

Php : l'objet PDO

Formation

Développeur Web et Web Mobile

Module : 04

Développer la partie back-end d'une application web

Séquence : 03

Développer une interface utilisateur web dynamique

Séance : 01


Développer des scripts serveurs

Libellé réduit:	DWWM
Type de document:	Ressource
Version:	1
Date de validation:	30/12/2016
Date de mise à jour:	02/12/2022

Sommaire

Sommaire

I	Introduction	1
II	Gestion de PDO par php	1
III	Les drivers pour les différentes bases de données	2
III.1	Mysql	2
III.2	Sql serveur	2
III.3	Source ODBC	2
IV	Connexion à la base de données depuis Php	3
IV.1	Principe	3
V	Gestion des erreurs	4
VI	Envoi de requêtes	5
VI.1	La méthode query() et les requêtes simples	5
VI.2	La méthode execute() et les requêtes préparées	5
	Préparation de la requête	6
	Exécution de la requête	6
VII	Récupération des données	8

	Auteur	Centre de Créteil	Formation	Date Mise à jour	Page 1
	Didier Bonneau	GRN 164	DWWM	02/12/2022	cours php - l'api pdo.docx

I INTRODUCTION

La hiérarchie de classes PDO (**P**HP **D**ata **O**bjects) permet l'utilisation d'une base de données à partir de Php en faisant abstraction du moteur de SGBD utilisé.

L'API PDO contient trois classes d'objet :

La classe PDO : connexion à la base de données

La classe PDOStatement : traitement de la réponse

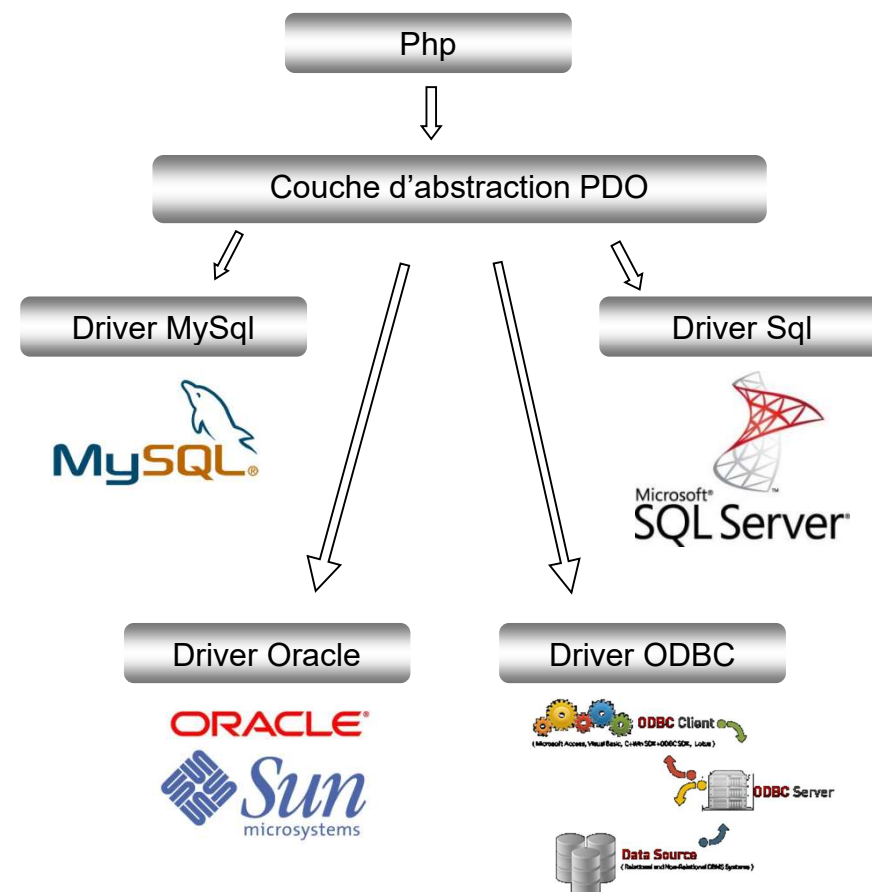
La classe PDOException : gestion des erreurs

L'API PDO va rendre le code totalement indépendant du type de la base de données. En effet, selon le SGBD utilisé, la syntaxe des requêtes faites peut varier. Avec Pdo, une couche d'abstraction va être mise en place évitant de réécrire du code en cas de changement de base.

