Lycée TOUCHARD WASHINGTON

Le langage PHP

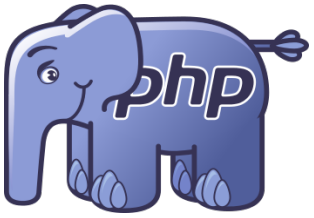

Table des matières

1 Historique

On doit la première version de **PHP** à ***Rasmus Lerdorf*** qui l'a mise au point pour ses propres besoins en 1994. Cette version était destinée à son usage personnel, d'où le nom (**Personal Home Pages**). Elle n'a pas été mise à disposition du public.

Vers 1995, une version qui permettait l'exécution de quelques macros fut mise à disposition (gérer un livre d'or, un compteur de hits et quelques autres choses).

En 1998, grâce à la collaboration de ***Zeev Suraski*** et ***Andy Gutmans***, fondateurs de la société **Zend**, la version 3 voit le jour.

Elle s'appelle désormais **PHP** : "**Hypertext Preprocessor**". La version 4 voit le jour en 2000.

En juillet 2004 la version 5 sort en release. La dernière mise à jour est la version 5.6.1 qui date du 3 mars 2016 .

En décembre 2015 la version 7 sort en release. La dernière mise à jour est la version 7.3,2 qui date du 7 février 2019.

2 Principe

PHP est un ensemble de fonctionnalité qu'il est possible d'insérer directement dans le code d'une page HTML, comme le JavaScript, mais ce code va s'exécuter du côté serveur.

Cela permet donc de cumuler les avantages du JavaScript et d'un script CGI.

Mais ne vous y trompez pas, écrire un code en PHP, ne dispense en rien du fait d'avoir à écrire des JavaScript, qui ont toujours une place dans les pages HTML, notamment pour vérifier la conformité des informations saisie dans les champs d'un formulaire par un visiteur, avant de traiter lesdites informations.

Le principe est relativement simple, les éléments du langage PHP se trouveront toujours entre 2 balises particulières.

```
<? et ?>
ou
<?php et ?>
```

Ses balises pourront se placer n'importe où dans le code HTML. Il est également possible d'en mettre plusieurs fois dans une page.

3 Fonctionnalités

PHP, comme le C, le JavaScript, ou encore le shell sous Unix, permet d'avoir un certain nombre de fonctionnalités.

3.1 Les variables

En PHP, toutes les variables sont précédées du symbole **$**.

Le nom de la variable peut comporter des lettres, des chiffres, ainsi que le caractère _.

Il n'est pas possible d'y inclure des espaces ou des caractères qui ne sont pas alphanumériques.

Comme en C, les instructions se terminent par un **;** .

Exemple de variables :

```
$a;
$a_nom_de_variable_plutot_long;
$1234;
$rienDeSpeciale23332___555_;
```

Une variable peut contenir aussi bien des chaînes de caractères que des nombres, des tableaux, des objets ou des booléens.

Exemple :

```
$unNombre = "coucou";
$uneChaine = 5430 ;
```

Il est possible d'avoir des variables dites "dynamiques".

Exemple :

```
$nom = "robert" ;
$je_suis_trop_fort="nom";
```

si j'écris :

```
print $nom;
print $je_suis_trop_fort;
print $$je_suis_trop_fort;
```

j'aurais :

```
robert
nom
robert
```

3.2 Les références aux variables:

Il peut être utile dans certains cas qu'une variable soit automatiquement mise à jour en fonction d'une autre.

Regardons le code suivant:

```
$a = 43 ;
$b = $a ;
$a = 123 ;
print $b;
```

affichera 43

Si par contre j'écris:

```
$a = 43 ;
$b = &$a ;
$a = 123 ;
print $b;
```

j'obtiendrais à l'affichage : 123

ATTENTION

Le code PHP, comme le code HTML, C ou JavaScript est interprété de haut en bas. Cela implique de faire attention à bien initialiser les variables AVANT de les traiter.

3.3 Les types

Par défaut, les variables en PHP ne sont pas typées (comme en JavaScript). Ainsi, il est possible d'écrire un code comme celui qui suit sans avoir pour autant des erreurs:

```
$a="coucou";
$a=3;
$a=true;
```

Cela permet une très grande flexibilité dans les scripts, mais cette flexibilité est parfois aussi source d'erreur ou du moins d'égarement.

