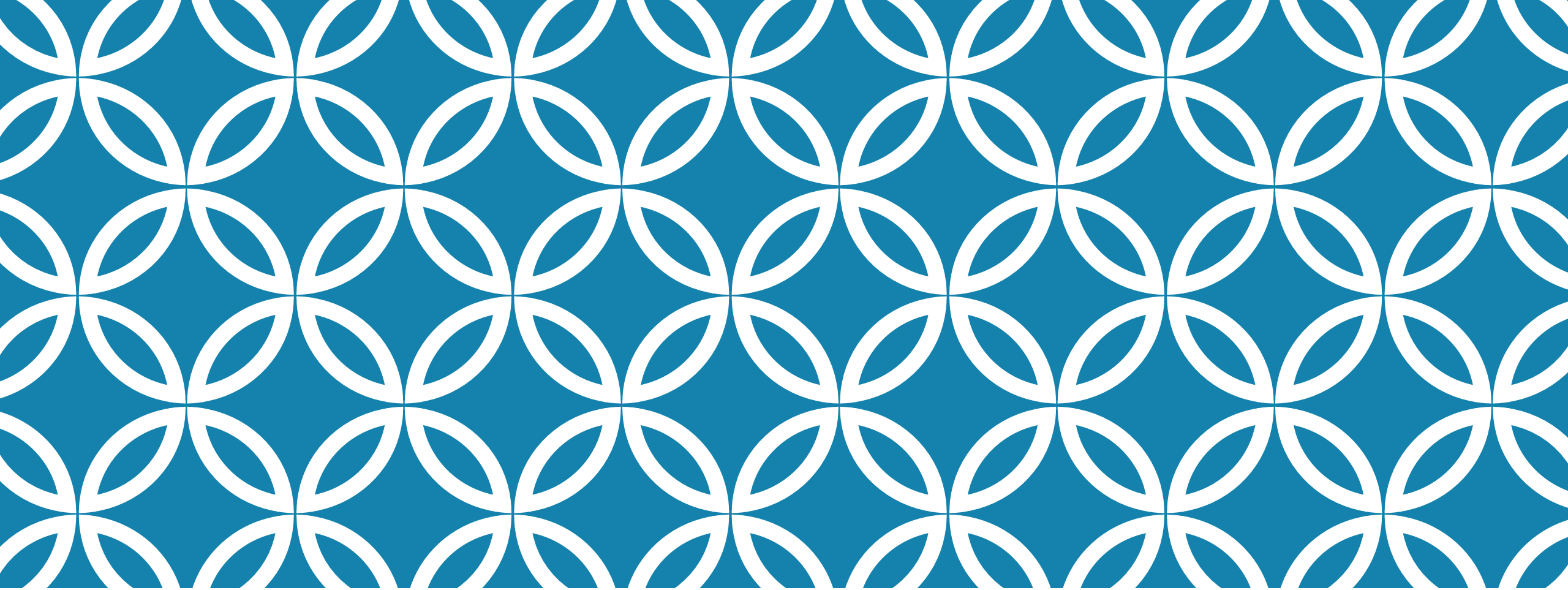


INTRODUCTION AU WEB

PHP

Par Jean-Pierre Lozi

Basé sur les cours d'Andrea Tettamanzi et Philippe Renevier

LE LANGAGE PHP

- PHP = Hypertext Preprocessor (acronyme récursif = **P**HP is a **H**ypertext **P**reprocessor)
- Créé par Rasmus Lerdof en 1994 (PHP voulait en fait dire Personal Home Page au début…)
- Version actuelle : PHP 5. PHP 7 va sortir bientôt (sauté une version…)
- Site officiel : http://www.php.net
- En 1998, 1% des serveurs web dans le monde avaient PHP d'installé
- En 2013, 39% des sites dans le monde utilisent PHP !
- Probablement le langage de script côté serveur le plus populaire
  - D'autres solutions existent : ASP de Microsoft, JSP basé sur java, scripts CGI (qui peuvent être écrits dans n'importe quel langage, comme du C par exemple, puis compilés…)

EXÉCUTION CÔTÉ SERVEUR

- On a vu JavaScript, qui est exécuté côté client
  - Les scripts sont téléchargés avec la page, et exécutés par le navigateur
- PHP est quand à lui exécuté côté serveur
  - **Les scripts sont toujours *stockés* et *exécutés* sur le serveur web par l'interpréteur PHP !**
- Idée de PHP : un script PHP génère une page HTML
- Utilise souvent des données stockées sur le serveur, soit dans des fichiers, soit dans des bases de données : **selon la requête de l'utilisateur et les données sur le serveur, page générée différente !**
- Par exemple : un forum. Tous les messages du forum sont stockés dans une base de données du serveur. Lorsqu'un utilisateur veut afficher un fil de discussion, le script PHP va chercher les bons messages dans la base de données, et générer la page.

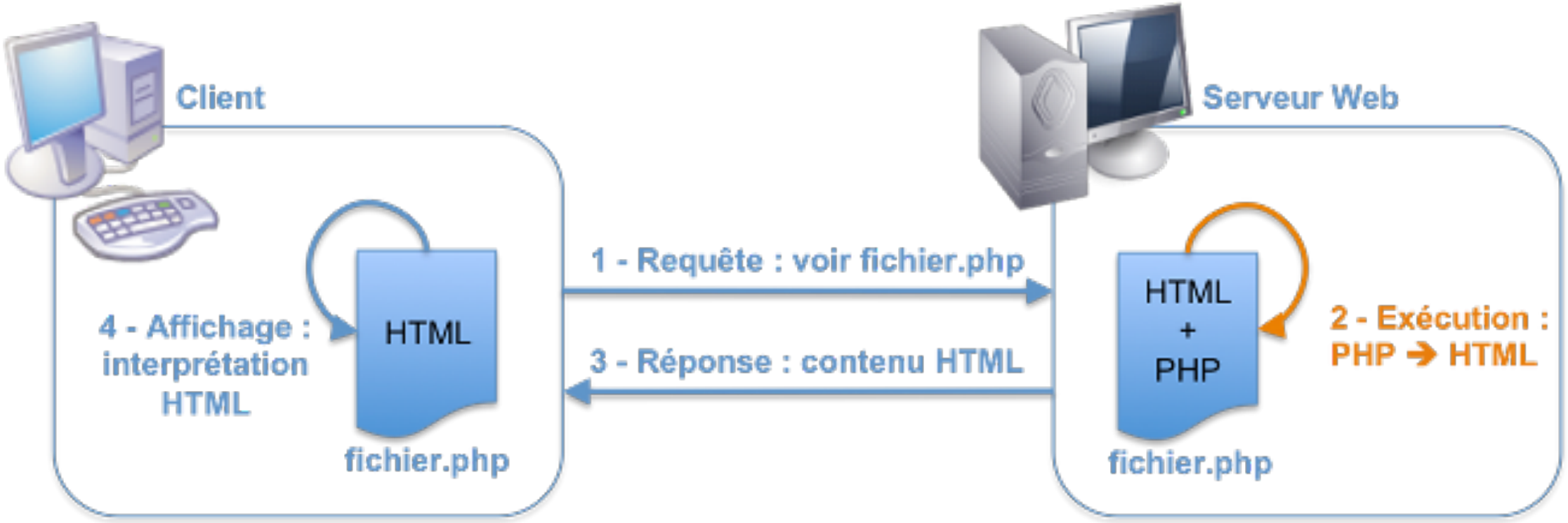
EXÉCUTION CÔTÉ SERVEUR

- Autre exemple : un site comme Facebook. Toutes les infos sur tous les comptes stockés sur les serveurs de Facebook. Lorsque vous vous connectez, un script sur le serveur génère une page HTML personnalisée pour vous : elle contiendra des posts récents de vos amis.
- Même chose pour votre banque en ligne (infos de compte stockées sur les serveurs de la banque), un moteur de recherche (le script fait appel au moteur de Google sur les serveurs et génère une page HTML avec vos résultats), etc.
- **En général :** PHP fait fonctionner les sites dynamiques en générant les pages web que demande l'utilisateur et JavaScript est plutôt utilisé pour de petits scripts sur la page permettant de la rendre plus interactive (menus déroulants complexes, cartes interactives, etc.)
- *Disclaimer : en fait plus compliqué que ça, frontière entre programmation client et serveur de plus en plus floue de nos jours : JavaScript peut faire des requêtes distantes dans un script du navigateur, récupérer des infos et les afficher, donc on pourrait par exemple programmer un forum en JavaScript, même si ce serait une implémentation assez exotique…*

EXÉCUTION CÔTÉ SERVEUR

- Sur demande d'un document PHP :
  - Le serveur identifie qu'il s'agit d'un fichier PHP (par défaut car l'extension est .php)
  - Il recherche les « balises » `<?php ... ?>`
  - Il lance l'interpréteur pour ces balises. Ces balises contiennent des **scripts PHP qui génèrent du HTML**
  - Chacun de ces scripts PHP est remplacé par le HTML qu'il a généré
  - S'il y avait aussi du HTML dans le fichier (hors des « balises » `<?php ... ?>`), celui ci n'est pas altéré
  - Au final, il ne reste plus que du HTML dans le document
  - **C'est ce fichier HTML qui est envoyé à l'utilisateur !**
- Cela veut dire que l'utilisateur d'un site écrit en PHP ne reçoit que du HTML (et CSS/JS).
- Il ne peut pas voir les scripts PHP, qui sont exécutés côté serveur, et qui n'ont servi qu'à générer du HTML !

