

Cours PHP

Introduction

PHP – Introduction

```
header("Content-Type: text/html;charset=UTF-8");  
header("Content-Type: text/html;charset=ISO-8859-1");
```

1 HTML et PHP

1.1 Généralités sur PHP

L'acronyme récursif de “**H**ypertext **P**reprocessor”.

PHP est un **générateur de code HTML**.

Un langage de script de programmation côté serveur.

Il est intégré à du code HTML.

Principalement utilisé pour produire des pages web, mais peut fonctionner également comme un langage de programmation d'objectif général.

PHP peut gérer des contenus de formulaires WEB, des graphiques, des cookies et des sessions, manipuler des fichiers texte ou autre, accéder à des bases de données.

PHP fonctionne sur de nombreux systèmes d'exploitation : **Linux, Unix, Microsoft, Mac OS X**, ...

PHP fonctionne sur de nombreux serveurs web : **Apache, IIS, PWS** (Microsoft Personal Web Server), **iPlanet Server, Netscape**, ...

PHP peut générer du texte, du graphisme, du Flash, du XML, du PDF à la volée et l'envoyer au navigateur.

PHP est orienté objet depuis la version 5.

PHP supporte de nombreux protocoles comme LDAP (Lightweight Directory Access Protocol), IMAP (Internet Message Access Protocol), SNMP (Simple Network Management Protocol), NNTP (Network News Transfer Protocol), POP3 (Post Office Protocol 3), HTTP (HyperText Transfer Protocol) ...

PHP supporte le format WDDX (Web Distributed Data eXchange), qui lui permet de communiquer avec d'autres langages web.

PHP supporte aussi les instanciations d'objets Java. (e.g. PHP/JAVA Bridge)

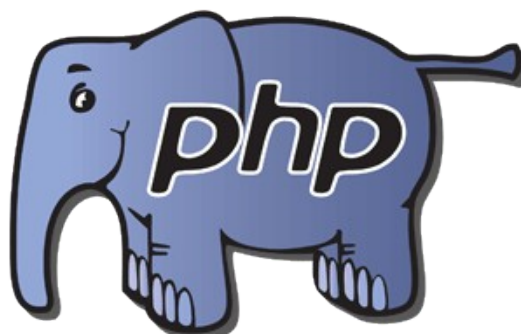


Fig 1.1 : L'ElePHPant, la mascotte

1.1.1 Histoires

- ≡ 1994. Le langage PHP est créé en 1994 par Rasmus Lerdorf, à partir du langage PERL puis réécrit en C, pour gérer sa page web personnelle
- ≡ 1995. La version 1.0 était sortie, son premier nom officiel est PHP Tools – l'abréviation de “**P**ersonal **H**ome **P**age Tools”.
- ≡ 1997. Version 3.0, il a été repris par Andi Gutmans et Zeev Suraski.
- ≡ 2004. Sortie de la version 5.0, le PHP supporte la programmation orientée objet (POO). Avant cette version, le PHP était un langage de programmation procédure.
- ≡ Le 3 Décembre 2015, sortie de la nouvelle version stable – version 7.0.

1.1.2 Avantage de PHP

- ≡ Gratuit,
- ≡ Fait pour le Web,
- ≡ Facile à apprendre,
- ≡ Portable,
- ≡ Disponibilité du code source (que vous pouvez donc modifier avant les releases),
- ≡ Haute performance,
- ≡ Principe des extensions,
- ≡ Support (Payant).

1.1.3 Nouveautés de PHP 5

- ≡ Une nouvelle approche POO.
- ≡ L'intégration de PECL (PHP Extensions C Libraries) : Extensions développées en C précédemment par PEAR.
- ≡ PEAR (pour PHP Extension and Application Repository) est un framework (ensemble de bibliothèques logicielles) de composants PHP créé par Stig S. Bakken en 1999.
- ≡ PDO (PHP Data Object).

PHP – Introduction

- ≡ SQLite,
- ≡ Trait, generators, ...

1.1.4 Usages

- ≡ Plus de 80% des pages web sont en PHP.
- ≡ La majorité des sites web PHP sont développées avec PHP 5.x.



Fig 1.2 : Sites en PHP

1.2 Le fonctionnement du WEB statique

Le web fonctionne en mode **Client/Serveur** – mode CS, via le protocole HTTP.

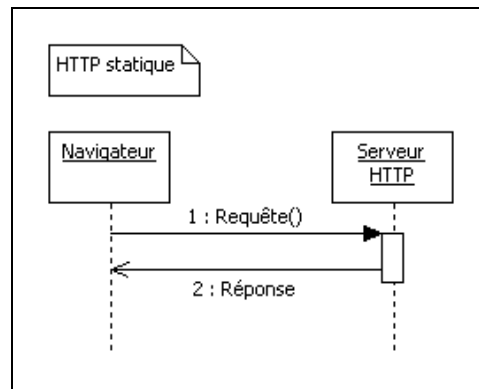


Fig. 1.3 : Architecture CS

Lorsque le client saisit dans la barre d'adresse de son navigateur une URL (Uniform Resource Locator) du type **http://www.serveur.com/site1/index.html** dans un premier temps le serveur est recherché (**www.serveur.com**) selon un certain protocole (**http://**) puis la page demandée, dans le chemin, est recherchée sur le serveur (**index.html**).

Si la page est trouvée, elle est envoyée au navigateur du client qui l'interprète et l'affiche, autrement une page "404" est affichée.

- ≡ Les extensions des pages HTML sont HTM ou HTML.
- ≡ L'extension des pages PHP est PHP (On peut trouver .inc, .inc.php, ...).
- ≡ Exemple de requête : <http://127.0.0.1/index.html>

1.2.1 Requête HTTP

Une requête HTTP est **un ensemble de lignes envoyé du navigateur au serveur**.

Elle comprend :

- ≡ Une ligne de requête (protocole, url, éventuellement des données),
- ≡ Les champs d'en-tête de la requête,
- ≡ Le corps de la requête.

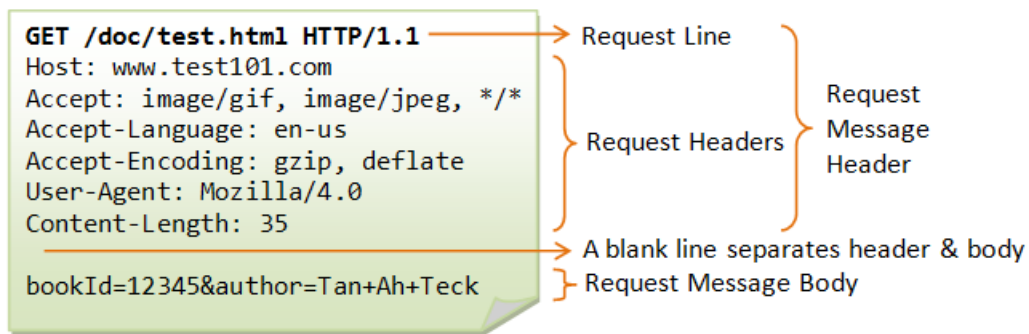


Fig. 1.4 : Requête HTTP

1.2.2 Réponse HTTP

Une requête HTTP est **un ensemble de lignes envoyé du serveur au navigateur**.

Elle comprend :

- ≡ Une ligne de statut,
- ≡ Les champs d'en-tête de la réponse,
- ≡ Le corps de la réponse.

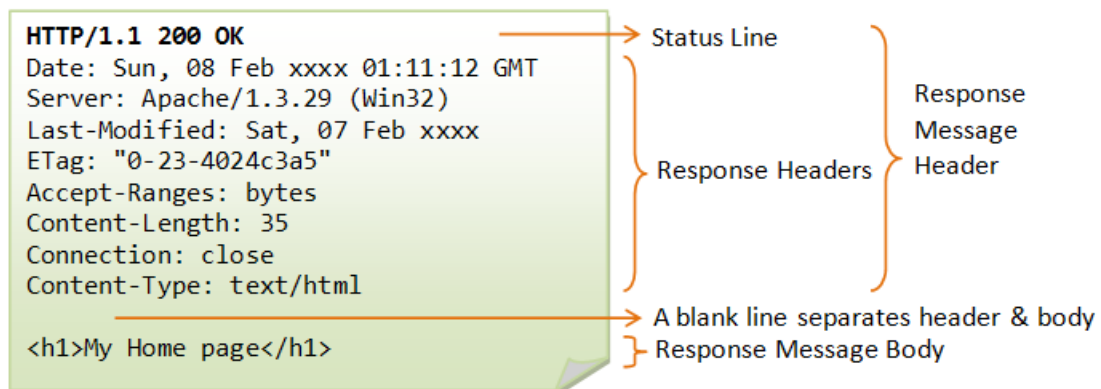


Fig. 1.5 : Réponse HTTP

1.3 Le fonctionnement de WEB dynamique avec PHP

Avec PHP le fonctionnement est quasiment identique.

Une étape supplémentaire est nécessaire : la génération dynamique de code HTML via l'interpréteur PHP qui est un module hébergé par un serveur HTTPd type Apache ou IIS ou autre.

