**Technologies du WEB**

**Atelier PHP**

**OBJECTIF :**

Le but de cet atelier est de créer un CRUD avec PHP.

Pour ce faire, nous présentons ici **QUELQUES** définitions et fonctionnalités de PHP.

**PHP : C’est QUOI ?**

* + 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 (PHP est distribué sous licence GNU GPL)
    - 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, Microsoft IIS, ...)

**Intégration PHP et HTML**

* **Principe**

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 php: <?php ligne de code PHP ?>
    - avec le style JavaScript :

<script language=«php»> ligne de code PHP </script>

Ils sont aussi intégrés via l’inclusion d’un fichier PHP.

Exemple :

Fichier Prinipal

<html>

<head>

<title> Fichier d’appel </title>

</head>

<body>

<?php

$salut = “ BONJOUR” ;

include “information.inc” ;

?>

</body>

</html>

Fichier à inclure : information.inc

<?php

$chaine=$salut. “ , C’est PHP “  ;

echo “  <table border= \“3“

<tr> <td width = “ 100%\ “ >

<h2> $chaine</h2>

</td> </tr></table> “;

?>

**Les variables**

**Principe**

* + - Commencent par le caractère $
    - N'ont pas besoin d'être déclarées
    - Une des fonctions de variables les plus utilisées est la fonction **isset**(var) qui permet de déterminer si une variable est bien définie

**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

**Exemple**

<?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 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()** permet d’avoir le nombre d’éléments d’un tableau

**Les classes et les objets**

Une classe est composée de deux parties:

* + - Les attributs: il s'agit des données représentant l'état de l'objet
    - Les méthodes : il s'agit des opérations applicables aux objets

**Exemple :**

<?php

//Définition de la classe client

class client {

//Les attributs de la classe

// Cet exemple oublie le concept d’encapsulation mais les attributs sont normalement privés

Public $nom;

Public $ville;

Public $naiss ;

//Les méthodes

Public function age() {

// la fct getdate() retourne un tableau contenant la date et l’heure courante.

$jour = getdate();

$an=$jour["year"];

$age = $an - $this->naiss;

echo "Il a $age ans cette année <br />" ;

}

}

//création d’un objet

$client1 = new client() ;

//affectation des propriétés de l’objet

$client1 -> nom = "Dupont" ; $client1-> naiss = "1961" ; $client1->ville = "Angers" ;

//utilisation des propriétés

echo "le nom du client1 est ", $client1->nom, "<br />" ;

echo "la ville du client1 est ", $client1-> ville, "<br />" ;

echo "le client1 est né en ", $client1->naiss, "<br />" ;

//appel de la méthode age()

$client1->age() ;

?>

**Interaction entre PHP et les formulaires :**

Les formulaires transmettent les informations qu’ils contiennent de deux manières différentes :

**La méthode GET** place les informations d'un formulaire directement à la suite de l'adresse URL de la page appelée.

**Exemple**

[traitementFormulaire.php?champ1=valeur1&champ2=valeur](http://www.boutell.com/gd)2

**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 ce qui assure une confidentialité efficace des données.

**Comment récupérer les données transmises par le formulaire ?**

Les tableaux associatifs **$\_GET** et **$\_POST** contiennent toutes les variables envoyées par un formulaire.

**Exemple :**

La page html contenant le formulaire :

<form method="GET" action="page\_cible.php">

<input type="text" name="Champ\_saisie" value="Texte"/>

<select name="Liste\_Choix" size="3">

<option value="Option\_1">Option\_1</option>

<option value="Option\_2">Option\_2</option>

<option value="Option\_3">Option\_3</option>

</select>

<textarea name="Zone\_Texte" cols="30" rows="5"> Texte par défaut </textarea>

<input type="checkbox" name="Case\_Cocher[]" value="Case\_1"> Case 1<br>

<input type="checkbox" name="Case\_Cocher[]" value="Case\_2"> Case 2<br>

<input type="checkbox" name="Case\_Cocher[]" value="Case\_3"> Case 3<br>

<input type="radio" name="Case\_Radio" value="Case radio 1"> radio 1<br>

<input type="radio" name="Case\_Radio" value="Case radio 2"> radio 2<br>

<input type="radio" name="Case\_Radio" value="Case radio 3"> radio 3<br>

<input type="reset" name="Annulation" value="Annuler">

<input type="submit" name="Soumission" value="Soumettre">

</form>

La page php qui reçoit les données et les affiche :

<?php

echo " test php <br>";

// Le ‘**.**’ permet de concaténer deux chaines de caractères.

