TP SIN
Connexion base de données
Langage PDO

Pré requis (l'élève doit savoir):

- Savoir utiliser un ordinateur
- Connaître le html et css et PHP

Programme

Objectif terminale :

L'élève doit être capable de se connecter à une base de données sur un serveur, de modifier ou de lire des données

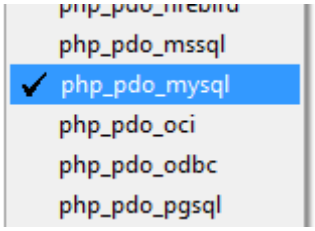
1. Introduction

Il existe plusieurs langages pour se connecter sur un serveur sur une base de données

- L'extension mysql_ : Elles permettent d'accéder seulement à une base de données mysql
- L'extension mysqli_ : Elles permettent d'accéder seulement à une base de données mysql, mais elles sont légèrement améliorées par rapport à la précédente.
- L'extension PDO : Elles permettent d'accéder à n'importe quel type de base de données.

2. Utilisation de l'extension PDO

Dans un premier temps, il faut contrôler que votre serveur peut être utilisé avec l'extension. Pour WAMP, 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 php_pdo_mysql est bien coché.



Pour les autres serveurs, enlever le point-virgule devant la ligne `php_pdo_mysql` dans le fichier php.ini.

3. Connexion à la base de données

Pour se connecter à une base de données nous avons besoin de connaître quatre paramètres.

- **le nom de l'hôte** : c'est l'adresse du serveur ou de l'ordinateur en local où MySQL est installé (comme une adresse IP). Le plus souvent, MySQL est installé sur le même ordinateur que PHP : dans ce cas, mettez la valeur `localhost` ou 127.0.0.1 (cela signifie « sur le même ordinateur »);

- **la base** : c'est le nom de la base de données à laquelle vous voulez vous connecter. Dans notre cas, la base s'appelle `sti2d` ;

- **le login** : il permet de vous identifier. Renseignez-vous auprès de votre hébergeur pour le connaître. Dans notre cas on prendra « hyrome » ;

- **le mot de passe** : Renseignez-vous auprès de votre hébergeur. Dans notre cas on prendra « 1234 ».

Dans un premier temps je vais utiliser un fichier de connexion « connexion.php » dans lequel je vais indiquer les différents paramètres.

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>Document sans titre</title>
6  </head>
7  <?php
8  $host="127.0.0.1";
9  $dbname="sti2d";
10 $login="hyrome";
11 $password="1234";
12 $pdo=new PDO('mysql:host='.$host.';dbname='.$dbname.'', $login,$password );
13
14 ?>
15 <body>
16 </body>
```

Mysql est le driver utilisé

Puis je vais réaliser un fichier dans lequel, je vais intégrer les requêtes et qui va commencer par l'appel du fichier de connexion.

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>Document sans titre</title>
6  </head>
7  <?php
8  require 'connexion.php';
9  ?>
```

Le souci, s'il y a une erreur de connexion, l'internaute risque de connaître vos paramètres de connexion. Pour cela, on va rajouter une ligne de contrôle d'erreur

```php
<?php
$host=[redacted]
$dbname
$login=
$password=
try
{
$pdo=new PDO('mysql:host='.$host.';dbname='.$dbname.'', $login,$password );
}
catch (Exception $erreur)
{
        die('Erreur : ' . $erreur->getMessage());
}
?>
<body>

</body>
</html>
```

Exemple : erreur dans le mot de passe

Remarque : les exceptions en PHP (gestion d'erreur)

