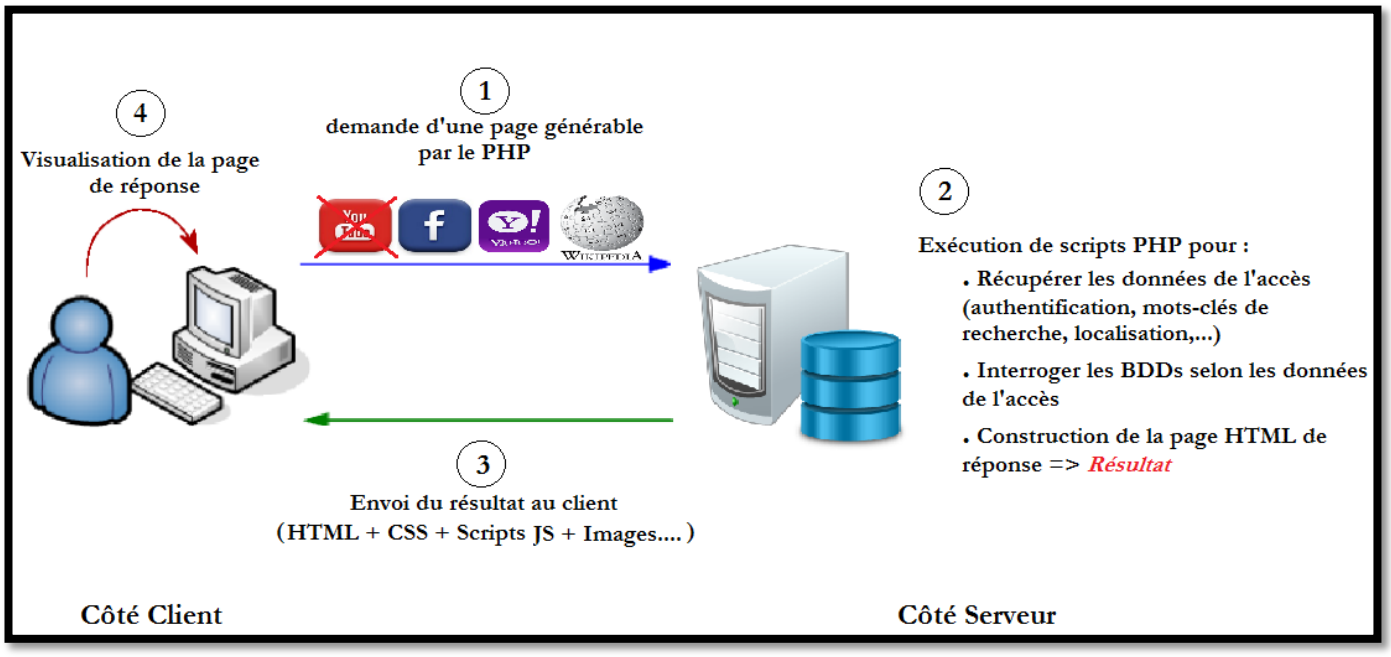
Cours n° 6 : *Introduction pratique au PHP*

A. Introduction

1. Le PHP, qu'est-ce que c'est ?

- Acronyme de **Hypertext Preprocessor**.
- Un langage de scripts **côté serveur**.
- Utilisé pour créer des ***pages Web dynamiques*** à travers un serveur http (contrairement au ***JavaScript*** qui permet de créer des ***pages Web interactives***).
- Permet de traiter les données des formulaires HTML.
- Permet de manipuler les BDDs enregistrées au sein du serveur.
- Les scripts PHP sont exécutés au sein du serveur Web (le ***WampServer*** par exemple) et le résultat de l'exécution (HTML, CSS, JS, images,….) est retourné au client.

2. Principe de fonctionnement



Le client demande une page Web dynamique **générable par le PHP** (qui est générée au sein du serveur par un ou plusieurs scripts PHP). Sa demande est transformée sous forme de requête http et transportée à travers le réseau Internet au serveur. Un ou plusieurs scripts PHP peuvent être interprétés (par l'**interpréteur PHP**) pour répondre à la requête du client. La première phase de la réponse consiste à récupérer toutes les données liées à la requête du client (mots-clés s'il s'agit d'une recherche, pseudo et mot de passe s'il s'agit d'une authentification, données d'un formulaire s'il s'agit d'une inscription, information de localisation,….). Une base de données est interrogée par la suite pour récupérer toutes les données qui correspondent à la demande du client. A la base des données récupérées, la réponse est

construite sous forme de document HTML (accompagné par des styles CSS, des scripts JS, des images,…) et envoyée au client.