En effet, il ne sera pas rare que dans des scripts importants, le programmeur pense qu'une variable

est d'un certain type à l'endroit où il l'utilise, alors qu'en fait, elle est d'un type complètement différent, voire même incompatible.

Exemple :

```
$a = 4;
...
$a = "salut" ;
...
// ici le programmeur pense que $a est de type entier
$a = $a * 5 ;
```

ce qui ne donnera pas le résultat espéré.

En réalité, en PHP il existe 6 types de donnés :

- entier
- double (réel)
- chaîne
- booléen
- tableau
- objet

Il est possible de tester le type d'une variable grâce à la fonction **gettype()**.

Exemple :

```
$a = 43 ;
print gettype($a);
$a="toto";
print gettype($a);
```

affichera :

```
integer
string
```

Il existe 2 façons pour modifier le type d'une variable. Soit avec la fonction **settype()**, soit avec un opérateur de **cast**.

Exemple :

```
$a = 3.14 ;
settype($a,integer); → $a est devenu un entier. sa valeur sera donc 3. La même chose est possible en écrivant :
$a = 3.14 ;
$a = (integer) $a ;
```

3.4 Les opérateurs

3.4.1 Opérateur d'affectation

comme en C : **=**

Exemple :

```
$mavariable = "salut" ;
```

3.4.2 Opérateur arithmétique

comme en C : **+, -, *, /, %**

Exemple :

```
$mavariable = 3 + 5 * 6 / 2 + 5 % 2;
```

3.4.3 Opérateur de concaténation

comme en JavaScript : **.**

Exemple :

```
$mavariable = "bonjour " . "le monde" ; // donne bonjour le monde
```

3.4.4 Opérateur d'affectation supplémentaire

comme en C : **+=, -=, *=, /=, %=, .=**

Exemples :

```
$x = 3;
$y = "coucou";
$x += 5 -> $x=8
$y .=" le monde"; -> $y="coucou le monde";
```

3.4.5 Opérateur de comparaison

comme en C : **==, !=, >, <, >=,<=** ainsi que **===** pour tester l'égalité des valeurs et le même type.

Exemple :

```
if ($a == $b) // test si $a possède la même valeur que $b
if ($a === $b) // test si $a possède la même valeur que $b ET est de même type
```

3.4.6 Opérateur logique

comme en C : **&&, ||, !, or, xor, and**

Exemple :

```
if ($a==5 && $b<3) // test si $a possède la valeur 5 et si $b est plus petit que 3
```

3.4.7 Incrémentation, décrémentation automatique

comme en C : **++, --**

Exemple :

```
$x=1;
$x++; -> $x = 2
```

3.5 Les constantes

Il est parfois préférable, plutôt que d'utiliser des variables, d'utiliser des "variables" dont la valeur ne doit pas changer, bref, des constantes.

Cela est faisable grâce à la fonction, **define()**.

Exemple :

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

Pour accéder à la constante, il n'est pas nécessaire d'avoir recours au symbole **$**.

Ainsi pour afficher la constante, je peux écrire :

```
print "la valeur de pi est : ".PI;
```

3.6 Les conditions

un branchement conditionnel se fait, comme en JavaScript, grâce à l'instruction **if**, avec éventuellement un **else**.

```
if (condition)
{
   ...
}
else
{
   ...
}
```

Il est également possible d'utiliser un **switch**.

```
switch (expression)
{
   case resultat1 :
            ...
   break;
   .....

   case resultatn :
            ...
   break;
   default :
            ...
}
```

Pour des questions pratiques, il peut être utile d'utiliser l'opérateur ternaire '**?**' .

Exemple :

```
($a == $b)?$a++:$b++;
        // si $a est égale à $b
        //        alors on incrémente $a
        //        sinon, on incrémente $b
```

3.7 Les boucles

Il existe 3 types de boucle en PHP :

Boucle indéfinie avec le test au début : tant que la condition est vraie, faire une suite d'instructions.

```
while (condition)
{
   ...
}
```

Boucle indéfinie avec le test à la fin : Faire une suite d'instruction tant que la condition est vraie.

```
Do
{
   ...
} while(condition) ;
```

Boucle définie : Lorsque l'on sait à l'avance combien de "tours de boucle " on doit faire, il faut utiliser une boucle pour.

