

Connexion à la base de données en PHP

Table des matières

I. Contexte	3
II. Introduction à PDO	3
III. Exercice : Appliquez la notion	4
IV. Le DataSourceName	5
V. Exercice : Appliquez la notion	6
VI. Connexion à la base	7
VII. Exercice : Appliquez la notion	9
VIII. Créer une base de données	9
IX. Exercice : Appliquez la notion	10
X. Essentiel	11
XI. Auto-évaluation	11
A. Exercice final	11
B. Exercice : Défi.....	13
Solutions des exercices	13

I. Contexte

Durée : 1 h

Environnement de travail : Local

Pré-requis : Connaître les bases du SQL et de la Programmation Orientée Objet

Contexte	Connaitre les bases du PHP
-----------------	-----------------------------------

Dans un site web, il n'est pas rare de stocker une partie des données dans une base de données. Il nous faut donc un moyen de communiquer avec depuis notre application, afin de lui envoyer les requêtes SQL et de traiter leur résultat. En PHP, le meilleur moyen d'y parvenir est la librairie PDO, qui offre un ensemble de méthodes unifiées pour tous les moteurs de base de données.

II. Introduction à PDO

Objectifs

- Apprendre ce qu'est PDO
- Comprendre l'utilité de cet outil

Mise en situation

PDO est une extension PHP permettant de gérer toute la communication avec une base de données. Grâce à PDO, nous allons pouvoir nous y connecter, lui envoyer des requêtes SQL et récupérer le résultat si besoin.

Complément	Avant PDO
-------------------	------------------

Historiquement, avant le développement de PDO, chaque moteur de base de données possédait ses propres fonctions PHP pour se connecter et communiquer avec la base. Ainsi, pour utiliser une base MySQL, il fallait utiliser les fonctions `mysql_connect`, puis `mysql_query` pour lancer la requête. Tandis que, si notre base de données était en DB2, il fallait plutôt utiliser `db2_connect` et `db2_exec`.

Ce procédé posait plusieurs problèmes : tout d'abord, il était **difficile de passer d'un projet à l'autre** s'ils n'utilisaient pas les mêmes bases de données, puisque les fonctions n'avaient pas le même nom. Par exemple, `mysql_query` et `db2_exec` permettent toutes les deux de lancer une requête. En changeant de projet, et donc de base de données, il fallait réapprendre à les manipuler.

Ensuite, en cas de changement de base de données pour une application, il fallait **vérifier tout le code** afin de modifier les appels à la base.

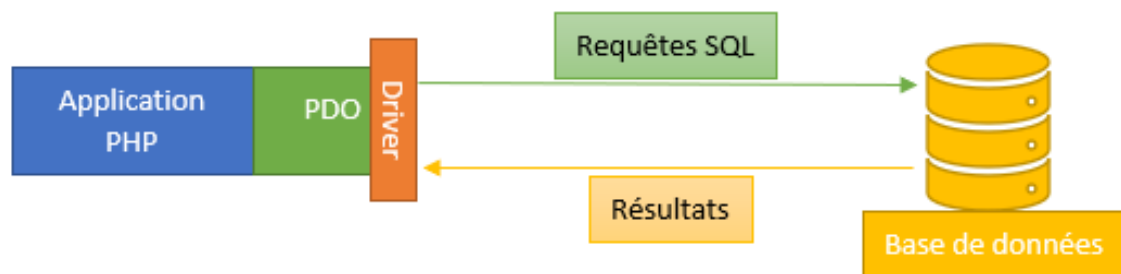
Enfin, en cas de mise à jour des librairies, comme le passage des fonctions `mysql_*` en `mysqli_*`, une **vérification intégrale du code** devait également être effectuée.

Afin de régler ces problèmes, il a été décidé d'unifier toutes les fonctions permettant de communiquer avec une base de données : c'est ainsi que PDO a vu le jour.

Qu'est-ce que PDO ?

PDO est **une interface orientée objet** qui permet de communiquer avec une base de données de manière uniforme, quel que soit son moteur. Ainsi, les méthodes à appeler seront toujours les mêmes, ce qui facilite grandement le travail : seule la syntaxe des requêtes pourra changer d'un moteur à l'autre.

En interne, PDO fonctionne grâce à un système de **drivers**. Il existe un driver par moteur de base de données et, au moment de la connexion, PDO s'occupe de charger le driver correspondant. Cette opération est transparente pour le développeur, qui ne manipulera qu'un objet de type PDO dans tous les cas.



Syntaxe À retenir

- PDO est une interface permettant de communiquer avec des bases de données. Il fonctionne, en interne, grâce à un système de drivers qui permet de communiquer avec tous les moteurs de base de données.

