Accès objet à Mysql avec PHP5

par mbella demazy

Date de publication : 29 mars 2011

Dernière mise à jour :

Cet article explique : Comment accéder à une base de donnée Mysql en orienté objet avec PHP 5 en utilisant l'extension Mysqli.

I - Introduction

Lorsqu'on décide de faire du PHP orienté objet sans avoir à faire de la programmation procédurale, très souvent on est coincé lorsqu'il s'agit d'accéder à une base de donnée et plus particulièrement à Mysql. En effet, l'extension Mysql très souvent utilisée en PHP pour se connecter à une base de données Mysql est une extension exclusivement procédurale et n'offre de ce fait aucun mécanisme d'accès aux données en orienté objet.

Cependant, la version 5 de PHP, à savoir PHP5 ayant été conçu pour accroître les capacités orientées objet de PHP, a prévu des moyens de le faire , et cet article nous montrera comment. Grâce à l évolution qu'a connu le langage PHP entre la version 4 et la version 5 la programmation orienté objet avec ce langage a connu pas mal d'évolution et d'amélioration ; et l'accès aux données n'est pas resté en marge de cette logique. C'est ainsi que l'accès objet à une base de donnée Mysql est désormais possible grâce à l'utilisation de l'extension Mysqli (ou Mysql improved pour améliorée) de PHP.

Cette extension a les avantages suivants par rapport à ces prédécesseurs :

- Une interface aussi bien procédural qu'orienté objet ;
- Le support des commandes préparées ;
- Le support des commandes multiples ;
- Le support des transactions ;
- Des capacités de débogage avancées.

II - Prérequis

Pour une bonne compréhension de cet article vous devez avoir :

- Les bases de la programmation avec le langage PHP ;
- La POO et son implémentation dans PHP 5 ;
- Les itérateurs en PHP5 - L'architecture MVC ;
- Le langage SQL et son utilisation avec le SGBD Mysql ;
- L'extension Mysqli de PHP 5.

III - Les classes de l'extension Mysqli.

Mysqli possède les classes suivantes à l'origine de ces avantages :

- La classe **mysqli** : elle permet de créer des objets de types mysqli_object. Cette dernière possède des méthodes et des propriétés nécessaires pour permettre par exemple de se connecter à la base de données, d'envoyer des requêtes préparées à cette dernière, ou alors d'effectuer des transactions ;
- La classe **mysqli_result** : elle permet de gérer les résultats des requêtes SQL effectuée à l'aide de l'objet mysqli précédent. C'est ainsi qu'elle possède des méthodes et des propriétés permettant de manipuler de plusieurs manières les résultats issus d'une requête SQL en rendant par exemple ceux-ci sous forme de tableau associatif, de tableau indicé, d'obtenir le nombre de tuple d'un résultat etc ;
- La classe **mysqli_stmt** : elle permet d'effectuer des requêtes préparées sur la base de données ;
- La classe **mysqli_driver** : elle permet la gestion des drivers mysqli ;

III-1 - La classe mysqli.

Cette classe possède un certain nombre de propriétés et de méthodes. Nous nous limiterons à n'évoquer que quelques-unes.

III-1-1 - Propriétés

mysqli->affected_rows : contient le nombre de lignes affectées par la dernière requêtes INSERT, UPDATE ,REPLACE ou DELETE.

mysqli->insert_id : retourne l'identifiant automatiquement généré pour un attribut déclaré AUTO_INCREMENT.

III-1-2 - méthodes

mysqli ([string $host [, string $username [, string $passwd [, string $base [, int $port [, string $socket]]]]]]) : permet de créer un objet mysqli, afin de se connecter à la base de données Mysql.

close(void) : permet de fermer une connexion à la base de donnée Mysql ; elle retourne true en cas de succès et false si non.

query(string $query [, int $mode]) : permet d'exécuter une requête sur la base de données ; le paramètre $mode est une constante valant MYSQLI_USE_RESULT ou MYSQLI_STORE_RESULT (valeur par défaut) suivant le comportement désiré.

select_db(string $db) : permet de sélectionner une base de données.

III-2 - La classe mysqli_result

III-2-1 - Propriétés

mysqli_result->field_count : récupère le nombre de champs dans un jeu de résultat.

mysqli_result->num_rows : retourne le nombre de ligne d'un résultat.