Remarque : Sur n'importe quelle page Web dynamique (par exemple celle de Facebook), faites ***clique droit*** puis ***Voir le code source de la page***, vous allez trouver uniquement un mélange de HTML, du CSS, et du JavaScript. Ceci c'est parce que tous les scripts PHP sont exécutés au niveau du serveur, et ce que vous obtenez c'est juste le résultat de cette exécution.

3. Installations nécessaires

Pour exécuter des scripts PHP, il faut deux composantes principales :

- Un **interpréteur de scripts PHP** (comme *Zend Engine*),
- Un **serveur http** (comme *Apache*),

En plus, pour manipuler des BDDs, il faut encore :

- Un serveur de gestion de base de données - **SGBD** (comme *MySQL*),

La combinaison entre les trois composantes forme ce que l'on appelle une ***plateforme de développement Web***, abréviée par ***AMP*** (**A**ppache **M**ySQL **P**HP). Parmi ces plateformes de développement Web on trouve ***WampServer*** et ***EasyPHP*** pour Windows, et ***LAMP*** pour Linux.

Installez le WampServer en suivant les instructions ci-dessous, si ça ne marche pas pour une raison ou une autre, alors tentez à installer le EaysPHP comme expliqué en bas.

B. Syntaxe générale

1. Premiers scripts PHP

Un script PHP doit être enregistré avec l'extension « **.php** », il doit contenir un ou plusieurs **codes PHP**. Un code PHP doit être délimité par **<?php** et **?>**. Un script PHP peut contenir aussi des codes HTML, CSS et JavaScript.

Pourquoi ce mélange ? Pourquoi pas uniquement du PHP ? Une page Web dynamique en général n'est pas dynamique à 100%, elle doit contenir quelques parties statiques et d'autres dynamiques. Donc, le code HTML représente la partie statique de la page à générer, tandis que les codes PHP permettent de générer la partie dynamique de cette page.

Exemple 1 : **« premier_script.php »**

```
<?php
        /*
        * La fonction echo permet d'afficher un contenu (texte, HTML, CSS,…)
        */
        echo "Ceci est mon premier script PHP";
?>
```

Explication: Ce script contient un seul code PHP qui permet d'afficher un texte simple à travers la fonction prédéfinie **echo**.

Exemple 2 : **« deuxieme_script.php »**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Premiers pas avec le PHP</title>
    <meta charset="UTF-8">
  </head>
  <body>
      <?php
          echo "Il est " . date("H:i:s", strtotime('1 hour')) . " chez le serveur";
      ?>
  </body>
</html>
```

Explication: Ce script contient un code HTML qui représente la partie statique de la page à générer, et un seul code PHP permettant de générer l'heure actuelle du serveur à l'intérieur de l'élément body. La fonction prédéfinie **date("H:i:s", strtotime('1 hour'))** permet d'afficher l'heure actuelle sur le serveur sous format **GMT+1**. Le point **.** permet de faire la concaténation entre chaines de caractères.

Exemple 3 : **« Générer un code HTML+CSS dynamiquement »**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Premiers pas avec le PHP</title>
    <meta charset="UTF-8">
  </head>
  <body>
      <?php
          echo "<p style='color :red'>Il est " . date("H:i:s", strtotime('1 hour')) . " chez le serveur</p>";
      ?>
  </body>
</html>
```