III. Exercice : Appliquez la notion

Question

[solution n°1 p.15]

Dans cet exercice, vous allez mettre en place une base de données MySQL dans votre environnement local et tester son fonctionnement depuis un script PHP. Pour cela, assurez-vous d'avoir bien sélectionné MySQL et PHPMyAdmin dans votre installation de XAMPP ; sinon, ajoutez-les.

Depuis l'interface PHPMyAdmin (accessible depuis le bouton **Admin** de MySQL dans le panneau de contrôle de XAMPP), créez une nouvelle base de donnée nommée `intro_pdo`. Exécutez ensuite le script suivant :

```

1 CREATE TABLE tableaux (
2   id INT(11) PRIMARY KEY AUTO_INCREMENT,
3   name VARCHAR(100),
4   painter VARCHAR(100)
5 );
6 INSERT INTO tableaux (name, painter) VALUES ("Mona Lisa", "Léonard de Vinci"), ("Ecole
   d'Athènes", "Raphaël"), ("Création d'Adam", "Michel-Ange");
  
```

Créez ensuite un fichier `index.php` et ajoutez-y le code suivant :

```

1 <?php
2 // Le "root" et la chaîne vide ci-dessous correspondent respectivement au login et au mot de
   passe de la base de données. Ce sont les valeurs par défaut, modifiez-les si vous les avez
   changés.
3 $pdo = new PDO('mysql:host=localhost;dbname=intro_pdo', 'root', '');
4 foreach ($pdo->query('SELECT * FROM tableaux') as $tableau) {
5   echo $tableau['name'].' par '.$tableau['painter'].'<br>';
6 }
  
```

Qu'est-ce qui s'affiche sur la page ?

Indice :

Si une erreur survient, vérifiez les informations de connexion (login et mot de passe) à la base de données.

Indice :

Assurez-vous que la base de données est lancée : le nom « MySQL » doit être surligné en vert.

<input type="checkbox"/>	Apache	5484 36048	80, 443	Stop	Admin	Config	Logs
<input type="checkbox"/>	MySQL	28088	3306	Stop	Admin	Config	Logs

Si ce n'est pas le cas, cliquez sur le bouton **Start**.

IV. Le DataSourceName

Objectifs

- Comprendre l'utilité d'un DSN
- Créer le DSN d'une base de données

Mise en situation

La première étape à réaliser pour que PDO puisse interagir avec la base de données est la connexion. Une fois établie, celle-ci sera ensuite maintenue pour lancer toutes les requêtes dont nous avons besoin, puis sera fermée à la fin de notre script. Cette connexion se fait grâce à une chaîne de caractères au format particulier : le **DataSourceName**.

Le préfixe

Le préfixe désigne le moteur de base de données qui sera utilisé, et donc le driver que l'objet PDO devra charger.

Chaque moteur est identifié par une chaîne de caractères qui lui est propre. Par exemple, pour utiliser PDO avec une base de données MySQL, le DSN devra être préfixé par `mysql`. En revanche, pour utiliser une base de données PostgreSQL, il faudra utiliser le préfixe `pgsql`. Tous les préfixes devront ensuite être suivis par deux points pour les séparer des attributs.

En cas de doute, la liste complète des drivers supportés par PDO, et leurs préfixes associés, est disponible dans la documentation PHP des drivers PDO¹.

Les attributs

Les attributs représentent les informations de connexion et leur valeur. Ils sont représentés par un ensemble de mots-clefs associés à une valeur via le symbole égal (=) et séparés par des points-virgules.

Ces attributs sont le plus souvent l'hôte, qui aura pour valeur l'adresse IP du serveur hébergeant notre base de données, le nom de la base de données avec laquelle nous souhaitons interagir, le nom de l'utilisateur et le mot de passe de connexion.

Cette liste d'attributs de base est une liste indicative : les attributs à fournir pour pouvoir se connecter, ainsi que leurs noms, peuvent changer d'un moteur à un autre. Il est donc important de vérifier dans la documentation quels sont les attributs nécessaires à la connexion.

Exemple Exemples de DSN

Pour se connecter à une base de données PostgreSQL, la documentation² indique que le préfixe est `pgsql` et qu'il y a besoin des attributs `host`, `port`, `dbname`, `user` et `password`. Le DSN aura donc la forme : `pgsql:host=localhost;port:5432;dbname=nomBDD;user=utilisateur;password=motDePasse`.

On remarque l'utilisation de `localhost` à la place de l'adresse IP : cela signifie simplement que notre base de données est hébergée sur le même serveur que notre application.

