Accéder aux bases de données avec PDO de PHP

1. Objectifs

- Être capable d'utiliser les API php pour accéder à une base de données MySQL

2. Prérequis

- Pour pouvoir comprendre au mieux ce tutoriel, quelques connaissances sont requises.
  - **MySQL**: Vous devez savoir ce qu'est une BDD (base de données), ainsi que tout le vocabulaire comme : table, champ, ligne, requête SQL, ...
  - Il vous est aussi conseillé de connaître les bases du langage SQL pour pouvoir comprendre les requêtes de base.

3. Définition

- **PDO** (PHP Data Objects) est une extension définissant l'interface pour accéder à plusieurs types de base de données, fournie automatiquement depuis PHP 5.1.
- Fournit une interface d'abstraction à l'accès aux données, plus sécurisé (si bien utilisé)
- **PDO** (PHP Data Objects) est une extension PHP qui définit une interface d'accès à une base de données.
- Il est inclus dans PHP depuis sa version 5.1. Pour la version 5.0, vous pouvez l'obtenir avec l'extension PECL.
- Son principal avantage est qu'il permet une abstraction pour l'accès aux données. C'est-à-dire que les fonctions pour exécuter des requêtes et pour récupérer des données sont les mêmes, quelque-soit le serveur SQL utilisé (**MySQL**, **PostgreSQL**, ...).
- Avec ce système, pour utiliser plusieurs SGBD il n'est pas nécessaire de changer les fonctions de communication dans tout le code, mais seulement les arguments envoyés au constructeur.

4. Présentation

1. **PHP propose plusieurs moyens de se connecter à une base de données MySQL**

    1. **L'extension mysql_**

apcpedagogie.com

- Ce sont des fonctions qui permettent d'accéder à une base de données MySQL et- donc de communiquer avec MySQL. Leur nom commence toujours par `mysql_`. Toutefois, ces fonctions sont vielles et on recommande de ne plus les utiliser aujourd'hui

2. **L'extension mysqli_**

   - ce sont des fonctions améliorées d'accès à MySQL elle proposent plus de fonctionnalités et sont plus à jour.

3. **L'extension PDO**

   - c'est un outil complet qui permet d'accéder à n'importe quel type de base de données. On peut donc l'utiliser pour se connecter aussi bien à MySQL que PosigreSQL ou Oracle.
   - Ce sont toutes des extensions car **PHP** est très modulaire. On peut très facilement ajouter ou supprimer des éléments à
   - PHP, car tout le monde n'a pas forcément besoin de toutes les fonctionnalités.
   - Les fonctions mysql_ ne. sont plus à utiliser (on dit qu'elles sont "obsolètes"). Il reste à choisir entre **mysqli_** et **PDO**. Nous allons ici utiliser **PDO** car c'est cette méthode d'accès aux bases de données qui sera la plus utilisée dans les prochaines versions de PHP.
   - Par ailleurs, le gros avantage de PDO est que vous pouvez l'utiliser de la même manière pour vous connecter à n importe quel autre type de données (PostgreSQL, Oracle...).

2. **L’extension PDO comporte trois classes :**

- La classe **`PDO`**, qui permet de créer des objets représentant la connexion à la base et qui dispose des méthodes permettant de réaliser les fonctions essentielles, à savoir l’envoi de requête, la création de requêtes préparées et la gestion des transactions.
- La classe **`PDOStatement`**, qui représente, par exemple, une requête préparée ou un résultat de requête SELECT. Ses méthodes permettent de gérer les requêtes préparées et de lire les résultats des requêtes.
- La classe **`PDOException`** qui permet de gérer et d’afficher des informations sur les erreurs à l’aide d’objets.

apcpedagogie.com

3. **Notions de style procédural et orienté objet**

   - Comme l'API Mysqli, la notion de style procédural et orienté objet existe pour l'API PDO de PHP. En effet, l'API PDO de PHP peut être utilisée soit de manière procédurale en utilisant des fonctions comme pdo_connect(), pdo_prepare(), pdo_execute(), pdo_fetch() etc., soit de manière orientée objet en utilisant des classes comme PDO, PDOStatement, PDOException, etc.
   - Ainsi, vous pouvez utiliser l'un ou l'autre style de programmation selon votre préférence ou selon les besoins de votre projet. Cependant, l'utilisation de la programmation orientée objet est souvent recommandée car elle offre une meilleure encapsulation, une réutilisabilité accrue du code et une meilleure maintenabilité à long terme.

