

Langage PHP

Savoirs	Champs ciblé
Choisissez un élément.	Choisissez un élément.

On peut introduire du **code PHP** dans une **page HTML** où l'on veut.
En pratique le **HTML** est placé en début et en fin de page ; le corps est codé en **PHP**.

PHP est sensible à la casse

1. CONSTANTES, VARIABLES, AFFECTATIONS, COMPARAISONS ET COMMENTAIRES EN PHP

Les commentaires :

Exemple de commentaires

```
// Commentaires hérités du C
/* commentaires hérités du C */
# Commentaires hérités du Shell
```

Les constantes :

Exemple : déclaration d'une constante

```
define ('CONSTANTE',10) ;
```

- ⇒ Les constantes sont généralement utilisées pour définir les paramètres d'une application WEB
- ⇒ Il existe plus de 700 variables prédéfinies en PHP comme PHP_VERSION

Convention d'écriture : Les constantes seront déclarées en **MAJUSCULES**, les variables en **minuscules**.

Exemple de code : les variables prédéfinies.

```
<pre>
<?php print_r(get_defined_constants()); ?>
</pre>
```

Les variables et leurs affectations : ⇒ Toutes les variables commencent par le symbole dollar « \$ »

- ⇒ **PHP** n'impose pas de déclarer les types des variables,
- ⇒ **PHP** imposera dynamiquement le type lors de l'affectation.
- ⇒ Toutefois il est possible (et conseillé !) de forcer le type en utilisant le **transtypage**.
- ⇒ 5 types possibles :
 - chaîne de caractères
 - variables numériques, nombres entiers
 - variables numériques, nombres flottants
 - variables booléennes : true ou false
 - le type Null

Types définis dynamiquement par PHP		Transtypage : forçage du type lors de la déclaration	
Variable de type :	Exemple :	Variable de type :	Exemple :
chaîne de caractères	\$abc= 'Bonjour';	chaîne de caractères	\$abc=(string) 1 ; // 1 est string
Numériques, entiers	\$a = 5 ;	Numériques, entiers	\$n=(int)1.8 ; // n vaut 1
Numériques, flottants	\$b = 1.5 ;	Numériques, flottants	\$f = (float) 1.45 ; // f vaut 1.45
booléennes : true ou false	\$var=true	booléennes : false	\$b=(bool) 0 ; // b vaut false
		booléennes : true	\$b=(bool) -3 ; // b vaut true
NULL : ce type composé d'une seule valeur (NULL) indique qu'une variable n'a pas de valeur.			

⇒ Opérations et affectations :

Exemple de code : **affectations** en PHP

<pre>\$i = 5 ; // i ppv 5 ou i ← 5 \$i + = 5 ; // peut être écrit \$i = \$i + 5 \$i ++ ; // peut être écrit \$i = \$i + 1</pre>	<pre>\$a = 3 * 2 ; // a = 6 \$b = pow (2 , 4) ; // b = 16 \$c = \$a + \$b ; // c = 22</pre>	<pre>\$i = (int) 5 ; // forçage de i en type int \$modulo = \$i % 2 ; // résultat 1 \$c = \$i / 2 ; // résultat 2</pre>
---	---	---

a) Les comparaisons :

Opérateur // signification	Exemple :	Résultat ⇒ renvoie « true » si :
== // égal à	\$a == \$b	\$a est égal à \$b
< // Inférieur à	\$a < \$b	\$a est plus petit que \$b
> // Supérieur à	\$a > \$b	\$a est plus grand que \$b
<= // Inférieur ou égal à	\$a <= \$b	\$a est inférieur ou égal \$b
>= // Supérieur ou égal à	\$a >= \$b	\$a est supérieur ou égal \$b
!= // différent de	\$a != \$b	\$a est différent de \$b
=== // identique en type et en valeur	\$a === \$b	\$a est identique à \$b et leur type est le même
!== // différent en type OU en valeur	\$a !== \$b	\$a est différent de \$b ou leur type est différent