III-2-2 - Méthodes

fetch_all ([int $resulttype]) : permet de lire tous les résultats d'une requête et de les retourner sous forme de tableau associatif, numérique ou les deux à la fois ceci suivant la valeur du paramètre $resulttype qui peut être : MYSQLI_ASSOC pour un tableau associatif MYSQLI_NUM pour un tableau numérique ; cette valeur étant la valeur par défaut, MYSQLI_BOTH pour les deux.

fetch_array ([int $resulttype]) : permet de retourner une ligne de résultat sous forme d'un tableau associatif, d'un tableau indexé, ou les deux suivant la valeur du paramètre $resulttype qui peut être : MYSQLI_ASSOC, MYSQLI_NUM, MYSQLI_BOTH (valeur par défaut).

fetch_assoc (void) : permet de récupérer une ligne de résultat sous forme de tableau associatif ;

fetch_object ([string $class_name [, array $params]]) : retourne la ligne courante d'un jeu de résultat sous forme d'objet ; en paramètre nous avons $class_name qui représente le nom de la classe à instancier , et $params un tableau contenant la liste des paramètres à passer au constructeur de cette classe.

fetch_row (void) : permet de récupérer une ligne de résultat sous forme de tableau indexé.

III-3 - La classe mysqli_stmt

III-3-1 - Propriétés

mysqli_stmt->affected_rows : retourne le nombre total de lignes modifiées, effacées, ou insérées par la dernière requête.

mysqli_stmt->insert_id : récupère l'ID généré par la dernière requête INSERT.

mysqli_stmt ->num_rows : retourne le nombre de ligne d'un résultat Mysql.

III-3-2 - Méthodes

prepare (string $query) : permet de préparer une requête SQL pour exécution.

reset (void) : permet d'annuler une requête préparée.

III-4 - La classe mysqli_driver

III-4-1 - Propriétés

Cette classe est pour le moment très peu documentée mais possède néanmoins les propriétés suivantes :

- **client_version** : qui indique la version du client ;
- **driver_version** : qui indique la version du driver Mysqli.

III-4-2 - Méthodes

Il n'y en a que deux pour le moment et sont également peu documentées il s'agit de :

- **embedded_server_end(void)** : qui permet d'arrêter le serveur embarqué ;
- **embedded_server_start (bool $start , array $arguments , array $groups)** : qui permet d'initialiser et de démarrer le serveur embarqué.

IV - Mise en □uvre

Pour illustrer l'utilisation de mysqli pour accéder à Mysql, nous allons mettre en place une application web basique. L'architecture ici étant l'architecture MVC (Model View Controller) cella nous permettra de séparer l'accès aux données de la présentation et des traitements.

Dans l'exemple que nous prenons, l'aspect sécurité est peu ou pas assez pris en compte ; il est donc possible que le code qui suit puisse être vulnérable aux attaques comme les SQL Injections ou toute autre attaque courante.

IV-1 - La base de données

Pour pouvoir observer comment tout ceci est mis en □uvre, nous avons besoin d'une base de données MySql. Cette base de données s'appellera " cabinet" et possèdera une table, la table " produit ". Le script SQL correspondant pour cette base de données est le suivant :

```
-- Base de données: `cabinet`
--
CREATE DATABASE `cabinet` DEFAULT CHARACTER SET latin1 COLLATE latin1_swedish_ci;
USE cabinet;

-- --------------------------------------------------------

--
-- Structure de la table `produit`
--

CREATE TABLE `produit` (
`prd_id` bigint(20) NOT NULL
 auto_increment COMMENT 'identification du produit, clé primaire de la table produit',
`prd_libelle` varchar(200) NOT NULL COMMENT 'libellé du produit',
`prd_reference` varchar(20) NOT NULL COMMENT 'reférence du produit:; doit etre unique',
`prd_fabricant` varchar(200) NOT NULL COMMENT 'fabricant du produit',
`prd_qteStock` int(11) NOT NULL COMMENT 'quantite de produit en stock',
`prd_qteAllert` int(11) NOT NULL COMMENT 'le stock d''alerte pour necessité un approvisionnement',
`prd_unite` varchar(100) default NULL COMMENT 'unité du produit',
PRIMARY KEY (`prd_id`),
UNIQUE KEY `prd_reference` (`prd_reference`)
) TYPE=InnoDB AUTO_INCREMENT=16 ;

--
-- Contenu de la table `produit`
--

INSERT INTO `produit` (`prd_id`, `prd_libelle`, `prd_reference`, `prd_fabricant`, `prd_qteStock`,
 `prd_qteAllert`, `prd_unite`)
VALUES
(4, 'alcool', 'alc', 'novartis', 10, 20, 'cm'),
(6, 'benzene', 'ben', 'chococam', 10, 2, 'cm'),
(8, '24DNPH', 'DNPH', 'areva', 124588, 1250, 'ml'),
(9, 'acide chlohydrique', 'hcl', 'chimie', 124588, 0, 'ml'),
(10, 'aspirine', 'asp', 'chimie', 124588, 0, 'ml'),
(11, 'uranium 235', 'U235', 'AIEA', 125, 12, 'bekerel'),
(12, 'polonium 210', 'pol', 'AIEA', 25, 0, 'ur'),
(13, 'bromure', 'br', 'chimie', 1254, 0, 'l'),
(14, 'urée', '201010', 'engrais', 100, 0, 'sac'),
(15, 'phosphate', 'ph', 'arcelor', 145, 0, 'cm');
```