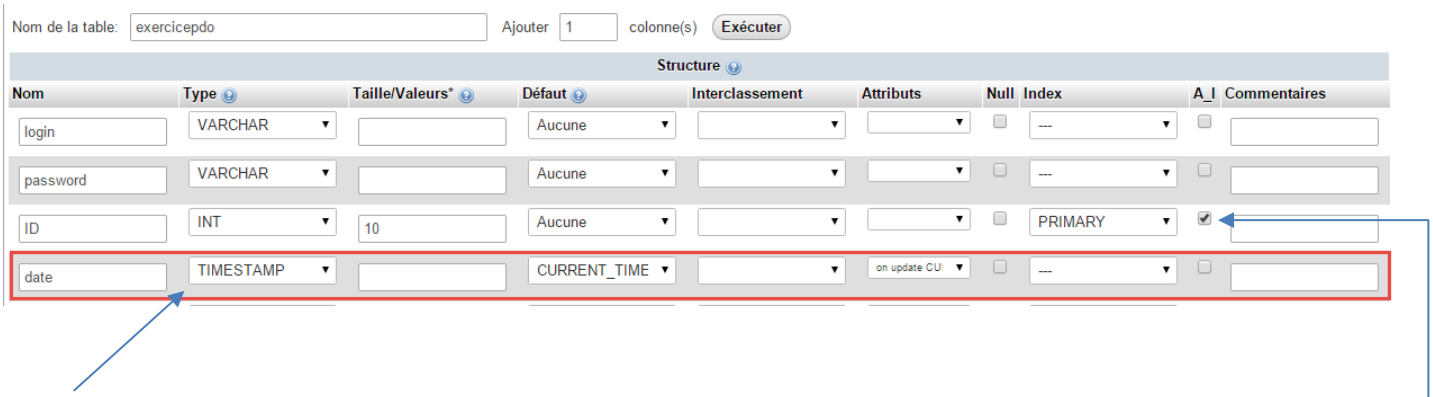
PHP a une gestion des exceptions similaire à ce qu'offrent les autres langages de programmation. Une exception peut être lancée ("throw") et attrapée ("catch") dans PHP. Le code devra être entouré d'un bloc try pour faciliter la saisie d'une exception potentielle. Chaque try doit avoir au moins un bloc catch ou finally correspondant.