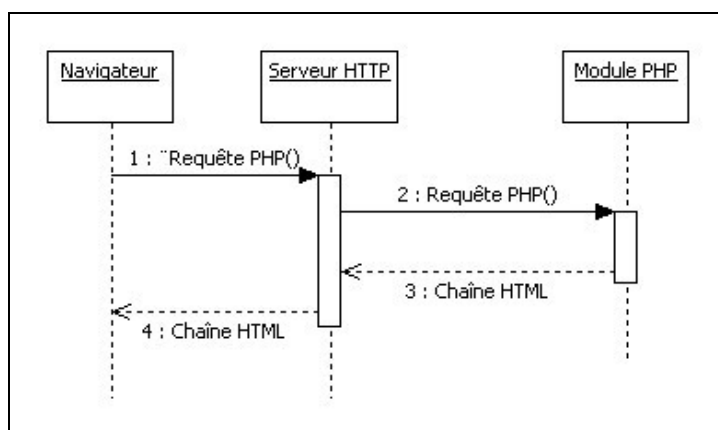


Fig. 1.6 : Architecture CS avec PHP

La saisie de l'URL sera du type **http://www.serveur/site2/page1.php**

La demande est envoyée sur l'Internet ou l'Intranet ou l'extranet, le serveur est repéré, il cherche la page, traite le code PHP, produit le code HTML et l'envoie au client.

Lorsque vous affichez la source via le navigateur d'une page PHP vous ne visualisez pas le code PHP mais **seulement le code HTML** généré par l'interpréteur PHP.

1.3.1 Avec un SGBDR

Une requête est ajoutée qui est traitée par un SGBDR (Système de Gestion de Bases de Données Relationnelles), e.g. Oracle, MySQL, MSS, PostgreSQL, ...

C'est donc une architecture à 3 niveaux.

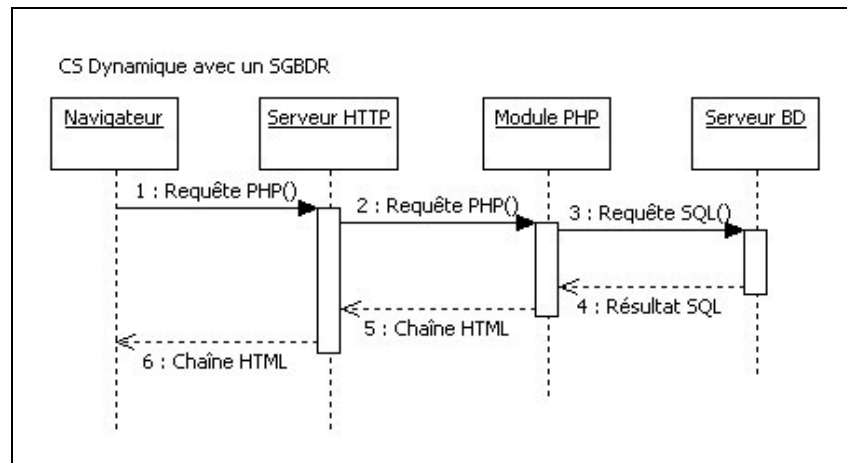


Fig. 1.6 : Architecture CS avec PHP et SGBDR

1.4 Environnement

Besoin minimum : Serveur web + module PHP.

À recommander : Des environnements de développement intégré (Apache + MySQL + PHP): XAMPP, WAMP, MAMP, ...

1.4.1 XAMPP

Site officiel : <https://www.apachefriends.org/fr/index.html>

Caractéristiques :

- ≡ cross platform,
- ≡ MariaDB, une bonne compatibilité,
- ≡ Perl (langage de programmation) peut être installé en option.

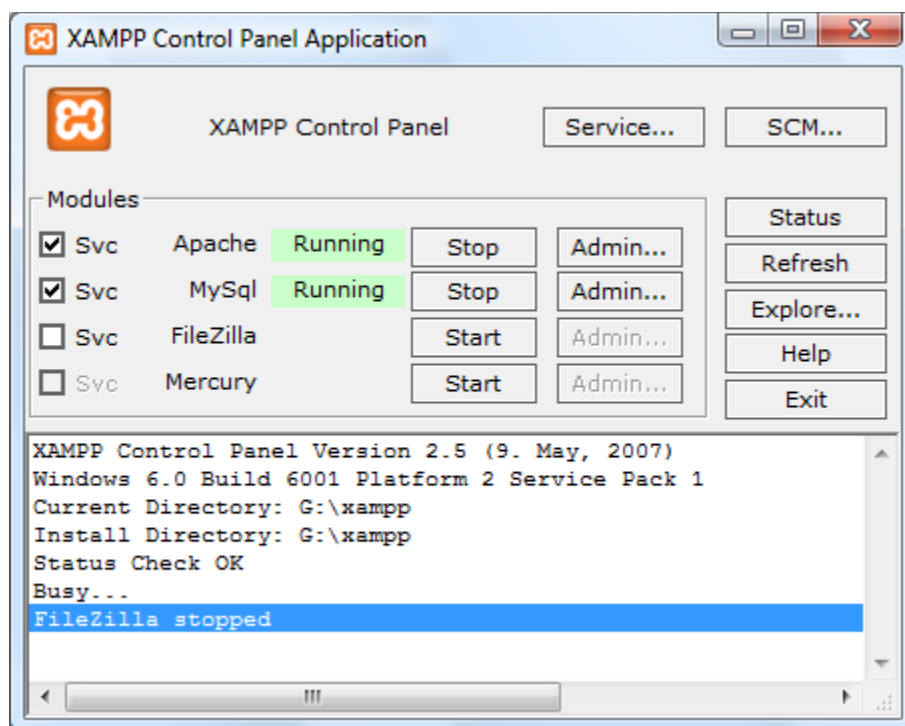


Fig. 1.7 : Panneau d'administration XAMPP

1.4.2 WAMP

Site officiel : <http://www.wampserver.com/>

Caractéristiques :

- ≡ conçu pour fonctionner sur Windows.

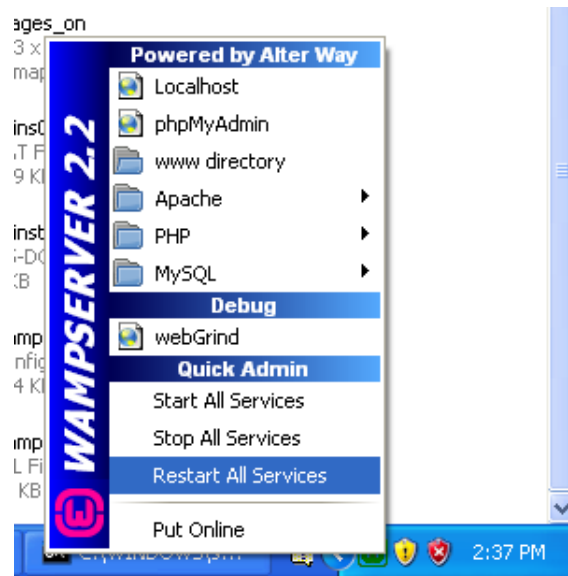


Fig. 1.8 : Menu WAMP

1.4.3 MAMP

Site officiel : <https://www.mamp.info/>

Caractéristiques :

- ≡ Fonctionne sur MAC OSX, mais aussi sur Windows,
- ≡ La version PRO est payant.



Fig. 1.9 : Panneau d'administration MAMP

1.5 Éditeurs

NetBeans (Gratuit, Sun puis Oracle), Eclipse (Gratuit, Eclipse), SublimeText (Gratuit), Notepad++ (Gratuit), DreamWeaver (Payant, Adobe), PhpStorm (Payant), etc.

1.5.1 NetBeans

Site officiel : <https://netbeans.org>

- ≡ Téléchargez l'IDE pour PHP, puis installez-le.
- ≡ Créez avant la création du premier workspace un dossier php dans votre arborescence de site web (par exemple C:\xampp\htdocs\php).
- ≡ Lancer NetBeans, choisissez le workspace créé.
- ≡ Créez un nouveau projet PHP.