¹ <https://www.php.net/manual/fr/pdo.drivers.php>

² <https://www.php.net/manual/fr/ref.pdo-pgsql.connection.php>

Dans le cas d'une base de données DB2 IBM, la documentation¹ indique un préfixe `ibm` et les attributs `database`, `hostname`, `port`, `username` et `password`, mais aussi la présence des champs `protocol` et `driver`, qui ont des valeurs fixes. Le DSN aura donc la forme :

```
ibm:DRIVER={IBM                                DB2                                ODBC
DRIVER};DATABASE=testdb;HOSTNAME=localhost;PORT=50000;PROTOCOL=TCPIP;UID=utilisateur;PWD=mo
```

Remarque

Il existe des cas, comme pour les bases de données MySQL, où le nom d'utilisateur et le mot de passe ne font pas partie du DSN, mais doivent être spécifiés ailleurs. Il faut donc toujours vérifier dans la documentation pour savoir comment former un DSN pour un moteur donné.

Syntaxe À retenir

- Un DSN est une chaîne de caractères permettant de se connecter à une base de données. Elle contient les informations de connexion (sous la forme d'un préfixe) indiquant le moteur de base de données à utiliser, et des attributs renseignant les valeurs des champs à fournir, comme l'adresse IP du serveur, le nom d'utilisateur ou le mot de passe.
- Les préfixes et les attributs à fournir peuvent varier d'un moteur de base de données à un autre. Il est donc important de toujours se référer à la documentation PHP pour savoir comment former un DSN.

Complément

Les différents pilotes PDO²

V. Exercice : Appliquez la notion

Question 1

[solution n°2 p.15]

Mettons en pratique nos nouvelles connaissances sur les DSN en écrivant nos propres adresses de connexion. Pour commencer, écrivez le DSN nécessaire pour se connecter à une base de données MySQL dont l'IP du serveur est `192.168.1.36` et dont le nom est `projetExo1`. Vous pouvez bien évidemment vous aider de la documentation PHP.

Indice :

La documentation est disponible à cette adresse : <https://www.php.net/manual/en/ref.pdo-mysql.connection.php>. Tous les champs ne sont pas obligatoires : dans notre cas, nous n'avons pas besoin du port, de l'encodage ou du socket Unix.

Question 2

[solution n°3 p.15]

Un autre projet nécessite de communiquer avec une base de données Oracle hébergée sur le même serveur que votre application. Cette base de données écoute le port `1521`, porte le nom `projetExo1` et son encodage est `UTF-8`. En vous aidant de la documentation PHP, écrivez le DSN permettant de se connecter à cette base.

Indice :

La documentation du DSN d'Oracle est disponible à l'adresse suivante : <https://www.php.net/manual/fr/ref.pdo-oci.connection.php>.

¹ <https://www.php.net/manual/fr/ref.pdo-ibm.connection.php>

² <https://www.php.net/manual/fr/pdo.drivers.php>

Indice :

Pour cet exercice, ignorez la partie utilisant le fichier `tnsnames.ora`. Il existe deux manières de se connecter à une base de données Oracle, mais celle qui nous intéresse ici est celle spécifiant les informations directement dans le DSN.

Question 3

[solution n°4 p.15]

Les bases de données utilisant le moteur SQLite sont particulières, puisque l'intégralité de la base est représentée par un fichier `.sq3` présent sur le disque. Ce format est très utile pour embarquer des bases de données dans des applications légères. Pour votre projet, la base de données est dans le fichier `/var/www/projetExo1/databases/db.sq3`. En vous aidant de la documentation PHP, écrivez le DSN permettant de s'y connecter.

Indice :

Le documentation du DSN SQLite est disponible à l'adresse suivante : <https://www.php.net/manual/fr/ref.pdo-sqlite.connection.php>.

VI. Connexion à la base

Objectifs

- Utiliser le DSN dans notre application
- Connecter une application PHP à une base de données

Mise en situation

Toute la communication entre l'application PHP et la base de données va se faire grâce à l'objet `PDO`. C'est lui qui va recevoir le DSN et qui va faire le lien entre notre code et les données de la base.

Méthode Créer un objet PDO

Pour se connecter à notre base, il faut créer une instance de la classe `PDO` en lui fournissant le DSN dans le constructeur. Par exemple, pour se connecter à une base de données PostgreSQL, il faut créer un objet `PDO` avec un DSN sous la forme :

```
1 <?php
2
3 $pdo = new
    PDO('pgsql:host=localhost;port=5432;dbname=nomBDD;user=utilisateur;password=motDePasse');
```

Le constructeur accepte également 3 autres arguments facultatifs : les identifiants (login et mot de passe), qui ne doivent être renseignés que s'ils ne sont pas présents dans le DSN, et un tableau d'options, qui dépendent du moteur de base de données. Par exemple, pour se connecter à une base de données MySQL dont les identifiants de connexion ne sont pas dans le DSN, il faudra rajouter ces informations dans le constructeur :

```
1 <?php
2
3 $pdo = new PDO('mysql:dbname=nomBDD;host=localhost', 'utilisateur', 'motDePasse');
```

