

Travail dirigé

IDE imposée : Netbeans
Utilisation de xdebug

Objectifs du TD :

Lire une documentation technique

Mettre en œuvre la classe PDO en utilisant des classes métier.

Prérequis : quelques notions de POO.

Partie 1 : Créer la base de données

Vous exécuterez le script crebase_client.sql que votre professeur vous a remis.

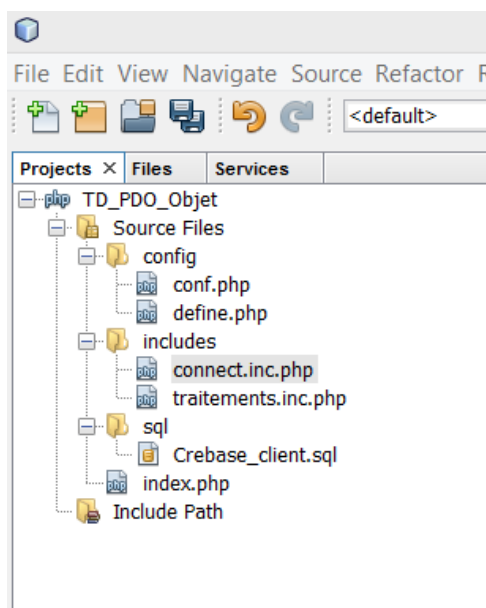
Vous utiliserez l'interface phpmyadmin pour exécuter ce script qui :

- ✓ Créera la base de données
- ✓ Créera les tables client et commande
- ✓ Remplira partiellement ces tables.


Partie 2 : Préparation du travail



Dans un premier temps, créer sous netbeans un projet TD_PDO_Objeto. Dont voici l'architecture :



Créer le fichier index.php. Netbeans fera l'essentiel, mais vous le modifierez ainsi :

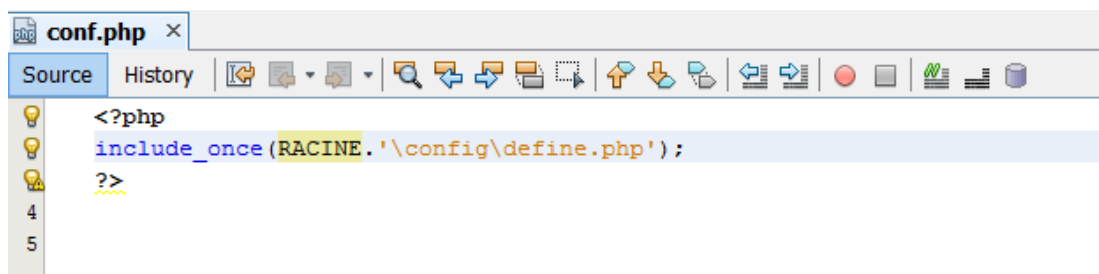


```

index.php x define.php x conf.php x
Source History
<!DOCTYPE html>
<html>
<head>
<title>TODO supply a title</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<?php
try {
define('RACINE', __DIR__);
include_once('config/conf.php');
include_once (INCLUDE_PATH . 'connect.inc.php');
$conn = connectionBd();
?>
<body>
<div>TODO write content</div>
</body>
<?php
} catch (Exception $ex) {
echo $ex->getMessage();
}
?>
</html>

```

Fichier conf.php

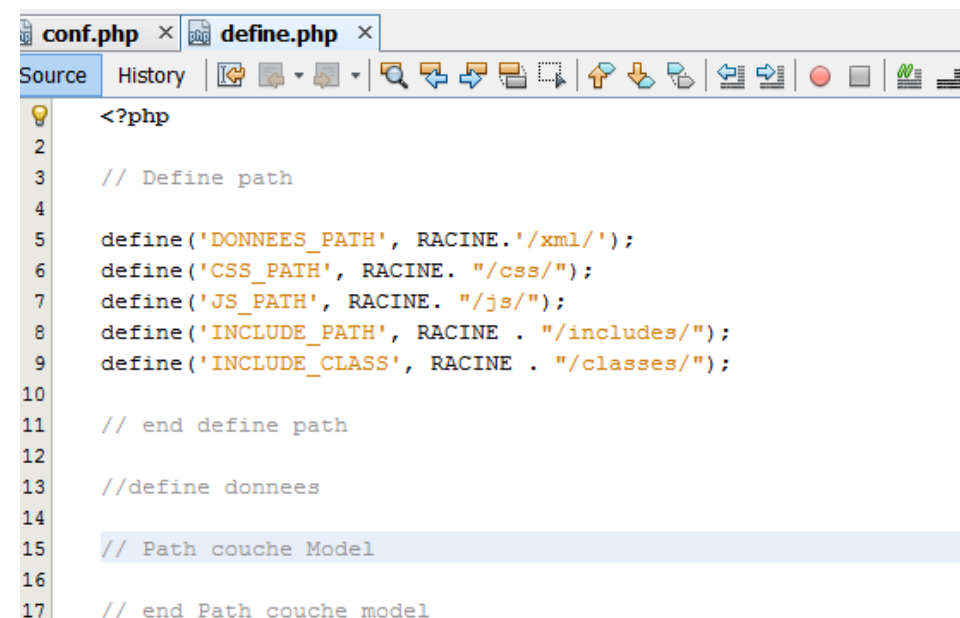


```

conf.php x
Source History
<?php
include_once(RACINE.'\config\define.php');
?>
4
5

