Créer une base de données et des tables avec PDO en PHP

Création d'une base de données en utilisant PDO

On va créer une nouvelle base de données avec PDO en PHP en utilisant la requête SQL `CREATE DATABASE` suivie du nom que l'on souhaite donner à la base de données.

Pour exécuter une requête SQL en PDO, on doit utiliser la méthode `exec()` qui va prendre en paramètre une requête SQL.

Création d'une base de données « pdodonnees ».

```
<!DOCTYPE html>
<html>
    <head>
        <title>BTS CIEL - PHP/BDD</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>
    <body>
        <h1>Bases de données</h1>
        <?php
            $servername = 'localhost';
            $username = 'root';
            $password = '';

            try{
                $dbco = new PDO("mysql:host=$servername",
$username, $password);
                $dbco->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                $sql = "CREATE DATABASE pdodonnees";
                $dbco->exec($sql);

                echo 'Base de données créée avec succès';
            }

            catch(PDOException $e){
              echo "Erreur : " . $e->getMessage();
            }
        ?>
```

```
    </body>
</html>
```

Création d'une table avec MySQL et PDO

Une base de données est constituée de tables. Les tables sont les cases dans lesquelles lesquelles on va stocker les données. Pour créer une nouvelle table dans une base de données, on va utiliser la requête SQL `CREATE TABLE` suivie du nom que l'on souhaite donner à notre table et on doit préciser entre parenthèse le nom des colonnes de la table ainsi que le type de données qui doit être stocké dans chaque colonne.

Parmi les types de données en MySQL, on peut citer :

- `INT` : accepte un nombre entier de 4 octets ;
- `VARCHAR` : accepte une chaine de longueur variable (entre 0 et 65 535 caractères) ;
- `TEXT` : accepte une chaine de caractère (entre 0 à 65 535 caractères) ;
- `DATE` : accepte une date.

On peut spécifier des attributs ou contraintes pour chacune des colonnes de la table. Ces attributs ou contraintes vont venir apporter des contraintes supplémentaires sur les données attendues (non nulle …) ou vont définir des comportements.

Attributs qu'on peut ajouter aux colonnes lors de la création de la table :

- `NOT NULL` – Signifie que chaque entrée doit contenir une valeur pour cette colonne. La valeur `null` n'est pas acceptée ;
- `UNIQUE` – Chacune des valeurs dans la colonne doit être unique ;
- `PRIMARY KEY` – Utile pour identifier de manière unique chaque nouvelle entrée dans une table. C'est une combinaison de `NOT NULL` et de `UNIQUE`. `PRIMARY KEY` ne doit s'appliquer qu'à une colonne dans une table. La colonne avec `PRIMARY KEY` est souvent une colonne d'ID (nombres qui s'auto-incrémentent) ;
- `FOREIGN KEY` – Utilisée pour empêcher des actions qui pourraient détruire les liens entre des tables. La `FOREIGN KEY` sert à identifier une colonne qui est identique à une colonne portant une `PRIMARY KEY` dans une autre table ;
- `DEFAULT` value – Sert à définir une valeur par défaut qui va être renseignée si aucune valeur n'est fournie ;
- `AUTO_INCREMENT` – MySQL va automatiquement incrémenter (ajouter 1) au champ pour chaque nouvelle entrée ;
- `UNSIGNED` – Utilisé pour les données de type nombre, cette contrainte permet de limiter les données reçues aux nombres positifs (0 inclus).

Maintenant, on va créer une table « utilisateurs » dans notre base « pdodonnees ».