$resultat=$\_GET["Champ\_saisie"]."<br>";

$resultat .= $\_GET["Liste\_Choix"]."<br>";

$resultat .= $\_GET["Zone\_Texte"]."<br>";

for ($i = 0; $i < count($\_GET["Case\_Cocher"]); $i++)

{$resultat .= $\_GET["Case\_Cocher"][$i]."<br>" ;}

$resultat .= $\_GET["Case\_Radio"]."<br>";

echo $resultat; ?>

La plupart des éléments d'un formulaire n'acceptent qu'une seule et unique valeur, laquelle est affectée à la variable correspondante dans le script de traitement.

$Champ\_Saisie ← "Ceci est une chaîne de caractères.";

Pour des cases à cocher et les listes à choix multiples, plusieurs valeurs peuvent être sélectionnées entraînant l'affectation d'un tableau de valeurs aux variables correspondantes.

$Case\_Cocher[0] ← "Case radio 1";

$Case\_Cocher[1] ← "Case radio 3";

**Interfaçage avec une base de données**

PHP propose de nombreux outils permettant de travailler avec la plupart des SGBDR

* + - Oracle, Sybase, Microsoft SQL Server, PostgreSQL ou encore MySQL

PHP fournit un grand choix de fonctions permettant de manipuler les bases de données.

* + - Quatre fonctions sont essentielles:
      * La fonction de connexion au serveur
      * La fonction de choix de la base de données
      * La fonction de requête
      * La fonction de déconnexion

Afin d’interfacer avec une base de données, PHP offre la classe **PDO** que nous allons utiliser.

**Connexion à la base de données :**

Pour se connecter à une base de données on instancie un objet PDO de la façon suivante :

$maDb\_connexion = new PDO('mysql:host=localhost;dbName=nomDeLaBase', 'userName', 'motDePasse');

On crée donc une nouvelle instance de PDO qu’on récupère dans la variable $maDb\_connexion. Le constructeur nécessite trois paramètres :

* le driver utilisé, l'adresse du serveur et le nom de la base noté ainsi driver:hot=serveur; dbName=nomDeLaBase'
* le nom d'utilisateur à utiliser pour la connexion au serveur
* le mot de passe du dit utilisateur

Ces informations différent un peu selon le pilote.

**Les requêtes SQL**

Les requêtes doivent répondre à la syntaxe SQL en général et éventuellement aux singularités des différents éditeurs de SGBDR.

Les requêtes SQL permettent d'accomplir une action sur une base de données comme

* + - la sélection d'informations,
    - la création de tables,
    - l'ajout, la suppression ou la modification des enregistrements.

**Requête SQL de sélection :**

Afin d’interroger une BD via PDO, nous utilisons la méthode query qui prend en paramètre la requête à exécuter.

Exemple:

$req=« select \* From maTable »;

$reponse = $\_bdd->query($req);

La variable $reponse contiendra un objet contenant la réponse de MySQL qui n’est pas directement exploitable.

Pour exploiter ces données nous utilisons la méthode fetch qui retourne une ligne ou fetchAll qui retourne un tableau contenant toutes les lignes du jeu d'enregistrements

L’un des paramètres des méthodes fetch et fetchAll est l’attribut fetch\_style qui permet de spécifier le type de la valeur de retour de fetch et fetchAll. Cette valeur est une constante de la classe PDO et qui commence par PDO::FETCH\_\*.