Exemple 4 : **« Générer un code JavaScript dynamiquement »**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Premiers pas avec le PHP</title>
    <meta charset="UTF-8">
  </head>
  <body>
      <?php
          echo "<script>alert('Il est " . date("H:i:s", strtotime('1 hour')) . " chez le serveur') ;</script>";
      ?>
  </body>
</html>
```

Explication: La fonction prédéfinie **echo** permet non seulement de générer un texte simple (comme dans les exemples 1 et 2), mais aussi des codes HTML+CSS, et des scripts JS.

2. Les variables :

Les variables doivent être précédées par **$** et peuvent avoir plusieurs types de données : ***Integer***, ***Float***, ***Chaine de caractères***, ***Booléen***, ***Array***, ***Ressource*** (utilisée avec les bases de données).

Exemple 5 : **« variables.PHP »**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Variables et types de données en PHP</title>
    <meta charset="UTF-8">
  </head>
  <body>
      <?php
          //Déclaration de variables
          $message = "Bonjour tout le monde";          //String
          $x = 9;                                      //Integer
          $y = 12.23;                                  //Float
          $reponse = true;                             //Boolean

          //Affichage de variables
          echo   $message . "<br>" ;
          echo   $x . "<br>" ;
          echo   ($x + $y) . "<br>" ;

          echo   "<h3 style='color:red'>" . $x  .  " + " .  $y  . " = "  .  ($x + $y)  .  "</h3><br>";
      ?>
  </body>
</html>
```

3. Les chaines de caractères :

Les chaines de caractères doivent être délimitées par des "" ou ' '. Le point (.) est utilisé pour faire la concaténation entre deux chaines de caractères. Plusieurs fonctions prédéfinies existent et qui permettent de manipuler les chaines de caractères comme celles utilisées dans l'exemple ci-après : ***strlen()***, ***str_word_count()***, ***str_replace()***.

Exemple 6 : **« chaines_de_caracteres.PHP »**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les chaînes de caractères en PHP</title>
    <meta charset="ISO-8859-15">
  </head>
  <body>
      <?php
          $message_1 = "Bonjour ";
          $message_2 = "tout le monde";

          //La concaténation est faite par l'opérateur "."
          $message = $message_1 . $message_2;

          echo $message . "<br/>";

          //strlen() retourne la taille d'une chaîne
          echo "Taille de la chaîne est : "  .  strlen($message)  .  "<br/>";

          //str_word_count() retourne le nombre de répétitions d'un mot dans une chaîne
          echo "Nombre de mots : " . str_word_count($message) . "<br/>";

          //str_replace() remplace toutes les occurrences d'un mot par un autre mot
          echo str_replace('tout le monde', 'à tous ', $message) .  "<br/>";
      ?>
  </body>
</html>
```

4. Les constantes :

Les constantes sont déclarées avec la fonction prédéfinie **define(nom_constante, valeur_constante)**. N'oubliez pas de précéder le nom de la constante par **$** au niveau de l'appel.

***Exemple 7 :* « constantes.PHP »**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les constantes en PHP</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <?php
        define('message', 'Bonjour tout le monde');     //Constante de type String
        define('x', 9);                                 //Constante de type Integer
        define('y', 12.23);                             //Constante de type Float
        define('reponse', true);                        //Constante de type Boolean
        echo message."<br/>";
        echo x + y;
    ?>
  </body>
</html>
```

5. Les fonctions :

Comme en JavaScript, les fonctions en PHP doivent être déclarées avec le mot-clé **function** et les valeurs de retour sont retournées avec le mot-clé **return**. La fonction n'est exécutée qu'après l'appel. Le script suivant contient plusieurs fonctions avec ou sans paramètres, et avec ou sans valeurs de retour. Les paramètres peuvent avoir *des valeurs par défaut* comme le cas de la fonction **addition($v1=21, $v2=34)**, cette fonction possède deux paramètres **$v1** et **$v2** et chaque paramètre a une valeur par défaut. Au niveau de l'appel de la fonction si un paramètre n'est pas mentionné alors c'est sa valeur par défaut qui est utilisée. L'appel **addition(4, 5)** retourne **9**, **addition(4)** fait l'addition de **4** avec la valeur par défaut de **$v2** donc le résultat sera **38**, tandis que **addition()** fait l'addition de la valeur par défaut de **$v1** avec la valeur par défaut de **$v2**, donc le résultat sera **55**.