ATTENTION, la variable affectée d'une boucle pour est incrémentée ou décrémentée automatiquement jusqu'à ce que la condition de la boucle soit fausse.

```
for (affectation; condition ; incrementation/decrementation)
{
   ...
}
```

3.8 Les tableaux

Il existe plusieurs façons pour créer un tableau.

La plus simple :

```
$tab[] = "un" ;
$tab[] = "deux" ;
$tab[] = "trois" ;
```

donne un tableau dont les indices vont de 0 à 2 et dont les valeurs sont respectivement "un", "deux", et "trois".

Une autre façon de procéder est d'utiliser la fonction **array()**:

```
$tab = array( "un", "deux", "trois") ;
```

Il est possible de connaître le nombre d'éléments d'un tableau grâce à la fonction **count()**.

```
count($tab) -> 3
```

Pour parcourir un tableau, il est toujours possible d'utiliser une boucle for, mais une instruction **foreach** est plus convenable en PHP.

Exemple :

```
foreach ($tab as $valeur){
   print $valeur;
}
```

Affichera :

```
un
deux
trois
```

Fusionner des tableaux est très simple en PHP, grâce à la fonction **array_merge**.

Exemple :

```
$tab2 = array("quatre", "cinq");
$tabfusion = array_merge($tab, $tab2);
foreach ($tabfusion as $valeur){
   print $valeur;
}
```

Affichera :

```
un
deux
trois
quatre
cinq
```

De même, ajouter des valeurs à un tableau se fera à l'aide de la fonction **array_push() .**

Exemple :

```
$nombredelements = array_push($tab, "quatre", "cinq");
foreach ($tab as $valeur){
   print $valeur;
}
print "il y a $nombredelements elements dans le tableau";
```

Affichera :

```
un
deux
trois
quatre
cinq
il y a 5 elements dans le tableau
```

La fonction **array_slice** va nous permettre d'extraire un bout de tableau d'un tableau.

Exemple :

```
$tab2 = array_slice($tab,2,2) ;
```

- le premier argument de **array_slice** est le tableau duquel on souhaite extraire des données (le tableau ne sera pas modifié!!).
- Le second argument est l'indice à partir duquel on souhaite commencer à extraire les données.
- Le dernier argument est le nombre de données que l'on souhaite extraire (s'il n'est pas renseigné, on va jusqu'à la fin du tableau).

$tab2 est le nouveau tableau créé avec les données extraites de **$tab**. ici **$tab2** sera constitué ainsi :

```
$tab2[0] -> "trois"
$tab2[1] -> "quatre"
```

3.9 Les tableaux multidimensionnels

Un tableau multidimensionnel est un tableau qui comporte d'autres tableaux.

Un exemple de déclaration d'un tableau multidimensionnel peut se faire de la façon suivante:

```
$multitab = array( array("l0c0","l0c1","l0c2"), array("l1c0","l1c1","l1c2"),
array("l2c0","l2c1","l2c2") );
```

soit à l'affichage :

```
echo $multitab[0][0];  -> l0c0
echo $multitab[2][2]; -> l2c2
echo $multitab[1][2]; -> l1c2
```

3.10 Les tableaux associatifs

Un tableau associatif est un tableau indexé par des chaînes de caractères plutôt que des chiffres.

Exemple :

```
$user[nom]="bidochon" ;
$user[prenom]="robert ";
$user["deuxieme prenom"]="gaston" ;
$user[age]=54 ;
```

On remarque que lorsque l'index est composé de caractères spéciaux comme un espace ou un caractère accentué, l'index est placé entre guillemets.

