“Web Dynamique”

- Génération automatique des pages par le serveur :
  - Le contenu dépend du visiteur
  - Parfois, on trouve un **système d’authentification** (ex : ENT)
  - Langages : PHP (Hypertext Preprocessor), JSP etc.
  - Utilisation d’une base de données pour générer les pages

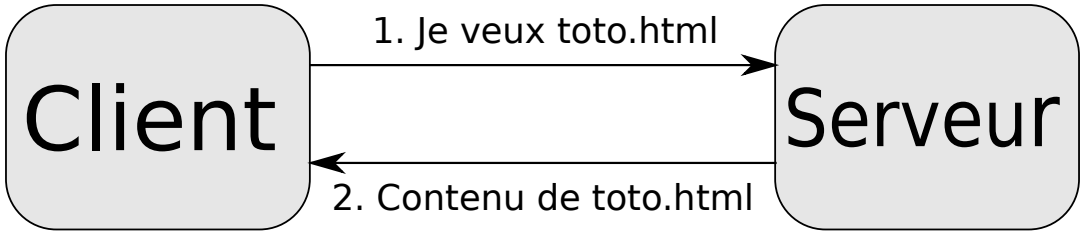
- Pages web dynamiques :
  - Exécution de scripts sur le client
  - Présentation et réorganisation dynamiques des données coté client
  - Langages : JavaScript, VBScript, etc.

“Web 2.0”

- Combinaison des deux aspects du Web dynamique
- Les scripts exécutés sur le client échangent des informations avec un serveur (AJAX, Flash, SilverLight)
- Mise à jour dynamique d’une partie de la page Web
- Permet de créer des **Applications Web Riches** (RIA) :
  - Gmail, Google Maps, Flickr, Deezer, etc.
- Permet d’organiser des **réseaux sociaux** :
  - Facebook, Myspace, etc.
- Permet de créer des **Wiki, blogs et travaux collaboratifs** :
  - Wikipedia, etc.

Le Web “statique”

Protocole HTTP

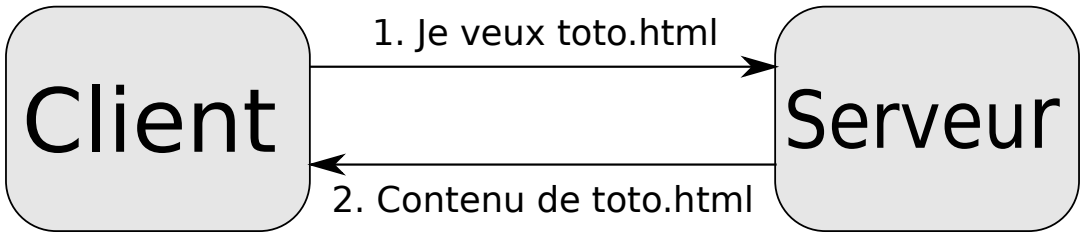


Requête :
GET /toto.html HTTP/1.0
Host : example.com
Referer : http ://example2.com/
User-Agent : Mozilla/5.0 (X11 ; U ; Linux x86_64 ; fr ; rv :1.9.0.4) Gecko/2008111217 Fedora/3.0.4-1.fc10 Firefox/3.0.4

Réponse :
HTTP/1.0 200 OK
Date : Fri, 31 Dec 1999 23 :59 :59 GMT
Server : Apache/0.8.4
Content-Type : text/html
Content-Length : 59
Expires : Sat, 01 Jan 2000 00 :59 :59 GMT
Last-modified : Fri, 09 Aug 1996 14 :21 :40 GMT
<TITLE>Exemple</TITLE>
<P>page d’exemple.</P>

Le Web “dynamique”

Protocole HTTP

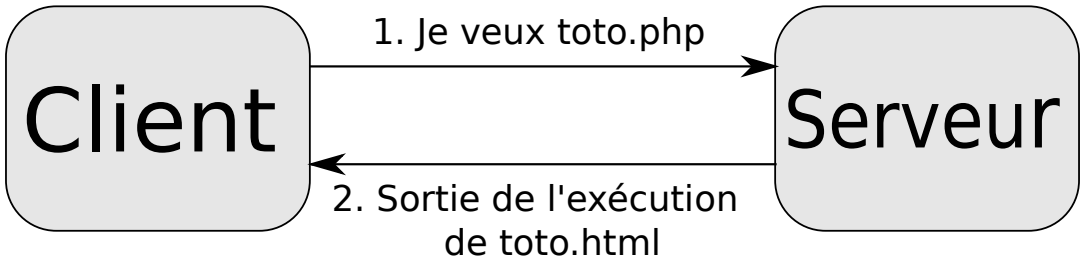


Requête :
GET /toto.html HTTP/1.0
Host : example.com
Referer : http ://example2.com/
User-Agent : Mozilla/5.0 (X11 ; U ; Linux x86_64 ; fr ; rv :1.9.0.4) Gecko/2008111217 Fedora/3.0.4-1.fc10 Firefox/3.0.4

Réponse :
HTTP/1.0 200 OK
Date : Fri, 31 Dec 1999 23 :59 :59 GMT
Server : Apache/0.8.4
Content-Type : text/html
Content-Length : 59
Expires : Sat, 01 Jan 2000 00 :59 :59 GMT
Last-modified : Fri, 09 Aug 1996 14 :21 :40 GMT
<TITLE>Exemple</TITLE>
<P>page d’exemple.</P>

Le Web “dynamique”

Protocole HTTP

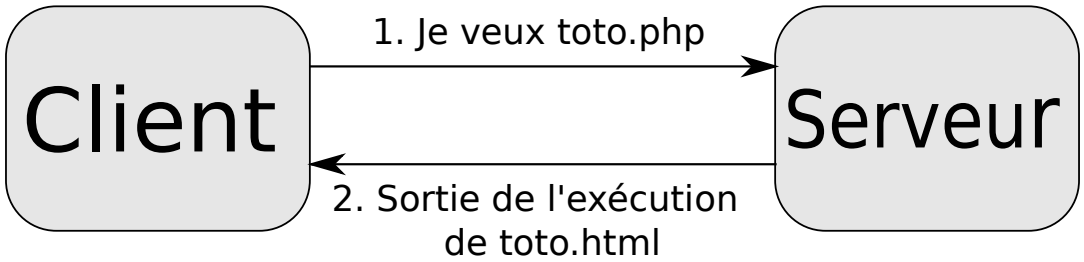


Requête :
GET /toto.php ?n1=10&n2=15 HTTP/1.0
Host : example.com
Referer : http ://example2.com/
User-Agent : Mozilla/5.0 (X11 ; U ; Linux x86_64 ; fr ; rv :1.9.0.4) Gecko/2008111217 Fedora/3.0.4-1.fc10 Firefox/3.0.4

Réponse :
HTTP/1.0 200 OK
Date : Fri, 31 Dec 1999 23 :59 :59 GMT
Server : Apache/0.8.4
Content-Type : text/html
Content-Length : 59
Expires : Sat, 01 Jan 2000 00 :59 :59 GMT
Last-modified : Fri, 09 Aug 1996 14 :21 :40 GMT
<TITLE>Exemple</TITLE>
<P>Résultat : 25.</P>

Le Web “dynamique”

Protocole HTTP



Requête :
POST /toto.php HTTP/1.0
Host : example.com
Referer : http ://example2.com/
User-Agent : Mozilla/5.0 (X11 ; U ; Linux x86_64 ; fr ; rv :1.9.0.4) Gecko/2008111217 Fedora/3.0.4-1.fc10 Firefox/3.0.4

n1=10&n2=15

Réponse :
HTTP/1.0 200 OK
Date : Fri, 31 Dec 1999 23 :59 :59 GMT
Server : Apache/0.8.4
Content-Type : text/html
Content-Length : 59
Expires : Sat, 01 Jan 2000 00 :59 :59 GMT
Last-modified : Fri, 09 Aug 1996 14 :21 :40 GMT
<TITLE>Exemple</TITLE>
<P>Résultat : 25.</P>

Pourquoi PHP ?

- Besoin d'un langage simple :
  - pour générer du HTML
  - pour communiquer avec une base de données
  - Langages : PHP (Hypertext Preprocessor), JSP etc.
  - pour gérer les sessions des utilisateurs

- Une solution :
  - **1994** : Invention de PHP par Rasmus Lerdorf
  - Il est interprété (PHP 3) ou compilé (PHP 4 et 5)
  - Il a une syntaxe proche du C (et de Perl)
  - **Open-source** et **multi-plateforme**

Premier programme

- Un premier programme “index.php” :

```
<html>
 <head>
  <title>Ma page PHP</title>
 </head>
 <body>
  <?
    echo "Bonjour,";
    echo "On est le ".date('d/M/Y');
  ?>
 </body>
</html>
```

- La balise <? permet d'entrer dans du code PHP
- La balise ?> permet de sortir du code PHP

Inclusion de fichiers

index.php :

```
<html>
 <head>
  <title>Titre</title>
 </head>
 <body>
  <?
   require("tete.inc.php");
   include("corps.html");
   require("pied.inc.php");
   ?>
 </body>
</html>
```

Mais aussi :

include_once et **require_once**

tete.inc.php :

```
<?
 echo "Bienvenue<br>";
?>
```

corps.html :

```
Corps du site<br>
```

pied.inc.php :

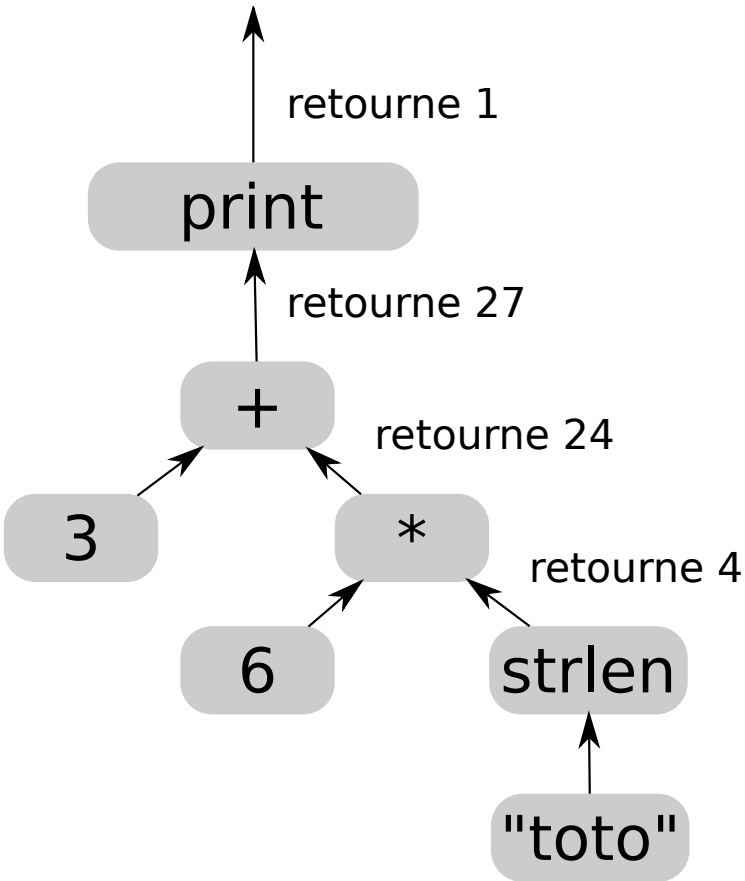
```
<?
 echo date('d/M/Y');
?>
```

Commentaires

```
<html>
 <head>
  <title>Titre</title>
 </head>
 <body>
  <?
    echo "Bonjour"; // commentaire
    echo "Salut"; /* commentaire
     sur plusieurs lignes. */
    echo "Coucou"; # commentaire
  ?>
  <!-- commentaire -->
 </body>
</html>
```

Instructions, opérations et fonctions

```
<?
  print(3+6*strlen("toto"));
?>
```



Variables

- En C ou en Java, à une variable sont associés :
  - Un nom (ou identifiant) ;
  - Un type ;
  - Une zone mémoire (désignée par une adresse).

```
int a;
a = 2;
```

- En PHP, à une variable sont associés :
  - Un nom (ou identifiant) commençant par $;
  - Un conteneur d'une valeur.

```
<?
  $a = 2;
?>
```

Les types des valeurs

- Les variables ne sont pas typées mais les valeurs ont un type :
  - **integer :** `7, 14, 255, 0xFF`
  - **boolean :** `TRUE, FALSE`
  - **double :** `1.95, 1.12e4`
  - **string :** `"bonjour", 'bonjour'`
  - **array :** `array(1,2,3)`
  - **object :** `new maclasse`
  - **ressource :** `mysql_connect("localhost", "moi", "")`
  - **null :** `null, NULL`

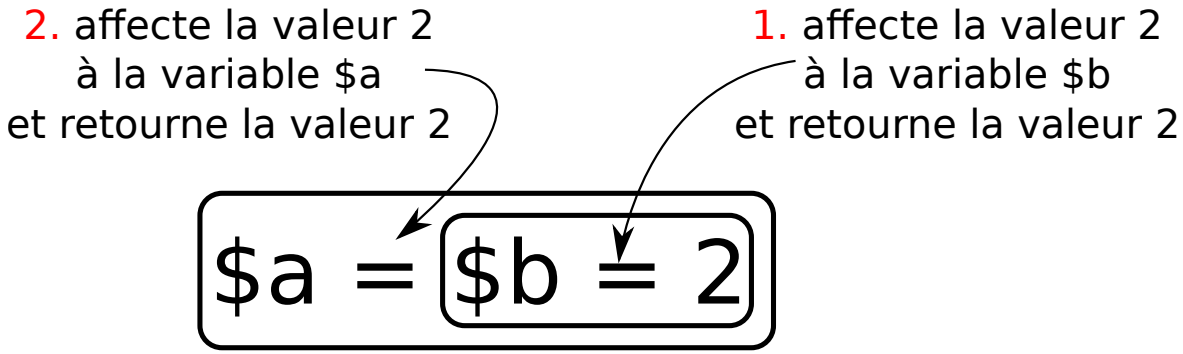
```
<?
  $a = 2;
  var_dump($a); // affiche int(2)
?>
```

Opérateur d'assignation

- En PHP, on ne déclare pas les variables
- L'opérateur = affecte la valeur d'une expression à une variable :

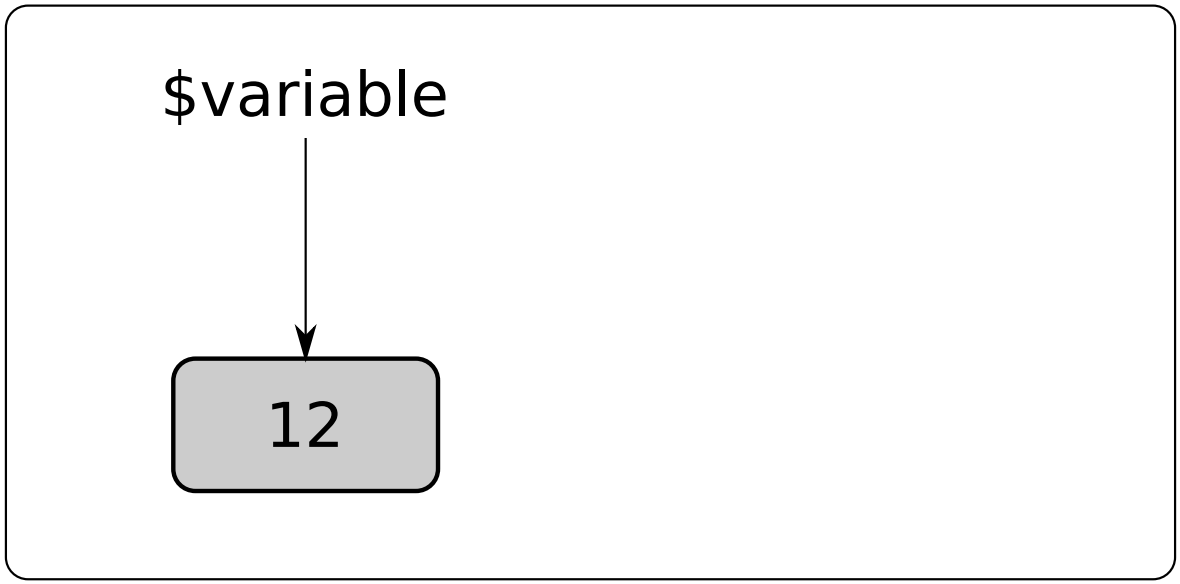
```
<?
  $a = expression;
?>
```

- L'opérateur = retourne la valeur de l'expression assignée à la variable



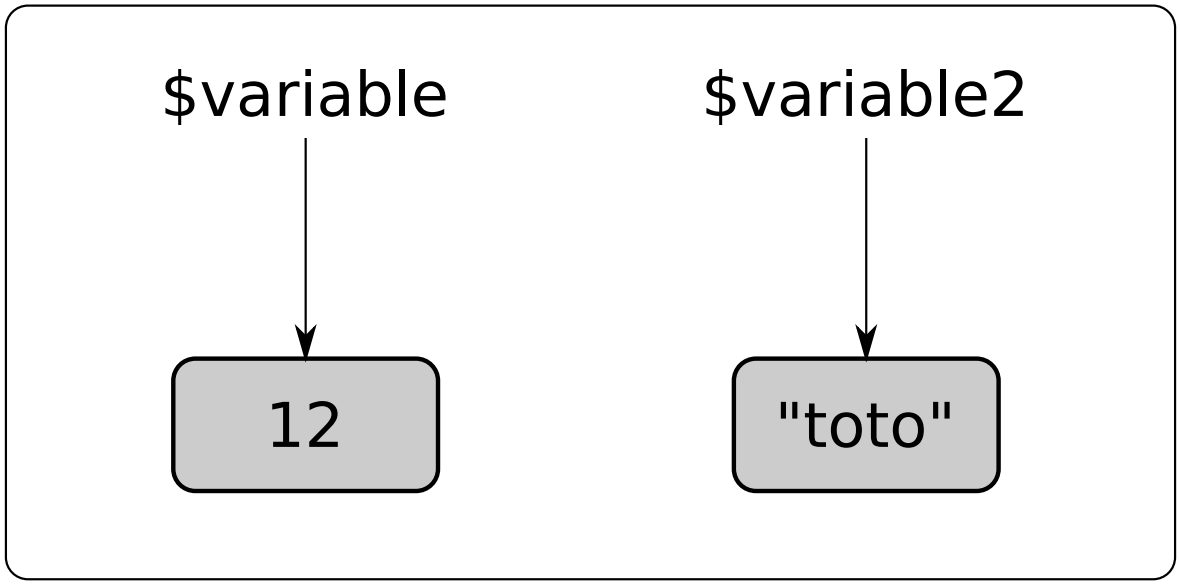
Affectations de valeur

```php
<?
$variable = 12;
$variable2 = "toto";
$variable2 = $variable;
$variable = 12.12+3;
?>
```



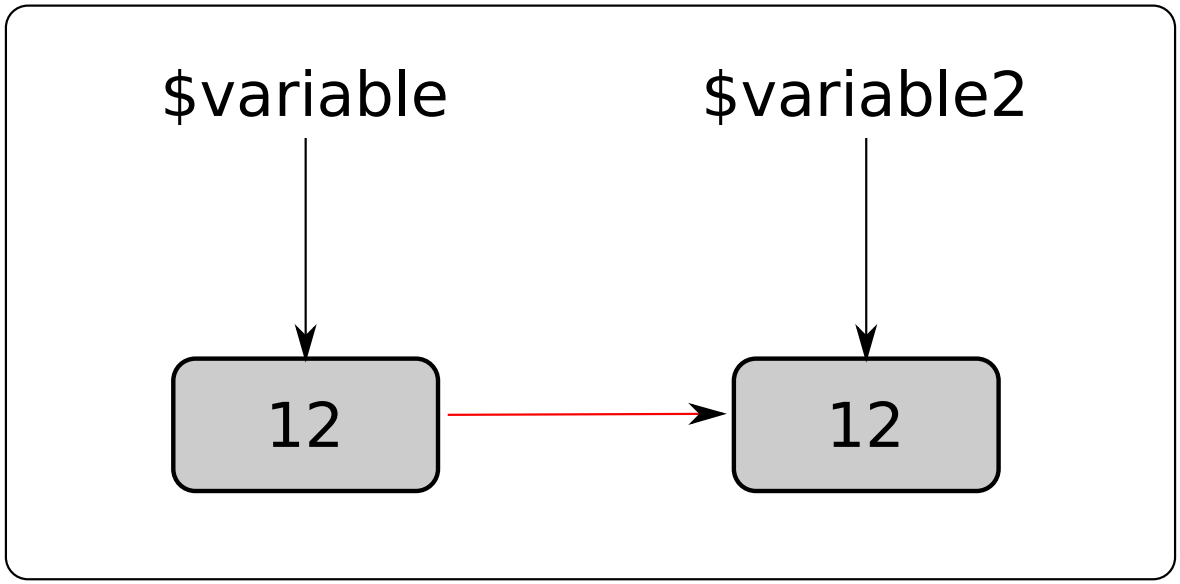
Affectations de valeur

```php
<?
$variable = 12;
$variable2 = "toto";
$variable2 = $variable;
$variable = 12.12+3;
?>
```