IV-2 - Connexion et déconnexion à la base de données

Nous allons illustrer cela, en écrivant deux fonctions que nous utiliserons tout au long de cette article, car toutes les manipulations que nous aurons à faire et qui porteront sur les données nécessiteront qu'on se connecte puis qu'on se déconnecte à la base de donnée.
Etant donné que la connexion à la base de donnée nécessite des paramètres de connexion que sont l'hôte, l'utilisateur, le login, le mot de passe ainsi que le nom de la base de donnée à laquelle on souhaite se connecter, ces paramètres seront également définis. De ce fait, nous allons créer un nouveau fichier config.class.php qui contiendra les méthodes connect() et disconnect() permettant respectivement de se connecter et se déconnecter

Le code complet de ce fichier est le suivant :

```
private $HOST = "localhost";
private $USER = "root";
private $PASSWORD = "";
private $DATABASE = "cabinet";
  public function connect()
 {
 //fonction permettant de ce connecter a la bd en utilisant l'extension mysqli
    //elle retourne l'objet de connection
 $con = new mysqli($this->HOST,$this->USER,$this->PASSWORD,$this->DATABASE);
 $con->set_charset("utf8"); //definition du jeu de caractère par defaut du client
 if(mysqli_connect_error())
```

```
 $con = 'echec de la connexion à la base de données ('. mysqli_connect_errno().')'. mysqli_connect_error();
 return $con;
  }
  public function disconnect(mysqli $con)
  {
 //cette fonction, permet de ce déconnecter de la BD
 //elle prend en entrée un objet de connection de type mysqli
 $val = $con->close();
 if($val!= true)
  {
  echo("echec de la fermeture de la connection a la BD");
   }
 }
}
```

IV-3 - La classe Produit

Précédemment nous avons fait allusion à la table "produit" dans notre base de données, maintenant, nous allons écrire la classe correspondante et nous écrirons également les méthodes d'accès aux enregistrements de cette table. La classe " Produit " dans un premier temps sera une classe basique contenant juste les propriétés, les méthodes d'accès aux propriétés, (les accesseurs de préfixe get_ et mutateurs de préfixe set_) et le constructeur de la classe. Une méthodes comme get_id() permetra d'obtenir la valeur de la propriété, $prd_id de la classe Produit ; tantdis que la méthode set_id($id) permet de définir ou de modifier cette valeur.

Le code de cette page est le suivant :

```
class Produit
{ private $prd_id;
private $prd_libelle;
private $prd_reference;
private $prd_fabricant;
private $prd_qteStock;
private $prd_qteAllert;
private $prd_unite;
public function __construct()
{//le constructuer de la classe produit
$this->prd_id = "";
$this->prd_libelle = "";
$this->prd_reference = "";
$this->prd_fabricant = "";
$this->prd_qteStock = "";
$this->prd_qteAllert = "";
$this->prd_unite = "";
}
public function get_id()
{
//retourne l'id du produit courant.
return $this->prd_id;
}
public function set_id($id)
{ //definit la valeur de l'id du produit
$this->prd_id = $id;
}
public function get_libelle()
{
//retourne le libelle du produit courant
return $this->prd_libelle;
}
public function set_libelle($libelle)
{
//defini le libelle du produit
$this->prd_libelle = $libelle;
}
public function get_reference()
{
//retourne la reference du produit courant
```

```
return $this->prd_reference;
}
public function set_reference($reference)
{
//definila reference du produit courant
$this->prd_reference= $reference;
}
public function get_fabricant()
{
//retourne le fabricant du produit
return $this->prd_fabricant;
}
public function set_fabricant($fabricant)
{
//defini le fabricant du produit
$this->prd_fabricant=$fabricant;
}
public function get_qteStock()
{
//retourne la quantité en stock
return $this->prd_qteStock;
}
public function set_qteStock($qteStock)
{
//defini la quantité en stock
$this->prd_qteStock=$qteStock;
}
public function get_qteAllert()
{ //retourne la quantité d'alerte
return $this->prd_qteAllert;
}
public function set_qteAllert($qteAllert)
{ //defini la quantité d'alerte
$this->prd_qteAllert = $qteAllert;
}
public function get_unite()
{ //retourne l'unite
return $this->prd_unite;
}
public function set_unite($unite)
{ //retourne l'unite
$this->prd_unite = $unite;
}
}
```