L'avantage de travailler en objet est que ces classes peuvent être étendues pour s'adapter à nos propres besoins.

II GESTION DE PDO PAR PHP

Par défaut dans Xampp, Php gère toute la hiérarchie de classe PDO. Vous pouvez vérifier dans le fichier « php.ini » que la ligne « extension=pdo_mysql » n'est pas en commentaire (pas de point-virgule devant).



III LES DRIVERS POUR LES DIFFERENTES BASES DE DONNEES

Pour pouvoir se connecter à différentes bases de données par PDO, Php a besoin de drivers spécifiques à chaque SGBD.

III.1 MYSQL

Pour MySql, le driver est déjà présent dans le répertoire « xampp\php\ext » de Xampp. Il est sous forme d'une DLL qui porte le nom de « php_pdo_mysql.dll ».

III.2 SQL SERVEUR

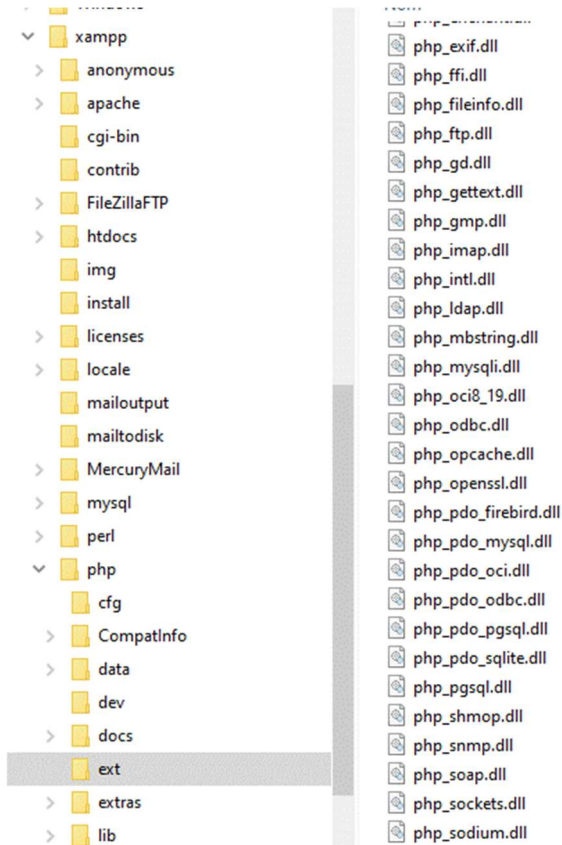
Pour Sql Server le problème est un peu plus délicat. Le driver dépend de la version de Php, le fait qu'il soit « Thread Safety » ou non et enfin de la version du compilateur C++ utilisé.

Pour connaître toutes ces informations, lancez le localhost de Xampp puis allez sur phpinfo().

Normalement le driver devrait être « php_pdo_sqlsrv_53_ts_vc6.dll ». Voir « <http://msdn.microsoft.com/en-us/library/cc296170.aspx> »

III.3 SOURCE ODBC

Pour une source ODBC le driver est « php_pdo_odbc.dll ». Il est présent par défaut.



IV CONNEXION A LA BASE DE DONNEES DEPUIS PHP

IV.1 PRINCIPE

La classe PDO permet la connexion à la base. Elle possède un constructeur dont le prototype est indiqué à droite.

Le principe est de créer un nouvel objet PDO et de récupérer sa référence :

```
$refPdo = new PDO($dsn, $userName, $password,
                  $driverOptions);
```

Les paramètres sont :

- **\$dsn**
Chaine de caractères spécifiant le type de la base, l'adresse du serveur et le nom de la base de données et l'encodage.
\$dsn pour mySql

'mysql:host=adr_serveur;dbname=nom_base;charset=xxx'
- **\$userName**
Chaine de caractères spécifiant le nom de l'utilisateur
- **\$password**
Chaine de caractères spécifiant le mot de passe de l'utilisateur
- **\$driverOptions**
C'est un tableau permettant de positionner certain options du driver (gestion des erreurs, type de réponse des interrogations, ...)
(voir documentation)

Constructeur de la classe PDO

```
PDO::__construct() ( string $dsn [, string  
$username [, string $password [, array  
$driver_options ]]] )
```

Exemple s :

\$dsn pour mySql sur localhost avec la base Personne

```
'mysql:host=localhost;dbname=Personne'
```