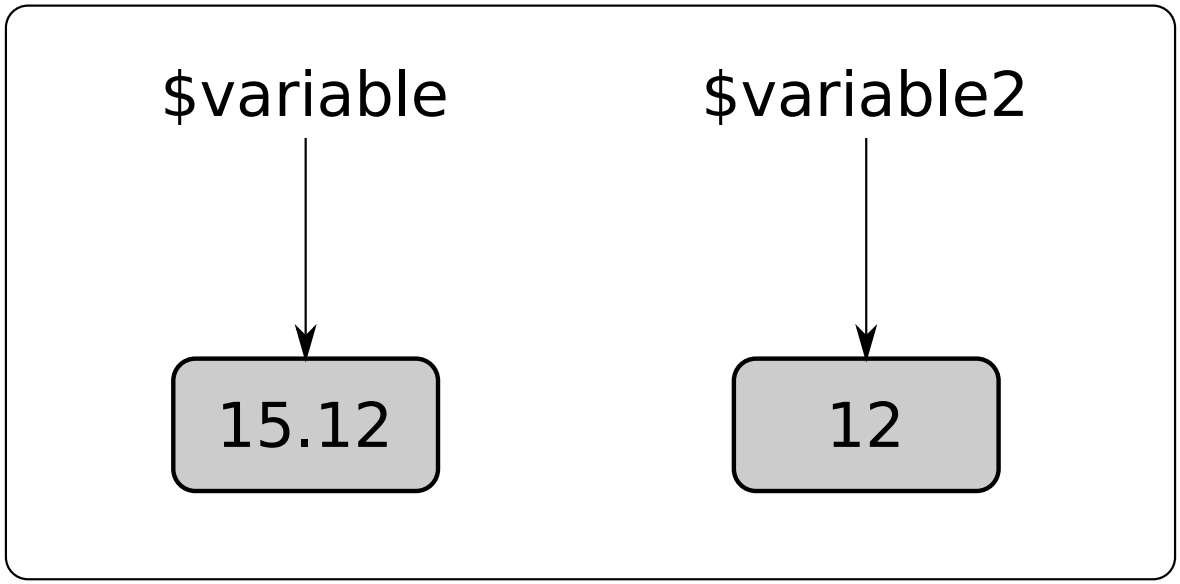
Affectations de valeur

```php
<?
$variable = 12;
$variable2 = "toto";
$variable2 = $variable;
$variable = 12.12+3;
?>
```



Affectations de valeur

```php
<?
$variable = 12;
$variable2 = "toto";
$variable2 = $variable;
$variable = 12.12+3;
?>
```



Opérateur d'assignation de référence

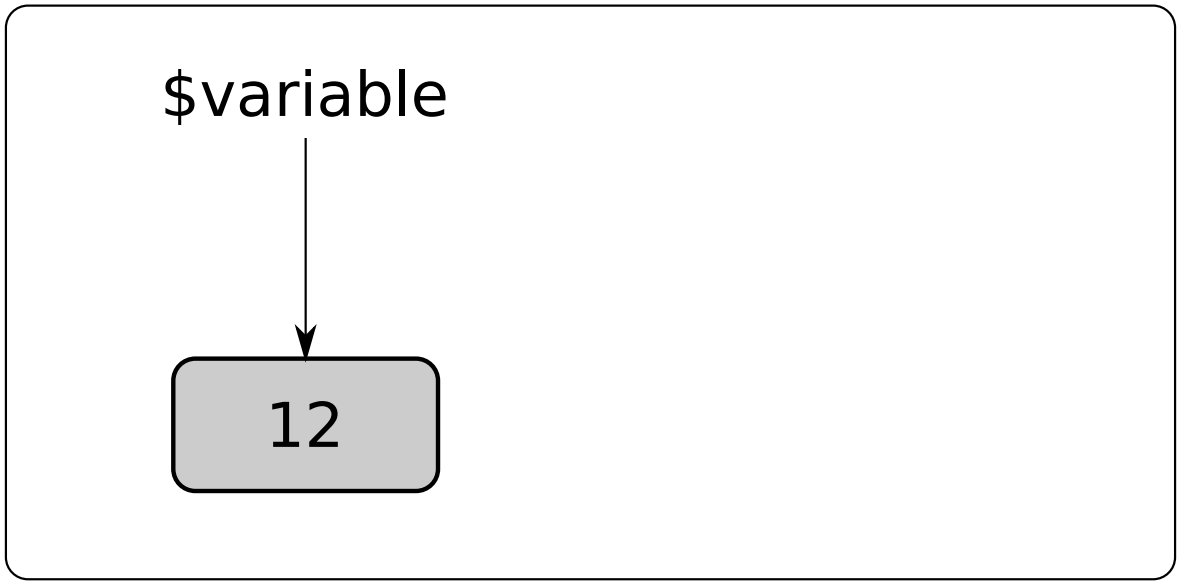
- Affectation de référence : l'opérande de droite est une variable précédée du caractère '&' :

```
$var2 = &$var1;
```

- Ici, l'opérateur = retourne la valeur présente dans le conteneur de la variable `$var1`.
- Après l'affectation, la variable `$var2` ne fait que référencer le conteneur associé à la variable `$var1`.

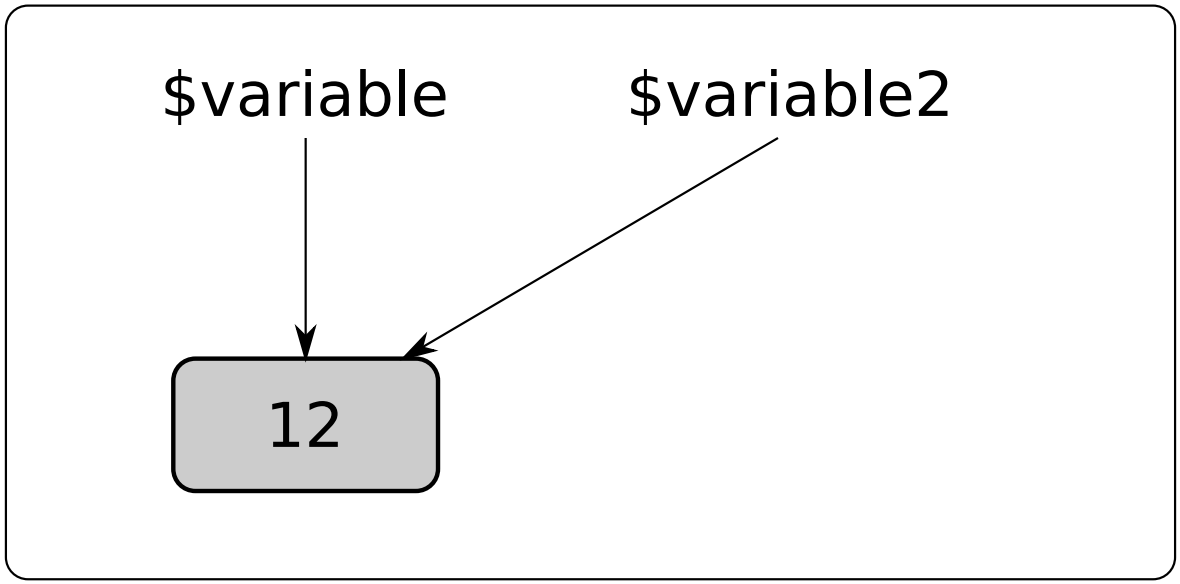
Affectations de référence

```php
<?
$variable = 12;
$variable2 = &$variable;
$variable2 = "toto";
$variable = 12.12;
?>
```



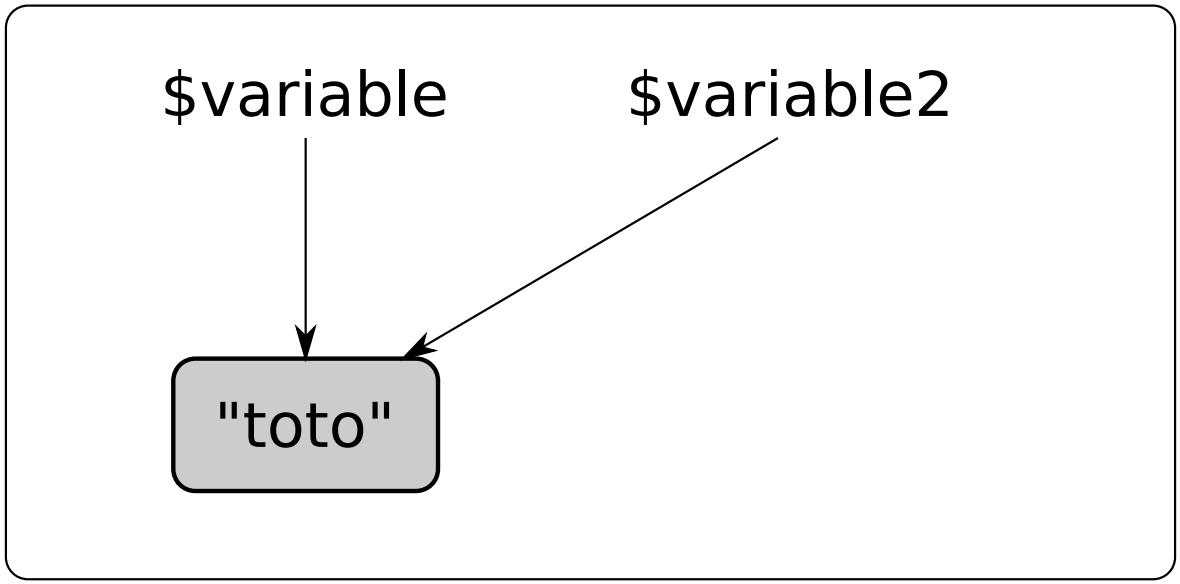
Affectations de référence

```php
<?
$variable = 12;
$variable2 = &$variable;
$variable2 = "toto";
$variable = 12.12;
?>
```



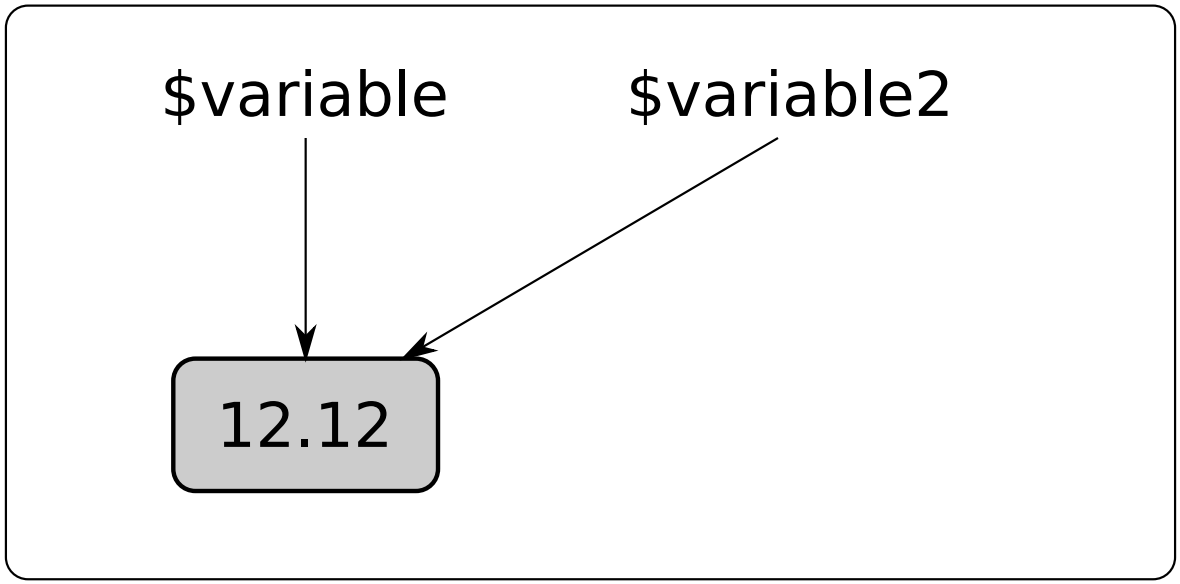
Affectations de référence

```php
<?
$variable = 12;
$variable2 = &$variable;
$variable2 = "toto";
$variable = 12.12;
?>
```



Affectations de référence

```php
<?
$variable = 12;
$variable2 = &$variable;
$variable2 = "toto";
$variable = 12.12;
?>
```



État d'une variable

- La fonction `isset($var)` retourne :
  - `FALSE` si la variable n'est pas initialisée ou a la valeur `NULL` ;
  - `TRUE` sinon.
- La fonction `empty($var)` retourne :
  - `TRUE` si une des conditions suivantes est vérifiée :
    - la variable n'est pas initialisée
    - la variable a la valeur `""` (chaîne vide)
    - la variable a la valeur `0` (entier)
    - la variable a la valeur `0.0` (flottant)
    - la variable a la valeur `"0"`
    - la variable a la valeur `NULL`
    - la variable a la valeur `FALSE`
    - la variable a la valeur `array()` (tableau vide)
  - `FALSE` sinon.
- La fonction `unset($var)` détruit une variable.

Type d'une variable

- Pour connaître le type de la valeur contenue dans le conteneur d'une variable `$var` :
  - `gettype($var)` retourne une chaîne de caractères contenant le type de la valeur (ex : `"integer"`)
  - `is_integer($var)` ou `is_int($var)`, `is_double($var)`, `is_scalar($var)`, `is_string($var)`, `is_bool($var)` , `is_array($var)` , `is_object($var)`, `is_ressource($var)`, `is_numeric($var)`

```
<?
   $var = 12;
   if (is_integer($var)) {
     echo "je suis un entier";
   }
?>
```

Conversion de type

- Opérateur de Cast :
  - `$var2 = (nouveau_type)$var`
  - ou même `$var = (nouveau_type)$var`

```
<?
   $var = "4.34 litre";
   $var = (double)$var;
   echo $var; // affiche 4.34
   $var = (integer)$var;
   echo $var; // affiche 4
   $var = (boolean)$var;
   echo $var; // affiche 1
?>
```

- On peut aussi utiliser la fonction `settype($var, "nouveau_type")`

Les constantes

- Pour définir une constante :

  `define("MA_CONSTANTE", 12.76, TRUE);`

  ↪ si le dernier paramètre vaut `TRUE`, le nom est insensible à la casse

- Pour savoir si une constante existe :

  `defined("MA_CONSTANTE")`

  ↪ retourne `TRUE` si la constante existe, `FALSE` sinon

- Utilisation d'une constante :

```
<?
   define("TOTO", 12.45, TRUE);
   echo TOTO, "<br/>";
   echo ToTo, "<br/>";
   if (defined("TOTO"))  echo "ok";
?>
```

Opérateurs numériques

Négation	`-$a`	Opposé de $a
Addition	`$a + $b`	Somme de $a et $b
Soustraction	`$a - $b`	Différence de $a et $b
Multiplication	`$a * $b`	Produit de $a et $b
Division	`$a / $b`	Quotient de $a et $b
Modulo	`$a % $b`	Reste de $a divisé par $b

Pre-incrémente	`++$a`	Incrémente $a de 1, puis retourne $a
Post-incrémente	`$a++`	Retourne $a, puis incrémente $a de 1
Pré-décrémente	`--$a`	Décrémente $a de 1, puis retourne $a
Post-décrémente	`$a--`	Retourne $a, puis décrémente $a de 1

Opérateurs logiques

et	`$a and $b`	TRUE si $a et $b valent TRUE
ou	`$a or $b`	TRUE si $a ou $b valent TRUE
ou exclusif	`$a xor $b`	TRUE si $a ou $b est égal TRUE mais pas les deux en même temps
non	`!$a`	TRUE si $a n'est pas égal à TRUE
et	`$a && $b`	TRUE si $a et $b sont égaux TRUE
ou	`$a || $b`	TRUE si $a ou $b est égal TRUE

- Attention à la précédence des opérateurs :

```
<?
   $e = false || true; // ($e = (false || true))
   $e = false or true; // (($e = false) or true)
   $e = false && true; // ($e = (false && true))
   $e = false and true; // (($e = false) and true)
?>
```

Opérateurs de comparaison

égal	$a == $b	TRUE si $a est égal à $b
identique	$a === $b	TRUE si $a et $b sont égaux et ont le même type
différent	$a != $b	TRUE si $a est différent de $b
différent	!$a <> $b	TRUE si $a est différent de $b
non identique	$a !== $b	TRUE si $a et $b sont différents ou n'ont pas le même type
plus petit	$a < $b	TRUE si $a est strictement plus petit que $b
plus grand	$a > $b	TRUE si $a est strictement plus grand que $b
inférieur ou égal	$a <= $b	TRUE si $a est plus petit ou égal à $b
supérieur ou égal	$a >= $b	TRUE si $a est plus grand ou égal à $b

```
<?
    var_dump(0 == "a"); // bool(true)
    var_dump(0 === "a"); // bool(false)
?>
```

Opérateurs de chaînes

- L'opérateur . permet de concaténer deux chaînes de caractères (comme le + en Java) :

```
<?
var_dump("Bonj"."our");// string(7) "Bonjour"
var_dump(1 . 2); // string(2) "12"
var_dump(1.2); // float(1.2)
$a = "Bonj";
$a = $a . "our";
var_dump($a); // string(7) "Bonjour"
$a = "Bonj";
$a .= "our";
var_dump($a); // string(7) "Bonjour"
?>
```

Opérateurs de commande

- L'opérateur ‘ (guillements obliques) permet d'exécuter des commandes shell :

```
<?
   $output = ‘ls -al‘;
   echo "<pre>$output</pre>";
?>
```

- Remarque : cet opérateur n'est pas actif lorsque le “safemode” est activé ou lorsque la fonction `shell_exec()` est désactivée.