4. Création de la base de données

Pour cela nous allons utiliser phpMyAdmin

Pour l'exercice on va créer une table suivante dans la base donnée sti2d

Voir : http://www.coursstimartinique.fr/video%20phpadmin.html

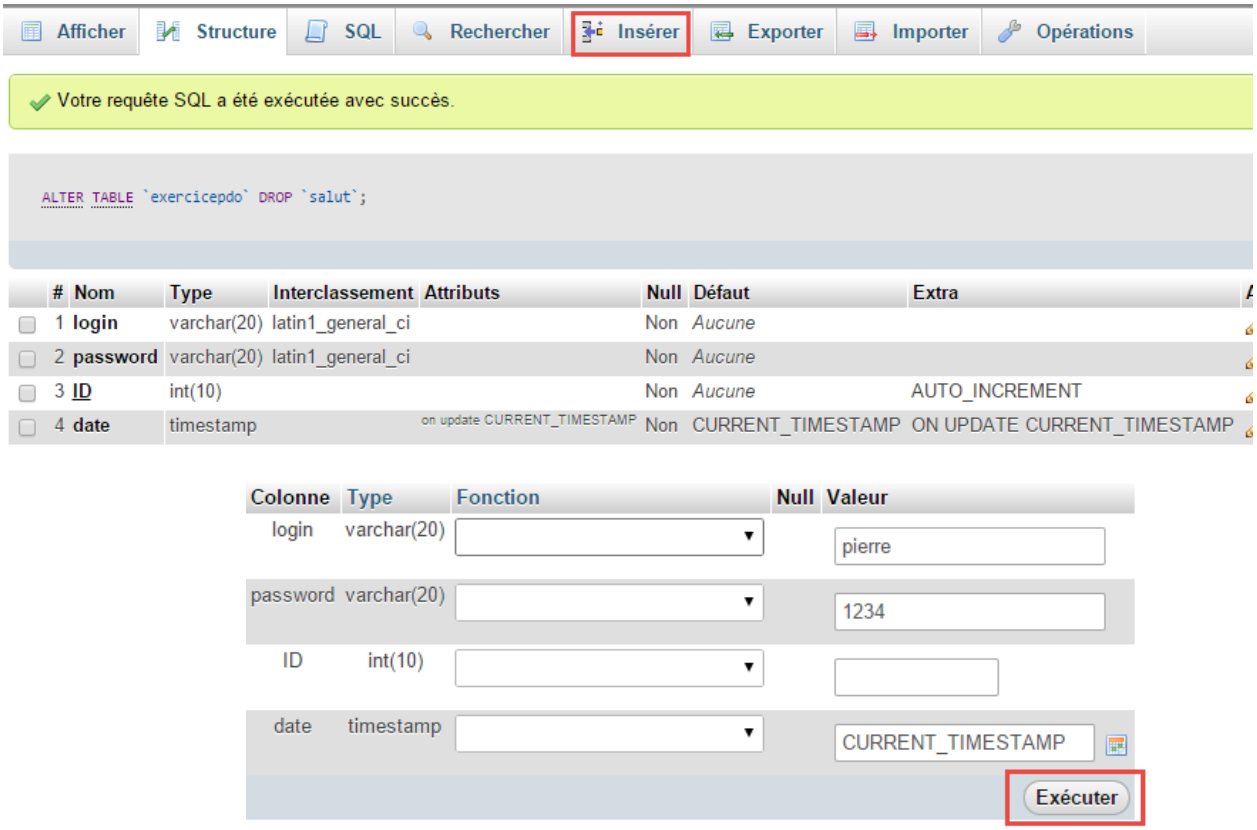


Champ permettant d'enregistrer automatiquement la date d'enregistrement des données

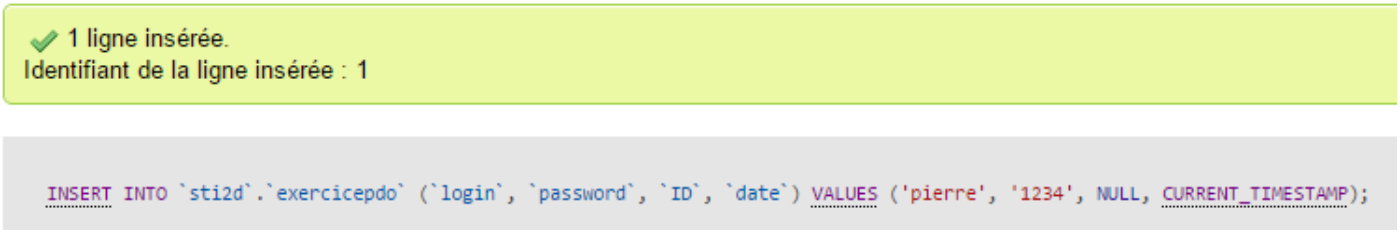
Nous avons aussi intégrer un champ ID dans lequel nous avons mis une clé qui s'incrémente automatiquement à chaque enregistrement

5. Enregistrement dans la base de données

Pour connaître la requête SQL, nous allons insérer des données dans la base depuis phpMyAdmin



Ce qui donne




```
INSERT INTO `sti2d`.`exercicepdo` (`login`, `password`, `ID`, `date`) VALUES ('pierre', '1234', NULL, CURRENT_TIMESTAMP);
```

On obtient

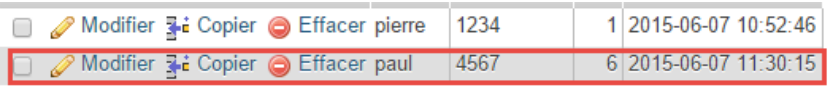
Maintenant nous allons essayer de faire la même chose en programmation

IL existe deux formes de requête : "exec" et "query".

En effet pour récupérer des données de la BDD il faut utiliser "query" et pour les trois autres (INSERT, DELETE et UPDATE) il faut utiliser "exec".

```php
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Document sans titre</title>
</head>
<?php
require "connexion.php";
//enregistrement
$pdo->exec('INSERT INTO sti2d.exercicepdo(login, password, ID, date) VALUES (\'paul\', \'4567\', NULL, CURRENT_TIMESTAMP)');
?>
<body>
</body>
</html>
```

Voilà le résultat



Nous allons refaire la même chose avec des variables, mais en utilisant une requête préparée

```php
<body>
<form action="enregistrer.php" method="get" name="form1" id="form1">
  <p>
    <label for="textfield">Rentrer le login:</label>
    <input type="text" name="login" id="login">
  </p>
  <p>
    <label for="textfield2">Rentrer le mot de passe:</label>
    <input type="password" name="password" id="password">
  </p>
    <p>
    <input type="submit" name="submit" id="submit" value="Envoyer">
  </p>
</form>
</body>
<?php
$login1=$_GET['login'];
$password1=$_GET['password'];
require "connexion.php";
//enregistrement
if (!empty($login1) && !empty($password1)) Contrôle si l'un des champs est vide
{
$req=$pdo->prepare('INSERT INTO exercicepdo(login, password) VALUES (:login2,:password2)');
$req->execute(array(
'login2'=>$login1,
'password2'=>$password1
));
echo 'Données enregistrées';
}
else
{
    echo 'le champ n est pas rempli';
}
?>
</html>
```