Une autre façon de remplir ce tableau est :

```
$user = array( nom => "bidochon", prenom => "robert", "deuxieme prenom" => "gaston", age =>54);
```

pour afficher un tableau associatif, il faut utiliser la syntaxe suivante :

```
foreach($user as $clef => $valeur){
   echo "nom de l'index :";
   echo $clef;
   echo ", valeur se trouvant à cet index :";
   echo $valeur;
}
```

Ce qui affichera :

```
nom de l'index : nom, valeur se trouvant à cet index : bidochon
nom de l'index : prenom, valeur se trouvant à cet index : robert
nom de l'index : deuxieme prenom, valeur se trouvant à cet index : gaston
nom de l'index : age, valeur se trouvant à cet index : 54
```

3.11 Les fonctions

Comme en C ou en JavaScript, il est possible d'utiliser et de définir des fonctions.
La syntaxe est la suivante :

```
function nom_de_la_fonction($parametre1, $parametre2, ...,$parametreN){
        // code de la fonction
        return une_valeur; // une_valeur peu être une variable. le return n'est pas obligatoire.
}
```

À l'inverse du C, il n'est pas nécessaire de définir le type de ce qui sera retourné par la fonction. Idem pour les paramètres de celle-ci. En revanche, toute variable définie dans la fonction restera **locale** à cette dernière.

Exemple :

```
function le_sens_de_la_vie(){
   $reponse="celui qu'on lui donne";
}
echo "quel est le sens de la vie ?"
le_sens_de_la_vie();
echo "la reponse est :".$reponse;
```

Affichera :

```
quel est le sens de la vie ?
la reponse est :                          // la variable réponse n'est pas visible ici
```

Idem, avec une variable définie à l'extérieur de la fonction.

```
$a = 5 ;
function doubleval()
{
   $a = $a *2 ;
   echo "la valeur de a dans la fonction est : $a";
}

echo "la valeur de a avant l'appel a la fonction est : $a";
doubleval();
echo "la valeur de a après l'appel a la fonction est : $a";
```

Affichera :

```
la valeur de a avant l'appel a la fonction est : 5
la valeur de a dans la fonction est : 0
la valeur de a après l'appel a la fonction est : 5
```

Si l'on souhaite modifier une variable définie à l'extérieur d'une fonction, il faut utiliser l'instruction **global.**

```
$a = 5 ;
function doubleval()
{
   global $a;
   $a = $a *2 ;
   echo "la valeur de a dans la fonction est : $a";
}
echo "la valeur de a avant l'appel a la fonction est : $a";
doubleval();
echo "la valeur de a après l'appel a la fonction est : $a";
```

Affichera :

```
la valeur de a avant l'appel a la fonction est : 5
la valeur de a dans la fonction est : 10
la valeur de a après l'appel a la fonction est : 10
```

4 Les formulaires

PHP est surtout pratique pour la facilité de traitement des informations en provenance de formulaires. Les données envoyées par un formulaire correspondent à tous le champs ayant un attribut **name**.

4.1 Les variables d'environnement

Il est possible d'avoir accès à un certain nombre de variables d'environnement.

$_SERVER[HTTP_USER_AGENT]	nom du navigateur client
$_SERVER[REMOTE_ADDR]	adresse IP du client
$_SERVER[REQUEST_METHOD]	Méthode utilisée par la page formulaire dont nous recevons les informations (**get** ou **post**).
$_SERVER[QUERY_STRING]	données envoyées par une méthode get
$_SERVER[REQUEST_URI]	adresse complète de la requête
$_SERVER[HTTP_REFERER]	adresse de la page à partir de laquelle la requête a été effectuée

Il existe d'autres variables prédéfinies telles que :

- $GLOBALS : Tableau de toutes les variables.
- $_COOKIE : Tableau des cookies.
- $_FILES : Tableau des informations relatives au fichier upload.
- $_ENV : Tableau des variables d'environnement de l'OS.
- $_SESSION : Tableau des variables de session.

4.2 Méthode GET ou méthode POST

Les données d'un formulaire sont envoyées par l'une des deux méthodes d'envois **post** ou **get**. Les données sont alors stockées respectivement dans les variables prédéfinies **$_POST** ou **$_GET** sous forme de tableau associatif où le nom de l'étiquette est le nom correspondant à l'attribut **name** des champs du formulaire.

