Partie 1 : Notion de la programmation en PHP

Table des matières

Partie 1 : Notion de la programmation en PHP

I

1. I- Introduction

1- Web

- On appelle **«Web»** (nom anglais signifiant **«toile»**), contraction de **«World Wide Web»** (d'où l'acronyme **www**), une des possibilités offertes par le réseau Internet de naviguer entre des documents reliés par des liens hypertextes.
- Le concept du Web a été mis au point au CERN (Centre Européen de Recherche Nucléaire) en 1991 par une équipe de chercheurs à laquelle appartenaient Tim-Berners LEE, le créateur du concept d'hyperlien, considéré aujourd'hui comme le père fondateur du Web.
- Le principe de web repose sur l'utilisation d'hyperliens pour naviguer entre des documents (appelés **«pages web»**) grâce à un logiciel appelé **navigateur**.
- Une page web est ainsi un simple fichier texte écrit dans un langage de description (appelé HTML), permettant de décrire la mise en page du document et d'inclure des éléments graphiques ou bien des liens vers d'autres documents à l'aide de balises.
- Au-delà des liens reliant des documents formatés, le web prend tout son sens avec le protocole HTTP permettant de lier des documents hébergés par des ordinateurs distants (appelés serveurs web, par opposition au client que représente le navigateur). Sur Internet les documents sont ainsi repérés par une adresse unique, appelée URL, permettant de localiser une ressource sur n'importe quel serveur du réseau internet.

2- PHP

- Il existe deux type de sites Internet : les sites **web statiques** et les sites **web dynamiques**.
- Les sites web dynamiques permettent de présenter les informations de différentes manières selon une certaine interaction avec le visiteur et cela grâce à un langage de programmation (PHP, Java, ASP ...)
- PHP signifiait à l'origine Personal Home Page
- PHP signifie aujourd'hui Php Hypertext Preprocessor car il renvoie à un navigateur un document HTML construit par le moteur de script Zend Engine 2 de PHP
- PHP est un langage de programmation utilisé dans la conception des applications web.

2. II- Notion de programmation en PHP

1- Variables, constantes et types

- **Les variables**

Une variable est le conteneur d'une valeur d'un des types utilisés par PHP (**entiers, flottants, chaînes de caractères, tableaux, booléens, ...**).

Chaque variable possède un identifiant particulier, qui commence toujours par le caractère dollar (**$**) suivi du nom de la variable. Les règles de création des noms de variable

sont les suivantes.

- Le nom commence par un caractère alphabétique, pris dans les ensembles [a-z], [A-Z] ou par le caractère de soulignement (_).
- La longueur du nom n'est pas limitée, mais il convient d'être raisonnable sous peine de confusion dans la saisie du code. Il est conseillé de créer des noms de variable le

plus **« parlant »** possible. En relisant le code contenant la variable $nomclient, par exemple, vous comprenez davantage ce que vous manipulez que si vous aviez écrit $x

ou $y.

- Les noms des variables sont sensibles à la casse (majuscules et minuscules). $mavar et $MaVar ne désignent donc pas la même variable.
- Affectation par valeur et par référence

L'affectation par valeur se fait à l'aide de l'opérateur **=**,

Exemple : $mavar = expression;

- Affectation par référence

L'affectation par référence est réalisée au moyen de l'opérateur **=**, l'opérande de droite est une variable qui doit être précédée du caractère **&**.

Exemple suivant :

$mavar1= ‘'ABIDJAN'';

$mavar2=‘'YAMOUSSOUKRO'';

$mavar2 = &$mavar1;

$mavar1=‘'NOE'';

- **Les constantes**

Vous serez parfois amené à utiliser de manière répétitive des informations devant rester constantes dans toutes les pages d'un même site.

PHP dispose d'un ensemble de constantes prédéfinies utilisables dans tous les scripts.

