

Institut Supérieur du Numérique

Cours Langages Web

PHP



Internet et les pages web

- ❖ HTML : Conception de pages destinées à être publiées sur Internet
- ❖ Page HTML : Contient le texte à afficher et des instructions de mise en page
- ❖ HTML est un langage de description de page et non pas un langage de programmation
 - pas d'instructions de calcul ou pour faire des traitements suivant des conditions
- ❖ Des sites de plus en plus riches en informations :
 - Nécessité croissante d'améliorer le contenu de sites
 - Mises à jour manuelles trop complexes
 - Pourquoi ne pas automatiser les mises à jour ?

Pages web statiques

- ❖ Fonctionnement : Leurs contenus ne changent ni en fonction du demandeur ni en fonction d'autres paramètres éventuellement inclus dans la requête adressée au serveur. Toujours le même résultat.
 - Rôle du serveur : localiser le fichier correspondant au document demandé et répond au navigateur en lui envoyant le contenu de ce fichier
- ❖ limites : Besoin de réponses spécifiques
 - passage de pages statiques à pages dynamiques

Création des pages web dynamiques

❖ PHP : Hypertext PreProcessor

- ✓ Connaît un succès toujours croissant sur le Web et se positionne comme un rival important pour ASP
- ✓ Combiné avec le serveur Web Apache et la base de données MySQL, PHP offre une solution particulièrement robuste, stable et efficace
- ✓ Gratuité : Tous les logiciels sont issus du monde des logiciels libres (Open Source).

Histoire et Origine

- ❖ Première version de PHP a été mis au point au début d'automne par Rasmus Lerdorf en 1994
 - Version appelée à l'époque Personal Home Pages
 - Pour conserver la trace des utilisateurs venant consulter son CV sur son site, grâce à l'accès à une base de données par l'intermédiaire de requêtes SQL
- ❖ La version 3.0 de PHP fut disponible le 6 juin 1998
- ❖ A la fin de l'année 1999, une version PHP4 est apparue
- ❖ En 2001 cinq millions de domaines utilisent PHP, trois fois plus que l'année 2000

PHP

- ❖ Un langage de scripts permettant la création d'applications Web
- ❖ Indépendant de la plate-forme utilisée puisqu'il est exécuté côté serveur et non côté client.
- ❖ La syntaxe du langage provient de celles du langage C, du Perl et de Java.
- ❖ Ses principaux atouts sont:
 - La gratuité et la disponibilité du code source
 - La simplicité d'écriture de scripts
 - La possibilité d'inclure le script PHP au sein d'une page HTML
 - La simplicité d'interfaçage avec des bases de données
 - L'intégration au sein de nombreux serveurs web (Apache ...)

Intégration PHP et HTML

- ❖ Les scripts PHP sont généralement intégrés dans le code d'un document HTML
- ❖ L'intégration nécessite l'utilisation de balises
 - avec le style xml : `<? ligne de code PHP ?>`
 - Avec le style html : `<?php ligne de code PHP ?>`
 - avec le style JavaScript :
`<script language=«php»> ligne de code PHP </script>`

Intégration PHP et HTML

❖ Intégration directe avec HTML

```
<html>
<head> <title> Mon script PHP
</title> </head>
<body>
//ligne de code HTML
< ?php
//ligne de code PHP
?>
//ligne de code HTML ....
</body>
</html>
```

Intégration PHP et HTML

❖ Simple exemple

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title> Bonjour depuis PHP </title>
</head>
<body>
<?php echo 'Bonjour généré
dynamiquement en PHP !'; ?>
</body>
</html>
```

Les institutions d'affichage

❖ La fonction echo : **echo Expression**;

- echo "Chaine de caracteres";
- echo (1+2)*87;

❖ La fonction print : **print(expression)**;

- print("Chaine de caracteres");
- print ((1+2)*87);

❖ La fonction printf : **printf (chaîne formatée)**;

- \$Perimetre = 1;
- printf ("Le périmètre du cercle est %d",\$Perimetre);

echo peut prendre plus d'un paramètre, print ne prend qu'un seul paramètre.

Syntaxe de base

❖ Typologie :

- Toute instruction se termine par un point-virgule

❖ Les commentaires :

- `/* Voici un commentaire! */`
- `// un commentaire sur une ligne`

❖ Les constantes :

- ✓ `Define("nom_constante", valeur_constante )`
 - `define ("ma_const", "Vive PHP4");`
 - `define ("an", 2002) ;`

Les variables

❖ Principe :

- Commencent par le caractère \$
- N'ont pas besoin d'être déclarées