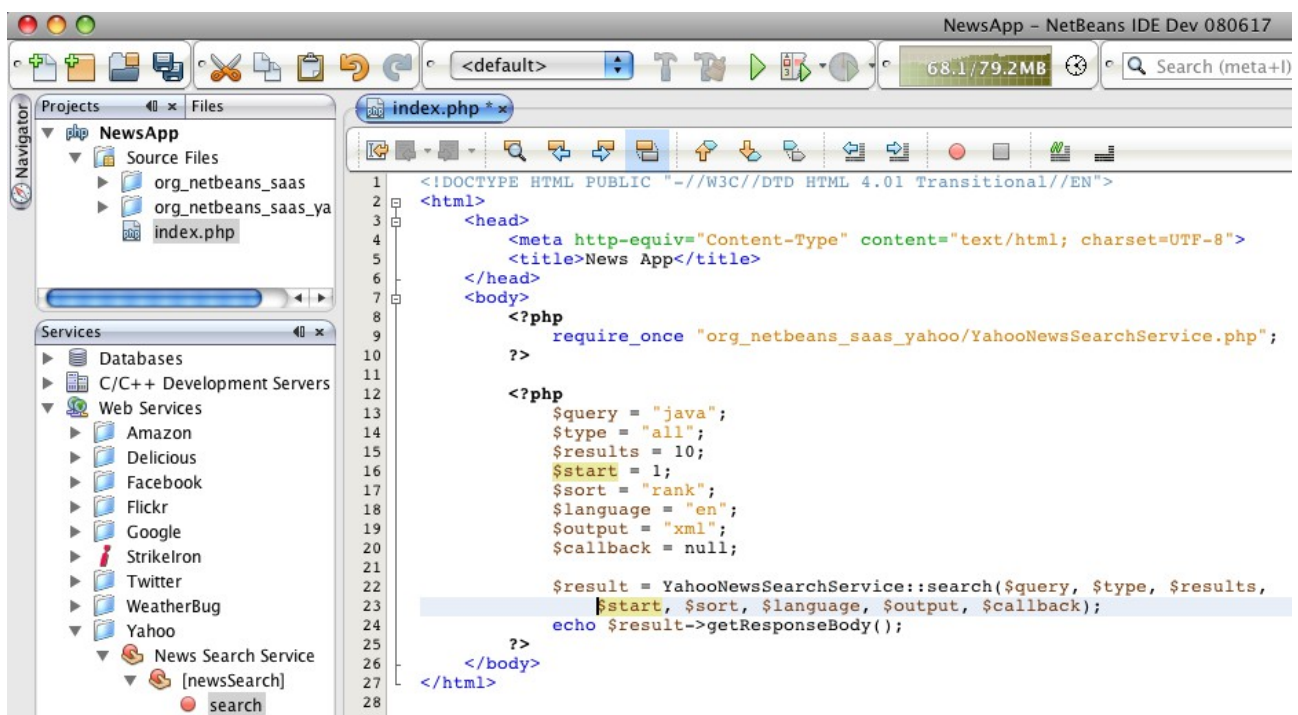


Fig. 1.10 : Netbeans pour PHP

1.5.2 Eclipse for PHP Developers

Site officiel : <http://www.eclipse.org/downloads/>

Guide d'installation : <http://wiki.eclipse.org/Eclipse/Installation>

- ≡ Téléchargez l'**Eclipse for PHP Developers**, puis installez-le.

PHP – Introduction

- ≡ Créez avant la création du premier workspace un dossier php dans votre arborescence de site web (par exemple C:\xampp\htdocs\php).
- ≡ Lancer Eclipse, choisissez le workspace créé.
- ≡ Créez un nouveau projet PHP.

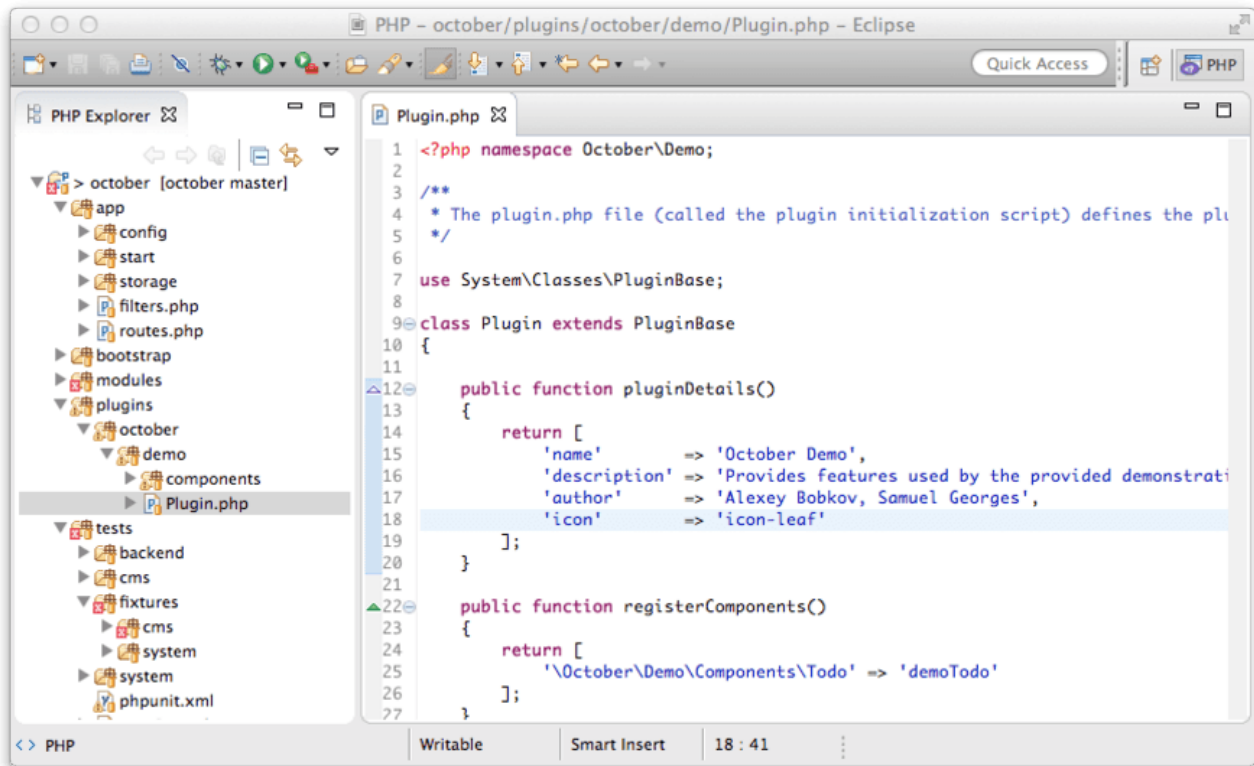


Fig. 1.11 : Eclipse pour développeurs PHP

1.5.3 Sublime Text

Site officiel : <http://www.sublimetext.com/>

Téléchargez puis installez-le.

Avantage : léger, thème personnalisé

PHP – Introduction

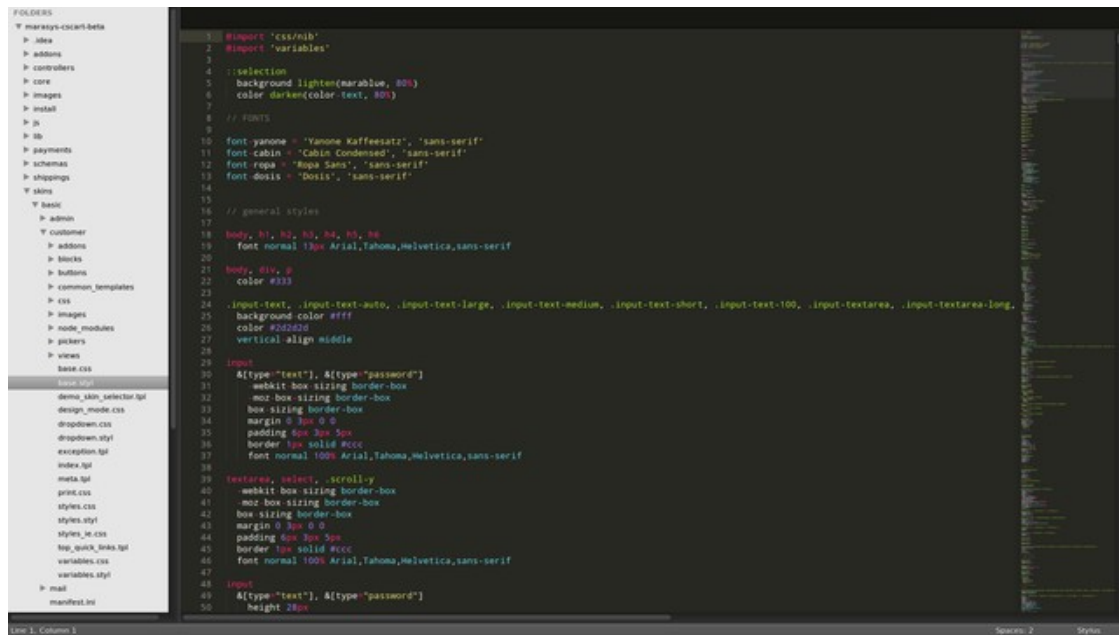


Fig. 1.12 : Sublime text

1.5.4 Notepad ++

Site officiel : <https://notepad-plus-plus.org/>

Téléchargez puis installez-le.