Notre table va contenir 7 colonnes :

- Id
- Nom
- Prenom
- Adresse
- Ville
- CodePostal
- Mail

```
<!DOCTYPE html>
<html>
    <head>
        <title>BTS CIEL - PHP/BDD</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>
    <body>
        <h1>Bases de données</h1>
        <?php
            $servname = 'localhost';
            $dbname = 'pdodonnees';
            $user = 'root';
            $pass = '';

            try{
                $dbco = new
PDO("mysql:host=$servname;dbname=$dbname", $user, $pass);
                $dbco->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
                $sql = "CREATE TABLE utilisateurs(
                        Id INT UNSIGNED AUTO_INCREMENT
PRIMARY KEY,
                        Nom VARCHAR(30) NOT NULL,
                        Prenom VARCHAR(30) NOT NULL,
                        Adresse VARCHAR(70) NOT NULL,
                        Ville VARCHAR(30) NOT NULL,
                        Codepostal INT UNSIGNED NOT NULL,
                        Mail VARCHAR(50) NOT NULL,
                        UNIQUE(Mail))";

                $dbco->exec($sql);
                echo 'La table a été créée avec succès';
            }

            catch(PDOException $e){
              echo "Erreur : " . $e->getMessage();
            }
```

```
        ?>

    </body>
</html>
```

On a donc créé la table « utilisateurs » en utilisant la requête SQL `CREATE TABLE utilisateurs`. Entre les parenthèses, on précise les colonnes que doit contenir la table en indiquant le type de données attendues et les contraintes relatives à chaque colonne et en définissant l'une de nos colonnes comme `PRIMARY KEY`.

On peut noter qu'on a ajouté une contrainte `UNIQUE` pour la colonne Mail de manière un peu différente du reste.

Après exécution du code, on peut vérifier dans phpMyAdmin que la table a bien été créée avec ses colonnes en cliquant sur le nom de la table dans la base de données puis en cliquant sur « Structure ».

Insérer des données dans une table

Pour insérer des données dans une table, on doit utiliser l'instruction SQL `INSERT INTO` suivie du nom de la table dans laquelle on souhaite insérer une nouvelle entrée avec sa structure puis le mot clef `VALUES` avec les différentes valeurs à insérer.

INSERT INTO nom_de_table (nom_colonne1, nom_colonne2, nom_colonne3, …)
VALUES (valeur1, valeur2, valeur3, …)

Il y a cependant quelques règles de syntaxe à respecter :

- Les valeurs de type chaine de caractère (String) doivent être placées entre apostrophes ;
- La valeur NULL ne doit pas être placée entre apostrophes ;
- Les valeurs de type numérique ne doivent pas être placées entre apostrophes.

A priori, on doit avoir autant de valeurs à insérer qu'il y a de colonnes dans votre table. Cependant, il n'est pas nécessaire de préciser les colonnes possédant un attribut `AUTO_INCREMENT` ou `TIMESTAMP` ni leurs valeurs associées puisque par définition MySQL stockera automatiquement les valeurs courantes.

```
<!DOCTYPE html>
<html>
    <head>
        <title>BTS CIEL PHP/BDD</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>
    <body>
        <h1>Bases de données</h1>
```

```
        <?php
            $servname = 'localhost';
            $dbname = 'pdodonnees';
            $user = 'root';
            $pass = '';

            try{
                $dbco = new
PDO("mysql:host=$servname;dbname=$dbname", $user, $pass);
                $dbco->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                $sql = "INSERT INTO
utilisateurs(Nom,Prenom,Adresse,Ville,Codepostal,Mail)
                        VALUES('Carnus','Charles','Avenue de
Bourran','Rodez',12000,'lycee@carnus.fr')";

                $dbco->exec($sql);
                echo 'Données ajoutées à la table';
            }

            catch(PDOException $e){
              echo "Erreur : " . $e->getMessage();
            }
        ?>

    </body>
</html>
```

