PHP

PDO Avancé

Dans ce module

- PDO Details
- PDO Persistance des connections
- PDO Sequential (Fetch)
- PDOStatements Styles
- PDO Transactions
- PDO Prepare
- Pratique

PDO Details

- PDO inclus d'autres fonctions utiles comme "lastInsertId()" qui permet d'obtenir le dernier id généré. Si vous réalisez plusieurs inserts en une seule requête, il faut bien comprendre qu'il n'est possible que de récupérer le dernier id. Il faudra mettre en place une autre mécanique.

```php
$result = $db->exec("INSERT INTO users (email) VALUES ('t1@gmail.com'), ('t2@gmail.com')");

if ($result === false) {
  echo $db->errorInfo()[2];
  exit();
}

$lastId = $db->lastInsertId(); // ID de t2@gmail.com
```

PDO Options

- PDO inclu des options définissables lors de l'ouverture de la connexion, soit lors de l'instanciation de la classe de PDO.
- L'option `PDO::ATTR_PERSISTENT` permet de maintenir les connexions PDO ouverte et de les partager entre les instances PHP.

```php
// Connexion à la BDD
$dsn = 'mysql:host=localhost;port=3306;dbname=my_site_web';
try {
    $db = new PDO($dsn, 'admin', 'password', [
          PDO::ATTR_PERSISTENT => true,
    ]);
} catch (PDOException $e) {
    echo $e->getMessage() . '<br>';
    exit();
}
```

PDO Sequential (Fetch one by one)

- Si la requête va retrouver un trop grand nombre de row, il convient de ne pas utiliser "fetchAll()" mais plutôt "fetch()" qui retournera un row à la fois.
- Dans ce cas, il faut utiliser une boucle pour utiliser "fetch()" (chaque appel de la fonction fetch() renvoie une ligne puis se préparera à renvoyer la suivante).

```php
$resultUsersQuery = $db->query('SELECT * FROM users');
if ($resultUsersQuery === false) {
  echo $db->errorInfo()[2];
  exit();
}

try {
    while ($row = $resultUsersQuery->fetch()) {
      echo $row['email'].'<br>';
    }
}
catch (PDOException $e) {
    echo $e->getMessage() . '<br>';
    exit();
}
```

user1@gmail.com
user2@gmail.com
user19@gmail.com

PDOStatement Styles

Fetch style	Résultat
PDO::FETCH_ASSOC	Array indexé par les noms de colonnes
PDO::FETCH_NUM	Array indexé par les numéros de colonnes (0, 1, 2, ...)
PDO::FETCH_BOTH	(Par défaut) PDO::FETCH_ASSOC + PDO::FETCH_NUM
PDO::FETCH_OBJ	Chaque row est objet anonyme avec des propriétés à partir des noms de colonnes

```php
$db->query('SELECT * FROM users')->fetchAll(PDO::FETCH_BOTH);
    // $db->query('SELECT * FROM users')->fetchAll(); // Equilavent à PDO::FETCH_BOTH
$db->query('SELECT * FROM users')->fetchAll(PDO::FETCH_ASSOC);
$db->query('SELECT * FROM users')->fetchAll(PDO::FETCH_NUM);
$db->query('SELECT * FROM users')->fetchAll(PDO::FETCH_OBJ);
```

PDO::FETCH_BOTH (Default)

```php
$users = $db->query('SELECT * FROM users')->fetchAll();
var_dump($users);
```

```
array (size=9)
  0 =>
    array (size=4)
      'id' => string '1' (length=1)
      0 => string '1' (length=1)
      'email' => string 'user1@gmail.com' (length=15)
      1 => string 'user1@gmail.com' (length=15)
  1 =>
    array (size=4)
      'id' => string '2' (length=1)
      0 => string '2' (length=1)
      'email' => string 'user2@gmail.com' (length=15)
      1 => string 'user2@gmail.com' (length=15)
  2 =>
    array (size=4)
      'id' => string '19' (length=2)
      0 => string '19' (length=2)
      'email' => string 'user19@gmail.com' (length=16)
      1 => string 'user19@gmail.com' (length=16)
```

PDO::FETCH_ASSOC

```php
$users = $db->query('SELECT * FROM users')->fetchAll(PDO::FETCH_ASSOC);
var_dump($users);
```

```
array (size=9)
  0 =>
    array (size=4)
      'id' => string '1' (length=1)
      'email' => string 'user1@gmail.com' (length=15)
  1 =>
    array (size=4)
      'id' => string '2' (length=1)
      'email' => string 'user2@gmail.com' (length=15)
  2 =>
    array (size=4)
      'id' => string '19' (length=2)
      'email' => string 'user19@gmail.com' (length=16)
```

PDO::FETCH_OBJ

```php
$users = $db->query('SELECT * FROM users')->fetchAll(PDO::FETCH_OBJ);
echo $users[0]->id . '<br>';
echo $users[0]->email . '<br>';
echo $users[1]->id . '<br>';
echo $users[1]->email . '<br>';
```