Le code suivant, quant à lui, contient la liste des méthodes complémentaires de la classe " Produit ". Celles-ci agiront sur les données, chacune d'une manière précise :

- **GetOneProduit** : elle prend en entrée l'identifiant de la connexion à Mysql et l'identifiant d'un produit dans la table produit, afin de le retourner sous la forme d'un objet de type " Produit " ;
- **GetAllProduit** : cette méthode prend en entrée l'identifiant de connexion à Mysql, l'ordre dans lequel les résultats devront être retournés, et la colonne suivant laquelle les résultats seront triés. Elle retourne la liste des produits présents dans la table " produit " sous forme d'un objet de type " Moniterateur " ;
- **UpdateProduit** : cette méthode permet de mettre à jour une ligne dans la table " produit ", elle prend en entrée l'identifiant de connexion à Mysql et d'autres paramètres correspondant aux propriétés de la classe " Produit ". Elle retourne true en cas de succès ou un message d'erreur si non ;
- **SetProduit** : cette méthode qui prend en entrée l'identifiant de connexion à Mysql, permet d'insérer un enregistrement dans la table " produit ", elle retourne true en cas de succès, ou un message d'erreur si non ;
- **DeleteOneProduit** : elle permet de supprimer un enregistrement dans la table produit; elle prend en entrée l'identifiant du produit à supprimer dans la table et l'identifiant de connexion à la base de données Mysql.

```
listing 4: les méthodes de la classe Produit et la classe Moniterateur
public function GetOneProduit(mysqli $id_con,$id)
{
/*
* cette methode permet d'obtenir un tuple de la table produit à partir
```

```
* de son id et de l'id de connection à la bd
* elle retourne un objet produit
*/
try
{
if(ctype_digit($id))
{
$sql = "SELECT * FROM produit WHERE prd_id ='$id'";
$result = $id_con->query($sql);
$ligne = $result->fetch_assoc();
$unobjet = new Produit();
$unobjet->set_id($ligne['prd_id']);
$unobjet->set_libelle($ligne['prd_libelle']);
$unobjet->set_reference($ligne['prd_reference']);
$unobjet->set_qteStock($ligne['prd_qteStock']);
$unobjet->set_qteAllert($ligne['prd_qteAllert']);
$unobjet->set_unite($ligne['prd_unite']);
$unobjet->set_fabricant($ligne['prd_fabricant']);
}
else
{
throw new Exception("Erreur sur l'identificateur");
}
}
catch (Exception $e)
{
echo $e->getMessage();
}
return $unobjet;
}

public function GetAllProduit(mysqli $id_con, $order, $by)
{
/*cette méthode permet d'obtenir la liste des produits da la base de données
* elle prend en entrée:
* $id_con : l'objet de connexion a la base de donnée,
* $by : la collone suivant laquelle les resultats seront ordonnés
* $order : l'odre de tri des résultats : ASC ou DESC
* cette méthode retourne un itérateur de type Moniterateur
* */
$tab = array('prd_id','prd_libelle','prd_reference','prd_fabricant','prd_qteStock','prd_qteAllert','prd_unite');
try
{
if((strtoupper($order)=="ASC" || strtoupper($order)=="DESC") && in_array($by,$tab))
{
$sql = "SELECT * FROM produit ORDER BY $by $order";
$myIt = new Moniterateur($sql,$id_con);
return $myIt;
}
else
{
throw new Exception("Erreur de paramètres");
}
}
catch (Exception $e)
{
return $e->getMessage();
}
}
public function
 UpdateProduit(mysqli $id_con,$id,$libelle,$reference,$fabricant,$qteStock,$qteAllert,$unite)
{
/* * cette methode met a jour un tuple de la table produit à