Pour définir des constantes personnalisées, utilisez la fonction **define()**, dont la syntaxe est la suivante :

define(nom_const, valeur_const, bool_casse)

Exemple :

```
<?php
//définition insensible à la casse
define("TVA",18,TRUE);
echo "Le taux de la TVA vaut ",TVA," %<br />";
echo " Le taux de la TVA vaut ",tva, "%<br />";
?>
```

- **Les types de données**

Dans PHP, il n'existe pas de déclaration explicite du type d'une variable lors de sa création. PHP reste un langage pauvrement typé comparé à Java ou au C.

PHP permet la manipulation d'un certain nombre de types de données différents dans lequel on distingue :

- Les types scalaires de base :

-Entiers, avec le type integer, qui permet de représenter les nombres entiers dans les bases 10, 8 et 16.

- Flottants, avec le type double ou float, au choix, qui représentent les nombres réels, ou plutôt décimaux au sens mathématique.

– Chaînes de caractères, avec le type string.

– Booléens, avec le type boolean, qui contient les valeurs de vérité TRUE ou FALSE (soit les valeurs 1 ou 0 si on veut les afficher).

- Les types composés :

– Tableaux, avec le type array, qui peut contenir plusieurs valeurs.

– Objets, avec le type object.

• Les types spéciaux :

– Type resource.

– Type null.

2- Les structures conditionnelles

Elles permettent d'exécuter un bloc d'instructions sous certaines conditions. Elles permettent de dire au système de prendre une décision en fonction d'une condition. En PHP on distingue :

- **La structure conditionnelle simple**

Elle exécute un bloc d'instructions lorsque la condition posée est vérifiée.

Syntaxe :

```
if(condition)
{
<Bloc d'instructions>
}
```

- **La structure conditionnelle alternative**

Elle exécute un bloc d'instructions lorsque la condition posée est vérifiée sinon, elle exécute un autre bloc d'instructions.

Syntaxe :

```
if(condition)
{
<Bloc d'instructions>
}
else
{
<Bloc d'instructions>
}
```

- **Les structures imbriquées**

Elles permettent de définir plusieurs structures simples à l'intérieur d'une structure alternative .

Syntaxe :

```
if(condition)
{
<Bloc d'instructions>
}
else if(condition2)
{
<Bloc d'instructions>
}
. . .
else
{
<Bloc d'instructions>
}
```

- **La structure de choix**

Elle est généralement utilisée lorsque vous avez plusieurs conditions à vérifier. Elle est une structure simplifiée des structures imbriquées mais ne s'utilise pas toujours comme telles.

Syntaxe :

```
switch (variable)
{
case <valeur1>:
<bloc d'instructions1>
```

break;

case <valeur2>:

<bloc d'instructions2>

break;

.

.

.

default:

<bloc d'instructionsn>

break;

}

- **Application 1**

Écrivez une expression conditionnelle utilisant une variable $moy et affiche un grade à un intervalle de moyene : grade A pour une note entre [18–20] B [14–18[, C [10–14[, et D [0–10[.

- **Application 2**

Rédigez une expression conditionnelle qui teste si un nombre est à la fois multiple de 3 et de 7.