```

Fichier define.php



```

conf.php x define.php x
Source History
<?php
// Define path
define('DONNEES_PATH', RACINE.'/xml/');
define('CSS_PATH', RACINE. "/css/");
define('JS_PATH', RACINE. "/js/");
define('INCLUDE_PATH', RACINE . "/includes/");
define('INCLUDE_CLASS', RACINE . "/classes/");
// end define path
//define donnees
// Path couche Model
// end Path couche model

```

Fichier connect.inc.php

```

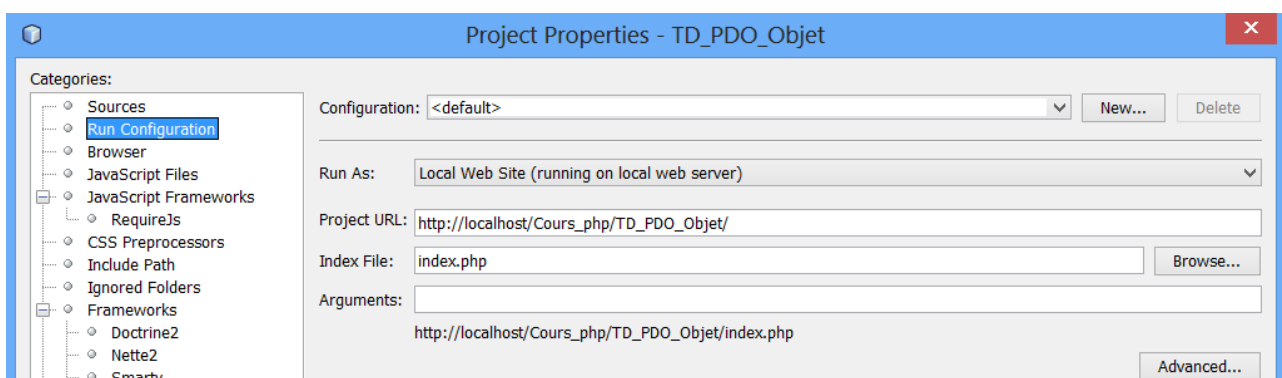
connect.inc.php x
Source History
<?php
2
3 function connectionBd() {
4     try {
5         $connect_str = "mysql:host=localhost;dbname=clicom";
6         $connect_user = "root";
7         $connect_pass = "";
8         $options[PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
9         return new PDO($connect_str, $connect_user, $connect_pass, $options);
10
11     } catch (PDOException $e) {
12         throw new Exception("Erreur à la connexion \n" . $e->getMessage());
13     }
14 }
15

```

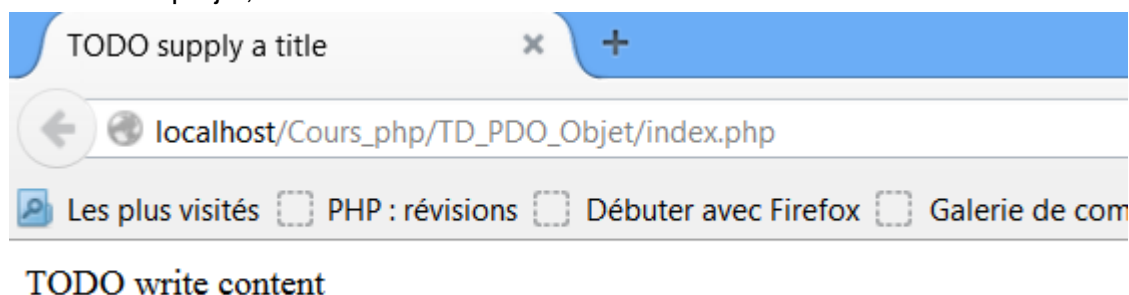


Pas top la connection root sans mot de passe, mais on corrigera cette faille au prochain TP. Ce n'est pas ici le but

