Humanbooster

PHP

BASE DE DONNÉES / PDO

Aurélien Delorme
OCT 2022

- Base de données / Type de bases de données
- Objectifs PDO
- PDO
- Connexion à une BDD
- Afficher des données
- Ajouter des données de manière sécurisé
- Editer des données de manière sécurisé
- Supprimer

Rappels BDD

Rappel : Une BDD permet de stocker des données de manière organisé

Il existe plusieurs types de BDD. Nous étudierons les bases de données relationnelles

Exemple Base de données Relationnelles

Microsoft SQL Server

MySQL

PostgreSQL

Parmi tant d'autres

D'autres exemples

mongoDB

BDD orienté document ne nécessite pas de schéma défini

OrientDB

BDD orienté graphe
(https://fr.wikipedia.org/wiki/Base_de_données_orientée_graphe)

Parmi tant d'autres

Chaque type de base de donnée répond à un besoin précis. Etudiez le besoin pour faire le bon choix !

SGBD : Système de gestion de base de donnée

Utilisateur

Envoi une requête sur notre serveur

Serveur PHP

Envoie des requêtes
Reçoit des réponses

Demande une page
Saisie des données

Se connecte au serveur MySql
Lui envoie des requêtes
Reçoit les réponses. Traite et/ou renvoie les données demandées

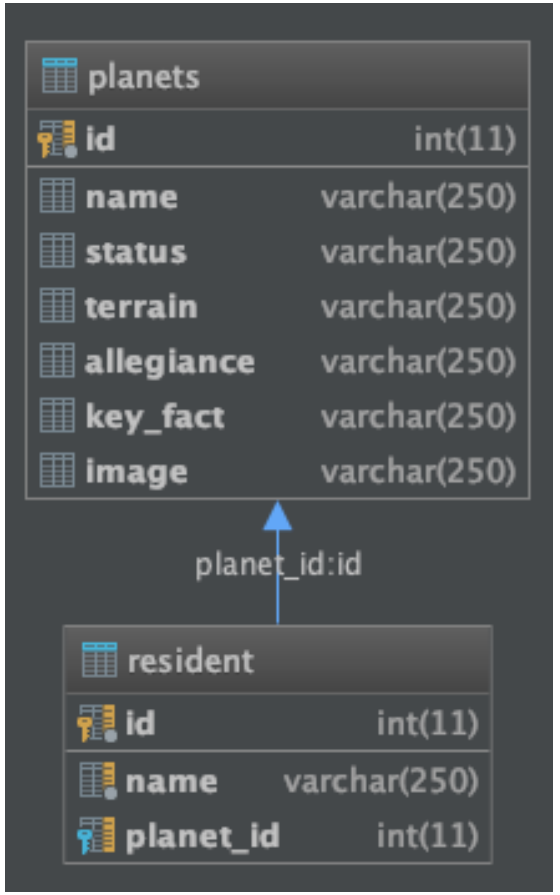
Composition d'un schéma relationnel (RAPPEL BDD)

- Une ou plusieurs table

- Qui comprendront chacune une ou plusieurs colonnes

- Qui comprendront chacune 0 ou plusieurs contraintes (nullable, not null, unique, auto-increment)

- Nos tables seront en relation via l'utilisation de clé primaire et de clé étrangère



Pourquoi utiliser PDO

- Standardise la méthodologie de connexion aux systèmes de gestion de bases de données.

- Méthode la plus largement utilisée car elle simplifie les migrations de BDD avec PHP.

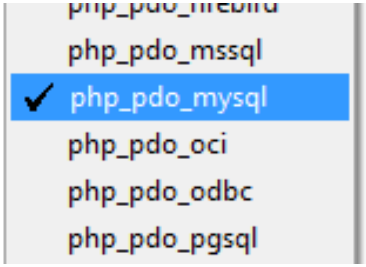
- Avant on utilisait directement les méthodes mysql de PHP mais ces dernières sont dépréciées.

- Il reste la possibilité d'utiliser les fonctions mysqli_ . C'est dernières seront en revanche moins flexibles.