Avantage : léger

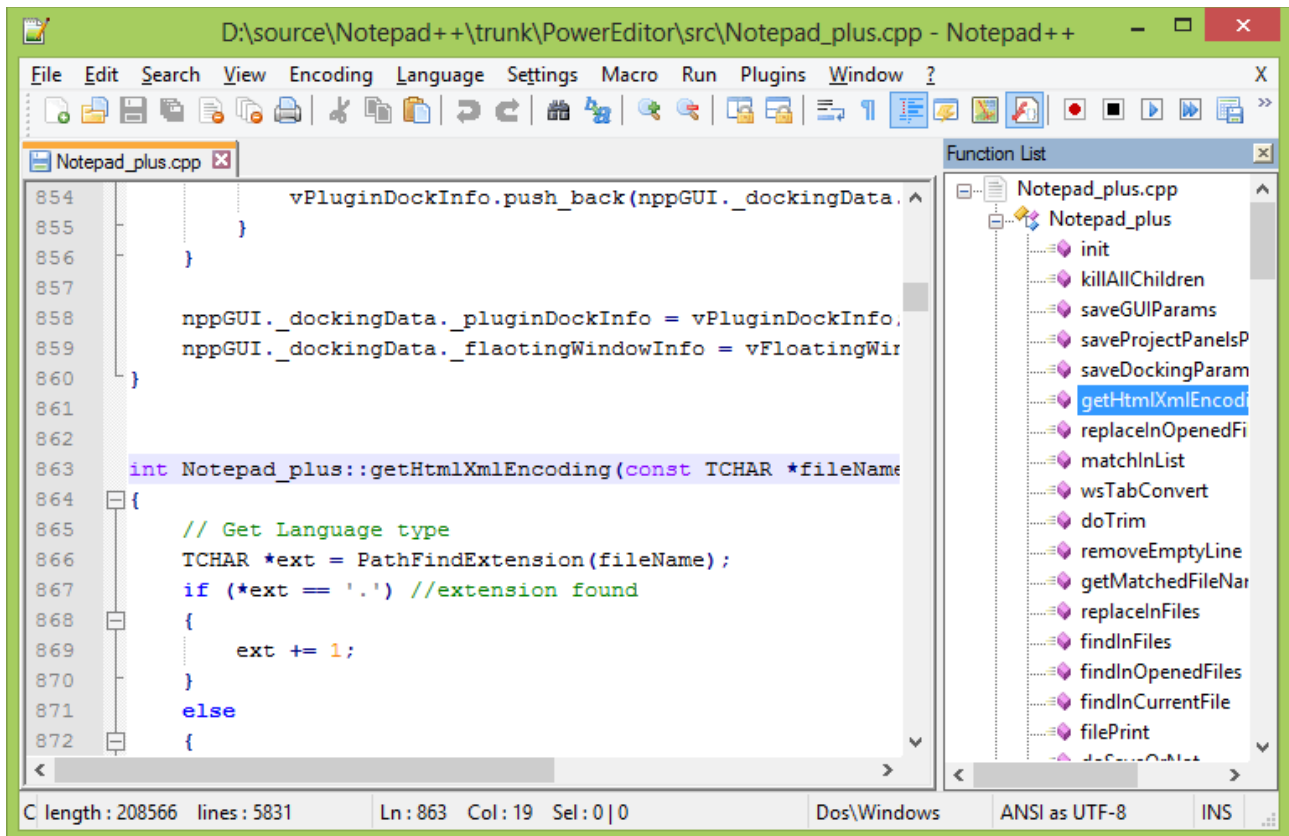


Fig. 1.13 : Nopepad++

1.5.5 DreamWeaver

Site officiel : <http://www.adobe.com/products/dreamweaver.html>

Avantage : convient pour le design visuel.

PHP – Introduction

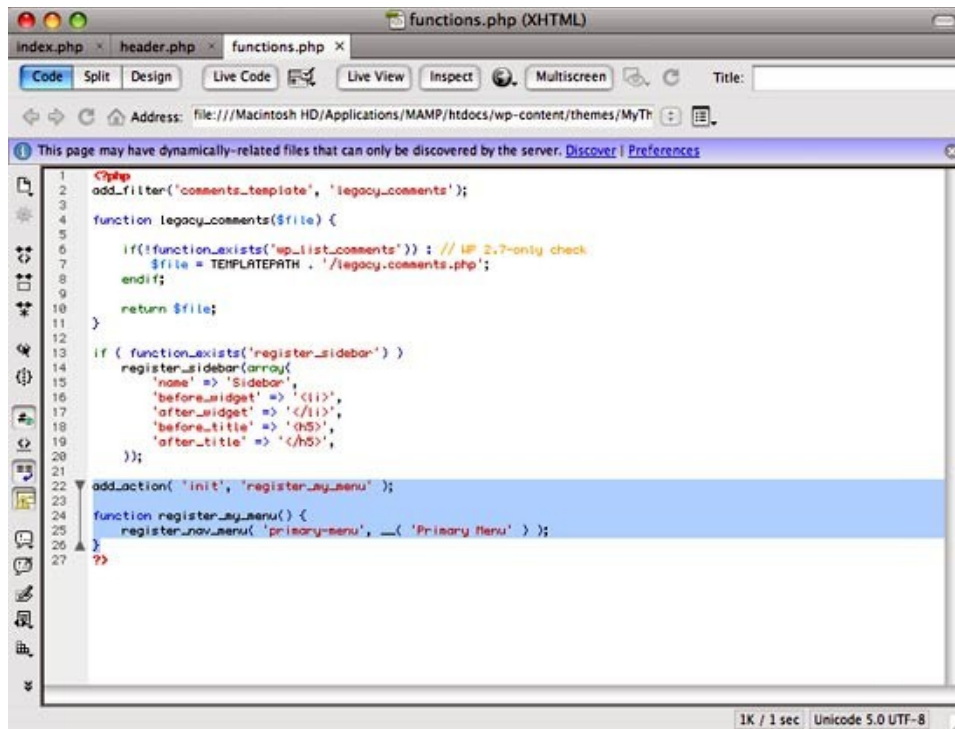


Fig. 1.14 : DreamWeaver

1.6 Une page PHP avec du HTML ou l'inverse

Résultat

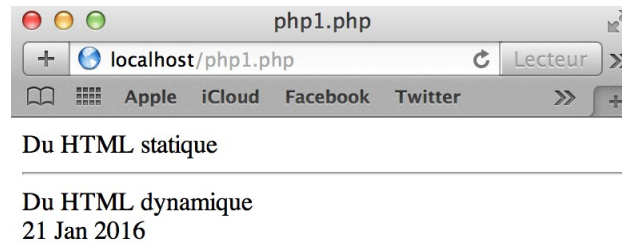


Fig. 1.15 : PHP dans HTML

Le code de php1.php :

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>php1.php</title>
  </head>

  <body>
    Du HTML statique
    <hr />
    <?php
      echo "Du HTML dynamique<br/>", date('d M Y');
    ?>
  </body>
</html>
```

Si vous affichez la source dans le navigateur, vous obtenez ça :

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>php1.php</title>
  </head>

  <body>
    Du HTML statique
    <hr>
    Du HTML dynamique<br>21 Jan 2016
  </body>
</html>
```

L'analyse du code :

- ≡ **Les balises <?php ...?> sont les balises standards**, tous les codes PHP doit être écrit entre ces deux balises.
- ≡ Le code PHP est intégré dans une page HTML.
- ≡ Le code PHP génère du code HTML, le seul qui est envoyé au navigateur et interprétable par celui-ci (nous verrons par la suite que l'on intègre aussi des autres codes, interprétables par le navigateur, du javascript ou du css par exemple).
- ≡ La fonction **echo** ou **print** sont utilisées pour générer du plain texte.
 - La syntaxe de **echo** ou **print** est la suivante :

```
echo ("my text")          OU      echo('my text')  
print ("my text") OU      print('my text')
```

Bien distinguez les guillemets droits **" "**, et les guillemets anglais **" "**.
Variable entre **guillemets simples** sera affichée comme plain texte.

- **echo** est plus rapide que **print**.

1.7 Une page 100% PHP

Résultat :



Fig. 1.16 : Page 100% PHP

Le code :

```
<!DOCTYPE html>
<?php
    echo("<html>");
    echo("<head>");
    echo("<meta http-equiv='Content-Type' content=\"text/html; charset=utf-
8\" />");
    echo("<title>Deuxième PHP</title>");
    echo("</head>");

    echo("<body>");
    echo("<center>");
    echo("<font face=\"Comic sans MS\" color=\"darkblue\">");
    echo("Date en\bleu ");
    echo(date("d M Y"));
    echo("</font>");
    echo("<hr />");
    echo("<font face=\"courier new\" color=\"red\">");
    echo("Heure en rouge \n");
    echo(date("h:i:s"));
    echo("</font>");
    echo("</body>");
    echo("</html>");
?>
```

L'analyse du code :

≡ Cette fois-ci tout le script est en PHP.