N'oubliez pas de renseigner l'URL du projet dans les propriétés du projet :



Si vous exécutez le projet, vous devriez obtenir ceci :



Partie 3 : Récupération d'un enregistrement sous forme d'objet

Dans le fichier traitements.inc.php, créer la fonction suivante :

```
function recuperationUnClient($unObjetPdo, $id) {
    $sql = "select * from CLIENT where NOCLI=:pnocli";
    $ligne = $unObjetPdo->prepare($sql);
    $ligne->bindValue(':pnocli', $id, PDO::PARAM_INT);
    $ligne->execute();
    $unClient = $ligne->fetch(PDO::FETCH_OBJ);
    var_dump($unClient);
}
```

Vous remarquerez qu'on passe en paramètre la connexion ouverte et l'id du client.
 Appelez cette fonction depuis votre IDE Netbeans

```
<body>
    <div>
        <p><?php recuperationUnClient($conn, 123); ?></p>
    </div>
</body>
```

Et testez. Vous devriez obtenir ceci dans votre navigateur :

```
object(stdClass) [3]
  public 'NOCLI' => string '123' (length=3)
  public 'TITRECLIENT' => string 'Monsieur' (length=8)
  public 'NOMCLI' => string 'Tienun' (length=6)
  public 'PRENOMCLI' => string 'Jean' (length=4)
  public 'ADRESSERUE1CLI' => string '112, rue du D?part' (length=18)
  public 'ADRESSERUE2CLI' => null
  public 'CPCLI' => string '13000' (length=5)
  public 'VILLECLI' => string 'Marseille' (length=9)
  public 'TELCLI' => string '0404040404' (length=10)
```

On voit que l'on a bien récupéré un objet.



Rajoutez cette instruction après le var_dump dans la fonction recuperationUnClient et testez.

echo \$unClient->NOMCLI;

Qu'obtenez-vous ?

Partie 4 : Affichage de toutes les lignes récupérées

On va maintenant récupérer toute la table client et afficher les numéros, noms et prénoms .

Ecrire la fonction suivante dans le fichier traitements.inc.php:

```
function AfficheTousClients($unObjetPdo){
    $sql = "select * from CLIENT";
    $lignes= $unObjetPdo->query($sql);
    // on va configurer le mode objet pour la lisibilité du code
    $lignes->setFetchMode(PDO::FETCH_OBJ);
    while($unClient = $lignes->fetch()){
        echo "Numéro du client : " . $unClient->NOCLI . " Nom du client : " . $unClient->NOMCLI . " Prénom : " . $unClient->PRENOMCLI . "<br>";
    }
}
```

Et appelez cette fonction depuis votre page index.php, à la suite de l'appel précédent :

```
<div>
    <p><?php recuperationUnClient($conn, 123); ?></p>
    <p><?php AfficheTousClients($conn); ?></p>
</div>
```

Vous devriez obtenir ceci :

```
public 'TELCLI' => string '0404040404' (length=10)
```

Tienun

Numéro du client : 123 Nom du client : Tienun Prénom : Jean
Numéro du client : 176 Nom du client : Terrature Prénom : Julie
Numéro du client : 2343 Nom du client : Morizet Prénom : Patricia
Numéro du client : 2345 Nom du client : Nolapin Prénom : Jean
Numéro du client : 2376 Nom du client : Entete Prénom : Martel
Numéro du client : 2388 Nom du client : Entete Prénom : Martel
Numéro du client : 2389 Nom du client : DUMANS Prénom : Henriette
Numéro du client : 98802 Nom du client : Cerf Prénom : Paulette

Partie 5 : Création d'une classe client pour récupérer un enregistrement

Dans cette partie, au lieu de récupérer un enregistrement dans un objet anonyme, on va le récupérer dans un objet de la classe client

✓ Création de la classe client :

Dans le dossier classes, créer le fichier client.class.php qui contient la classe Client :

```
<?php
class Client {

    private $NOCLI;
    private $TITRECLIENT;
    private $NOMCLI;
    private $PRENOMCLI;
    private $ADRESSEURUE1CLI;
    private $ADRESSEURUE2CLI;
    private $CPCLI;
    private $VILLECLI;
    private $TELCLI;

    public function afficheUnClient(){
        return "Le client dont le numéro est égal à " . $this->NOCLI . " s'appelle " . $this->TITRECLIENT . " " . $this->NOMCLI . " " . $this->PRENOMCLI . "<br>";
    }
}
```



Vous ferez attention à la casse, les noms de colonne de la table et les noms des attributs privés de la classe doivent impérativement s'écrire de la même façon.