Insérer plusieurs entrées dans une table

```
<!DOCTYPE html>
<html>
    <head>
        <title>BTS CIEL - PHP/BDD</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>
    <body>
        <h1>Bases de données</h1>
        <?php
            $servname = 'localhost';
            $dbname = 'pdodonnees';
```

```
            $user = 'root';
            $pass = '';

            try{
                $dbco = new
PDO("mysql:host=$servname;dbname=$dbname", $user, $pass);
                $dbco->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                $sql1 = "INSERT INTO
utilisateurs(Nom,Prenom,Adresse,Ville,Codepostal,Mail)
                        VALUES('Nom1','Prenom1','Rue
1','Toulouse',31000,'nom1@carnus.fr')";
                $dbco->exec($sql1);

                $sql2 = "INSERT INTO
utilisateurs(Nom,Prenom,Adresse,Ville,Codepostal,Mail)
                        VALUES('Nom2','Prenom2','Avenue
2','Paris',75000,'nom2@carnus.fr')";
                $dbco->exec($sql2);

                echo 'Données ajoutées à la table';
            }

            catch(PDOException $e){
              echo "Erreur : " . $e->getMessage();
            }
        ?>

    </body>
</html>
```

L'un des inconvénients de cette méthode est que s'il y a un problème d'exécution en cours du script, certaines entrées vont être insérées et pas d'autres et certaines entrées pourraient ne pas avoir toutes leurs données insérées.

```
<!DOCTYPE html>
<html>
    <head>
        <title>BTS CIEL - PHP/BDD</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>
    <body>
        <h1>Bases de données</h1>
```

```
        <?php
            $servname = 'localhost';
            $dbname = 'pdodonnees';
            $user = 'root';
            $pass = '';

            try{
                $dbco = new
PDO("mysql:host=$servname;dbname=$dbname", $user, $pass);
                $dbco->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                $dbco->beginTransaction();

                $sql1 = "INSERT INTO
utilisateurs(Nom,Prenom,Adresse,Ville,Codepostal,Mail)
                        VALUES('Nom3','Prenom3','Rue
3','Rodez',12000,'nom3@carnus.fr')";
                $dbco->exec($sql1);

                $sql2 = "INSERT INTO
utilisateurs(Nom,Prenom,Adresse,Ville,Codepostal,Mail)
                        VALUES('Nom4','Prenom4','Avenue
4','Rodez',12000,'nom4@carnus.fr')";
                $dbco->exec($sql2);

                $dbco->commit();
                echo 'Données ajoutées à la table';
            }

            catch(PDOException $e){
                $dbco->rollBack();
              echo "Erreur : " . $e->getMessage();
            }
        ?>

    </body>
</html>
```

Si on tente d'exécuter le code une 2ème fois, une erreur va être lancée car on a une contrainte `UNIQUE` sur le champ Mail de ma table. La méthode `rollBack()` va donc s'exécuter et aucune transaction ne va être validée.

Mettre à jour des données dans une table

On va utiliser l'instruction SQL `UPDATE` suivie du nom de la table pour mettre à jour des données dans une table. Cette instruction doit être accompagnée de `SET` qui va servir à préciser la colonne à mettre à jour ainsi que la nouvelle valeur pour la colonne.

Dans la table « utilisateurs », il y a une erreur l'adresse mail de l'utilisateur « Nom2 Prenom2 », on utilise donc `UPDATE` pour `SET` (régler) une nouvelle valeur pour la colonne Mail de cet utilisateur.

Pour ne mettre à jour que la valeur du Mail correspondant à cette entrée, on va utiliser `WHERE` en donnant une condition sur l'id.

```
<!DOCTYPE html>
<html>
    <head>
        <title>BTS CIEL - PHP/BDD</title>
        <meta charset='utf-8'>
    </head>
    <body>
        <h1>Bases de données</h1>
        <?php
            $servname = "localhost"; $dbname = "pdodonnees";
$user = "root"; $pass = "";

            try{
                $dbco = new
PDO("mysql:host=$servname;dbname=$dbname", $user, $pass);
                $dbco->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                //On prépare la requête et on l'exécute
                $sth = $dbco->prepare("
                  UPDATE utilisateurs
                  SET Mail='prenom2.nom2@carnus.fr'
                  WHERE Id=3
                ");
                $sth->execute();

                //On affiche le nombre d'entrées mise à jour
                $count = $sth->rowCount();
```

```
                print('Mise à jour de ' .$count. '
entrée(s)');
            }

            catch(PDOException $e){
                echo "Erreur : " . $e->getMessage();
            }
        ?>

    </body>
</html>
```

Ajouter une colonne dans une table

Pour ajouter une colonne, on utilise ADD avec le nom de la colonne à ajouter et le type de données attendu.