Gérer les erreurs

En cas de problème lors de la connexion, le constructeur va lancer une exception de type `PDOException`. Pour gérer ce cas d'erreur, il faut donc enfermer la création de chaque nouvelle instance dans un bloc `try` et `catch` l'exception.

```
1 <?php
2
3 try {
4     $pdo = new PDO('mysql:dbname=nomBDD;host=localhost', 'utilisateur', 'motDePasse');
5 } catch (PDOException $e) {
6     // Gestion de l'exception
7 }
```

Remarque

Une `PDOException` hérite de la classe `Exception` : elle dispose donc des mêmes méthodes permettant de récupérer les informations de l'exception, comme `getMessage()`, `getFile()`...

Attention Risque de sécurité

Si cette exception n'est pas attrapée, elle risque d'afficher un message d'erreur comportant les données de connexion à la base de données (comme le DSN ou le login/mot de passe fourni). Ces informations très sensibles et leur affichage représentent une très importante faille de sécurité. Il est donc recommandé de toujours `catch` les `PDOException` et de ne jamais les `throw` directement. Il est également recommandé de masquer l'affichage des messages d'erreur sur les serveurs de production pour éviter tout risque.

Déconnexion et connexions persistantes

Une fois l'objet créé et la connexion effectuée, elle sera automatiquement fermée lorsque le destructeur de l'objet sera appelé : soit si l'objet est détruit manuellement, soit automatiquement à la fin du script.

```
1 <?php
2
3 $pdo = new PDO('mysql:dbname=nomBDD;host=localhost', 'utilisateur', 'motDePasse');
4 $pdo = null; //Déconnexion
5
6 $pdo2 = new PDO('mysql:dbname=nomBDD;host=localhost', 'utilisateur', 'motDePasse');
7 // Fin du script : déconnexion automatique
```

Pour garder la connexion ouverte entre les différents scripts, il est possible de créer une connexion persistante. Ainsi, plutôt que de se fermer automatiquement, la connexion restera ouverte et, au prochain appel d'une connexion avec le même DSN, c'est la connexion précédemment ouverte qui sera utilisée au lieu d'en créer une nouvelle.

Cela se fait en utilisant l'option `PDO::ATTR_PERSISTENT` au moment de l'ouverture de la connexion.

```
1 <?php
2
3 // Le tableau d'options est le 4ème paramètre du constructeur
4 $pdo = new PDO('mysql:dbname=nomBDD;host=localhost', 'utilisateur', 'motDePasse',
5     [PDO::ATTR_PERSISTENT => true]);
```

Attention

Tous les moteurs de base de données ne sont pas compatibles avec cette option. De plus, d'autres moteurs peuvent n'autoriser qu'un certain nombre de connexions en parallèle, limite qui serait atteinte rapidement en utilisant des connexions persistantes, ce qui provoquerait des erreurs de connexion. Cette solution ne doit être envisagée que dans certains cas où la connexion à la base de données est très lente, et son utilisation doit être surveillée attentivement.

Syntaxe **À retenir**

- Pour se connecter à la base de données via PDO, il faut créer une instance de PDO en lui passant le DSN dans le constructeur. S'ils ne sont pas spécifiés dans le DSN, le login et le mot de passe peuvent être passés en second et troisième paramètres. La création de l'objet lance une `PDOException` en cas d'erreur, qu'il est conseillé d'attraper pour éviter les failles de sécurité.
- La connexion persiste tant que l'objet existe : elle est donc détruite automatiquement à la fin du script. Il est possible de changer ce comportement en utilisant des connexions persistantes via l'option `PDO::ATTR_PERSISTENT`, mais le nombre de connexions parallèles doit être surveillé pour éviter la saturation.

ComplémentLes connexions PDO en PHP¹

VII. Exercice : Appliquez la notion

Question

[solution n°5 p.15]

Dans le premier exercice, le script de connexion à la base de données était très simpliste et peu sécurisé :

```
1 <?php
2 // Le "root" et la chaîne vide ci-dessous correspondent respectivement au login et au mot de
  passe de la base de données. Ce sont les valeurs par défaut, modifiez-les si vous les avez
  changés.
3 $pdo = new PDO('mysql:host=localhost;dbname=intro_pdo', 'root', '');
4 foreach ($pdo->query('SELECT * FROM tableaux') as $tableau) {
5     echo $tableau['name'].' par '.$tableau['painter'].'<br>';
6 }
```

Modifiez ce script de manière à sécuriser la connexion en évitant la remontée du message d'erreur par défaut. À la place, loggez l'erreur dans un fichier texte et affichez simplement le message : « Une erreur de connexion à la base de données est survenue ».