4. **Structure standard pour Php natif et Bootstrap**

```
mon-site/
├── ? assets/                    # Tous les assets front-end
│   ├── ? css/
│   │   ├── bootstrap.min.css    # Bootstrap CSS
│   │   ├── style.css            # CSS personnalisé
│   │   ├── responsive.css       # Media queries
│   │   └── admin.css            # Styles admin
│   ├── ? js/
│   │   ├── bootstrap.bundle.min.js  # Bootstrap JS
│   │   ├── jquery.min.js        # jQuery (si nécessaire)
│   │   ├── main.js              # JS personnalisé
│   │   └── charts.js            # Pour les graphiques
│   ├── ? images/
│   │   ├── ? icons/
│   │   │   ├── favicon.ico
│   │   │   └── logo.png
│   │   ├── ? photos/
│   │   ├── ? banners/
│   │   └── ? uploads/           # Images uploadées
│   └── ? fonts/
│       ├── bootstrap-icons.woff
│       └── custom-font.woff2
├── ? includes/                  # Fichiers inclus (header, footer, etc.)
│   ├── header.php
│   ├── footer.php
```

apcpedagogie.com

```
│   ├── navbar.php
│   ├── sidebar.php
│   ├── head.php                # Meta tags, CSS links
│   └── scripts.php             # JS scripts
├── 📁 pages/                   # Pages du site
│   ├── index.php               # Page d'accueil
│   ├── about.php               # À propos
│   ├── contact.php             # Contact
│   ├── login.php               # Connexion
│   ├── register.php            # Inscription
│   └── 📁 admin/               # Pages administration
│       ├── dashboard.php
│       ├── etudiants.php
│       ├── notes.php
│       └── filieres.php
├── 📁 config/                  # Configuration
│   ├── database.php
│   ├── config.php
│   └── constants.php
├── 📁 classes/                 # Classes PHP
│   ├── Database.php
│   ├── User.php
│   ├── Etudiant.php
│   ├── Note.php
│   └── Security.php
├── 📁 functions/               # Fonctions utilitaires
│   ├── helpers.php
│   ├── auth.php
│   ├── validation.php
│   └── upload.php
├── 📁 uploads/                 # Fichiers uploadés
│   ├── 📁 documents/
│   ├── 📁 photos_etudiants/
│   └── 📁 temp/
├── 📁 admin/                   # Section administration
│   ├── index.php
│   ├── 📁 includes/            # Includes spécifiques admin
│   └── 📁 assets/              # Assets spécifiques admin
└── 📁 FICHIERS RACINE
    ├── index.php               # Point d'entrée principal
    ├── .htaccess               # Rewrite rules
    ├── composer.json           # Si utilisation de Composer
    ├── README.md
```

apcpedagogie.com

```
    ├── robots.txt
    ├── sitemap.xml
    └── favicon.ico
```

5. Activez PDO de PHP

- Normalement, PDO est activé par défaut. Pour le vérifier (figure suivante), faites un clic gauche sur l'icône de WAMP dans la barre des tâches, puis allez dans le menu **PHP/Extensions PHP** et vérifiez que l'option **pdo_mysql** est bien cochée.

  Accéder aux bases de données avec PDO de PHP

6. Se connecter à MySQL avec PDO

1. **Style procédural ou orienté objet**

   - Les styles procédural et orienté objet, pour se connecter à MySQL avec PDO, sont deux approches différentes pour écrire du code qui interagit avec une base de données MySQL en utilisant l'interface de PDO.
   - La principale différence entre le style procédural et orienté objet pour se connecter à MySQL avec PDO est la manière dont le code est écrit et organisé. Le style procédural utilise des fonctions pour interagir avec la base de données, tandis que le style orienté objet utilise des méthodes. Le style orienté objet peut être plus facile à maintenir et à comprendre pour les développeurs qui sont familiers avec la programmation orientée objet.
   - Voici un exemple d'utilisation:
   - **Style procédural**

```
// Connexion à la base de données
$pdo = new PDO('mysql:host=localhost;
                dbname=ma_base_de_donnees',
               'mon_utilisateur', 'mon_mot_de_passe');

// Requête SQL
$sql = "SELECT * FROM ma_table";

// Exécution de la requête SQL
$result = $pdo->query($sql);
```

```
// Parcourir les résultats
while ($row = $result->fetch()) {
    // Traitement des données
}
```