Pour une méthode GET : *$nomDeVariable = $_GET["nomDuChampDeMonFormulaire"] ;*
Pour une méthode POST : *$nomDeVariable = $_POST["nomDuChampDeMonFormulaire"] ;*

Application :

Page de formulaire envoyant des données vers le script traitement.php

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Formulaire d'authentification</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <form action="traitement.php" method="post">
            <label>Login
                <input type="text" name="identifiant" placeholder="Votre login" required/>
            </label>
            <br/>
            <label>Mot de passe
                <input type="password" name="mdp" placeholder="Votre mot de passe" required/>
            </label>
            <br/>
            <label>Courriel
                <input type="email" name="courriel" placeholder="Votre adresse mail" required/>
            </label>
            <input type="submit"/>
        </form>
    </body>
</html>
```

Traitement.php

```php
<?php
// tester la méthode d'envois du formulaire
if ($_SERVER['REQUEST_METHOD']=='POST')
{
    echo "données envoyées via la méthode post<br/>";
    // afficher les données entrantes
    echo "<pre>";
    print_r($_POST);
    echo "</pre>";
    // afficher les données entrantes individuellement
    echo "login : ".$_POST['identifiant']."<br/>";
    echo "mot de passe : ".$_POST['mdp']."<br/>";
    echo "adresse mail : ".$_POST['courriel']."<br/>";
}
```

login : coucou
mot de passe : toto
adresse mail : bidon@ici

Coté formulaire :

Login coucou
Mot de passe ••••
Courriel bidon@ici Envoyer

Coté traitement.php :

Donnée envoyées via la méthode post :

Array
{
    [identifiant] => coucou
    [mdp] => toto
    [courriel] => bidon@ici
}

4.3 Utilisation de filtres

Lors de l'exemple précédent, on constate qu'un champ n'a pas été correctement complétés par l'utilisateur. Avant le traitement il est nécessaire de vérifier la validité des données transmises. Il est donc d'usage d'utiliser la fonction **filter_input**, qui comme son nom l'indique utilise des **filtres**, afin de s'assurer que les données reçues sont du bon type ou de la bonne forme.

traitement.php avec utilisation des filtres

```php
<?php
if (filter_input(INPUT_SERVER, 'REQUEST_METHOD')=='POST')
{
    echo "données envoyées via la méthode post avec securisation<br/>";
    // afficher les données entrantes individuellement
    echo "login : ".filter_input(INPUT_POST,'identifiant')."<br/>";
    echo "mot de passe : ".filter_input(INPUT_POST,'mdp')."<br/>";
    echo "adresse mail : ".filter_input(INPUT_POST,'courriel',FILTER_VALIDATE_EMAIL)."<br/>";

}
```

Avec la fonction filter_input, nous n'accédons pas directement aux variables prédéfinies.

Résultats :

formulaire	traitement.php
Login coucou Mot de passe •••• Courriel bidon@ici Envoyer	données envoyées via la méthode post avec securisation login : coucou mot de passe : toto adresse mail :

Comme l'adresse mail n'était pas correcte, celle-ci n'est pas affiché.

Si l'on revient sur le formulaire afin de modifier le champ incriminé, nous obtenons ceci :

formulaire	traitement.php
Login coucou Mot de passe •••• Courriel bidon@ici.fr Envoyer	données envoyées via la méthode post avec securisation login : coucou mot de passe : toto adresse mail : bidon@ici.fr

Comme le champ est à présent conforme syntaxiquement, il est affiché.

Quelques exemple de filtre à compléter avec des drapeaux éventuellement voir https://www.php.net/manual/fr/filter.filters.validate.php

FILTER_VALIDATE_BOOLEAN	Valide si la variable contient un booléen
FILTER_VALIDATE_DOMAIN	Valide si la variable contient un nom de domaine
FILTER_VALIDATE_EMAIL	Valide si la variable contient une adresse émail correctement formatée
FILTER_VALIDATE_FLOAT	Valide si la variable contient un nombre réel
FILTER_VALIDATE_INT	Valide si la variable contient un entier
FILTER_VALIDATE_IP	Valide si la variable contient une adresse IP
FILTER_VALIDATE_MAC	Valide si la variable contient une adresse mac
FILTER_VALIDATE_REGEXP	Valide si la variable contient est conforme à une expression régulière
FILTER_VALIDATE_URL	Valide si la variable contient une URL valide

4.4 La gestion des éléments d'un formulaire

Exemple de récupération des données en provenance d'un formulaire incluant un élément **select multiple** :