6. Protection des fichiers sur le serveur

Le gros souci, c'est que n'importe qui connaissant le lien, peut accéder à votre mot de passe. Pour éviter cela, on va intégrer dans le dossier du fichier de connexion différents fichiers dont un .htaccess.

a. Les fichiers htaccess

Les fichiers *.htaccess* sont des fichiers de configuration d'Apache, permettant de définir des règles dans un répertoire et dans tous ses sous-répertoires (qui n'ont pas de tel fichier à l'intérieur). On peut les utiliser pour protéger un répertoire par mot de passe, ou pour changer le nom ou l'extension de la page index, ou encore pour interdire l'accès au répertoire.
http://www.commentcamarche.net/contents/7-apache-les-fichiers-htaccess

On va créer depuis le bloc-nots un fichier .text, puis enregistrer en écrivant ".htaccess". Les apostrophes permettent à Windows d'accepter la création d'un fichier sans nom.

Dans ce fichier écrire :

AuthName "Page d'administration protégée"

AuthType Basic

AuthUserFile ".htpasswd" (fichier dans lequel sera enregistré le login et le mot de passe pour accéder au dossier depuis le navgateur)

Require valid-user

- AuthName : c'est le texte qui invitera l'utilisateur à inscrire son login et son mot de passe.;
- AuthUserFile : c'est le chemin absolu vers le fichier .htpasswd (que vous mettrez dans le même répertoire que le .htaccess).

Maintenant, il faut connaître le chemin absolu du fichier « .htpasswd » chez l'hébergeur.

On va créer un programme PHP (chemin.php) suivant sur la même racine que « .htpasswd »:

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>Document sans titre</title>
6  </head>
7  <?php echo realpath('chemin.php'); ?>
8  <body>
9  </body>
10 </html>
```

Ouvrir ensuite ce fichier depuis l'hébergeur et vous obtiendrez le chemin

On va créer ensuite le fichier « .htpasswd »

Ecrire dedans le nom des personnes pouvant accéder au document suivi du mot de passe crypté correspondant.

Remarque :

Pour connaître la valeur cryptée d'un mot, écrire le programme php suivant

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Document sans titre</title>
</head>
<?php echo crypt('toto'); ?>
<body>
</body>
</html>
```

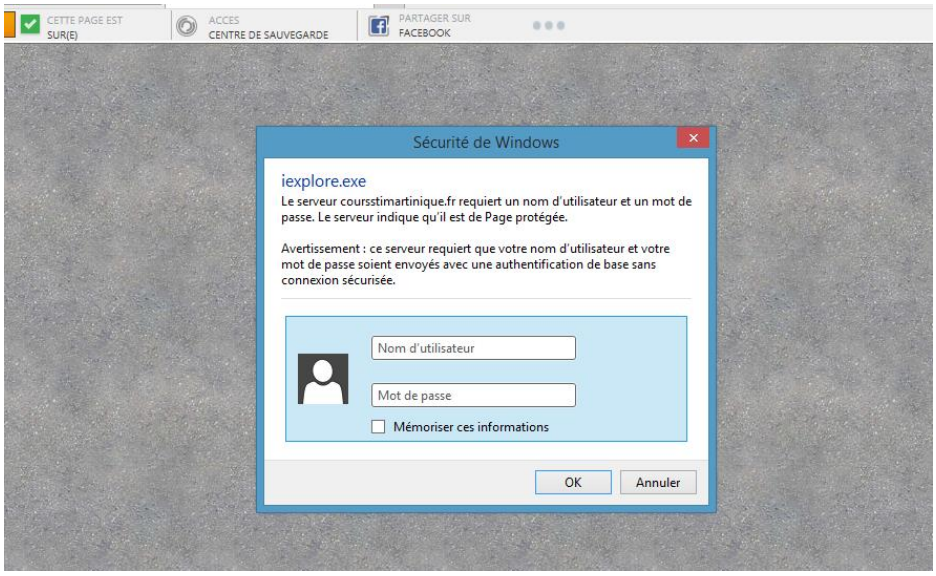
On obtient :