isset () : Détermine si une variable est définie et est différente de NULL	<pre><?php \$var = ""; // Ceci est vrai, alors le texte est affiché if (isset(\$var)) {echo 'Cette variable existe';} ?></pre>
empty () : Détermine si une variable est vide Une variable est considérée comme vide si elle n'existe pas, ou si sa valeur équivaut à FALSE	<pre><?php \$var = 0 ; //Evalué à vrai car \$var est vide if (empty(\$var)) { echo '\$var vaut soit 0, vide, ou pas définie du tout'; }</pre>

2. LES STRUCTURES ALTERNATIVES EN PHP

Ecriture algorithmique	Ecriture en PHP
début	< ?php
age = 15 ;	\$age = 15 ;
majorite = 18	\$majorite = 18 ;
si (age ≥ majorite) alors écrire (vous êtes	if(\$age >= \$majorite) { echo'Vous êtes majeur' ;
majeur)	}
sinon écrire (vous êtes	else { echo'Vous êtes mineur' ; }
mineur)	?>
fin	

Structure alternative à choix multiples :

Ecriture algorithmique	Ecriture en PHP
Début nombre ← 4 selon le cas (nombre) si (nombre = 1) écrire (nombre = 1) si (nombre = 2) écrire (nombre = 2) si (nombre = 3) écrire (nombre = 1) autre cas écrire (nombre est < 1 ou > 3) fin	<pre>< ?php \$nombre=4; //Modifier la valeur de \$nombre switch(\$nombre) { case 1: echo "\\$nombre est égal à 1"; break; case 2: echo "\\$nombre est égal à 2"; break; case 3: echo "\\$nombre est égal à 3"; break; default: echo "\\$nombre est < à 1 ou > à 3"; break; } ?></pre>

3. LES STRUCTURES ITERATIVES EN PHP

La boucle « tant que » :

Ecriture algorithmique	Ecriture en PHP
début compteur \leftarrow 1 Tant que (compteur < 10) début écrire (valeur de la variable compteur) ; compteur \leftarrow compteur + 1 fin fin	<pre>< ?php \$compt = 1; while(\$compt < 10) { echo "\$compt / "; \$compt++; } ?></pre>

La boucle « répéter ... tant que » :

Ecriture algorithmique	Ecriture en PHP
Début compteur ← 10 répéter début écrire (valeur de la variable compteur) ; compteur ← compteur + 1 fin Tant que (compteur < 10) fin	<pre>< ?php \$compt = 10 ; do { echo "\$compt / "; \$compt++; } while(\$compt < 10) ; ?></pre>

La boucle « Pour » :

Ecriture algorithmique	Ecriture en PHP
début Pour (compteur = 1 à 10 par pas de 1) début écrire (valeur de la variable compteur) ; fin fin	<pre>< ?php for (\$compt = 1; \$compt <= 10 ; \$compt++) { echo "\$compt / "; } ?></pre>

⇒ La boucle « **foreach** » est une boucle dédiée aux tableaux :

Ecriture algorithmique	Ecriture en PHP
début Pour (i = 0 à i=3 par pas de 1) faire début val ← tab [i] écrire (val) retour chariot fin fin <div style="display: inline-block; vertical-align: middle; margin-left: 20px;"> Pointeur <div style="border: 1px solid black; padding: 2px; display: inline-block;">i</div> → <div style="border: 1px solid black; padding: 2px; display: inline-block; vertical-align: top;"> tab "a" "b" "c" "d" </div> → <div style="border: 1px solid black; padding: 2px; display: inline-block; vertical-align: top;"> val </div> </div>	<pre>< ?php \$tab = array ("a", "b", "c", "d"); foreach (\$tab as \$val) { print("\$val "
); } ?></pre>

A chaque itération **foreach** associe la valeur de l'élément du tableau en cours à la variable **\$val** puis place son pointeur sur l'élément suivant.