Pour provoquer simplement une erreur afin de tester votre développement, il est possible de mettre un préfixe erroné dans le DSN, ou simplement de stopper MySQL dans le panel d'administration de XAMPP. Pour rappel, la structure et le contenu de la base de données définis dans l'exercice précédent sont :

```
1 CREATE TABLE tableaux (
2     id INT(11) PRIMARY KEY AUTO_INCREMENT,
3     name VARCHAR(100),
4     painter VARCHAR(100)
5 );
6 INSERT INTO tableaux (name, painter) VALUES ("Mona Lisa", "Léonard de Vinci"), ("Ecole
  d'Athènes", "Raphaël"), ("Création d'Adam", "Michel-Ange");
```

VIII. Créer une base de données

Objectifs

- Lancer des requêtes via PDO
- Créer une base de données depuis l'application PHP

¹ <https://www.php.net/manual/fr/pdo.connections.php>

Mise en situation

Maintenant que nous sommes connectés à notre base de données, nous allons pouvoir lancer nos premières requêtes. Notre premier objectif va être de créer une base de données directement depuis notre application PHP grâce à une requête `CREATE DATABASE`.

Remarque

Étant donné que nous souhaitons créer une base de données, l'attribut `dbname` du DSN (désignant le nom de la base de données sur laquelle faire les requêtes) n'est pas renseigné. Une fois notre base de données créée, il sera possible de créer une nouvelle connexion avec un DSN qui comportera cet attribut.

Attention

La méthode `exec()` peut retourner `false`, mais également une valeur qui peut être apparentée à `false` (comme un 0). Il est donc primordial d'utiliser l'égalité stricte `!==` pour éviter les faux négatifs.

Syntaxe À retenir

- Il existe plusieurs méthodes permettant de lancer une requête depuis un objet PDO. Pour récupérer le nombre de lignes affectées, il faut utiliser la méthode `exec()`. Cette méthode retourne également `false` en cas d'erreur lors de l'exécution de la requête.

Complément

Les connexions PDO en PHP¹

IX. Exercice : Appliquez la notion

Question 1

[solution n°6 p.15]

Dans les exercices précédents, nous avons créé manuellement des bases de données et des tables permettant de gérer les tableaux d'un musée, et une simple application PHP permettant d'afficher la liste des tableaux présents en base.

Si nous décidons de vendre cette solution à plusieurs musées, il faudra procéder à la création des bases de données et des tables sur chaque serveur de chaque musée. Il est donc intéressant d'automatiser cette installation, plutôt que d'avoir à lancer les requêtes SQL nous-mêmes.

Créez un script `install.php` qui va s'occuper de créer une base de données appelée `musee` sur le serveur local.

Question 2

[solution n°7 p.15]

La méthode `exec` peut être utilisée dans d'autres cas que la création de base de données. Ici, si on lance le script d'installation sur un serveur sur lequel une base de données appelée `musee` existe déjà, alors la requête va échouer. Modifiez le script pour lancer une requête `DROP DATABASE IF EXISTS musee`, qui va supprimer la base de données si elle existe déjà, avant de faire le `CREATE DATABASE`.

¹ <https://www.php.net/manual/fr/pdo.connections.php>

Question 3

[solution n°8 p.16]

Pour terminer notre script d'installation, il nous faut maintenant créer la table permettant à notre application de fonctionner, et d'y insérer quelques données de test. Pour cela, ajoutez le code nécessaire permettant d'envoyer les requêtes suivantes grâce à la méthode `exec`.

```
1 CREATE TABLE tableaux (  
2   id INT(11) PRIMARY KEY AUTO_INCREMENT,  
3   name VARCHAR(100),  
4   painter VARCHAR(100)  
5 );
```

Puis, si la création de la table s'est bien déroulée :

```
1 INSERT INTO tableaux (name, painter) VALUES ("Mona Lisa", "Léonard de Vinci"), ("Ecole  
d'Athènes", "Raphaël"), ("Création d'Adam", "Michel-Ange");
```

Indice :

Attention : le DSN de l'objet PDO utilisé pour créer la base de données n'a pas d'attribut `dbname`, donc si vous utilisez le même objet, PDO ne saura pas où créer la table. Il va falloir instancier un nouvel objet PDO.

X. Essentiel

XI. Auto-évaluation

A. Exercice final

Exercice 1

[solution n°9 p.16]

Exercice

À quoi sert PDO ?