1X9FHfCKp$5mWq/tCCioU7IdCyMfr7R1

Donc dans le fichier « .htpasswd », on va écrire

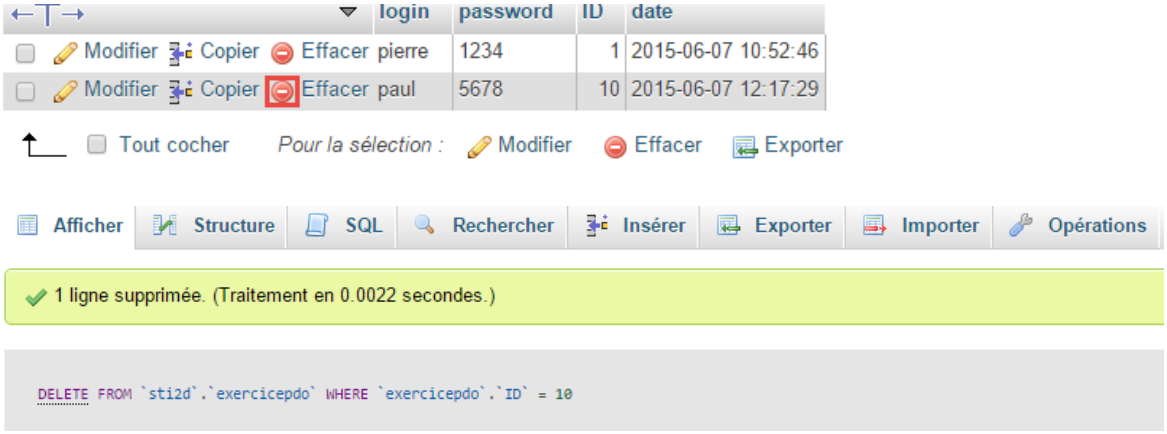
Pierre : 1X9FHfCKp$5mWq/tCCioU7IdCyMfr7R1

Lorsqu'une personne voudra accéder au dossier, on aura :



7. Effacer, modifier et sélectionner des données

Comme le paragraphe précédent nous allons utiliser phpMyAdmin, pour connaître la requête pour éffacer.



Le souci, c'est qu'il faut d'abord connaître les valeurs de chaque champ pour définir un pointeur.

Donc dans un premier temps on va créer une liste dans laquelle, on va récupérer toutes les valeurs d'un champ dans la base de données

```
21  ?>
22  <body>
23      <form action="supprimer.php" method="get" name="form1" id="form1">
24          <p>Sélectionner un login, puis appuyer sur supprimer</p>
25      <select id="login" name="login">
26      <?php
27          $req1=$pdo->query('SELECT * FROM connexion');
28    while($donne=$req1->fetch())
29    {
30          ?>
31          <option <?php echo $donne['login']; ?>><?php echo $donne['login']; ?></option>
32          <?php
33    }
34          ?>
35          </select>
36          <p>
37              <input type="submit" name="submit" id="submit" value="supprimer">
38          </p>
39
40      </form>
```

Chaque fois que vous appelez $req1->fetch(), vous passez à l'entrée suivante. La boucle est donc répétée autant de fois qu'il y a d'entrées dans votre table.

a. Les critères de sélection

En modifiant la requête précédente, il est possible de filtrer et trier très facilement vos données. Pour cela nous allons utiliser les mots clés suivant :

- WHERE : Ce mot va permettre de trier les données suivant des paramètres;

$reponse = $pdo->query('SELECT login,password FROM exercicepdo WHERE login=\'pierre\'');

- ORDER BY : Permet d'ordonner nos résultats. Exemple, on peut classer en fonction de l'âge croissant du client ;

$reponse = $pdo->query('SELECT login,password FROM exercicepdo ORDER BY age');
Pour l'âge décroissant on rajoute « DESC » à la fin « ORDER BY age DESC »

- LIMIT : Permet de ne sélectionner qu'une partie des résultats (par exemple les 10 premiers).

$reponse = $pdo->query('SELECT login,password FROM exercicepdo LIMIT 0,10');

Le premier nombre indique à partir de quel nombre on veut lire la table (0). Le deuxième indique combien d'entrées on veut sélectionner (10).