```
<html>
   <body>
      <form action="listemul.php" method="post">
         <select name="choix[]" multiple="true">
            <option value="cd"> CD </option>
            <option value="cdr"> CDR </option>
            <option value="dvdr"> DVDR </option>
            <option value="cdrw"> CDRW </option>
         </select>
         <input type="submit" value="go" />
      </form>
   </body>
</html>
```

listemul.php

```
<html>
   <body>
      <?php
         $choix=$_POST["choix"];
         $i=1;
         foreach( $choix as $valeurchoix ) {
            echo "choix $i : $valeurchoix <br />";
            $i++;
         }
      ?>
   </body>
</html>
```

Vous pouvez faire la même chose avec des **checkbox** :

```
<html>
   <body>
      <form action="listemul.php" method="post">
         <input type=checkbox name="checkchoix[]" value="cd" /> CD<br />
         <input type=checkbox name="checkchoix[]" value="cdr" /> CDR<br />
         <input type=checkbox name="checkchoix[]" value="dvdr" /> DVDR<br />
         <input type=checkbox name="checkchoix[]" value="cdrw" /> CDRW<br />
         <input type="submit" value="go" />
      </form>
   </body>
</html>
```

listemul.php

```
<html>
   <body>
      <?php
         $i=1;
         foreach( $checkchoix as $valeurchoix )
         {
            echo "choix $i : $valeurchoix <br />";
            $i++;
         }
      ?>
   </body>
</html>
```

Formulaire de téléchargement de fichier :

```
<html>
   <body>
      <form enctype="multipart/form-data" action="gestionupload.php" method="post">
         <input type="file" name="fichierAuploader" />
         <input type="submit" value="telechargement" />
      </form>
   </body>
</html>
```

Pour le téléchargement de fichier, il existe des variables globales:

ELEMENT	CONTIENT	EXEMPLE
$_FILES['fichierAuploader']['name']	Nom du fichier	Toto.png
$_FILES['fichierAuploader']['size']	Taille du fichier	368
$_FILES['fichierAuploader']['type']	Type du fichier	Image/png
$_FILES['fichierAuploader']['tmp_name']	Nom temporaire du fichier qui sera chargé sur la machine serveur	tmp24654654sdfs
$_FILES['fichierAuploader']['error']	code d'erreur	UPLOAD_ERR_OK (0) → pas d'erreur

gestionupload.php

```
<?php
   $uploaddir = '/var/www/uploads/';
   $uploadfile = $uploaddir . basename($_FILES['fichierAuploader']['name']);
   if (move_uploaded_file($_FILES['fichierAuploader']['tmp_name'], $uploadfile))
   {
       echo "Le fichier est valide, et a été téléchargé avec succès.
       Voici plus d'informations :\n";
   }
  else
   {
     echo "Attaque potentielle par téléchargement de fichiers.
            Voici plus d'informations :\n";
   }
   echo 'Voici quelques informations de débogage :';
   print_r($_FILES);
?>
```

4.5 Les sessions

Une façon de pouvoir conserver une variable d'une page à une autre est d'utiliser des variables de sessions.

Chaque visiteur accédant à votre page web se voit assigner un identifiant unique, appelé '**identifiant de session**'. Il peut être stocké soit dans un cookie, soit propagé dans l'URL.

Le support des sessions vous permet d'enregistrer un nombre illimité de variables qui doivent être préservées entre les requêtes. Lorsqu'un visiteur accède à votre site, PHP va vérifier automatiquement (si **session.auto_start** est activé1) ou sur demande (explicitement avec **session_start()** ou implicitement avec **session_register()**) s'il existe une session du même nom. Si c'est le cas, l'environnement précédemment sauvé sera recréé.

Exemple 1. Enregistrer une variable avec $_SESSION.

```
<?php
   session_start();
   if (!isset($_SESSION['compteur']))
   {
     $_SESSION['compteur'] = 0;
   }
   else
   {
     $_SESSION['compteur']++;
   }
?>
```

Exemple 2. Retirer une variable de session avec $_SESSION et register_globals inactif.

