](#) » [accueil](#)

### Derniers logiciels installés

[Android studio](#)

[SFML](#)

[Cog](#)

[Bio Python](#)

[Mozart-Oz](#)



1 sur 60

[suivant](#)

### Dernières modifications

[Bienvenue](#) 2 semaines 6 jours

[Accompagnement rentrée](#) 1 mois 1 semaine

[Gestion des commutateurs](#) 3 mois 1 semaine

[Environnement de travail](#) 3 mois 2 semaines

[espaces projets](#) 6 mois 2 semaines

### Recherche

[Rechercher](#)

### Acces libre

A 17 h 43 il y a 9 salle(s) disponible(s) **en libre service** : td6.02-td6.03-td6.10-td6.12-td6.13-td6.14-td6.15-td6.16-td6.17

### Création Compte

#### Procédure d'inscription

[Créer mon compte informatique](#)

[Questions fréquemment posées \(FAQ\)](#)

[À propos](#)

### Accès direct

- [Offre en ligne](#)
- [Candidature](#)
- [Infos rapides](#)

### Informatique

- [Intel-Micron 3D XPoint At Xroads](#)
- [Cooler Master MasterCase 5 Review](#)
- [Thecus W5000 WSS NAS Reivew](#)
- [Be Quiet! Dark Power Pro 11 1200W PSU Review](#)

[Voir](#)

[Modifier](#)

[Nœuds](#)

[Suivi](#)

#### Mot de passe actuel

The password cannot be changed using this website

#### Mot de passe

Sécurité du mot de passe :

#### Confirmer le mot de passe

Pour modifier le mot de passe actuel, saisissez le nouveau mot de passe dans les deux champs de texte.

#### Paramètres de langue

##### Langue

☐ Anglais (English)

☒ Français

La langue par défaut de ce compte pour les courriels.

### Mon espace

- [Mon compte](#)
- [Se déconnecter](#)

### Quota



Imprimées 93 Restantes 157

### Navigation

- [Compte](#)
- [Création de compte informatique - Formation](#)
- [Documents pédagogiques](#)

20585

### info\_seealso

uid=e20140029164,ou=people,dc=umontpellier,dc=fr

Contient le code étudiant

#### ▼ Base de données

##### ▼ Base mysql

###### Mysql

OUI ▼

Demande de création de compte Mysql

###### Mot de passe Mysql \*

Plus de 7 caractères.

- [Accès à phpMyadmin](#)
- ligne de commande : mysql -h venus -u login -p

##### ► Base Oracle

##### ► Base Postgresql

##### ► Base PostGis

#### ▼ SERVICES

###### MSDNAA

OUI ▼

Enregistrer



# Annexes

# Webographie

- Site officiel de PHP : <http://www.php.net/>
- Le manuel PHP : <http://php.net/manual/fr/index.php>
- OpenClassRooms : <https://openclassrooms.com/>



# Bibliographie

- PHP 5 avancé.  
Eric Daspet & Cyril Pierre de Geyer.  
(ISBN : 978-2-212-12369-2).