- ☐ À charger n'importe quel type de données
- ☐ À communiquer avec n'importe quelle base de données
- ☐ À communiquer avec n'importe quel système d'exploitation

Exercice

Qu'est-ce qu'un DSN ?

- ☐ Une chaîne de caractères permettant de se connecter à une base de données
- ☐ Une requête SQL envoyée via PDO
- ☐ Un système permettant de lier une URL à une adresse IP

Exercice

Quel est le format d'un DSN ?

- ☐ moteur:attribut=valeur,attribut=valeur,attribut=valeur
- ☐ moteur:attribut=valeur;attribut=valeur;attribut=valeur
- ☐ attribut=valeur;attribut=valeur;attribut=valeur:moteur
- ☐ attribut=valeur,attribut=valeur,attribut=valeur:moteur

Exercice

Laquelle de ces propositions est un DSN MySQL valide ?

- ☐ mysql:host=localhost;database=testdb
- ☐ mysql:user=login;password;host=localhost;dbname=testdb
- ☐ mysql:host=localhost;dbname=testdb
- ☐ sql:host=localhost;dbname=testdb

Exercice

Comment se connecter à une base de données grâce à PDO ?

- ☐ En utilisant la méthode `connect` de l'objet PDO `$pdo = new PDO(); $pdo->connect($dsn);`
- ☐ Via la méthode statique `getInstance` de la classe PDO `$pdo = PDO::getInstance($dsn);`
- ☐ En créant une nouvelle instance du driver de PDO correspondant `$pdo = new MySQL($dsn);`
- ☐ En créant une nouvelle instance de la classe PDO `$pdo = new PDO($dsn);`

Exercice

Comment savoir si une erreur s'est produite lors de la connexion ?

- ☐ La création de l'objet retourne `false` en case d'erreur
- ☐ En utilisant la méthode `hasError()` de l'objet PDO
- ☐ En attrapant une `PDOException` au moment de la connexion

Exercice

Comment se déconnecter d'une base de données ?

- ☐ En utilisant la méthode `disconnect()` de l'objet PDO
- ☐ En détruisant l'objet PDO
- ☐ En envoyant une requête SQL `DISCONNECT`

Exercice

Qu'est-ce qu'une connexion persistante ?

- ☐ Une connexion qui ne se déconnecte pas et qui peut être réutilisée d'un script à l'autre
- ☐ Une connexion qui permet d'envoyer plusieurs requêtes dans un même script
- ☐ Une connexion qui permet d'envoyer des requêtes `INSERT` pour faire persister nos données en base

Exercice

À quoi sert la méthode `exec()` ?

- ☐ Lancer une requête SQL et récupérer le résultat
- ☐ Lancer une requête SQL et récupérer le statut de la requête
- ☐ Lancer une requête SQL et récupérer le nombre de lignes affectées

Exercice

Comment savoir si une erreur s'est produite lors d'une requête ?

- ☐ La méthode `exec` retourne `false` en cas d'erreur
- ☐ En utilisant la méthode `hasError()` de l'objet PDO
- ☐ En attrapant une `PDOException` au moment de la requête

B. Exercice : Défi

Pour ce défi, vous allez concevoir une base de données pour une application de gestion immobilière et créer un script permettant à une agence de l'installer sur ses serveurs.

Question

[solution n°10 p.19]

Une agence immobilière doit gérer des **biens immobiliers**. Chaque bien possède un **prix** et une **adresse**, ainsi qu'un **identifiant numérique** permettant de récupérer ces informations.

Un bien est également composé d'un certain nombre de **pièces**. Ces pièces seront gérées dans une table à part, et sont composées d'un **identifiant numérique**, d'un **nom** (cuisine, salon...) et d'une **surface**.

Chaque pièce n'appartenant qu'à un seul bien, elles possèdent également une **référence vers le bien** auquel elles sont rattachées.

Concevez une base de données permettant de gérer ces données, puis créez un script en PHP permettant de créer la base de données et les tables.

Indice :

N'oubliez pas de supprimer la base de données du même nom si elle existe déjà.

Solutions des exercices

p. 4 Solution n°1

La page affiche une liste de peintures avec leurs peintres respectifs :