EXÉCUTION CÔTÉ SERVEUR



Source : www.enseignement.polytechnique.fr

EXÉCUTION CÔTÉ SERVEUR

- Mais au fait, qui est le serveur ?
  - Pour vous, www-mips ! C'est sur cette machine que sont stockés vos fichiers web !
  - La preuve : vous y accédez par http://www-mips.unice.fr/~nom_utilisateur
- Mais on n'a jamais copié nos fichiers sur www-mips !
  - Non, parce que toute l'université utilise un *système de fichiers partagé*
  - Ce qui veut dire que vos stockées sur des machines dédiées, et accessibles par toutes les machines de l'université, comme si elles étaient locales (tout passe par le réseau…)
  - Ce qui veut dire que lorsque vous éditez des fichiers dans www, vous pouvez voir le résultat sur toute machine (de salle de TP par exemple) comme si les modifications étaient locales.
  - **Même chose pour www-mips ! Il voit vos fichiers dans www/ comme des fichiers locaux !**
  - Il peut donc envoyer les fichiers HTML et CSS à des clients distants, et il peut exécuter les scripts PHP dans www/, pour générer du HTML et envoyer le résultat…

Université Nice Sophia Antipolis

SYNTAXE

- Syntaxe de PHP assez proche de celle de JavaScript
  - En fait, syntaxes de C, C++, Java, JavaScript, PHP, etc., proches : beaucoup de langages se ressemblent. Pas R (désolé…)
- Quelques différences de syntaxe :
  - En PHP, on ne déclare pas les variables. Si on accède à une variable que l'on n'a jamais déclarée, elle est créée sur le champ avec une valeur par défaut !
  - On met toujours un signe $ devant les noms de variables
  - Concaténation avec un point au lieu d'un + !
- Fonction PHP la plus courante : `echo` permet d'écrire du HTML !
  - Par exemple : `echo "<p>Hello, world!</p>";`
  - Crée un paragraphe qui contient « Hello, world! »
- Autre fonction php : `date()` renvoie la date courante sous forme de chaîne de caractères

UN EXEMPLE, PUREMENT EN PHP

```php
<?php
    echo "<!DOCTYPE html>\n\n";
    echo "<html>\n";
    echo "<head>\n";
    echo "<meta charset="utf-8" />\n";
    echo "<title>HTML avec PHP</title>\n";
    echo "</head>\n\n";
    echo "<body>\n";
    echo "<h1>HTML + PHP</h1>\n";
    echo "<p>Nous sommes le "
         . date ("j/m/Y") . "</p>\n";
    echo "</body>\n";
    echo "</html>\n";
?>
```

Côté serveur

```html
<!DOCTYPE html>

<html>
<head>
<meta charset="utf-8" />
<title>HTML avec PHP</title>
</head>

<body>
<h1>HTML + PHP</h1>
<p>Nous sommes le 14/11/2015</p>
</body>
</html>
```

Côté client

UN EXEMPLE, HTML + PHP

```php
<!DOCTYPE html>

<html>
<head>
<meta charset="utf-8" />
<title>HTML avec PHP</title>
</head>

<body>
<h1>HTML + PHP</h1>
<p>Nous sommes le <?php echo date("j/m/Y") ?></p>
</body>
</html>
```

Côté serveur

```html
<!DOCTYPE html>

<html>
<head>
<meta charset="utf-8" />
<title>HTML avec PHP</title>
</head>

<body>
<h1>HTML + PHP</h1>
<p>Nous sommes le 14/11/2015</p>
</body>
</html>
```

Côté client

UN EXEMPLE, HTML + PHP, VARIANTE

```php
<?php
      // Calcul préalable
      $date = "Nous sommes le ";
      $date .= date("d/m/Y") . "</p>";
?>
<!DOCTYPE html>

<html>
<head>
<meta charset="utf-8" />
<title>HTML avec PHP</title>
</head>

<body>
<h1>HTML + PHP</h1>
<?php echo $date ?>
</body>
</html>
```

Côté serveur

```html
<!DOCTYPE html>

<html>
<head>
<meta charset="utf-8" />
<title>HTML avec PHP</title>
</head>

<body>
<h1>HTML + PHP</h1>
<p>Nous sommes le 14/11/2015</p>
</body>
</html>
```

Côté client

SYNTAXE

- Pour les commentaires : même chose que JavaScript (`//` ou `/* */`)
- Les blocs : entre `{}`, pas nécessaire si une seule instruction (dans les `if`, `for`, `while`…)
  - Toujours nécessaire pour les fonctions !
- Opérateurs : similaires à JavaScript (=, ==, *, +, -, /, %, +=, ++, …)
- Créer et appeller une fonction, similaire à JavaScript :

```php
<?php
    function disBonjour($qui) {
        echo "Bonjour $php !";
    }

    disBonjour("toto");
?>
```

UN EXEMPLE DE FONCTION PHP : DATE()

- `string date(string $format [, int $timestamp = time() ])`

Renvoie une date sous forme d'une chaîne, au format donné par la chaîne `$format`. La date est fournie par le `$timestamp`, au format UNIX timestamp. Par défaut, date/heure courantes utilisées.
UNIX timestamp = nombre de secondes depuis le premier janvier 1970.

```
// Aujourd'hui, le 12 April 2006, 10:16:18 am
$aujourdhui = date("F j, Y, g:i a"); // April 12, 2006, 10:16 am
$aujourdhui = date("m.d.y"); // 04.12.06
$aujourdhui = date("j, m, Y"); // 12, 04, 2006
$aujourdhui = date("Ymd"); // 20060412
$aujourdhui = date('\C\'\e\s\t\ \l\e\ jS \j\o\u\r\.'); // C'est le 12th jour.
$aujourdhui = date("D M j G:i:s T Y"); // Wen Apr 12 10:16:18 Paris, Madrid 2006
$aujourdhui = date("H:i:s"); // 10:16:18

// Notation française
$aujourdhui = date("d/m/y"); // 12/04/06
$aujourdhui = date("d/m/Y"); // 12/04/2006
```

UN EXEMPLE DE FONCTION PHP : STRTOTIME()

- `int strtotime(string $time[, int $now = time() ])`

Essaie de lire une date au format anglais US dans la chaîne $time, et de la transformer en timestamp UNIX relativement au timestamp now, ou à la date courante si ce dernier est omis.

```php
$now = strtotime("now");
$Xmas = strtotime("25 december 2015");
$Xmas = strtotime("2015-12-25"); // Même chose
$Xmas = strtotime("2015-12-25 11:50:00"); // Idem avec l'heure
$nextWeek = strtotime("+1 week");
$nextThursday = strtotime("next Monday");
```

EXERCICE : CALCUL DE DATE

Dans le fichier `date.php`, afficher jours et de secondes restants avant le 21 juin 2016.

Quelques indications :

- Notez qu'il y a 60 * 60 * 24 = 86 400 secondes dans un jour
- La fonction `floor()` de php permet d'obtenir la valeur entière inférieure (`ceiling()` pour la valeur entière supérieur). Par exemple : `$val = floor(9.999); // 9.0`
- Pour convertir un flottant en entier, transtypage avec `(int)` :

```php
echo 9.0; // Affiche "9.0"
echo (int)(9.0); // Affiche "9"
```

- On veut le résultat dans un paragraphe HTML

EXERCICE : CALCUL DE DATE

```
<!DOCTYPE html>
<html>
<head><meta charset="utf-8" /><title>Date</title></head>
<body>
<?php
    $dateString = "21 june 2016";
    $date = strtotime($dateString);
    $diff = $date - strtotime("now");

    $resteJ = $diff / (60*60*24);
    $resteJ = (int)floor($resteJ);
    $resteS = $diff - $resteJ * (60*60*24);

    $tempsRestant = "<p>Il reste " . $resteJ . " jours et "
                    . $resteS . " secondes avant le $dateString</p>";
?>
</body>
</html>
```

CHAÎNES DE CARACTÈRES

```
$chaine = "une chaîne de caractères";
```

- `strlen($chaine)` pour en connaître la taille
- En fait, une chaîne = un tableau de caractères : on peut accéder au (i+1)ème caractère avec `$chaine[$i]` (syntaxe des tableaux similaire à celle de JavaScript). Par exemple :

```
$chaine = "toto";
$chaine[1] = $chaine[3] = "i";
echo $chaine; // Affichera "titi"
```

- On a vu la concaténation avec le point (.) et l'opérateur .= :

```
$chaine = $chaine . " !";   ⇔   $chaine .= " !";
```