```
<?php
   session_start();
   unset($_SESSION['compteur']);
?>
```

4.6 Les cookies

Une façon de pouvoir conserver des informations entre 2 connexions à un site est de placer un petit fichier sur la machine cliente. C'est le principe des cookies.

Il est possible de créer un cookie grâce à la fonction **setcookie().**

bool setcookie (string name [, string value [, int expire [, string path [, string domain [, int secure]]]]])

setcookie() définit un cookie qui sera envoyé avec le reste des en-têtes. Les cookies doivent passer avant tout autre en-tête (c'est une restriction des cookies, pas de PHP). Cela vous impose d'appeler cette fonction avant toute balise <HTML> ou <HEAD>.

Tous les arguments sauf **name** (nom) sont optionnels. Si seul le nom est présent, le cookie portant ce nom sera supprimé du navigateur de l'internaute. Vous pouvez aussi utiliser une chaîne vide comme valeur, pour ignorer un argument. Le paramètre expire est un **timestamp** UNIX, du même genre que celui retourné par **time()** ou **mktime()**. Le paramètre **secure** indique que le cookie doit être uniquement transmis à travers une connexion HTTPS sécurisée.

Une fois que le cookie a été placé, il est accessible dans les variables globales **$_COOKIE**. Les valeurs de cookies existent aussi dans la variable **$_REQUEST**.

Erreurs communes :

- Les cookies ne seront accessibles qu'au chargement de la prochaine page, ou au rechargement de la page courante.
- Les cookies doivent être effacés avec les mêmes paramètres que ceux utilisés lors de leur création.

Exemple 1. Exemples avec setcookie()

```
<?php
 setcookie("TestCookie","Valeur de test");
 setcookie("TestCookie",$value,time()+3600);  /* expire dans une heure */
 setcookie("TestCookie",$value,time()+3600,"/~rasmus/",".utoronto.ca",1);
?>
```

Notez que la partie "valeur" du cookie sera automatiquement encodée URL lorsque vous envoyez le cookie et, lorsque vous le recevez, il sera automatiquement décodé, et affecté à la variable du même nom que le cookie. Pour voir le résultat, essayez les scripts suivants :

Exemple 2. Affectation des valeurs de cookie

```
<?php
   // Afficher un cookie
   echo $_COOKIE["TestCookie"];

   // Une autre méthode pour afficher tous les cookies
   print_r($_COOKIE);
?>
```

Lorsque vous effacez un cookie, vous devriez toujours vous assurer que sa date d'expiration est déjà passée, pour déclencher le mécanisme de votre navigateur.

Voici comment procéder :

Exemple 3. Exemple d'effacement de cookies avec setcookie()

```
<?php
   // utilisation de la date moins une heure
   setcookie ("TestCookie", "", time() - 3600);
   setcookie ("TestCookie", "", time() - 3600, "/~rasmus/", ".exemple.com", 1);
?>
```

Vous pouvez aussi utiliser les cookies avec des tableaux, en utilisant la notation des tableaux. Cela a pour effet de créer autant de cookies que votre tableau a d'éléments, mais lorsque les cookies seront reçus par PHP, les valeurs seront placées dans un tableau :

Exemple 4. Utilisation des tableaux avec setcookie()

```
<?php
   setcookie( "cookie[three]", "cookiethree" );
   setcookie( "cookie[two]", "cookietwo" );
   setcookie( "cookie[one]", "cookieone" );
   if ( isset( $cookie ) )
   {
      while( list( $name, $value ) = each( $cookie ) )
      {
         echo "$name == $value<br/>\n";
      }
   }
?>
```

Note : Pour plus d'informations sur les cookies, voyez les spécifications de cookies de Netscape à **http://www.netscape.com/newsref/std/cookie_spec.html** et RFC 2965.

Vous pourrez noter que le paramètre expire prend un timestamp unique, et non pas la date au format Jour, **JJ-Mois-AAAA HH:MM:SS GMT**, car PHP fait la conversion en interne.

Le navigateur « Microsoft Internet Explorer 4 » utilisé avec le Service Pack 1 ne gère pas bien les cookies qui possèdent un paramètre ***path***.

Netscape Communicator 4.05 et Microsoft Internet Explorer 3.x semblent ne pas gérer correctement les cookies lorsque ***path*** et ***time*** ne sont pas fournis.

5 Les fichiers

Pour alléger les scripts PHP, il est possible de placer les scripts contenant des fonctions, par exemple, dans un fichier, que l'on va inclure dans un autre fichier php.

Cela se fait grâce à la fonction ***include() ou include_once() ou require() ou require_once;***

Exemple :

desfonctions.inc