```
<?php
    $txt = "Hello world!";
    $x = 5;
    $y = 10.5;
    echo "Teste $txt!";
    echo "<br>" ;
    echo "Teste " . $txt . "!";
?>
```

Portée des variables en PHP

- ❖ La portée d'une variable est la partie du script où la variable peut être référencée/utilisée.
- ❖ PHP a trois portées variables différentes:
 - ✓ Local
 - ✓ Global
 - ✓ Statique
- ❖ Visibilité des variables :
 - ✓ Variable locale :

Visible uniquement à l'intérieur d'un contexte d'utilisation
 - Variable globale :

Visible dans tout le script

Les variables

❖ **Les variables dynamiques** : Permettent d'affecter un nom différent à une autre variable

- `$nom_variable = 'nom_var';`
- `$$nom_variable = valeur; // équivaut à $nom_var = valeur;`

❖ Affectation par valeur et par référence

- Affectation par valeur : `$b = $a`
- Affectation par (référence) variable : `$c = &$a`

```
<?php
    $a = 8;$b = $a;$a = 6;
    echo $b;
    echo "<br>$a";
?>
```

```
<?php
    $a = 8;$b = &$a;$a = 6;
    echo $b;
    echo "<br>$a";
?>
```

Les variables prédéfinies

❖ Variables prédéfinies :

- Les variables d'environnement dépendant du client

Variable	Description
<code>\$_SERVER["HTTP_HOST"]</code>	Nom d'hôte de la machine du client (associée à l'adresse IP)
<code>\$_SERVER["HTTP_REFERER"]</code>	URL de la page qui a appelé le script PHP
<code>\$_SERVER["HTTP_ACCEPT_LANGUAGE"]</code>	Langue utilisée par le serveur (par défaut en-us)
<code>\$_SERVER["HTTP_ACCEPT"]</code>	Types MIME reconnus par le serveur (séparés par des virgules)
<code>\$_SERVER["CONTENT_TYPE"]</code>	Type de données contenu présent dans le corps de la requête. Il s'agit du type MIME des données
<code>\$_SERVER["REMOTE_ADDR"]</code>	L'adresse IP du client appelant le script CGI
<code>\$_SERVER["PHP_SELF"]</code>	Nom du script PHP

Les variables prédéfinies

❖ Variables prédéfinies :

- Les variables d'environnement dépendant du serveur

Variable	Description
<code>\$_SERVER["SERVER_NAME"]</code>	Le nom du serveur
<code>\$_SERVER["HTTP_HOST"]</code>	Nom de domaine du serveur
<code>\$_SERVER["SERVER_ADDR"]</code>	Adresse IP du serveur
<code>\$_SERVER["SERVER_PROTOCOL"]</code>	Nom et version du protocole utilisé pour envoyer la requête au script PHP
<code>\$_SERVER["DATE_GMT"]</code>	Date actuelle au format GMT
<code>\$_SERVER["DATE_LOCAL"]</code>	Date actuelle au format local
<code>\$_SERVER["\$DOCUMENT_ROOT"]</code>	Racine des documents Web sur le serveur

Les types de données

❖ Principe :

- Pas besoin d'affecter un type à une variable avant de l'utiliser
- La même variable peut changer de type en cours de script
- Les variables issues de l'envoi des données d'un formulaire sont de type string

❖ Les différents types de données :

- Les entiers : le type **Integer**
- Les flottants : le type **Double**
- Les tableaux : le type **array**
- Les chaînes de caractères : le type **string**
- Les objets

Les types de données

❖ Le transtypage :

- La fonction **settype()** permet de convertir le type auquel appartient une variable

```
<?php
    $nbre=10;
    Settype($nbre, "double");
    Echo " la variable $nbre est de type " , gettype($nbre);
?>
```

Les types de données

❖ Le transtypage :

- Transtypage explicite : le cast

(int), (integer) ; (double), (float); (string); (array); (object)

```
<?php
$var=" 100 FRF ";
Echo " pour commencer, le type de la variable $var est " , gettype($var) ;
$var =(double) $var;
Echo "<br> Après le cast, le type de la variable $var est " , gettype($var);
Echo "<br> et a la valeur $var ";
? >
```

Les types de données

❖ Détermination du type de données :

- `gettype()`
- `settype()`
- `is_long()`
- `is_double()`
- `is_string()`
- `is_array()`
- `is_object()`
- `is_bool()`

Les types de données

- ❖ `var_dump()` affiche les informations structurées d'une variable, y compris son type et sa valeur.