Opérateurs d'affectation combinée

addition	`$a += $b`	additionne $a et $b puis affecte le résultat à $a
soustraction	`$a -= $b`	soustrait $a et $b puis affecte le résultat à $a
multiplication	`$a *= $b`	multiplie $a et $b puis affecte le résultat à $a
division	`$a /= $b`	divise $a et $b puis affecte le résultat à $a
modulo	`$a %= $b`	divise $a et $b puis affecte le reste à $a
concaténation	`$a .= $b`	concatène $a et $b puis affecte le résultat à $a

```php
<?
   $a = 13;
   $a += 12;
   echo $a; // affiche 25
?>
```

Block d'instructions

- Comme en **C** ou en **Java**, on définit un block d'instructions à l'aide des accolades ouvrantes et fermantes { } :

```
<?
   if ($a == 2) {
      echo "instruction 1";
      echo "instruction 2";
   }
?>
```

if, boucles while et do...while

- On utilise le **if**, du **while** et le **do...while** de la même façon qu'en **C** ou qu'en **Java** :

```
<?
   $a = 1;
   if ($a == 2) echo "oui"; else echo "non";
   while ($a < 4) {
    echo $a;
    $a++;
   }
   do {
    echo $a;
    $a--;
   } while ($a>0);
?>
```

boucle for

- La syntaxe du **for** est la même qu'en **C** ou qu'en **Java** :

```
for (expression; expression; expression) instruction;
```

```
<?
   for ($a=0; $a<10; $a++) {
      echo $a.":";
      for ($b=0; $b<10; $b+=2)
      echo ($a+$b)."␣";
      echo "<br/>";
   }
?>
```

break et continue

- la commande **break** arrête l'exécution de la boucle :

```
<?
for ($i = 0; $i < 5; $i++) {
  if ($tab[$i]=="bonjour") break;
  echo $tab[$i];
}
?>
```

Truc
Toto
Bonjour
Bip
Salut

- la commande continue arrête l'itération en cours de la boucle :

```
<?
for ($i = 0; $i < 5; $i++) {
  if ($tab[$i]=="bonjour") continue;
  echo $tab[$i];
}
?>
```

Truc
Toto
Bonjour
Bip
Salut

switch

```php
<?
switch ($a) {
case 0 :
 echo '0';
 break;
case 1 :
 echo '1';
 break;
default :
 echo 'default';
}
?>
```

```php
<?
switch ($a) {
case "a" :
 echo 'a';
 break;
case "b" :
 echo 'b';
 break;
default :
 echo 'default';
}
?>
```

Fonctions

```
<?
  function ajouter(&$a (1) , $b=5 (2)) {
    $a+=$b;
  }

  $n = 12;
  ajouter($n, 2);
  var_dump($n); // affiche int(14)
  ajouter($n);
  var_dump($n); // affiche int(19)
?>
```

1. Passage d'un paramètre par référence.
2. La valeur par défaut du paramètre est 5.

Portée des variables

```php
<?
  function modif() {
    $var = "salut";
  }

  $var = "toto";
  var_dump($var); 1
  modif();
  var_dump($var); 2
?>
```

1 `string(4) "toto"`

2 `string(4) "toto"`

```php
<?
  function modif() {
    global $var;
    $var = "salut";
  }

  $var = "toto";
  var_dump($var); 3
  modif();
  var_dump($var); 4
?>
```

3 `string(4) "toto"`

4 `string(5) "salut"`

Affichage des chaînes

```php
<?
  $a = "bonjour";
  $b = "salut";
  $c = 2;

  echo "$a $b\n"; 1
  echo '$a $b\n'; 2
  echo "\n";
  echo "$a $b{$c}\n"; 3
  echo date('d')."\n"; 4
  echo "date('d')\n"; 5
?>
```

1. bonjour salut
2. $a $b\n
3. bonjour l
4. 18
5. date('d')

Les caractères

```
<?
  $chaine="ABCDEF";
  for ($i = 0; $i<strlen($chaine); $i++) {
    echo ord($chaine{$i}).”\n”; 1
  }

  $chaine="";
  for ($i = 0; $i<6; $i++) {
   $c = rand(65,90);
   $chaine.=chr($c);
  }
  echo "$chaine\n"; 2
?>
```

1
65
66
67
68
69
70

2 GZXNIY

Affichage formaté

```
<?
$chaine = "Bonjour";
$nombre = "65";
$valeur = "65535";
$flotant = "12.2345";
printf("%s\n", $chaine); 1
printf("%c %d\n", $nombre, $nombre); 2
printf("%x %o\n", $valeur, $valeur); 3
printf("%'#8.3f\n", $flotant); 4
$a = sprintf("%'#8.3f", $flotant);
var_dump($a); 5
$a = array("65", "66","67");
vprintf("%c %c %c\n", $a); 6
$b = vsprintf("%c %c %c", $a);
var_dump($b); 7
?>
```

1. Bonjour
2. A 65
3. ffff 177777
4. ##12.235

5. string(8) "##12.235"
6. A B C
7. string(5) "A B C"

Modification de la casse

```
<?
  $chaine = "PHP est super bien !\n";
  echo strtolower($chaine); 1
  echo strtoupper($chaine); 2
  echo ucwords($chaine); 3
  echo ucfirst($chaine); 4
?>
```

1 php est super bien !

2 PHP EST SUPER BIEN !

3 PHP Est Super Bien !

4 PHP est super bien !

Gestion des espaces

```
<?
  $a="␣␣␣...Salut./././.";
  echo "[".ltrim($a)."]\n"; 1
  echo "[".ltrim($a,"␣.")."]\n"; 2
  echo "[".rtrim($a,"./")."]\n"; 3
  echo "[".trim($a,"␣./")."]\n"; 4
?>
```

1. [...Salut././.]
2. [Salut././.]
3. [...Salut]
4. [Salut]

Caractères spéciaux dans les URL et en XHTML

```
<?
  $a="<b>détruire</b>";
  $b=htmlentities($a);
  echo $b."\n"; 1
  $c=html_entity_decode($b);
  echo $c."\n"; 2
  $b=strip_tags($a);
  echo $b."\n"; 3
  $b=urlencode($a);
  echo $b."\n"; 4
  $c=urldecode($b);
  echo $c."\n"; 5
?>
```

```
1 &lt;b&gt;d&eacute;truire&lt;b&gt;
2 <b>détruire</b>
3 détruire
4 %3Cb%3Ed%E9truire%3C%2Fb%3E
5 <b>détruire</b>
```

Recherche de sous-chaînes

```
<?
$ch = "bonjour salut bonjour";
$nb = substr_count($ch, "bonjour");
var_dump($nb); 1
$ch2 = str_replace("bonjour", "salut", $ch);
var_dump($ch2); 2
$pos = strpos($ch,"salut");
var_dump($pos); 3
$pos = strpos($ch,"Salut");
var_dump($pos); 4
$pos = stripos($ch,"Salut");
var_dump($pos); 5
$ch3 = substr($ch,8,5);
var_dump($ch3); 6
?>
```

```
1 int(2)
2 string(17) "salut salut salut"
3 int(8)
4 bool(false)
5 int(8)
6 string(5) "salut"
```

Comparaison de chaînes de caractères

```
<?
$ch1=11;
$ch2="11toto";
var_dump($ch1); 1
var_dump($ch2); 2
var_dump($ch1==$ch2); 3
var_dump($ch1===$ch2); 4
var_dump("$ch1"==$ch2); 5
var_dump($ch1*$ch2); 6
var_dump("$ch1"*$ch2); 7
?>
```

```
1 int(11)
2 string(6) "11toto"
3 bool(true)
4 bool(false)
5 bool(false)
6 int(121)
7 int(121)
```

Comparaison de chaînes de caractères

```
<?
var_dump(strcmp("toto2","toto2"));          → int(0)   1
var_dump(strcmp("toto12","toto2"));         → int(-1)  2
var_dump(strcmp("toto2","toto12"));         → int(1)   3
var_dump(strcasecmp("toto","ToTo"));        → int(0)   4
var_dump(strnatcmp("toto12","toto2"));      → int(1)   5
?>
```

```
<?
$ch1 = "abc";
$ch2 = "bcd";
if ($ch1 < $ch2) echo "<"; else echo ">";
?>
```

Tableaux

- On peut indicer les tableaux avec des entiers ou des chaînes de caractères ;
- La fonction `count($tab)` retourne le nombre d'éléments présents dans le tableau.

```
<?
  $a[2] = 12;
  $a[4] = 23;
  $a["toto"] = 12.13;
  var_dump($a); 1
  $b = count($a);
  var_dump($b); 2
?>
```

1
```
array(3) {
  [2]      => int(12)
  [4]      => int(23)
  ["toto"] => float(12.13)
  [clé]    => valeur
}
```

2
```
int(3)
```

Tableaux

- Le mot clé **array** : Il prend un nombre variable de paramètres sous la forme "clé => valeur" (ou simplement "valeur") :

```
<?
  $a = array(12=>3, "a"=>12.12, 15, "c", "1"=>2);
  var_dump($a); 1
  $a = array(1,2,3,4);
  var_dump($a); 2
?>
```

1
```
array(5) {
    [12]    => int(3)
    ["a"]   => float(12.12)
    [13]    => int(15)
    [14]    => string(1) "c"
    [1]     => int(2)
}
```

2
```
array(4) {
    [0]    => int(1)
    [1]    => int(2)
    [2]    => int(3)
    [3]    => int(4)
}
```

Intervalles

```
<?
$a=range(1,4);
var_dump($a); 1
$a=range(0,30,10);
var_dump($a); 2
$a=range('d','g');
var_dump($a); 3
?>
```

1
```
array(4) {
  [0] => int(1)
  [1] => int(2)
  [2] => int(3)
  [3] => int(4)
}
```

2
```
array(4) {
  [0] => int(0)
  [1] => int(10)
  [2] => int(20)
  [3] => int(30)
}
```

3
```
array(4) {
  [0] => string(1) "d"
  [1] => string(1) "e"
  [2] => string(1) "f"
  [3] => string(1) "g"
}
```

Ré-indexation

```php
<?
  $a = array(1,"a"=>2);
  $a[] = 3;
  var_dump($a); 1
  unset($a[1]);
  $a[] = 4;
  var_dump($a); 2
  $a = array_values($a);
  var_dump($a); 3
?>
```

1
```
array(3) {
  [0]    => int(1)
  ["a"]  => int(2)
  [1]    => int(3)
}
```

2
```
array(3) {
  [0]    => int(1)
  ["a"]  => int(2)
  [2]    => int(4)
}
```

3
```
array(3) {
  [0]    => int(1)
  [1]    => int(2)
  [2]    => int(4)
}
```

Tableaux multidimensionnels

```php
<?
   $a = array(12=>array(1,15=>2), 15=>2);
   var_dump($a); 1
   var_dump($a[12][15]); 2
   $a[12][20] = 2;
   var_dump($a[12]); 3
?>
```

2 int(2)

1
```
array(2) {
  [12]=> array(3) {
    [0]=> int(1)
    [15]=> int(2)
  }
  [15]=> int(2)
}
```

3
```
array(4) {
  [0]  => int(1)
  [15] => int(2)
  [20] => int(2)
}
```

foreach

- La boucle **foreach** parcourt tous les couples "clé ⇒ valeur" contenus dans un tableau :

```
<?
   $a = array(1=>12, "a"=>12.12, "c"=>3, 4);
   foreach ($a as $k=>$v)
      echo $k."=>".$v."\n"; 1
   foreach ($a as $v)
      echo $v."\n"; 2
?>
```

1 { 1=>12, a=>12.12, c=>3, 2=>4

2 { 12, 12.12, 3, 4

reset et each

```php
<?
   $a = array(1=>12, "a"=>12.12, "c"=>3, 4);
   reset($a);
   while ($tab=each($a))
     echo $tab[0]."=>".$tab[1]."\n"; (1)
   reset($a);
   while ($tab=each($a))
     echo $tab["key"]."=>".$tab["value"]."\n"; (1)
?>
```

(1)
$$\begin{cases} 1=>12 \\ a=>12.12 \\ c=>3 \\ 2=>4 \end{cases}$$

list

- la fonction **list** affecte plusieurs variables simultanément :

```
<?
   $a = array("a", 12, "c");
   list($x,$y,$z)=$a;
   var_dump($x); 1
   var_dump($y); 2
   var_dump($z); 3
   list($i,,$j)=$a;
   var_dump($i); 4
   var_dump($j); 5
   list($i,$j)=$a;
   var_dump($i); 6
   var_dump($j); 7
?>
```

1. string(1) "a"
2. int(12)
3. string(1) "c"
4. string(1) "a"
5. string(1) "c"
6. string(1) "a"
7. int(12)

list et each

```php
<?
   $a = array(1=>12, "a"=>12.12, "c"=>3, 4);
   reset($a);
   while (list($k,$v)=each($a))
     echo $k."=>".$v."\n"; (1)
?>
```

(1)
```
1=>12
a=>12.12
c=>3
2=>4
```

Manipulation d'un tableau

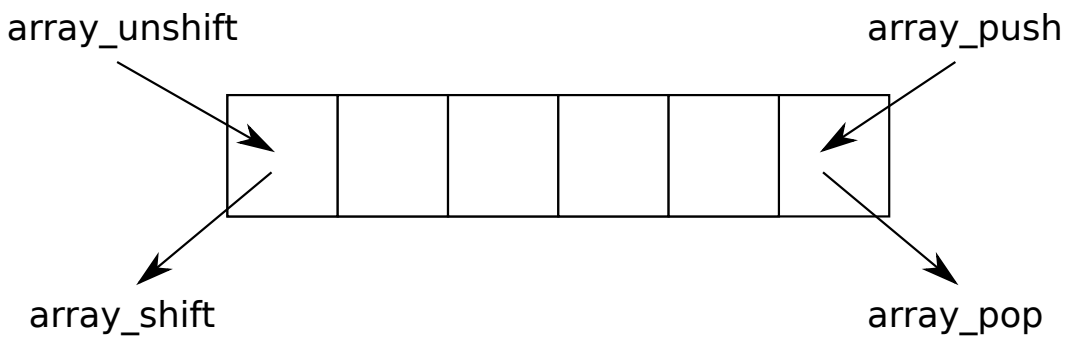
- `array_push($tab, $var, $var2, ...)` :

  empile des valeurs à la fin du tableau
- `$var = array_pop($tab)` :

  dépile une valeur située à la fin du tableau
- `array_unshift($tab, $var, $var2, ...)` :

  ajoute des valeurs au début du tableau
- `$var = array_shift($tab)` :

  supprime et retourne la première valeur du tableau



Manipulation d'un tableau

```php
<?
  $a=array(1, "a"=>2, 3);
  array_push($a, 12.12);
  array_unshift($a, "toto");
  var_dump($a); 1
  $b = array_pop($a);
  var_dump($b); 2
  $c = array_shift($a);
  var_dump($c); 3
?>
```

1
```
array(5) {
    [0]     => string(4) "toto"
    [1]     => int(1)
    ["a"]   => int(2)
    [2]     => int(3)
    [3]     => float(12.12)
}
```

2 float(12.12)

3 string(4) "toto"

Fusion de tableaux

```php
<?
  $a=array(1,"a"=>2, 3, "b"=>4, 4=>5);
  $b=array("c"=>6, 1=>7, 8, "b"=>9);
  $c=array_merge($a,$b);
  print_r($c); 1
  $c=array_merge_recursive($a,$b);
  print_r($c); 2
?>
```

1
```
Array (
  [0] => 1
  [a] => 2
  [1] => 3
  [b] => 9
  [2] => 5
  [c] => 6
  [3] => 7
  [4] => 8
)
```

2
```
Array (
  [0] => 1
  [a] => 2
  [1] => 3
  [b] => Array ( [0] => 4 [1] => 9 )
  [2] => 5
  [c] => 6
  [3] => 7
  [4] => 8
)
```

Intersection et différence de tableaux

```php
<?
  $a=array(1,"a"=>2, 3=>3, "b"=>4, 4=>5);
  $b=array("c"=>1, 1=>3, 4);
  $c=array_intersect($a,$b);
  print_r($c); 1
  $c=array_diff($a,$b);
  print_r($c); 2
?>
```

1
```
Array (
  [0] => 1
  [3] => 3
  [b] => 4
)
```

2
```
Array (
  [a] => 2
  [4] => 5
)
```

Tri de tableaux indicés

```php
<?
  $a=array("a10", "b11", "b"=>"a9", "C12", "c12");
  sort($a);
  print_r($c); 1
  rsort($a);
  print_r($c); 2
  natsort($a);
  print_r($c); 3
  natcasesort($a);
  print_r($c); 4
?>
```

1
```
Array (
  [0] => C12
  [1] => a10
  [2] => a9
  [3] => b11
  [4] => c12
)
```

2
```
Array (
  [0] => c12
  [1] => b11
  [2] => a9
  [3] => a10
  [4] => C12
)
```

1
```
Array (
  [4] => C12
  [2] => a9
  [3] => a10
  [1] => b11
  [0] => c12
)
```

2
```
Array (
  [2] => a9
  [3] => a10
  [1] => b11
  [0] => c12
  [4] => C12
)
```

Tri personalisé

```php
<?
  function comparaison($a, $b) {
    return ($a[0]+$a[1]) - ($b[0]+$b[1]);
  }

  $c = array(array(1,5),array(2,2), array(1,4));
  usort($c, "comparaison");
  print_r($c); 1
?>
```

1
```
Array (
  [0] => Array ([0] => 2 [1] => 2)
  [1] => Array ([0] => 1 [1] => 4)
  [2] => Array ([0] => 1 [1] => 5)
)
```

Tri de tableaux associatifs

```php
<?
  $a = array("a"=>"c", "b"=>"a", "c"=>"d");
  asort($a);
  print_r($a); 1
  arsort($a);
  print_r($a); 2
  sort($a);
  print_r($a); 3
?>
```

1
```
Array (
  [b] => a
  [a] => c
  [c] => d
)
```

2
```
Array (
  [c] => d
  [a] => c
  [b] => a
)
```

3
```
Array (
  [0] => a
  [1] => c
  [2] => d
)
```

Tri de tableaux associatifs

```php
<?
  $a = array("a"=>"c", "b"=>"a", "c"=>"d");
  ksort($a);
  print_r($a); 1
  krsort($a);
  print_r($a); 2
?>
```

1
```
Array (
  [a] => c
  [b] => a
  [c] => d
)
```

2
```
Array (
  [c] => d
  [b] => a
  [a] => c
)
```

Tri de tableaux associatifs

```php
<?
  function compar1($a,$b) {
    return ($a[0]+$a[1])-($b[0]+$b[1]);
  }

  function compar2($a,$b) {
    return strlen($a) - strlen($b);
  }

  $a = array("aa"=>array(0,1),
             "aaa"=>array(2,2),
             "a"=>array(1,2));
  uasort($a,"compar1");
  print_r($a); 1
  uksort($a,"compar2");
  print_r($a); 2
?>
```

1
```
Array (
  [aa] => Array([0] => 0 [1] => 1)
  [a] => Array([0] => 1 [1] => 2)
  [aaa] => Array([0] => 2 [1] => 2)
)
```

2
```
Array (
  [a] => Array([0] => 1 [1] => 2)
  [aa] => Array([0] => 0 [1] => 1)
  [aaa] => Array([0] => 2 [1] => 2)
)
```

Tri de tableaux associatifs

```php
<?
  function filtre($a) {
    return ($a[0] <= $a[1]);
  }

  $a = array("a"=>array(0,1),
             "b"=>array(3,2),
             "c"=>array(1,2),
             "d"=>array(1,0));

  $selection = array_filter($a, "filtre");
  print_r($selection); 1
?>
```

1
```
Array (
  [a] => Array([0] => 0 [1] => 1)
  [c] => Array([0] => 1 [1] => 2)
)
```

Appliquer une fonction à un tableau

```php
<?
  function affichage($a) {
   echo "<b>".$a[0]."</b> : ".$a[1]."<br/>\n"; 1
  }

  $a = array(array(0,1),
             array(3,2),
             array(1,2),
             array(1,0));

  array_walk($a, "affichage");
?>
```

1
0 : 1
3 : 2
1 : 2
1 : 0

Chaînes et tableaux

```php
<?
  $ch = "J'aime le PHP. Vive le web!";
  $tab = explode(' ',$ch);
  print_r($tab); 1
  $tab = explode('.',$ch);
  print_r($tab); 2
  $tab = array("J'aime", "le", "PHP.");
  $ch = implode(" ",$tab);
  echo $ch."\n"; 3
  $ch = implode("--",$tab);
  echo $ch."\n"; 4
?>
```

1
```
Array (
  [0] => J'aime
  [1] => le
  [2] => PHP.
  [3] => Vive
  [4] => le
  [5] => web !
)
```

2
```
Array (
  [0] => J'aime le PHP
  [1] => Vive le web !
)
```

3 J'aime le PHP.

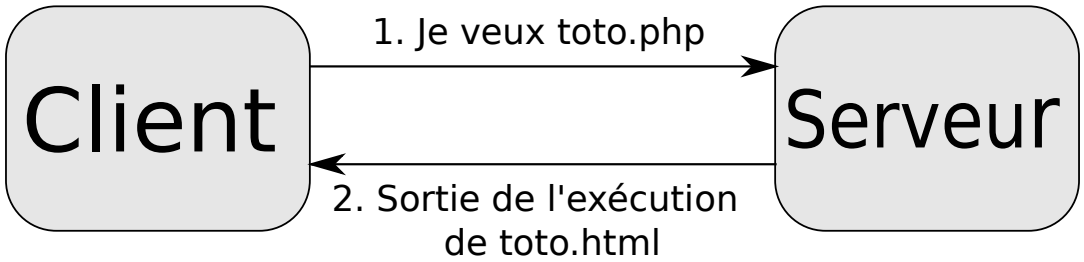
4 J'aime--le--PHP.

