PWS - PDO

Manipulation de données stockées: PDO

Pierre Blarre - Florian Rodriguez

Introduction

- Tout site ou application dynamique doit pouvoir stocker et interagir avec des données.
- L'extension PHP Data Objects (PDO) définit une interface pour accéder à une base de données depuis PHP.
- Elle permet de se connecter et d’interagir avec différents types de SGBD (MySQL, Oracle, PostGreSQL, etc.) avec une syntaxe PHP similaire.
- Elle permet de sécuriser les requêtes SQL en utilisant des requêtes préparées. (prévention des injections SQL)

Connexion à une base de données

- Exemple de connexion

```php
<?php
try {
    $dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
} catch (PDOException $e) {
    print "Erreur !: " . $e->getMessage() . "<br/>";
    die();
}
```

- Exemple avec sqlite

```php
<?php
try {
 $dbh = new PDO('sqlite:mydatabase.db');
} catch (PDOException $e) {
 print "Erreur !: " . $e->getMessage() . "<br/>";
 die();
}
```

Récupération de donnée

```php
$sql = "SELECT * FROM utilisateurs";
$stmt = $pdo->query($sql);

if ($stmt->rowCount() > 0) {
    while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
        echo 'ID : ' . $row['id'] . ' - Nom : ' . $row['nom'] . ' - Prénom : ' . $row['prenom'] . ' - Email : ' . $row[
    }
} else {
    echo 'Aucun utilisateur trouvé.';
}
```

- La méthode **query()** permet d'exécuter une requête SQL et de retourner un objet PDOStatement.
- Un objet PDOStatement est un objet qui contient le résultat de la requête.
- La méthode **rowCount()** permet de récupérer le nombre de lignes retournées par la requête.
- La méthode **fetch()** retourne un tableau associatif contenant les données de la ligne courante et avance le pointeur sur la ligne suivante.
    - `PDO::FETCH_ASSOC` : retourne un tableau associatif indexé par le nom de la colonne.
    - `PDO::FETCH_NUM` : retourne un tableau indexé par le numéro de la colonne.
    - `PDO::FETCH_BOTH` (mode par défaut): retourne un tableau associatif indexé par le nom de la colonne et par le numéro de la colonne.
    - `PDO::FETCH_OBJ` : retourne un objet anonyme avec des propriétés qui correspondent aux colonnes.
    - voir la documentation (fr) (en) pour plus de détails et d'autres options

Exemple

```php
<?php
$db = new PDO('mysql:host=localhost;dbname=php', 'php', 'php');

$request = $db->query('SELECT id, nom, forcePerso, degats, niveau, experience FROM personnages');
while ($perso = $request->fetch(PDO::FETCH_ASSOC)) // Chaque entrée sera placée dans un array.
{
    echo '</br>'.$perso['nom'], ' a ', $perso['forcePerso'], ' de force, ', $perso['degats'], ' de dégâts, ',
        $perso['experience'], ' d\'expérience et est au niveau ', $perso['niveau'];
}
```

–> Lien vers la page de test:
http://localhost:2023/exemple-pdo-select.php

Requêtes SQL avec exec

```php
<?php
$sql = "INSERT INTO utilisateurs (nom, prenom, email) VALUES ('Doe', 'John', 'john.doe@uga.fr')";
if ($pdo->exec($sql) !== false) {
    echo 'Utilisateur ajouté.';
} else {
    echo 'Erreur lors de l\'ajout de l\'utilisateur.';
}
```

- La méthode **exec()** permet d'exécuter une requête SQL et de retourner le nombre de lignes affectées.
- Elle est utilisée pour les requêtes qui ne retournent pas de données (INSERT, UPDATE, DELETE, etc.).
- **exec()** et **query()** sont deux méthodes de la classe PDO qui permettent d'exécuter des requêtes SQL sans passer par une requête préparée.
- ***Attention*** *: les requêtes construites avec* ***exec()*** *et* ***query()*** *ne sont pas sécurisées contre les injections SQL.*
  => À éviter.

Exemple

```php
<?php
$db = new PDO('mysql:host=localhost;dbname=php', 'php', 'php');
  $sql = "INSERT INTO personnages (nom, forcePerso, degats, niveau, experience) VALUES ('Zorglub', 80, 10, 1, 0)";
  if ($db->exec($sql) !== false) {
      echo 'Personnage ajouté.';
  } else {
      echo 'Erreur lors de l\'ajout du personnage.';
  }
```

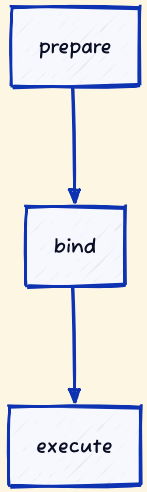
–> Lien vers la page de test:
http://localhost:2023/exemple-pdo-insert.php

Requêtes SQL avec prepare et execute

```php
<?php
$sql = "INSERT INTO utilisateurs (nom, prenom) VALUES (:nom, :prenom)";
$stmt = $db->prepare($sql);
$stmt->bindParam(':nom', $nom);
$stmt->bindParam(':prenom', $prenom);

$nom = 'Doe';
$prenom = 'John';
if ($stmt->execute() !== false) {
    echo 'Utilisateur ajouté.';
} else {
    echo 'Erreur lors de l\'ajout de l\'utilisateur.';
}

$nom = 'Smith';
$prenom = 'Jane';
$stmt->execute();
```