* partir de son ID
* en entrée on a les differents champs correspondant au
* propriété de la table produit dans la BD et l'id de la connexion a mysql
* */
$sql= "UPDATE produit SET
prd_libelle ='$libelle',prd_fabricant = '$fabricant',prd_qteStock='$qteStock',
prd_qteAllert= '$qteAllert', prd_unite = '$unite'
WHERE prd_id = '$id'";
```

```
try {
$result = $id_con->query($sql);
if(!$result)
{
throw new Exception("Erruer lors de la mise à jour de l'enregistrement");
}
}
catch
(Exception $e)
{$result = $e->getMessage();
}return $result;
}
public function SetProduit(mysqli $id_con)
{
/*
* cette methode permet d'insérer un nouvel enregistrement
* dans la table produit;
* en entrée on a l'identifiant de la connexion à la BD
*/
$id = $this->get_id();
$libelle = $this->get_libelle();
$reference = $this->get_reference();
$fabricant = $this->get_fabricant();
$qteStock = $this->get_qteStock();
$qteAllert = $this->get_qteAllert();
$unite = $this->get_unite();
$sql = "INSERT INTO produit VALUES ('$id','$libelle','$reference','$fabricant','$qteStock','$qteAllert','$unite')
try
{
$result = $id_con->query($sql);
if(!$result)
{
throw new Exception("Une erreur c'est produite lors de l'insertion de l'enregistrement dans la BD");
}
} catch (Exception $e)
{$result = $e->getMessage();
}
return $result;
}
public function DeleteOneProduit($id,mysqli $id_con)
{
/*
* cette methode se charge de supprimer un tuple dans la table
* produit identifié par son id
* en entrée on a :
* - l'id du produit à supprimer
* - l'identifiant de connexion à la BD
*/
try
{
if(ctype_digit($id))
{
$sql = "DELETE FROM produit WHERE prd_id = '$id'";
$result = $id_con->query($sql);
}
else
{
throw new Exception("Erreur sur l'identificateur");
}
}
catch (Exception $e)
{
$result = $e->getMessage();
}
return $result;
}
//la classe Moniterateur qui implémente l'interface Iterator
class Moniterateur implements Iterator
{private $result = null;
private $current = null;
private $key = 0;
```

```
protected $sql = null;
function __construct($sql,mysqli $con)
{//le constructerur de la classe: il prend en entrée une requette sql
//et un identifiant de connexion à la base de données mysql
$this->result = $con->query($sql);
}
public function current()
{//retourne la valeur de l'élément courant de notre collection
return $this->current;
}
public function key()
{
//retourne la clé de lélément corant de notre collection
return $this->key;
}
public function next()
{
//cette méthode retourne l'élément suivant de notre collection
$this->current = $this->result->fetch_assoc();
$this->key++;
}
public function rewind() {
//on déplace le pointeur vers la première enregistrement de notre collection
$this->result->data_seek(0);
$this->key = -1;
$this->next();
}
public function valid()
{
//on teste si le résultat courrant esr correcte: en d'autre terme s'il s'agit d'un tableau
return is_array($this->current);
}
}
```

On se rend bien compte en lisant les méthodes de cette classe que la logique est pratiquement la même :

- Dans un premier temps on écrit la requête SQL qui doit être exécutée ;
- Ensuite on l'exécute, via la méthode " query " de l'objet mysqli ;
- Enfin les résultats sont récupérés et traités en utilisant la méthode " fetch_assoc " de l'objet mysqli_result en vu d'être retourner sous le format souhaiter : un objet ou un itérateur.