Autres fonctions utiles sur les tableaux

- La fonction `array_unique($tab)` supprime les valeurs en double dans le tableau (une seule clé est conservée).
- La fonction `$slice = array_slice($t, $p, $n)` extrait les `$n` éléments du tableau `$t` à partir de la position `$p`.
  (voir la documentation pour les autres utilisations)
- La fonction `shuffle($a)` mélange les éléments d'un tableau et renumérote les éléments.

GET

Protocole HTTP



Requête :

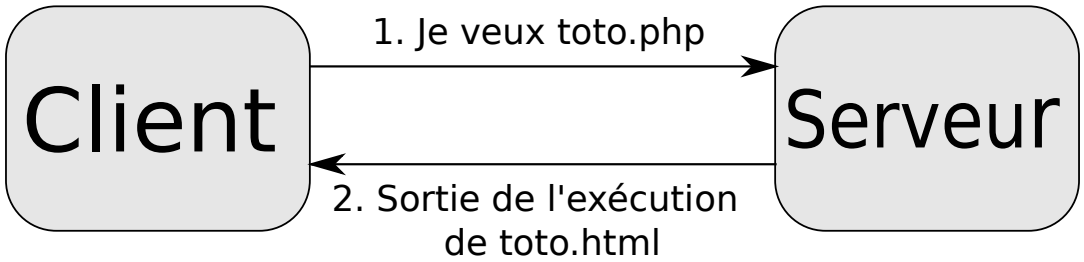
GET /toto.php ?n1=10&n2=15 HTTP/1.0
Host : example.com
Referer : http ://example2.com/
User-Agent : Mozilla/5.0 (X11 ; U ; Linux x86_64 ; fr ; rv :1.9.0.4) Gecko/2008111217 Fedora/3.0.4-1.fc10 Firefox/3.0.4

Réponse :

HTTP/1.0 200 OK
Date : Fri, 31 Dec 1999 23 :59 :59 GMT
Server : Apache/0.8.4
Content-Type : text/html
Content-Length : 59
Expires : Sat, 01 Jan 2000 00 :59 :59 GMT
Last-modified : Fri, 09 Aug 1996 14 :21 :40 GMT
<TITLE>Exemple</TITLE>
<P>Résultat : 25.</P>

POST

Protocole HTTP

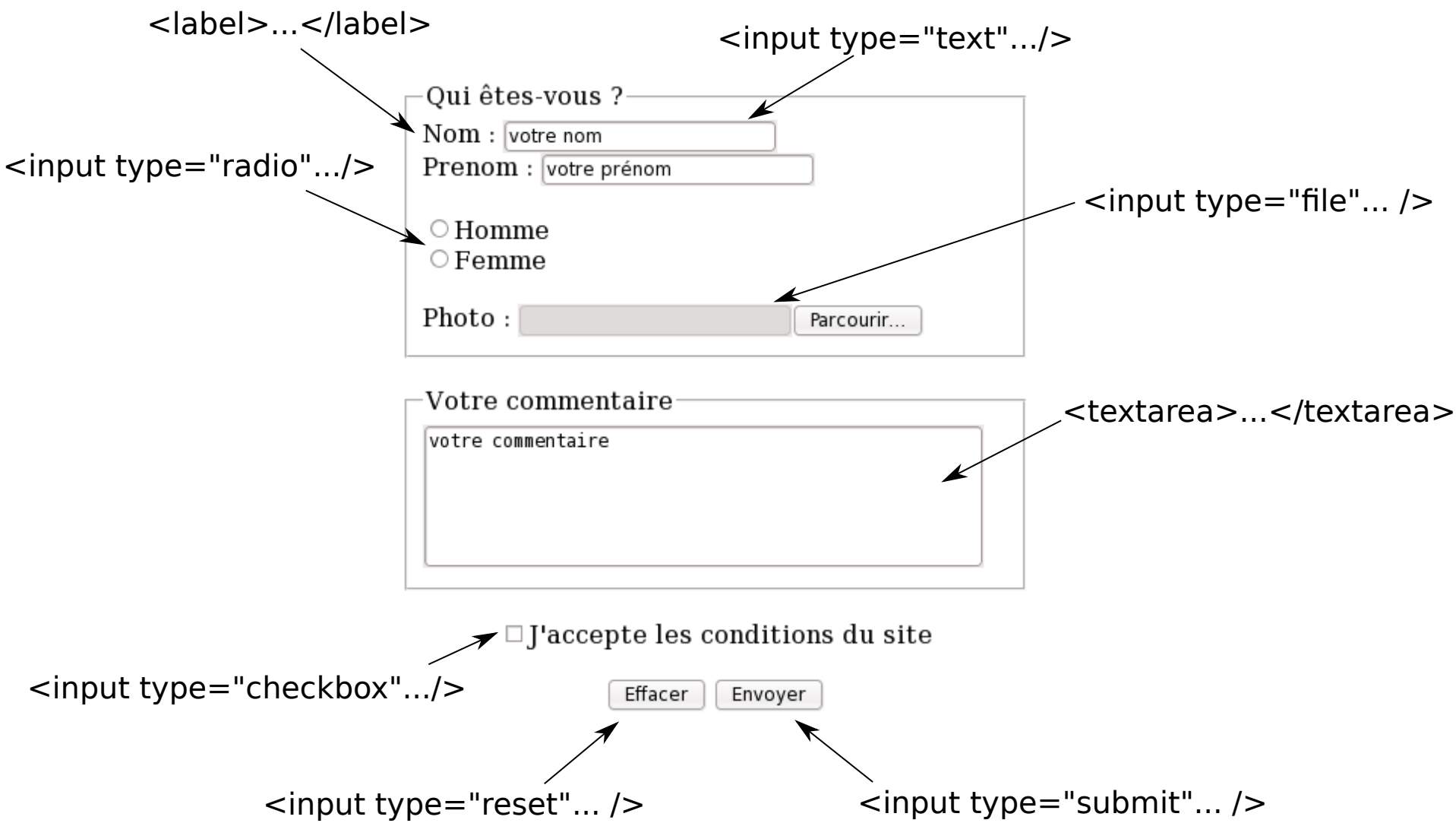


Requête :
POST /toto.php HTTP/1.0
Host : example.com
Referer : http ://example2.com/
User-Agent : Mozilla/5.0 (X11 ; U ; Linux x86_64 ; fr ; rv :1.9.0.4) Gecko/2008111217 Fedora/3.0.4-1.fc10 Firefox/3.0.4

n1=10&n2=15

Réponse :
HTTP/1.0 200 OK
Date : Fri, 31 Dec 1999 23 :59 :59 GMT
Server : Apache/0.8.4
Content-Type : text/html
Content-Length : 59
Expires : Sat, 01 Jan 2000 00 :59 :59 GMT
Last-modified : Fri, 09 Aug 1996 14 :21 :40 GMT
<TITLE>Exemple</TITLE>
<P>Résultat : 25.</P>

Les formulaires en HTML



Les formulaires en HTML

```
<html>
 <body>
  <form action="page.php" method="post">
   <fieldset>
     <legend>Qui etes-vous ?</legend>
     <label>Nom :</label>
     <input type="text" name="nom" value="votre nom"/>
     <label>Prenom :</label>
     <input type="text" name="prenom" value="votre prenom"/>
     <input type="radio" name="sexe" value="homme"/>Homme
     <input type="radio" name="sexe" value="femme"/>Femme
     <label>Photo :</label>
     <input type="file" name="photo" accept="image/jpeg" />
   </fieldset>
   <fieldset>
    <legend>Votre commentaire</legend>
    <textarea name="commentaire">votre commentaire</textarea>
   </fieldset>
   <center>
    <input type="checkbox" name="valid" value="valid"/>J accepte...
    <input type="reset" value="Effacer">
    <input type="submit" value="Envoyer">
   </center>
  </form>
 </body>
</html>
```

- Envoi des données au serveur par la méthode `POST` sur la page `page.php` lorsque l'utilisateur clique sur le bouton `Envoyer`.

Variables “super-globales”

- Les variables **super-globales** sont accessibles dans tous les contextes

Fichier **test.php** :

```
<?
  function toto() {
    echo $_SERVER["PHP_SELF"]."\n"; (1)
  }

  toto();

  echo $_SERVER["PHP_SELF"]."\n"; (1)
?>
```

(1) "test.php"

Variable $_POST

- Fichier **index.html**

```
<html>
<body>
 <form method="post" action="traitement.php">
 <label>Nom : </label>
 <input type="text" name="nom"/>
 <input type="submit" value="Envoyer"/>
 </form>
</body>
</html>
```

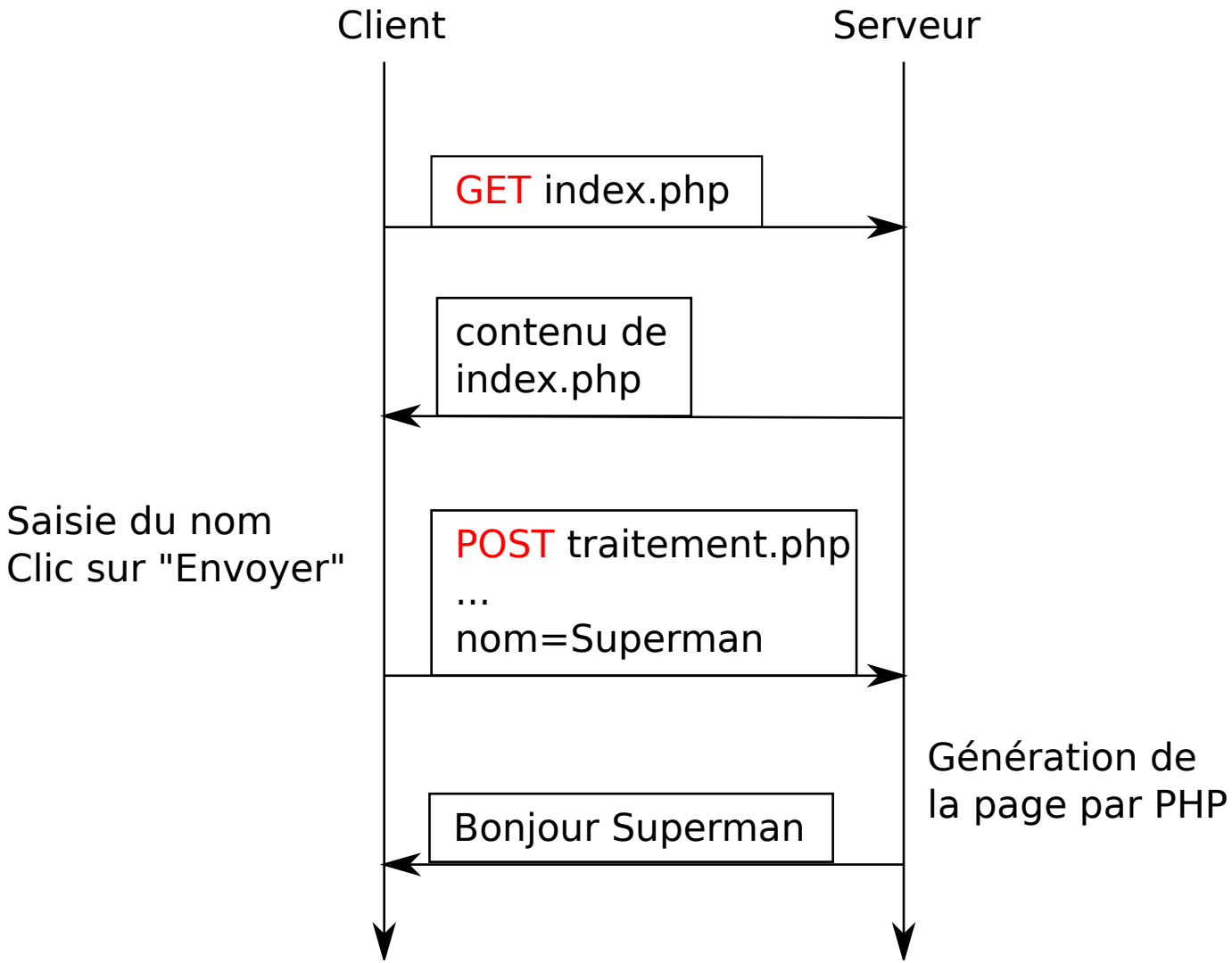
Requête :
POST traitement.php
...
nom=Superman

- Fichier **traitement.php**

```
<html>
<body>
 Bonjour <? echo $_POST['nom']; ?>.</br>
</body>
</html>
```

Réponse :
...
Bonjour Superman.
...

Variable $_POST



Variable $_GET

- Fichier **index.html**

```
<html>
<body>
 <form method="get" action="traitement.php">
 <label>Nom : </label>
 <input type="text" name="nom"/>
 <input type="submit" value="Envoyer"/>
 </form>
</body>
</html>
```

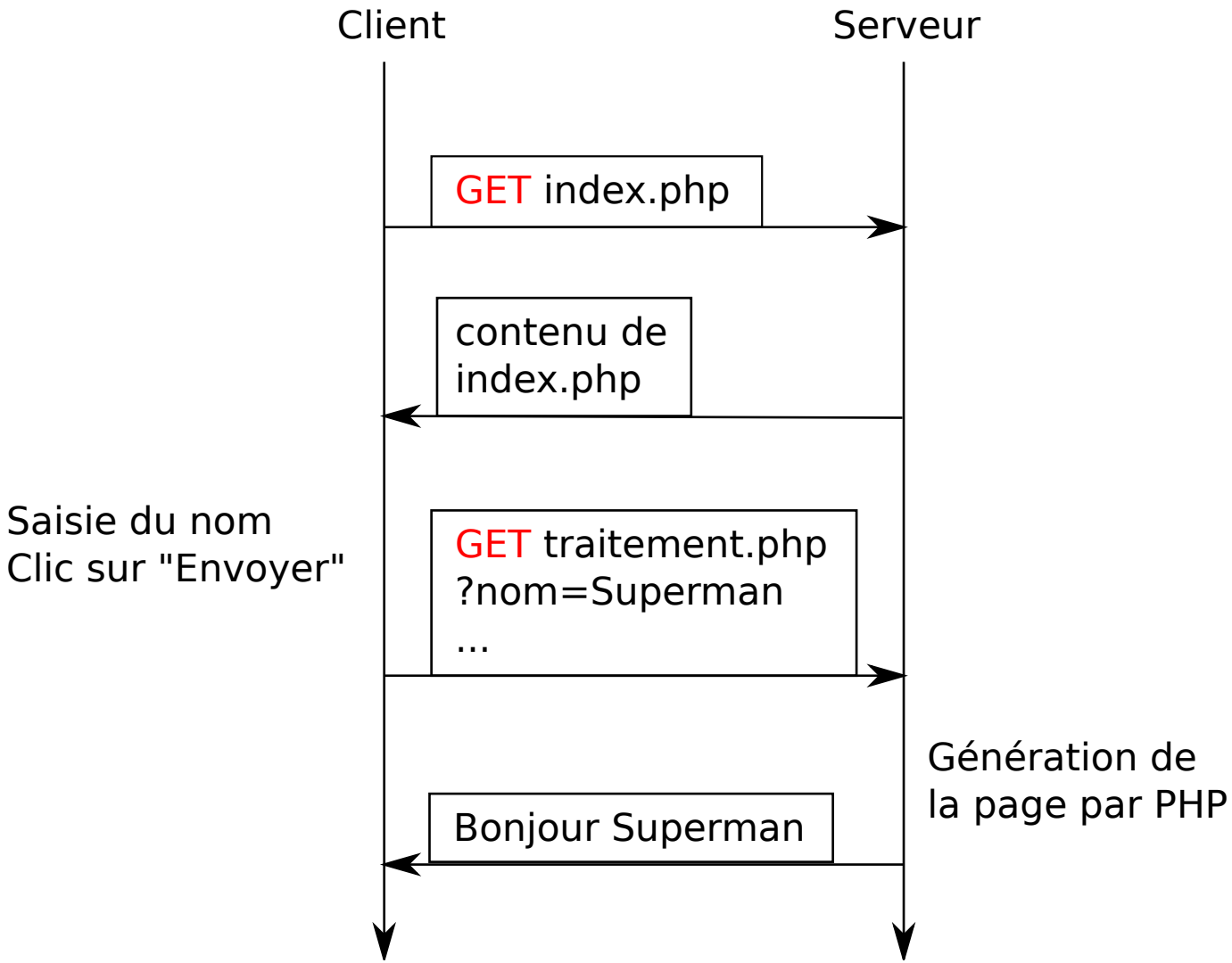
Requête :
GET traitement.php?nom=superman
...