- Extension orienté objet

Configurer notre environnement pour utiliser PDO

Normalement activé par défaut



L'extension suivante de votre php.ini ne dois pas être commenté (pas de « ; »). Par défaut aussi

extension=php_pdo_mysql.dll

www > phpinfo.php >

```php
<?php
phpinfo();
```

PDO

PDO support	enabled
PDO drivers	sqlite, mysql

pdo_mysql

PDO Driver for MySQL	enabled
Client API version	mysqlnd 5.0.12-dev - 20150407 - $Id: 38fea24f2847fa7519001be390c98ae0acafe387 $

Directive	Local Value	Master Value
pdo_mysql.default_socket	*no value*	*no value*

pdo_sqlite

PDO Driver for SQLite 3.x	enabled
SQLite Library	3.15.1

Création d'un nouvel objet PDO

```php
<?php
$bdd = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'user', 'password');
?>
```

mysql:host: -Le nom de domaine ou l'adresse IP de votre base de donnée

- dbname: le nom de la base de donnée que vous souhaitez utiliser

- charset: Le type d'encodage de votre base de donnée

- 'user': le nom de l'utilisateur mysql que vous souhaitez utiliser

- 'password': Le mot de passe de l'utilisateur à utiliser

Astuce 1 : Utilisation d'un try/catch pour pour afficher les erreurs

```php
try{
    $host = 'localhost';
    $dbName = 'business-case';
    $user = 'root';
    $password = '';
    $pdo = new PDO(
        'mysql:host='.$host.';dbname='.$dbName.';charset=utf8',
        $user,
        $password);
    // Cette ligne demandera à pdo de renvoyer les erreurs SQL si il y en a
    $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
}
catch (PDOException $e) {
    throw new InvalidArgumentException('Erreur connexion à la base de
données : '.$e->getMessage());
    exit;
}
```

Le message je suis connecté apparaitra seulement lorsque ma connexion avec la base de donnée fonctionnera.

Sélectionner et afficher des donnée

1) Appel de la fonction **query** de notre objet PDO

```
$reponse = $pdo->query('SELECT * FROM restaurants');
```

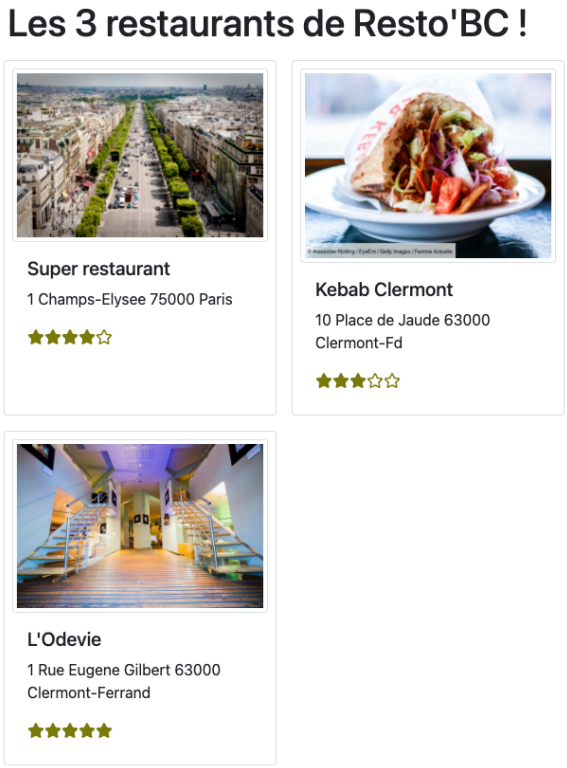
La variable $response contiendra un nouvel objet de type PDOStatement qui contiendra nos résultats

```
var_dump($reponse->fetchAll());
```

La méthode fetchAll nous permettra de récupérer tous nos enregistrements dans un tableau numéroté

```
var_dump($reponse->fetch());
```

La méthode fetch nous permettra de récupérer les enregistrements un a un



La méthode fetch()/fetchall()

La méthode fetch()

```
array (size=20)
  'id' => string '1' (length=1)
  0 => string '1' (length=1)
  'nom' => string 'Super restaurant' (length=16)
  1 => string 'Super restaurant' (length=16)
  'num_rue' => string '1' (length=1)
  2 => string '1' (length=1)
  'nom_rue' => string 'Champs-Elysee' (length=13)
  3 => string 'Champs-Elysee' (length=13)
  'ville' => string 'Paris' (length=5)
  4 => string 'Paris' (length=5)
  'code_postal' => string '75000' (length=5)
  5 => string '75000' (length=5)
  'type' => string 'Gastronomique' (length=13)
  6 => string 'Gastronomique' (length=13)
  'image_link' => string 'images/champ-elysee.jpeg' (length=24)
  7 => string 'images/champ-elysee.jpeg' (length=24)
  'star' => string '4' (length=1)
  8 => string '4' (length=1)
  'slug' => string 'gastro' (length=6)
  9 => string 'gastro' (length=6)
```