1
user1@gmail.com
2
user2@gmail.com

PDO Transaction

- Une transaction de base de donnée permet de garantir l'exécution de toutes les requêtes entre le début et la fin de la transaction. De plus, si une des requêtes ne peut pas être exécuter, cela annule (rollback) toutes les requêtes de la transaction.
- Une transaction avec PDO présente donc 4 fonctions **: Atomicité, Consistance, Isolation et Durabilité (ACID).** N'importe quel travail mené à bien dans une transaction, même s'il est effectué par étapes, est garanti d'être appliqué à la base de données sans risque, et sans interférence pour les autres connexions, quand il est validé.
- Une transaction PDO se déroule comme suit :
  1. **Démarrer la transaction**
  2. **Exécuter toutes les requêtes voulues**
  3. Terminer la transaction soit : **Commit** (enregistrer) ou **Rollback** (annuler)
- Une transaction est à utiliser uniquement s'il s'agit d'**au moins 2 requêtes devant être garantie ensemble (Tout ou Rien).**

→ ACID Wikipédia

PDO Transaction - Example

```php
$db->beginTransaction(); // Début de la transaction (START)

try {
  $db->exec("INSERT INTO users (email) VALUES ($email)");
  $userId = $db->lastInsertId();
  $db->exec("INSERT INTO orders (user_id, price) VALUES ($userId, $price)");

  $db->commit(); // Enregistrement (END)

} catch (Exception $e) {
  $db->rollBack(); // Annulation (END)
  echo $e->getMessage();
  exit();
}
```

PDO Prepare

1. Avant d'exécuter une requête la BDD va analyser, compiler et ensuite choisir le meilleur plan d'approche pour l'exécution. Sur des requêtes complexes, répétées dans une même instance PHP, ce processus répétitif deviendra lourd en charge de calcul. Il convient alors de créer une préparation unique de ce processus.
2. De plus, l'injection SQL est un vrai problème de sécurité et il convient alors d'utiliser une approche plus avancée pour pallier à ce risque. Ce risque existe parce qu'il est imparable d'insérer en BDD des données venant de formulaires utilisables par une personne dont non ne connaît pas les intentions, même au sein d'une entreprise.

➢ Pour répondre à ces 2 problématiques, PDO va nous permettre de **préparer les requêtes en amont et d'y insérer les paramètres de la requête**.

```php
// Préparation
$req = $db->prepare("INSERT INTO users (email, password) VALUES (:email, :password)");

// Ajouts des paramètres
$req->bindParam(':email', $_POST['email']);
$req->bindParam(':password', $_POST['password']);

// Exécution
$req->execute(); // true or false or PDOException
```

PDO Prepare - Example

```php
// $_POST = [
//     'items' => [48, 19, 568]
// ]

// Préparation
$req = $db->prepare("INSERT INTO users_items (user_id, item_id) VALUES (:user_id, :item_id)");
$req->bindParam(':user_id', $_SESSION['userId']);

foreach($_POST['items'] as $itemId) {
    $req->bindParam(':item_id', $itemId);
    $req->execute();
}
```

PDO Prepare – Style Example

```php
$email = $_POST['email'];
$password = $_POST['password'];
$users = $db->query("SELECT * FROM users WHERE email = '$email' AND pwd = '$password'")
            ->fetchAll(PDO::FETCH_ASSOC);
```

```php
$req = $db->prepare("SELECT * FROM users WHERE email = :email AND pwd = :pwd");
$req->bindParam(':email', $_POST['email']);
$req->bindParam(':pwd', $_POST['pwd']);
$req->execute();
$users = $req->fetchAll(PDO::FETCH_ASSOC);
```

```php
$req = $db->prepare("SELECT * FROM users WHERE email = ? AND pwd = ?");
$req->bindParam(1, $_POST['email']);
$req->bindParam(2, $_POST['pwd']);
$req->execute();
$users = $req->fetchAll(PDO::FETCH_ASSOC);
```

PDO Prepare – Style Example

```php
$req = $db->prepare("SELECT * FROM users WHERE email = :email AND pwd = :pwd");
$req->execute([ // Paramètres par array
    ':email' => $_POST['email'],
    ':pwd' => $_POST['password'],
]);
$users = $req->fetchAll(PDO::FETCH_ASSOC);
```

```php
$req = $db->prepare("SELECT * FROM users WHERE email = ? AND pwd = ?");
$req->execute([ // Paramètres par array
    $_POST['email'],
    $_POST['password'],
]);
$users = $req->fetchAll(PDO::FETCH_ASSOC);
```

Pratique

- Reprenez le TP de la première partie du cours de PDO et modifiez toutes vos requêtes en requêtes préparées, car actuellement votre application est sensible au Injection SQL !

- Ensuite, vous devrez permettre à vos utilisateurs connectés d'écrire des notes/mémo visibles uniquement par eux même.
  - Chaque note est composée d'un titre et d'un texte séparé.
  - Créer une nouvelle table BDD pour accueillir les notes. Vous devrez choisir le nom de la table et des champs.
  - La liste des notes peut être affichée sur la page home de leur créateur.
  - Sur cette même page, vous pouvez y mettre le formulaire HTML pour ajouter une note.
  - N'oubliez pas de créer un fichier PHP (script) séparé pour traiter l'ajout d'une note.

- Pour finir, ajoutez une pagination à l'affichage des notes d'un utilisateur. La pagination sera de 5 notes par page. Vous allez devoir adapter votre requête SQL de récupération des notes en utilisant les mots clés « LIMIT » et « OFFSET ».

PHP

PDO Avancé

Fin du module