- Fichier **traitement.php**

Réponse :
...
Bonjour Superman.
...

```
<html>
<body>
 Bonjour <? echo $_GET['nom']; ?>.</br>
</body>
</html>
```

Variable $_GET



Bouton radio

- Fichier **index.html** :

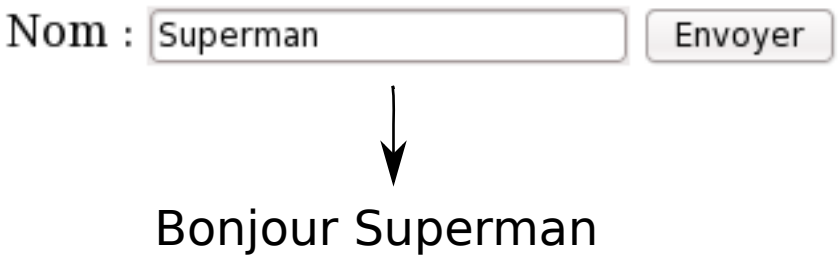
```
<html>
 <body>
  <form method="post" action="traitement.php">
   <input type="radio" name="sexe" value="homme">Homme</br>
   <input type="radio" name="sexe" value="femme">Femme</br>
   <input type="submit" value="Envoyer"/>
  </form>
 </body>
</html>
```

- Fichier **traitement.php** :

```
<html>
 <body>
  <? if ($_POST['sexe']==='homme') { ?>
   Bonjour, vous etes un homme.
  <?} else {?>
   Bonjour, vous etes une femme.
  <? } ?>
 </body>
</html>
```

Avec un seul fichier

```
<html>
<body>
 <? if (isset($_POST['nom'])) {?>
    Bonjour <? echo $_POST['nom']; ?>
 <?} else {?>
 <form method="post" action="<? echo $_SERVER["PHP_SELF"]; ?>" >
 <label>Nom : </label>
 <input type="text" name="nom"/>
 <input type="submit" value="Envoyer"/>
 </form>
 <?}?>
</body>
</html>
```

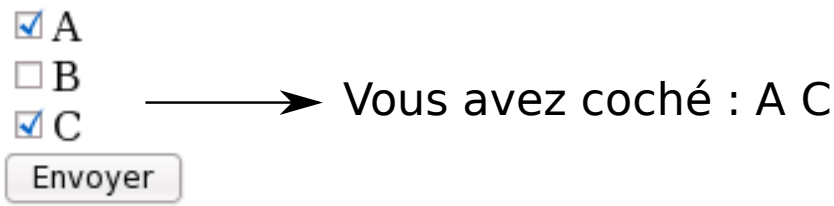


Valeurs multiples et checkboxes

- Fichier **index.html** :

```
<html>
<body>
 <form method="post" action="traitement.php">
 <input type="checkbox" name="choix[]" value="A">A</br>
 <input type="checkbox" name="choix[]" value="B">B</br>
 <input type="checkbox" name="choix[]" value="C">C</br>
 <input type="submit" value="Envoyer"/>
 </form>
</body>
</html>
```

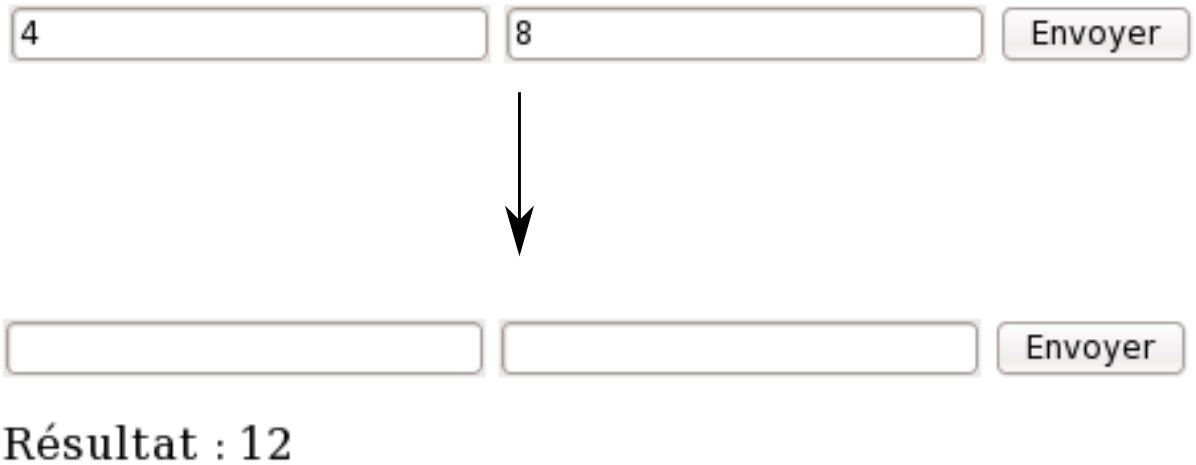
- Fichier **traitement.php** :




```
<html>
<body>
Vous avez coche :
<? foreach ($_POST['choix'] as $v)
    echo "$v ";  ?>
<br/>
</body>
</html>
```

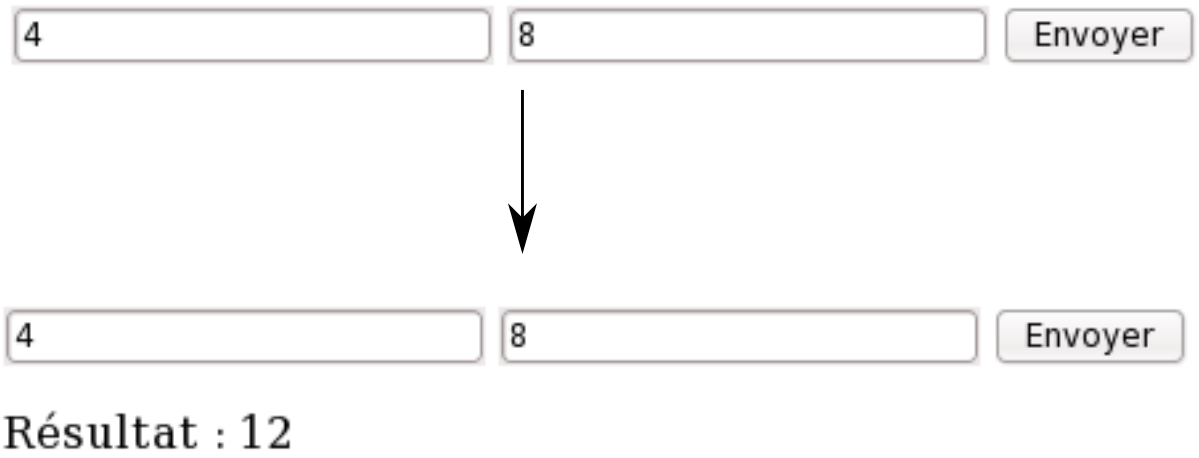
Maintient de l'état du formulaire

```
<html>
<body>
 <form method="post" action="<? echo $_SERVER["PHP_SELF"]; ?>">
 <input type="text" name="a" />
 <input type="text" name="b" />
 <input type="submit" value="Envoyer"/>
 </form>
 <? if (isset($_POST['a']) && isset($_POST['b'])) {?>
    Resultat : <? echo $_POST['a']+$_POST['b']; ?>
 <?}?>
</body>
</html>
```



Maintient de l'état du formulaire

```
<html>
<body>
 <form method="post" action="<? echo $_SERVER["PHP_SELF"]; ?>">
 <input type="text" name="a" value="<? echo $_POST['a'];?>" />
 <input type="text" name="b" value="<? echo $_POST['b'];?>" />
 <input type="submit" value="Envoyer"/>
 </form>
 <? if (isset($_POST['a']) && isset($_POST['b'])) {?>
    Resultat : <? echo $_POST['a']+$_POST['b']; ?>
 <?}?>
</body>
</html>
```



Transfert de fichiers

```
<html>
<body>
 <form enctype="multipart/form-data" method="post"
       action="<? echo $_SERVER["PHP_SELF"]; ?>">
 <input type="file" name="fichier" />
 <input type="submit" value="Envoyer" />
 </form>
<? var_dump($_FILES); (1) ?>
</body>
</html>
```

(1)
```
array(1) {
  ["fichier"]=> array(5) {
    ["name"]]=> string(3) "a.c"
    ["type"]=> string(10) "text/plain"
    ["tmp_name"]=> string(14) "/tmp/phpi5JOt8"
    ["error"]=> int(0)
    ["size"]=> int(185)
  }
}
```

Transfert de fichiers

```
<html>
<body>
 <form enctype="multipart/form-data" method="post"
       action="<? echo $_SERVER["PHP_SELF"]; ?>">
 <input type="file" name="fichier" />
 <input type="submit" value="Envoyer" />
 </form>
<? if (isset($_FILES['fichier'])) {
        $tmpname = $_FILES['fichier']['tmp_name'];
        $newname = "fichier_".$_FILES['fichier']['name'];
        echo $tmpname." ".$newname."<br/>";
        $result = move_uploaded_file($tmpname, $newname);
        if ($result==TRUE) echo "transfert ok !<br/>";
        else echo "erreur : ".$_FILES['fichier']['error'];
}
?>
</body>
</html>
```

Transfert de fichiers

Aix*Marseille université

Les erreurs possibles :

- `UPLOAD_ERR_OK` (*valeur 0*) :
  ↪ pas d'erreur
- `UPLOAD_ERR_INI_SIZE` (*valeur 1*) :
  ↪ la taille du fichier dépasse la valeur présente dans le fichier php.ini.
- `UPLOAD_ERR_FORM_SIZE` (*valeur 2*) :
  ↪ la taille du fichier dépasse celle fixée dans le formulaire (champ caché `MAX_FILE_SIZE`).
- `UPLOAD_ERR_PARTIAL` (*valeur 3*) :
  ↪ le fichier a été partiellement téléchargé.
- `UPLOAD_ERR_NO_FILE` (*valeur 4*) :
  ↪ aucun fichier n'a été téléchargé.

Ouverture d'un fichier

```php
<?
        $id = fopen("toto.txt", "w");
        var_dump($id); 1
        fclose($id);
        var_dump($id); 2

        $id = tmpfile();
        var_dump($id); 3
        fclose($id);
        var_dump($id); 4
?>
```

1. resource(5) of type (stream)
2. resource(5) of type (Unknown)
3. resource(6) of type (stream)
4. resource(6) of type (Unknown)

- Une bonne façon de faire :

```php
<?
        $id = fopen("toto.txt", "w");
        if ($id===FALSE) die("Erreur");
        $r = fclose($id);
        if ($r===FALSE) die("Erreur");
?>
```

Ouverture d'un fichier

Aix*Marseille université

r : ouverture en lecture seule et la lecture commence au début du fichier. *Le fichier doit exister*.

r+ : le fichier est ouvert en lecture et écriture, les opérations commencent au début. *Le fichier doit exister*.

w : le fichier est ouvert en écriture et l'écriture commence au début du fichier. *Le fichier est créé s'il n'existe pas*.

w+ : le fichier est ouvert en écriture et en lecture et les opérations commencent au début du fichier. *Le fichier est créé s'il n'existe pas*.

a : le fichier est ouvert en écriture et l'écriture commence à la fin du fichier. *Le fichier est créé s'il n'existe pas*.

a+ : le fichier est ouvert en écriture et lecture. L'écriture commence à la fin et la lecture au début. *Le fichier est créé s'il n'existe pas*.

Verrouillage de fichiers

- **Problème de concurrence** :

  Plusieurs clients demandent une page simultanément
  ⇒ deux scripts modifient un fichier en même temps
  ⇒ Conflit

- **Solution** : verrouillage du fichier

```
<?
  $id = fopen("toto.txt", "w");
  echo $id."\n";
  flock($id, LOCK_EX); Verrouillage en lecture et en écriture
  ....
  flock($id, LOCK_UN); Déverrouillage
  fclose($id);
?>
```

- `LOCK_SH` : Verrouillage en écriture seulement

Écriture dans un fichier

```php
<?
  $id = fopen("toto.txt", "w");
  flock($id, LOCK_SH);
  fwrite($id, "mon texte\n"); Écriture de "mon texte\n"
  $nb = 100;
  fwrite($id, $nb); Écriture de "100"
  flock($id, LOCK_UN);
  fclose($id);
?>
```

- Contenu du fichier `toto.txt` à la fin de l'exécution :

```
mon texte
100
```

Exemple d'écriture dans un fichier

```
<html>
<body>
 <form action="<? echo $_SERVER['PHP_MYSELF']; ?>" method="post">
  <b>Nom :</b> <input type="text" name="nom"/><br/>
  <b>Prenom :</b> <input type="text" name="prenom"/><br/>
  <input type="submit" value="Envoyer"/>
 </form>
 <?
  if (isset($_POST['nom']) && isset($_POST['prenom'])) {
   $nom = $_POST['nom'];
   $prenom = $_POST['prenom'];
   $id = fopen("liste.txt", "a");
   if ($id===FALSE) die("erreur");
   flock($id, LOCK_SH);
   fwrite($id, "$nom;$prenom\n");
   flock($id, LOCK_UN);
   $r = fclose($id);
   if ($r===FALSE) die("erreur");
  }
 ?>
</body>
</html>
```

```
jean;pascal
didier;julien
paul;jacques
michel;jean
```

Lecture dans un fichier

- Contenu du fichier `toto.txt` avant l'exécution :

```
mon texte
100
```

```php
<?
  $id = fopen("toto.txt", "r");
  $s = fgets($id, 256);
  var_dump($s); string(10) "mon texte\n"
  $s = fgets($id, 256);
  var_dump($s); string(4) "100\n"
  fclose($id);
?>
```

- La fonction `fgets` prend deux paramètres : une ressource pointant sur un fichier et le nombre maximum de caractères à lire

Exemple de lecture dans un fichier

```
<table border="1">
<tr><th>Nom</th><th>Prenom</th></tr>
<?
  $id = fopen("liste.txt", "r");
  while ($ligne=fgets($id)) {
   $t = explode(";", $ligne);
   echo "<tr><td>$t[0]</td><td>$t[1]</td></tr>";
  }
  flcose($id);
?>
</table>
```

```
jean;pascal
didier;julien
paul;jacques
michel;jean
```

$\Rightarrow$

Nom	Prenom
jean	pascal
didier	julien
paul	jacques
michel	jean

Lecture de données formatées

```
<table border="1">
 <tr><th>Nom</th><th>Prenom</th></tr>
 <?
  $id = fopen("liste.txt", "r");
  while ($t=fgetcsv($id, 100 (1), ";" (2))) {
   echo "<tr><td>$t[0]</td><td>$t[1]</td></tr>";
  }
  flcose($id);
 ?>
 </table>
```

(1) : nombre maximum de caractères à lire

(2) : caractère séparateur

```
jean;pascal
didier;julien
paul;jacques
michel;jean
```

$\Rightarrow$

Nom	Prenom
jean	pascal
didier	julien
paul	jacques
michel	jean

Lecture de la totalité d'un fichier

```
<table border="1">
 <tr><th>Nom</th><th>Prenom</th></tr>
 <?
   $f = file("liste.txt");
   foreach ($f as $ligne) {
    $t = explode(";", $ligne);
    echo "<tr><td>$t[0]</td><td>$t[1]</td></tr>";
   }
 ?>
 </table>
```

```
jean;pascal
didier;julien
paul;jacques
michel;jean
```

$\Rightarrow$

Nom	Prenom
jean	pascal
didier	julien
paul	jacques
michel	jean

Manipulations de fichiers

- Copie de fichiers :

```
<?
  $res = copy("liste.txt", "liste2.txt");
  if ($res===FALSE) die("erreur");
?>
```

- Renommer un fichier :

```
<?
  $res = rename("liste.txt", "liste2.txt");
  if ($res===FALSE) die("erreur");
?>
```

- Supprimer un fichier :

```
<?
  $res = unlink("liste.txt");
  if ($res===FALSE) die("erreur");
?>
```

Manipulations de fichiers

- Existence :

```
<?
  if (file_exists("liste.txt"))
   echo "le fichier existe";
  else echo "le fichier n'existe pas";
?>
```

- Créer un fichier vide :

```
<?
  if (!file_exists("liste.txt"))
   touch("liste.txt", time());
?>
```

- Taille d'un fichier :

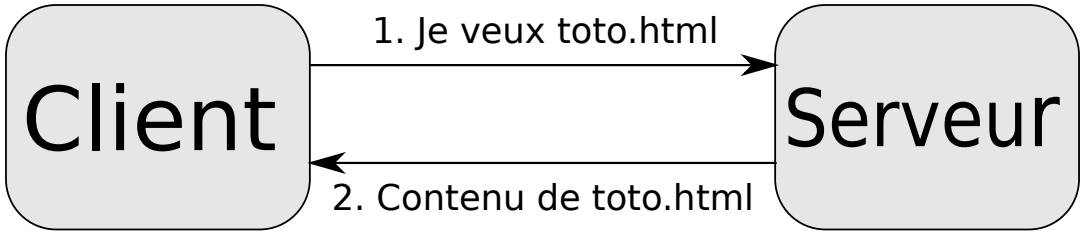
```
<?
  $nombre_octets = filesize("liste.txt");
  echo $nombre_octets;
?>
```

Autres fonctions utiles

- `is_file("fichier")` = true s'il s'agit d'un fichier
- `is_readable("fichier")` = true si le fichier est dispo en lecture
- `is_writable("fichier")` = true si le fichier est dispo. en écriture
- `filetype("fichier)` = le type du fichier
- `basename("img/truc/toto.php")` = "toto.php"
- `realpath("toto.php")` = chemin complet du fichier
- `fseek($id, n)` → se positionne sur le n-ème octet
- `rewind($id)` → se positionne au début du fichier
- `ftell($id)` = position courante du curseur dans le fichier
- `fgetc($id)` = le prochain caractère du fichier

Headers

Protocole HTTP



Requête :
GET /toto.html HTTP/1.0
Host : example.com
Referer : http ://example2.com/
User-Agent : Mozilla/5.0 (X11 ; U ; Linux x86_64 ; fr ; rv :1.9.0.4) Gecko/2008111217 Fedora/3.0.4-1.fc10 Firefox/3.0.4

Réponse :
HTTP/1.0 200 OK
Date : Fri, 31 Dec 1999 23 :59 :59 GMT
Server : Apache/0.8.4
Content-Type : text/html
Content-Length : 59
Expires : Sat, 01 Jan 2000 00 :59 :59 GMT
Last-modified : Fri, 09 Aug 1996 14 :21 :40 GMT
<TITLE>Exemple</TITLE>
<P>page d'exemple.</P>

Modification des headers

- La fonction `header(...)` permet d'ajouter (ou de modifier) des informations présentes dans l'en-tête retourné au client :

```
<?
  ...
  if ($autorized) { ?>
   <html>
    <body>
      Bienvenue</br>
    </body>
   </html>
 <? } else
   header('Location: http://www.google.fr');
?>
```

- La fonction **header** doit être appelée avant que le moindre contenu ne soit envoyé

Error 404 : Page Not Found

- Pour simuler le fait qu'une page n'a pas été trouvée sur le serveur :

```
<?
  header("HTTP/1.0 404 Not Found");
  exit; Termine l'exécution du programme
  echo "Bienvenue"; Cette instruction n'est pas exécutée
?>
```

- Attention :

```
<?
  header("HTTP/1.0 404 Not Found");
  echo "Bienvenue"; "Bienvenue" est envoyé au client...
  exit; trop tard !
?>
```

Redirection

- Pour rediriger le client vers une autre page :

```
<?
  if (!$autorized) {
   header('Location: http://www.google.fr');
   exit; pour éviter que la suite ne soit envoyé au client...
  }
  ...
?>
```

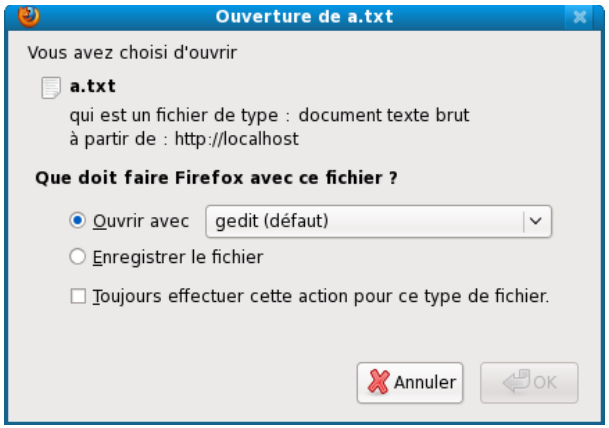
- Pour rediriger le client après un certain délai :

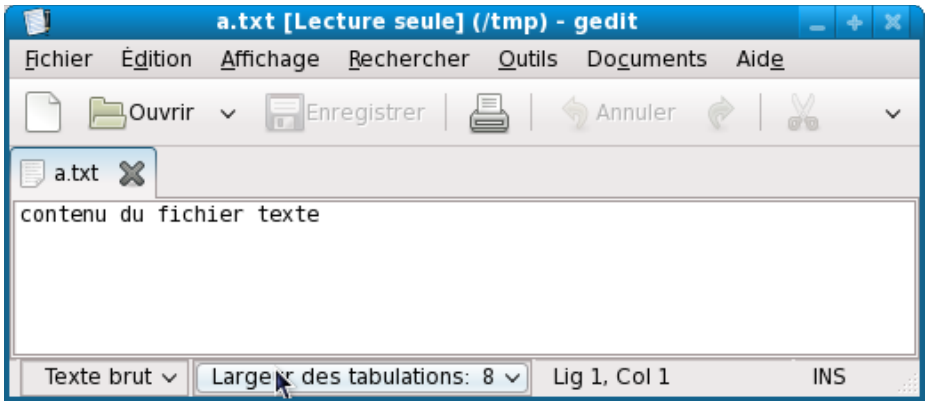
```
<?
  echo "Vous allez etre redirige...<br/>"
  header('Refresh:10; http://www.newsite.fr');
?>
```

Content-Type et Content-Disposition

- Il est possible d'envoyer un fichier texte en PHP :

```
<?
  header('Content-Type: text/plain');
  header('Content-Disposition: attachement;filename="a.txt"');
?>
contenu du fichier texte
```





- Il est également possible d'envoyer une image :

```
<?
  header('Content-Type: image/jpeg');
  header('Content-Disposition: attachement;filename="a.jpg"');
  readfile("image.jpg"); écrit le contenu du fichier ici !
?>
```

Application : restreindre l'accès à un fichier

- Il est possible de restreindre l'accès à un fichier :

```
<?
  ... Initialisation de la variable $accespossible
  if ($accespossible) {
   header('Content-Type: application/pdf');
   header('Content-Disposition: attachement;filename="a.pdf"');
   readfile('doc.pdf');
   exit;
  } else { ?>
  <html>
   <body>
    Vous n'avez pas acces a ce fichier !<br/>
   </body>
  </html>
  <? }
?>
```

- **Attention** : le fichier ne doit pas être accessible directement sur le serveur, c'est-à-dire, sans utiliser ce fichier php.

