PHP

PHP DATA OBJECT LAYER (*PDO*)

PDO ?

- **PHP Data Object Layer**
- **Interface commune d'accès à plusieurs SGBD**
- **Ecrite en C**
  performances
- **Utilisation de mécanismes PHP >5.1 évolués**
  objets et classes, reflexion, interfaces, exceptions

UTILITE

- **Nombreuses bibliothèques d'accès aux données**
  interfaces proches mais non identiques

- **Code souvent historique voire archaïque**
  possibilités modernes offectes par PHP non exploitées

- **Sous exploitation des capacités du SGBD**
  ex : extensions MySQL

SGBD COUVERTS

- MySQL 3, 4, 5
- PostgreSQL
- SQLite 2, 3
- ODBC
- DB2 (IBM)
- Oracle
- FireBird
- Sybase, MSSQL Server

INSTALLATION

- **Deux composants**
  - noyau (core) : prise en charge de l'interface
  - drivers : prise en charge d'une base de données particuliere
  ex : pdo_mysql

- **Noyau intégré au système par défaut**

- **Drivers non activés par défaut (sauf *pdo_sqlite*)**
  à activer (php.ini)

UTILISATION

- **Connexion à la base**
  fonction du SGBD (connectionString)
  utilisateur (login / mot de passe)

- **Requêtes SQL**
  - sans résultat *(INSERT, UPDATE, DELETE)*
  - avec résultat(s) *(SELECT)*
  tuples

- **Exploitation des *tuples***
  boucle

CONNEXION

- **Création d'un objet PDO**
  __construct (string $dsn, [string $user, string $pwd, [array $options]])

- **Date Source Name (DSN)**
  syntaxe propre au driver de BD

```
<?php

 //  Connexion MySQL
$db =  new PDO('mysql:host=localhost; dbname=maBase', $login, $mdp);

// Connexion PostgreSQL
$db = new PDO('pgsql:host=localhost port=5432 dbname=maBase user=toto password=monMdp');

// Connexion SQLite
$db = new PDO('sqlite:/chemin/vers/fichier_bd');
?>
```

ERREUR SUR CONNEXION

- **Levée d'une exception**

```
<?php
        try {
                $db = new PDO ( … );
        }
         catch (PDOException $e) {
                echo $e->getMessage ();
        }

?>
```

CONNEXION PERSISTANTE

- **Réutilisation d'une connexion précédemment établie**
  gain en performance (connexion lente – cf Oracle)

- **Paramètre à la création**
  option **PDO::ATTR_PERSISTENT** *à* **TRUE**

```
<?php
        $options= array(PDO::ATTR_PERSISTENT => TRUE) ;
        try {
                $db = new PDO("dsn", $login, $mdp, $options);
        }
        catch (PDOException $e) {
                echo $e->getMessage();
        }
?>
```

DECONNEXION

- **À la destruction de l'objet**
  fin du script

- **À l'affectation de null**
  $db = null;

- **Exception : connexion persistante**
  mise en cache de l'objet

EXECUTION DE REQUETES

- **Exécution directe**
  constitution de la requête (concaténation de chaines et variables)
  →erreurs, failles de sécurité(injection SQL)

- **Requête préparées**
  →sécurité, performance

- **Procédures stockées**
  enregistrement de la requête dans la base
  PL-SQL (Oracle)
  Transact-SQL (MS SQL)
  →partage, sécurité, performance

EXEC ()

- **Requêtes modifiant la base de données**
  INSERT, UPDATE, DELETE

- **Méthode exec()**
  paramètre : *la requête SQL*
  retour : *nombre de lignes affectées*
  FALSE sur erreur

```
<?php
        $db = new PDO("mysql:host=localhost;  dbname=maBase",  $login,  $mdp);

        $db->exec ("UPDATE Personne SET mail='mmaldo@gmail.com'  WHERE ID=12345");
        $db->exec ("INSERT INTO Personne SET mail='mmaldo@gmail.com'  WHERE ID=12345");

?>
```

RAPPELS SQL

- **Requête insertion**

INSERT INTO table
(champ$_1$, champ$_2$, … champ$_n$)
VALUES
(val$_1$, val$_2$, …,val$_n$)

- **Requête mise à jour**

UPDATE table
SET champ$_1$ = val$_1$,
champ$_2$ = val$_2$, …
champ$_n$ = val$_n$
WHERE …

- **Requête suppression**

DELETE table
WHERE …

CODE DE RETOUR

- **Exécution normale**

  nombre de lignes

  

  peut valoir 0 !