***Exemple 8 :* « fonctions.php »**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les fonctions en PHP</title>
    <meta charset = "ISO-8859-15">
  </head>
  <body>
      <?php
          //Fonction sans paramètres ni valeur de retour
          function bonjour_a_tous(){
              echo "Bonjour tout le monde <br>";
          }
          //Fonction avec paramètres mais sans valeur de retour
          function special_bonjour($nom){
              echo "Bonjour " . $nom . "<br>";
          }
          //Fonction avec paramètres et valeur de retour
```

```
                function addition($v1 = 21, $v2 = 34){
                        return $v1 + $v2;
                }

                //Appels des fonctions
                bonjour_a_tous();
                special_bonjour("Mohamed");
                echo addition(4, 5) . "<br/>";
                echo addition(4) . "<br/>";
                echo addition() . "<br/>";
        ?>
  </body>
</html>
```

6. La portée de variables :

Les variables globales en PHP ne peuvent pas être utilisées à l'intérieur de fonctions par un simple appel comme en JavaScript. L'appel à la ligne 11 du script suivant retourne une erreur vu que la variable ***message*** n'est pas reconnue à l'intérieur de la fonction. Pour résoudre ce problème, chaque variable globale en PHP doit être appelée à travers le tableau prédéfini de variables globales **$GLOBALS**, comme c'est fait à la ligne 12. Les variables statiques doivent être précédées par le mot-clé **static**.

***Exemple 9 :* « portee_variables.php »**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Portée de variables en PHP</title>
    <meta charset = "ISO-8859-15">
  </head>
  <body>
    <?php
        $message = "Bonjour tout le monde <br><br>";          //Une variable globale
        function afficher(){
            //echo $message ;                                 //Cet appel génère une erreur !!!
            echo $GLOBALS['message'];                         //Affichage de la variable globale
        }
        afficher();
    ?>
    <?php
        function variable_statique(){
            static $nbre_appels = 1;
            echo "Cette fonction a été appelée " . $nbre_appels . " fois <br>";
            $nbre_appels++;
        }

        variable_statique();
        variable_statique();
        variable_statique();
    ?>
  </body>
</html>
```

7. Les tableaux indexés :

Les tableaux en PHP sont déclarés par la fonction prédéfinie **array(valeur1, valeur2,….)**. La première valeur prend l'indice **0**, la deuxième l'indice **1**, et ainsi de suite. Lisez attentivement le script suivant pour comprendre comment déclarer un tableau, comment accéder à ses valeurs, comment les modifier, et comment ajouter de nouvelles valeurs. La fonction prédéfinie **count()** permet de retourner la taille d'un tableau.

***Exemple 10 :* « tableaux_indexes.php »**

```
<!DOCTYPE html>
<html>
   <head>
      <title>Les tableaux indexés en PHP</title>
      <meta charset="ISO-8859-15">
   </head>
   <body>
       <?php
          //Création d'un tableau indexé
          $tab = array(10, "message", 10.34);
          echo "Valeur 1: " . $tab[0] . ", Valeur 2: " . $tab[1] . ", Valeur 3: " . $tab[2] . "<br><br>";


          //Modifier des valeurs
          $tab[0] = 20;

          //Ajouter une nouvelle valeur
          $tab[3] = "autre valeur";

          //Parcourir un tableau
          echo "La fonction count() retourne la taille d'un tableau <br><br>";
          for ($i = 0; $i < count($tab); $i++){
                    if($i == 0){
                              echo "1 <sup>ère</sup> valeur = " . $tab[$i] . "<br>";
                    } else {
                              echo ($i + 1) . "<sup>ème</sup> valeur = " . $tab[$i] . "<br/>";
                    }
          }
      ?>
   </body>
</html>>
```

8. Les tableaux multidimensionnels (matrices) :

***Exemple 11 :* « matrices.php »**

```
<!DOCTYPE html>
<html>
   <head>
      <title>Les tableaux multidimensionnels en PHP</title>
      <meta charset="ISO-8859-15">
   </head>
   <body>


      <?php
         //Création d'un tableau multidimensionnel (Matrice)
```

```
        $matrice = array(
            array(10, 20, 30),
            array (40, 50, 60),
            array (70, 80, 90),
            array (100, 110, 120)
         );
        //Parcourir une matrice
        for ($i=0;$i<4; $i++){
            for ($j=0;$j<3; $j++){
              echo $matrice[$i][$j]   .  " ";
           }
           echo "<br/>";
        }
    ?>
  </body>
</html>
```