Par exemple, on pourrait ajouter une colonne « Date » dans notre table « utilisateurs » qui stockerait automatiquement la date d'inscription des utilisateurs.

```
<!DOCTYPE html>
<html>
    <head>
        <title>CIEL - PHP/BDD</title>
        <meta charset='utf-8'>
    </head>
    <body>
        <h1>Bases de données MySQL</h1>
        <?php
            $servname = "localhost"; $dbname = "pdodonnees";
$user = "root"; $pass = "";

            try{
                $dbco = new
PDO("mysql:host=$servname;dbname=$dbname", $user, $pass);
                $dbco->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                /*Un utilisateur ne pourra jamais changer la
structure d'une table,
                 *il n'est pas nécessaire d'utiliser les
requêtes préparées*/
                $sql = "
                  ALTER TABLE utilisateurs
```

```
                    ADD Date TIMESTAMP
                ";

                $dbco->exec($sql);
                echo 'Colonne ajoutée avec succès';
            }

            catch(PDOException $e){
                echo "Erreur : " . $e->getMessage();
            }
        ?>

    </body>
</html>
```

Supprimer une colonne dans une table

Pour supprimer une colonne dans une table, on utilise `ALTER TABLE` avec l'instruction `DROP COLUMN`.

A la différence de l'instruction SQL `ADD` + nom de colonne pour ajouter une colonne, pour supprimer une colonne il faut utiliser l'instruction `DROP COLUMN` + le nom de la colonne.

On peut essayer de supprimer la colonne « Date » qu'on vient juste de créer dans la table « utilisateurs ».

```
<!DOCTYPE html>
<html>
    <head>
        <title>BTS CIEL - PHP/BDD</title>
        <meta charset='utf-8'>
    </head>
    <body>
        <h1>Bases de données</h1>
        <?php
            $servname = "localhost"; $dbname = "pdodonnees";
$user = "root"; $pass = "";

            try{
                $dbco = new
PDO("mysql:host=$servname;dbname=$dbname", $user, $pass);
                $dbco->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                $sql = "
```

```
                    ALTER TABLE utilisateurs
                    DROP COLUMN Date
                ";

                $dbco->exec($sql);
                echo 'Colonne supprimée avec succès';
            }

            catch(PDOException $e){
                echo "Erreur : " . $e->getMessage();
            }
        ?>

    </body>
</html>
```

Supprimer une ou plusieurs entrées choisies d'une table

Pour supprimer des données d'une table, on utilise l'instruction SQL `DELETE FROM`.

Pour préciser quelles entrées doivent être supprimées, on va accompagner `DELETE FROM` d'une clause `WHERE` permettant de cibler des données en particulier dans la table.

Pour supprimer une entrée en particulier, on utilise l'instruction `WHERE` sur une colonne « id » en ciblant un « id » précis. On peut également supprimer plusieurs entrées en donnant une inégalité en condition de l'instruction `WHERE` (tous les « id » supérieurs à 3 par exemple) ou en ciblant un autre type de données (supprimer toutes les entrées dont la valeur dans la colonne « Prenom » est « Prenom1 » par exemple).

