PDO : PHP Data Object

Tous ne sont que des extensions...

- Les fonstions mysql_* : API mysql
- Les fonctions mysqli_* aussi
- Il en est de même pour PDO

	Extension MySQL	Extension mysqli	PDO (avec le pilote PDO MySQL Driver et MySQL Native Driver)
Version d'introduction en PHP	Avant 3.0	5.0	5.0
Inclus en PHP 5.x	Oui, mais désactivé	Oui	Oui
Statut de développement MySQL	Maintenance uniquement	Développement actif	Développement actif depuis PHP 5.3
Recommandée pour les nouveaux projets MySQL	Non	Oui	Oui
L'API supporte les jeux de caractères	Non	Oui	Oui
L'API supporte les commandes préparées	Non	Oui	Oui
L'API supporte les commandes préparées côté client	Non	Non	Oui
L'API supporte les procédures stockées	Non	Oui	Oui
L'API supporte les commandes multiples	Non	Oui	La plupart
L'API supporte toutes les fonctionnalités MySQL 4.1 et plus récent	Non	Oui	La plupart

Qu'est ce que PDO

- PDO signifie Php Data Object
- C' est une couche d'abstraction des fonctions d'accès aux bases de données
- Les requêtes ne seront pas automatiquement compatibles avec n'importe quelle base de données
- Seules les fonctions d'accès sont universelles :

  pour mysql:
  - mysql_connect()
  - mysql_query()
  - mysql_result()
  - mysql_fetch_array()
  - mysql_real_escape_string()
  - etc.

Pourquoi PDO plutôt qu'un autre ?

- D'autres alternatives existent : MDB2 de PEAR
- PDO : extension compilée de PHP donc performances supérieures
- Standard : http://www.phpteam.net/index.php/articles/les-nouveautes-de-php-6
- Depuis la version 5 de php, MySQL est désactivé

PDO : Encore en développement

- Certains pilotes de PDO sont encore en développement
- D'autres n'existent pas encore
- Liste des pilotes disponibles :

Nom du driver	Bases de données supportées
PDO_CUBRID	Cubrid
PDO_DBLIB	FreeTDS / Microsoft SQL Server / Sybase
PDO_FIREBIRD	Firebird/Interbase 6
PDO_IBM	IBM DB2
PDO_INFORMIX	IBM Informix Dynamic Server
PDO_MYSQL	MySQL 3.x/4.x/5.x
PDO_OCI	*Oracle Call Interface*
PDO_ODBC	ODBC v3 (IBM DB2, unixODBC et win32 ODBC)
PDO_PGSQL	PostgreSQL
PDO_SQLITE	SQLite 3 et SQLite 2
PDO_4D	4D

La classe PDO

- commit : Valide une transaction
- __construct : Crée une instance PDO qui représente une connexion à la base
- errorCode : Retourne le SQLSTATE associé avec la dernière opération sur la base de données
- errorInfo : Retourne les informations associées à l'erreur lors de la dernière opération sur la base de données
- exec : Exécute une requête SQL et retourne le nombre de lignes affectées
- getAttribute : Récupère un attribut d'une connexion à une base de données
- PDO::getAvailableDrivers : Retourne la liste des pilotes PDO disponibles
- inTransaction : Vérifie si nous sommes dans une transaction
- lastInsertId : Retourne l'identifiant de la dernière ligne insérée ou la valeur d'une séquence
- prepare : Prépare une requête à l'exécution et retourne un objet
- query : Exécute une requête SQL, retourne un jeu de résultats en tant qu'objet PDOStatement
- quote : Protège une chaîne pour l'utiliser dans une requête SQL PDO
- rollBack : Annule une transaction (même si le script php plante)
- setAttribute : Configure un attribut PDO
- beginTransaction : Démarre une transaction

Exemple d'instanciation

fichier my_setting.ini:

```
[database]
driver = mysql
host = localhost
;port = 3306
schema = db_schema
username = user
password = secret
```

Classe PHP:

```
<?php

class MyPDO extends PDO {

  public function __construct($file = 'my_setting.ini') {
    if (!$settings = parse_ini_file($file, TRUE))
      throw new exception('Impossible d'ouvrir ' . $file . '.');

    $dns = $settings['database']['driver'] .
            ':host=' . $settings['database']['host'] .
            ((!empty($settings['database']['port'])) ? (';port=' . $settings['database']['port']) :
'') .
            ';dbname=' . $settings['database']['schema'];

    parent::__construct($dns, $settings['database']['username'], $settings['database']['password']);
  }

}
?>
```

La classe PDOStatement

- bindColumn : Lie une colonne à une variable PHP
- bindParam : Lie un paramètre à un nom de variable spécifique
- bindValue : Associe une valeur à un paramètre
- closeCursor : Ferme le curseur, permettant à la requête d'être de nouveau exécutée
- columnCount : Retourne le nombre de colonnes dans le jeu de résultats
- debugDumpParams : Détaille une commande préparée SQL
- errorCode : Récupère le SQLSTATE associé lors de la dernière opération sur la requête
- errorInfo : Récupère les informations sur l'erreur associée lors de la dernière opération sur la requête
- execute : Exécute une requête préparée
- fetch : Récupère la ligne suivante d'un jeu de résultat PDO
- fetchAll : Retourne un tableau contenant toutes les lignes du jeu d'enregistrements
- fetchColumn : Retourne une colonne depuis la ligne suivante d'un jeu de résultats
- fetchObject : Récupère la prochaine ligne et la retourne en tant qu'objet
- getAttribute : Récupère un attribut de requête
- getColumnMeta : Retourne les métadonnées pour une colonne d'un jeu de résultats
- nextRowset : Avance à la prochaine ligne de résultats d'un gestionnaire de lignes de résultats multiples
- rowCount : Retourne le nombre de lignes affectées par le dernier appel à la fonction PDOStatement::execute()
- setAttribute : Définie un attribut de requête
- setFetchMode : Définit le mode de récupération par défaut pour cette requête