```
<?php
    $b = 3.1;
    $c = true;
    var_dump($b);
    var_dump($c);
    var_dump($b, $c);
    var_dump(is_double($b));
    var_dump(is_int($b));
?>
```

Les opérateurs

❖ Les opérateurs :

- les opérateurs de calcul : +, -, *, /
- les opérateurs d'assignation : +=, -=, *=, /=
- les opérateurs d'incrémentement : ++, --
- les opérateurs de comparaison : ==, <, >, <=, >=, !=
- les opérateurs logiques (|| ou OR), (&& ou AND), XOR, !

Les instructions conditionnelles

❖ L'instruction if :

- **if (condition réalisée)** { liste d'instructions }

❖ L'instruction if ... Else

- **if (condition réalisée)** { liste d'instructions }
else { autre série d'instructions }

❖ L'instruction if ... elseif ... Else

- **if (condition réalisée)** { liste d'instructions }
elseif (autre condition) { autre série d'instructions }
else { série d'instructions }

Les instructions conditionnelles

❖ Exemple

```
<?php
    $t = date("H");
    if ($t < "10") {
        echo "Have a good morning!";
    } elseif ($t < "20") {
        echo "Have a good day!";
    } else {
        echo "Have a good night!";
    }
?>
```

Les instructions conditionnelles

❖ L'instruction switch :

```
switch (Variable) {
```

```
case Valeur1: Liste d'instructions break;
```

```
case Valeur2: Liste d'instructions break;
```

```
case Valeurs...: Liste d'instructions break;
```

```
default: Liste d'instructions;
```

```
}
```

Les instructions conditionnelles

```
<?php
    $favcolor = "red";
    switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, nor green!";}
?>
```

Les boucles

❖ La boucle for

- `for ($i=1; $i<6; $i++) { echo "$i<br>"; }`

❖ La boucle while

- `While(condition) {bloc d'instructions ;}`

```
<?php
    $x = 1;

    while($x <= 5) {
        echo "The number is: $x <br>";
        $x++;
    }

?>
```

Les boucles

❖ La boucle do...while

- Do {bloc d'instructions ;}while(condition) ;

```
<?php
    $x = 1;
    do {
        echo "The number is: $x <br>";
        $x++;
    } while ($x <= 5);

?>
```

Les boucles

❖ La boucle foreach (PHP4)

- Foreach (\$tableau as \$valeur) {insts utilisant \$valeur ;}

```
<?php
    $colors = array("red", "green", "blue", "yellow");

    foreach ($colors as $value) {
        echo "$value <br>";
    }
?>
```

Intégration PHP et HTML

- ❖ Inclure un fichier PHP dans un fichier HTML : include()

Fichier Principal

```
<html>
<head>
<title> Fichier d'appel </title>
</head>
<body>
<?php $salut = "BONJOUR" ;
include "information.php" ;
?>
</body>
</html>
```

Fichier à inclure : information

```
<?php
$chaine = $salut. " , C'est PHP " ;
echo "<table border= '3'>
    <tr><td width='100%'>
        <h2> $chaine</h2>
    </td></tr>
</table>";
?>
```

Les fonctions

❖ Déclaration et appel d'une fonction

```
Function nom_fonction($arg1, $arg2, ...$argn)
{
déclaration des variables ;
bloc d'instructions ;
//fin du corps de la fonction
return $resultat ;
}
```

❖ Fonction avec nombre d'arguments inconnu

- **func_num_args()** : fournit le nombre d'arguments qui ont été passés lors de l'appel de la fonction
- **func_get_arg(\$i)** : retourne la valeur de la variable située à la position \$i dans la liste des arguments passés en paramètres, Ces arguments sont numérotés à partir de 0

Les fonctions

❖ Fonction avec nombre d'arguments inconnu :

```
<?php
function produit()
{
    $nbarg = func_num_args() ;
    $prod=1 ;
    // la fonction produit a ici $nbarg arguments
    for ($i=0 ; $i <$nbarg ; $i++)
    { $prod *= func_get_arg($i); }
    return $prod;
}
echo "le produit est : ", produit (3, 77, 10, 5, 81, 9), "<br />" ;
// affiche le produit est 8 419 950
?>
```

Les fonctions

❖ Passage de paramètre par référence :

- Pour passer une variable par référence, il faut que son nom soit précédé du symbole & (exemple &\$a)

```
<?
function dire_texte($qui, &$texte)
{
    $texte = "Bienvenue $qui";
}
$chaine = "Bonjour ";
dire_texte("cher etudiant", $chaine);
echo $chaine;
// affiche "Bienvenue cher etudiant"
?>
```

❖ L'appel récursif : PHP admet les appels récursifs de fonctions

Les fonctions

❖ Variables locales et variables globales :

- variables en PHP : global, static, local
- toute variable déclarée en dehors d'une fonction est globale
- utiliser une variable globale dans une fonction
- Pour conserver la valeur acquise par une variable entre deux appels de la même fonction : l'instruction static.
- Les variables statiques restent locales à la fonction et ne sont pas réutilisables à l'extérieur.

Les fonctions

❖ Les variables statiques :