IV-4 - le contrôleur de la classe " Produit "

Maintenant nous allons créer un fichier ctrl.produit.php qui contiendra les codes nécessaires aux différents traitements qui seront effectués. Avant tout traitement, la méthode " connect " de connexion à la base de données est appelée et à la fin de chaque traitement, la méthode de déconnexion, " disconnect " est appelée. Dans chaque partie il y est question d'un traitement spécifique appelant une méthode précise de la classe produit

- **$operation == new** : ici le traitement à effectuer est l'enregistrement d'une nouvelle ligne dans la table " produit " ;
- **$operation == update** : ici le traitement c'est la mise à jour d'un enregistrement dans la table " produit " ;
- **$operation == delete** : ici le traitement c'est la suppression d'un enregistrement dans la table "produit".

```
/include_once 'config.class.php';
$cobj = new Config();
$operation = trim($_GET['action']);
if(isset($_GET['id'])&& ctype_digit($_GET['id']))
{
$id= trim($_GET['id']);
}
$mod = trim($_GET['mod']);
$id_con = $cobj->connect();
switch ($mod) {
```

```
case "produit":{//gestion de ce qui concerne la table produit
if($operation == 'update'){
$prd = new Produit();
$prd->set_id($id);
$prd->set_fabricant(trim($_REQUEST['fabricant']));
$prd->set_libelle(trim($_REQUEST['nom_produit']));
$prd->set_qteAllert(trim($_REQUEST['stock_allert']));
$prd->set_qteStock(trim($_REQUEST['qte_stock']));
$prd->set_reference(trim($_REQUEST['reference']));
$prd->set_unite(trim($_REQUEST['unite']));
$result = $prd->UpdateProduit($id_con,$prd->get_id(),$prd->get_libelle(),$prd->get_reference(),$prd-
>get_fabricant(),
$prd->get_qteStock(),$prd->get_qteAllert(),$prd->get_unite());
$cobj->disconnect($id_con);
}
if($operation =='delete'){
$prd = new Produit();
$result = $prd->DeleteOneProduit($id,$id_con);
$cobj->disconnect($id_con);}
if($operation =='new'){
$prd = new Produit();
$prd->set_fabricant(trim($_REQUEST['fabricant']));
$prd->set_libelle(trim($_REQUEST['nom_produit']));
$prd->set_qteAllert(trim($_REQUEST['stock_allert']));
$prd->set_qteStock(trim($_REQUEST['qte_stock']));
$prd->set_reference(trim($_REQUEST['reference']));
$prd->set_unite(trim($_REQUEST['unite']));
$result = $prd->SetProduit($id_con);
$cobj->disconnect($id_con);
}
break;}}
```

IV-5 - Test de l'application

Nous allons maintenant utiliser les méthodes de notre classe en vue d'illustrer de manière pratique ce que nous obtenons comme résultat.

IV-5-1 - Insertion d'un nouvel enregistrement dans la table " produit "

Nous allons dans un premier temps créer un simple formulaire qui va nous permettre d'enregistrer des données dans la table "produit", le code complet de ce formulaire est le suivnat :

```
listing 6 la vue nouveau produit: new_produit.php
<!DOCTYPE
 html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Nouveau Produit</title></head>
<body>
<div>
<form action="ctrl.produit.php?action=new&mod=produit" name="form_produit" id="form_produit"
 method="post" >
<fieldset>
<legend>Gestion des Produits</legend>
<table>
<tr>
<td><label>Nom Produit</label></td>
<td><input name="nom_produit" type="text" id="nom_produit"/></td>
</tr>
<tr>
<td><label>Fabricant</label></td>
<td><input name="fabricant" type="text" id="fabricant" /></td>
</tr>
```

```
<tr>
<td><label>Quantité en Stock</label></td>
<td><input name="qte_stock" type="text" id="qte_stock"/></td>
</tr>
<tr>
<td><label>Stock d'alerte</label></td>
<td><input name="stock_allert" type="text" id="stock_allert"/></td>
</tr>
<tr>
<td><label>Unité Produit</label></td>
<td><input name="unite" type="text" id="unite" /></td>
</tr>
<tr>
<td><label>Réference Produit</label></td>
<td><input name="reference" type="text" id="reference"/></td>
</tr>
<tr>
<td> </td>
<td> </td>
</tr>
<tr>
<td align="center"><input name="Enregistrer" type="submit" value="Enregistrer" /></td>
<td align="center"><input name="Annuler" type="reset" value="Annuler" /></td>
</tr>
</table>
</fieldset>
</form>
</div>
</body>
</html>
```

Et à l'execution, on obtient le résultat suivant :

Fig.1 enregistrement d'un nouveau produit

IV-5-2 - Affichage des données de la table produit

Pour cela, nous allons utiliser un itérateur. Il s'agit d'une interface qui fournit un ensemble de méthodes permettant de passer en revue une collection d'objet en utilisant l'opérateur foreach comme on le fait de manière classique pour un tableau. Nous utiliserons pour notre cas l'itérateur Iterator : il possède les méthodes suivantes que nous redéfinirons :

- current() : retourne l'élément courant de la collection ;
- key() : retourne la valeur de la clé de l'élément courant de la collection ;
- next() : permet de se positionner sur l'élément suivant de la collection ;
- rewind() : permet de se positionner sur le premier élément de la collection ;
- valid() : permet de savoir si l'on a atteint la fin de la collection.