CHAÎNES DE CARACTÈRES

- Caractères spéciaux : retour à la ligne = `\n`, tabulation = `\t`
- Possible d'utiliser soit des guillemets simples, soit des guillemets doubles
- Mais comme les chaînes de caractère en PHP contiennet souvent du HTML, et que le HTML contient beaucoup de guillemets doubles, on utilise plus souvent des guillemets simples en php pour éviter d'avoir à « échapper » les guillemets doubles de HTML ! Par exemple :

```php
// Compliqué et assez laid:
echo"<input type=\"text\" id=\"champ_entree\" value=\"Texte ici\" />";

// Plus simple et plus propre :
echo '<input type="text" id="champ_entree" value="Texte ici" />';
```

CHAÎNES DE CARACTÈRES

- Mais attention : **guillemets simples et doubles ont un comportement différent !**
  - Dans les guillemets doubles, on peut utiliser des variables, et on peut utiliser des accolades pour délimiter les noms de variables : `$` et `{}` sont interprétés. Les caractères spéciaux (`\n`, `\t`) sont également interprétés. Possible d'utiliser `\"` et `\\` pour un guillemet double ou un antislash.

```php
$animal = "serpent";
echo "lion\n$animal"; // Affiche "lion" et "serpent" sur deux lignes

// Comment afficher "le lion mange les serpents ?"

echo "Le lion mange les $animals";
// Affiche "Le lion mange les " ! Car $animals n'existe pas

/* On peut utiliser des accolades pour isoler les noms de variables dans des
guillemets doubles */

echo "Le lion mange les {$animal}s";
// Affiche bien "Le lion mange les serpents"
```

CHAÎNES DE CARACTÈRES

- Mais attention : **guillemets simples et doubles ont un comportement différent !**
  - Dans les guillemets simples, rien n'est interprété, sauf `\'` pour afficher un guillemet simple et `\\` pour afficher un antislash (\).

```php
$animal = "serpent";

echo 'Le lion mange l\'$animal';
// Affiche "Le lion mange l'$animal"...
echo 'Le lion mange les {$animal}s';
// Affiche "Le lion mange les {$animal}s"...

// Par conséquent, il faut utiliser la concaténation :
echo 'Le lion mange l\'' . $animal;
echo 'Le lion mange les ' . $animal . 's';
```

CHAÎNES DE CARACTÈRES

Des fonctions courantes avec les chaines de caractère (en plus de `strlen()`) :

- `string substr(string $string , int $start [, int $length ])`
  Renvoie une sous-chaîne de la chaîne `$string`, commençant à l'indice `$start` et de longueur `$length`. Si `$start` est négatif, part de la fin ! Si `$length` est négatif, dit jusqu'où aller en partant de la fin de la chaîne...

```php
echo substr("abcdef", 1); // bcdef
echo substr("abcdef", 1, 3); // bcd
echo substr("abcdef", 0, 4); // abcd
echo substr("abcdef", 0, 8); // abcdef
echo substr("abcdef", -1, 1); // f
echo substr("abcdef", -2); // retourne "ef"
```

```php
echo substr("abcdef", -3, 1); // retourne "d"
echo substr("abcdef", 0, -1); // retourne "abcde"
echo substr("abcdef", 2, -1); // retourne "cde"
echo substr("abcdef", 4, -4); // retourne false
echo substr("abcdef", -3, -1); // retourne "de"
```

CHAÎNES DE CARACTÈRES

- `mixed str_replace(mixed $search, mixed $replace, mixed $subject [, int &$count ])`

Cette fonction cherche, dans la chaîne `$subject`, les occurences de `$search`, et les remplace par `$replace`. Chaîne `$subject` non modifiée, mais renvoie une chaîne avec les occurences remplacées. Si on passe un paramètre `$count`, il contiendra le nombre de remplacements effectués.

Possible d'utiliser des tableaux pour `$search` et pour `$replace` pour remplacer plusieurs choses d'un coup (on verra les tableaux plus tard).

Remarquez que `$count` est passé par référence ici (c'est ce que veut dire le `&`) et donc il peut être modifié par la fonction : une fonction peut produire plusieurs valeurs en n'ayant qu'une valeur de retour.

Exemple de `str_replace()` :

```php
echo str_replace("world","Peter","Hello world!"); // Affiche : "Hello, Peter!"
```

CHAÎNES DE CARACTÈRES

- `int strcmp(string $str1, string $str2)`
  Permet de comparer deux chaînes alphabétiquement. Si la fonction renvoie un nombre négatif, `$str1` est avant `$str2` (dans l'ordre alphabétique), si la fonction renvoie un nombre positif, `$str1` et `$str2` sont identiques, et enfin s la fonction renvoie un nombre positif, `$str1` est après `$str2`.

- `mixed strpos(string $haystack, mixed $needle [, int $offset = 0 ])`
  Recherche la chaîne `$needle` dans `$haystack`, en commençant à l'indice `$offset`.
  Renvoie la position ou `false` si la chaîne n'a pas été trouvée

- `string str_repeat(string $input, int $multiplier)`
  Renvoie une chaîne répétée un nombre `$multiplier` fois

- `string trim(string $str [, string $character_mask = " \t\n\r\0\x0B" ])`
  Supprime espaces (ou autres caractères) en début et fin de chaîne

Toutes les fonctions pour chaînes sur http://php.net/manual/fr/ref.strings.php
Il y en a plus de 100 ! Un avantage de PHP, il y a une fonction pour tout…

Université Nice Sophia Antipolis

TABLEAUX

- Pour déclarer un tableau, on utilise `array()`. Par exemple :

```
$tableau = array("pommes", "poires", "oranges");
echo $array[1]; // Affiche "poires"
$array[2]{6} =  "r"; // Affiche "oranger"
```

- Nombre d'éléments dans un tableau : avec `count()`

```
for ($i = 0 ; $i < count($array) ; $i++)
    echo $array[$i]. " ";
/* Affiche "pommes poires oranger". */
```

- Pour un tableau en deux dimensions, logiquement :

```
$tableau = array (
    array('O', 'X', ' '),
    array('X', 'X', 'O'),
    array('O', 'O', 'X')
);
$tableau[0][2] = 'X';
```

TABLEAUX

- Depuis PHP 5.4, au lieu de array(), on peut utiliser des crochets pour déclarer un tableau comme en JavaScript :

```
$tableau = ["pommes", "poires", "oranges"];
$tableau = [
    ['O', 'X', ' '],
    ['X', 'X', 'O'],
    ['O', 'O', 'X']
];
```

- En plus des tableaux avec des indices entiers, PHP permet d'utiliser des chaînes de caractères comme indice des tableaux ! Par exemple :

```
$array = array(
    "foo" => "bar",
    "bar" => "foo"
);
echo $array["foo"]; // Affiche "bar";
echo $array["bar"]; // Affiche "foo";

/* Ajoute un élément d'indice foobar et de valeur 1.3 : */
$array["foobar"] = 1.3;
```

TABLEAUX

- En fait, les tableaux peuvent mélanger indices entiers et textuels :

```php
$array = array(
    "foo" => "bar",
    100 => -100,
    -100 => 100,
    "bar" => "foo", // Virgule finale OK
);

echo $array[100]; // Affiche "-100"
echo $array["-100"]; // Marche aussi ! Affiche "100"
```

- Pas nécessaire de spécifier les indices pour chacun des éléments :

```php
$array = array(
            "a",
            "b",
        6 => "c",
            "d",
);

var_dump($array);
```

```
array(4) {
    [0]=> string(1) "a"
    [1]=> string(1) "b"
    [6]=> string(1) "c"
    [7]=> string(1) "d"
}
```

TABLEAUX

- Possible d'ajouter un élément à la fin d'un tableau avec [], et possibilité de supprimer un élément avec `unset()` (équivalent au `delete` de JavaScript) :

```php
$arr = array(5 => 1, 12 => 2);
$arr[] = 56;     // Identique à $arr[13] = 56;
                 // à cet endroit du script
$arr["x"] = 42; // Ceci ajoute un nouvel élément au
                 // tableau avec la clé "x"
unset($arr[5]); // Ceci efface l'élément du tableau
unset($arr);     // Ceci efface complètement le tableau
```

- Si on a une fonction qui renvoie un tableau, possible d'accéder directement à une case de ce tableau sans avoir besoin de stocker le résultat dans une variable intermédiaire :

```php
function getArray() {
    return array(1, 2, 3);
}

// Depuis PHP 5.4
$secondElement = getArray()[1];
```

TABLEAUX

- Affichage d'un tableau et de ses éléments :

```php
$fruits = array('fraise' => 'rouge', 'banane' => 'jaune');
echo "$fruits\n";
// Erreur : on ne peut pas afficher un tableau comme ça

print_r($fruits);
/* Affiche :
Array (
    [fraise] => rouge
    [banane] => jaune
)
*/

var_dump($fruits);
/* Affiche :
array(2) {
  ["fraise"]=>
  string(5) "rouge"
  ["banane"]=>
  string(5) "jaune"
*/
```

```php
echo "Une banane est $fruits['banane'].\n";
// Erreur !!!!!!

echo "Une banane est {$fruits['banane']}.\n";
// Affiche : "Une banane est jaune."

echo 'Une banane est {$fruits["banane"]}.\n';
// Affiche : "Une banane est {$fruits["banane"]}."  !

echo "Une banane est " . $fruits['banane']  . ".\n";
// Affiche : "Une banane est jaune."
```