Fetch styles

- PDO::FETCH_BOTH par défaut
- PDO::FETCH_ASSOC: retourne un tableau indexé par le nom de la colonne comme retourné dans le jeu de résultats
- 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_BOUND: retourne TRUE et assigne les valeurs des colonnes de votre jeu de résultats dans les variables PHP à laquelle elles sont liées avec la méthode PDOStatement::bindParam()
- PDO::FETCH_CLASS: retourne une nouvelle instance de la classe demandée, liant les colonnes du jeu de résultats aux noms des propriétés de la classe. Si fetch_style inclut PDO::FETCH_CLASS (c'est-à-dire PDO::FETCH_CLASS | PDO::FETCH_CLASSTYPE), alors le nom de la classe est déterminé à partir d'une valeur de la première colonne
- PDO::FETCH_INTO: met à jour une instance existante de la classe demandée, liant les colonnes du jeu de résultats aux noms des propriétés de la classe
- PDO::FETCH_LAZY: combine PDO::FETCH_BOTH et PDO::FETCH_OBJ, créant ainsi les noms des variables de l'objet, comme elles sont accédées
- PDO::FETCH_NUM: retourne un tableau indexé par le numéro de la colonne comme elle est retourné dans votre jeu de résultat, commençant à 0
- 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 classe PDOException

- PDOException::getMessage — Récupère le message de l'exception
- PDOException::getPrevious : Retourne l'exception précédente
- PDOException::getCode : Récupère le code de l'exception
- PDOException::getFile : Récupère le fichier dans lequel l'exception est survenue
- PDOException::getLine : Récupère la ligne dans laquelle l'exception est survenue
- PDOException::getTrace : Récupère la trace de la pile
- PDOException::getTraceAsString : Récupère la trace de la pile en tant que chaîne
- PDOException::__toString : Représente l'exception sous la forme d'une chaîne
- PDOException::__clone : Clone l'exception

Transactions avec PDO

```
<?php
try {
  $dbh = new PDO('odbc:SAMPLE', 'db2inst1', 'ibmdb2',
                 array(PDO::ATTR_PERSISTENT => true));
  echo "Connecté\n";
} catch (Exception $e) {
  die("Impossible de se connecter: " . $e->getMessage());
}

try {
  $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  $dbh->beginTransaction();
  $dbh->exec("insert into staff (id, first, last) values (23, 'Joe', 'Bloggs')");
  $dbh->exec("insert into salarychange (id, amount, changedate)
  values (23, 50000, NOW())");
  $dbh->commit();
} catch (Exception $e) {
  $dbh->rollBack();
  echo "Failed: " . $e->getMessage();
}
?>
```

Requêtes préparées

- Pourquoi les requêtes préparées ?

  Inconvénients:
  - plus de lignes de code que pour une requête simple
  - exécution unique : performances moindres par rapport à une exécution directe
  - résultat final identique : exécuter une requête
  - débogage de la requête légèrement plus complexe

  Avantages:
  - impose une certaine rigueur de programmation
  - optimisation du temps d'exécutions requis pour les requêtes exécutées plus d'une fois
  - plus grande sécurité au niveau des requêtes
  - séparation requête SQL des données

Requêtes préparées : exemple

```
<body>
   <form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
     <fieldset>
       <legend>Recherche de client</legend>
       <label>Ville </label><input type="text" name="ville" /><br /><br />
       <label>Id_client</label><input type="text" name="id_client" /><br />
       <input type="submit" value="Envoyer" />
     </fieldset>
   </form>

   <?php
   if (isset($_POST[‘ville’]) && isset($_POST[‘id_client’])) {
     $ville = strtolower($_POST[‘ville’]);
     $id_client = $_POST[‘id_client’];
     include(‘connexpdo.inc.php’);
     $idcom = connexpdo('magasin', 'myparam');
     //$reqprep = $idcom->prepare("SELECT prenom,nom FROM client WHERE lower(ville)=:ville AND id_client>=:id_client");
     $reqprep = $idcom->prepare("SELECT prenom,nom FROM client WHERE lower(ville)=:ville AND id_client>=:id_client");
     //*****Liaison des paramètres

     $reqprep->bindParam(':ville',$ville,PDO::PARAM_STR);
     $reqprep->bindParam(':id_client',$id_client,PDO::PARAM_INT);
     $reqprep->bindParam(3,$id_client,PDO::PARAM_INT);
     $reqprep->execute();

     //*****Liaison des résultats à des variables
     $reqprep->bindColumn('prenom', $prenom);
     $reqprep->bindColumn('nom', $nom);

     //*****Affichage
     echo "<div><h3>Il y a ", $reqprep->rowCount(), " client(s) habitant à ", ucfirst($ville), " et dont l'identifiant est
supérieur à $id_client</h3><hr />";
     while ($result = $reqprep->fetch(PDO::FETCH_BOUND)) {
       echo "<h3> $prenom $nom</h3>";
     }
     echo "</div>";
     $reqprep->closeCursor();
     $idcom = null;
   }
   ?>
```