```
<?php
function cumul ($prix) {
    static $cumul = 0 ;
    static $i = 1 ;
    echo "Total des achats $i = ";
    $cumul += $prix;
    $i++ ;
    return $cumul ; }
echo cumul (175), "<br />" ;
echo cumul (65), "<br />" ;
echo cumul (69), "<br />" ;
?>
```

Les tableaux

- ❖ Création à l'aide de la fonction `array()`
- ❖ Uniquement des tableaux à une dimension
- ❖ Les éléments d'un tableau peuvent pointer vers d'autres tableaux
- ❖ Les éléments d'un tableau peuvent appartenir à des types distincts
- ❖ L'index d'un tableau en PHP commence de 0
- ❖ Pas de limites supérieures pour les tableaux
- ❖ La fonction `count()` pour avoir le nombre d'éléments d'un tableau

Les tableaux indicés et les tableaux associatifs

❖ Tableau indicé

- Accéder aux éléments par l'intermédiaire de numéros

```
$tableau[indice] = valeur;  
$jour[3] = "Mercredi";  
$note[0] = 20;  
$tableau = array(valeur0, valeur1,..., valeurN);  
$jour = array("Dimanche", "Lundi", "Mardi",  
"Mercredi", "Jeudi", "Vendredi", "Samedi");  
$note = array(20, 15, 12.6, 17, 10, 20, 11, 18, 19);  
$variable = $tableau[indice];  
$JJ = $jour[6]; // affecte "Samedi" à $JJ  
echo $note[1] + $note[5];
```

Les tableaux indicés et les tableaux associatifs

❖ Tableau indicé

```
$jour = array("Dimanche", "Lundi", "Mardi",  
"Mercredi", "Jeudi", "Vendredi", "Samedi");
```

Tableau des jours :

```
array (size=7)  
0 => string 'Dimanche' (length=8)  
1 => string 'Lundi' (length=5)  
2 => string 'Mardi' (length=5)  
3 => string 'Mercredi' (length=8)  
4 => string 'Jeudi' (length=5)  
5 => string 'Vendredi' (length=8)  
6 => string 'Samedi' (length=6)
```

```
$note = array(20, 15, 12.6, 17, 10, 20, 11, 18, 19);
```

Tableau des notes :

```
array (size=9)  
0 => int 20  
1 => int 15  
2 => float 12.6  
3 => int 17  
4 => int 10  
5 => int 20  
6 => int 11  
7 => int 18  
8 => int 19
```

Les tableaux indicés et les tableaux associatifs

❖ Tableau associatif (ou table de hachage)

- Les éléments sont référencés par des chaînes de caractères associatives en guise de nom: la clé d'index

```
$tableau["indice"] = valeur;
```

```
$jour["Dimanche"] = 7 ;
```

```
$jour["Mercredi"] = "Le jour des enfants" ;
```

```
$tableau = array(ind0 => val0, ind1 => val1,..., indN => valN);
```

```
$jour = array("Dimanche" => 1, "Lundi" => 2, "Mardi" => 3,  
"Mercredi" => 4, "Jeudi" => 5, "Vendredi" => 6, "Samedi" => 7);
```

```
$variable = $tableau["indice"];
```

```
$JJ = $jour["Vendredi"]; //affecte 6 à $JJ
```

```
echo $jour["Lundi"]; //retourne la valeur 2
```

Les tableaux indicés et les tableaux associatifs

❖ Tableau associatif (ou table de hachage)

```
$jour = array("Dimanche" => 1, "Lundi" => 2, "Mardi" => 3,  
"Mercredi" => 4, "Jeudi" => 5, "Vendredi" => 6, "Samedi" => 7);
```

Tableau des jours :

```
array (size=7)  
    'Dimanche' => int 1  
    'Lundi' => int 2  
    'Mardi' => int 3  
    'Mercredi' => int 4  
    'Jeudi' => int 5  
    'Vendredi' => int 6  
    'Samedi' => int 7
```

Tableaux multidimensionnels

- ❖ Pas d'outils pour créer directement des tableaux multidimensionnels
- ❖ L'imbrication des tableaux est possible

```
$tab1 = array(Val0, Val1,..., ValN);  
$tab2 = array(Val0, Val1,..., ValN);  
// Création d'un tableau à deux dimensions  
$tableau = array($tab1, $tab2);  
$mois = array("Janvier", "Février", "Mars", "Avril", "Mai", "Juin", "Juillet", "Août",  
"Septembre", "Octobre", "Novembre", "Décembre");  
$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi");  
$element_date = array($mois, $jour);  
  
$variable = $tableau[indice][indice];  
$MM = $element_date[0][0]; //affecte "Janvier" à $MM  
echo $element_date[1][5] . " 7 " . $element_date[0][2] . "2002"; // retourne "Jeudi 7 Mars  
2002"
```

Tableaux multidimensionnels