- **Style orienté objet**

```
// Connexion à la base de données
$pdo = new PDO('mysql:host=localhost;
                dbname=ma_base_de_donnees',
                'mon_utilisateur', 'mon_mot_de_passe');

// Requête SQL
$sql = "SELECT * FROM ma_table";

// Préparation de la requête SQL
$stmt = $pdo->prepare($sql);

// Exécution de la requête SQL
$stmt->execute();

// Parcourir les résultats
while ($row = $stmt->fetch()) {
    // Traitement des données
}
```

2. **Connexion**

- Pour ouvrir une connexion à MySQL et sélectionner une base de données à l'aide de PDO en PHP, voici les étapes à suivre:
- **1-** Créer un fichier **config.php** qui contient les informations de connexion à la base de données, comme l'**hôte**, le nom d'**utilisateur**, le **mot de passe** et le **nom de la base de données**.

```
define('DB_HOST', 'localhost');     // l'hôte de la base de données MySQL
define('DB_NAME', 'nom_de_la_base_de_donnees');    // le nom de la base de
données MySQL
define('DB_USER', 'nom_d_utilisateur');    // le nom d'utilisateur MySQL
define('DB_PASSWORD', 'mot_de_passe');    // le mot de passe MySQL
define('DB_CHARSET', 'utf8');    // le jeu de caractères pour la base de
```

apcpedagogie.com

```
données MySQL
```

- **2-** Dans un autre fichier **connexion.php**, inclure le fichier **config.php** et créer une instance PDO en utilisant ces informations.

```
<?php
require_once('config.php');

try {
    $dsn = "mysql:host=".DB_HOST.";dbname=".DB_NAME.";charset=".DB_CHARSET;
    $pdo = new PDO($dsn, DB_USER, DB_PASSWORD);
    echo "Connexion réussie à la base de données " . DB_NAME . ".";
} catch(PDOException $e) {
    echo "La connexion a échoué : " . $e->getMessage();
}
```

- Dans ce code, nous avons utilisé la fonction **require_once()** pour inclure le fichier **config.php** qui contient les informations de connexion à la base de données. Ensuite, nous avons utilisé un bloc `try...catch` pour gérer les erreurs potentielles lors de la connexion à la base de données.
- Dans le bloc **try**, nous avons défini une chaîne de **connexion PDO (DSN)** en utilisant les informations de connexion à la base de données stockées dans le fichier **config.php**. Ensuite, nous avons créé un objet PDO en utilisant le DSN, le nom d'utilisateur et le mot de passe de MySQL.
- Enfin, nous avons activé le mode de gestion d'erreur PDO pour générer des exceptions en cas d'erreur, puis nous avons affiché un message de confirmation si la connexion a réussi, ou un message d'erreur si la connexion a échoué.

  Notez que ce code est un exemple et qu'il doit être adapté en fonction des spécificités de votre projet. Il est important de garder à l'esprit les bonnes pratiques de sécurité lors de la manipulation des informations de connexion à la base de données, en évitant notamment de stocker les mots de passe en clair dans des fichiers PHP.

7. Méthodes de connexion PDO couramment utilisés

1. **Première méthode**

```
<?php
require_once('config.php');

// Créer une nouvelle connexion PDO
try {
    $pdo = new
PDO("mysql:host=".DB_HOST.";dbname=".DB_NAME.";charset=".DB_CHARSET, DB_USER,
DB_PASSWORD);

    // Activer le mode d'erreur PDO
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Activer le mode d'émulation PDO pour désactiver l'utilisation de
'prepared statements'
    $pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
} catch(PDOException $e) {
    // En cas d'erreur de connexion à la base de données, afficher un message
d'erreur et arrêter le script
    die("Erreur de connexion à la base de données: ".$e->getMessage());
}
?>
```

2. **Deuxième méthode**

```
<?php
 Class connectionDB{
private $server="mysql:host=localhost;dbname=testingDB";
private $username="root";
private $password="123";
private
$options=array(PDO::ATTR_ERRMODE=>PDO::ERRMODE_EXCEPTION,PDO::ATTR_DEFAULT_FETCH_MODE=>PD
protected $connPDO;

public function open(){
    try {
        $this->connPDO = new PDO($this->server,$this->username,$this-
>password,$this->options);
        return $this->connPDO;
    } catch (PDOException $e) {
        echo "This is some problem in connection:".$e->getMessage();
```

apcpedagogie.com

```
    }
   }
   public function close(){
    $this->connPDO=null;
   }
}
?>
```