TABLEAUX

Quelques fonctions qui utilisent des tableaux :

- `array explode(string $delimiter, string $string [, int $limit])`
  Permet de découper des chaînes de caractères en sous-chaînes délimitées par le délimiteur `$delimiter` (avec la possibilité de s'arrêter après un certain nombre d'éléments avec `$limit`). Très souvent utile ! La fonction `implode()` fait l'inverse. Un exemple de `explode()` :

```php
$liste = "jaune,bleu,vert,rouge,violet";
$tableau = explode(",", $liste);
echo $tableau[3]; // Affiche "rouge"
```

- `bool in_array ( mixed $needle , array $haystack [, bool $strict = FALSE ] )`
  Cherche si l'élément `$needle` se trouve dans le tableau `$haystack` et renvoie `true` si c'est le cas. Si le paramètre `$strict` est à `true`, vérifie que le type est également le bon (comparaison avec `===` au lieu de `==`). Un exemple de `in_array()` :

```php
$utilisateurs_connus = ["Dupont", "Martin", "Smith", "Sanchez"];
if(in_array($utilisateur, $utilisateurs_connus)) {
    echo "<p>Vous êtes identifié !</p>";
}
```

TABLEAUX

- `array array_values ( array $array )`

Réindexe les valeurs d'un tableau. Pratique quand on efface un élément !

```
$a = array(0 => 'un', 1 => 'deux', 2 => 'trois');

unset($a[1]);
/* Cela va produire un tableau qui aurait été $a = array(0 => 'un', 2 =>
'trois') et non pas $a = array(0 => 'un', 1 =>'trois'). */

$b = array_values($a);
/* Maintenant b est le tableau array(0 => 'un', 1 =>'trois'). */
```

- Toutes les fonctions pour les tableaux : https://secure.php.net/manual/fr/ref.array.php

Encore une fois, un très grand nombre…

PASSER DES PARAMÈTRES AUX PAGES

- Chaque page est exécutée indépendamment : comment permettre une continuité de la navigation ? I.e., vous voulez que l'utilisateur puisse demander quelque chose (« afficher les résultats pour la recherche de la chaîne "toto" sur mon site », « afficher les 30 dernières opérations sur mon compte en banque », etc.), et que la page qui est chargée sache ce qu'a demandé l'utilisateur, pour pouvoir effectuer la bonne action (« toto », « 30 » dans les exemples).
- En clair : les scripts PHP ont souvent besoin de données d'entrée pour produire une page HTML en sortie. Ceci est effectué de deux manières.
  - **Via des formulaires :** l'utilisateur remplit des champs texte, coche des boutons radio ou des boîtes, sélectionne des éléments dans des menus, et lorsqu'il clique sur un bouton, un script PHP est chargé, et ce script reçoit toutes les infos du formulaire, sur lesquelles il peut travailler (il peut par exemple les stocker dans une base de données).
  - **Via l'URL :** on peut directement passer des variables dans l'URL à un script. Vu qu'une page peut être générée dynamiquement (en PHP par exemple), les liens sont générés et peuvent contenir des variables passées aux pages suivantes qui dépendent du contexte. Cas le plus simple !

PASSER DES PARAMÈTRES AUX PAGES

- On a déjà vu rapidement dans le premier cours comment passer des paramètres dans une URL. Il faut ajouter `?` à la fin de l'URL, puis `paramètre=valeur` pour chaque paramètre, en séparant chaque paramètre par `&`, par exemple :
`http://www-mips.unice.fr/~login/afficher.php?page=contact&format=mobile`
ou :
```
<a href="afficher.php?page=contact&format=mobile">Contact</a>
```

- Ensuite, dans le script PHP, on peut récupérer les valeurs dans le tableau superglobal `$_GET`. Les clés seront le nom des paramètres, et les valeurs, leurs valeurs… Par exemple :
```
echo $_GET["page"];  // contact
echo $_GET["format"];  // mobile
```

- Souvent, on ne sait pas si une variable a été passée à une page ou pas. On peut le savoir en utilisant `isset()` qui permet de savoir si une variable est définie ou si un élément d'un tableau existe :
```
if (isset($_GET["page"]) echo $_GET["page"]; // contact
if (isset($_GET["format"]) echo $_GET["format"]; // mobile
```

Université Nice Sophia Antipolis

PASSER DES PARAMÈTRES AUX PAGES

- Attention, puisque le contenu des variables sont passées dans l'URL, elles doivent être encodées pour être passées dans l'URL. Par exemple, si vous avez des espaces, ceux-ci doivent être remplacés en `%20` ou en `+`. Si vous avez des caractères spéciaux, ceux-ci doient être également encodés. Par exemple cette URL est incorrecte :
  http://www-mips.unice.fr/~login/biographie.php?prenom=Antonín Leopold&nom=Dvořák
  Il faudra plutôt :
  http://www-mips.unice.fr/~login/biographie.php?prenom=Anton%C3%ADn%20Leopold&nom=Dvo%C5%99%C3%A1k

- Vous pouvez utiliser la fonction `urlencode()` de php pour encoder des chaînes de caractères pour leur utilisation dans une URL. Par exemple :

```php
/* On a $nom="Antonín Leopold" et $prenom="Dvořák" (données lues dans un
   fichier par exemple). On veut générer un lien. On fait : */
echo '<a href="biographie.php?prenom=' . urlencode($nom)
     . '&nom=' .urlencode($prenom) . '">';

echo "La biographie de $nom $prenom.</a>"
```

- La fonction inverse de `urlencode()` est `urldecode()` mais pas nécessaire pour des variables GET, automatiquement décodées avant d'être mises dans le tableau `$_GET` !

PASSER DES PARAMÈTRES AUX PAGES

```php
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Ma page perso</title>
</head>

<body>
<!-- Mon en-tête qui est le même dans toutes les pages -->

<!-- On pourrait mettre index.php, index.php?page=cv, etc.
index.php jamais nécessaire... -->
<a href="?">Accueil</a>
<a href="?page=cv">Mon CV</a>
<a href="?page=contact">Contact</a>

<?php
    if (!isset($_GET["page"]))
        $page="accueil.php";
    else
        $page = $_GET["page"] .".php";

    include($page);
?>

<!-- Mon pied de page qui est le même dans toutes les pages -->
</body>
</html>
```

index.php

```php
<h1>Mes études</h1>
<p>L1 MASS à Nice</p>

<h1>Mes hobbies</h1>
<p>La formule 1</p>
```

cv.php

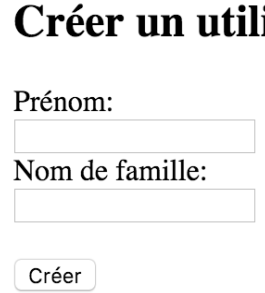
```php
<p>Pour me contacter,
<a href="fabrice.dupont@unice.fr">cliquez ici.</a></p>
```

contact.php

PASSER DES PARAMÈTRES AUX PAGES

- Pour les formulaires, deux méthodes : protocoles. GET fait la même chose que ce qu'on a vu jusqu'à présent, va encoder toutes les valeurs des champs du formulaire dans l'URL quand l'utilisateur le soumet. Il faut aussi donner le script PHP vers où est posté le formulaire. Par exemple :

```
<h2>Créer un utilisateur :</h2>
<form action="ajout_utilisateur.php" method="get">
    Prénom:<br />
    <input type="text" name="prenom" value=""> <br />
    Nom de famille:<br />
    <input type="text" name="nom" value=""> <br /><br />
    <input type="submit" value="Créer">
</form>
```

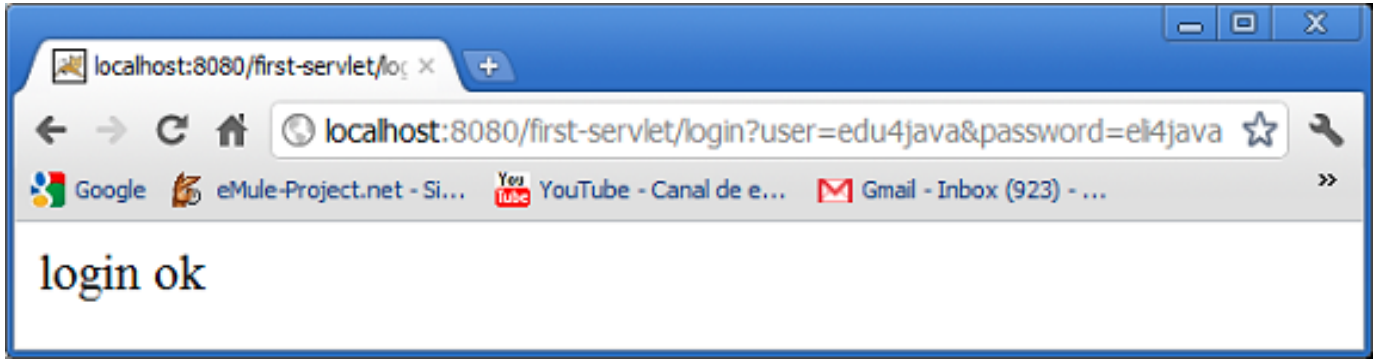


- Supposons que l'utilisateur entre « Jean » dans le champ texte `prenom` et qu'il entre « Dupont » dans le champ texte `nom`. Lorqu'il clique sur « Créer », la page suivante est chargée (on peut le voir dans la barre de navigation du navigateur) :
  http://www-mips.unice.fr/~login/ajout_utilisateur.php?nom=Jean&prenom=dupont