```
<?php
   function un() {
       echo "un<br/>";
   }
   function deux() {
       echo "deux<br/>";
   }
   function trois() {
       echo "trois<br/>";
   }
?>
```

monscriptutilisateur.php

```
<?php
   require_once("desfonctions.inc");
   echo "<html><body>";
   un();
   deux();
   trois();
   echo "</body></html>";
?>
```

La fonction ***include_once*** ou ***require_once*** évite les problèmes d'inclusion multiple.

toto.php

```
<?php
   include("desfonctions.inc");
?>
```

titi.php

```
<?php
   include("toto.php");
   include("desfonctions.inc");
?>
```

L'exécution du fichier titi.php provoquera des erreurs d'inclusion multiples.

Fatal error: Cannot redeclare un() (previously declared in desfonctions.inc:3) in **desfonctions.php** on line **2**

En remplaçant les ***include*** par les ***include_once*** ce problème est résolu.

6 Les images dynamiques

En php, il est possible de créer des images de façon dynamique.

Exemple :

img.php

```
<?php
   header("Content-type: image/png");
   $image = imagecreate( 200, 200);
   imagepng($image);
?>
```

dynimage.html

```
<html>
   <body>
      <img src="img.php" alt="une image dynamique" />
   </body>
</html>
```

Cela va afficher une image vide de 200x200.

Pour créer une image avec un fond noir, et une phrase écrite en bleu, on peut écrire le code suivant :

```
<?php
   header ("Content-type: image/png");
   $im = @imagecreate (50, 100) or die ("Impossible d'initialiser la librairie GD");
   $background_color = imagecolorallocate ($im, 0, 0, 0);
    $text_color = imagecolorallocate ($im, 0, 0, 255);
   imagestring ($im, 1, 5, 5,  "Juste du texte bleu", $text_color);
   imagepng ($im);
?>
```

Il est possible de faire des rectangles, des arcs, des cercles, des polygones, etc. Faites une recherche sur www.php.net avec comme mot clef ***imagecreate.***

7 PHP/apache versus ASP/IIS

le "couple" apache/PHP n'est pas le seul utilisé pour le développement de site WEB dynamique.

Il existe un autre "couple célèbre", il s'agit de IIS/ASP : **Internal Internet Server & Active Server Page**. Ses deux produits sont développés et diffusés par Microsoft.

Si au début de PHP, il n'y avait aucune comparaison possible entre les 2 produits, ce n'est aujourd'hui, plus le cas. En effet, chacun offre à présent à peu de choses près les mêmes fonctionnalités, même si Microsoft propose un environnement de développement plus élaboré grâce, entre autres, à ASP.NET.

8 Conclusion

Le langage PHP est utilisé pour la construction de page web dynamique. Il permet d'interagir facilement avec des formulaire des base de données. Il ne permet cependant pas d'accéder à des périphériques de la machine serveur (imprimante, caméra, capteurs etc.)

Si l'on s'aperçoit que le langage par lui-même n'est pas très compliqué à utiliser, il s'avère que la difficulté réside désormais plus dans l'architecture et l'agencement des pages d'un site WEB, plutôt que dans la complexité du langage.

Une fois que l'on sait quelles informations on souhaite publier, et sous quelle forme, le développement technique d'un site WEB est bien plus simple.

Le développement d'un site WEB n'est pas que l'affaire de technicien.

Pour réaliser un bon site WEB, il faut des infographistes, des commerciaux, des chefs de projet, des informaticiens, et surtout, des personnes qui ont des informations à faire passer (information, produit -> site marchand, etc.) sur le WEB.

Le langage PHP est toujours en évolution, dernière version 8.4 sortie en novembre 2024

Lien vers la documentation officielle complète : https://www.php.net/manual/fr/