* PDO::FETCH\_BOTH (défaut): retourne un tableau indexé par les noms de colonnes et aussi par les numéros de colonnes, commençant à l'index 0, comme retournés dans le jeu de résultats
* PDO::FETCH\_OBJ : retourne un objet anonyme avec les noms de propriétés qui correspondent aux noms des colonnes retournés dans le jeu de résultats

**La gestion des sessions**

* + La session est un mécanisme permettant de mettre en relation les différentes requêtes du même client sur une période de temps donnée.
  + Les sessions permettent de conserver des informations relatives à un utilisateur lors de son parcours sur un site web
  + Des données spécifiques à un visiteur pourront être transmises de page en page afin d'adapter personnellement les réponses d'une application PHP
  + 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 et de créer un répertoire. 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\_ID = 7edf48ca359ee24dbc5b3f6ed2557e90 ?>

**Destruction d’une session**

La session en cours peut être détruite par la fonction session\_destroy(). Cette commande supprime toutes les informations relatives à l'utilisateur.

**session\_destroy();**

**Les variables de session**

Les variables de session sont chargées dans une session par l'intermédiaire de la fonction session\_register()

<?php

session\_start();

session\_register("nom\_variable");

...

session\_register("nom\_variableN");

?>

Une fois la variable enregistrée, elle est accessible à travers le tableau associatif **$\_SESSION["nom\_variable"]**

**Les fonctions de sessions**

session\_start() -- Initialise les données de session

session\_id() -- Affecte et/ou retourne l'identifiant de session courante

session\_name() -- Affecte et/ou retourne le nom de la session courante

session\_register() -- Enregistre une variable dans la session courante

session\_destroy() -- Détruit toutes les données enregistrées d'une session

session\_is\_registered() -- Indique si une variable a été enregistrée dans la session ou pas

session\_unregister() -- Supprime une variable dans la session courante

session\_unset() -- Détruit toutes les variables de session

session\_cache\_expire() -- Retourne la date d'expiration du cache de la session

session\_save\_path() -- Affecte et/ou retourne le chemin de sauvegarde de la session courante

session\_decode() -- Décode les données de session à partir d'une chaîne

session\_encode() -- Encode les données de session dans une chaîne

Commençons maintenant la création de notre site de gestion de personne.

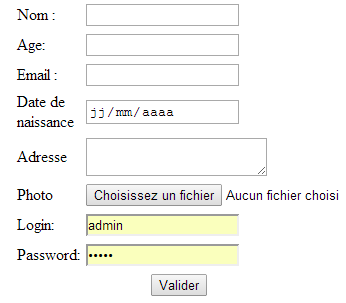
La base de données à utiliser est la base « formaweb» contenant la table personne dont voici la structure :

|  |  |  |  |
| --- | --- | --- | --- |
| **Colonne** | **Type** | **Null** | **Défaut** |
| ***id*** | int(11) | Non |  |
| nom | varchar(50) | Non |  |
| age | int(11) | Non |  |
| email | varchar(50) | Non |  |
| adresse | varchar(255) | Non |  |
| photo | varchar(255) | Non |  |
| date | varchar(50) | Non |  |
| login | varchar(25) | Non |  |
| pwd | varchar(25) | Non |  |

Le champ ‘id’ sera incrémenté automatiquement.

Afin de finaliser toute la partie CRUD, nous présentons ici l’interface de la partie insertion ainsi que le code. Ensuite, vous aurez à terminer l’affiche de la liste des personnes, la modification et la suppression de personne.

Le formulaire d’inscription est le suivant :



Le code source permettant l’affichage du formulaire et la validation de l’inscription après le remplissage du formulaire et la validation par le bouton valider est le suivant :

< !Doctype html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title> Ajout </title>

</head>

<body>

**<!-- nous avons ajouté enctype="multipart/form-data" pour permettre de gérer les uploads -->**

<form method="post" enctype="multipart/form-data">

<table width="200" align="center" >

<tr>

<td>Nom :</td>

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

<tr>

<td>Age: </td>

<td> <input name="age" type="number" /></td>

</tr>

<td>

Email :