Récupération des données une à une. A chaque appel de la fonction fetch(), le prochain résultat est récupéré

```php
while ($bar = $query->fetch()) {
    $adresse = $bar['num_rue'].' '.$bar['nom_rue'].' '.$bar['code_postal'].' '.$bar['ville'];
    $string = '';
    for($i=1;$i<=5; $i++){
        if($bar['star']>=$i){
            $string .= '<i class="fas fa-star yellow"></i>';
        } else {
            $string .= '<i class="far fa-star yellow"></i>';
        }
    }
    echo('  <div class="card m-2 p-2 " style="width: 18rem;">
    <img src="'.$bar['image_link'].'" class="card-img-top img-thumbnail" alt="...">
        <div class="card-body">
            <h5 class="card-title">'.$bar['nom'].'</h5>
            <p class="card-text">'.$adresse.'</p>
            '.$string.'
        </div>
    </div>');
}
?>
```

La méthode fetchAll()

Tous les résultats sont retournés

```
array (size=3)
  0 =>
    array (size=20)
      'id' => string '1' (length=1)
      0 => string '1' (length=1)
      'nom' => string 'Super restaurant' (length=16)
      1 => string 'Super restaurant' (length=16)
      'num_rue' => string '1' (length=1)
      2 => string '1' (length=1)
      'nom_rue' => string 'Champs-Elysee' (length=13)
      3 => string 'Champs-Elysee' (length=13)
      'ville' => string 'Paris' (length=5)
      4 => string 'Paris' (length=5)
      'code_postal' => string '75000' (length=5)
      5 => string '75000' (length=5)
      'type' => string 'Gastronomique' (length=13)
      6 => string 'Gastronomique' (length=13)
      'image_link' => string 'images/champ-elysee.jpeg' (length=24)
      7 => string 'images/champ-elysee.jpeg' (length=24)
      'star' => string '4' (length=1)
      8 => string '4' (length=1)
      'slug' => string 'gastro' (length=6)
      9 => string 'gastro' (length=6)
  1 =>
    array (size=20)
      'id' => string '2' (length=1)
      0 => string '2' (length=1)
      'nom' => string 'Kebab Clermont' (length=14)
      1 => string 'Kebab Clermont' (length=14)
      'num_rue' => string '10' (length=2)
      2 => string '10' (length=2)
      'nom_rue' => string 'Place de Jaude' (length=14)
      3 => string 'Place de Jaude' (length=14)
      'ville' => string 'Clermont-Fd' (length=11)
      4 => string 'Clermont-Fd' (length=11)
      'code_postal' => string '63000' (length=5)
      5 => string '63000' (length=5)
      'type' => string 'Fast Food' (length=9)
      6 => string 'Fast Food' (length=9)
```

Utilisation de la clause SQL Where

Création d'une requête préparée

Permet de sécuriser les appels de notre base de donnée afin d'éviter notamment les injections SQL

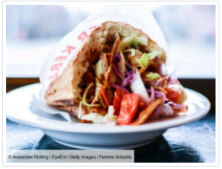
```php
$query = $connexion->prepare( query: 'SELECT * FROM restaurants WHERE slug = :slug');
$query->execute(['slug' => $type]);
```

1) Utilisation de la fonction PDO prepare
2) Nous envoyons dans un tableau les paramètres a remplacer dans notre requête. Dans notre cas, PHP aura besoin de l'url de la ressource. Il est possible de multiplier les paramètres.
3) Il est aussi possible d'utiliser toutes les autres clause du langage MySQL (order by, limit, jointures …).

```php
var_dump($query->fetchAll());
die();
```

```
array (size=1)
  0 =>
    array (size=20)
      'id' => string '2' (length=1)
      0 => string '2' (length=1)
      'nom' => string 'Kebab Clermont' (length=14)
      1 => string 'Kebab Clermont' (length=14)
      'num_rue' => string '10' (length=2)
      2 => string '10' (length=2)
      'nom_rue' => string 'Place de Jaude' (length=14)
      3 => string 'Place de Jaude' (length=14)
      'ville' => string 'Clermont-Fd' (length=11)
      4 => string 'Clermont-Fd' (length=11)
      'code_postal' => string '63000' (length=5)
      5 => string '63000' (length=5)
      'type' => string 'Fast Food' (length=9)
      6 => string 'Fast Food' (length=9)
      'image_link' => string 'images/kebab.jpeg' (length=17)
      7 => string 'images/kebab.jpeg' (length=17)
      'star' => string '3' (length=1)
      8 => string '3' (length=1)
      'slug' => string 'fast-food' (length=9)
      9 => string 'fast-food' (length=9)
```