4. DECLARATIONS ET APPELS DE FONCTIONS EN PHP

Structure d'une fonction :

```

Function Nom_Fonction ($param1 = "valeur par défaut 1 ", $param2 = "valeur par défaut 2 " ... )
{
    Bloc_De_Code ;
    return $resultat;
}
  
```

Remarque 1 : toutes le fonctions ne renvoient pas de résultats

Remarque 2 : il n'est pas nécessaire d'initialiser tous les paramètres transmis

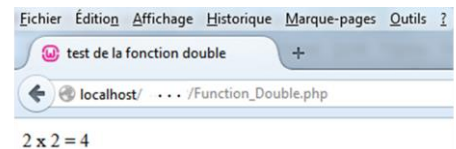
Exemple : mise en œuvre d'une fonction double(\$n) :

```

1 <HTML><HEAD>
2 <TITLE>test de la fonction double</TITLE></HEAD>
3 <BODY>
4 <?php
5   Function double($n)
6   {
7       $resultat = $n + $n ;
8       return $resultat;
9   }
10  $a=2;
11  $b=double($a);
12  echo "2 x $a = $b";
13  ?>
14 </BODY></HTML>

```

Visualisation du résultat :



5. LES STRUCTURES ET FONCTIONS NATIVES D’AFFICHAGES ET DE DATAGE EN PHP

⇒ La fonction phpinfo();

Description de la fonction	Ecriture en PHP
Renvoie la version du soft utilisé	< ?php phpinfo(); ?>

⇒ La structure echo

Description de la fonction	Ecriture en PHP
<p>Avec les guillemets PHP interprète le contenu et remplace les variables par leur valeurs</p> <p>Avec les apostrophes PHP ne fait que copier-coller le contenu</p>	<pre> < ?php \$a =7; \$b=8; \$somme=\$a+\$b; echo "a + b = \$somme
"; //Affiche : a + b = 15 echo 'a + b = \$somme'; //Affiche : a + b = \$somme ?> </pre>

Le retour chariot : Dans le cas d'un echo, le \n est vu comme un retour chariot **dans les sources** du HTML généré.

Solution :

```
echo "<br>";
```

Exemple de la structure echo :

```

1 <html><body>
2 <?php
3 $str1="Cette chaîne a été";
4 $str2="construite avec plusieurs chaînes.";
5 echo "<h2>PHP is Fun!</h2>";
6 echo "Hello world!<br>";
7 echo 'I\'m about to learn PHP!<br>';
8 echo "Cette ", "chaîne ", "a été ", "construite ", "avec plusieurs chaînes.<br>";
9 echo "$str1 $str2<br>";
10 echo $str1.$str2;
11 ?>
12 </body></html>

```

Opérateur d'échappement : antislash : \
 ⇒ Il permet d'insérer les symboles guillemets ' et ".
 Exemple : ligne 7

Opérateur de concaténation : le point.
 Exemple : ligne 10 : Echo \$str1.\$str2 ;

PHP is Fun!

Hello world!
 I'm about to learn PHP!
 Cette chaîne a été construite avec plusieurs chaînes.
 Cette chaîne a été construite avec plusieurs chaînes.
 Cette chaîne a été construite avec plusieurs chaînes.

⇒ La structure print : exemple :

```

1 <html><body>
2 <?php
3 $a=5;$b=2;$somme=$a+$b;
4 print("Bonjour le monde<br>");
5 print "print() fonctionne aussi sans les parenthèses.<br>";
6 print "$a+$b=$somme<br>";
7 print ' $a+$b=$somme ' ;
8 ?>
9 </body></html>

```

http://localhost/.../Cours_PHP_5_5_print.php

Bonjour le monde
 print() fonctionne aussi sans les parenthèses.
 5+2=7
 \$a+\$b=\$somme

Différence entre print et echo ⇒ echo est une structure du langage PHP, plus rapide, mais ne renvoie pas de valeur.