Exemple : on va supprimer tous les utilisateurs dont le nom est « Prenom2 ».

```
<!DOCTYPE html>
<html>
    <head>
        <title>BTS CIEL - PHP/BDD</title>
        <meta charset='utf-8'>
    </head>
    <body>
        <h1>Bases de données MySQL</h1>
        <?php
            $servname = "localhost"; $dbname = "pdodonnees";
$user = "root"; $pass = "";

            try{
                $dbco = new
PDO("mysql:host=$servname;dbname=$dbname", $user, $pass);
```

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

                $sql = "DELETE FROM utilisateurs WHERE
Prenom='Prenom2'";
                $sth = $dbco->prepare($sql);
                $sth->execute();

                $count = $sth->rowCount();
                print('Effacement de ' .$count. ' entrées.');
            }

            catch(PDOException $e){
                echo "Erreur : " . $e->getMessage();
            }
        ?>

    </body>
</html>
```

Supprimer toutes les données d'une table

C'est une action est irréversible. Réfléchissez donc bien avant d'exécuter ce genre d'action.
Pour supprimer toutes les données d'une table sans pour autant supprimer la table ni sa structure, il suffit d'utiliser l'instruction SQL `DELETE FROM` sans préciser d'instruction `WHERE`.

Exemple : on va effacer toutes les données de la table « utilisateurs ».

```
<!DOCTYPE html>
<html>
    <head>
        <title>BTS CIEL - PHP/BDD</title>
        <meta charset='utf-8'>
    </head>
    <body>
        <h1>Bases de données</h1>
        <?php
            $servname = "localhost"; $dbname = "pdodonnees";
$user = "root"; $pass = "";

            try{
                $dbco = new
PDO("mysql:host=$servname;dbname=$dbname", $user, $pass);
```

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

                $sql = "DELETE FROM utilisateurs";
                $sth = $dbco->prepare($sql);
                $sth->execute();

                $count = $sth->rowCount();
                print('Effacement de ' .$count. ' entrées.');
            }

            catch(PDOException $e){
                echo "Erreur : " . $e->getMessage();
            }
        ?>

    </body>
</html>
```

Supprimer complètement une table de la base de données

C'est une action est irréversible. Réfléchissez donc bien avant d'exécuter ce genre d'action.
Pour supprimer complètement une table, on utilise l'instruction SQL `DROP TABLE` suivie du nom de la table que l'on souhaite supprimer.

```
<!DOCTYPE html>
<html>
    <head>
        <title>BTS CIEL - PHP/BDD</title>
        <meta charset='utf-8'>
    </head>
    <body>
        <h1>Bases de données</h1>
        <?php
            $servname = "localhost"; $dbname = "pdodonnees";
$user = "root"; $pass = "";

            try{
                $dbco = new
PDO("mysql:host=$servname;dbname=$dbname", $user, $pass);
                $dbco->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                $sql = "DROP TABLE utilisateurs";
```

```
                $dbco->exec($sql);

                echo 'Table bien supprimée';
            }

            catch(PDOException $e){
                echo "Erreur : " . $e->getMessage();
            }
        ?>

    </body>
</html>
```

Supprimer une base de données

C'est une action est irréversible. Réfléchissez donc bien avant d'exécuter ce genre d'action.

Pour supprimer une base de données, on utilise l'instruction SQL `DROP DATABASE` suivie du nom de la base de données que l'on souhaite supprimer.

```
<!DOCTYPE html>
<html>
    <head>
        <title>BTS CIEL - PHP/BDD</title>
        <meta charset='utf-8'>
    </head>
    <body>
        <h1>Bases de données</h1>
        <?php
            $servname = "localhost"; $dbname = "pdodonnees";
$user = "root"; $pass = "";

            try{
                $dbco = new
PDO("mysql:host=$servname;dbname=$dbname", $user, $pass);
                $dbco->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

                $sql = "DROP DATABASE pdodonnees";
                $dbco->exec($sql);

                echo 'Base de données supprimée';
            }
```

```
            catch(PDOException $e){
                echo "Erreur : " . $e->getMessage();
            }
        ?>

    </body>
</html>
```