```
$mois = array("Janvier", "Février",  
"Mars", "Avril", "Mai", "Juin", "Juillet",  
"Août", "Septembre", "Octobre",  
"Novembre", "Décembre");
```

```
$jour = array("Dimanche", "Lundi",  
"Mardi", "Mercredi", "Jeudi", "Vendredi",  
"Samedi");
```

```
$element_date = array($mois, $jour);
```

Tableaux multidimensionnels :

```
array (size=2)
```

```
0 =>
```

```
array (size=12)
```

```
0 => string 'Janvier' (length=7)
```

```
1 => string 'Février' (length=8)
```

```
2 => string 'Mars' (length=4)
```

```
3 => string 'Avril' (length=5)
```

```
4 => string 'Mai' (length=3)
```

```
5 => string 'Juin' (length=4)
```

```
6 => string 'Juillet' (length=7)
```

```
7 => string 'Août' (length=5)
```

```
8 => string 'Septembre' (length=9)
```

```
9 => string 'Octobre' (length=7)
```

```
10 => string 'Novembre' (length=8)
```

```
11 => string 'Décembre' (length=9)
```

```
1 =>
```

```
array (size=7)
```

```
0 => string 'Dimanche' (length=8)
```

```
1 => string 'Lundi' (length=5)
```

```
2 => string 'Mardi' (length=5)
```

```
3 => string 'Mercredi' (length=8)
```

```
4 => string 'Jeudi' (length=5)
```

```
5 => string 'Vendredi' (length=8)
```

```
6 => string 'Samedi' (length=6)
```

Lecture des éléments d'un tableau

❖ Avec une boucle for :

```
for ($i=0; $i<count($tab) ; $i++){  
    if ($tab[$i]== "a") {echo $tab[$i], "<br />";}  
}
```

❖ Avec une boucle while :

```
$i=0;  
while ($i <count($tab) ){  
    echo $tab[$i], "<br /> ";  
    $i++;  
}
```

Lecture des éléments d'un tableau

❖ Avec la boucle foreach :

```
$jour = array("Dimanche", "Lundi", "Mardi",  
"Mercredi", "Jeudi", "Vendredi", "Samedi");  
$i = 0;  
foreach($jour as $JJ) {  
    echo "La cellule n° ". $i . " : " . $JJ . "<br>";  
    $i++;  
}
```

Lecture des éléments d'un tableau

- ❖ Parcours d'un tableau associatif : Réalisable en ajoutant avant la variable \$valeur, la clé associée

```
$tableau = array(clé1 => val1, clé2 => val2, ..., cléN => valN);  
foreach($tableau as $clé => $valeur) {  
    echo "Valeur ($clé): $valeur"; }
```

```
$jour = array("Dimanche" => 7, "Lundi" => 1, "Mardi" => 2,  
    "Mercredi" => 3, "Jeudi" => 4, "Vendredi" => 5, "Samedi" => 6);  
foreach($jour as $sJJ => $nJJ) {  
    echo "Le jour de la semaine n° ". $nJJ . " : " . $sJJ . "<br>"; }
```

Date et heure PHP

❖ La fonction PHP Date() est utilisée pour formater une date et/ou une heure.

❖ *Exemple :*

```
<?php
    echo "Today is " . date("Y/m/d") . "<br>";
    echo "Today is " . date("Y.m.d") . "<br>";
    echo "Today is " . date("Y-m-d") . "<br>";
    echo "Today is " . date("l") . "<br>";
    echo date("d") . "<br>";
    echo date("m") . "<br>";
    echo date("Y") . "<br>";
?>
```

Today is 2022/03/24
Today is 2022.03.24
Today is 2022-03-24
Today is Thursday
24
03
2022

Date et heure PHP

❖ L'exemple ci-dessous affiche l'heure actuelle dans le format spécifié

❖ *Exemple :*

```
<?php  
    echo "The time is " . date("h:i:sa");  
?>
```

PHP et les formulaires

❖ Formulaire HTML

- ✓ Retourne des informations saisies par un utilisateur vers une application serveur
- ✓ La création d'un formulaire nécessite la connaissance de quelques balises HTML indispensables :
 - Structure : un formulaire commence toujours par la balise `<form>` et se termine par la balise `</form>`
 - Champ de saisie de text en ligne :
`<input type = "text" name ="nom_du_champ" value="chaîne">`
 - Boutons d'envoi et d'effacement :
`<input type=" submit " value = "Envoyer">`
`<input type = "reset" name ="efface" value = "Effacer">`
 - Case à cocher et bouton radio :
`<input type = "checkbox" name ="case1" value="valeur_case">`
`<input type = "radio" name ="radio1" value ="valeur_radio">`

Méthodes d'envoi get et post

- ❖ Transmission selon une des deux méthodes d'envoi GET ou POST
 - ✓ La méthode **GET** place les informations d'un formulaire directement à la suite de l'adresse URL de la page appelée.
 - Inconvénients :
 - rendre visibles les données dans la barre d'adresse du navigateur.
 - De plus, la longueur totale est limitée à 255 caractères, ce qui rend impossible la transmission d'un volume de données important
 - ✓ La méthode **POST** regroupe les informations dans l'entête d'une requête HTTP
 - Assure une confidentialité efficace des données

Afficher des données issues d'un formulaire

- ❖ Un simple formulaire HTML avec la méthode **GET** :