- La page `ajout_utilisateur.php` peut donc stocker `$_GET["nom"]` et `$_GET["prenom"]` dans une base de données, par exemple…

PASSER DES PARAMÈTRES AUX PAGES

- Problème de la méthode GET : tout est encodé dans l'URL ! Ce qui veut dire qu'une fois le formulaire soumis, lorsque le navigateur charge la page suivante, on voit toutes les variables dans la barre de navigation du navigateur…



- C'est tout d'abord problématique lorsque l'utilisateur envoie beaucoup de données. Cela fait des URLs interminables et c'est assez peu esthétique et peu confidentiel : supposez que vous écriviez un e-mail, que vous cliquiez sur « Envoyer » et dans la page suivante vous voyiez le contenu de votre e-mail dans la barre de navigation…

- C'est très problématique lorsque l'utilisateur entre un mot de passe dans un champ de mot de passe (`<input type="password" . . .`) : bien que seules des puces soient affichées dans le champ de mot de passe, lorsque l'utilisateur valide le formulaire, le mot de passe sera visible en clair dans la barre de navigation…

- Par conséquent, lorsqu'on a des formulaires demandant d'entrer beaucoup de données et/ou des données confidentielles, on préfèrera la méthode POST qui envoie le contenu des champs du formulaire directement dans la requête HTTP !

PASSER DES PARAMÈTRES AUX PAGES

- Pour utiliser la méthode `post`, on met seulement `method=post` au lieu de `method=get` dans la balise form. Puis on récupère les résultats dans le tableau superglobal `$_POST`, tout simplement.
- Dans l'URL dans la barre de navigation on ne verra que le nom de la page, aucune variables. Où sont-elles passées ? Directement dans la requête HTTP qui est invisible de l'utilisateur...
- Requête HTTP en GET :

```
GET /path/script.php?home=Cosby&favorite+flavor=flies  HTTP/1.0
From:  frog@jmarshall.com
User-Agent:  HTTPTool/1.0
Content-Type:  application/x-www-form-urlencoded
Content-Length:  32
```

- Requête HTTP en POST :

```
POST /path/script.php  HTTP/1.0
From:  frog@jmarshall.com
User-Agent:  HTTPTool/1.0
Content-Type:  application/x-www-form-urlencoded
Content-Length:  32

home=Cosby&favorite+flavor=flies
```

STRUCTURES CONDITIONNELLES

- Pour les `if`, pareil qu'en JavaScript :

```
$marquePluriel = ""; // chaine vide
if ($nbJour > 1) $marquePluriel = "s"; // S'il y a plus d'un jour
$texte = "<p>Il y a $nbJour jour$marquePluriel</p>";
```

- Un autre exemple :

```
$heure = date("G");

if ($heure < "12") echo "<p>Bonjour !</p>";
else if ($heure < "19") echo "<p>Bonne après-midi !</p>";
else if ($heure < "23") echo "<p>Bonne soirée !</p>";
else echo "<p><em>Bonne nuit !</em></p>";
```

Remarquez que l'on fait des comparisons numériques non pas sur des entiers mais sur des chaînes de caractère qui contiennent des nombes, et cela fonctionne : *PHP les convertit en nombre automatiquement !*

STRUCTURES CONDITIONNELLES

- Opérateur ternaire :

```
<condition> ? <que_renvoyer_si_vrai> : <que_renvoyer_si_faux>
```

- Par exemple, on pourra utiliser :

```php
$favoriteColor = isset($_GET["color"]) ? $_GET["color"] : "blue";
```

- Ce qui est quand même moins fastidieux que :

```php
if(isset($_GET["color"]))
    $favoriteColor = $_GET["color"];
else
    $favoriteColor = "blue";
```

- Pratique pour tout mettre sur une ligne lorsqu'on veut mettre une valeur ou une autre dans une variable d'après une condition... Mais n'en abusez pas ! Par exemple, à éviter :

```php
$message = $isWinner
    ? "Félicitations ! Vous avez gagné une somme considérable d'argent!"
    : "Désolé, vous avez perdu ! Mais vous aurez plus de chance la prochaine fois...";
```

STRUCTURES CONDITIONNELLES

- Lorsqu'on a beaucoup de choix, on peut utiliser un `switch` (existe aussi en JS) :

```
switch ($i) {
     case "apple":
          echo "\$i est une pomme...";
          break;
     case "bar":
          echo "\$i est une barre...";
          break;
     case "cake":
          echo "\$i est un gateau...";
          echo "<em>bon choix !</em>";
          break;
     default:
          echo "nourriture inconnue...";
}
```

```
switch($beer) {
     case "Grimbergen":
     case "Leffe":
          echo "Tiens, une bière Belge...";
     case "Guinness":
          echo "Bon choix !";
          break;
     case "33";
          echo "Mauvais choix !";
}
```

BOUCLES

- Les boucles `for` existent en PHP fonctionnent de la même manière qu'en JavaScript. On a aussi les mots-clé `break` et `continue`, qui ont la même signification. Ces quatre exemples de boucles `for` affichent les chiffres de 1 à 10 :

```php
/* exemple 1 */
for ($i = 1; $i <= 10; $i++)
{
    echo $i;
}

/* exemple 2 */
for ($i = 1; ; $i++)
{
    if ($i > 10) {
        break;
    }
    echo $i;
}
```

```php
/* exemple 3 */
$i = 1;
for (; ; ) {
    if ($i > 10) {
        break;
    }
    echo  $i;
    $i++;
}

/* exemple 4 */
for ($i = 1, $j = 0; $i <= 10; $j += $i, print $i, $i++);
```

BOUCLES

- Les boucles `while` marchent de la même manière qu'en JavaScript. Au passage, PHP a quelque chose que JavaScript n'a pas : on peut donner un argument à break pour dire de combien de boucles/`switch`es on veut sortir.

```php
for ($i = 0 ; $i < 100 ; $i++)
    for ($j = 0 ; $j < 100 ; $j++)
        for ($k = 0 ; $k < 100 ; $k++)
            if (calcul_magique($j, $k) > 1000)
                break 2;
```

```php
$i = 0;
while ( $i++ ) {
    switch ( $i ) {
        case 5:
            echo "At 5<br>\n";
            break 1; /* Exit only the switch. */
        case 10:
            echo "At 10; quitting<br>\n";
            break 2; /* Exit the switch and the while. */
        default:
        break;
    }
}
```

BOUCLES

- Une boucle bien spécifique à PHP… La boucle `foreach` permet d'itérer sur les éléments d'un tableau. Très utile… Deux syntaxes, avec ou sans clés :

```php
/* exemple foreach 1 : la valeur seulement */
$a = array(
     "un" => 1,
     "deux" => 2,
     "trois" => 3,
     "dix-sept" => 17 );

foreach ($a as $v) {
    echo "Valeur courante de \$a: $v.\n";
}

/* Affiche :
Valeur courante de $a: 1
Valeur courante de $a: 2
Valeur courante de $a: 3
Valeur courante de $a: 17
*/
```

```php
/* exemple foreach 2 : valeur + clé */

$a = array(
     "un" => 1,
     "deux" => 2,
     "trois" => 3,
     "dix-sept" => 17 );

foreach ($a as $k => $v) {
    echo "$k: $v, ";
}
/* Affiche : un: 1, deux: 2, trois: 3, dix-sept: 17, */
```

BOUCLES

- Attention : ordre du tableau ≠ ordre des clés !

```php
$fruits = array(
    0 => "lemon",
    2 => "orange",
    1 => "banana",
    3 => "apple" );

for ($i = 0 ; $i < count($fruits) ; $i++)
    echo "fruits[" . $i . "] = " . $fruits[$i] . ", ";
echo "\n";

foreach ($fruits as $key => $val)
    echo "fruits[" . $key . "] = " . $val . ", ";
echo "\n";
```

```
fruits[0] = lemon, fruits[1] = banana, fruits[2] = orange, fruits[3] = apple,
fruits[0] = lemon, fruits[2] = orange, fruits[1] = banana, fruits[3] = apple,
```