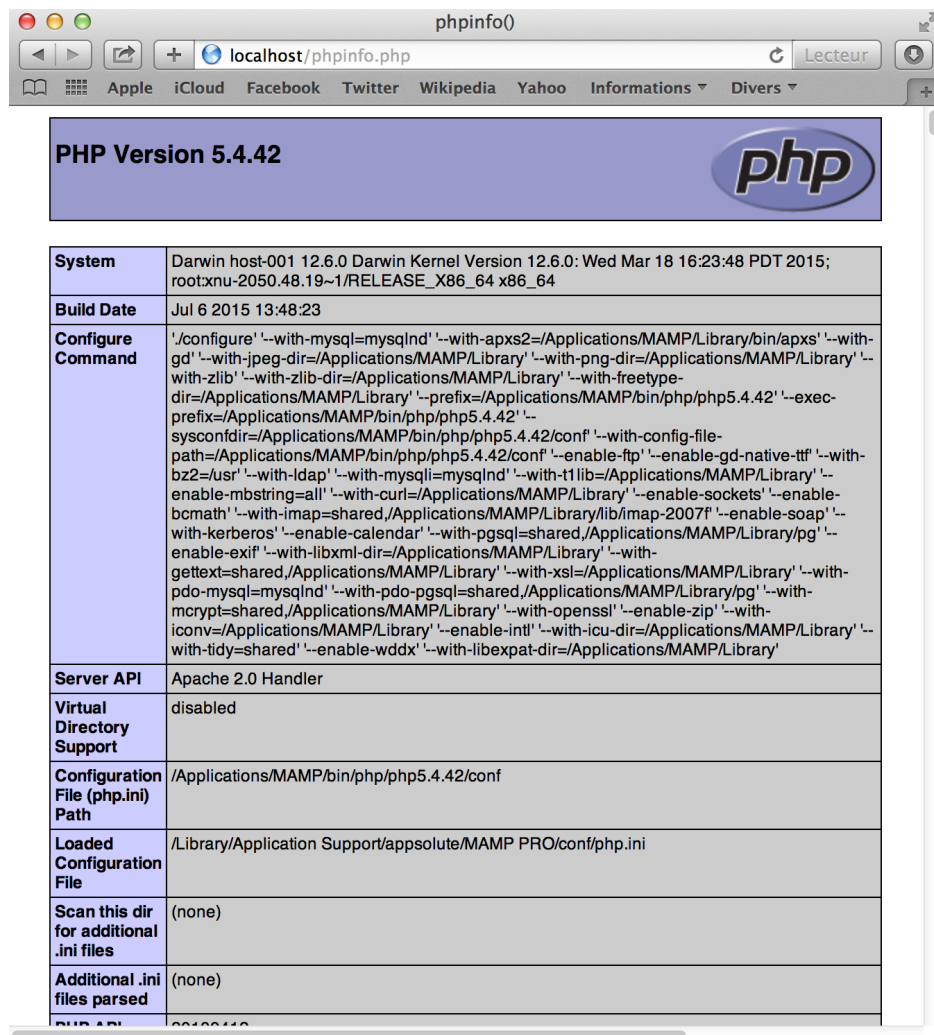
PHP – Introduction

- ≡ Bien entendu un seul appel à la fonction **echo** serait possible et souhaitable.
- ≡ Notez le caractère d'échappement **** est utilisée entre les guillemets.
L'échappement fonctionne qu'entre les **double guillemets**.
- ≡ Notez les **\n** (ou **\r**) et **\t** pour générer des retours chariot et des tabulations.
- ≡ Inspectez le code source dans le navigateur.

1.8 Phpinfo

Informations sur PHP : créez une page, nommez-la `phpinfo.php` par exemple. Mettez du code dessous :

```
<?php
    phpinfo();
?>
```



System	Darwin host-001 12.6.0 Darwin Kernel Version 12.6.0: Wed Mar 18 16:23:48 PDT 2015; root:xnu-2050.48.19~1/RELEASE_ARM_T8040
Build Date	Jul 6 2015 13:48:23
Configure Command	'/configure' '--with-mysql=mysqlnd' '--with-apxs2=/Applications/MAMP/Library/bin/apxs' '--with-gd' '--with-jpeg-dir=/Applications/MAMP/Library' '--with-png-dir=/Applications/MAMP/Library' '--with-zlib' '--with-zlib-dir=/Applications/MAMP/Library' '--with-freetype-dir=/Applications/MAMP/Library' '--prefix=/Applications/MAMP/bin/php/php5.4.42' '--exec-prefix=/Applications/MAMP/bin/php/php5.4.42' '--sysconfdir=/Applications/MAMP/bin/php/php5.4.42/conf' '--with-config-file-path=/Applications/MAMP/bin/php/php5.4.42/conf' '--enable-ftp' '--enable-gd-native-ttf' '--with-bz2=usr' '--with-ldap' '--with-mysqli=mysqlnd' '--with-t1lib=/Applications/MAMP/Library' '--enable-mbstring=all' '--with-curl=/Applications/MAMP/Library' '--enable-sockets' '--enable-bcmath' '--with-imap=shared,/Applications/MAMP/Library/lib/imap-2007f' '--enable-soap' '--with-kerberos' '--enable-calendar' '--with-pgsql=shared,/Applications/MAMP/Library/pg' '--enable-exif' '--with-libxml-dir=/Applications/MAMP/Library' '--with-gettext=shared,/Applications/MAMP/Library' '--with-xsl=/Applications/MAMP/Library' '--with-pdo-mysql=mysqlnd' '--with-pdo-pgsql=shared,/Applications/MAMP/Library/pg' '--with-mcrypt=shared,/Applications/MAMP/Library' '--with-openssl' '--enable-zip' '--with-iconv=/Applications/MAMP/Library' '--enable-intl' '--with-icu-dir=/Applications/MAMP/Library' '--with-tidy=shared' '--enable-wddx' '--with-libexpat-dir=/Applications/MAMP/Library'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/Applications/MAMP/bin/php/php5.4.42/conf
Loaded Configuration File	/Library/Application Support/appsolute/MAMP PRO/conf/php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)

Fig. 1.17 : *phpinfo*

2 Constantes, variables, opérateurs

2.1 Convention de langage

- ≡ Les nom des variable sont sensible à la casse et précédés par un **\$**.
- ≡ Les **lettres** (non accentuées), **_** et **chiffres** sont autorisés pour composer les noms des variables.
- ≡ Par convention, les noms des variables sont en minuscules avec des majuscules en début de nom composés ou un underscore comme séparateur de mots (**CamelCase**).
- ≡ Par convention, les nom des constantes sont en majuscules.
- ≡ La casse sensible s'applique à :
 - variables,
 - constantes,
 - clés tableaux (array keys),
 - propriétés de classe (class properties),
 - constantes de classe (class constants)
- ≡ La casse **insensible** s'applique à :
 - fonctions,
 - constructeurs de classe (class constructors),
 - méthodes de classe (class methods)
 - keywords and constructs (if, else, null, foreach, echo, etc..)

2.2 Les commentaires

2.2.1 Une seule ligne

// En début de ligne ou en fin de ligne. Tout ce qui suit est ignoré.

2.2.2 Plusieurs lignes

*/**

Bloc de lignes en commentaires

**/*

2.3 Les constantes

Une constante est un identificateur qui ne change pas de valeur. Le nom de la constante sont en **majuscules**. Son identificateur **n'est pas précédé par un \$**.

On définit une contante avec l'opérateur **define**.

Pour tester son existence : **defined**("NOM_DE_CONSTANTE")

Syntaxe :

```
define("NOM_DE_CONSTANTE", valeur)
```

Exemples :

```
define("SALUTATION", "Bonjour");  
define("PI", 3.14);
```

```
echo "<br />", SALUTATION;
```

```
if(defined("PI")) echo PI;
```

NOTE : Pour créer des contantes dans une classe, utilisez syntaxe :

```
const MY_CONST = val
```

Exemple :

```
class MyClass  
{  
    const CONSTANT = 'constant value';  
  
    function showConstant() {  
        echo self::CONSTANT . "\n";  
    }  
}
```


2.4 Les variables

Une variable est une espace de stockage. Il est possible stocker des texte, chiffres ou les autres types dans ces espaces de stockage, pour que nous pouvons les manipuler et utiliser.

Les identificateurs (noms) des variables sont précédés par un **\$**.

Exemples :

```
$unMot = "Bonjour";  
$uneChiffre = "123";
```

2.4.1 Les types de PHP variable

entiers (int)

réel (double)

chaîne (string)

booléen (boolean)

tableau (array)

objet (object)

Pour les booléens, TRUE (Vrai) renvoie 1 et FALSE (Faux) renvoie 0 depuis la version 5.3 (avant renvoyait vide).

2.5 Types de variables et portée

1. Les variables “superglobales” : **\$_SERVER**, **\$_GET**, **\$_POST**, **\$_REQUEST**, **\$_COOKIE**, ... (informations sur le serveur, contenus de l'url, ...) sont disponibles dans chaque script. Tout en majuscules.

Ce sont des tableaux associatifs ou tableaux à clés.

2. Les constantes sont globales (disponible dans tout le document).
3. Les variables déclarées dans un script (hors d'une fonction) sont locales au script, i.e. Disponibles dans tout le script mais inaccessibles dans les fonctions.
4. Les variables déclarées dans une fonction sont locales à la fonction, i.e. Seulement accessibles dans la fonction.
5. Les variables déclarées comme **globales** (qualificateur **global**) dans les fonctions font référence à une variable déclarée comme globale au niveau du script : elles sont globales au script donc accessibles dans le script et dans la fonction. **Très mauvaise pratique.**

Le passage de valeurs ou de références via des passages de paramètres est préférable.

6. Les variables déclarées comme **statiques** (qualificateur **static**) dans une fonction sont statiques (locales globales), i.e. Elles ne sont pas “effacées” de la mémoire à la fin de l'exécution de la fonction. Elles sont principalement utilisées dans le cadre de la récursivité.

Tableau récapitulatif

Niveau / Qualificateur	Aucun qualificateur	Qualificateur global	Qualificateur static
Script	Locale (e.g. partout dans le script sauf dans les fonctions)	Globale	
Fonction	Locale (e.g. Dans la fonction)	Globale (à condition qu'elle déclarée global dans le script)	Locale-globale

2.5.1 Portée constantes



PORTEES (Constantes)

Affichage dans le script de la constante: 3.14

Affichage dans la fonction de la constante, donc globale: 3.14

Fig. 2.1 : Portées – constantes

```
<?php

// En-tête
echo("<title>PORTEES</title>\n<strong>PORTEES (Constantes)</strong><hr
/>");

/**
 * My function
 */
function myFunc(){
    echo("Affichage dans la fonction de la constante, donc globale:
".PI);
}

define("PI", 3.14);                // Déclaration de la constante globale

// =====
echo("Affichage dans le script de la constante: ".PI."<br />");
// Afficher la constante
myFunc();
// L'appel de la fonction

?>
```

Conclusion :

Quelque soit la position (script ou fonction), la constante est disponible et donc son contenu peut être affiché.