3. **comparaison**

- Les deux méthodes sont similaires et utilisent la même extension PHP PDO (PHP Data Objects) pour interagir avec la base de données MySQL. Cependant, la première méthode encapsule la connexion à la base de données dans une classe, tandis que la deuxième méthode utilise une configuration externe stockée dans un fichier séparé.
- Les avantages de la première méthode sont l'encapsulation de la connexion à la base de données dans une classe, ce qui facilite la réutilisation du code, et la possibilité de définir des fonctions supplémentaires pour manipuler la base de données. En revanche, cela peut rendre le code plus verbeux et moins lisible.
- La deuxième méthode a l'avantage d'être plus concise et de séparer la configuration de la connexion à la base de données de la logique d'application. Cela permet également de modifier la configuration de la base de données plus facilement, car elle est stockée dans un fichier séparé.
- **Dans l'ensemble, les deux méthodes sont valables et le choix dépend des préférences personnelles et des besoins de l'application. Si vous préférez encapsuler la connexion à la base de données dans une classe et ajouter des fonctions supplémentaires pour manipuler la base de données, la première méthode pourrait être préférable. Si vous voulez une approche plus concise et modulaire, la deuxième méthode pourrait être préférable.**

8. Les différents modes de PDO : `Errmode`, `Emulate` et `Fetchmode`

1. **Le mode d'erreur PDO (`Mode Errmode`)**

- Si vous avez renseigné les bonnes informations (nom de l’hôte, de la base, le login et le mot de passe), rien ne devrait s’afficher à l’écran.
- Toutefois, s’il y a une erreur (vous vous êtes trompés de mot de passe ou de nom de base de données, par exemple), PHP risque d’afficher toute la ligne qui pose

l'erreur, ce qui inclut le mot de passe !

- Le mode d'**erreur PDO** est une fonctionnalité de PDO qui vous permet de contrôler la façon dont les erreurs sont gérées par PDO. Il y a trois modes d'erreur PDO différents :
  - **1.** `PDO::ERRMODE_SILENT:`
    - En utilisant **PDO::ERRMODE_SILENT**, on peut spécifier que l'on ne souhaite pas que les erreurs générées par PDO interrompent l'exécution du script.
    - Par exemple, supposons que nous avons le code suivant pour se connecter à une base de données MySQL :

```
try {
    $pdo = new
PDO('mysql:host=localhost;dbname=ma_base_de_donnees',
'nom_utilisateur', 'mot_de_passe');
} catch(PDOException $e) {
    echo 'Erreur de connexion : '.$e->getMessage();
}
```

    - Dans ce cas, si la connexion à la base de données échoue, une exception **PDOException** sera levée et le script s'arrêtera.
    - Cependant, en utilisant PDO::ERRMODE_SILENT, on peut spécifier que l'on ne souhaite pas que l'exception soit levée et que le script continue à s'exécuter en cas d'erreur. Pour cela, il suffit d'ajouter la ligne suivante après l'instanciation de PDO :

```
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_SILENT);
```

    - Avec cette ligne, si la connexion à la base de données échoue, une erreur sera générée mais le script continuera à s'exécuter sans s'interrompre.
    - Il est important de noter que même si PDO::ERRMODE_SILENT permet de continuer à exécuter le script en cas d'erreur, il est important de prendre en compte ces erreurs pour éviter des comportements imprévus ou des données corrompues. Pour cela, il est recommandé de vérifier les codes d'erreur retournés par PDO en utilisant la méthode **PDO::errorCode()** et de gérer les erreurs de manière appropriée.
  - **2.** `PDO::ERRMODE_WARNING:`

apcpedagogie.com

- **PDO::ERRMODE_WARNING** est une constante de la bibliothèque PDO de PHP qui définit le mode de gestion des erreurs pour une connexion à une base de données.
- PDO émettra une alerte PHP pour chaque erreur rencontrée, mais le script continuera de s'exécuter.
- Lorsqu'une requête PDO échoue, elle peut générer une erreur qui peut interrompre l'exécution du script PHP. En utilisant **PDO::ERRMODE_WARNING**, on peut spécifier que l'on souhaite que PDO génère une erreur PHP de type "warning" lorsque cela se produit.
- Pour cela, il suffit d'ajouter la ligne suivante après l'instanciation de PDO :

```
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);
```

- Avec cette ligne, si la connexion à la base de données échoue, une erreur de type "warning" sera générée mais le script continuera à s'exécuter.
- Voici un exemple d'utilisation de **PDO::ERRMODE_WARNING**:

```
try {
    $pdo = new PDO("mysql:host=localhost;dbname=mydatabase",
"username", "password");
    $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING);
} catch(PDOException $e) {
    echo "Erreur de connexion : " . $e->getMessage();
}
```

- Dans cet exemple, nous configurons PDO pour émettre un avertissement en cas d'erreur en utilisant setAttribute(**PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING**). Cela permettra à votre application de continuer à s'exécuter même si des erreurs de base de données se produisent, tout en vous informant de ces erreurs.