- 1 Mona Lisa par Léonard de Vinci
- 2 Ecole d'Athènes par Raphaël
- 3 Création d'Adam par Michel-Ange

p. 6 Solution n°2

Le DSN pour se connecter à la base de données est : `mysql:host=192.168.1.36;dbname=projetExo1;`

p. 6 Solution n°3

Le DSN permettant de se connecter à la base de données Oracle est `oci:dbname=//localhost:1521/projetExo1;charset=UTF-8.`

p. 7 Solution n°4

Le DSN permettant de se connecter à la base de données SQLite est : `sqlite:/var/www/projetExo1/databases/db.sqlite3.`

p. 9 Solution n°5

```
1 <?php
2 try {
3     $pdo = new PDO('mysql:host=localhost;dbname=intro_pdo', 'root', '');
4     foreach ($pdo->query('SELECT * FROM tableaux') as $tableau) {
5         echo $tableau['name'].' par '.$tableau['painter'].'<br>';
6     }
7 } catch (PDOException $e) {
8     file_put_contents('dblogs.log', $e->getMessage().PHP_EOL, FILE_APPEND);
9     echo 'Une erreur est survenue';
10 }
```

p. 10 Solution n°6

```
1 <?php
2 $pdo = new PDO('mysql:host=localhost', 'root', '');
3 if ($pdo->exec('CREATE DATABASE musee') !== false) {
4     echo 'Base de données créée<br>';
5 } else {
6     echo 'Impossible de créer la base<br>';
7 }
```

p. 10 Solution n°7

```

1 <?php
2 $pdo = new PDO('mysql:host=localhost', 'root', '');
3 if ($pdo->exec('DROP DATABASE IF EXISTS musee') !== false) {
4     if ($pdo->exec('CREATE DATABASE musee') !== false) {
5         echo 'Base de données créée<br>';
6     } else {
7         echo 'Impossible de créer la base<br>';
8     }
9 } else {
10     echo 'Impossible de supprimer la base<br>';
11 }

```

p. 11 Solution n°8

```


1 <?php
2 $pdo = new PDO('mysql:host=localhost', 'root', '');
3 if ($pdo->exec('DROP DATABASE IF EXISTS musee') !== false) {
4     if ($pdo->exec('CREATE DATABASE musee') !== false) {
5         $museumPdo = new PDO('mysql:dbname=musee;host=localhost', 'root', '');
6         if ($museumPdo->exec('CREATE TABLE tableaux (
7             id INT(11) PRIMARY KEY AUTO_INCREMENT,
8             name VARCHAR(100),
9             painter VARCHAR(100)
10         )') !== false) {
11             if ($museumPdo->exec('INSERT INTO tableaux (name, painter) VALUES ("Mona Lisa",
12                 "Léonard de Vinci"),("Ecole d\'Athènes", "Raphaël"),("Création d\'Adam", "Michel-Ange");')
13                 !== false) {
14                 echo 'Installation terminée !';
15             } else {
16                 echo 'Impossible de créer les données de test<br>';
17             }
18         } else {
19             echo 'Impossible de créer la table<br>';
20         }
21     } else {
22         echo 'Impossible de créer la base<br>';
23     }
24 } else {
25     echo 'Impossible de supprimer la base<br>';
26 }

```

Exercice p. 11 Solution n°9

Exercice


À quoi sert PDO ?

- ☐ À charger n'importe quel type de données
- ☒ À communiquer avec n'importe quelle base de données
- ☐ À communiquer avec n'importe quel système d'exploitation
-  PDO permet de communiquer avec des bases de données. Grâce à son système de drivers, il peut s'adapter à n'importe quel moteur de base de données.

Exercice

Qu'est-ce qu'un DSN ?


- ☒ Une chaîne de caractères permettant de se connecter à une base de données
- ☐ Une requête SQL envoyée via PDO
- ☐ Un système permettant de lier une URL à une adresse IP
On parle de DNS (Domain Name System) et non pas de DSN.

 Le DataSourceName est une chaîne de caractères contenant toutes les informations permettant de se connecter à une base de données.

Exercice

Quel est le format d'un DSN ?


- ☐ moteur:attribut=valeur,attribut=valeur,attribut=valeur
- ☒ moteur:attribut=valeur;attribut=valeur;attribut=valeur
- ☐ attribut=valeur;attribut=valeur;attribut=valeur:moteur
- ☐ attribut=valeur,attribut=valeur,attribut=valeur:moteur

 Un DSN est composé du moteur de la base de données, puis d'un ensemble d'attributs et de valeurs, séparés par des points-virgules.

Exercice

Laquelle de ces propositions est un DSN MySQL valide ?

- ☐ mysql:host=localhost;database=testdb
L'attribut database n'existe pas.
- ☐ mysql:user=login;password;host=localhost;dbname=testdb
Les noms d'utilisateur et mots de passe n'ont pas d'attributs en MySQL.
- ☒ mysql:host=localhost;dbname=testdb
- ☐ sql:host=localhost;dbname=testdb
Le préfixe doit être mysql.

 D'après la documentation PHP sur le DSN MySQL¹, le préfixe doit être `mysql` et les champs `dbname` et `host` permettent d'identifier la base de données et le serveur.

Exercice

Comment se connecter à une base de données grâce à PDO ?