2.5.2 Portée locales

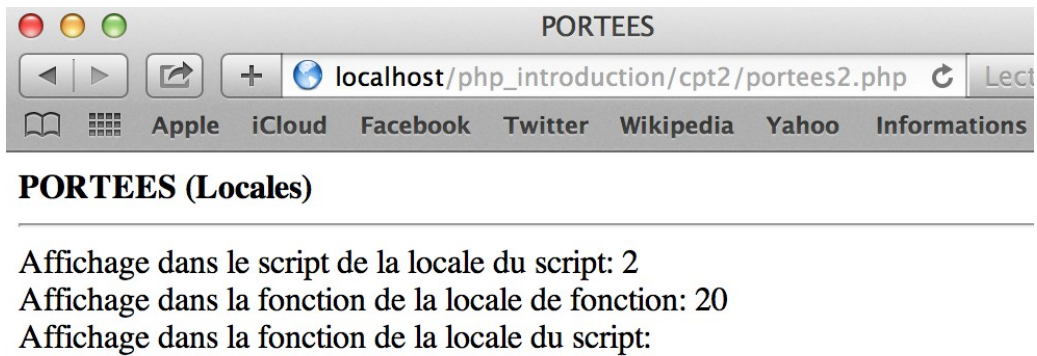


Fig. 2.2 : Portées – locales

```
<?php

// En-tête
echo("<title>PORTEES</title>\n<strong>PORTEES (Locales)</strong><hr />");

/**
 * My function
 **/
function myFunc(){
    $varInFunc = 20;
    echo("Affichage dans la fonction de la locale de fonction: ".
$varInFunc."<br />");
    echo("Affichage dans la fonction de la locale du script: ".
$varInScript."<br />");
}

$varInScript = 2;           // Déclaration de la variable du script

// =====
echo("Affichage dans le script de la locale du script: ".
$varInScript."<br />");      // Afficher la locale du script
myFunc();
// L'appel de la fonction

?>
```

conclusion :

En fonction du “lieu” (script ou fonction)

Un script accède

Une fonction accède

2.5.3 Portée globales



Fig. 2.3 : Portées – globale

```
<?php

// En-tête
echo("<title>PORTEES</title>\n<strong>PORTEES (Globales)</strong><hr
/>");

/**
 * My function
 **/
function myFunc() {
    global $varGlobal;           // Accéder la variable globale
    echo("Affichage dans la fonction de la variable globale : ".
$varGlobal."<br />");
}

$varGlobal = 20;                // Déclaration de la globale

// =====
echo("Affichage dans le script de la globale : ". $varGlobal."<br />");
    // Afficher la globale du script
myFunc();                      // L'appel de la fonction
?>
```

2.5.4 Portée statiques



Fig. 2.4 : Portées – statiques

```
<?php

// En-tête
echo("<title>PORTEES</title>\n<strong>PORTEES (Statiques)</strong><hr
/>");

/**
 * My function
 */
function myFunc(){
    static $varStatic = 0;
    echo("Static dans la fonction: ".$varStatic."<br />");
    $varStatic ++;
}

// =====
myFunc();           // L'appel de la fonction
myFunc();           // L'appel de la fonction
myFunc();           // L'appel de la fonction
myFunc();           // L'appel de la fonction

?>
```

2.6 Les opérateurs de conversion

Ces opérateurs opèrent un CAST, i.e. un trans-typage.

Fonction	Fonctionnalité
(int)	Convertit en entier
(string)	Convertit en chaîne
(double)	Convertit en réel
(boolean)	Convertit en booléen
(array)	Convertit en tableau
(object)	Convertit en objet

Exemples :

```
$vsNumber = "123";           // type string
$viNumber = (int)$vsNumber;   // type int
```

c.f. Aussi les fonction `intval()`, `floatval()` et `stringval()` ainsi que `setType()` et `getType()`.

2.7 Opérateurs divers

2.7.1 Suppression d'erreur : @

L'exécution continue en ignorant des erreurs, l'erreur ne sera pas visible.

e.g. : `@fopen(...)` ...

2.7.2 Opérateur d'exécution de commandes système : ` (back quote)

Permet l'exécution d'une commande système (si c'est autorisé).

```
$command = `ls`;  
echo $command;
```


2.8 Fonction d'exécution de commandes système

Pour exécuter une commande système il est possible d'utiliser les fonction suivantes : **system**, **passthru** ou **exec**.

2.8.1 system

Exécute un command système et afficher le résultat (en texte).

Syntaxe :

```
system("commande", $varOutput);
```

2.8.2 passthru

Exécute un command système et afficher le résultat (en texte, peut être binaire).

Syntaxe :

```
passthru("commande", $varOutput);
```

2.8.3 exec

Exécute un command système et renvoie le résultat dans un tableau.

Syntaxe :

```
exec("command", $arrayOutput, $intIndicator);
```

Exemple code :

```
<?php

    // --- Ce script liste les fichiers du dossier courant.
    // --- $indicator renvoie 0 si OK et 1 si KO
    // --- $lsDerniereLigne correspond à la dernier élément de $tFichiers.
$lsDerniereLigne = exec("ls -li", $tFichiers, $indicator);

if($indicator != 0) die("Commande incorrecte");
    $liCountFichiers = count($tFichiers);           // Nombre de élément
dans le tableau

    for($i=0; $i<$liCountFichiers; $i++)
    {
        $lsLigne    = $tFichiers[$i];             // Une ligne
d'élément

        $tElements = explode(" ", $lsLigne);       // Exploder la chaîne
de caractère en tableau
        $liCountElements = count($tElements);       // Nombre de élément
dans le tableau
        $cleFin = $liCountElements - 1;             // Clé de la
dernier élément dans le tableau
        if(is_file($tElements[$cleFin])){
            echo $tElements[$cleFin], "<br />";
            // --- Ne fonctionne pas au niveau de c:\\ ie hors de l'arbo du
serveur httpd -- windows
        }
    }
?>
```

2.9 Les fonctions sur les variables

Fonction	Description
<code>empty(\$var)</code>	Renvoie vrai si la variable est vide, i.e. Si la variable n'a pas été initialisée ou si la variable contient "" ou 0 ou NULL ...
<code>isset(\$var)</code>	Renvoie True si la variable existe, False dans le cas contraire.
<code>unset(\$var)</code>	Détruit une variable
<code>gettype(\$var)</code>	Retourne le type de la variable sous forme de chaîne
<code>settype(\$var, "newType")</code>	Convertit la variable en type newType (cast)
<code>is_callable("function")</code>	Renvoie vrai si la fonction est callable
<code>is_scalar(\$var)</code>	Renvoie vrai si la variable est scalaire
<code>is_string(),</code> <code>is_bool(),</code> <code>is_numeric(),</code> <code>is_nan(),</code> <code>is_long(),</code> <code>is_double(),</code> <code>is_float(),</code> <code>is_integer(),</code> <code>is_int(),</code> <code>is_array(),</code> <code>is_object(),</code> <code>is_resource()</code>	Chaque fonction renvoie vrai si la variable est du type considéré.

Exemple pour `empty()` :

```
<?php
    $v;

    echo "$v*  


```

PHP – Introduction

```
if(empty($v)) echo "Vide<br />"; else echo "Plein<br />";

$v = "Mot";
echo "$v*<br />"; // --- Renverra Plein
if(empty($v)) echo "Vide<br />"; else echo "Plein<br />";
?>
```

2.10 Les opérateurs

2.10.1 Les opérateurs arithmétiques

Permet d'effectuer des opérations arithmétiques.

Opération	Opérateur
Addition	+
Soustraction	-
Multiplication à	*
Division fractionnaire	/
Modulo	%

Exemples :

```
<?php
    $n = 10;
    echo("Division fractionnaire : " . $n / 3); // --- renvoie 3,3333 (Cf
round($n,m) ou ceil($n) ou floor($n) ou printf plus loin.
    echo("Reste de la division entière : " . $n % 3); // --- Renvoie 1
?>
```

2.10.2 Les opérateurs d'assignation

Permet de modifier la valeur arithmétique d'une variable.

Opération	Opérateur
Affectation	=
Incrémentation	++
Décrémentation	--
Augmentation	+=
Diminution	-=

PHP – Introduction

Multiplication	<code>*=</code>
Division fractionnaire	<code>/=</code>
Modulo	<code>%=</code>

Exemples :

```
<?php
    $v1 = 5;
    $v1++;
    echo("Incrémentation : " . $v1 . "<br />"); // --- affichera 6

    $v1 = 5;
    $v1 += 2;
    echo("Augmentation : " . $v1 . "<br />"); // --- affichera 7
?>
```

2.10.3 Les opérateurs de comparaison

Permet de comparer des valeurs.

Opération	Opérateur
Egalité	<code>==</code>
Identité (valeur et type)	<code>===</code>
Inférieur à	<code><</code>
Supérieur à	<code>></code>
Inférieur ou égal	<code><=</code>
Supérieur ou égal	<code>>=</code>
Différent de	<code>!=</code>

Exemple sur l'égalité et l'identité :

```
<?php
```

PHP – Introduction

```
$test = 3==3;
print("3==3 --> *$test*<br />"); //renvoie vrai donc 1.

$test = 3=='3';
print("3=='3' --> *$test*<br />"); //renvoi vrai donc 1.

$test = 3==='3';
print("3==='3' --> *$test*<br />"); //renvoi faux donc "".

?>
```

2.10.4 Les opérateurs logiques

Permet de tester des conditions complexes.