Pourquoi certains membres de la classe sont en privé et d'autres en public ?*

Dans le fichier traitements.inc.php, rajouter la fonction RecupUnObjetClient suivante :

```
function RecupUnObjetClient($unObjetPdo,$id){
    $sql = "select * from CLIENT where NOCLI=:pnocli";
    $lignes = $unObjetPdo->prepare($sql);
    $lignes->bindValue(':pnocli', $id, PDO::PARAM_INT);
    $lignes->execute();
    $unClient = $lignes->fetchObject('Client');
    var_dump($unClient);
    return $unClient;
}
```

Cette fonction va :

- ✓ aller chercher un client dont l'id est passé en paramètre.
- ✓ instancier un objet de la classe Client et va mapper les champs de l'enregistrement retourné sur les attributs qu'elle est capable de reconnaître dans la classe Client.
- ✓ retourner un objet de la classe client, que l'on va récupérer et exploiter dans la page index.php.



- ✓ Si une propriété de la classe ne correspond à aucun champ, elle aura la valeur null
- ✓ Si un champ de la table ne correspond à aucune propriété de la classe, une propriété publique sera créée et initialisée avec la valeur du champ
- ✓ Toutes les valeurs issues de la table sont de type string même si elles concernent un entier

Rajouter le code suivant dans la page index.php :

```
<?php RecupUnObjetClient($conn, 123); ?>
<p><?php AfficheTousClients($conn); ?></p>
<p><?php $unClient=RecupUnObjetClient($conn,123); ?></p>
<p><?php echo $unClient->afficheUnClient() ?></p>
</div>
/body>
```

Et vous devriez arriver au résultat suivant en exécutant le projet :

 Numéro du client : 98802 Nom du client : Cerf Prénom : Paulette

```
object(Client)[4]
  private 'NOCLI' => string '123' (length=3)
  private 'TITRECLIENT' => string 'Monsieur' (length=8)
  private 'NOMCLI' => string 'Tienun' (length=6)
  private 'PRENOMCLI' => string 'Jean' (length=4)
  private 'ADRESSERUE1CLI' => string '112, rue du D?part' (length=18)
  private 'ADRESSERUE2CLI' => null
  private 'CPCLI' => string '13000' (length=5)
  private 'VILLECLI' => string 'Marseille' (length=9)
  private 'TELCLI' => string '0404040404' (length=10)
```

Le client dont le numéro est égal à 123 s'appelle Monsieur Tienun Jean



Dans la page index.php, rajouter la ligne suivante :

```
<p><?php echo $unClient->afficheUnClient() ?></p>
<p><?php echo $unClient->NOMCLI ?></p>
</div>
```

Que se passe-t-il ?

Partie 6 : Récupération de plusieurs enregistrements dans un tableau d'objets

Dans la partie précédente, on a récupéré le résultat d'une requête dans un objet d'une classe donnée. L'enregistrement a été mappé sur l'objet.

Dans cette partie, on va récupérer plusieurs enregistrements dans un tableau d'objet grâce à la méthode `fetchAll` de la classe PDO.

Vous allez créer la fonction `RecupPlusieursObjetsClient(...)` dans le fichier `traitements.inc.php`

```
function RecupPlusieursObjetsClient($unObjetPdo) {  
    $sql = "select * from CLIENT";  
    $lignes = $unObjetPdo->query($sql);  
    // on va configurer le mode objet pour la lisibilité du code  
    $tableauClients= $lignes->fetchAll(PDO::FETCH_CLASS, 'Client');  
    var_dump($tableauClients);  
}
```

On remarque l'utilisation de la méthode `fetchAll` qui accepte en paramètre la constante de classe de la classe PDO (`PDO::FETCH::CLASS`) et le nom de la classe. On verra ci-dessous que cette méthode renvoie un tableau d'objets de la classe passée en deuxième paramètre, ici `Client`.