Types MIME (Multipurpose Internet Mail Extensi

Aix*Marseille université

- **application/octet-stream** : flux de données arbitraire
- **application/ogg** : Ogg
- **application/pdf** : PDF
- **application/xhtml+xml** : XHTML
- **application/x-shockwave-flash** : Flash
- **audio/mpeg** : fichier MP3 ou MPEG
- **audio/x-ms-wma** : fichier Windows Media Audio
- **audio/vnd.rn-realaudio** : fichier RealAudio
- **audio/x-wav** : fichier WAV
- **image/gif** : image GIF
- **image/jpeg** : image JPEG
- **image/png** : image PNG
- **image/tiff** : image TIFF
- **text/css** : Feuille de style
- **text/html** : fichier HTML
- **text/plain** : Données textuelles
- **text/xml** : fichier XML
- **video/mpeg** : vidéo MPEG-1
- **video/mp4** : vidéo MP4
- **video/quicktime** : vidéo QuickTime
- **video/x-flv** : Vidéo Flash

Les cookies

- Un cookie est un petit fichier placé sur l'ordinateur du visiteur
- Les cookies servent à stocker de l'information chez le visiteur
- PHP permet d'écrire et de lire les cookies sur l'ordinateur du visiteur
- Le visiteur peut interdire ou supprimer les cookies
- Le visiteur peut modifier les informations contenues dans les cookies
- Chaque site peut écrire un nombre limité de cookies sur chaque client
- Un cookie ne doit pas dépasser 4 Kio

Remarque : les cookies sont souvent utilisés pour stocker chez le visiteur ses préférences (présentation personnalisée du site).

Écriture des cookies

- Pour écrire un cookie, il faut utiliser la fonction `setcookie` :

```
<?
  setcookie("nom", "valeur");
?>
```

- Comme pour la fonction `header`, la fonction `setcookie` doit être appelée avant d'écrire sur la sortie standard.
- Par défaut, le cookie expire à la fermeture du navigateur et est accessible par toutes les pages de votre domaine. Pour changer cela :

```
<?
  setcookie("nom",
            "valeur",
            time()+3600, valable une heure
            "/chemin/",
            "www.domaine.com"
           );
?>
```

Lecture des cookies

- La lecture des cookies se fait via la variable superglobale `$_COOKIE` :
  - Contenu de la page **page1.php** :

```
<?
  setcookie("nom1", "valeur1");
  setcookie("nom2", "valeur2");
?>
<a href="page2.html">page2</a>
```

  - Contenu de la page **page2.php** :

```
<?
  echo $_COOKIE["nom1"]."</br>";  affiche valeur1
  echo $_COOKIE["nom2"]."</br>";  affiche valeur2
?>
```

- **Attention** : Les cookies ne sont pas immédiatement accessibles par la page qui vient de les créer.

Suppression du contenu d'un cookie

- Pour supprimer le contenu d'un cookie :

```
<?
  setcookie("nom"); affecte la chaîne vide au cookie
?>
```

- **Attention** :
  La suppression du cookie est effective au rechargement de la page.

Tableaux associatifs et cookies

Exemple avec des tableaux associatifs :

- Contenu de la page **page1.php** :

```
<?
  setcookie("tab['key1']", "valeur1");
  setcookie("tab['key2']", "valeur2");
  setcookie("tab['key3']", "valeur3");
?>
<a href="page2.html">page2</a>
```

- Contenu de la page **page2.php** :

```
<?
  foreach ($_COOKIE["tab"] as $cle=>$valeur) {
    echo $cle."=>".$valeur."␣"; (1)
  }
?>
```

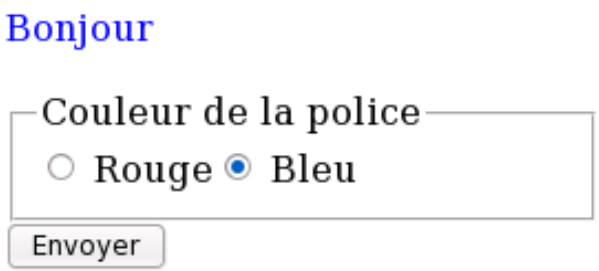
(1) `key1=>valeur1 key2=>valeur2 key3=>valeur3`

Exemple d'utilisation des cookies

- Fin du fichier **index.php** :

Initialisation de la variable $couleur (slide suivant)

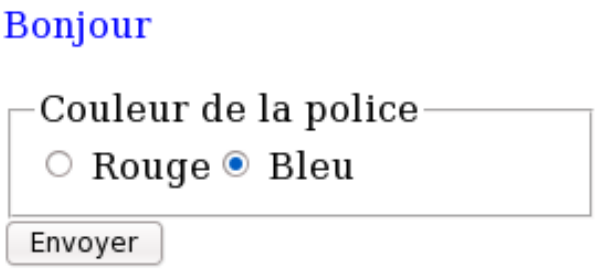
```
<?
 function addColor($name, $label, $couleur) {
   echo '<input type="radio" name="couleur" value="'.$name.'" ';
   if ($couleur===$name) echo 'checked="checked"';
   echo '/> '.$label;
 }
?>
<html>
 <body>
  <font style="color:<? echo $couleur; ?>">Bonjour</font><br/>
  <form action="<? echo $_SERVER['PHP_SELF']; ?>" method="post">
   <fieldset><legend>Couleur de la police</legend>
    <?
     addColor('red', 'Rouge', $couleur);
     addColor('blue', 'Bleu', $couleur);
    ?>
   </fieldset><input type="submit"/>
  </form>
 </body>
</html>
```

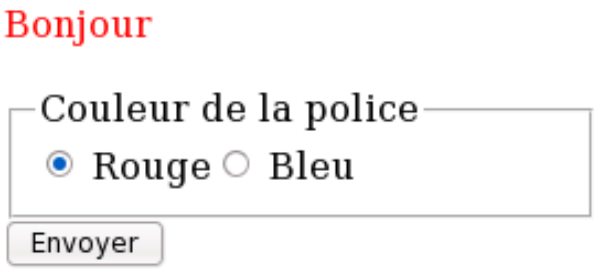


Exemple d'utilisation des cookies

- Au début du fichier **index.php** :

```
<?
  $couleur = 'red'; couleur par défaut
  if (isset($_POST['couleur'])) { traitement des données du formulaire
    $couleur=$_POST['couleur'];
    setcookie("couleur", $couleur); sauvegarde de la préférence chez le visiteur
  } else if (isset($_COOKIE['couleur'])) 
    $couleur=$_COOKIE['couleur']; lecture de la préférence chez le visiteur
?>
```





Le mécanisme des sessions

Objectif : Conserver des informations d'un même client entre les pages.

Les étapes du mécanisme des sessions :

1. au début de chaque page, l'appel de `session_start()` crée une session ou restaure la session trouvée sur le serveur via l'identifiant de session passé dans une requête GET, POST ou par un cookie.
2. Au moment de la création de la session, chaque utilisateur se voit attribuer un identifiant (de session) composé de 26 caractères aléatoires. Il est transmit d'une page à l'autre par l'intermédiaire d'un cookie placé sur le poste du client, soit dans l'URL.
3. le tableau superglobal `$_SESSION` permet de stocker des données liées à la session et accessibles sur toutes les pages du site.

Exemples avec cookies

index.php :

```
<?
  session_start();
  $_SESSION['info']="moninformation";
  echo '<a href="pagesuivante.php">page suivante</a>';
?>
```

pagesuivante.php :

```
<?
  session_start();
  echo $_SESSION['info']; affiche moninformation
?>
```

Sans cookies

Sans cookie, on peut transmettre l'identifiant de session via l'URL.

le contenu de $SID est de la forme 'PHPSESSID=identifiant de 26 car.'

index.php :

```
<?
  session_start();
  $_SESSION['info']="moninformation";
  echo '<a href="pagesuivante.php?'.$ID.'">';
  echo 'page suivante'
  echo '</a>';
?>
```

pagesuivante.php :

```
<?
  session_start();
  echo $_SESSION['info']; affiche moninformation
?>
```

Mécanisme d'authentification

Aix*Marseille université

```php
session_start();

if (isset($_POST['login'])
 && isset($_POST['passwd'])
 && verifierPassword($_POST['login'],
                     $_POST['passwd']))
        $_SESSION['login'] = $_POST['login'];
else if (isset($_GET['deconnexion']))
        $_SESSION['login']=null;

$login = null;
if (isset($_SESSION['login']))
        $login = $_SESSION['login'];
if ($login===null) {
        include("FormulaireConnexion.php");
} else
        include("FormulaireDeconnexion.php");
```

Mécanisme d'authentification

FormulaireConnexion.php :

```
<form method="POST" action="index.php">
login :
<input type="text" name="login"/><br/>
mot de passe :
<input type="password" name="passwd"/><br/>
<input type="submit"/>
</form>
```

FormulaireDeconnexion.php :

```
Bonjour <? echo $login; ?>,<br/>
Pour vous deconnecter, cliquez
<a href="index.php?deconnexion">ici</a>.
```

Mécanisme d'authentification

image.php :

```
<?
  session_start();

  $login = null;
  if (isset($_SESSION['login']))
        $login = $_SESSION['login'];

  if ($login===null) {
   header("Location:index.php");
  } else {
   header("Content-Type:image/jpeg");
   readfile("img/toto.jpg");
  }

?>
```

Programmation Orientée Objet

La programmation orientée objet de PHP 5 est similaire à celle de Java.

Nous allons voir comment :

- Définir une classe
- Créer une instance
- Accéder aux propriétés et invoquer des méthodes
- Modifier l'accessibilité aux propriétés et aux méthodes
- Définir un constructeur et un destructeur
- Définir des interfaces, des classes abstraites
- Utiliser l'héritage
- Cloner des instances

Définition d'une classe

Pour définir une classe, on utilise le mot-clé `class`.

```php
<?
   class uneClasse {

      public $p1;
      public $p2 = 12;
      public $p3 = array("toto", 12);
      // public $p4 = strlen('toto'); interdit !

      public function maMethode($arg1, $arg2) {
         ...
      }
   }
?>
```

Création d'une instance

Pour créer une instance d'une classe, on utilise le mot-clé `new`.

```
<?
   class uneClasse {

     public $p1;
     public $p2 = 12;
     public $p3 = array("toto", 12);
     // public $p4 = strlen('toto'); interdit !

     public function maMethode($arg1, $arg2) {...}
   }

  $i1 = new maClasse();
  $i2 = new maClasse();
?>
```

Accès aux propriétés et invocation de méthodes

Aix*Marseille université

```
<?
  class UneClasse {
    public $p;
    public function maMethode($arg) { ... }
  }

  $i = new UneClasse();

  $i->p = 12; pas de $ dans le nom de la propriété
  echo $i->p;
  $i->p = array(1,2);
  echo $i->p[1]; idem avec les tableaux

  $i->maMethode("toto"); avec la -> comme en C++
?>
```

$this

```php
<?
  class maClasse {
          public $p = 0;

          public function maMethode() {
           $this->p+=1;
           echo $this->p."\n";
          }

          public function maMethodeBis() {
           $this->maMethode();
          }
  }
  $i = new maClasse();
  $i->maMethode(); affiche 1
  $i->maMethode(); affiche 2
  $i->maMethodeBis(); affiche 3
?>
```

Constantes – propriétés et méthodes statiques

Aix*Marseille université

```php
<?
  class maClasse {
   const c=2;
   public static $s = 4;

   public static function maMethode() {
    self::$s+=self::c;
    echo self::$s."\n";
   }
  }

  $i1 = new maClasse();
  $i2 = new maClasse();
  $i1->maMethode(); affiche 6
  maClasse::maMethode(); affiche 8
  $i2->maMethode(); affiche 10
  echo maClasse::$s.' '.maClasse::c; affiche 10 2
?>
```

Constructeur et destructeur

```
class maClasse {
        private $nom;

        function __construct($nom) {
                $this->nom = $nom;
        }

        function __destruct() {
                echo $this->nom."␣est␣mort.\n";
        }
}

$i1 = new maClasse("moi");
$i2 = new maClasse("toi");

$i1 = null;
```

affichage de 'moi est mort.'

affichage de 'toi est mort'

Interface

```php
interface monInterface1 {
        public function methode1();
        public function methode2($arg);
}

interface monInterface2 {
        public function methode3();
}


class maClasse implements monInterface1,
                          monInterface2 {
        public function methode1() {...}
        public function methode2($arg) {...}
        public function methode3() {...}
}
```

Héritage

```php
class maClasse {
        public $n;
        function __construct($n) { $this->n = $n; }
        function afficher() { echo $this->n."\n"; }
}

class maClasseHeritee extends maClasse {
        public $v;

        function __construct($n, $v=2) {
                parent::__construct($n);
                $this->v = $v;
        }

        function afficher() {
                echo $this->n." ".$this->v."\n";
        }
}
```

Héritage

```
$i1 = new maClasse("instance1");
$i2 = new maClasseHeritee("instance2",12);
$i1->afficher(); affiche 'instance1'
$i2->afficher(); affiche 'instance2 12'

var_dump($i1 instanceof maClasse); bool(true)
var_dump($i2 instanceof maClasse); bool(true)
var_dump($i1 instanceof maClasseHeritee); bool(false)
var_dump($i2 instanceof maClasseHeritee); bool(true)

var_dump($1);
```

Late Static Bindings (Résolution statique à la vol

Aix*Marseille université

```
class maClasse {
 function afficher1() { self::afficher2(); }
 static function afficher2() {
  echo "maClasse\n";
 }
}

class maClasseHeritee extends maClasse {
 static function afficher2() {
  echo "maClasseHeritee\n";
 }
}

$i1 = new maClasse();
$i2 = new maClasseHeritee();
$i1->afficher1(); maClasse
$i2->afficher1(); maClasse
```

Late Static Bindings (Résolution statique à la vol

Aix*Marseille université

```
class maClasse {
 function afficher1() { static::afficher2(); }
 static function afficher2() {
  echo "maClasse\n";
 }
}

class maClasseHeritee extends maClasse {
 static function afficher2() {
  echo "maClasseHeritee\n";
 }
}

$i1 = new maClasse();
$i2 = new maClasseHeritee();
$i1->afficher1(); maClasse
$i2->afficher1(); maClasseHeritee
```

Classe abstraite

```php
abstract class maClasseAbstraite {
        abstract public function getName();

        public function afficherNom() {
                echo $this->getName()."\n";
        }
}

class maClasse extends maClasseAbstraite {
        public function getName() { return "moi"; }
}

$i = new maClasse();
$i->afficherNom(); affiche 'moi'
```

Visibilité

- public : utilisable par n'importe quelle partie du programme.
- protected : utilisable uniquement par les classes et parents hérités.
- private : utilisable uniquement par la classe qui les a définis.

```
class maClasse {
        public $pub;
        protected $pro;
        private $pri;
        public function methodePublique() { ... }
        protected function methodeProtegee() { ... }
        private function methodePrivee() { ... }
}

class maClasseHeritee extends maClasse {
        public function test() {

         echo $this->pub; $this->methodePublique();
         echo $this->pro; $this->methodeProtegee();
         echo $this->pri; $this->methodePrivee();   interdit !
        }
}
```

Visibilité

- public : utilisable par n'importe quelle partie du programme.
- protected : utilisable uniquement par les classes et parents hérités.
- private : utilisable uniquement par la classe qui les a définis.

```
class maClasse {
        public $pub;
        protected $pro;
        private $pri;
        public function methodePublique() { ... }
        protected function methodeProtegee() { ... }
        private function methodePrivee() { ... }
}

class maClasse2 // qui n'étend pas maClasse {
        public function test() {
         $i = new maClasse();
         echo $i->pub; $i->methodePublique();
         echo $i->pro; $i->methodeProtegee(); interdit !
         echo $i->pri; $i->methodePrivee();   interdit !
        }
}
```

Clonage

```php
<?
  class maClasse {
          public $nom;

          function __construct($nom) {
                  $this->nom = $nom;
          }

          function __clone() {
                  $this->nom = "Clone de ".$this->nom;
          }
  }

  $i = new maClasse("moi");
  $c = clone $i;

  echo $i->nom."\n";    affiche 'moi'
  echo $c->nom."\n";    affiche 'Clone de moi'

?>
```

Méthodes magiques – Set et Get

```
class maClasse {
        private $props;

        public function __set($prop, $val) {
                echo "$prop <- $val\n";
                $this->props[$prop] = $val;
        }

        public function __get($prop) {
                if (!isset($this->props[$prop]))
                        return "erreur";
                else return $this->props[$prop];
        }
}
$i = new maClasse();
$i->toto = 2; affiche 'toto <- 2'
echo $i->toto."\n"; affiche '2'
echo $i->a."\n"; affiche 'erreur'
```

Méthodes magiques – Isset et Unset

```
class maClasse {
        private $props;

        avec les fonctions __set et __get définies sur le slide précédent

        public function __isset($prop) {
                return isset($this->props[$prop]);
        }

        public function __unset($prop) {
                echo "destruction de $prop\n";
                unset($this->props[$prop]);
        }
}
$i = new maClasse();
$i->toto = 2; affiche 'toto <- 2'
var_dump(isset($i->toto)); affiche 'bool(true)'
unset($i->toto); affiche 'destruction de toto'
```

Modèle-Vue-Contrôleur

Le **Modèle-Vue-Contrôleur** (MVC) est un méthode de conception utilisée pour organiser l'interface homme-machine (IHM) d'une application.

Le modèle : gère le comportement et les données de l'application.
↪ Exemple : gestion des interactions avec une base de données.

La vue : est une interface avec laquelle l'utilisateur interagit. Elle présente des parties du modèle à l'utilisateur et reçoit les actions de l'utilisateur.
↪ Exemple : code HTML/JavaScript présenté aux clients.

Le controleur : analyse les requêtes des clients, décide les actions à effectuer sur le modèle, et choisit les vues à envoyer aux clients.
↪ Exemple : analyse des informations passées dans l'URL

Projet MVC – Réalisation d'un site de sondages

Aix*Marseille université

Présentation du projet : Les utilisateurs postent des sondages constitués d'une question et de plusieurs réponses. Une fois le sondage posté, il est accessible aux visiteurs du site qui pourront voter pour une des réponses proposées. Les nombres de voix obtenues pour chaque réponse sont comptabilisés et affichés sous la forme d'un graphique. Tous les visiteurs peuvent chercher parmi les sondages et voter.

Fichier 'index.php'

```php
session_start();

function getActionByName($name) {
        $name .= 'Action'; require("actions/$name.inc.php");
        return new $name();
}

function getViewByName($name) {
        $name .= 'View'; require("views/$name.inc.php");
        return new $name();
}

function getAction() {
        if (!isset($_REQUEST['action'])) $action = 'Default';
        else $action = $_REQUEST['action'];
        $actions = array('Default', 'SignUp', 'Login', ...);
        if (!in_array($action, $actions)) $action = 'Default';
        return getActionByName($action);
}

$action = getAction(); $action->run(); $view = $action->getView();
$model = $action->getModel(); $view->run($model);
```

Les actions du projet

Les actions possibles d'un visiteur :

- SignUpForm : affichage du formulaire d'inscription
- SignUp : demande d'inscription
- Login : connexion du visiteur
- Logout : déconnexion du visiteur
- UpdateUserForm : affichage du formulaire de modification de profil
- UpdateUser : modification du profil
- AddSurveyForm : affichage du formulaire d'ajout de sondage
- AddSurvey : ajout d'un sondage
- GetMySurveys : affichage des sondages du visiteur
- Search : recherche
- Vote : prise en compte d'un vote

La classe Action

Aix*Marseille université

Toutes les actions sont définies en étendant la classe suivante :

```
abstract class Action {
        private $view;
        private $model;
        protected $database;

        public function __construct(){ ... }

        protected function setView($view) { ... }
        protected function setModel($model) { ... }
        protected function getSessionLogin() { ... }
        protected function setSessionLogin($login) { ... }

        public function getView() { ... }
        public function getModel() { ... }

        abstract public function run();
}
```

Exemple : l'action SignUpForm

```
class SignUpFormAction extends Action {

   /**
    * Dirige l'utilisateur vers le formulaire d'inscription.
    *
    * @see Action::run()
    */
    public function run() {
       $this->setModel(new MessageModel());
       $this->getModel()->setLogin($this->getSessionLogin());
       $this->setView(getViewByName("SignUpForm"));
    }

}
```

L'action commence par construire le modèle (nous verrons par la suite ce que contient les instances de la classe *MessageModel*). Ensuite, elle détermine la vue qui va être utilisée pour afficher le modèle.