Opération	Opérateur	Description
ET logique	AND, &&	Les 2 doivent être vrais
OU inclusif	OR,	Au moins un des deux doit être vrai
NON logique	!	L'inverse
OU exclusif	XOR	Seul un des deux doit être vrai

La XOR permet, par exemple, de mettre en place des contraintes d'exclusion (La lumière est soit allumée soit éteint).

Table de vérité de conditions complexes.

C1	C2	Opérateur	C
O	O	AND	O
O	X	AND	X
X	O	AND	X
X	X	AND	X
O	O	OR	O

O	X	OR	O
X	O	OR	O
X	X	OR	X
O		NOT	X
X		NOT	O
O	O	XOR	X
O	X	XOR	O
X	O	XOR	O
X	X	XOR	X

Exemple AND :

```
<?php
    $ville = "Paris";
    $age   = 16;

    if($ville == "Paris" and $age >= 18) echo("Majeur parisien");
    else echo("Ou mineur ou non parisien");
?>
```

2.10.5 Les opérateurs binaires

Permet d'effectuer des opérations binaires (Bit à bit).

Opérateur	opération
&	ET
	OU inclusif
^	OU exclusif
~	NON
<<	Décalage à gauche : Multiplie n fois par 2

>>	Décalage à droite : Divise n fois par 2
----	---

Exemple :

```
<?php
    $v1 = 2;
    $v2 = $v1 << 2; // $v2 = 8 -> 2 * 22
    $v2 = $v1 >> 1; // $v2 = 1 -> 2 / 21
    echo $v2;
?>
```

2.11 Les dates

Principales fonctions sur les dates :

Fonction	Type renvoie	Description
time()	Int	Retourne le nombre de secondes écoulées depuis le 1 janvier 1970, 00:00:00
date("format")	String	Renvoie la date et l'heure du jour sous forme de chaîne de caractères formatée. e.g. date("D, d M Y, H:i:s ")
date("format", timestamp)	String	Renvoie la date passée en deuxième paramètre selon le format précisé comme premier paramètre. Par exemple date("D d M Y", time() - 360000) → 100H avant.
setlocale(LC_TIME, "fr_FR") strftime("%A %d %B %Y", time())	String	Renvoie la date du jour en français (c.f. Doc manuel) Vérifier des codes langue-pays disponibles dans la console : locale -a
strtotime("une date")	Int	Renvoie le nombre de secondes écoulées entre le 1/1/1970 et la date spécifiée en anglais. "une date" peut prendre les valeurs : "now", "1 january 2000", ...
mktime(h,m,s,m,d,y)	Int	Revoie le nombre de secondes écoulées entre le 1/1/1970 et la date spécifiée.
getdate([timestamp])	Array	Revoie un tableau avec la date et l'heure du jour
checkdate(mois, jour, année)	Booléen	Renvoie true si la date est valide.
gmdate("format")	String	Renvoie la date et l'heure GMT du jour sous forme de chaîne de caractères formatée.

Note : PHP 5.2.2 ajoute les millisecondes (u).

Quelques éléments de formatage : **d m Y H:i:s** → jour mois année heure minute seconde.

Exemples :

≡ **time()** : revoie le TimeStamp, i.e. Le nombre de secondes écoulées entre la date où la fonction est exécutée et le 1^{er} janvier 1970 à 00:00:00 GMT.

```
echo("La fonction time : " . time() . "<br />");
```

Affiche : La fonction time : 1453590000

≡ **date("format")** : renvoie la date du jour formatée.

```
echo("La fonction date('d-m-Y') : " . date('d-m-Y') . "<br />");
```

Affiche : La fonction date('d-m-Y') : 24-01-2016

```
echo("Heure date('H:i:s') : " . date('H:i:s') . "<br />");
```

Affiche : Heure date('H:i:s') : 00:00:00

≡ **date("format", TimeStamp)** : renvoie une date formatée.

```
$ldHier = time() - (60 * 60 * 24);  
$ldDemain = time() + (60 * 60 * 24);  
echo "La fonction date('Y-m-d' , ldHier) : " . date('d-m-Y', $ldHier);  
echo "La fonction date('Y-m-d' , ldDemain) : " . date('d-m-Y', $ldDemain);
```

Affiche : La fonction date ('Y-m-d' , ldHier) : 23-01-2016

La fonction date('Y-m-d' , ldDemain) : 25-01-2016

≡ **strftime("format", TimeStamp)** et **setlocale(Constante_paramètres, "langue-location")**

strftime formate la date et l'heure en fonction de la localisation précisée avec **setlocale()**.

Syntaxe :

```
strftime(format,timestamp);
```

Parameter	Description
<i>format</i>	<p>Required. Specifies how to return the result:</p> <ul style="list-style-type: none"> ≡ %a - abbreviated weekday name ≡ %A - full weekday name ≡ %b - abbreviated month name ≡ %B - full month name ≡ %c - preferred date and time representation ≡ %C - century number (the year divided by 100, range 00 to 99) ≡ %d - day of the month (01 to 31) ≡ %D - same as %m/%d/%y ≡ %e - day of the month (1 to 31) ≡ %g - like %G, but without the century ≡ %G - 4-digit year corresponding to the ISO week number (see %V). ≡ %h - same as %b ≡ %H - hour, using a 24-hour clock (00 to 23) ≡ %I - hour, using a 12-hour clock (01 to 12) ≡ %j - day of the year (001 to 366) ≡ %m - month (01 to 12) ≡ %M - minute ≡ %n - newline character ≡ %p - either am or pm according to the given time value ≡ %r - time in a.m. and p.m. notation ≡ %R - time in 24 hour notation ≡ %S - second ≡ %t - tab character ≡ %T - current time, equal to %H:%M:%S ≡ %u - weekday as a number (1 to 7), Monday=1. Warning: In Sun Solaris Sunday=1 ≡ %U - week number of the current year, starting with the first Sunday as the first day of the first week ≡ %V - The ISO 8601 week number of the current year (01 to 53), where week 1 is the first week that has at least 4 days in the current year, and with Monday as the first day of the week ≡ %W - week number of the current year, starting with the first Monday as the first day of the first week ≡ %w - day of the week as a decimal, Sunday=0 ≡ %x - preferred date representation without the time ≡ %X - preferred time representation without the date ≡ %y - year without a century (range 00 to 99) ≡ %Y - year including the century ≡ %Z or %z - time zone or name or abbreviation

	≡ %% - a literal % character
<i>timestamp</i>	Optional. Specifies a Unix timestamp that represents the date and/or time to be formatted. Default is the current local time (time())

setlocale modifie les paramètres locaux. **LC_TIME** pour la date et l'heure.

```
setlocale(LC_TIME, "fr_FR");  
echo("Date en français : " . strftime("%A %d %B %Y",time()));
```

Affiche : Date en Français : Dimanche 24 janvier 2016

≡ **mktime()**

renvoie le nombre de secondes écoulées depuis le 01/01/1970 et la date spécifiée au format (h,m,s,M,d,y).

```
echo("mktime(2,0,0,1,1,1970));
```

Affiche : 3600

mktime() permet de créer un date.

```
echo Date("d-m-Y", mktime(0,0,0,01,24,2016)); // Affiche 24-01-2016
```

≡ **strtotime()**

Renvoie le nombre de seconde écoulées depuis le 01/01/1970 et la date spécifiée aux formats variés, expressions textuelles d'anglais supporté.

Exemples :

```
echo strtotime("now") . "<br />";
echo strtotime("tomorrow") . "<br />";
echo strtotime("yesterday") . "<br />";
echo strtotime("01/24/2016") . "<br />";
echo strtotime("10 September 2000") . "<br />";
echo strtotime("+1 day") . "<br />";
echo strtotime("+1 week") . "<br />";
echo strtotime("+1 week 2 days 4 hours 2 seconds") . "<br />";
echo strtotime("next Thursday") . "<br />";
echo strtotime("last Monday") . "<br />";
echo strtotime("4pm + 2 Hours") . "<br />";
echo strtotime("now + 2 fortnights") . "<br />";
echo strtotime("last Monday") . "<br />";
echo strtotime("2pm yesterday") . "<br />";
echo strtotime("7am 12 days ago") . "<br />";
```

≡ **getdate([timestamp])**

Renvoie la date du jour ou une date (si le timestamp est renseigné) sous forme de tableau associatif.

```
var_dump(getdate());
```

Affiche :

```
array(11) {
  ["seconds"]=>
  int(0)
  ["minutes"]=>
  int(0)
  ["hours"]=>
```

PHP – Introduction

```
int(0)

["mday"]=>

int(24)

["wday"]=>

int(7)

["mon"]=>

int(1)

["year"]=>

int(2016)

["yday"]=>

int(24)

["weekday"]=>

string(9) "Sunday"

["month"]=>

string(7) "January"

[0]=>

int(1453590000)

}
```

Affiche une date avec getdate() :

```
$aujourdhui = getdate();
$jour = $aujourdhui['mday'];
$mois = $aujourdhui['month'];
$annee = $aujourdhui['year'];
print("$jour-$mois-$annee<br />");
```

Affiche : 24-January-2016

À tester :

PHP – Introduction

```
foreach(getdate(strToTime("now")) as $valeur) {  
    echo "<br/>$valeur";  
}
```


Exercice :

Affichez la date du jour en français avec un tableau associatif – getdate() (c.f. le **switch** et les **tableaux**).

La date sera du type “dimanche 24 janvier 2016”.

(Au moins deux façons différentes.)