BOUCLES

- Attention : ordre du tableau ≠ ordre des clés ! Du coup, deux fonctions de tri. `sort()` trie le tableau sur ses valeurs, et réaffecte les clés 0, 1, 2,... aux valeurs triées. `asort()` trie le tableau sur ses valeurs, mais préserve les clés. `rsort()` et `arsort()` : ordre inverse.

```
Après sort :  fruits[0] = apple, fruits[1] = banana, fruits[2] = lemon, fruits[3] = orange,
Après asort : fruits[3] = apple, fruits[1] = banana, fruits[0] = lemon, fruits[2] = orange,
```

```php
$fruits = array(
    0 => "lemon",
    2 => "orange",
    1 => "banana",
    3 => "apple" );

$fruits_saved = $fruits;
sort($fruits);
echo "Après sort : ";
foreach ($fruits as $key => $val)
    echo "fruits[" . $key . "] = " . $val . ", ";
echo "\n";

$fruits = $fruits_saved;
asort($fruits);
echo "Après asort : ";
foreach ($fruits as $key => $val)
    echo "fruits[" . $key . "] = " . $val . ", ";
echo "\n";
```

BOUCLES

- Petite subtilité : l'opérateur ++. Il peut être utilisé avant ou après une variable. I.e., `$i++` ou `++$i`, par exemple. Seulement une différence si on lit la variable.
- Avec `$i++`, on lit la valeur de la variable d'abord, puis on l'incrémente. Alors qu'avec `++$i;` on incrémente d'abord la valeur de la variable, puis on la lit. Du coup, on a :

```
$i = 10; $a = $i++; // Maintenant $a vaut 10, et $i vaut 11
$i = 10; $a = ++$i; // Maintenant $a vaut 11 et $i vaut 11
```

- N'a pas d'influence dans une boucle for « classique » :

```
for($i = 0 ; $i<3 ; ++$i) var_dump $i;
for($i = 0 ; $i<3 ; $i++) var_dump $i;
```

- Mais aura une influence dans ce cas, par exemple :

```
for($i = 0 ; $i < 3 ; $j = ++$i ) var_dump($j); // NULL, 1, 2
for($i = 0 ; $i < 3 ; $j = $i++ ) var_dump($j); // NULL, 0, 1
```

FICHIERS

- Comment stocker des informations à long terme sur le serveur ? Par exemple, supposons qu'on veuille créer un forum. Où sont stockés les messages ?
- **Deux possibilités :** dans des fichiers ou dans une base de données.
- **Avantage des fichiers :** on peut y écrire absolument ce que l'on veut, il n'y a pas de format imposé. Désavantage : on doit créer un format « à nous », et le manipuler sans aucune aide. Code de bas niveau.
- **Avantage des bases de données :** plus simples à manipuler. Comme tout est stocké dans des tables bien définies, langage optimisé (SQL) facile d'utilisation pour accéder aux données.

FICHIERS

```
array glob(string $pattern[, int $flags = 0 ])
```

- Fonction pour récupérer tous les fichiers dont le chemin correspond au motif `$pattern`.
- `$pattern` est proche de celui d'un « `ls` » dans un terminal. Par exemple :
  - `*.php` = tous les fichiers php dans le dossier dans lequel s'exécute le script.
  - `../news/*.*` = on remonte d'un niveau dans la hiérarchie (`www/` si le script était dans `www/mon_dossier/` par exemple), on entre dans le répertoire news (donc `www/news/`), et on récupère tous les fichiers.
- Renvoie un tableau de chaînes de caractères, chaque chaîne représentant le nom/chemin de l'un des fichiers/répertoires qui ont été trouvé.
- Les options possibles (`$flags`) sont :
  - `GLOB_MARK` : ajoute un slash à la fin de chaque répertoire renvoyé
  - `GLOB_NOSORT` : renvoie les fichiers tels qu'ils sont trouvés dans le répertoire, sans les trier
  - `GLOB_NOCHECK` : renvoie `$pattern` si aucun fichier n'a été trouvé (au lieu d'un tableau vide)
  - `GLOB_NOESCAPE` : n'interprète pas `\` comme caractère d'échappement
  - `GLOB_BRACE` : permet d'utiliser par exemple `{'a','b','c'}` pour « matcher » un caractère qui est soit `a`, soit `b`, soit `c`
  - **`GLOB_ONLYDIR` : renvoie seulement les répertoires, pas les fichiers**
  - `GLOB_ERR` : Stop lors d'une erreur (comme des dossiers non lisibles), par défaut, les erreurs sont ignorées.

FICHIERS

- Que sont ces options (`GLOB_MARK`, `GLOB_NOSORT`, ...) ? Ce sont des constantes globales. C'est pour cela qu'on y accède sans `$`. Vous pouvez vous-même définir des constantes globales avec :

```php
define("FOO", "something");
define("FOO2", 1);
define("FOO_BAR", 3, 4);

echo FOO; // Affiche "something"
```

- Pour les options de `glob`, on peut en mettre plusieurs avec `|`, qui est l'opérateur binaire « ou »
- Comment ça marche ? Chaque option a en fait un seul bit qui vaut 1, et faire un ou entre elles active les bits qui correspondent à chaque option. La fonction `glob` lit ensuite chaque bit pour voir si chaque option est activée ou pas. **Technique utilisée par pas mal de fonctions !**

```
GLOB_ERR     = 0...0000001
GLOB_MASK    = 0...0000010
GLOB_NOSORT  = 0...0000100
```

```
GLOB_NOCHECK              = 0...0001000
GLOB_NOESCAPE             = 0...0100000
GLOB_ERR | GLOB_NOSORT    = 0...0101000
```

FICHIERS

- Quelques autres fonctions pour tester les propriétés des fichiers / répertoires :
  - `is_dir($chemin)` renvoie true ssi `$chemin` (par ex. `"images/a.png"`) est un répertoire
  - `is_file($chemin)` renvoie true ssi `$chemin` est un fichier
  - `is_readable($chemin)` renvoie true ssi `$chemin` est un fichier autorisé en lecture
  - `is_writable($chemin)` renvoie true ssi `$chemin` est un fichier autorisé en écriture
  - `basename($chemin)` renvoie le nom du fichier uniquement. Par exemple, pour `"images/a.png"`, renverra `"a.png"`

- `array file (string $filename[, int $flags = 0])`
  Permet de lire un fichier en entier et renvoie le fichier dans un tableau (ou `false` en cas d'erreur). `$filename` peut être un chemin comme dans les fonctions précédentes. Chaque ligne du fichier stockée dans une case du tableau. Options utiles : `FILE_IGNORE_NEW_LINES` ne met pas de retour à la ligne à la fin de chaque élément du tableau et `FILE_SKIP_EMPTY_LINES` saute les lignes vides.

FICHIERS

- `string file_get_contents (string $filename [,bool $use_include_path = false [,resource $context [,int $offset = -1 [,int $maxlen ]]]])`

Lit le fichier *_en entier_* dans une seule chaîne de caractères (valeur de retour, `false` si erreur). On n'utilisera pas `$use_include_path` ou `$context`, mais `$offset` et `$maxlen` peuvent être utiles : `$offset` est la position à laquelle on commence dans le fichier (n^{ième} caractère) et `$maxlen` est le nombre maximum de caractères à lire.

Du coup, comme il y a deux arguments que l'on n'utilise pas au milieu, on peut les ignorer en leur donnant leur valeur par défaut : `false` et `NULL`. Par exemple, on pourra faire :

```php
// Lire le fichier "fichiers_texte/f1.txt" à partir du caractère 500
file_get_contents("fichiers_texte/f1.txt", false, NULL, 500);
```

Même chose pour toute autre fonction qui a des arguments que vous voulez ignorer !

FICHIERS

- `int file_put_contents(string $filename, mixed $data[, int $flags = 0[, resource $context]])`

L'inverse de `file_get_contents()` : permet d'écrire la chaîne `$data` dans le fichier `$filename`. En fait `$data` peut aussi être un tableau, auquel cas, toutes les cases sont écrites les unes après les autres, sans séparateur.

Si le fichier n'existe pas, il est créé et rempli. Si le fichier existait déjà, son contenu est remplacé, sauf si l'option (`$flag`) `FILE_APPEND` est mise. Dans ce cas, les données sont écrites à la fin du fichier.

Encore une fois il y a plusieurs options, qu'on peut combiner avec l'opérateur `|`. Même si on ne s'intéresse qu'à `FILE_APPEND` ici.

FICHIERS

- **Exercice :** enregistrer la date et l'adresse IP de chaque accès à votre page web dans un fichier `"log.txt"`. Faire également une page qui affiche tous les accès…

…sachant que `$_SERVER` est un tableau qui contient des informations sur l'accès à la page courante : sa cellule `REMOTE_ADDR` contient l'adresse IP du client, et sa cellule `REQUEST_TIME` contient le timestamp UNIX du moment où la requête HTTP a été effectuée.

