MINI_PROJET : DÉVELOPPEMENT APPLICATION PHP OBJET

Lancer MAMP ou WAMP ou autres, ouvrir phpMyAdmin et créer une nouvelle bdd avec comme nom « TPL3INFO».
Créer ensuite une nouvelle table « utilisateur » ayant les attributs qui sont définis plus loin.
Créer un dossier appelé « projet » et y créer l'arborescence suivante :

- projet/classes/
- projet/js/
- projet/inc/
- projet/images/
- projet/admin/

ATTENTION :
Toutes les classes seront sauvegardées dans le dossier « classes/ ».
Tous les fichiers de traitement relatifs à l'admin seront dans le dossier « admin/ »
Pour chaque partie on donne le squelette qu'il faudra compléter.

1. La base de données

On demande de développer la classe « Mysql » qui permet de se connecter à la bdd. Cette classe contient :
- des attributs privés :
    - serveur (host) : définir « locahost » comme valeur par défaut
    - login
    - mot de passe
    - nom de la bdd
    - identifiant de connexion : la valeur retourné par mysql_connect() ou autre fonction.
    -

- Les méthodes publiques suivantes :
    - set_serveur($s),
    - set_login($s),
    - set_mdp($s),
    - set_bdd($s) : initialise la variable respective.
    - connexion() : permet de se connecter à la bdd et d'affecter l'identifiant de la connexion à l'attribut privé.
    - get_cnx() : renvoi la valeur de l'identifiant de connexion.
    - deconnexion() : permet de fermer la connexion à la bdd
    - requete($q) : permet d'exécuter une requête $q et de renvoyer son résultat.

Dans chaque méthode, en cas d'erreur afficher un message et interrompre le traitement en utilisant la fonction exit().

Ci-dessous le squelette du fichier qu'il faut compléter :

```
<?php
class Mysql
{
private $_serveur = "……";
private $_login;
```

```
private $_mdp;
private $_bdd;
private $_cnx;
public function set_serveur($s)
{
$this->_serveur = $s;
}
public function set_login($s)
{
$this->…… = $s;
}
public function …… ($s)
{
$this->_mdp = $s;
}
public function …… ($s)
{
$this->……= $s;
}
public function ……_cnx()
{
return $this->_cnx;
}
public function connexion()
{
$this->_cnx = mysql_connect($this->_serveur, $this->……, ……);
if (……)
exit("Erreur de connexion bdd : " . mysql_error());
if (!mysql_select_db(……))
exit("Erreur : bdd inexistante : " . ……());
}
public function requete($q)
{
$res = ……($q);
if (!$res) exit("<pre>Erreur dans la requete [$q] : " . ……() . "</pre>");
return ……;
}
}
?>
```

Enregistrer le fichier (dans « /classes/Mysql.php » Créer et compléter le fichier « inc/connexion.php » dont le code est le suivant :

```
<?php
include(".../Mysql.php");
$bdd = new ...;
...->set_serveur("localhost");
...->set_login("...");
...-> ...(" ...");
...->set_bdd("...");
...->connexion();
?>
```

2. Gestion des utilisateurs

2.1. La classe

Créer la classe « Utilisateur » dont le contenu est :
- Les attributs privés :

- id : identifiant unique, auto-increment
- nom : obligatoire, varchar(50), not null
- prenom : obligatoire, varchar(50), not null
- d_naissance : non obligatoire, date
- mail : taille entre 5 et 50 caractères, unique, obligatoire, varchar(50), not null
- mdp : taille entre 4 et 15 caractères, la valeur par défaut est '1234.

- Les méthodes publiques suivantes :

- Les différents setters (set_nom($s) …) en prenant en compte les remarques relatives à chaque attribut.
- enregistrer($bdd) : insérer un nouvel enregistrement
- supprimer($bdd) : supprimer un enregistrement dont l'identifiant est défini par set_id().

Développer l'interface d'ajout, donc pour le moment ces méthodes sont suffisantes. Ci-dessous le code du fichier classes/Utilisateur.php qu'il faut compléter :

```
<?php
class ...
{
private $_id;
private $_nom;
private ...
private ...
private ...
private ...
public function set_nom($s)
{
if (strlen($s) == 0) exit("Utilisateur : le nom est obligatoire");
$this->... = $s;
}
public function set_prenom($s)
{
...
...
}
public function set_mail($s)
{
if (...) exit("Utilisateur : le ... est obligatoire");
...
}
public function set_mdp($s)
{
if (...) $s = "1234";
if (...) exit("Utilisateur : le mdp doit être compris entre 4 et 15 caractères");
...
```

```
}
public function set_dnaissance($s) // format d'entrés : jj/mm/aaaa
{
$this->_d_naissance = $s;
}
public function set_id($x)
{
$this->_id = $x;
}
public function enregistrer(Mysql $bdd)
{
$q = "INSERT INTO utilisateur (id, nom, ...)
VALUES (null, '$this->_nom', ... )";
return $bdd->...;
}
function supprimer()
{
...
...
}
}
?>
```