9. Les structures de contrôle :

Voici la syntaxe de chacune des structures ***if***, ***if else***, ***if elseif*** :

```
if (condition){
      //actions;
}
```

```
if (condition){
      //actions;
} else {
      //autres actions;
}
```

```
if (condition_1){
      //action 1;
} elseif (condition_2){
      //action 2;
} else {
     //autres actions
}
```

Exemple 12 : **« conditions.php »**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Branchement conditionnel en PHP</title>
    <meta charset = "ISO-8859-15">
  </head>
  <body>
     <?php
          function periode_de_vie($age){
              if($age < 0){
                        echo "Pas encore né" ;
              } elseif($age > 0 && $age <= 2){
                        echo "Bébé" ;
              } elseif($age > 2 && $age <= 12){
                        echo "Enfant" ;
              } elseif($age > 12 && $age <= 18){
                        echo "Adolescent" ;
              } elseif($age > 18 && $age <= 70){
                        echo "Adulte" ;
              } else {
                        echo "Personne âgée" ;
              }
          }

        periode_de_vie(13);
    ?>
```

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

Le script suivant montre la syntaxe de la structure ***switch*** :

```
<?php
    $couleur_preferee = "rouge";
    switch ($couleur_ preferee) {
        case "rouge":
            echo "Votre couleur préférée est le Rouge";
            break;
        case "blue":
            echo " Votre couleur préférée est le Bleu";
            break;
        default:
            echo "Votre couleur préférée n'est ni le rouge ni le bleu";
    }
?>
```

Voici la syntaxe de chacune des structures ***while***, ***do while***, ***for***:

```
while (condition){
      //actions;
}
```

```
do {
      //actions;
} while (condition);
```

```
for ($i = 0; $i < 10; $i++){
      //actions;
}
```

C. Traitement des formulaires HTML avec PHP

Lors de la création d'un formulaire HTML par la balise **<form>**, on précise comment les données du formulaire *sont envoyées* au serveur et par quel script *elles seront traitées*. Le traitement est fait par un script côté serveur (le PHP dans notre cas) et l'envoi est fait soit par la méthode **GET** soit par la méthode **POST**. Ces deux informations sont à mentionner par les attributs ***method*** et ***action*** de la balise **<form>**. Par exemple, le formulaire **<form method="get" action="traitement.php">…</form>** précise que les données sont envoyées au serveur à travers la méthode **GET** et elles seront traitées par le script **traitement.php**.

La méthode **GET** envoie les données du formulaire dans l'URL donc elle dépend de la taille maximale de l'URL, et les données sont envoyées en clair (**Attention :** s'il s'agit d'un mot de passe alors sa valeur sera envoyée en clair dans l'URL !!!). La méthode **POST** qui est la plus utilisée envoie les données, *d'une façon transparente*, à travers la méthode **HTTP POST**.

1. Utilisation de la méthode GET:

La page HTML suivante (**formulaire.HTML**) contient un formulaire avec deux champs **nom** et **prénom**. Les données de ces deux champs sont envoyées au serveur par la méthode **GET** et elles seront traitées par le script **traitement_get.php**.

Exemple 13 : « formulaire_1.HTML »

```
<!DOCTYPE html>
<html>
<head>
   <title>Traitement de formulaires avec PHP</title>
   <meta charset="ISO-8859-15">
</head>
<body>
      <div align="center">
         <form method="get" action="traitement_get.php">
            <table>
               <tr>
                  <td><label>Nom</label></td><td><input type="text" name="nom"></td>
               </tr>
               <tr>
                  <td><label>Prénom</label></td><td><input type="text" name="prenom"></td>
               </tr>
               <tr>
                  <td colspan="2" align="center"><input type="submit" value="Valider les données"></td>
               </tr>
            </table>
         </form>
      </div>
   </body>
</html>
```

Le PHP possède un tableau prédéfini **$_GET** qui permet de conserver toutes les données du formulaire qui sont envoyées par la méthode **GET**. La récupération des valeurs du formulaire est faite à travers le nom de chaque composant du formulaire. Le formulaire de la page HTML précédente contient deux composants identifiés respectivement par **name="nom"** et **name="prenom"**. Donc, au niveau du serveur la récupération du nom et du prénom saisis est faite par **$_GET["nom"]** et **$_GET["prenom"]**. Le script PHP suivant permet de récupérer les données de la page **formulaire.HTML** et les afficher.

Exemple 14 : « traitement_get.PHP »