\$username par défaut pour wamp

```
'root'
```

\$password par défaut pour wamp


```
" (vide) ou omis
```

Connexion sur wamp par défaut à la base Personne

```
$connect = new  
PDO('mysql:host=localhost;dbname=Personne', 'root', '')
```

Il faut préciser l'encodage des caractères dans le paramètre \$dsn

```
'mysql:host=localhost;dbname=Personne;charset=utf8'
```

	Auteur	Centre de Créteil	Formation	Date Mise à jour	Page 3
	Didier Bonneau	GRN 164	DWWW	02/12/2022	cours php - l'api pdo.docx

V GESTION DES ERREURS

La classe PDO est capable de lever une Exception (erreur) si la connexion ne peut être réalisée.

Pour prendre en compte cette exception, il faut entourer l'instruction de création de la connexion par un bloc « try / catch ».

L'objet « exception » récupéré dans le bloc « catch » est de type « PDOException ». On peut invoquer les méthodes « errorCode() et errorInfos() » pour avoir le détail de cette exception.

Afin de gérer les exceptions d'autres natures (erreur dans un ordre SQL, erreur d'insertion d'un champ NULL non autorisé, ...) il est impératif de modifier les attributs par défaut de la connexion. Ceci se fait soit en utilisant le troisième paramètre lors de la connexion, soit en utilisant la méthode setAttribute() de l'objet PDO :

- `PDO::ATTR_CASE`: force les noms de colonnes à une casse particulière.
 - `PDO::CASE_LOWER` : force les noms de colonnes à être en minuscules.
 - `PDO::CASE_NATURAL` : laisse les noms de colonnes inchangées.
 - `PDO::CASE_UPPER` : force les noms de colonnes à être en majuscules.
- `PDO::ATTR_ERRMODE` : rapport d'erreurs.
 - `PDO::ERRMODE_SILENT` : assigne simplement les codes d'erreur.
 - `PDO::ERRMODE_WARNING`: émet une alerte [E_WARNING](#).
 - `PDO::ERRMODE_EXCEPTION` : émet une [exception](#).

(extrait du manuel Php)

Exemple de code


```
try {
    $connect = new
        PDO('mysql:host=localhost;dbname=Personne;
            charset=utf8', 'root', '')
    ...
    ...
} catch ( PDOException $err ) {
    echo $err->getMessage() ;
}
```

Définition de setAttribute() :

```
bool PDO::setAttribute ( int $attribute , mixed
    $value )
```

Exemple pour gérer les exceptions :

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

	Auteur	Centre de Créteil	Formation	Date Mise à jour	Page 4
	Didier Bonneau	GRN 164	DWWW	02/12/2022	cours php - l'api pdo.docx

VI ENVOI DE REQUETES

L'envoi de requêtes Sql vers la base de données peut se faire de deux façons différentes.

1. La première est une méthode de base qui consiste à créer l'ordre Sql simple sous forme d'une chaîne de caractères puis de l'exécuter par la méthode « query() ».
2. La deuxième méthode est plus efficace, rapide et plus sécurisée contre les infections Sql. Elle consiste à préparer la requête que l'on veut envoyer et de l'exécuter par la méthode « execute() » en ayant au préalable lié les paramètres de la requête avec les valeurs voulues.

VI.1 LA METHODE QUERY() ET LES REQUETES SIMPLES

Cette méthode peut être prise pour n'importe quel ordre Sql vers la base :

SELECT, INSERT, UPDATE, DROP, DELETE, CALL, ...