2.2 Ajout d'un nouvel utilisateur

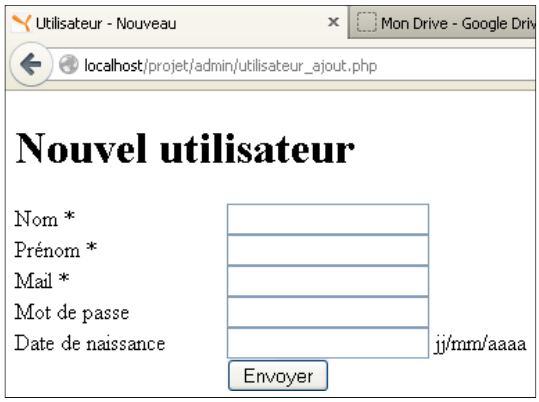
Créer le formulaire d'ajout (ci-contre) qui sera appelé « admin/utilisateur_ajout.php ». Les noms des champs (attributs'id' et 'name' des input) seront dans l'ordre : nom, prenom, mail, mdp, d_naissance, b_ajouter

Pour la balise FORM, définir la valeur de l'attribut ACTION :
« utilisateur_ajout_action.php »

```
<form id="form1" name="form1" method="post"
action="utilisateur_ajout_action.php">
```

Créer et compléter le fichier « /admin/utilisateur_ajout_action.php » dont le code est le suivant :

```
<?php
include(".../connexion.php");
include("...");
$u = new Utilisateur();
$u->set_nom($_REQUEST['nom']);
$u->...
$u->...
$u->...
$u->set_mdp($_REQUEST['mdp']);
if (...->...($bdd))
print "Ajout utilisateur ok.";
?>
```



Tester le formulaire de saisie en ouvrant l'url « http://localhost/projet/admin/utilisateur_ajout.php »

Vérifier depuis phpmyadmin que les données sont bien enregistrées. En cas d'erreurs PHP corriger le code source et tester de nouveau.

2.3 *Liste des utilisateurs existants*

Cette étape consiste à développer la page qui affiche la liste des utilisateurs.

Modifier le fichier « classes/Utilisateur » en ajoutant les méthodes suivantes. ATTENTION : effectuer une copie de sauvegarde de votre fichier !

- Les différent getters (get_nom() …)
- modifier($bdd) : mettre à jour un enregistrement existant
- get_un($bdd, $id) : envoi un objet rempli avec les données de l'utilisateur dont l'id est passé en paramètre.
- get_liste($bdd, $order_by='id', $order_type='ASC') : renvoi le contenu de la table sous forme de tableau d'objets. Si le 2ème paramètre est spécifié, le tri se fait sur le nom de la colonne défini, sinon le tri se fait sur l'id. Le tri se fait selon le 3ème paramètre : ASCendant (par défaut) ou DESCendant.

```
…
public function get_id()
{ return $this->_id; }
public function get_nom()
{ return $this->... }
public function get_...()
{ return ... }
public function ... { ... }
public function ...
public function ...
public function get_un(..., ...)
{
$q = "SELECT ... WHERE ...";
$res = ...->requete(...);
$row = mysql_fetch_array($res);
$u = new ...;
$u->set_d_naissance($row['d_naissance']);
$u->set_id($row['...']);
$u->set_mail(...);
$u->set_... (...);
$u->...
$u->...
... $u;
}
public function get_liste(..., ..., ...)
{
$q = "SELECT * FROM utilisateur ORDER BY $order_by $order_type";
$res = $bdd->requete($q);
while($row = mysql_fetch_array($res))
{
$u = new Utilisateur();
$u->set_d_naissance($row['d_naissance']);
```

```
$u->set_id($row['id']);
$u->set_mail($row['mail']);
$u->set_mdp($row['mdp']);
$u->set_nom($row['nom']);
$u->set_prenom($row['prenom']);
$a_user[] = $u;
}
return $a_user;
}
```

3. *Gestion des utilisateurs – Les écrans.*

Pour la création des utilisateurs, créer et développer les pages php suivantes :
- « utilisateur_ajout.php » : il s'agit d'un simple formulaire pour saisir les données d'un nouvel utilisateur. Attention, on n'aura pas de champ « id » dans cet écran.
- « utilisateur_ajout_action.php » : cette page sera appelée quand on valide la page précédente. Elle récupère les données depuis le formulaire et utilise les classes développées ci-dessus afin d'insérer un nouvel enregistrement dans la bdd.

Utiliser le même principe pour développer les 2 pages nécessaires à la modification d'un utilisateur dont l'identifiant est passé en paramètre (par la méthode get())
Développer une page « utilisateur_supp_action.php » qui permet de supprimer un utilisateur dont l'identifiant est passé en paramètre (par la méthode get())
Développer une page « utilisateur_liste.php » qui affiche la liste des utilisateurs avec des liens nécessaires vers les pages créés ci-dessus.

4. Améliorations !

Développer des feuilles de style pour améliorer l'affichage de vos pages et des scripts Javascript pour dynamiser vos pages web et contrôler le contenu des champs des formulaires.

NB : ***Vous pouvez mettre en œuvre toute technologie apprise durant votre parcours de licence !***