Pour enregistrer les accès, il suffit d'ajouter dans notre page web :

```php
<?php
    file_put_contents("log.txt", $_SERVER["REMOTE_ADDR"] . "#"
                        . $_SERVER["REQUEST_TIME"] . "\n", FILE_APPEND);
/*
   Par exemple, le fichier produit pourra ressembler à :
   214.198.46.53#1448051626
   57.88.87.138#1448051687
   131.144.224.94#1448051728
*/
?>
```

FICHIERS

- **Exercice :** enregistrer la date et l'adresse IP de chaque accès à votre page web dans un fichier `"log.txt"`. Faire également une page qui affiche tous les accès…

Puis, pour afficher les accès, on pourra faire :

```php
<?php
    $log = file("log.txt");

    if ($log !== false) {
        foreach ($log as $line) {
            $ip_time = explode("#", $line);
            echo "<div class=\"log_line\">\n";
            echo "<span class=\"ip_address\">{$ip_time[0]}</span>\n";
            echo "<span class=\"ip_address\">"
                . date("d/m/Y", $ip_time[1]) . "</span>\n";
            echo "</div>\n";
        }
    }
    else {
        echo "<p>Erreur : impossible d'ouvrir le fichier d'historique.</p>\n";
    }
?>
```

FICHIERS

- **Exercice :** enregistrer la date et l'adresse IP de chaque accès à votre page web dans un fichier `"log.txt"`. Faire également une page qui affiche tous les accès…

Problème de cette implémentation : le fichier est chargé entièrement en mémoire à chaque fois dans un tableau avec `file()` !

Problématique si c'est un gros fichier, et qu'on a beaucoup de requêtes en même temps : par exemple, **si c'est un fichier de 100 Mo et qu'on a 200 requêtes en même temps, 20 Go de mémoire absorbée d'un coup !** Par conséquent, les fonctions qu'on a vues sont simples d'utilisation, mais pas recommandées…

Il est recommandé de lire le fichier ligne à ligne !

FICHIERS

- `resource fopen(string $filename, string $mode)`

  Ouvre le fichier `$filename` dans le mode `$mode`, et renvoie une « ressource » qui représente le fichier, ou `false` s'il y avait une erreur. Les modes d'ouverture sont les suivants :
  - **`"r"` ouvre le fichier en lecture seule, place le pointeur au début du fichier**
  - `"r+"` ouvre le fichier en lecture et écriture, place le pointeur au début du fichier
  - `"w"` ouvre le fichier en écriture seule et l'efface s'il existe déjà !
  - `"w+"` ouvre le fichier en lecture et écriture et l'efface s'il existe déjà.
  - **`"a"` ouvre le fichier en écriture seule, le crée s'il n'existe pas, place le pointeur à la fin : utile pour écrire à la fin du fichier ! Et si on veut pouvoir lire aussi, option suivante :**
  - `"a+"` ouvre le fichier en lecture et écriture, le crée s'il n'existe pas, et place le pointeur à la fin
  - D'autres modes existent… Qui font une erreur si le fichier existe déjà par exemple.

- `bool fclose(resource $handle)`

  Ferme un fichier créé par `fopen()`. On doit lui passer la valeur de retour de `fopen()`. Renvoie `true` si le fichier a bien été fermé, et `false` s'il y a eu une erreur (rare). Entre l'ouverture et la fermeture on peut faire des lectures et des écritures ligne à ligne ou même caractère par caractère…

FICHIERS

- `string fgets(resource $handle[, int $length ])`

  Lit une ligne dans le fichier représenté par `$handle`. On peut spécifier une longueur de ligne maximum, auquel cas on ne lira que les `$length` premiers caractères de la ligne si elle est plus longue que cela. La fonction renvoie la ligne lue, ou renvoie `false` si la fin du fichier a été atteinte.

- `int fwrite(resource $handle, string $string[, int $length])`

  Ecrit `$string` dans le fichier représenté par `$handle`. Si on spécifie `$length`, s'arrête soit à la fin de la chaîne, soit à la longueur `$length` (le premier qui arrive). Renvoie le nombre de caractères écrits, ou false si une erreur s'est produite.

- **De nombreuses fonctions de lecture et d'écriture, permettant de faire des choses plus ou moins complexes, voir http://php.net/manual/fr/refs.fileprocess.file.php !**

FICHIERS

- `int fseek(resource $handle, int $offset[, int $whence = SEEK_SET])`

Permet de se déplacer dans un fichier représenté par `$handle`. On peut utiliser `$offset` pour se déplacer où l'on veut dans le fichier. La façon dont `$offset` est interprétée dépend du paramètre `$whence` :

- Si `$whence == SEEK_SET` (par défaut), place le pointeur du fichier en position `$offset` : par exemple, si `$offset == 0`, on se remet au début de fichier. S'il vaut `10`, on se place au 10ème caractère. Donc `$offset` doit être positif.
- Si `$whence == SEEK_CUR`, place le pointeur à la position courante + `$offset`. Par exemple, si `$offset == 10`, on avance de 10 caractères. Si `$offset == -10`, on recule de 10 caractères.
- Si `$whence == SEEK_END`, place le pointeur à la position finale + `$offset`. Par exemple, si `$offset == -10`, on se place 10 caractères avant la fin. Donc `$offset` doit être négatif.

- Si vous savez ouvrir un ficher (`fopen()`), le lire (`fgets()`), y écrire (`fwrite()`), vous déplacer dedans(`fseek()`), et le fermer (`fclose()`), vous pouvez à peu près tout faire...

FICHIERS

- **Exercice :** enregistrer la date et l'adresse IP de chaque accès à votre page web dans un fichier `"log.txt"`. Faire également une page qui affiche tous les accès...

Meilleure implémentation :

```php
<?php
    $handle = fopen("log.txt", "r");

    if ($handle !== false) {
        while (($line = fgets($handle)) !== false) {
            $ip_time = explode("#", $line);
            echo "<div class=\"log_line\">\n";
            echo "<span class=\"ip_address\">{$ip_time[0]}</span>\n";
            echo "<span class=\"ip_address\">"
                . date("d/m/Y", $ip_time[1]) . "</span>\n";
            echo "</div>\n";
        }

        fclose($handle);
    }
    else {
        echo "<p>Erreur : impossible d'ouvrir le fichier d'historique.</p>\n";
    }

?>
```

EXEMPLE : LIVRE D'OR

Nom :
John Smith
E-mail :
john.smith@gmail.com

Page web :
http://foobar.com

Message:
Site génial !

Réinitialiser Envoyer

Nom : Jean Dupont
E-mail : jean.dupont@unice.fr
Page web : http://www-mips.unice.fr/~jdupont
Message : J'adore ce site !

Nom : Claire Martin
E-mail : cmartin@unice.fr
Page web : http://www-mips.unice.fr/~cmartin
Message : Super site, je reviendrai !

`index.php` : formulaire + messages déjà enregistrés

Merci, John Smith, d'avoir signé mon livre d'or !

Revenir au livre d'or

`ajout.php` : page chargée après soumission du formulaire : stocke le message dans le fichier `entrees.txt` et affiche un message. Lien pour revenir sur `index.php`.

Nom :

E-mail :

Page web :
http://

Message:

Réinitialiser Envoyer

Nom : Jean Dupont
E-mail : jean.dupont@unice.fr
Page web : http://www-mips.unice.fr/~jdupont
Message : J'adore ce site !

Nom : Claire Martin
E-mail : cmartin@unice.fr
Page web : http://www-mips.unice.fr/~cmartin
Message : Super site, je reviendrai !

Nom : John Smith
E-mail : john.smith@gmail.com
Page web : http://foobar.com
Message : Site génial !

`index.php` lorsqu'on y revient

EXEMPLE : LIVRE D'OR

```
<div class="message">
<div><span class="field_title">Nom :</span> Jean Dupont</div>
<div><span class="field_title">E-mail :</span> jean.dupont@unice.fr</div>
<div><span class="field_title">Page web :</span> http://www-mips.unice.fr/~jdupont</div>
<div><span class="field_title">Message :</span> J'adore ce site !</div></div>
<div class="message"><div><span class="field_title">Nom :</span> Claire Martin</div>

<div><span class="field_title">Nom :</span> Claire Martin</div> ...
```

`entrees.txt` : les entrées du livre d'or sont stockées en HTML (incomplet), à chaque soumission du formulaire, on en ajoute une nouvelle, formatée avec `divs`, `spans`, etc...

EXEMPLE : LIVRE D'OR – INDEX.PHP


```
<!doctype html>
<html>

<head>
<meta charset="utf-8" />
<title>Mon livre d'or</title>
<style>
.field_title { font-weight : bold; }
.message {
    margin-top : 10px;
    border : 1px dashed black;
    width: 320px;
    background : #EEEEEE
}
</style>
</head>

<body>
```

index.php(1)

EXEMPLE : LIVRE D'OR – INDEX.PHP


```
<form method="post"  action="ajout.php">

Nom  : <br />
<input name="name"  type="text"  size="50"  maxlength="40"  />
<br />
E-mail  : <br />
<input name="email"  type="text"  size="50"  maxlength="40"  />
<br /><br />
Page  web  : <br />
<input name="site"  type="text"  size="50"
       value="http://"  maxlength="40"  />
<br /><br />
Message:<br />
<textarea name="msg"  cols="50"  rows="6"></textarea>
<br /><br />

<input type="reset"  value="Réinitialiser"  />
<input type="submit"  value="Envoyer"  />

</form>
```

index.php (2)

EXEMPLE : LIVRE D'OR – INDEX.PHP


```php
<?php
    $file_contents = file_get_contents("entrees.txt");

    if ($file_contents !== false)
        echo $file_contents;
    else
        echo "<strong>Une erreur est survenue.</strong>\n";
?>

</body>
</html>
```

index.php (3)

Merci, John Smith, d'avoir signé mon livre d'or !