- La méthode **prepare()** permet de préparer une requête SQL et de retourner un objet PDOStatement.
- La méthode **bindParam()** permet de lier des valeurs à des paramètres de la requête.
- La méthode **execute()** permet d'exécuter la requête préparée.
- Les paramètres peuvent être représentés par **?** ou par des marqueurs nommés (précédés de deux points **:param**).
- Les requêtes préparées permettent de réduire les risques d'injections SQL.
- Lien vers la page de test: http://localhost:2023/exemple-pdo-prepare.php

Exemple d'une sélection SQL avec PDO:

```php
<?php

$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);

$stmt = $dbh->prepare("SELECT * FROM REGISTRY where name = ?");
if ($stmt->execute(array($_GET['name']))) {
    while ($row = $stmt->fetch()) {
        print_r($row);
    }
}
?>
```

Une entité, un objet

Rappels sur la structure d'une BDD

	id	nom	forcePerso	degats	niveau	experience
	16	Vyk12	5	55	4	20
	18	Tchaper	5	5	1	0
	19	Neo	5	0	1	20

```php
<?php
// On admet que $db est un objet PDO
$request = $db->query('SELECT id, nom, forcePerso, degats, niveau, experience FROM personnages');

while ($perso = $request->fetch(PDO::FETCH_ASSOC)) // Chaque entrée sera placée dans un array.
{
  echo $perso['nom'], ' a ', $perso['forcePerso'], ' de force, ', $perso['degats'], ' de dégâts, ', $perso['experience'
}
?>
```

Comment créer des classes

- Pour rappel, un *getter* est une méthode chargée de **renvoyer** la valeur d'un attribut.
- Tandis qu'un *setter* est une méthode chargée **d'assigner** une valeur à un attribut en vérifiant son **intégrité** (si vous assignez la valeur sans aucun contrôle, vous perdez tout l'intérêt qu'apporte le principe d'encapsulation).
- Pour construire nos setters, il faut donc nous pencher sur les valeurs possibles de chaque attribut :
    - les valeurs possibles de l'**identifiant** sont tous les nombres entiers strictement positifs
    - les valeurs possibles pour le **nom du personnage** sont toutes les chaînes de caractères
    - les valeurs possibles pour la **force du personnage** sont tous les nombres entiers allant de 1 à 100
    - les valeurs possibles pour les **dégâts du personnage** sont tous les nombres entiers allant de 0 à 100
    - les valeurs possibles pour le **niveau du personnage** sont tous les nombres entiers allant de 1 à 100
    - les valeurs possibles pour l'**expérience du personnage** sont tous les nombres entiers allant de 1 à 100.

Une entité, un objet

Reprenons le code pour récupérer les données de la BDD et modifions le pour manipuler des objets

```php
<?php
// On admet que $db est un objet PDO
$request = $db->query('SELECT id, nom, forcePerso, degats, niveau, experience FROM personnages');

while ($perso = $request->fetch(PDO::FETCH_ASSOC)) // Chaque entrée sera placée dans un array.
{
    // On passe les données (stockées dans un tableau) du personnage au constructeur de la classe.
    // On admet que le constructeur de la classe appelle chaque setter pour assigner les valeurs qu'on lui a données au
    $perso = new Personnage($perso);
    echo $perso->nom(), ' a ', $perso->forcePerso(), ' de force, ', $perso->degats(), ' de dégâts, ', $perso->experienc
}
?>
```

Avec PDO::FETCH_CLASS

```php
<?php
class Personnage
{
  private $id;
  private $nom;
  private $forcePerso;
  private $degats;
  private $niveau;
  private $experience;
}

$request = $db->query('SELECT id, nom, forcePerso, degats, niveau, experience FROM personnages');
$request->setFetchMode(PDO::FETCH_CLASS, 'Personnage');

while ($perso = $request->fetch()) {
        echo $perso->nom, ' a ', $perso->forcePerso, ' de force, ', $perso->degats, ' de dégâts, ', $perso->experience,
}
?>
```

Attaque par injection SQL

- L'injection SQL est une technique qui permet d'injecter du code SQL dans une requête SQL via les données fournies par l'utilisateur.
- Exemple d'injection SQL

```
$sql = "SELECT * FROM utilisateurs WHERE nom = '" . $_GET['name'] . "'";
$stmt = $pdo->query($sql);
```

  - Si l'utilisateur entre le nom **= ' OR 1=1; --**, la requête deviendra : **SELECT * FROM utilisateurs WHERE name = '' OR 1=1; –'**
  - La condition 1=1 est toujours vraie, donc la requête retournera tous les utilisateurs de la table.
  - Le double tiret – permet de commenter le reste de la requête et d'ignorer le reste de la requête.

–> Lien vers la page de test:
http://localhost:2023/exemple-injection-sql.php?name=%3D+%27+OR+1%3D1%3B+--

Autre exemple d'injection SQL

```php
$pdo = new PDO('mysql:host=localhost;dbname=test') ;
$sql = "SELECT * FROM utilisateurs WHERE nom='$nom' AND passwd='$passwd'";

$stmt = $pdo->query($sql);

  if ($stmt->rowCount() > 0) {
      echo 'Bienvenue ' . $nom . ' :)';
      echo 'Vos informations sensible' . $prenom . ' - ' . $email . ' - ' . $passwd;
  } else {
      echo 'Login ou mot de passe incorrect.';
  }
```

- Si l'utilisateur entre pour **login: admin et pour passwd: 1234' OR '1'='1**
- La requête deviendra:
  `SELECT * FROM utilisateurs WHERE nom='admin' AND passwd='1234' OR '1'='1'`
- La condition '1'='1' est toujours vraie, donc la requête retournera l'utilisateur admin.
- *Bienvenue admin :)*
- Lien vers la page de test: http://localhost:2023/exemple-injection-sql2.php