Dans le fichier `index.php`, rajouter l'appel à la méthode et exécutez :

```
<p><?php //echo $unClient->NOMCLI ?></p>  
<p><?php RecupPlusieursObjetsClient($conn); ?></p>  
</div>  
body</body>
```

Vous devriez obtenir le résultat suivant :

Le client dont le numero est egal a 123 s appelle Monsieur Tienun Jean

```
array (size=8)
0 =>
  object(Client) [2]
    private 'NOCLI' => string '123' (length=3)
    private 'TITRECLIENT' => string 'Monsieur' (length=8)
    private 'NOMCLI' => string 'Tienun' (length=6)
    private 'PRENOMCLI' => string 'Jean' (length=4)
    private 'ADRESSERUE1CLI' => string '112, rue du D?part' (length=18)
    private 'ADRESSERUE2CLI' => null
    private 'CPCLI' => string '13000' (length=5)
    private 'VILLECLI' => string 'Marseille' (length=9)
    private 'TELCLI' => string '0404040404' (length=10)
1 =>
  object(Client) [5]
    private 'NOCLI' => string '176' (length=3)
    private 'TITRECLIENT' => string 'Madame' (length=6)
    private 'NOMCLI' => string 'Terrature' (length=9)
    private 'PRENOMCLI' => string 'Julie' (length=5)
    private 'ADRESSERUE1CLI' => string 'R?sidence Mermoz' (length=16)
    private 'ADRESSERUE2CLI' => string '1234 Boulevard des Aviateurs' (length=28)
    private 'CPCLI' => string '14000' (length=5)
    private 'VILLECLI' => string 'Caen' (length=4)
    private 'TELCLI' => string '0202020202' (length=10)
2 =>
  object(Client) [6]
    private 'NOCLI' => string '2343' (length=4)
    private 'TITRECLIENT' => string 'Mademoiselle' (length=12)
    private 'NOMCLI' => string 'Morizet' (length=7)
    private 'PRENOMCLI' => string 'Patricia' (length=8)
    private 'ADRESSERUE1CLI' => string 'Hameau de Pau' (length=13)
    private 'ADRESSERUE2CLI' => string '23 Boulevard du Lvc' (length=19)
```

On voit bien qu'il s'agit d'un tableau (array) d'objets de la classe Client.

On va maintenant parcourir ce tableau grâce à la boucle foreach.

Pour cela créez une fonction AfficheTousClientsObjet dans le fichier traitements.inc.php :

```
function AfficheTousClientsObjet($unObjetPdo) {
    $tableauClients = RecupPlusieursObjetsClient($unObjetPdo);
    echo "<p>liste des clients <br>";
    foreach ($tableauClients as $unObjetClient) {
        echo $unObjetClient->afficheUnClient() . "<br>";
    }
    echo "</p>";
}
```

On voit que cette fonction récupère dans un tableau d'objets tous les objets clients renvoyés par la fonction RecupPlusieursObjetsClient que l'on a modifié ainsi :


```

2
3
4 function RecupPlusieursObjetsClient($unObjetPdo) {
5     $sql = "select * from CLIENT";
6     $lignes = $unObjetPdo->query($sql);
7     // on va configurer le mode objet pour la lisibilité du code
8     if ($lignes->rowCount() > 0) {
9         $tableauClients = $lignes->fetchAll(PDO::FETCH_CLASS, 'Client');
10        return $tableauClients;
11        //var_dump($tableauClients);
12    } else {
13        throw new Exception('Aucun client trouvé');
14    }
15 }

```

On voit que cette fonction retourne un tableau d'objets de la classe client ou déclenche une exception si aucun enregistrement trouvé.

Modifiez la page index.php pour afficher tous les clients :

```

<p><?php //echo $unClient->NOMCLI ?></p>
<p><?php AfficheTousClientsObjet($conn); ?></p>

```

Exécutez le projet. Vous devriez obtenir ceci :

liste des clients

Le client dont le numéro est égal à 123 s'appelle Monsieur Tienun Jean

Le client dont le numéro est égal à 176 s'appelle Madame Terrature Julie

Le client dont le numéro est égal à 2343 s'appelle Mademoiselle Morizet Patricia

Le client dont le numéro est égal à 2345 s'appelle Monsieur Nolapin Jean

Le client dont le numéro est égal à 2376 s'appelle Monsieur Entete Martel

Le client dont le numéro est égal à 2388 s'appelle Monsieur Entete Martel

Le client dont le numéro est égal à 2389 s'appelle Madame DUMANS Henriette

Le client dont le numéro est égal à 98802 s'appelle Madame Cerf Paulette



Modifiez la requête de la méthode RecupPlusieursObjetsClient :

\$sql = "select * from CLIENT where nocli=1";

Que se passe-t-il ?

Enfin pour finir :

Que signifie cette instruction ?

\$requete->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, '???');

Mettez là en œuvre et fournir une preuve de ce qu'elle fait lors de son exécution