Le modèle

Le modèle contient les classes permettant de manipuler les objets “métiers” du site (ici, les sondages) et de modifier la base de données. Aucune fonctionnalité de représentation des données n’est fournie par le modèle.

La classe permettant de représenter un sondage est donnée ci-dessous :

```
class Survey {
        private $id;
        private $owner;
        private $question;
        private $responses;

        public function __construct($owner, $question) { ... }
        public function setId($id) { ... }
        public function addResponse($response) { ... }
        public function getId() { ... }
        public function getOwner() { ... }
        public function getQuestion() { ... }
        public function getResponses() { ... }
        public function computePercentages() { ... }
}
```

Le modèle

Aix*Marseille université

Vous avez également une classe représentant une réponse :

```
class Response {
  private $id;
  private $survey;
  private $title;
  private $count;
  private $percentage;

  public function __construct($survey, $title, $count = 0) { ... }
  public function setId($id) { ... }
  public function computePercentage($total) { ... }
  public function getId() { ... }
  public function getSurvey() { ... }
  public function getTitle() { ... }
  public function getCount() { ... }
  public function getPercentage() { ... }
}
```

Les interaction avec la base de données se feront via une instance de classe *Dababase* (créée dans le constructeur de classe *Action*). Nous détaillerons les fonctionnalités de cette classe plus tard.

La classe Model

Toutes les extensions de la classe *Model* peuvent être passée à des vues pour être affichée. La classe *Model* contient les informations de base nécessaire à l'affichage de la page :

```
class Model {
        private $login;
        private $loginError;

        public function __construct() { ... }
        public function getLogin() { ... }
        public function setLogin($login) { ... }
        public function getLoginError() { ... }
        public function setLoginError($loginError) { ... }
}
```

Dans le projet, nous avons deux classes qui étendent *Model* :

- MessageModel : contient un message à afficher.
- SurveysModel : contient une liste de sondages à afficher.

La classe View

Les différentes vues du projet sont définies en étendant la classe *View* :

```
abstract class View {
        public function run($model) {
                $login = $model->getLogin();
                require("templates/page.inc.php");
        }

        private function displayLoginForm($model) { ... }
        private function displayLogoutForm($model) { ... }
        private function displayCommands($model) { ... }
        private function displaySearchForm($model) { ... }

        protected abstract function displayBody($model);

}
```

Les vues définissent la méthode *displayBody* afin de générer le contenu de la page (en conservant les bordures, le formulaire de connexion, etc.).

La génération de la page – page.inc.php

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>...</head>
<? $login = $model->getLogin(); ?>
<body>
  <div class="header">
    <div class="title">Site de sondages</div>
      <div class="loginForm"><?
          if ($login===null) $this->displayLoginForm($model);
          else $this->displayLogoutForm($model);
      ?></div>
    </div>
...
    <div class="content">
      <? $this->displayBody($model); ?>
    </div>
...
</body>
</html>
```

Les vues

Les différentes vues du projet :

- DefaultView : page vide
- MessageView : affiche un message à l'utilisateur
- SignUpFormView : formulaire d'inscription
- UpdateUserFormView : formulaire de modification de mot de passe
- AddSurveyFormView : formulaire d'ajout de sondage
- SurveysView : liste de sondages

Remarques :

- *DefaultView* utilise un modèle de type *Model*.
- *MessageView*, *SignUpFormView*, *UpdateUserFormView*, *AddSurveyFormView* utilisent un modèle de type *MessageModel*.
- *SurveysView* utilise un modèle de type *SurveysModel*.

Exemple : la vue SurveysView

La vue suivante permet d'afficher une liste de sondages :

```
class SurveysView extends View {

  public function displayBody($model) {
    if (count($model->getSurveys())===0) {
      echo "Aucun sondage ne correspond a votre recherche.";
      return;
    }

    foreach ($model->getSurveys() as $survey) {
      $survey->computePercentages();
      require("templates/survey.inc.php");
    }
  }

}
```

Elle inclut le fichier *survey.inc.php* pour générer le code HTML représentant le sondage contenu dans la variable `$survey`.

Exemple : déroulement d'une inscription

- Le client demande la page *index.php* ;
- Le serveur affiche la vue par défault (vide) ;
- Le client clique sur *Inscrivez-vous !*
- Le navigateur demande *index.php ?action=SignUpForm* ;
- Le serveur affiche la vue *SignUpFormView* ;
- L'utilisateur remplit le formulaire et l'envoie ;
- Le navigateur poste le formulaire vers *index.php ?action=SignUp* ;
- Le serveur effectue l'inscription si aucune erreur ne s'est produite ;
- Le serveur affiche la vue *MessageView* (avec un message de confirmation) ou *SignUpFormView* (avec un message d'erreur) ;
- ...

SQLite et PDO

SQLite écrit les données relatives à la base dans un unique fichier sur le disque dur du serveur. La commande **sqlite3** permet d'ouvrir la base et effectuer des requêtes SQL (pratique pour débugger).

PDO (*PHP Data Objects*) est une interface pour accéder à une base de données depuis PHP. Elle gère la connexion, l'envoie des requêtes, la déconnexion à la base de données. Elle permet de changer plus facilement de moteur de bases de données (Par exemple, SQLite → MySQL).

Ouverture de la base :

```
<?
try {
 $bd = new PDO("sqlite:database.sqlite");
} catch (PDOException $e) {
 die('Erreur : ' . $e->getMessage());
}
?>
```

SQLite et PDO

Aix*Marseille université

Une fois l'instance de PDO construite, vous effectuez des requêtes avec :

- `$bd->exec($requete)` : pour modifier la base de données
- `$bd->query($requete)` : pour extraire des données de la base

Exemples :

```
$bd->exec("CREATE TABLE users (".
          " nickname char(20),".
          " password char(50)".
          ");");

$res =$bd->exec('UPDATE users SET password="'.
          md5($password).'" WHERE nickname="'.$nickname.'"; ');
echo "nombre de lignes modifiees = $res";

$res = $bd->query("select nickname from users;");
```

Injections SQL

Le code suivant :

```
$nickname = 'aa"; DELETE FROM users; '.
                'SELECT * FROM users WHERE nickname="';
$password = "truc";
$res = $bd->exec( 'UPDATE users SET password="'.
          md5($password).'" WHERE nickname="'.$nickname.'"; ');
```

exécute la requête SQL suivante :

```
UPDATE users SET password="...." WHERE nickname="aa";
DELETE FROM users;
SELECT * FROM users WHERE nickname="";
```

Protection contre les injections SQL :

```
$r = $bd->prepare( 'UPDATE users SET password = ? '.
                   'WHERE nickname = ?' );
$r->execute(array(md5($password), $nickname));
```

SQLite et PDO

Aix*Marseille université

Pour faire une requête SQL :

```
$res = $bd->query("select * from sondages");
var_dump($res);
```

affiche 'object(PDOStatement)#2 (1) {
["queryString"]=> string(19) "select * from sondages" }'

Pour connaître le nombre de lignes :

```
echo "nombre de lignes : ".$res->rowCount()."\n";
```

Pour parcourir les lignes :

```
$res = $bd->query("select * from sondages");
while ($ligne = $res->fetch()) {
  echo $ligne['createur'].
       " pose la question : ".
       $ligne['question']."\n";
}
```

SQLite et PDO

Aix*Marseille université

Pour mettre toutes les lignes dans un tableau :

```
$res = $bd->query("select * from sondages");
$lignes = $res->fetchAll();
foreach ($lignes as $ligne) {
 echo $ligne['createur'].
      " pose la question : ".
      $ligne['question']."\n";
}
```

Voir aussi, dans la classe PDOStatement, les méthodes :

- bindColumn : attache une variable à une colonne
- errorInfo : information d'erreur
- fetchColumn : récupère la valeur dans une colonne donnée
- closeCursor : ferme le curseur

SQLite et PDO

Aix*Marseille université

Voir aussi, dans la classe PDO, les méthodes :

- beginTransaction : démarre une transaction
- commit : valide une transaction
- rollback : annule une transaction
- errorInfo : Retourne les informations associées à l'erreur
- errorCode : Retourne le SQLSTATE associé avec la dernière opération
- prepare : Prépare une requête à l'exécution et retourne un objet
- quote : Protège une chaîne pour l'utiliser dans une requête SQL PDO

```
$string = 'Chaine \' particuliere';
print "non echappee : $string\n";
print "echappee :" . $bd->quote($string) . "\n";
```

Chaine non echappee : Chaine ' particuliere
Chaine echappee : 'Chaine '' particuliere'

Base de données du projet

Aix*Marseille université

Les trois tables utilisées dans le projet :

```
users(nickname char(20), password char(50));

surveys(id integer primary key autoincrement,
        owner char(20), question char(255));

responses(id integer primary key autoincrement,
          id_survey integer,
          title char(255),
          count integer);
```

Exemple : sauvegarde d'un sondage

Aix*Marseille université

```
public function saveSurvey(&$survey) {
  if ($survey->getId()!==null) return false;
  $this->connection->beginTransaction();
  $r = $this->connection->prepare('INSERT INTO surveys(owner, '.
                                  'question) VALUES (?,?);');
  if ($r===false) {
    $this->connection->rollback(); return false;
  }
  if ($r->execute(array($survey->getOwner(),
                        $survey->getQuestion()))===false) {
    $this->connection->rollback(); return false;
  }
  $id = $this->connection->lastInsertId();
  $survey->setId($id);
  foreach ($survey->getResponses() as $response)
    if(!$this->saveResponse($response)) {
      $this->connection->rollback();
      return false;
    }
  $this->connection->commit();
  return true;
}
```

La classe Database

Aix*Marseille université

```
class Database {

  private $connection;

  public function __construct() { ... }

  public function checkPassword($nickname, $password) { ... }
  public function addUser($nickname, $password) { ... }
  public function updateUser($nickname, $password) { ... }
  public function saveSurvey(&$survey) { ... }
  public function loadSurveysByOwner($owner) { ... }
  public function loadSurveysByKeyword($keyword) { ... }
  public function vote($id) { ... }

  private function createDataBase() { ... }
  private function checkNicknameValidity($nickname) { ... }
  private function checkPasswordValidity($password) { ... }
  private function checkNicknameAvailability($nickname) { ... }
  private function saveResponse(&$response) { ... }
  private function loadSurveys($arraySurveys) { ... }
  private function loadResponses(&$survey, $arrayResponses) { ... }
}
```

Persistance et mapping objet-relationnel (ORM)

Aix*Marseille université

Les besoins :

- Sauvegarde simple des objets en base de données :

```
$user = new User();
$user->nickname = "bob";
$user->password = md5("truc");
$user->save();
```

- Création automatique des tables ;
- Chargement des objets ;
- Gestion des relations entre les objets/enregistrements ;
- ...

Quelques solutions en PHP :

- Propel
- Doctrine
- Zend Framework
- Redbean
- ...

Rappels – HTML

Le **HTML** est un langage qui permet de décrire une page Web.

```
<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
 <head>
  <title>
   Exemple
  </title>
 </head>
 <body>
  izgz hzegizihzi zihzg zeighze <a href="toto.html">toto</a>
  <b>aaaa</b>
 </body>
</html>
```

Rappels – CSS

Le **Cascading Style Sheets** (**CSS**) est un langage qui sert à décrire la présentation des documents HTML et XML.

```
body{
   overflow-y:scroll;
   background-color: #FFFFFF;
   white-space: nowrap;
   text-align:center;
}

.main {
      text-align:left;
      white-space: normal;
      display:inline-block;
      color: #000000;
      background-color: #FFFFFF;
      border: 0px;
}
```

Rappels – JavaScript

Aix*Marseille université

Le **JavaScript** est le langage principalement utilisé dans les pages Web :

```
function addPointToChart(time, id, value) {
  if (data.length <= id)
        data.push({ label : "objective "+(id+1), data : [] });
  if (data[id].data.length >= 10)
        data[id].data = data[id].data.slice(1);
  data[id].data.push([time, value]);
}

function analyseMessageForChart(msg) {
  var expression = /([0-9]*) sec.*obj = \((.*)\)/;
  var res = msg.match(expression);
  if (res!=null) {
        var time = parseInt(res[1]);
        var points = res[2].split(", ");
        for(var i=0;i<points.length;i++)
              addPointToChart(time,i,parseInt(points[i]));
        updateChart();
  }
}
```

Rappels – DOM

Le **Document Object Model** (**DOM**) est une recommandation du W3C qui décrit une interface permettant à des programmes d'accéder ou de mettre à jour le contenu, la structure ou le style de documents XML.

```
<script type="text/javascript">
   document.firstChild.firstChild.lastChild;
   document.firstChild.childNodes[0].lastChild;
   document.firstChild.childNodes[0].childNodes[1];
   var x = document.getElementsByTagName('P')[0].lastChild;
   document.getElementById('a')
           .firstChild.nodeValue='bold bit of text';
   var x = document.createElement('br');
   document.getElementById('e').appendChild(x);
   var x = document.createTextNode('exemple');
   document.getElementById('b').appendChild(x);
   ...
</script>
```

jQuery

jQuery est une bibliothèque JavaScript libre qui permet de simplifier la programmation en JavaScript (AJAX et interaction avec HTML).

Elle est disponible sous Licence MIT, GNU GPL.

"Installation" :

- Télécharger le code JavaScript sur le site de JQuery ;
- Inclure le code suivant dans l'en-tête de votre fichier HTML :

```
<script type="text/javascript" src="jquery-xxx.min.js">
</script>
```

Exemple d'utilisation :

```
<script type="text/javascript">
$(function() {
    $("a").click(function() { alert("Hello world!"); });
});
</script>
```

jQuery – Sélectionner des éléments

Aix*Marseille université

Sélectionner un élément par son identifiant :

```
$("#id_element").addClass("maClasseCss");
```

Sélectionner un élément par sa classe :

```
$(".classeCss").click(function() { alert("coucou"); });
```

Sélectionner des elements contenus dans un autre élément :

```
$("#list > li").addClass("blue"); ou
$("#list").find("li").addClass("blue");
```

Tout sélectionner :

```
$(*).addClass("blue");
```

Appliquer une fonction sur un ensemble d'éléments :

```
$("#list").find("li").each(function(i) {
      $(this).append( "Je suis le numero " + i);
});
```

jQuery – Modifier les attributs et les propriétés

Aix*Marseille université

Les attributs (valeurs initiales présentes dans le HTML) :

```
$("#id_element").attr("title","toto");
a = $("#id_element").attr("title");
$("#id_element").removeAttr("title");
```

Les propriétés (valeurs courantes dans le DOM) :

```
$("#id_element").prop("checked",true);
a = $("#id_element").prop("checked");
$("#id_element").removeProp("toto");
```

La valeur :

```
$("#id_element").val("coucou");
v = $("#id_element").val();
```

jQuery – Parcourir les éléments

Aix*Marseille université

Appliquer une fonction sur un ensemble d'éléments :

```
$("#list").find("li").each(function(i) {
      $(this).append( "Je suis le numero " + i);
});
```

Récupérer la liste des fils d'un élément :

```
$('#list').children().css('background-color', 'red');
```

Récupérer le premier élément :

```
$('#list').children().first().css('background-color', 'red');
```

Récupérer le dernier élément :

```
$('#list').children().last().css('background-color', 'red');
```

Filtrer les éléments :

```
$('#list').children().filter(function(i) {return i%3== 0;})
          .css('background-color', 'red');
```

(Voir les autres fonctionnalités sur le site de jQuery)

jQuery – Ajouter, supprimer des éléments

Aix*Marseille université

Modifier et récupérer le contenu des éléments :

```
$("#list").find("li").html("<b>toto</b>");
$("#list").find("li").text("<b>toto</b>");
c_html = $("#list").find("li").html();
c_text = $("#list").find("li").text();
```

Ajouter dans des éléments :

```
$("#list").find("li").append("toto");
```

Supprimer un ensemble d'éléments du DOM :

```
$("#list").find("li").remove();
```

Insérer avant et après des éléments :

```
$('#test').children().before("<li>avant chaque element</li>");
$('#test').children().after("<li>apres chaque element</li>");
```

(Voir les autres fonctionnalités sur le site de jQuery)

jQuery – CSS

Aix*Marseille université

Les classes *css* :

```
r = $("#id_element").hasClass("maClasseCss");
$("#id_element").removeClass("maClasseCss");
$("#id_element").toggleClass("maClasseCss");
```

Jouer avec le style :

```
$("#id_element").css('background-color', 'red');
color = $("#id_element").css('background-color');
```

Jouer avec la hauteur et la largeur :

```
$("#id_element").height(100);
h = $("#id_element").height();
$("#id_element").width(100);
w = $("#id_element").width();
```

Connaître la position d'un élément :

```
p = $("#id_element").offset(); (par rapport au document)
alert(p.left+" "+p.top);
p = $("#id_element").position(); (par rapport au parent)
alert(p.left+" "+p.top);
```

jQuery – Les événements

Aix*Marseille université

La souris :

```
$("#id_element").click(function() {  alert("click."); });
$("#id_element").mouseup(function(){$(this).append('up ');}).
                 .mousedown(function(){$(this).append('down ');});
```

Le clavier :

```
$('#input_text').keydown(function(event) {
   if (event.keyCode == '13') {
      alert("ok");
    }
});
```

Les changements :

```
$("#id_element").change(function() {  alert("change."); });
```

(Voir les autres fonctionnalités sur le site de jQuery)

jQuery – Les autres fonctionnalités

- Les effets et animations
- Des outils pour simplifier la programmation en JavaScript
- JQuery UI :
  - Librairie de widgets
  - Interaction (Drag & drop, tri, etc).
  - Thèmes
  - Effets

Ajax (Asynchronous JavaScript and XML)

Ajax est un acronyme pour Asynchronous JavaScript and XML.

Il est un ensemble de technologies libres couramment utilisées sur le Web :

- **HTML** (ou XHTML) pour la structure sémantique des informations ;
- **CSS** pour la présentation des informations ;
- **DOM** et **JavaScript** pour afficher et interagir dynamiquement avec l'information présentée ;
- l'objet **XMLHttpRequest** pour échanger les données de manière asynchrone avec le serveur Web ;
- **XML** (parfois les fichiers texte ou JSON) pour le format des données informatives.

XMLHttpRequest : Création

Fonction Javascript pour créer un objet XMLHttpRequest :

```
function createXhrObject()
{
    if (window.XMLHttpRequest)
        return new XMLHttpRequest();

    if (window.ActiveXObject) {
        var names = [
            "Msxml2.XMLHTTP.6.0",
            "Msxml2.XMLHTTP.3.0",
            "Msxml2.XMLHTTP",
            "Microsoft.XMLHTTP"
        ];
        for(var i in names) {
            try{ return new ActiveXObject(names[i]); }
            catch(e){}
        }
    }
    window.alert("pas de XMLHTTPRequest.");
    return null;
}
```

XMLHttpRequest : Les méthodes et propriétés

Aix*Marseille université

```
interface XMLHttpRequest {

  // requête
  void open(DOMString method, DOMString url);
  void open(DOMString method, DOMString url, boolean async);
  ...
  void setRequestHeader(DOMString header, DOMString value);
  void send();
  void send(Document data);
  ...
  void abort();

  // réponse
  readonly attribute unsigned short status;
  readonly attribute DOMString statusText;
  DOMString getResponseHeader(DOMString header);
  DOMString getAllResponseHeaders();
  readonly attribute DOMString responseText;
  readonly attribute Document responseXML;

  ...
}
```

XMLHttpRequest : Les méthodes et propriétés

Aix*Marseille université

```
interface XMLHttpRequest {

  ...