- **Application 3**

Utilisez les variables $n1 ,$n2 et $op pour réaliser un script effectuant une opération parmi les quatre opérations arithmétiques élémentaires suivant la valeur de la variable $opération (utiliser l'instruction switch).

```php
//Correction application 1
<?php
        $moy=rand(0,20);
        if($moy>=18 && $moy<=20)
        {
          echo "Vous avez le grade A <br>";
         }
        else if($moy>=14 && $moy<18)
        {
          echo "Vous avez le grade B <br>";
         }
        else if($moy>=10 && $moy<14)
         {
          echo "Vous avez le grade C <br>";
         }
        else if($moy>=0 && $moy<10)
         {
          echo "Vous avez le grade D <br>";
         }
        else
         {
         echo "$note n'est pas une note valide.";
         }
?>
```

```php
////Correction application 2
```

```php
<?php
        $n= rand(1,100);
        if($nb%3==0 && $nb%7==0){
          echo "$n est un multiple de 3 et de 7 <br>";
        }
        else {
          echo "$n n'est pas un multiple commun de 3 et 7 <br>";
        }

?>
```

```php
////Correction application 3
<?php
        $n1=rand(1,100);
        $n2=rand(1,100);;
        $op='+';
        switch ($op) {
          case '+':
             $r=$n1+$n2;
            echo "$n1+$n2= ".$r."<br>";
            break;
          case '-':
             $r=$n1-$n2;
            echo " $n1-$n2 =". $r."<br>";
            break;
         case '*':
             $r=$n1*$n2;
             echo " $n1*$n2 =". $r."<br>";
             break;
         case '/':
             if($n2!=0){
                $r=$n1/$n2;
                echo "$n1/$n2 =". $r."<br>";
             }
             else echo "La division par zero est impossible";
             break;

          default:
            echo 'Merci de choisir une operation (+,-,*,/)';
            break;
        }
?>
```

3. III- Les boucles et les tableaux

1- Les boucles

Les boucles permettent de répéter des opérations élémentaires un grand nombre de fois sans avoir à réécrire le même code. Selon l'instruction de boucle utilisée, le nombre d'itérations peut être défini à l'avance ou être déterminé par une condition particulière.

Ainsi, il existe quatre types de boucle en PHP que sont :

- **La boucle For**

La structure for est une structure itérative qui lors de son utilisation, initialise la variable, définit la condition d'arrêt et le compteur.

Syntaxe :

```
for(initialisation; condition; compteur)
{
Bloc d'instructions;
}
```

Exemple :

```
<?php
for($i=1;$i<7;$i++)
{
echo "<h$i> $i :Titre de niveau $i </h$i>";
}
?>
```

- **La boucle while**

La boucle while permet de réaliser une action de manière répétitive tant qu'une condition est vérifiée ou qu'une expression quelconque est évaluée à TRUE et donc de l'arrêter quand elle n'est plus vérifiée (évaluée à FALSE).

Syntaxe :

```
while(condition)
{
Bloc d'instructions ;
}
```

Exemple :

```
<?php
//nombre initial à trouvé
$initial =495;
//compteur
$i=1;
//premier tirage la fonction mt_rand(min,max) permet de générer un nombre entre min et max
//vous pouvez utiliser rand() aussi mais la fonction mt_rand() est plus rapide .
$r=mt_rand(100,1000);
//boucle de tirage
while ($r!=$initial) {
$r=mt_rand(100,1000);
$i++;
}
```

```
echo "$initial trouvé après $i itérations";
?>
```

- **La boucle do .. While**

Avec l'instruction do...while, la condition n'est évaluée qu'après une première exécution des instructions du bloc compris entre do et while.

Syntaxe :

```
do
{
Bloc d'instructions1 ;
}
while( condition );
```

Exemple :

```
<?php
//nombre initial à trouvé
$initial =495;
//premier tirage la fonction mt_rand(min,max) permet de générer un nombre entre min et max
//vous pouvez utiliser rand() aussi mais la fonction mt_rand() est plus rapide .
$r=mt_rand(100,1000);
//boucle de tirage
//compteur
$i=0;
do{
$r=mt_rand(100,1000);
$i++;
}while($r!=$initial);
echo "$initial trouvé après $i itérations";
?>
```

- **La boucle foreach**

Introduite à partir de la version 4.0 de PHP, l'instruction foreach permet de parcourir rapidement l'ensemble des éléments d'un tableau, ce que fait aussi une boucle for, mais foreach se révèle beaucoup plus efficace

La boucle foreach est particulièrement efficace pour lister les tableaux associatifs dont il n'est nécessaire de connaître ni le nombre d'éléments ni les clés.