Revenir au livre d'or

EXEMPLE : LIVRE D'OR – INDEX.PHP

```
<!doctype html>

<html>

<head>
<meta charset="utf-8" />
<title>Mon livre d'or</title>
</head>

<body>

<?php

    // On remplace les caractères spéciaux par
    // leurs entités HTML. Par exemple, il est
    // nécessaire de remplacer < par &lt; sinon
    // le navigateur croira que c'est le début
    // d'une balise et pas juste le signe < dans
    // un message...

    $name = htmlspecialchars($_POST['name']);
    $email = htmlspecialchars($_POST['email']);
    $site = htmlspecialchars($_POST['site']);
    $msg = htmlspecialchars($_POST['msg']);

    if (empty($email) || empty($name) || empty($msg)) {
        echo "<strong>Désolé, tous les champs sont "
                . "requis !</strong>";
    }
    else {

        file_put_contents("entrees.txt",
            "<div class=\"message\">" .
            "<div><span class=\"field_title\">" .
            "Nom :</span> $name</div>" .
            "<div><span class=\"field_title\">" .
            "E-mail :</span> $email</div>" .
            "<div><span class=\"field_title\">" .
            "Page web :</span> $site</div>" .
            "<div><span class=\"field_title\">" .
            "Message :</span> $msg</div>" .
            "</div>\n", FILE_APPEND);

        echo "<p>Merci, $name, d'avoir signé mon livre d'or !</p>";
    }
?>

<a href="index.php">Revenir au livre d'or</a>

</body>
</html>
```

BASES DE DONNÉES

- **Ecrire des données dans des fichiers : peu pratique.** Par exemple, si je veux récupérer tous les messages du livre d'or sans les autres champs, comment faire ? *Très compliqué, il faut lire le fichier, enlever toutes les balises, trouver ce qui est un message et ce qui est autre chose…*
- **Avec les bases de données,** on crée des tables qui contiennent des données, puis un langage standardisé, **SQL pour Standard Query Language** utilisé pour tout : création et suppression de tables, ajout et retrait de données…
- Ensuite, il faut avoir un SGBD (système de gestion de bases de données) installé sur un serveur. Plusieurs implémentations : MySQL et PostgreSQL sont libres, mais aussi possible d'utiliser Oracle, par exemple…
- Il faut s'assurer d'avoir un compte utilisateur sur le SGBD, pour pouvoir s'y connecter. Puis, on peut créer une base de données, avec par exemple, en SQL :

```
CREATE DATABASE IntroWebDB;
```

BASES DE DONNÉES

- Pour supprimer une base de données :

```
DROP DATABASE IntroWebDB;
```

- Une fois qu'on a une base de données, on peut créer des tables dedans. Ce sont en fait des tableaux de données, bidimensionnels. Chaque cellule a un type. Types courants :
  - `VARCHAR(n)` : une chaîne de caractères de taille `n` (pour des champs de texte de taille réduite, conviendrait pour le nom, l'e-mail, ou le site web dans un livre d'or, par exemple).
  - `TEXT` : pour une chaîne de caractères longue (par exemple le commentaire dans un livre d'or).
  - `INT, UNSIGNED INT` : un entier quelconque, ou un entier non-signé (positif).
  - `DECIMAL(a, b)` : un nombre décimal, avec `a` chiffres avant la virgule et `b` chiffres après.
  - Nombreux autres types, pour MySQL, voir : par exemple: http://dev.mysql.com/doc/refman/5.7/en/data-types.html

- Par exemple, pour créer une table qui contiendra des adresses IP et dates (pour notre historique, comme nous l'avons fait précédemment), stockées sous forme de chaîne de caractère et entier :

```
CREATE TABLE Log ( IPAddress VARCHAR (15),
                   Timestamp UNSIGNED INT );
```

BASES DE DONNÉES

- On peut ensuite supprimer la table avec :

```
DROP TABLE Log;
```

- Pour ajouter des éléments dans la table, on pourra faire :

```
INSERT INTO Log VALUES ('214.198.46.53', '1448051626');
INSERT INTO Log VALUES ('57.88.87.138', '1448051687');
```

- Enfin, pour récupérer toutes les valeurs de la table :

```
SELECT * FROM Log;
```

- Même chose, avec les résultats triés sur le champ `Timestamp`, dans l'ordre descendant (`ASC` sinon) :

```
SELECT * FROM Log ORDER BY Timestamp DESC; -- Trié sur les timestamps
```

- On peut récupérer seulement les adresses IP, ou seulement les timestamps :

```
SELECT IPAddress FROM Log;
SELECT Timestamp FROM Log;
```

- On peut aussi ajouter des conditions sur les lignes qu'on sélectionne :

```
SELECT Timestamp FROM Log WHERE IPAddress = '214.198.46.53';
SELECT IPAddress FROM Log WHERE Timestamp >= 1448051687;
```

BASES DE DONNÉES

Comment utiliser une base de données MySQL avec PHP ?

- `resource mysql_connect (string $server, string $username, string $password)`
  Permet de se connecter à une base de données, et renvoie un objet qui modélise la connexion, que l'on peut utiliser si on a plusieurs connexion. Si on n'a qu'une seule connexion, on peut l'ignorer : toutes les fonctions suivantes utiliseront notre unique connexion.

- `bool mysql_select_db(string $database_name[, resource $link_identifier = NULL])`
  Permet de sélectionner la base de données sur laquelle on travaille. En deuxième paramètre, on peut spécifier la connexion si nécessaire.

- `mixed mysql_query ( string $query [, resource $link_identifier = NULL ] )`
  Permet d'exécuter une requête SQL, avec éventuellement une connexion. Le résultat dépend de la requête. S'il s'agit d'une insertion (`INSERT INTO ...`) renverra `true` si pas d'erreur ou `false` en cas d'erreur. S'il s'agit d'une lecture (`SELECT ... FROM ...`) renverra un objet qui permet de récupérer toutes les lignes sélectionnées...

BASES DE DONNÉES

- `array mysql_fetch_array(resource $result)`
  Si on a exécuté une requête `SELECT` avec `mysql_query()`, on peut appeller `mysql_fetch_array()` sur le résultat de `mysql_query()` pour récupérer la ligne suivante des résultats sous la forme de tableau PHP. Renvoie `false` lorsqu'il n'y a plus de résultat.

- `bool mysql_close ([ resource $link_identifier = NULL ])`
  Ferme la connexion à la base. On peut spécifier la connexion en paramètre, pas nécessaire si on a une seule connexion. Renvoie `false` en cas d'erreur.

- Ces cinq fonctions permettent déjà de faire beaucoup de choses avec une base de données : *on se connecte au SGBD, on choisit la base de données, on exécute des requêtes et on récupère le résultat, et enfin on ferme la connexion…*

- Pour plus de fonctions : http://php.net/manual/fr/book.mysql.php

BASES DE DONNÉES

- **Exemple :** de nouveau, fichier d'historique avec IPs et timestamps

```php
<?php
    $host="localhost";
    $user="jplozi";
    $password="password";
    $database="IntroWebDB";

    mysql_connect($host,  $user,  $password);

    if (!mysql_select_db($database))
        exit("Erreur de connexion à la base...");

    $sql = "INSERT INTO Log VALUES ('{$_SERVER['REMOTE_ADDR']}',  "
         . "'{$_SERVER['REQUEST_TIME']}')";

    if (!mysql_query($sql))
        exit("Impossible d'insérer les données...");
}
?>
```

BASES DE DONNÉES

```php
<?php
    $host="localhost";
    $user="jplozi";
    $password="password";
    $database="IntroWebDB";

    mysql_connect($host, $user, $password);

    if (!mysql_select_db($database))
        exit("Erreur de connexion à la base...");

    $sql = "SELECT * FROM Log ORDER BY Timestamp DESC";

    $result = mysql_query($sql);

    while($rows = mysql_fetch_array($result)) {
        echo "<div class=\"log_line\">\n";
        echo "<span class=\"ip_address\">{$rows['IPAddress']}</span>\n";
        echo "<span class=\"ip_address\">"
            . date("d/m/Y", $rows['Timestamp']) . "</span>\n";
        echo "</div>\n";
    }

    mysql_close();
?>
```

POUR ALLER PLUS LOIN…

- En huit séances, on a touché à beaucoup de choses : HTML, CSS, JavaScript, PHP et SQL…
- **Impossible de tout faire, idée : vous donner les bases pour chacune de ces technologies.**
- Lorsque vous programmerez, vous aurez sûrement besoin de choses que l'on n'a pas vu dans le cours… **Lorsque cela arrive : Google !**
- C'est en programmant de petits sites web, et en cherchant comment faire ce que l'on ne sait pas faire sur internet lorsqu'on en a besoin qu'on apprend le mieux…