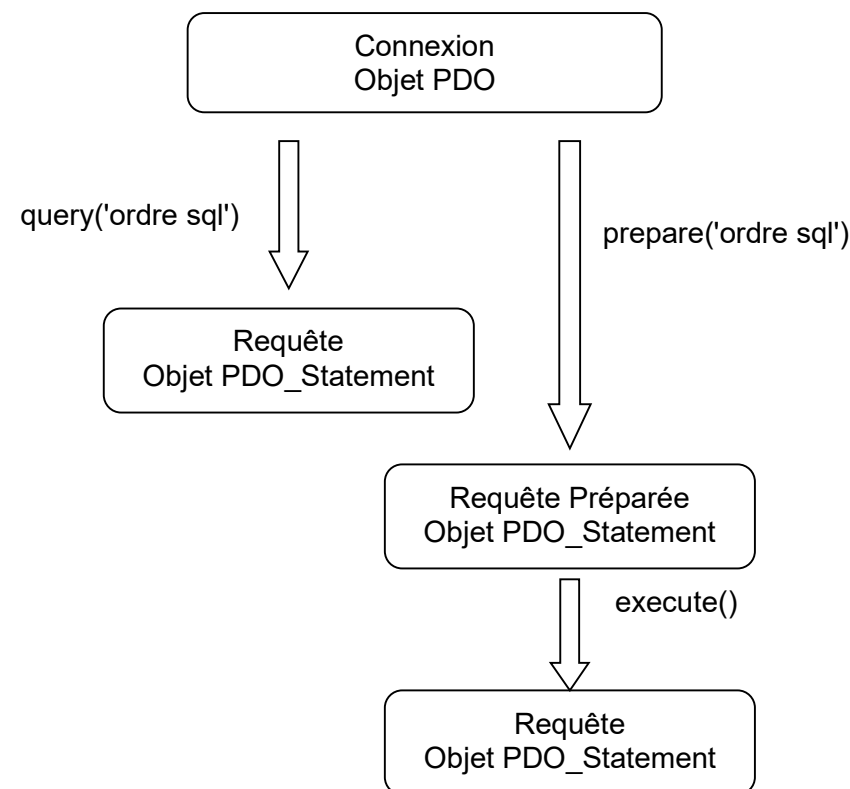
L'ordre Sql sera une chaîne de caractère fixe. Attention aux simples cotes (AltGr 7) sur certain ordre Sql.

Le retour de cette méthode est un objet de type « PDO_Statement » s'il n'y a pas d'erreur de syntaxe ou autre dans votre ordre Sql. Dans le cas contraire, le retour est de type booléen à False.

VI.2 LA METHODE EXECUTE() ET LES REQUETES PREPAREES

La seule différence est que l'on peut exécuter plusieurs fois cette même requête en modifiant certaines valeurs entre deux exécutions.

Les valeurs réelles des différents paramètres seront fournies au moment de l'exécution.



Exemple de requête simple :

```

$req = 'SELECT * FROM personnes WHERE
      prenom= \'paul\'';
$res = $pdo->query($req);
  
```

Préparation de la requête

Les paramètres de la requête peuvent être fournis de deux façons différentes :

- Les paramètres non nommés.
- Les paramètres nommés.

Les paramètres non nommés

Pour indiquer les paramètres non nommés dans la requête, il suffit de mettre des marqueurs « ? » là où c'est nécessaire dans la chaîne de requête.

Les valeurs réelles de ces paramètres devront être données dans l'ordre d'apparition des marqueurs.

Les paramètres nommés

Pour une requête nommée, on nomme les paramètres avec des noms en les faisant précéder du signe « : ».

Dans ce cas, les valeurs réelles peuvent être fournies dans n'importe quel ordre.

Exécution de la requête

Une fois la requête préparée, il faut lier les valeurs réelles avec les paramètres. Là encore deux possibilités s'offrent à nous :

- Utilisation de la méthode « bindParam() » puis « execute() » sans paramètres
- Passage d'un tableau simple de « valeurs » ou associatif « paramètre/valeur » à la méthode « execute() »

Exemple de requêtes préparées :

Paramètre non nommé

```
$req = 'SELECT * FROM personnes WHERE
      prenom = ?';
$resp = $pdo->prepare($req);
```

? : marqueur

Paramètre nommé

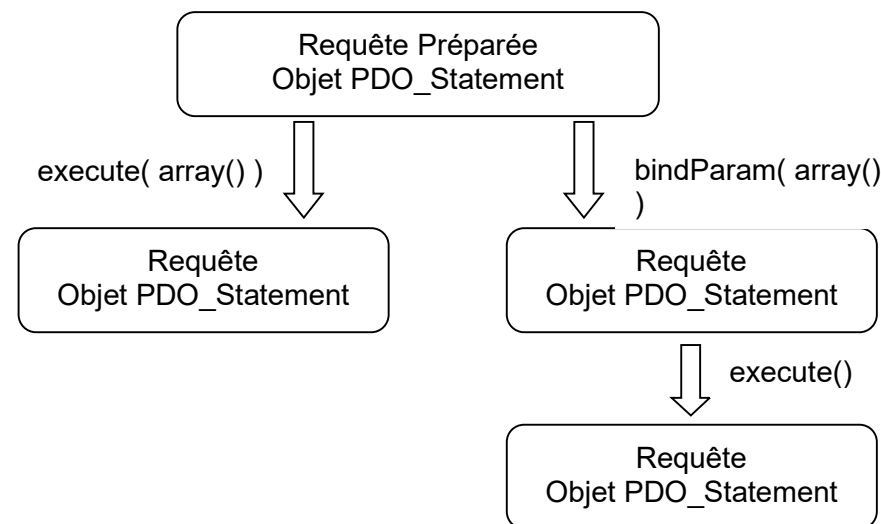
```
$req = 'SELECT * FROM personnes WHERE
      prenom = :prenom';
$resp = $pdo->prepare($req);
```