- **3. `PDO::ERRMODE_EXCEPTION:`**
  - PDO::ERRMODE_EXCEPTION: lèvera une exception **PDOException** pour chaque erreur rencontrée. Vous pouvez ensuite capturer ces exceptions dans un bloc try...catch pour les gérer.

apcpedagogie.com

```
$pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
```

- Avec cette ligne, si une requête PDO échoue, une exception PDOException sera déclenchée et le script s'arrêtera, sauf si cette exception est capturée et gérée par un bloc try-catch
- Il est recommandé d'utiliser PDO::ERRMODE_EXCEPTION pour garantir une gestion efficace des erreurs. Cela permet de faciliter le débogage et de détecter les erreurs dès qu'elles se produisent. En utilisant des blocs try-catch pour capturer les exceptions, il est possible de gérer les erreurs de manière appropriée et de fournir un feedback clair aux utilisateurs en cas d'erreur.
- Voici un exemple d'utilisation de PDO::ERRMODE_EXCEPTION :

```
try {
    $pdo = new PDO("mysql:host=localhost;dbname=mydatabase",
"username", "password");
    $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
} catch(PDOException $e) {
    echo "Erreur de connexion : " . $e->getMessage();
}
```

- Dans cet exemple, nous configurons PDO pour lancer une exception en cas d'erreur en utilisant setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION). Cela permettra à votre application de gérer facilement les erreurs de base de données en utilisant des blocs try/catch pour capturer et traiter les exceptions qui pourraient être levées.

2. **Le mode Emulate**

- Le mode d'émulation PDO, ou PDO Emulation Mode en anglais, est une fonctionnalité de PDO qui permet à PDO de fonctionner avec des SGBD qui ne sont pas officiellement pris en charge par PDO (Microsoft Access, MongoDB, Cassandra, Redis...)
- Le mode d'émulation PDO utilise une couche d'abstraction supplémentaire pour simuler la compatibilité avec un SGBD non pris en charge.

apcpedagogie.com

- Pour activer le mode d'émulation PDO, vous pouvez définir l'option PDO::ATTR_EMULATE_PREPARES à true lors de la création d'une instance PDO. Voici un exemple :

```
$dsn = 'mysql:host=localhost;dbname=ma_base_de_donnees';
$username = 'mon_utilisateur';
$password = 'mon_mot_de_passe';
$options = [
    PDO::ATTR_EMULATE_PREPARES => true, // Activer le mode d'émulation PDO
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION // Activer le mode d'erreur
exception
];
$pdo = new PDO($dsn, $username, $password, $options);
```

3. **Le mode `Mode Fetchmode`**

- Le mode **Fetchmode** de PDO définit la façon dont les résultats de la base de données sont récupérés après l'exécution d'une requête SQL. Il spécifie le format dans lequel les données sont renvoyées par la méthode PDOStatement::fetch(). Les valeurs possibles sont :
  - **1-`PDO::FETCH_ASSOC`**: Ce mode retourne un tableau associatif où le nom de la colonne est utilisé comme clé et la valeur est le contenu de la cellule correspondante.

```
$stmt = $pdo->prepare('SELECT name, age FROM users');
$stmt->execute();
$result = $stmt->fetch(PDO::FETCH_ASSOC);

// Le tableau $result aura cette structure :
// $result = array(
//     'name' => 'John',
//     'age' => 30
// );
```

  - **2-`PDO::FETCH_NUM`**: Ce mode retourne un tableau indexé numériquement où l'indice commence à partir de zéro pour chaque colonne.

```
$stmt = $pdo->prepare('SELECT name, age FROM users');
$stmt->execute();
$result = $stmt->fetch(PDO::FETCH_NUM);
```

apcpedagogie.com

```
// Le tableau $result aura cette structure :
// $result = array(
//     0 => 'John',
//     1 => 30
// );
```

- **3-`PDO::FETCH_BOTH`**: Ce mode retourne à la fois un tableau associatif et un tableau numérique indexé.

```
$stmt = $pdo->prepare('SELECT name, age FROM users');
$stmt->execute();
$result = $stmt->fetch(PDO::FETCH_BOTH);

// Le tableau $result aura cette structure :
// $result = array(
//     'name' => 'John',
//     0 => 'John',
//     'age' => 30,
//     1 => 30
// );
```