⇒ Print est aussi une structure de langage, mais il se comporte comme une fonction, il renvoie un entier après exécution.

Autres fonctions d'affichage :

Printf : pour gérer des chaînes formatées : voir cours de langage C.

Print_r : affiche un tableau

var_dump(\$var) ; ⇒ affiche les informations structurées d'une variable, y compris son type et sa valeur. Les tableaux et les objets sont explorés récursivement, avec des indentations, pour mettre en valeur leur structure.

⇒ Le timestamp, La fonction time() et la fonction mktime() :

Description de la fonction	Ecriture en PHP
<p>⇒ Le timestamp unix représente le nombre de secondes écoulées depuis le 1er janvier 1970 à minuit GMT.</p> <p>Pour obtenir le timestamp actuel, il vous suffit d'utiliser la fonction time().</p> <p>⇒ La fonction time() ne prend aucun argument et retourne la date courante en secondes depuis le 1er janvier 1970.</p> <p>⇒ La fonction mktime retourne, pour une date donnée, le timestamp lui correspondant.</p> <p>int mktime (int hour, int minute, int second, int month, int day, int year [, int is_dst])</p>	<pre><?php echo time(); // Affichera 10639xxxxx ?> <?php // Secondes écoulées entre le 1er janvier 1970 et le //27 mai 2002 à midi echo mktime(12, 0, 0, 5, 27, 2002); //Affichera 10224xxxxx ?></pre>

⇒ La fonction date :

Description de la fonction	Ecriture en PHP
La fonction date() permet d'obtenir une multitude de représentations d'un timestamp	<pre><?php \$aujourd'hui = date("d/m/Y"); echo \$aujourd'hui; // affiche 01/01/2015 echo "
"; // retour chariot echo date("d/m/y"); // affiche 01/01/15 car y!=Y ?></pre>

Code	Description
d	Jours du mois sur 2 chiffres
D	Jours de la semaine en 3 lettres ... and in English
w	Jours de la semaine numérique/0:dimanche 6:samedi
l	Jours de la semaine version longue et en anglais
F	Mois textuel version longue et en anglais
m	Le mois de 01 à 12
M	Le mois en 3 lettres en anglais
g	Heure sur 12h sans les zéros initiaux de 1 à 12
h	Heures au format 12h
I	Indique si l'heure d'été est activée 0 ou 1
c	Date au format ISO 8601

Code	Description
J	Jours du mois sans les zéros initiaux
z	Jour de l'année de 0 à 366
W	Numéro de la semaine dans l'année de 1 à 52
Y	Année à 4 chiffres
y	Année à 2 chiffres
n	Mois sans les zéros initiaux 1 à 12
t	Nombre de jours dans le mois donné 28 à 31
G	Heure sur 24h sans les zéros initiaux de 0 à 23
H	Heures au format 24h
s	Secondes 00 à 59
r	Date au format RFC 822

6. LES CHAINES DE CARACTERES EN PHP : FONCTIONS COURAMMENT UTILISEES.

Description de la fonction	Ecriture en PHP
Longueur d'une chaîne :	<code>echo strlen(\$str); // affiche la longueur d'une chaîne de caractères</code>
Position d'une chaîne dans une autre	<code>echo strpos(\$s1, \$s2); // renvoie la position de \$s2 dans \$s1</code>
Formatage d'une chaîne avec printf	<code>\$string = " %s : %d "; printf (\$string, "Cas n°", 17); // Renvoie Cas n° : 17</code>
Remplacement	<code>\$str = str_replace('old', 'new', \$str);</code>
Suppression des caractères superflus aux extrémités d'une chaîne.	<code>\$clean=trim(" bonjour ! \n "); echo \$clean; //renvoie "bonjour !"</code>
Formatage HTML Retourne la chaîne de caractères sans traduire le caractère spécial HTML	<code>\$string = " essai &agrave faire"; echo htmlspecialchars(\$string); // renvoie: essai &agrave faire echo "\$string";// renvoie: essai à faire</code>
Re-direction d'une page vers une URL	<code>Header('location : http://chevrollier.paysdelaloire.e-lyco.fr');</code>

7. LES TABLEAUX EN PHP

Un tableau est créé en utilisant la structure de langage **array()** ou par **[]** depuis PHP.