```
<?php
      $nom = $_GET["nom"] ;
      $prenom = $_GET["prenom"] ;

      echo "Les donnees de votre formualire sont:<br>";
      echo "Nom: " . $nom . ", Prenom: " . $prenom ;
?>
```

2. Utilisation de la méthode POST:

Dans la page **formulaire_1.HTML** remplacez **method="get"** par **method="post"** et **action="traitement_get.php"** par **action="traitement_post.php"**. Dans ce cas les données du formulaire vont être envoyées par la méthode **POST**. Au niveau du serveur, ces données sont récupérées à travers le tableau prédéfini **$_POST** qui permet de conserver toutes les données du formulaire qui sont envoyées par la méthode **POST**. Donc, **$_POST["nom"]** et **$_POST["prenom"]** retournent le nom et le prénom saisis dans le formulaire. Le script PHP suivant permet de récupérer les données de la page modifiée **formulaire.HTML** et les afficher.

Exemple 15 : « traitement_post.PHP »

```php
<?php
	$nom = $_POST["nom"] ;
	$prenom = $_POST["prenom"] ;

	echo "Les donnees de votre formulaire sont:<br>";
	echo "Nom: " . $nom . ", Prenom: " . $prenom ;
?>
```

3. Traitement des différents champs d'un formulaire HTML :

Supposant dans ce qui suit que la méthode utilisée est **POST**. Nous allons voir comment récupérer la valeur de chaque champ dans un formulaire HTML.

- **<input type="text" name="nom"/>: $_POST['nom']** retourne la valeur saisie dans ce champ. C'est le même principe pour tous les champs **input** (**type** égal à **"password"**, **"number"**, **"date"**,…).

- **Boutons radios à sélection unique :** Supposons que l'on a trois boutons radios comme suit :

```php
<input type="radio" name="diplome" value="L" checked/>
<input type="radio" name="diplome" value="M" />
<input type="radio" name="diplome" value="D" />
```

Alors, **$_POST['diplome']** retourne la valeur du bouton sélectionné (donc la valeur **"L"**, **"M"** ou **"D"**). Le traitement peut être effectué comme suit :

```php
< ?php
	$diplome = $_POST['diplome'] ;
	if($diplome == "L"){
		echo "vous êtes en Licence";
	} elseif($diplome == "M") {
		echo "vous êtes en Master";
	} else {
		echo "vous êtes en Doctorat";
	}
?>
```

- **Eléments de type select :** Le même principe des boutons **radio** est appliqué pour les éléments de type **select**. C'est-à-dire, **$_POST['nom_du_select']** retourne la valeur de l'option choisie. Supposons que l'on a un élément **select** avec deux options comme suit :

```php
<select name="ville">
	<option value="orn">Oran</option>
	<option value="tlm" selected>Tlemcen</option>
</select>
```

Le traitement en PHP est fait comme suit :

```php
< ?php
	$ville = $_POST['ville'] ;
	if($ville == "orn"){
		echo "vous habitez à Oran";
	} else {
		echo "vous habitez à Tlemcen";
	}
?>
```

- **Cases à cocher (checkbox) :** Supposons que l'on a trois choix présentant des activités sportives :

```
<input type="checkbox" name="foot" checked/>
<input type="checkbox" name="cyclisme" />
<input type="checkbox" name="natation" checked/>
```

Lors de la validation du formulaire, uniquement les valeurs des cases cochées sont envoyées au serveur. Donc vu que l'on a deux cases cochées, le serveur reçoit uniquement deux valeurs **$_POST['foot']** et **$_POST['natation']**, l'appel **$_POST['cyclisme']** retourne une erreur. Pour détecter si une case est cochée ou pas on peut utiliser la fonction prédéfinie **isset()** qui permet de tester si la valeur d'un champ existe et si elle est différente de NULL. Avec cette fonction, le traitement de ces trois cases peut être fait comme suit :

```
< ?php
   if(isset($_POST['foot'])){  echo "vous faites du Football <br>";  }
   if(isset($_POST['cyclisme'])) {  echo "vous faites du cyclisme <br>";  }
   if(isset($_POST['natation'])) {  echo "vous êtes de la natation <br>";  }
?>
```