- **4-`PDO::FETCH_OBJ`**: Ce mode retourne un objet avec des noms de propriétés qui correspondent aux noms des colonnes de la table.

```
$stmt = $pdo->prepare('SELECT name, age FROM users');
$stmt->execute();
$result = $stmt->fetch(PDO::FETCH_OBJ);

// L'objet $result aura ces propriétés :
// $result->name = 'John'
// $result->age = 30
```

- Il est important de noter que les modes Fetchmode peuvent varier en fonction du type de base de données utilisé. Les modes Fetchmode sont définis à l'aide de la méthode PDOStatement::setFetchMode().

9. Traiter les requêtes SQL avec l'objet PDO

- l'objet PDO en PHP propose deux méthodes pour réaliser des requêtes SQL sur une base de données :
  - ***La méthode `query()`*** :

apcpedagogie.com

- Cette méthode est utilisée pour exécuter une requête SQL qui ne nécessite pas de paramètres liés. Par exemple, une requête SELECT simple peut être exécutée en utilisant la méthode **query()**. Cette méthode renvoie un objet **PDOStatement** qui peut être utilisé pour récupérer les résultats de la requête.

```
// Connexion à la base de données
.......
// Exécution d'une requête SELECT
$sql = "SELECT nom, prenom, age FROM utilisateurs";
$stmt = $pdo->query($sql);

// Récupération des résultats
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
    echo "Nom : {$row['nom']}, Prénom : {$row['prenom']}, Âge :
{$row['age']}
";
}

// Fermeture de la connexion à la base de données
$pdo = null;
```

- ***La méthode*** ***`prepare()`*** :
  - La méthode `prepare()` est une méthode PDO en PHP qui permet de **préparer une requête SQL avant de l'exécuter**. Elle renvoie un objet **PDOStatement** qui peut être utilisé pour exécuter la requête plusieurs fois avec des paramètres différents, ce qui est utile notamment pour éviter les injections SQL.
  - La méthode `prepare()` est utilisée pour préparer une requête SQL qui nécessite des paramètres liés. Cette méthode renvoie également un objet **PDOStatement** qui peut être utilisé pour exécuter la requête. Les paramètres liés sont utilisés pour sécuriser les requêtes et éviter les attaques d'injection SQL. Les valeurs des paramètres sont passées à la requête à l'aide de la méthode `execute()`.
  - Il est important de noter que ces méthodes sont utilisées en conjonction avec les méthodes permettant de récupérer les résultats, telles que `fetchAll()` et `fetch()`.