Il prend un nombre illimité de paramètres, séparés par une virgule, sous la forme d'une paire **key => value**.

Si aucune clé n'est spécifiée, l'indice maximal existant est repris, et la nouvelle clé sera incrémentée de 1. Si aucun indice n'existe, la clé sera zéro. Exemple :

```

1 <html><body>
2 <pre>
3 <?php
4 $a = array ('a' => 'apple', 'b' => 'banana', 'c' => array ('x', 'y', 'z'));
5 print_r ($a);
6 ?>
7 </pre>
8 </body></html>

```

Cette instruction demande l'affichage de tous les éléments du tableau

Fonctions couramment utilisées :

`$array[] = "Georges"; // ajoute une valeur, key=0`
`$array[] = "Jacques"; // ajoute une valeur, key=1`
`unset($array[0]); // Ceci efface l'élément du tableau`
`unset($array); // Ceci efface complètement le tableau`
`$nb = count ($array); //nombre d'éléments dans le tableau`
`$key = array_search('Jacques', $array); // position d'un élément dans le tableau renvoie key=1`
`if (in_array('Georges', $array)){echo 'Trouvé';} // vérifie si un tableau contient une valeur`

Les super-globales sont des tableaux prédéfinis accessibles partout dans le code :

Liste des superglobales	Ecriture en PHP
\$GLOBALS[]	<p>Tableau qui contient toutes les variables disponibles dans un contexte global.. Les noms des variables sont les index du tableau.</p> <pre> 1 <html><body> 2 <?php 3 function test() { 4 \$foo = "variable locale"; 5 echo '\$foo dans le contexte global : ' . \$GLOBALS["foo"] . "\n". "
"; 6 echo '\$foo dans le contexte courant : ' . \$foo . "\n"; 7 } 8 \$foo = "Exemple de contenu"; 9 test(); 10 ?> 11 </body></html> </pre> <p>Le script renvoie : \$foo dans le contexte global : Exemple de contenu \$foo dans le contexte courant : variable locale</p>
\$_SERVER []	Contient les informations sur les requêtes en cours sur PHP et sur serveur http
\$_GET []	Contient les données envoyées via l'URL voir 5.10.1
\$_POST []	Contient les données envoyées via la méthode POST voir 5.10.2
\$_FILES []	Contient la liste et les caractéristiques des fichiers transmis au serveur par le visiteur
\$_COOKIE []	Tableau associatif de variables, passé au script courant, via des cookies HTTP
\$_SESSION []	Tableau associatif des valeurs stockées dans les sessions, et accessible au script courant.
\$_REQUEST []	Tableau contenant par défaut le contenu des variables \$_GET, \$_POST et \$_COOKIE.
\$_ENV []	Contient les informations « bas niveaux » sur le système d'exploitation

8. LA GESTION DE FICHIERS EN PHP

Un peu de philosophie : Pourquoi utiliser des fichiers en PHP plutôt qu'une base de données ?

- ⇒ La gestion d'un fichier texte stocké sur le serveur PHP est plus rapide d'accès que la gestion d'une SGDB, MAIS les outils de gestion des SGDB sont plus performants.
- ⇒ Si deux personnes interviennent en même temps sur un fichier, les données peuvent être inconsistantes, ce qui n'est pas le cas sur une SGDB.
- ⇒ En bref : pour les petites applications, avec peu d'évolution de contenu, la gestion par fichier semble suffisante et plus rapide ; si le volume des données, le nombre de clients, le nombres d'évolutions des données augmentent il faut mettre en œuvre une SGBD.