Syntaxe :

```
foreach($tableau as $valeur)
{
```

```
//bloc utilisant la valeur de l'élément courant
} ;
```

Ou

```
foreach($tableau as $cle=>$valeur)
{
//bloc utilisant la valeur et la clé de l'élément courant
}
```

Exemple :

```
<?php
for($i=0;$i<=8;$i++)
{
$tab[$i] = pow(2,$i);
}
$val ="Une valeur";
echo"Les puissances de 2 sont :";
foreach($tab as $val)
{
echo $val." : ";
}
?>
//--------------------------------------
<?php
for($i=0;$i<=8;$i++)
{ $tab[$i] = pow(2,$i);
}
foreach($tab as $ind=>$val)
{
echo " 2 puissance $ind vaut $val <br />";
}
?>
```

2- Les tableaux

Les tableaux sont représentés par le type **array** et sont d'une utilisation courante dans les scripts.

La possibilité de stocker un grand nombre de valeurs sous un seul nom de variable offre des avantages appréciables, notamment une grande souplesse dans la manipulation des données.

Les nombreuses fonctions natives de PHP applicables aux tableaux permettent les opérations les plus diverses dans la gestion des tableaux.

La fonction array() permet de créer de manière rapide des tableaux indicés ou associatifs.

C'est elle qui sera le plus souvent utilisée pour la création

Nous pouvons créer trois types de tableaux à savoir :

- **Les tableaux à indices numériques**

Il s'agit d'une simple liste d'éléments. Pour les créer, on emploi le mot clé **array()**, et on sépare les élément par des virgules.

Dans la liste, chaque élément est repéré par un numéro unique. Ce numéro est appelé index et est attribué automatiquement suivant les ordres.

Syntaxe :

$tab = array(val1,val2,...)

Exemple :

```
<?php
$etudiant = array("Lionnel", "Sebastian", "Kevin", "Aude", "Rodrigue");
echo $etudiant[0].'<br>';
echo $etudiant[1].'<br>';
echo $etudiant[2].'<br>';
echo $etudiant[3].'<br>';
echo $etudiant[4].'<br>';
?>
```

- **Les tableaux associatifs**

Les tableaux associatifs fonctionnent sur le même principe que les tableaux à indices numériques, sauf qu'au lieu de numéroter les cases, on va les étiqueter en leur donnant à chacune un nom différent.

Dans ce cas on parle de 'clé'. Ce type de tableau est très pratique dans le cas où l'on s'intéresse à la signification des valeurs contenues.

Syntaxe :

$tableau= array(cle1=>valeur1 , cle2=>valeur2 , ...);

Exemple :

```
<?php
$personne = array( "prenom" => “Kassele Josiane",
"nom" => "Blon",
"telephone" => "05142546", );
//affichage de l'élément nom
echo $personne['nom'];
```

?>

- **Les tableaux multidimensionnel**

Un tableau multidimensionnel est un tableau contenant lui même un ou plusieurs autres tableaux en valeurs.

Syntaxe :

$tableau= array($tab1,$tab2,$tab3);

Exemple :

```
<?php
$pers1 = array('prenom' => 'Josiane', 'nom' => 'Blon', 'telephone' => '05421552') ;
$pers2 =array('prenom' => 'Alex', 'nom' => Bora', 'telephone' => '08541202') ;
$pers3 =array('prenom' => 'Adjoua', 'nom' => 'Koua', 'telephone' => '06321202') ;
$personnes = array( $pers1,$pers2,$pers3);
echo $personnes[1]['prenom'];
?>
```

- **Lecture à l'aide de la boucle foreach**

Elle se révèle particulièrement efficace pour les tableaux associatifs mais fonctionne également pour les tableaux indicés.

Contrairement à la boucle for, l'instruction foreach() ne nécessite pas de connaître par avance le nombre d'éléments du tableau à lire.