</td>

<td> <input name="mail" type="email" required /></td>

</tr>

<tr>

<td>Date de naissance</td>

<td> <input name="dn" type="date" required></textarea></td>

</tr>

<tr>

<td>Adresse</td>

<td> <textarea name="adr"></textarea></td>

</tr>

<tr>

<td>Photo</td>

**<!—L’utilisation du type file permet de mieux gérer les uploads -->**

<td> <input name="fichier" type="file" /></td>

</tr>

<tr>

<td>Login: </td>

<td> <input name="login" type="text" /></td>

</tr>

<tr>

<td>Password: </td>

<td> <input name="pwd" type="password"/></td>

</tr>

<tr>

<td colspan="2" align="center"><input name="ok" type="submit" /> </td>

</tr>

</table>

</form>

Quelques astuces

Afin de gérer les fichiers et les pièces jointes on copie toujours les documents dans un dossier que nous spécifions. Vous pouvez nommer vos fichiers avec l’id de l’entité qui est unique ca vous garantit la non duplicité.

copy($\_FILES['fichier']['tmp\_name'],'docs/'.$\_FILES['fichier']['name']);

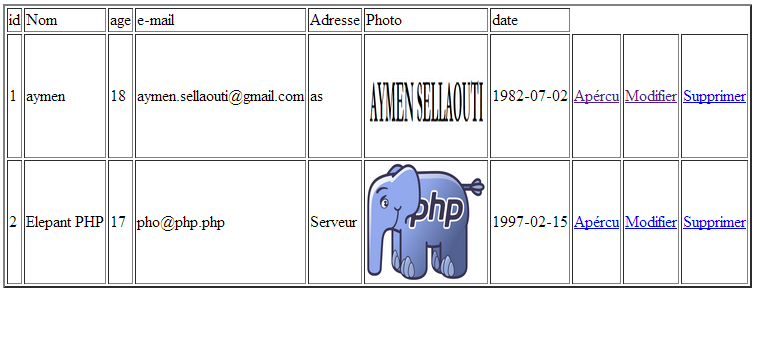
Pour accéder aux données d'une pièce jointe on n’utilise plus de post mais plutôt FILES

header("page"); permet de rediriger vers page.

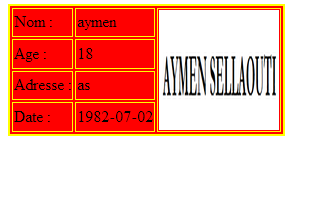
Une fois l’insertion des personnes terminée, vous devez ajouter une page permettant la gestion des personnes. Cette page est appelée « GestionPersonne.php ». Dans cette page il y aura l’affichage de la liste des personnes existant dans la base de données. Pour chaque personne, nous aurons trois liens permettant d’afficher les détails d’une personne, de la supprimer ou de la modifier la personne. Après la suppression ou la modification, la page de gestion sera réaffichée.

**ASTUCE : Afin de fournir l’information concernant l’id de la personne à supprimer, afficher ou modifier, pensez à l’intégration de la variable id dans l’url de la page cible.**

Voici les interfaces des différentes pages du projet

Page « GestionPersonne.php »

L’appui sur le lien ‘Apércu’ renvoi à la page ‘apercu.php’ :



Page « apercu.php »

En appuyant sur le lien Modifier, une page « modifier.php» contenant les détails de la personne à modifier est affichée. L’utilisateur aura la possibilité de changer les informations relatives à la personne sélectionnée. L’appuie sur le bouton ‘modifier’ permet de confirmer la modification et de réafficher la page « GestionPersonne.php».



Page « modifier.php »

L’appui sur le lien supprimer dans la page « GestionPersonne.php » permet de supprimer la personne sélectionnée et de réafficher la liste des personnes mise à jour.

**Authentification et Gestion des sessions**

Afin de sécuriser votre application, ajouter une page d’authentification permettant de limiter l’accès aux utilisateurs déjà inscrit. Dans le cas où l’utilisateur n’est pas inscrit, redirigez le vers le formulaire d’inscription.

Protéger les autres pages en utilisant les sessions. Si un utilisateur n’est pas connecté et qu’il veuille accéder à une des pages, redirigez-le vers la page d’authentification.

Remarque :

Gérer la partie CSS et JS. Tout apport est un plus.

L’ajout de nouvelles fonctionnalités l’est aussi.