Formats de fichier gérés en PHP : ⇒ GZIP ⇒ CSV ⇒ PNG

Un fichier **CSV** est un fichier texte compatible EXCEL, contenant des données sur chaque ligne séparées par un caractère de séparation (généralement une virgule, un point-virgule ou une tabulation).

CSV : Comma-separated values

Exemple :

prénom,Nom
Adriana,Lima
Carmen,Crue

Fonctions de gestion des fichiers	Descriptions :
Fopen () :	Permet d'ouvrir un fichier
Fgets () :	Renvoie une ligne du fichier
Fread () :	Renvoie n caractères du fichier
Fwrite () :	Ecrit dans un fichier
Fclose () :	Ferme la connexion au fichier

Mode D'ouverture de fichier :

Mode :	Ecriture en PHP
r	Ouvre en lecture seule, place le pointeur en début de fichier
r+	Ouvre en lecture et écriture, place le pointeur en début de fichier
w	Ouvre en écriture seule, place le pointeur en début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas il est créé.
w+	Ouvre en lecture et écriture, place le pointeur en début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas il est créé.
a	Ouvre en écriture seule, place le pointeur à la fin du fichier. Si le fichier n'existe pas il est créé.
a+	Ouvre en lecture et écriture, place le pointeur à la fin du fichier. Si le fichier n'existe pas il est créé.

Exemple 1 : Création d'un fichier CSV en PHP :

```

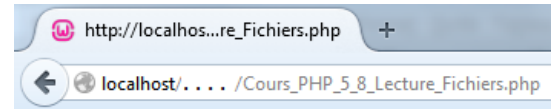
1 <?php
2 $lignes[] = array('prénom', 'Nom'); // Les lignes du tableau
3 $lignes[] = array('Adriana', 'Lima');
4 $lignes[] = array('Carmen', 'Crue');
5
6 $chemin = 'fichier.csv'; // adressage relatif
7 $delimiteur = ','; // Pour une tabulation, utiliser $delimiteur = "t"
8 $fichier_csv = fopen($chemin, 'w+');// Création du fichier csv, w+ : en lecture écriture
9
10 foreach($lignes as $ligne)
11 {
12     fputcsv($fichier_csv, $ligne, $delimiteur); // Boucle foreach sur chaque ligne du tableau
13 } // chaque ligne est insérée dans le fichier
14 fclose($fichier_csv); // fermeture du fichier csv
15 ?>

```

Visualisation de fichier.csv :

	A	B	C
1	prénom	Nom	
2	Adriana	Lima	
3	Carmen	Crue	
4			

Visualisation



Les 2 champs de la ligne 1 sont : prénom Nom
Les 2 champs de la ligne 2 sont : Adriana Lima
Les 2 champs de la ligne 3 sont : Carmen Crue

Exemple 2 : Lecture de données d'un fichier CSV en PHP :

```

1 <?php
2 $ligne = 1; // compteur de ligne
3 $fic = fopen("fichier.csv", "a+");
4 while($stab=fgetcsv($fic,1024,', '))
5 {
6     $champs = count($stab);//nombre de champ dans la ligne en question
7     echo "<b> Les " . $champs . " champs de la ligne " . $ligne . " sont : </b>";
8     $ligne++;
9     //affichage de chaque champ de la ligne en question
10    for($i=0; $i<$champs; $i++){echo $stab[$i]. " ";}
11    echo "<br />";
12 }
13 ?>

```

9. LES SESSIONS ET LES COOKIES EN PHP

Les sessions permettent de conserver les informations concernant un utilisateur tout au long de sa navigation (ex : panier d'achat, espace sécurisé.)