Sa syntaxe varie en fonction du type de tableau

Syntaxe :

Pour les tableaux indicés, vous écrivez le code suivant :

```
foreach($tab as $valeur) {
//bloc de code utilisant les valeurs de la variable $valeur;
}
```

Pour les tableaux associatifs, vous disposez d'une syntaxe plus perfectionnée.

```
foreach($tab as $cle=>$valeur)
{ //bloc de code utilisant les valeurs des variables $cle et $valeur;
}
```

Exemple :

```
<?php $tab=array('Abidjan","Bouake","Bassam");
echo "<H3>Lecture des valeurs des éléments </H3>";
foreach($tab as $ville)
{
echo "<b>$ville</b> <br>";
}
```

```
echo"<hr>"; echo "<h3>lecture des indices et des valeurs des éléments </h3>";
foreach($tab as $indice=>$ville)
{
echo "L'élément d'indice <b>$indice</b> a la valeur <b>$ville</b><br>";
}
?>
```

4. IV- Les formulaires

1- Introduction

Un formulaire permet l'interactivité entre un site et ses visiteurs. Ils constituent pour cette raison la base de la création de sites dynamiques

Tout échange entre visiteur et serveur passe par un formulaire, dans lequel l'utilisateur peut saisir textes ou mots de passe, opérer des choix grâce à des boutons radio, des cases à cocher ou des listes de sélection, voire envoyer ses propres fichiers depuis le poste client.

2- Modes de transmission d'un formulaire

<form action="" method="“> </form>

- **L'attribut "action" de la balise de formulaire**

L'attribut action sert à définir la page appelée par le formulaire. C'est cette page qui recevra les données du formulaire et qui sera chargée de les traiter.

- **Transmission par la méthode GET**

Elle permet d'envoyer les données d'un formulaire, dans ce cas, les données transiteront par l'URL. Ce qui les rend visibles et modifiables très simplement par l'internaute.

- **Transmission par la méthode POST**

Elle permet d'envoyer les données d'un formulaire,les variables sont transmises de manière cachée : c'est généralement la méthode préférée des développeurs.

3- Récupération de données

- **Récupération par la méthode GET**

Elle se fait grâce à la fonction **$_GET** qui récupère les données envoyées par la méthode GET.

Créer un page formulaire.php

```
<p> Veuillez taper votre nom et prénoms : </p>
<form action="resul.php" method=“get">
<p>
<input type="text" name=“nom" />
<input type="text" name="prenoms" />
```

<input type="submit" value="Valider" />

</form>

Créer un page resul.php

Bonjour !

Votre identité est : <?php echo $_GET['nom'].' '; echo $_GET['prenoms']; ?>

Si tu veux changer votre identité clique ici pour revenir à la page formulaire

- **Récupération par la méthode POST**

Elle se fait grâce à la fonction **$_POST** qui récupère les données envoyées par la méthode POST.

Créer un page formulaire.php

 Veuillez taper votre nom et prénoms :

<form action="resul.php" method="post">

<input type="text" name="nom" />

<input type="text" name="prenoms" />

<input type="submit" value="Valider" />

</form>

Créer un page resul.php

Bonjour !

Votre identité est : <?php echo $_POST['nom'].' '; echo $_POST['prenoms']; ?>

Si tu veux changer votre identité clique ici pour revenir à la page formulaire

3-Transmission et récupération d'une variable par un lien

- **Transmission de données par l'URL**

<a href="page.php?para1=val1&para2=val2 ... &paraN=valN">

Exemple :

Créer un page resul2.php

Bonjour !

Votre identité est : <?php echo $_GET['nom'].' '; echo $_GET['prenoms']; ?>

Si tu veux changer votre identité clique ici pour revenir à la page formulaire

5. V- Session & Cookies

1- Les Cookies

Un cookie est un petit fichier texte qui ne peut contenir qu'une quantité limitée de données.

Les cookies vont être stockés sur les ordinateurs de vos visiteurs. Ainsi, à tout moment, un utilisateur peut lui même supprimer les cookies de son ordinateur.

De plus, les cookies vont toujours avoir une durée de vie limitée. On pourra définir la date d'expiration d'un cookie.

Généralement, nous allons utiliser les cookies pour faciliter la vie des utilisateurs en pré-enregistrant des données les concernant comme un nom d'utilisateur par exemple.