  // gestionnaire d'évènement
  attribute Function onreadystatechange;

  // états
  const unsigned short UNSENT = 0;
  const unsigned short OPENED = 1;
  const unsigned short HEADERS_RECEIVED = 2;
  const unsigned short LOADING = 3;
  const unsigned short DONE = 4;
  readonly attribute unsigned short readyState;

};
```

XMLHttpRequest : Les états

Aix*Marseille université

Les différents états de XMLHttpRequest :

- `UNSENT` (0) : L'objet a été construit.
- `OPENED` (1) : La méthode `open` a été invoquée avec succès. Le header de la requête peut être modifié avec la méthode `setRequestHeader`.
- `HEADERS_RECEIVED` (2) : Le header HTTP de la réponse a été reçu.
- `LOADING` (3) : Le corps de la réponse est en cours de téléchargement.
- `DONE` (4) : Le transfert des données est terminé (ou erreur !).

XMLHttpRequest : Requête asynchrone GET

Exemple de requête GET :

```
xhr = createXhrObject();

xhr.onreadystatechange  = function() {
    if(xhr.readyState  == 4)
    {
        if(xhr.status  == 200) {
            alert(xhr.responseText);
        }
        else {
            alert("Error : "+xhr.status);
        }
    }
};

xhr.open("GET", "partie.php?a=12&b=13",  true);
xhr.send(null);
```

XMLHttpRequest : Requête asynchrone POST

Aix*Marseille université

Exemple de requête POST :

```
xhr = createXhrObject();

xhr.onreadystatechange = function() {
    if(xhr.readyState == 4)
    {
        if(xhr.status == 200) {
            alert(xhr.responseText);
        }
        else {
            alert("Error : "+xhr.status);
        }
    }
};

xhr.open("POST", "partie.php", true);
xhr.setRequestHeader("Content-Type",
        "application/x-www-form-urlencoded");
xhr.send("a=12&b=13");
```

AJAX et jQuery

Exemple de requête POST :

```
$.ajax({
   type: "POST",
   url: "toto.php",
   data: {numero : "123", nom : "bob"}
}).done(function( msg ) { alert( msg ); })
   .fail(function() { alert("erreur"); })
   .always(function() { alert("fini"); });
```

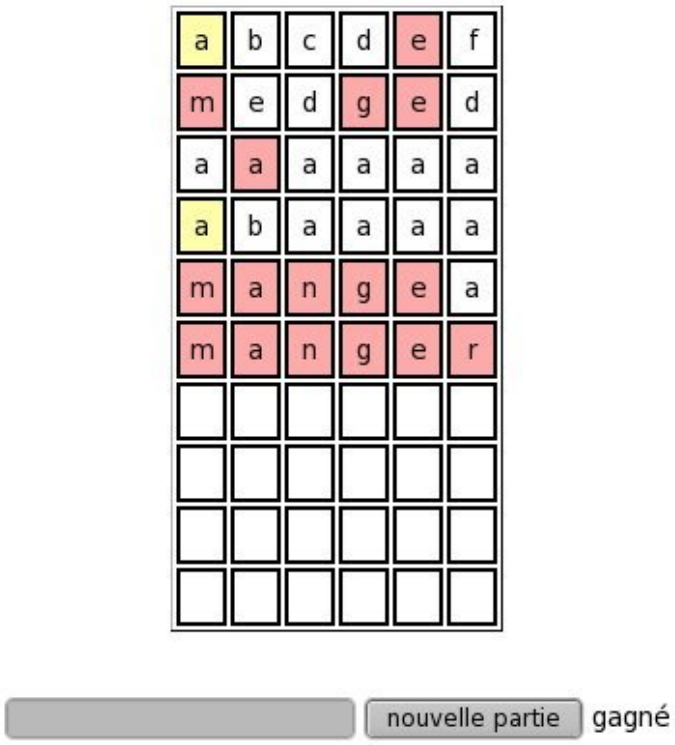
Exemple de requête GET :

```
$.get("test.php", { 'choices[]': ["Jon", "Susan"]} );
```

Projet AJAX

Présentation du projet : Réalisation d'un jeu client/serveur en Ajax.

Le "plateau" de jeu est constitué d'une grille de 6 colonnes et 10 lignes :



Projet AJAX

Aix*Marseille université

Déroulement de la partie :

- La grille est vide et le serveur choisit un mot à faire deviner au joueur.
- Le joueur tape un mot de 6 lettres dans la zone de texte puis appuie sur la touche entrée pour soumettre le mot au serveur.
- Le serveur associe alors une couleur à chaque lettre du mot :
  - La couleur rouge est associée aux lettres bien placées.
  - La couleur jaune est associée aux lettres mal placées mais présentes dans le mot à trouver.
- Les couleurs associées à chaque lettre sont envoyées au client.
- La partie s'arrête une des deux conditions est vérifiées :
  - Le joueur a trouvé le mot $\rightarrow$ gagné ;
  - Le joueur a rempli toutes les lignes de la grille $\rightarrow$ perdu ;

Projet AJAX

Aix*Marseille université

Travail à réaliser : Réaliser ce jeu en utilisant la technologie Ajax.

- Le client (écrit en JavaScript et HTML) doit être capable de :
  - afficher la grille
  - afficher les réponses du serveur
  - envoyer des requêtes au serveur
- Le serveur (écrit en PHP) doit être capable de :
  - répondre au client
  - suivre la partie en utilisant le mécanisme des sessions.

Projet AJAX

Le serveur répond à deux types de requêtes :

- **Action d'initialisation** : demande du client au serveur de choisir un nouveau mot et de lui communiquer la taille de la grille via l'URL `"jeu.php?action=init"`.
- **Action de jeu** : Le client soumet un mot au serveur via l'URL `"jeu.php?action=jeu&mot=lemotsoumis"`.

A la suite de chacune des requêtes, le serveur envoie une réponse au client. Cette réponse doit être **facilement interprétable** par le code JavaScript qui est exécuté sur le client.

JSON

Aix*Marseille université

Pour faire transiter les données du serveur vers le client, nous allons utiliser le format de données textuel JSON (JavaScript Oject Notation) :

```
{"machin": {
  "taille": "12",
  "style": "gras",
  "bidule": {
    "machin": [
      {"style" : "italique" },
      {"style" : "gras" }
    ]
  }
}}
```

L'avantage de JSON est qu'il est reconnu nativement par JavaScript (mais le langage XML aurait également pu être utilisé)

JSON

Aix*Marseille université

Les éléments en JSON :

- Les objets : `{chaîne : valeur, chaîne : valeur...}`
- Les tableaux : `[valeur, valeur, ...]`
- Les valeurs : chaîne, nombre, objet, tableau, true, false, null
- Les chaînes : `"abcdef"` ou `"abcd\n\t"`
- Les nombres : `-1234.12`

```
{"unObjet": {
  "unTableau": [12, 13, 53],
  "unNombre" : 53,
  "unChaîne" : "truc\n"
  "unObjet" : { "style" : "gras" }
}}
```

JSON, AJAX et PHP

Aix*Marseille université

Côté serveur :

```
header('Content-type: application/json');
$a = array('init', 10, 6);
echo json_encode($a);
```

Côté client :

```
var reponse = eval('(' + xhr.responseText + ')');
if (reponse[0]=='init')
   construireGrille(reponse[1], reponse[2]);
```

jQuery, AJAX, JSON et PHP

Aix*Marseille université

Côté serveur :

```
header('Content-type: application/json');
$a = array('nom'=>$_POST['nom'],
           'maj'=>strtoupper($_POST['nom']));
echo json_encode($a);
```

Côté client :

```
$.ajax({
  type: "post",
  url: "a.php",
  data: { nom : "bob" }
}).done(function( msg ) {
  alert( msg['nom']+"=>"+msg['maj'] );
});
```

Projet AJAX

Exemple de protocole de communication du serveur vers le client :

- `["init", 6, 10]`
- `["ligne", "l", "#FFAAAA", "a", "#FFFFAA", ...]`
- `["perdu", "l", "#FFFFFF", "a", "#FFFFFF", ...]`
- `["gagne", "l", "#FFAAAA", "a", "#FFAAAA", ...]`

Le serveur ne doit jamais envoyé le mot à trouver au client !

Projet AJAX

Aix*Marseille université

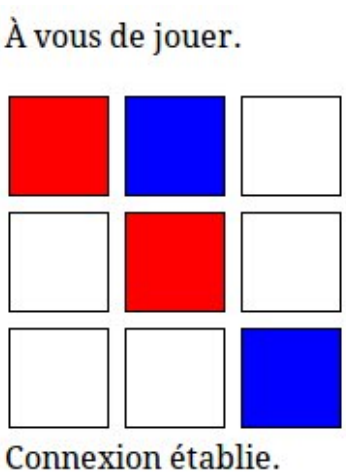
Améliorations possibles :

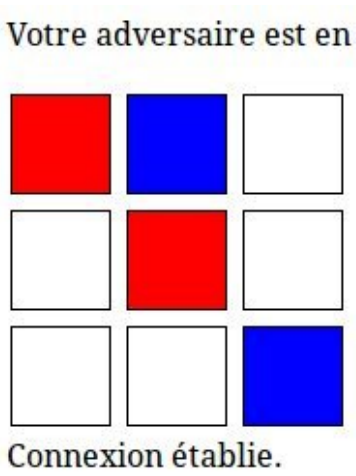
- Comptes utilisateurs
- Scores (temps pour résoudre dix mots par exemple)
- Classements
- Vérification des mots dans un dictionnaire
- Jeux à plusieurs

Problématique

Problématique : Nous souhaitons réaliser un jeu de morpion en réseau.

- Les clients se connectent au jeu ;
- Les clients jouent chacun leur tour ;
- Les coups d'un joueur sont répercutés sur la grille de l'autre joueur ;
- Le serveur organise la partie (débute la partie, décide le gagnant...) ;





Problématique

Problème : En PHP, chaque requête du client est traitée indépendamment.

Il n'y a donc pas de :

- Processus persistant et de donnée en mémoire persistante
  ⇒ Difficulté pour conserver l'état courant de la partie
  (*i.e.* orchestrer plusieurs clients)
- Connexion persistante (le client demande et le serveur répond)
  ⇒ Difficulté pour envoyer des événements aux clients.

Solution :

- Côté client : WebSocket de HTML 5
- Côté serveur : Java et Jetty (serveur HTTP et moteur de servlet libre)

Autres solutions :

- AJAX et l'approche Comet
- Socket.IO (développement du client et du serveur en JavaScript)
- ...

WebSocket

Aix*Marseille université

Objectif (Wikipedia) : Obtenir un canal de communication bidirectionnel et full-duplex sur un socket TCP pour les navigateurs et les serveurs web.

Demande de connexion du client et "handshake" :

```
GET /ws HTTP/1.1
Host: pmx
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Version: 6
Sec-WebSocket-Origin: http://pmx
Sec-WebSocket-Extensions: deflate-stream
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
```

Réponse du serveur :

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
```

Client – WebSocket et JavaScript

Aix*Marseille université

Demande d'ouverture de la connexion avec le serveur :

```
if ('WebSocket' in window)
        ws = new WebSocket('ws://toto.com:8081');
else if ('MozWebSocket' in window)
        ws = new MozWebSocket('ws://toto.com:8081');
else alert("Pas de WebSocket");
```

Mise en place des "callbacks" :

```
ws.onopen = onOpen;
ws.onclose = onClose;
ws.onerror = onError;
ws.onmessage = onMessage;
```

Exemples de "callbacks" :

```
function onOpen(event) { alert("Connexion etablie."); }
function onClose(event) { alert("Connexion perdue."); }
function onError(event) { alert("Erreur"); }
function onMessage(event) { alert(event.data); }
```

Client – WebSocket et JavaScript

Aix*Marseille université

Envoie des messages vers le serveur :

```
for (var i = 0; i < 3; i++) {
  for (var j = 0; j < 3; j++) {
    $('#grid').append(
        '<div id="c'+i+""+j+'" class="case"></div>');
  }
  $('#grid').append("<br/>");
}

$('.case').click(function() {
  var id = $(this).attr('id');
  var r = parseInt(id.charAt(1));
  var c = parseInt(id.charAt(2));
  ws.send("P#"+r+"#"+c);
});
```

Serveur – Jetty - Simple serveur HTTP

Aix*Marseille université

Création d'un serveur HTTP avec Jetty :

```
public static void main(String[] args) throws Exception {
    Server httpServer = new Server(8080);
    ResourceHandler resourceHandler = new ResourceHandler();
    resourceHandler.setDirectoriesListed(true);
    resourceHandler.setWelcomeFiles(new String[]
                                        { "index.html" });
    resourceHandler.setResourceBase("/home/toto/client");
    httpServer.setHandler(resourceHandler);
    httpServer.start();
    httpServer.join();
}
```

Serveur – Jetty - Serveur WebSocket

Aix*Marseille université

Création d'un serveur WebSocket avec Jetty :

```
public static void main(String[] args) throws Exception {
   Server wsServer = new Server(8081);
   wsServer.setHandler(new MyServer());
   wsServer.start();
   wsServer.join();
}
```

La classe qui reçoit les notifications de connexion :

```
public class MyServer extends WebSocketHandler {

   public WebSocket doWebSocketConnect(
                              HttpServletRequest request,
                              String protocol) {
      Client client = new MyClient(this);
      return client;
   }
}
```

Serveur – Jetty - Serveur WebSocket

Aix*Marseille université

Gestion d'une connexion entre un client et le serveur :

```
public class MyClient implements WebSocket.OnTextMessage {
  private Connection connection;

  public void onOpen(Connection connection) {
    this.connection = connection;
    //TODO
  }

  public void onMessage(String data) { //TODO }

  public void onClose(int code, String msg) { // TODO }

  public void send(String data) {
    try { connection.sendMessage(data); }
    catch (IOException e) { connection.close(); }
  }
}
```

Jeu de morpion – Protocole

Aix*Marseille université

Les messages du serveur :

- W : Attendre le début de la partie.
- P : Vous devez jouer un coup.
- O : Votre adversaire est en train de jouer.
- V : Vous avez gagné.
- L : Vous avez perdu.
- D#r#c#j : Le joueur j à jouer la case (r, c).

Les messages du client :

- P#r#c : Je joue la case (r, c).

Jeu de morpion – Client

Aix*Marseille université

La page HTML du client :

```
<html>
  <head>
    <script type="text/javascript" src="jquery.js">
    </script>
    <script type="text/javascript" src="client.js">
    </script>
     ...
  </head>
  <body onload="init()">
    <div id="gameMessage" class="message"></div><br/>
    <div id="grid"></div>
    <div id="connectionMessage" class="message"></div>
  </body>
</html>
```

Jeu de morpion – Client

Aix*Marseille université

La fonction *init()* :

```
function init() {
  if ('WebSocket' in window)
    ws = new WebSocket('ws://localhost:8081');
  else if ('MozWebSocket' in window)
    ws = new MozWebSocket('ws://localhost:8081');

  ws.onopen = onOpen;
  ws.onmessage = onMessage;
  ws.onclose = onClose;

  // Code pour creer la grille.

  // Code pour ecouter les clics de souris.
}
```

Jeu de morpion – Client

Aix*Marseille université

Code pour créer la grille :

```
var i,j;
for (i = 0; i < 3; i++) {
  for (j = 0; j < 3; j++)
    $('#grid').append(
      '<div id="c'+i+""+j+'" class="case"></div>');
  $('#grid').append("<br/>");
}
```

Code pour écouter les clics de souris :

```
$('.case').click(function() {
  var id = $(this).attr('id');
  var r = parseInt(id.charAt(1));
  var c = parseInt(id.charAt(2));
   ws.send("P#"+r+"#"+c);
});
```

Jeu de morpion – Client

Traitement des messages du serveur :

```
function onMessage(event) {
  switch (event.data.charAt(0)) {
    case 'W' : setGameMessage("Attendre."); break;
    case 'P' : setGameMessage("Jouer."); break;
    case 'O' : setGameMessage("Votre adv. joue."); break;
    case 'V' : setGameMessage("Victoire."); break;
    case 'L' : setGameMessage("Perdu."); break;
    case 'D' : var r = parseInt(event.data.charAt(2));
               var c = parseInt(event.data.charAt(4));
               var p = parseInt(event.data.charAt(6));
               if (p==1) $("#c"+r+""+c).addClass("red");
               else $("#c"+r+""+c).addClass("blue");
               break;
  }
}

function setGameMessage(m) { $('#gameMessage').html(m); }
```

Jeu de morpion – Client

Traitement des connexions et déconnexions :

```
function onOpen() {
  $('#connectionMessage').html("Connexion etablie.");
}

function onClose() {
  $('#connectionMessage').html("Connexion perdue.");
}
```

Jeu de morpion – Serveur

```
public class Main {
  public static void main(String[] args) throws Exception {
    Server wsServer = new Server(8081);
    wsServer.setHandler(new MorpionServer());
    wsServer.start();

    Server htmlServer = new Server(8080);
    ResourceHandler rHandler = new ResourceHandler();
    rHandler.setDirectoriesListed(true);
    rHandler.setWelcomeFiles(new String[] {"index.html"});
    rHandler.setResourceBase("client");
    htmlServer.setHandler(rHandler);
    htmlServer.start();

    wsServer.join();
    htmlServer.join();
  }
}
```

Jeu de morpion – Serveur

```
public class MorpionServer extends WebSocketHandler {

  private Game game = new Game();

  public WebSocket doWebSocketConnect(
      HttpServletRequest request,
      String protocol) {
    Client client = new Client(this);
    return client;
  }

  public void addPlayer(Client client) {
    game.addPlayer(client);
    if (game.isComplete()) {
      game.start();
      game = new Game();
    }
  }
}
```

Jeu de morpion – Serveur

```
public class Client implements WebSocket.OnTextMessage {
  private Connection connection;
  private MorpionServer server;
  private Game game;
  private int position;

  public Client(MorpionServer server) {
    this.server = server;
  }

  public void onOpen(Connection connection) {
    this.connection = connection;
    server.addPlayer(this);
  }

  public void onMessage(String data) {
    game.onMessage(position, data);
  }
```

Jeu de morpion – Serveur

(Suite de la classe *Client*)

```
    public void onClose(int closeCode, String message) {
        game.finish();
    }

    public void setGame(Game game, int position) {
        this.game = game;
        this.position = position;
    }

    public void send(String data) {
        try { connection.sendMessage(data); }
        catch (IOException e) { connection.close(); }
    }

    public void close() { connection.close(); }
}
```

Jeu de morpion – Serveur

```
public class Game {

  private Client[] players;
  private int curPlayer;
  private int grid[][];

  public Game() {
    players = new Client[2];
    grid = new int[3][3];
  }

  public void addPlayer(Client client) {
    if (players[0] == null) {
      players[0] = client; client.setGame(this, 1);
      client.send("W");
    } else { players[1] = client;
             client.setGame(this, 2);
    }
  }
```

Jeu de morpion – Serveur

```
public boolean isComplete() {
  return (players[1] != null);
}

public void start() {
  curPlayer = 1;
  players[curPlayer - 1].send("P");
  players[2 - curPlayer].send("O");
}

public void finish() {
  for (int i = 0; i < 2; i++)
    if (players[i] != null) {
      players[i].close(); players[i] = null;
    }
}

private isWinner(int position) { ... }
```

Jeu de morpion – Serveur

```
    public void onMessage(int position, String d) {
        if (position != curPlayer) return;
        if (!d.matches("^P#[0-9]#[0-9]$")) return;
        int c = d.charAt(2) - '0'; int r = d.charAt(4) - '0';

        if (grid[c][r] != 0) return; grid[c][r] = position;

        players[0].send("D#" + c + "#" + r + "#" + curPlayer);
        players[1].send("D#" + c + "#" + r + "#" + curPlayer);

        if (isWinner(position)) {
            players[position - 1].send("V");
            players[2 - position].send("L"); finish();
        } else {
            curPlayer = 3 - curPlayer;
            players[curPlayer - 1].send("P");
            players[2 - curPlayer].send("O");
        }
    }
}
```