Fonctions permettant de gérer les sessions	Ecriture en PHP
Initialiser une session	session_start() ;
Mettre une variable dans la session NE RIEN ENVOYER AU NAVIGATEUR AVANT L'INITIALISATION DES SESSIONS	\$_SESSION['auteur']= "Emile Zola" ; var_dump(\$_SESSION) ; echo 'Hello' .\$_SESSION['auteur']. ' ! ' ;
Fermer une session	session_destroy() ;

```

1 <?php
2 session_start();
3 if (!isset($_SESSION['visite']))
4 {
5     $_SESSION['visite']=1;
6     echo "première visite";
7 }
8 else
9 {
10    $_SESSION['visite']++;
11    echo "Vous avez visité cette page " .$_SESSION['visite']. " fois";
12 }
13 ?>

```

Visualisation

Vous avez visité cette page 2 fois

La gestion des cookies s'effectue avec la seule fonction « **setcookie()** »

Exemples de la fonction setcookie	Ecriture en PHP
Création du cookie « moncookie » contenant la chaîne de caractères « hello »	<pre><?php setcookie("moncookie","hello"); echo "cookie crée"; ?></pre>
Utilisation de la superglobale \$_COOKIE []	<pre><?php echo "Valeur du cookie : ".\$_COOKIE['moncookie']; ?></pre>
pour supprimer un cookie il faut supprimer son contenu. Attention le cookie sera vraiment supprimé lors de la fermeture de la session ou du navigateur.	<pre><?php setcookie("moncookie"); ?></pre>

10. ENVOI DE FORMULAIRE : IL Y A DEUX "METHODES" POUR ENVOYER LES DONNEES D'UN FORMULAIRE

La méthode **POST** est utilisée dans 99% des cas

⇒ La méthode **get** : les données transitent par l'URL.

Cette méthode est assez peu utilisée car on ne peut pas faire transiter plus de 256 caractères dans l'URL.

⇒ La méthode **post** : les données ne transiteront pas par l'URL, l'utilisateur ne les verra donc pas passer dans la barre d'adresse. Cette méthode permet d'envoyer autant de données que l'on veut. Néanmoins, les données ne sont pas plus sécurisées qu'avec la méthode GET.

a) Exemple : mise en œuvre de la méthode GET :



```

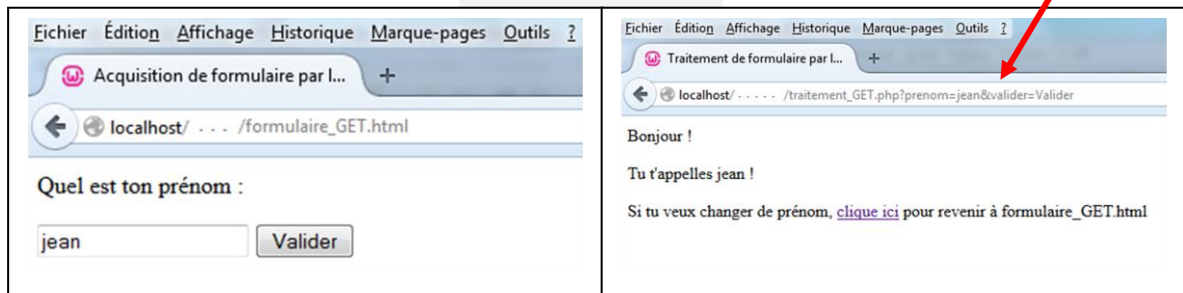
1 <HTML><HEAD>
2 <meta http-equiv="Content-Type" content="text/html"; charset="utf-8"/>
3 <TITLE>Acquisition de formulaire par la méthode GET</TITLE></HEAD>
4 <BODY>
5 <p>Quel est ton prénom :</p>
6 <form method="get" action="traitement_GET.php" />
7 <p>
8 <input type="text" name="prenom" />
9 <input type="submit" name="valider" value="Valider" />
10 </p>
11 </form></BODY></HTML>
  
```

```

1 <HTML><HEAD>
2 <meta http-equiv="Content-Type" content="text/html"; charset="utf-8"/>
3 <TITLE>Traitement de formulaire par la méthode POST</TITLE></HEAD>
4 <BODY>
5 <p>Bonjour !</p>
6 <p>Tu t'appelles <?php echo $_GET['prenom']; ?> !</p>
7 <p>Si tu veux changer de prénom,<br>
8 <a href="formulaire_GET.html">clique ici</a> pour revenir à formulaire_GET.html</p>
9 </BODY></HTML>
  