Les restaurants fast-food de Resto'BC !



Kebab Clermont
10 Place de Jaude 63000
Clermont-Fd
★★★☆☆

Insertion en base de donnée :

```php
require 'utils/bdd.php';
$req = $bdd->prepare(
    query: 'INSERT INTO restaurants(nom, num_rue, email)
VALUES(:nom, :num_rue, :email)');
$req->execute([
    'nom' => $_POST['nom'],
    'num_rue' => $_POST['num_rue'],
    'email' => $_POST['email'],
]);
}
```

Nous utiliserons toujours une requête préparée afin de prévenir toutes sortes d'attaques possible (comme une injection SQL par exemple).

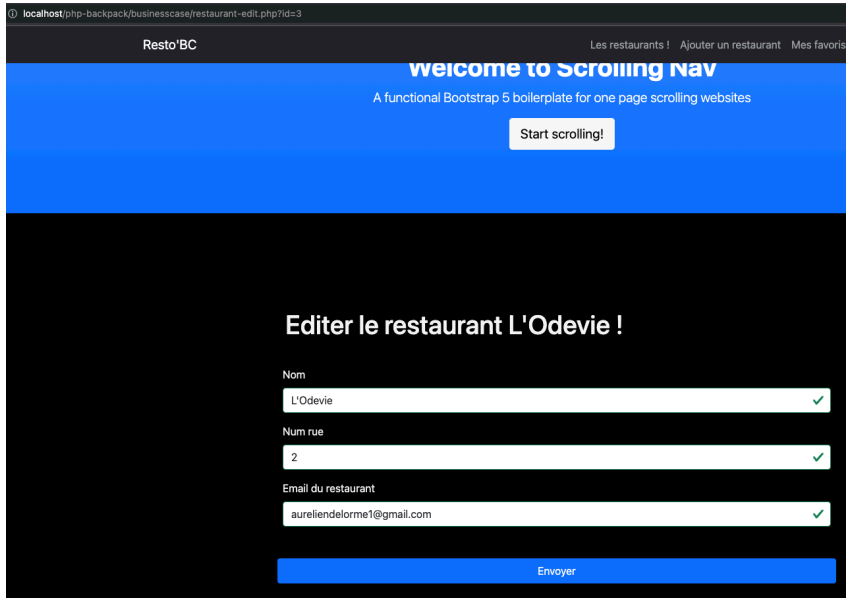
Suppression de données

```php
<?php
    require 'utils/bdd.php';
    $id = $_GET['id'];

    $res = $bdd->prepare( query: 'DELETE FROM restaurants WHERE id = :id');
    $res->execute(['id'=> $id]);

    header( header: 'Location: restaurants.php');
?>
```

Nous effectuons encore ici une requête préparée qui ira supprimer un restaurant en fonction de son ID

Editer des données

1) Création d'un formulaire d'édition
2) Selection de l'a ligne qu'il faut mettre à jour
3) Affichage du formulaire avec les valeurs pré-remplies
4) Lorsque le formulaire est soumis, on le vérifie
5) On effectue notre requête de mise à jour en fonction du paramètre GET envoyé




```php
$restaurant = findRestaurantById($_GET['id']);
```

```php
require 'utils/bdd.php';
$req = $bdd->prepare(
    query: 'UPDATE restaurants SET nom = :nom, num_rue = :num_rue, email = :email WHERE id = :id');
$req->execute([
    'nom' => $_POST['nom'],
    'num_rue' => $_POST['num_rue'],
    'email' => $_POST['email'],
    'id'=> $restaurant['id']
]);
header( header: "Location: restaurants.php");
```

```php
<form class="row g-3" method="post" action="restaurant-edit.php?id=<?php echo($_GET["id"]);?>">
```

Echanges / Questions