    Script php

```
// Connexion à la base de données
$db = new PDO('mysql:host=localhost;dbname=test', 'utilisateur',
              'motdepasse');

// Préparation de la requête
$stmt = $db->prepare("SELECT * FROM ma_table WHERE id = :id");

// Liaison de paramètres
$stmt->bindParam(':id', $id);

// Exécution de la requête
$stmt->execute();

// Récupération des résultats
$resultat = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

Interprétation

- Dans cet exemple, la méthode `prepare()` est utilisée pour préparer la requête SQL, avec un paramètre nommé ":id". La méthode **bindParam()** est utilisée pour lier une valeur à ce paramètre, puis la méthode `execute()` est utilisée pour exécuter la requête. Enfin, la méthode **fetchAll()** est utilisée pour récupérer les résultats sous forme d'un tableau associatif.
- La méthode `prepare()` est utile pour plusieurs raisons, notamment pour éviter les attaques d'injection SQL en liant les paramètres de manière sécurisée, ainsi que pour améliorer les performances en réutilisant la même requête SQL plusieurs fois avec des paramètres différents.

> - *La méthode **`query()`** renvoie un objet de type **`PDOStatement`**.*
> - *La méthode **`fetch()`** renvoie un tableau associatif dont les clés sont le nom des champs. Avec while, on parcoure le résultat de la requête ligne par ligne à la manière d'un **`foreach`**.*

10. Exercice d'application

- **App01**
  - Création d'une base de données MySQL et développement d'un site Web avec PHP :

- **Objectif :**
  - Créer une base de données MySQL avec quatre tables et développer un site Web pour gérer les informations des clients, produits, commandes et livraisons.
- **Tâches à réaliser :**
  - **1-** Créer une base de données MySQL avec le nom "**gestion_commerce**".
  - **2-** Créer une table "**clients**" avec les champs "**client_id**" (clé primaire auto-incrémentée), "**nom**", "**prenom**", "**email**" et "**adresse**".
  - **3-** Créer une table "**produits**" avec les champs "**produit_id**" (clé primaire auto-incrémentée), "**nom**", "**description**", "**prix**" et "**quantite**".
  - **4-** Créer une table "**commandes**" avec les champs "**commande_id**" (clé primaire auto-incrémentée), "**client_id**" (clé étrangère), "**produit_id**" (clé étrangère), "**quantite**" et "**date_commande**".
  - **5-** Créer une table "**livraisons**" avec les champs "**livraison_id**" (clé primaire auto-incrémentée), "**commande_id**" (clé étrangère), "**date_livraison**" et "**adresse_livraison**".
  - **6-** Créer un fichier **config.php** pour la configuration de la connexion à la base de données.
  - **7-** Créer un fichier **dbconnect.php** qui contiendra le code pour établir une connexion à la base de données en utilisant les informations définies dans le fichier **config.php**.
  - **8-** Créer un fichier **index.php** pour la page d'accueil du site ou le Template du site.
    - Il existe de nombreux templates gratuits et responsives pour la gestion de commerce disponibles en ligne. Voici quelques-uns que vous pouvez utiliser :
      - BootstrapMade : https://bootstrapmade.com
      - Start Bootstrap : https://startbootstrap.com
      - Colorlib : https://colorlib.com
      - FreeHTML5.co : https://freehtml5.co
      - Creative Tim : https://www.creative-tim.com
      - BootstrapDash : https://www.bootstrapdash.com
      - ShapeBootstrap : https://shapebootstrap.net
      - AdminLTE : https://adminlte.io/
      - CoreUI : https://coreui.io/
      - Gentelella : https://colorlib.com/polygon/gentelella/

apcpedagogie.com

        - Start Bootstrap : https://startbootstrap.com/themes/sb-admin-2/
        - Bootstrapious : https://bootstrapious.com/free-templates
        - Templated : https://templated.co/
      - Ces templates sont tous gratuits et compatibles avec Bootstrap, ce qui signifie qu'ils sont responsive et faciles à personnaliser. Vous pouvez également trouver d'autres templates gratuits en ligne en cherchant sur des sites comme ThemeForest, Colorlib, ou Github.
    - **9-** Créer un fichier **login.php** pour la page de connexion.
    - **10-** Créer un fichier **menu.php** pour afficher le menu du site.
    - **11-** Créer un fichier **header.php** pour afficher l'en-tête du site.
    - **12-** Créer un fichier **footer.php** pour afficher le pied de page du site.
    - **13-** Créer un fichier **deconnexion.php** pour se déconnecter du site et Rredirigez l'utilisateur vers la page de connexion.
    - **14-** Créer un fichier **affichageClient.php** pour afficher les informations des clients.
    - **15-** Créer d'autres fichiers PHP pour les pages suivantes : **affichageProduit.php**, **affichageCommande.php** et **affichageLivraison.php**.
- **Contraintes techniques :**
    - Utiliser PHP avec PDO pour le développement du site.
    - Utiliser des sessions pour gérer la connexion des utilisateurs.
- **Livrerables attendus :**
    - Le script SQL de création de la base de données et des quatre tables.
    - Les fichiers PHP pour les différentes pages du site.
    - Un rapport décrivant les fonctionnalités du site et les choix techniques effectués.

  Remarque :
    - Il est recommandé de travailler en équipe de deux ou trois personnes pour réaliser ce projet.

  Solution

  Création de la base de données:

```
CREATE DATABASE gestion_commerce;
```

  Création de la table "clients":

apcpedagogie.com

```
CREATE TABLE clients (
    client_id INT(11) NOT NULL AUTO_INCREMENT,
    nom VARCHAR(255) NOT NULL,
    prenom VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    adresse VARCHAR(255) NOT NULL,
    PRIMARY KEY (client_id)
);
```

Création de la table "produits":

```
CREATE TABLE produits (
    produit_id INT(11) NOT NULL AUTO_INCREMENT,
    nom VARCHAR(255) NOT NULL,
    description VARCHAR(255) NOT NULL,
    prix DECIMAL(10,2) NOT NULL,
    quantite INT(11) NOT NULL,
    PRIMARY KEY (produit_id)
);
```

Création de la table "commandes":