```
<form action="action.php" method="get">  
  <p>Votre nom : <input type="text" name="nom" /></p>  
  <p>Votre âge : <input type="text" name="age" /></p>  
  <p><input type="submit" value="OK"></p>  
</form>
```

- ❖ Afficher des données issues d'un formulaire avec \$_GET:

```
Bonjour, <?php echo $_GET['nom']; ?>.  
Tu as <?php echo (int)$_GET['age']; ?> ans.
```

Afficher des données issues d'un formulaire

- ❖ Un simple formulaire HTML avec la méthode **POST** :

```
<form action="action.php" method="post">  
  <p>Votre nom : <input type="text" name="nom" /></p>  
  <p>Votre âge : <input type="text" name="age" /></p>  
  <p><input type="submit" value="OK"></p>  
</form>
```

- ❖ Afficher des données issues d'un formulaire avec `$_POST`

```
Bonjour, <?php echo $_POST['nom']; ?>.  
Tu as <?php echo (int)$_POST['age']; ?> ans.
```

La fonction htmlspecialchars

- ❖ La fonction htmlspecialchars() convertit les caractères spéciaux en entités HTML. Cela signifie qu'il remplacera les caractères HTML comme < et > par < and > Cela empêche les attaquants d'exploiter le code en injectant du code HTML ou Javascript attaques XSS (Cross-site Scripting) dans les formulaires.

```
Bonjour, <?php echo htmlspecialchars( $_POST['nom']); ?>.  
Tu as <?php echo (int)htmlspecialchars($_POST['age']); ?> ans.
```

- ❖ La fonction **trim ()** : retourne la chaîne, après avoir supprimé les caractères invisibles en début et fin de chaîne.

Afficher des données issues d'un formulaire

- ❖ La fonction **stripslashes()** : Supprime les antislashes d'une chaîne.
- ❖ **\$_GET** et **\$_POST** : sont des tableaux de données associatifs et superglobaux (c'est à dire visibles de partout dans le programme) et contient tous les couples variable/valeur.
- ❖ La fonction **test_input()** a pour rôle de nettoyer et de valider les données entrées par l'utilisateur via le formulaire.

```
<?php
function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;}
?>
```

Afficher des données issues d'un formulaire

- ❖ `$_SERVER` est un tableau de données associatifs contenant des informations créées par le serveur web.
 - `$_SERVER["PHP_SELF"]` est une variable super globale qui renvoie le nom de fichier du script en cours d'exécution.
 - `$_SERVER["REQUEST_METHOD"]` : retourne la méthode utilisée
 - La fonction `htmlspecialchars($_SERVER["PHP_SELF"])` est utilisée pour sécuriser l'URL du
- ❖ la fonction `test_input()` prépare les données entrées par l'utilisateur pour qu'elles soient sûres à utiliser, en les nettoyant et en les sécurisant contre les attaques XSS (Cross-Site Scripting) potentielles.

Afficher des données issues d'un formulaire

```
<?php $nom = $age = "";  
if ($_SERVER["REQUEST_METHOD"] == "POST") {  
    $nom = test_input($_POST["nom"]);  
    $age = test_input($_POST["age"]);  
}  
function test_input($data) {  
    $data = trim($data);  
    $data = stripslashes($data);  
    $data = htmlspecialchars($data);  
    return $data;}  
?  
<form action="<?php echo htmlentities($_SERVER["PHP_SELF"]); ?>"  
method="post">  
    Nom : <input type="text" name="nom"><br>  
    Age : <input type="text" name="age"><br>  
        <input type="submit">  
</form>  
<?php echo $nom; ?>  
<?php echo $age; ?> ans.
```

Les cookies

- ❖ Un cookie est souvent utilisé pour identifier un utilisateur. Un cookie est un petit fichier que le serveur intègre sur l'ordinateur de l'utilisateur. Chaque fois que le même ordinateur demande une page avec un navigateur, il enverra également le cookie.
 - ❖ Avec PHP, vous pouvez à la fois créer et récupérer des valeurs de cookie.
 - ❖ Pour des raisons de sécurité, les cookies ne peuvent être lus que par des pages issues du serveur qui les a créés
 - ❖ Les cookies permettent de conserver des renseignements utiles sur chaque utilisateur, et de les réutiliser lors de sa prochaine visite
- Exemple : personnaliser la page d'accueil ou les autres pages du site

Les cookies

❖ Un cookie est créé avec la fonction *setcookie()*

setcookie(nom_var, valeur_var, date_expiration, chemin);

- Nom_var : nom de la variable qui va stocker l'information sur le poste client et qui sera utilisée pour récupérer cette information dans la page qui lira le cookie. C'est la seule indication obligatoire pour un cookie.
- Valeur_var : valeur stockée dans la variable.
- Date_expiration : la date à laquelle le cookie ne sera plus lisible et sera effacé du poste client. on utilise en général la date du jour, définie avec la fonction *time()* à laquelle on ajoute la durée de validité désirée.
- Chemin : définit la destination (incluant les sous-répertoire) à laquelle le navigateur doit envoyer le cookie.

Les cookies

❖ Exemple :