- **Cas d'erreur**

  FALSE

- **En conséquence…**

```
<?php
        $res = $db->exec("UPDATE  Personne SET mail='mmaldo@gmail.com' where … ");
        if ($res == 0)                          // Faux !
                echo "erreur requete !";

        if ($res === FALSE)                     // Correct …
                echo "erreur requete !";
?>
```

GESTION D'ERREUR – errorCode()

- **retourne un état SQL (SQLSTATE)**
  - *00000 : succès*
  - *42000 : erreur de syntaxe SQL*

liste des codes d'erreur : http://docstore.mik.ua/orelly/java-ent/jenut/ch08_06.htm

```
<?php
        // erreur : le champ maile  n'existe pas !
        $res = $db->exec("UPDATE  Personne SET maile='mmaldo@gmail.com' where … ");
        if ($res === FALSE) {
                echo "code de retour SQLSTATE : ";
                echo $db->errorCode ();
        }
?>
```

code de retour SQLSTATE : 42000

GESTION D'ERREUR – errorInfo()

- **retourne un tableau d'information sur l'erreur**
  - 0 : code d'erreur SQLSTATE
  - 1 : code d'erreur driver
  - 2 : message d'erreur

```
<?php
        // erreur : le champ maile  n'existe pas !
        $res = $db->exec("UPDATE  Personne SET maile='mmaldo@gmai
        if ($res === FALSE) {
                echo "code de retour errorInfo : ";
                print_r ($db->errorInfo ());
        }
?>
```

```
Array (
[0] => 42000
[1] => 1064
[2] => You have an error in your
SQL syntax; check the manual … )
```

GESTION D'ERREUR - Exceptions

- **Bonne manière de procéder**
- **Paramétrage pour des levées d'exceptions**

  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
- **Toute erreur maintenant lèvera (*throw*) une exception**

```
<?php
        $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXC
        // erreur : le champ maile  n'existe pas !
        try {
                $res = $db->exec("UPDATE  Personne SET maile='mmaldo@gmail.com' where … ");
                …
        } catch (PDOException $e) {
                echo $e->getMessage ();
        }
?>
```

SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual …

QUERY ()

- **Requêtes renvoyant des lignes (tuples)**
  SELECT

- **Méthode query()**
  paramètre : *la requête SQL*
  retour : *objet de type PDOStatement*
  FALSE sur erreur

```
<?php
        $db = new PDO("mysql:host=localhost;  dbname=maBase",  $login,  $mdp);

        $res = $db->query("SELECT * FROM Personne WHERE Cp like '33%' ");

?>
```

RAPPELS SQL

- **Jointures**
  liaisons entre tables (N tables, N-1 jointures)
  jointures internes (inner join),
  jointures gauches (left join), droites (right join), pleines (outer join)

- **Alias**
  raccourcis de table *(champs qualifiés, obligatoire si ambiguïtés)*
  raccourcis de champ *(facultatifs)*

- **Fonctions d'agrégat**
  SUM, COUNT, MIN, MAX, AVG
  group by <les champs du select>
  having

- **Tri**
  order by champ1 [ASC | DESC] [, champ1 [ASC | DESC] …]

fetch()