Le code comple est le suivant :

```
 <!DOCTYPE
 html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Liste des Produits</title>
</head>
<body>
<?php
include_once 'config.class.php';
$cobj = new Config();
//connexion a la BD
$id_con=$cobj->connect();
$lst_prd = $prd->GetAllProduit($id_con,'ASC','prd_libelle');
?>
<table width="600" border="1" cellpadding="1" cellspacing="0">
<tr>
<th scope="col">Libelle</th>
<th scope="col">Fabricant</th>
<th scope="col">Unité</th>
<th scope="col">Quantité en stock</th>
<th scope="col">Stock d'allerte</th>
<th scope="col">Opération</th>
</tr>
<?php
foreach ($lst_prd as $prd){
echo(
"<tr>".
"<td><a href='update_produit.php?id=".$prd['prd_id']."'>".$prd['prd_libelle']."</a></td>".
"<td>".$prd['prd_fabricant']."</td>".
"<td>".$prd['prd_unite']."</td>".
"<td>".$prd['prd_qteStock']."</td>".
"<td>".$prd['prd_qteAllert']."</td>".
"<td><a href='ctrl.produit.php?action=delete&mod=produit&id=".$prd['prd_id']."'>Supprimer</td>".
"</tr>");
}//deconnexion de la BD
$cobj->disconnect($id_con);
?>
</table>
</body>
</html>
```

Et à l'execution, on obtient le résultat suivant :

Libelle	Fabricant	Unité	Quantité en stock	Stock d'allerte	Opération
24DNPH	areva	ml	124588	1250	Supprimer
acide chlohydrique	chimie	ml	124588	0	Supprimer
alcool	novartis	cm	10	20	Supprimer
aspirine	chimie	ml	124588	0	Supprimer
benzene	chococam	cm	10	2	Supprimer
bromure	chimie	l	1254	0	Supprimer
phosphate	arcelor	cm	145	0	Supprimer
polonium 210	AIEA	ur	25	0	Supprimer
uranium 235	AIEA	bekerel	125	12	Supprimer
urée	engrais	sac	100	0	Supprimer
vitamine c	novartis	boite	0	0	Supprimer

Fig. 2 : liste des produits

On remarque bien la présence de notre nouvel enregistrement " vitamine c " à la dernière ligne de notre tableau.

IV-5-3 - Details/Modification

Dans cette partie, nous allons maintenant créer une page permettant de voir les détailles et de modifier les données relatives à un produit donné.

Le code complet pour cette page est le suivant :

```
<!DOCTYPE
 html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Mise à jour Produit</title>
</head>
<body>
<div>
<?php
//récupération de l'enregistrement a partir de son id
include_once 'config.class.php';
$id = trim($_GET['id']);
$cobj = new Config();
//connexion a la BD
$id_con=$cobj->connect();
$prd = new Produit();
$ligne = $prd->GetOneProduit($id_con,$id);
?>
<form action="ctrl.produit.php?action=update&id=<?php echo $id;?>&mod=produit" name="form_produit"
 id="form_produit" method="post" >
<fieldset><legend>Mise à jour d'un Produit</legend>
<table>
<tr>
<td><label>Nom Produit</label></td>
<td><input name="nom_produit" type="text" id="nom_produit" value="<?php echo $ligne->get_libelle();?>"/
></td>
</tr>
<tr>
<td><label>Fabricant</label></td>
<td><input name="fabricant" type="text" id="fabricant" value="<?php echo $ligne->get_fabricant();?>"/
></td>
```

```
</tr>
<tr>
<td><label>Quantité en Stock</label></td>
<td><input name="qte_stock" type="text" id="qte_stock" value="<?php echo $ligne->get_qteStock();?>"/></
td>
</tr>
<tr>
<td><label>Stock d'alerte</label></td>
<td><input name="stock_allert" type="text" id="stock_allert" value="<?php echo $ligne-
>get_qteAllert();?>"/></td>
</tr>
<tr>
<td><label>Unité Produit</label></td>
<td><input name="unite" type="text" id="unite" value="<?php echo $ligne->get_unite();?>"/></td>
</tr>
<tr>
<td><label>Référence Produit</label></td>
<td><input name="reference" type="text" id="reference" value="<?php echo $ligne-
>get_reference();?>"/></td>
</tr>
<tr>
<td> </td>
<td> </td>
</tr>
<tr>
<td align="center"><input name="Modiffier" type="submit" value="Modiffier"/></td>
<td align="center"><input name="Annuler" type="reset" value="Annuler" /></td>
</tr>
</table>
</fieldset>
</form>
</div>
</body>
</html>
```