Ainsi, dès qu'un utilisateur connu demande à accéder à une page de notre site, les cookies vont également automatiquement être envoyées dans la requête de l'utilisateur. Cela va nous permettre de l'identifier et de lui proposer une page personnalisée.

- **Créer un cookie en PHP**

Pour créer un cookie en PHP, nous allons utiliser la fonction **setcookie().**

Une particularité notable de cette fonction est qu'il va falloir l'appeler avant d'écrire tout code HTML pour qu'elle fonctionne puisque les cookies doivent être envoyés avant toute autre sortie. **Cette restriction provient du protocole HTTP et non pas de PHP**.

Syntaxe :

setcookie(name, value, expire, path, domain, secure, httponly)

- **name** : Le nom du cookie. Le nom d'un cookie est soumis aux mêmes règles que les noms des variables.
- **value** : La valeur du cookie. Comme cette valeur est stockée sur l'ordinateur d'un utilisateur, on évitera de stocker des informations sensibles.
- **expire** : La date d'expiration du cookie sous forme d'un timestamp UNIX (nombre de secondes écoulées depuis le 1er janvier 1970). Si aucune valeur n'est passée en argument, le cookie expirera à la fin de la session (lorsque le navigateur sera fermé).
- **path** : Le chemin sur le serveur sur lequel le cookie sera disponible. Si la valeur est '/', le cookie sera disponible sur l'ensemble du domaine. Si la valeur est '/cours/', le cookie ne sera disponible que dans le répertoire (le dossier) /cours/ (et dans tous les sous-répertoires qu'il contient).
- **domain** : Indique le domaine ou le sous domaine pour lequel le cookie est disponible.
- **secure** : Indique si le cookie doit uniquement être transmis à travers une connexion sécurisée HTTPS depuis le client. Si la valeur passée est true, le cookie ne sera envoyé que si la connexion est sécurisée.
- **httponly** : Indique si le cookie ne doit être accessible que par le protocole HTTP. Pour que le cookie ne soit accessible que par le protocole http, on indiquera la valeur true. Cela permet d'interdire l'accès au cookie aux langages de scripts comme le JavaScript par exemple, pour se protéger potentiellement d'une attaque de type XSS.

```php
<?php
    setcookie('nom1', 'UVCI');
    setcookie('nom2', 'AEJ', time()+3600*24, '/', '', true, true);
?>
<!DOCTYPE html>
```

```
<html>
    <head>
        <title>Cours Cookies & Session</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="fichier.css">
    </head>

    <body>
        <h1>Formation AEJ</h1>

        <p>Partenariat UVCI</p>
    </body>
</html>
```

- Nous utilisons ensuite la fonction time() sans lui passer d'argument pour récupérer la valeur du timestamp actuel. Nous allons nous servir de cette valeur en lui ajoutant un certain nombre de secondes pour définir la date d'expiration de notre deuxième cookie nom2. Dans le cas présent, on définit une durée de vie de 24h pour le cookie (3600 secondes * 24 à partir de sa date de création).
- Toujours pour notre deuxième cookie, nous utilisons la valeur par défaut pour le chemin du serveur sur lequel le serveur est accessible, c'est à dire la valeur / qui signifie que le cookie sera accessible sur l'ensemble d'un domaine ou d'un sous-domaine (c'est-à-dire dans tous ses répertoires).
- On ne précise pas de domaine de validité ici car nous travaillons en local. Si j'avais voulu rendre mon cookie disponible pour tout mon site, j'aurais précisé uvci.edu.ci.
- Finalement, nous précisons les valeurs true pour les arguments « secure » (passage par une connexion sécurisée pour transmettre le cookie) et « httponly » (obligation d'utiliser le protocole HTTP pour accéder au cookie).
- **Récupérer la valeur d'un cookie**

Pour récupérer la valeur d'un cookie, nous allons utiliser la variable superglobale **$_COOKIE**. Cette superglobale est un tableau associatif qui utilise les noms des cookies en clefs et associe leurs valeurs en valeurs du tableau.