- **Récupération des tuples**

- **En fonction de la structure de données d'accueil**
  - *tableau*
  - *tableau associatif*
  - *chaîne*
  - *objet standard*
  - *objet specifique*
  - *fonction de rappel (callback)*
  - *iterator*
  - *…*

fetch() - *tableau*

```
<?php
 $res = $db->query("SELECT * FROM Personne");
while ($ligne = $res->fetch(PDO::FETCH_NUM)){
                // $ligne  est un tableau indexé par des entiers
}

$res = $db->query("SELECT * FROM Personne");
while ($ligne = $res->fetch(PDO::FETCH_ASSOC)){
                // $ligne  est un tableau associatif don't les clés sont les noms de champs
}

$res = $db->query("SELECT * FROM Personne");
while ($ligne = $res->fetch(PDO::FETCH_BOTH)){
                // $ligne  est un tableau à la fois indexé par des entiers et associatif
}

?>
```

fetchColumn() - *chaîne*

- **Cas de requête renvoyant une seule colonne**

```
<?php
// premier exemple
 $res = $db->query("SELECT nom FROM Personne");
while ($nom = $res-> fetch(PDO::FETCH_COLUMN)){
                // $nom est une chaine de caractères
}

// deuxième exemple
$res = $db->query("SELECT nom FROM Personne WHERE ID = 12345");
If ($nom = $res->fetchColumn() )
                echo Bonjour $nom !

?>
```

fetch() – *objet standard*

- **Création d'un objet de la class stdClass**
  champ de la requête = propriété (attribut public) de la classe

```
<?php

 $res = $db->query("SELECT nom, prenom, mail FROM Personne WHERE ….");
while ($obj= $res-> fetch(PDO::FETCH_OBJ)){
                // $nom est un objet de la classe standard stdClass
                echo "Bonjour " . $obj->nom  . "  " . $obj->prenom . "<br/>";
}
```

fetch() – *objet spécifique*

- **Création d'un objet dont la classe est précisée**
  require_once ou autoload

- **Appel du constructeur de la classe**
  après assignation des champs par PDO (défaut)
  avant : PDO::FETCH_CLASS|PDO::FETCH_PROPS_LATE

```
<?php
require_once (« /classes/Personne.class.php »);

$res = $db->query("SELECT nom, prenom,mail FROM  Personne");
$res->setFetchMode(PDO::FETCH_CLASS,"Personne");

while ($unePersonne = $res->fetch()) {
        // $ unePersonne  est une instance  de Personne
}
?>
```

Fetch() – *iterator*

- **PDOStatement implémente l'interface iterator**

```
<?php

 $res = $db->query("SELECT nom, prenom, mail FROM Personne WHERE ….", PDO::FETCH_ASSOC);
foreach ($res as $ligne) {
                // $nom est un tableau associatif constitué des de la classe standard stdClass
                echo "Bonjour " . $ligne["nom"]  . "  " . $ligne["prenom"]. "<br/>";
}
```

fetchAll()

- **Récupération de tous des tuples d'un coup**
  tableau de tableaux associatifs (PDO::FETCH_ASSOC)
  ou d'objets (PDO::FETCH_CLASS)

coût en mémoire si de très nombreux résultats

- **Interêt**
  travail hors connexion

```
<?php
…
 $res = $db->query("SELECT nom, prenom, mail FROM Personne WHERE
$lesPersonne s= $res->fetchAll (PDO::FETCH_ASSOC );

print_r ($lesPersonnes);
?>
```

```
Array (
[0] => Array (
            [nom] => zetofrais
            [prenom] => melanie
            [mail] => mz@gmail.com
    )
[1] => Array (
            [nom] => enfaillite
            [prenom] => melusine
            [mail] => mf@wanadoo.fr
    )
)
```

fetch() – *fonction de rappel*

- **Applique la fonction précisée à chaque résultat**

```
<?php
function afficherPersonne($nom, $prenom, $mail)
{
        ...
}

$res = $db->query("SELECT nom, prenom, mail FROM Personne WHERE ...");

$res->fetchAll(PDO::FETCH_FUNC, "afficherPersonne");

?>
```

PROBLEMES

- **Performances**
  ré interprétation des requêtes à chaque exécution

- **Sécurité**
  données utilisateurs intégrées aux requêtes
  présence de caractères spéciaux (non quotés)
  → injection SQL

SOLUTIONS

- **Quote ()**
  quotation (*échappement) des caractères spéciaux*

- **Requêtes préparées**
  (*cf chapitre suivant)*

```
<?php
        $requete = " SELECT * FROM Personne WHERE login =" . $db->quote($_POST['login'])
                ."AND passwd=" . $db->quote($_POST['pass']);
        $res = $db->query ($requete);
?>
```

CONCLUSION