```
<?php
    $cookie_name = "user";
    $cookie_value = "Sidi";
    setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
    // 86400 = 1 day
?>
```

- ❖ Le cookie expirera après 30 jours (86400 * 30). Le « / » signifie que le cookie est disponible sur l'ensemble du site Web (sinon, sélectionnez le répertoire que vous préférez).

Les cookies

- ❖ Les cookies sont accessibles dans le tableau associatif **\$_COOKIE**
- ❖ Nous utilisons également la fonction **isset()** pour savoir si le cookie est défini :

```
<?php
    $cookie_name = "user";
    $cookie_value = "Sidi";
    setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
    if(!isset($_COOKIE[$cookie_name])) {
        echo "Cookie named ".$cookie_name." is not set!";
    } else {
        echo "Cookie " . $cookie_name . " is set!<br>";
        echo "Value is: " . $_COOKIE[$cookie_name];
    }
?>
```

Les sessions

- ❖ Lorsque vous travaillez avec une application, vous l'ouvrez, faites quelques modifications, puis vous la fermez. Cela ressemble beaucoup à une session. L'ordinateur sait qui vous êtes.
- ❖ Il sait quand vous démarrez l'application et quand vous la terminez.
- ❖ Mais sur Internet, il y a un problème : le serveur Web ne sait pas qui vous êtes ni ce que vous faites, car l'adresse HTTP ne conserve pas l'état.

Les sessions

- ❖ Les variables de session résolvent ce problème en stockant les informations utilisateur à utiliser sur plusieurs pages (par exemple, nom d'utilisateur, couleur préférée, etc.). Par défaut, les variables de session durent jusqu'à ce que l'utilisateur ferme le navigateur.
- ❖ Alors, Les variables de session contiennent des informations sur un seul utilisateur et sont disponibles pour toutes les pages d'une seule application.

Les sessions

- ❖ Chaque visiteur en se connectant à un site reçoit un numéro d'identification dénommé identifiant de session (SID)
- ❖ La fonction **session_start()** se charge de générer automatiquement cet identifiant unique de session. Elle doit être placée au début de chaque page afin de démarrer ou de continuer une session.

```
<?php
    session_start();
    $Session_ID = session_id();
    $_SESSION["favcours"] = "programmation";
    $_SESSION["favlangage"] = "php";
    echo "Session variables are set : $Session_ID ";
?>
```

Obtenir les valeurs des variables de session

- ❖ Notez également que toutes les valeurs des variables de session sont stockées dans la variable globale `$_SESSION` :

```
<?php  
    session_start();  
    echo "Favorite cours is " . $_SESSION["favcours"] . "<br>";  
    echo "Favorite langage is " . $_SESSION["favlangage"] . ".";  
?>
```

Obtenir les valeurs des variables de session

- ❖ Une autre façon d'afficher toutes les valeurs de variable de session

```
<?php
    session_start();
    print_r($_SESSION);
?>
```

Modifier une variable de session

- ❖ Pour modifier une variable de session :

```
<?php  
    session_start();  
    $_SESSION["favcours"] ="database";  
    print_r($_SESSION);  
?>
```

Détruire une session PHP

- ❖ Pour supprimer toutes les variables de session globales et détruire la session, utilisez ***session_unset()*** et ***session_destroy()***

```
<?php
    session_start();
    // remove all session variables
    session_unset();
    // destroy the session
    session_destroy();
?>
```

Connexion à la base de données

- ❖ Avant de pouvoir accéder aux données de la base de données MySQL, nous devons pouvoir nous connecter au serveur:

```
<?php
    $servername = "localhost";
    $username = "root";
    $password = "";

    // Create connection
    $conn = mysqli_connect($servername, $username, $password);