```
CREATE TABLE commandes (
    commande_id INT(11) NOT NULL AUTO_INCREMENT,
    client_id INT(11) NOT NULL,
    produit_id INT(11) NOT NULL,
    quantite INT(11) NOT NULL,
    date_commande DATE NOT NULL,
    PRIMARY KEY (commande_id),
    FOREIGN KEY (client_id) REFERENCES clients(client_id),
    FOREIGN KEY (produit_id) REFERENCES produits(produit_id)
);
```

Création de la table "Livraisons":

```
CREATE TABLE livraisons (
  livraison_id INT NOT NULL AUTO_INCREMENT,
  commande_id INT NOT NULL,
  date_livraison DATE NOT NULL,
  adresse_livraison VARCHAR(255) NOT NULL,
  PRIMARY KEY (livraison_id),
  FOREIGN KEY (commande_id) REFERENCES commandes(commande_id)
```

apcpedagogie.com

```
);
```

- **App02**
  - **Énoncé**
    - Soit la base de données "**notes_des_étudiants**" dont le modèle logique relationnel est donnée ci-dessous.
      - **etudiants** (**#numetu**, nom, prenom,genre, datenaiss, rue, cp, ville)
      - **matieres** (**#codeMat**, libelleMat, coefMat)
      - **epreuves** (**#numepreuve**, datepreuve, lieu, codemat#)
      - **notations** (**#numetu**, **#numepreuve**, note)
    - Se connecter à **phpMyAdmin** (interface web écrite en PHP qui permet de gérer des bases de données MySQL).
    - Depuis l'onglet "Bases de données" créer une nouvelle base de données nommée "**notes_des_étudiants**".
    - Créer une base de données "**notes_des_étudiants**" avec les tables suivantes: **etudiants**, **matieres**, **epreuves** et **notations**.
    - Créer un fichier **config.php** contenant les informations de connexion à la base de données.
      - Le fichier **config.php** contient généralement les informations de configuration importantes pour une application PHP, telles que les informations de connexion à la base de données ou les paramètres de l'application. Il peut également contenir des constantes ou des variables globales nécessaires à l'ensemble de l'application.
      - Dans un projet PHP, le fichier "**config.php**" est généralement placé dans un dossier nommé "**includes**" ou "config" à la racine du projet. Ce dossier peut également contenir d'autres fichiers de configuration et fichiers inclus dans les pages du projet.
      - Il est recommandé de garder le contenu de ce fichier confidentiel et de ne pas le partager ou le mettre en ligne avec le reste de votre code source pour des raisons de sécurité.
    - Ajouter des instructions de gestion d'erreur pour afficher les messages d'erreur appropriés en cas d'échec de la connexion.

apcpedagogie.com

- Créer une page web pour afficher la liste des étudiants enregistrés dans la base de données avec la possibilité de les ajouter, modifier ou supprimer.
- Créer une page web pour afficher la liste des matières enregistrées dans la base de données avec la possibilité de les ajouter, modifier ou supprimer.
- Créer une page web pour afficher la liste des épreuves enregistrées dans la base de données avec la possibilité de les ajouter, modifier ou supprimer.
- Créer une page web pour afficher la liste des notations enregistrées dans la base de données avec la possibilité de les ajouter, modifier ou supprimer.
- Utiliser Bootstrap 5 pour améliorer le design et l'expérience utilisateur de chaque page.
- **Fichiers PHP nécessaires:**
    - Vous devez suivez l'architecture suivante pour réaliser votre projet
    - Dans le dossier "**public**" :
        - **index.php**: le point d'entrée de l'application
        - **.htaccess**: un fichier de configuration pour le serveur web
    - Dans le dossier "**src**" :
        - **controllers/ClientController.php**: un contrôleur pour gérer les actions relatives aux clients
        - **models/Client.php** : un modèle pour représenter les clients dans la base de données
        - **views/client/index.php**: une vue pour afficher la liste des clients
        - Dans le dossier "**vendor**" :
        - **vendor/autoload.php**: le fichier d'autoloading de Composer
    - Dans le dossier "**config**" :
        - **config.php**: un fichier de configuration pour la base de données
    - Dans le dossier "**templates**" :
        - **templates/header.php**: un modèle pour l'en-tête commun de toutes les pages
        - **templates/footer.php**: un modèle pour le pied de page commun de toutes les pages

apcpedagogie.com

- Dans le dossier "**tests**" :
  - tests/ClientTest.php : un test unitaire pour la classe Client.

Solution

Cours PHP

Source : https://apcpedagogie.com/acceder-aux-bases-de-donnees-avec-pdo-de-php · téléchargé le 08/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com