Pour accéder à la valeur d'un cookie, il faut renseigner le nom du cookie en clef de ce tableau.

```
<?php
    setcookie('nom1', 'UVCI');
    setcookie('nom2', 'AEJ', time()+3600*24, '/', '', true, true);
?>
<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP & MySQL</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="fichier.css">
    </head>

    <body>
        <h1>Formation AEJ</h1>


        <?php
            if(isset($_COOKIE['nom1'])){
                echo 'Votre ID de session est le ' .$_COOKIE['nom1'];
            }
        ?>
          <p>Partenariat UVCI</p>
```

```
    </body>
</html>
```

2- Les Session

Une session en PHP correspond à une façon de stocker des données différentes pour chaque utilisateur en utilisant un identifiant de session unique.

Les identifiants de session vont généralement être envoyés au navigateur via des cookies de session et vont être utilisés pour récupérer les données existantes de la session.

Un des grands intérêts des sessions est qu'on va pouvoir conserver des informations pour un utilisateur lorsqu'il navigue d'une page à une autre. De plus, les informations de session ne vont cette fois-ci pas être stockées sur les ordinateurs de vos visiteurs à la différence des cookies mais plutôt côté serveur ce qui fait que les sessions vont pouvoir être beaucoup plus sûres que les cookies.

- **Démarrer une session en PHP**

Pour démarrer une session en PHP, on va utiliser la fonction **session_start()**. Cette fonction va se charger de vérifier si une session a déjà été démarrée en recherchant la présence d'un identifiant de session et, si ce n'est pas le cas, va démarrer une nouvelle session et générer un identifiant de session unique pour un utilisateur.

La fonction **session_start()** est appelée avant toute autre opération dans nos pages, c'est-à-dire au début de celles-ci.

- **Définir et récupérer des variables de session**

Pour définir et récupérer les valeurs des variables de session, nous allons pouvoir utiliser la variable superglobale **$_SESSION**.

Cette superglobale est un tableau associatif qui stocke les différentes variables de sessions avec leurs noms en index du tableau et leurs valeurs en valeurs du tableau.

Nous allons donc très simplement pouvoir à la fois définir de nouvelles variables de session et modifier ou récupérer les valeurs de nos variables de session.

```
// Pour définir
<?php
    //On démarre une nouvelle session
    session_start();

    //On définit des variables de session
    $_SESSION['nom'] = 'MIZAN';
    $_SESSION['poids'] = 5;
?>
<!DOCTYPE html>
<html>
    <head>
        <title>Cours Cookies & Session</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="fichier.css">
    </head>

    <body>
        <h1>Formation AEJ</h1>

        <p>Partenariat UVCI</p>
```

```
    </body>
</html>
```

```
// Pour recupérer
<?php
    //On démarre une nouvelle session
    session_start();

?>
<!DOCTYPE html>
<html>
    <head>
        <title>Cours Cookies & Session</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="fichier.css">
    </head>

    <body>
      <?php
            echo 'Bonjour ' .$_SESSION['nom']. ',
            tu as ' .$_SESSION['poids']. ' Kg';
        ?>
    </body>
</html>
```

- **Terminer une session et détruire les variables de session**

Une session PHP se termine généralement automatiquement lorsqu'un utilisateur ferme la fenêtre de son navigateur.

Il peut être cependant parfois souhaitable de terminer une session avant. Pour faire cela, nous allons pouvoir utiliser les fonctions **session_destroy()** qui détruit toutes les données associées à la session courante et **session_unset()** qui détruit toutes les variables d'une session.

La fonction **session_destroy()** va supprimer le fichier de session dans lequel sont stockées toutes les informations de session. Cependant, cette fonction ne détruit pas les variables globales associées à la session (c'est-à-dire le contenu du tableau $_SESSION) ni le cookie de session.