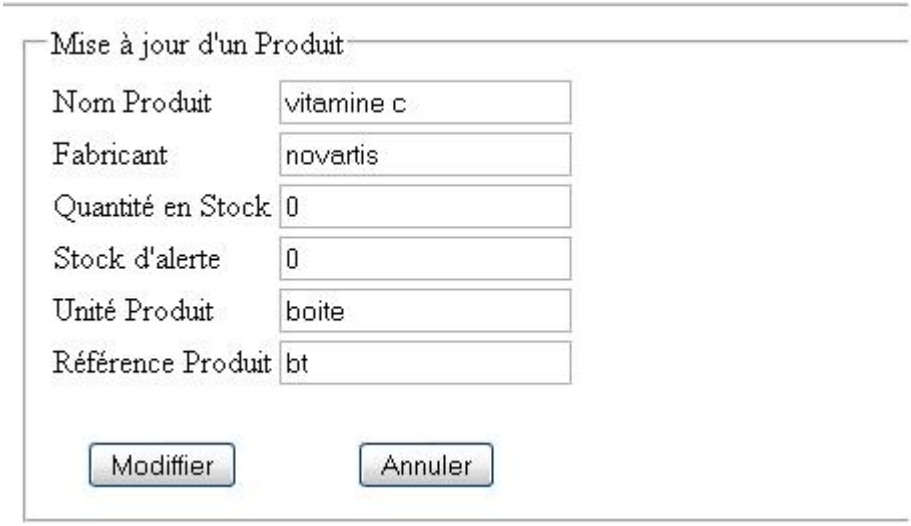

Fig. 3 : détailles / modification d'un produit

IV-5-4 - Suppression d'un enregistrement

La colonne opération du tableau de la liste des produits contient un lien permettant de supprimer un produit de notre base de données ; ce lien pointe directement sur le fichier ctrl.produit.php dont nous avons parlé plus haut. Nous pouvons donc cliquer sur ce lien pour supprimer notre produit " vitamine C " et en revenant à notre listing on se rend bien compte comme le montre la capture ci-dessous que ce produit n'existe plus dans notre liste.

Libelle	Fabricant	Unité	Quantité en stock	Stock d'allerte	Opération
24DNPH	areva	ml	124588	1250	Supprimer
acide chlohydrique	chimie	ml	124588	0	Supprimer
alcool	novartis	cm	10	20	Supprimer
aspirine	chimie	ml	124588	0	Supprimer
benzene	chococam	cm	10	2	Supprimer
bromure	chimie	l	1254	0	Supprimer
phosphate	arcelor	cm	145	0	Supprimer
polonium 210	AIEA	ur	25	0	Supprimer
uranium 235	AIEA	bekerel	125	12	Supprimer
urée	engrais	sac	100	0	Supprimer

Fig. 4 suppression d'un produit

V - Conclusion

Nous venons à travers cet article de voir comment accéder à Mysql en orienté objet grâce à l'extension mysqli ; nous avons également pu effectuer des traitements grâce aux classes mysqli_stmt et mysqli_result et à leurs méthodes. Désormais vous pouvez développer votre application PHP5 en adoptant une approche complètement orienté objet, même au niveau de l'accès aux données. Cependant pour ceux qui ne sont pas encore familiarisés à l'approche orientée objet de PHP5 et veulent pouvoir tirer profit des avantages de l'extension mysqli, ils peuvent dans ce cas opter pour l'approche procédurale de cette API ; cella pourrait d'ailleurs faire l'objet d'un article futur. Etant donné que cet article ne saurait être exhaustif sur ce sujet, nous vous invitons à compléter vos connaissances en faisant un tour sur Internet et en lisant quelques livres spécialisés.

VI - Remerciements