    // Check connection
    if (!$conn) {
        die("Connection failed: " . mysqli_connect_error());
    }
    echo "Connected successfully";

?>
```

- ❖ La connexion sera fermée automatiquement à la fin du script. Pour fermer la connexion auparavant, utilisez **mysqli_close(\$conn)**

Créer une base de données MySQL

❖ Créer la base de données myDB :

```
$servername = "localhost";
$username = "root";
$password = "";

$conn = mysqli_connect($servername, $username, $password);
if(!$conn){
    die("Connection failed: ".mysqli_connect_error());
}
// Create database
$sql = "CREATE DATABASE myDB";
if(mysqli_query($conn, $sql)){
    echo "Database created successfully";
}else{
    echo "Error creating database: ". mysqli_error($conn);
}
mysqli_close($conn);
```

Créer une table

- ❖ Créer la table personne (id, firstname, lastname) :

```
<?php

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";

$conn = mysqli_connect($servername, $username, $password,$dbname);
if(!$conn){
    die("Connection failed: ".mysqli_connect_error());
}
// Create database
$sql = "CREATE TABLE Personne ( id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL)";
if(mysqli_query($conn, $sql)){
    echo "Table Personne created successfully";
}else{
    echo "Error creating table : ". mysqli_error($conn);
}
mysqli_close($conn);

?>
```

Insérer des données

❖ Insérer une ligne:

```
<?php
    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "myDB";
    $conn = mysqli_connect($servername, $username, $password,$dbname);
    if(!$conn){
        die("Connection failed: ".mysqli_connect_error());
    }
    $sql = "INSERT INTO personne (firstname, lastname) VALUES ('Med', 'Ali')";

    if (mysqli_query($conn, $sql)) {
        echo "New record created successfully";
    } else {
        echo "Error: " . $sql . "<br>" . mysqli_error($conn);
    }
    mysqli_close($conn);
?>
```

Insérer des données

❖ Insérer des lignes :

```
<?php
    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "myDB";
    $conn = mysqli_connect($servername, $username, $password,$dbname);
    if(!$conn){
        die("Connection failed: ".mysqli_connect_error());
    }
    $sql = "INSERT INTO personne (firstname, lastname) VALUES ('Ahmed', 'mohamed');";
    $sql .= "INSERT INTO personne (firstname, lastname) VALUES ('BA', 'Ali');";
    $sql .= "INSERT INTO personne (firstname, lastname) VALUES ('Mariem', 'Ahmed')";

    if (mysqli_multi_query($conn, $sql)) {
        echo "New records created successfully";
    } else {
        echo "Error: " . $sql . "<br>" . mysqli_error($conn);
    }
    mysqli_close($conn);
?>
```

Les requêtes préparé

- ❖ Une requête préparée ou requête paramétrable est utilisée pour exécuter la même requête plusieurs fois, avec une grande efficacité et protège des injections SQL.

```
<?php
    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "myDB";

    $conn = new mysqli($servername, $username, $password,$dbname);
    if ($conn->connect_error) { die("Connection failed: " . $conn->connect_error);}

    $stmt = $conn->prepare("INSERT INTO personne (firstname, lastname) VALUES (?, ?)");
    $stmt->bind_param("ss", $firstname, $lastname);
    $firstname = "Sidi";
    $lastname = "Ali";
    $stmt->execute();
    echo "New records created successfully";
    $stmt->close();
    $conn->close();

?>
```

Sélectionner des données

```
<?php
    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "myDB";

    $conn = new mysqli($servername, $username, $password,$dbname);
    if ($conn->connect_error) { die("Connection failed: " . $conn->connect_error);}

    $sql = "SELECT id, firstname, lastname FROM personne";
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        while($row = $result->fetch_assoc()) {
            echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
        }
    } else {
        echo "0 results";
    }
    $conn->close();
?>
```

- ❖ La fonction `fetch_assoc()` retourne un tableau associatif qui contient les lignes

Supprimer des données

```
<?php
    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "myDB";

    $conn = new mysqli($servername, $username, $password,$dbname);
    if ($conn->connect_error) { die("Connection failed: " . $conn->connect_error);}

    $sql = "DELETE FROM Personne WHERE id=3";

    if ($conn->query($sql)) {
        echo "Record deleted successfully";
    } else {
        echo "Error deleting record: " . $conn->error;
    }
    $conn->close();
?>
```

Modifier des données

```
<?php
    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "myDB";

    $conn = new mysqli($servername, $username, $password,$dbname);
    if ($conn->connect_error) { die("Connection failed: " . $conn->connect_error);}

    $sql = "UPDATE Personne SET lastname='Ism' WHERE id=2";

    if ($conn->query($sql)) {
        echo "Record updated successfully";
    } else {
        echo "Error updating record: " . $conn->error;
    }
    $conn->close();
?>
```