On peut aussi mélanger tous ces critères :

SELECT * prix FROM exercicepdo WHERE login='pierre' OR login='1234' ORDER BY age DESC LIMIT 0,10

Dans certain cas on est obligé d'intégrer une variable dans la sélection. Pour cela on va utiliser une requête « prepare »

Exemple :

$req=$pdo->prepare('SELECT * FROM temperature LIMIT :v1,4');

$req->bindValue(':v1', intval($x2), PDO::PARAM_INT);

$req->execute();

Maintenant on rajoute la ligne pour supprimer le login sélectionné

```php
<?php
require "connexion.php";
$login1=$_GET['login'];
$req = $pdo->prepare('DELETE FROM exercicepdo WHERE login=:login2');
$req->execute(array('login2' => $login1));
?>
```

Ce qui fait

```
Les fichiers à association dynamique de ce document ont peut-être été modifiés et doivent être à nouveau détectés par le serveur. Détecter
```

```php
<!DOCTYPE html>
<html>
<head>
    <title>supprimer sur la PDO</title>
</head>
<?php
$erreur ="";
require "connexion.php"; //récupere le fichier de connexion
if(!empty($_GET['login']))
{

 $req=$pdo->prepare('DELETE  FROM connexion WHERE login=:login2');
 $req->execute(array(
    'login2'=>$_GET['login']
    ));
    $erreur="élément supprimé";
 $req->closeCursor();
}


?>
<body>
    <form action="supprimer.php" method="get" name="form1" id="form1">
        <p>Sélectionner un login, puis appuyer sur supprimer</p>
    <select id="login" name="login">
    <?php
        $req1=$pdo->query('SELECT * FROM connexion');
  while($donne=$req1->fetch())
 {
        ?>
        <option <?php echo $donne['login']; ?>><?php echo $donne['login']; ?></option>
        <?php
  }
        ?>
        </select>
        <p>
            <input type="submit" name="submit" id="submit" value="supprimer">
        </p>

    </form>
        <p><?php echo $erreur ?></p>
</body>
</html>
```

La ligne « $reponse->closeCursor() ; » provoque la « fermeture du curseur d'analyse des résultats ». Cela signifie, en d'autres termes, que vous devez effectuer cet appel à closeCursor() chaque fois que vous avez fini de traiter le retour d'une requête, afin d'éviter d'avoir des problèmes à la requête suivante.
Si au lieu de supprimer, on veut modifier, on remplacera DELETE par UPDATE.

Exemple :

UPDATE exercicepdo SET login = 'pierre' WHERE login = ‘paul'

On remplace tous les login « pierre » par « paul »

Dans le cas où l’on ne veut remplacer qu’une seule ligne, il faut indiquer un paramètre unique pour chaque donnée.
Dans notre cas on prendra l’ID

UPDATE exercicepdo SET login = ‘pierre’, password = ‘4567’ WHERE ID = 3

Avec requête préparée :

<?php

```
$req = $pdo->prepare('UPDATE exercicepdo SET login = :login1, password= :password1 WHERE ID = :ID1');

$req->execute(array(

    'login1' => $login,

    'password1' => $password,

    'ID1' => $ID

    ));

?>
```

Remarque : Le mot-clé WHERE est tout simplement indispensable. Il nous permet de dire à MySQL quelle entrée il doit modifier (sinon, toutes les entrées seraient affectées !). On se base très souvent sur le champ ID pour indiquer quelle entrée doit être modifiée.

8. Résumé

- INSERT INTO : ajout d'une entrée ;

Exemple :

```
$enregistrer=$pdo->prepare('INSERT INTO day(semaine) VALUES (:semaine5)');

$enregistrer->execute(array('semaine5'=>$_POST['semaine']));

echo("valeur enregistrée");

$enregistrer->closeCursor();
```

- UPDATE : modification d'une ou plusieurs entrées ;

Exemple :

```
$valeur=2 ;

$modifier=$pdo->prepare('UPDATE day SET lundi=:nom5 WHERE ID=:id5');

$modifier->execute(array('nom5'=>$_POST['nom'],'id5'=>$valeur));

 echo("valeur modifiée");

$modifier->closeCursor();
```

- DELETE : suppression d'une ou plusieurs entrées ;

Exemple :

```
$supprimer=$pdo->prepare('DELETE FROM day WHERE heure=:heure5');

$supprimer->execute(array('heure5'=>$_POST['heure']));

$supprimer->closeCursor();
```

- SELECT : Sélectionne une ou plusieurs entrées.

Exemple :

```
$detecter=$pdo->prepare('SELECT * FROM day WHERE heure=:heure5');

$detecter->execute(array('heure5'=>$_POST['heure']));
```

```
        while ($donne=$detecter->fetch())
    {
                $valeur=$donne['ID'];
        }
$detecter->closeCursor();
```