```

Visualisation attendue :

Notez le passage des paramètres dans la barre d'URL



Exemple : mise en œuvre de la méthode POST :



```

1 <HTML><HEAD>
2 <meta http-equiv="Content-Type" content="text/html"; charset="utf-8"/>
3 <TITLE>Acquisition de formulaire par la méthode POST</TITLE></HEAD>
4 <BODY>
5 <p>Quel est ton pr&eacute;nom :</p>
6 <form method="post" action="traitement.php" />
7 <p>
8 <input type="text" name="prenom" />
9 <input type="submit" name="valider" value="Valider" />
10 </p>
11 </form></BODY></HTML>
  
```

Fichier : formulaire.html

- ⇒ Les données du formulaire sont traitées dans la page indiquée dans le paramètre action.
- ⇒ La méthode POST utilisée ici indique la façon dont PHP effectue le transfert des données : elles sont « postées » dans un tableau associé nommé \$_POST.
- ⇒ Ce transfert est totalement transparent. L'utilisateur ne peut pas modifier les valeurs transmises.
- ⇒ Le nom de l'élément de formulaire constitue la clé de ce tableau.

Élément du tableau	Variable associée
<input type="text" name="prenom" />	\$_POST["prenom"]
<input type="submit" name="valider" value="Valider" />	\$_POST["valider"]

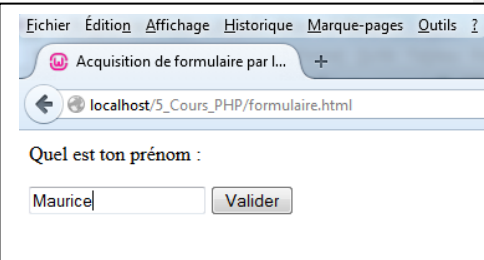
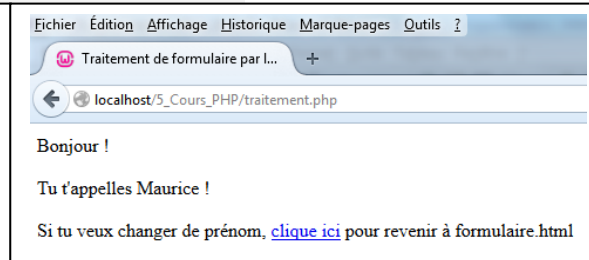
⇒ La page traitement.php reçoit le prénom dans une variable nommée \$_POST['prenom']

```

1 <HTML><HEAD>
2 <meta http-equiv="Content-Type" content="text/html"; charset="utf-8"/>
3 <TITLE>Traitement de formulaire par la m&eacute;thode POST</TITLE></HEAD>
4 <BODY>
5 <p>Bonjour !</p>
6 <p>Tu t'appelles <?php echo $_POST['prenom']; ?> !</p>
7 <p>Si tu veux changer de pr&eacute;nom, <a href="formulaire.html">clique ici</a> pour revenir &agrave; formulaire.html</p>
8 </BODY></HTML>
  
```

Fichier : traitement.php

Visualisation attendue :

 <p>Fichier Édition Affichage Historique Marque-pages Outils ?</p> <p>Acquisition de formulaire par I... +</p> <p>localhost/5_Cours_PHP/formulaire.html</p> <p>Quel est ton prénom :</p> <p>Maurice Valider</p>	 <p>Fichier Édition Affichage Historique Marque-pages Outils ?</p> <p>Traitement de formulaire par I... +</p> <p>localhost/5_Cours_PHP/traitement.php</p> <p>Bonjour !</p> <p>Tu t'appelles Maurice !</p> <p>Si tu veux changer de prénom, clique ici pour revenir à formulaire.html</p>
--	--