- ☐ En utilisant la méthode `connect` de l'objet PDO `$pdo = new PDO(); $pdo->connect($dsn);`
- ☐ Via la méthode statique `getInstance` de la classe PDO `$pdo = PDO::getInstance($dsn);`
- ☐ En créant une nouvelle instance du driver de PDO correspondant `$pdo = new MySQL($dsn);`
- ☒ En créant une nouvelle instance de la classe PDO `$pdo = new PDO($dsn);`

¹ <https://www.php.net/manual/fr/ref.pdo-mysql.connection.php>

- Q Pour se connecter à une base de données, il suffit de créer une nouvelle instance de la classe PDO en lui donnant le DSN en paramètre.

Exercice

Comment savoir si une erreur s'est produite lors de la connexion ?

- ☐ La création de l'objet retourne `false` en case d'erreur
- ☐ En utilisant la méthode `hasError()` de l'objet PDO
- ☒ En attrapant une `PDOException` au moment de la connexion

- Q Si la connexion échoue, une `PDOException` est levée. Il est important d'attraper ces exceptions, car le message d'erreur généré pourrait contenir des informations de connexion, ce qui représenterait une faille de sécurité.

Exercice

Comment se déconnecter d'une base de données ?

- ☐ En utilisant la méthode `disconnect()` de l'objet PDO
- ☒ En détruisant l'objet PDO
- ☐ En envoyant une requête SQL `DISCONNECT`

- Q La déconnexion se fait dans le destructeur de l'objet PDO, donc au moment où l'objet est détruit : soit au moment d'une destruction manuelle (via `unset` ou un écrasement de variable), soit automatiquement, à la fin du script.

Exercice

Qu'est-ce qu'une connexion persistante ?

- ☒ Une connexion qui ne se déconnecte pas et qui peut être réutilisée d'un script à l'autre
- ☐ Une connexion qui permet d'envoyer plusieurs requêtes dans un même script
- ☐ Une connexion qui permet d'envoyer des requêtes `INSERT` pour faire persister nos données en base

- Q Une connexion persistante ne se déconnecte pas au moment de la destruction de l'objet. Si une nouvelle connexion est créée avec le même DSN, alors la même connexion sera réutilisée.

Exercice

À quoi sert la méthode `exec()` ?

- ☐ Lancer une requête SQL et récupérer le résultat
- ☐ Lancer une requête SQL et récupérer le statut de la requête
- ☒ Lancer une requête SQL et récupérer le nombre de lignes affectées

- Q `exec()` permet de lancer une requête SQL et de récupérer le nombre de lignes qui ont été affectées par cette requête.

Exercice

Comment savoir si une erreur s'est produite lors d'une requête ?

- ⦿ La méthode `exec` retourne `false` en cas d'erreur
 - En utilisant la méthode `hasError()` de l'objet PDO
 - En attrapant une `PDOException` au moment de la requête
- Q Si la requête échoue, alors la méthode `exec()` retournera `false`. Il ne faut pas oublier d'utiliser l'égalité stricte au moment de comparer le retour : la fonction peut retourner 0 si aucune ligne n'est affectée, ce qui peut être interprété comme « faux » par PHP en cas d'égalité simple.

p. 13 Solution n°10

```

1 <?php
2 try {
3     $pdo = new PDO('mysql:host=localhost', 'root', '');
4     $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
5     if ($pdo->exec('DROP DATABASE IF EXISTS realEstate') !== false) {
6         if ($pdo->exec('CREATE DATABASE realEstate') !== false) {
7             $realEstate = new PDO('mysql:dbname=realEstate;host=localhost', 'root', '');
8             if ($realEstate->exec('CREATE TABLE realEstates (
9                 id INT(11) PRIMARY KEY AUTO_INCREMENT,
10                address VARCHAR(500),
11                price DECIMAL (20,2)
12            )') !== false) {
13                 if ($realEstate->exec('CREATE TABLE rooms (
14                     id INT(11) PRIMARY KEY AUTO_INCREMENT,
15                     realEstateId INT(11),
16                     name VARCHAR(100),
17                     surface DECIMAL (20,2),
18                     FOREIGN KEY (realEstateId) REFERENCES realEstates(id)
19                 )') !== false) {
20                     echo 'Installation terminée !';
21                 } else {
22                     echo 'Impossible de créer la table rooms<br>';
23                 }
24             } else {
25                 echo 'Impossible de créer la table realEstates<br>';
26             }
27         } else {
28             echo 'Impossible de créer la base<br>';
29         }
30     } else {
31         echo 'Impossible de supprimer la base<br>';
32     }
33 } catch (PDOException $exception){
34     die($exception->getMessage());
35 }
36
37

```