Corrigé

2.12 La classe DateTime

La classe DateTime() (php >= 5.3) permet une gestion avancée et facilitée des dates.

Syntaxes :

Méthode	Description
\$d = new DateTime()	Crée un objet date – “Now” par défaut
\$d = new DateTime("2016-01-24 00:00:00")	Crée un objet date
\$d->add(new DateInterval("expression"))	(Expérimentale) ajoute un intervalle
\$d->sub(new DateInterval("expression"))	(Expérimentale) soustrait un intervalle
\$d->diff(\$d2)	Calcule la différence entre 2 dates

Script :

```
<?php
// --- datetime_1.php
header("Content-type: text/html;charset=UTF-8");
setlocale(LC_TIME, 'fr_FR.utf8','fra');
// --- Syntaxe objet
//$dateTime1 = new DateTime('2008-08-03 14:52:10');
$dateTime1 = new DateTime();
echo "Date 1 après création : ", $dateTime1->format('d F Y H:i:s'), "\n";
echo "<br/>";
// --- Nouvelle date
$dateTime2 = new DateTime();
echo "Date 2 après création : ", $dateTime2->format('d F Y H:i:s'), "\n";
echo "<br/>";
// --- Ajoute de date: P5D ou P5M5D ou P5Y5M5D ... (Le format doit etre Y
M D)
// --- Ajoute de time: PT5H ou PT5H ou PT5S ... (Le format doit etre H M
S)
// --- Renvoie un objet DateInterval : la difference entre 2 objets
DateTime
```

PHP – Introduction

```
$dateTime2->add(new DateInterval("P5D")); // + 5 jours
echo "<br/>Date 2 après ajout : ", $dateTime2->format('d F Y H:i:s'),
"\n";

$dateDiff = $dateTime2->diff($dateTime1, true);
echo "<br/>Différence : ", $dateDiff->format('%D'), "<br/>\n";

echo "<br/>";
$timestamp1 = $dateTime1->getTimestamp();
$timestamp2 = $dateTime2->getTimestamp();

echo "Date 1 : ";
echo strftime("%A %d %B %Y %T", $timestamp1), "<br/>\n";
echo "Date 2 : ";
echo strftime("%A %d %B %Y %T", $timestamp2), "<br/>\n";
?>
```

Affiche :

Date 1 après création : 24 January 2016 00:00:00

Date 2 après création : 24 January 2016 00:00:00

Date 2 après ajout : 29 January 2016 00:00:00

Différence : 05

Date 1 : Mercredi 24 janvier 2016 00:00:00

Date 2 : Vendredi 29 janvier 2016 00:00:00

2.13 Les chaînes de caractères

Une chaîne de caractère est une suite de caractères. Les constantes caractères sont délimitées par des guillemets("").

Comme certains caractères font partie du langage PHP ou HTML en tant que caractères spéciaux, il faut utiliser que caractère \ comme caractère d'échappement.

Caractère	Description
\n ou PHP_EOL(symbole correct de "End Of Line", pour CLI ou logs)	Changement de ligne (LF)
\r ou PHP_EOL	Retour chariot (CR)
\t	Tabulation
\\	Pour afficher \
\\$	Pour afficher \$
\"	Pour afficher "

Pour Windows un saut de ligne c'est \r\n.

Sous Unix ou Linux \n suffit.

Quelques fonctions de base sur les chaînes de caractères.

Fonction	Type renvoie	Description
substr("chaîne", départ[, nombre])	String	Extrait une sous-chaîne. Si départ est négatif, on commence par la fin de la chaîne.

<code>strlen("chaîne")</code>	Int	Calcule la longueur d'une chaîne.
<code>substr_count("chaîne" , "sous-chaîne")</code>	Int	Calcule le nombre d'occurrences d'une sous-chaîne.
<code>trim("chaîne")</code> <code>ltrim("chaîne")</code> <code>rtrim("chaîne")</code>	String	Elimine les espaces avant et après, à gauche, à droite Trim enlève en plus les <code>\r\n</code> , <code>\t</code> ,...
<code>strtoupper("chaîne")</code>	String	En majuscules.
<code>strtolower("chaîne")</code>	String	En minuscules.
<code>ucfirst("chaîne")</code>	String	En majuscule le premier caractère et ne modifie pas le reste.
<code>ucwords("chaîne")</code>	String	Met en majuscules le premier caractère de chaque mot du texte et ne modifie pas le reste.
<code>strpos("chaîne", "chaîne recherchée" [,offset])</code>	Int	Recherche la première sous-chaîne dans une chaîne. Renvoie la position ou false si non trouvée.
<code>strrpos("chaîne", "sous-chaîne" [,offset])</code>	Int	Recherche la dernière sous-chaîne dans une chaîne. Renvoie la position ou false si non trouvée.
<code>str_replace("chaîne recherchée", "Chaîne de remplacement", "Chaîne" [, int &\$count])</code>	String	Remplace une sous-chaîne dans une chaîne.
<code>str_repeat("car", n)</code>	String	Renvoie une chaîne composée de n "car".
<code>str_pad("chaîne", n, "car")</code>	String	Complète la chaîne avec tel caractère jusqu'à ce que la chaîne soit de longueur n.
<code>chr(n)</code>	Char	Retourne un caractère ascii par rapport à son code.
<code>ord(char)</code>	Int	Retourne le code ascii du caractère.

NB : on peut considérer une chaîne de caractères comme un tableau dans la mesure où il est possible de manipuler chaque caractère avec la fonction **substr(chaîne, position,1)** sachant que le premier caractère a la position 0.

Pour supprimer le dernier caractère **substr(chaîne,0,-1)**

c.f. Aussi `stripos()`, `strripos()`, `str_ireplace` – manipulation case **insensible**.

PHP – Introduction

Exemple :

```
<?php
    echo "<br />Code département : ", substr("75011", 0, 2);
    echo "<br />Code Commune : ", substr("75011", 2);
    echo "<br />Le premier o de bonsoir : ", strpos('Bonjour','o');
    echo "<br />Le dernier o de bonsoir : ", strrpos('Bonjour','o');
    echo "<br />Majuscules : ", strtoupper('Bonjour');
    echo "<br />Minuscules : ", strtolower('Bonjour');
    echo "<br />Nom propre : ", ucfirst('bonjour');
    echo "<br />Noms propres : ", ucwords('bonjour tout le monde');
    echo "<br />Trim : *", trim(' bonjour '), "*";
    echo "<br />Longueur de bonjour : ", strlen('bonjour');
    echo "<br />Nombre de o dans bonjour : ", substr_count('bonjour','o');

    echo "<br />Extrait à partir de la première occurrence : ", strchr('il
était une fois une marchande de foie dans la ville de ...','foi');
    echo "<br />Extrait à partir de la dernière occurrence : ", strrchr('il
était une fois une marchande de foie dans la ville de ...','foi');

    echo "<br />chr(65) : ", chr(65);
    echo "<br />ord('A') : ", ord('A');
?>
```

Exercices :

1. Afficher les trois derniers caractères d'une chaîne.
2. Afficher une chaîne en alternant majuscule et minuscule (bonjour → BoNjOuR). c.f. le FOR.

Corrigé

Quelques fonctions sur les nombres

Fonction	Description
Pi()	Le nombre PI.
Abs(n)	Valeur absolue.
Round(n[, p])	Valeur arrondie. Round(3.14) renvoie 3, Round(3.14,1) renvoie 3.1. Round(6.4) renvoie 6. Round(6.6) renvoie 7.
Floor(n)	Valeur arrondie à l'entier inférieur. Floor(6.4) renvoie 6. Floor(6.6) renvoie 6.
Ceil(n)	Valeur arrondie à l'entier supérieur. Ceil(6.4) renvoie 7. Floor(6.6) renvoie 7.
Pow(n, puissance)	Elévation à la puissance.
Sqrt(n)	Racine carrée.
Min(tableau)	Valeur minimum d'un tableau (Accepte aussi comme argument une liste de valeurs).
Max(tableau)	Valeur maximum d'un tableau (Accepte aussi comme argument une liste de valeurs).
Cos(n)	Cosinus.
Sin(n)	Sinus.
Tan(n)	Tangente.
Rand(), rand(min, max)	Génère une valeur aléatoire entre 0 et 32768 (range of Int) ou entre min et max inclus.

PHP – Introduction

Les fonctions de recherche et de remplacement

str_replace()

```
$resultat = str_replace($motif, $remplacement, $origine [, $compteur]);
```

```
// Avec une chaîne de caractères
$origine = "Il fait mauvais, il fait même très mauvais";
$motif = "mauvais";
$remplacement = "beau";
$resultat = str_replace($motif, $remplacement, $origine, $compteur);
echo "<hr />";
echo "$origine --> $resultat : <strong>$compteur</strong> occurrences
remplacées.";
```

Affiche : Il fait mauvais, il fait même très mauvais --> Il fait beau, il fait même très beau : 2 occurrences remplacées.

```
// Avec un tableau
$origine = "Il fait mauvais, il fait même très mauvais";
$motif = array("fait", "mauvais");
$remplacement = array("fera", "beau");
$resultat = str_replace($motif, $remplacement, $origine, $compteur);
echo "<hr />";
echo "<strong>$compteur</strong> occurrences remplacées.";
foreach($resultat as $valeur){
    echo "<br /> $valeur";
}
```

Affiche : 4 occurrences remplacées.

il fera beau

il fera même très beau