:nom
Nom du paramètre

\$req : chaîne de la requête préparée -> de type String

\$pdo : connexion à la base de données -> de type PDO

\$resp : réponse de la requête -> de type PDO_Statement



La méthode bindParam()

Cette méthode permet de lier un paramètre à une variable ou valeur réelle. Il faut donc autant d'appels à « bindParam() » que de paramètres présents dans la requête.

Il est possible de préciser le type de données, éventuellement sa longueur (chaîne) et s'il s'agit d'un paramètre en entrée ou sortie ou les deux.

Il existe aussi la méthode « bindValue() » qui associe une valeur à un paramètre.

Une fois les paramètres liés aux valeurs ou variables, il faut exécuter la méthode « execute() » sans aucun paramètres.

Le tableau simple ou associatif

Pour une requête non nommée, il suffit de passer un tableau simple à la méthode « execute() » en respectant bien l'ordre des marqueurs.

```
execute(array($var1, $var2, ...))
```

Pour des paramètres nommés, le tableau associatif passé à « execute() » doit avoir les clés identiques aux noms des paramètres nommés et comme valeurs les variables ou valeurs réelles.

```
execute(array(' :param1' => $var1, ' :param2' => $var2, ...))
```

Définition de la méthode bindParam()

```
bool PDOStatement::bindParam ( mixed
    $parameter, mixed &$variable [, int $data_type =
    PDO::PARAM_STR [, int $length [, mixed
    $driver_options ]]] )
```

(extrait manuel Php)

Exemples

avec bindParam() et execute()

```
$resp->bindParam(1, $prenom, PDO::PARAM_STR, 25);
// avec marqueur
ou
$resp->bindParam(':prenom', $prenom,
    PDO::PARAM_STR, 25); // avec param nommé
$resp->execute();
```

avec execute() et tableau associatif

```
$resp->execute(array($prenom));
// avec marqueur
ou
$resp->execute(array(':prenom'=>$prenom));
// avec param nommé
```

VII RECUPERATION DES DONNEES

Une fois la requête exécutée, il faut traiter la réponse (le PDO_Statement) pour récupérer les données.

Plusieurs méthodes existent :

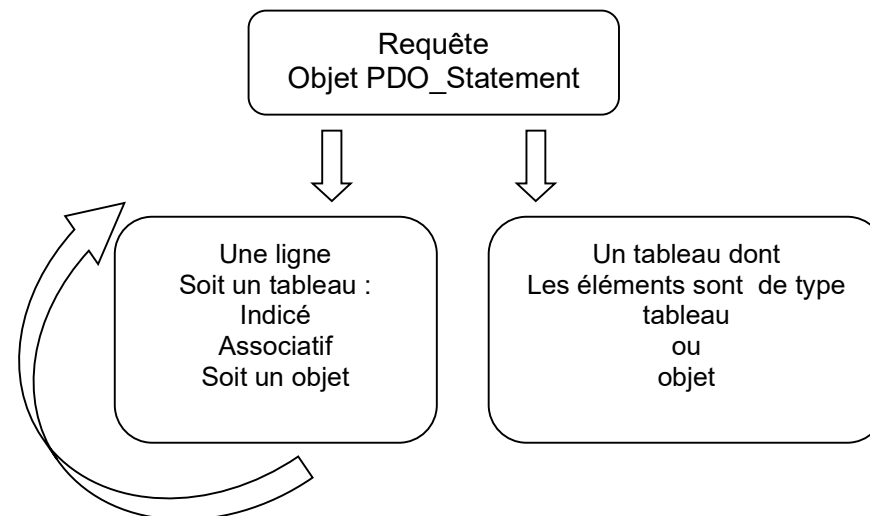
- Itérer sur l'objet PDO_Statement pour récupérer à chaque tour de boucle une ligne de la réponse sous forme :
 - d'un tableau indicé ou associatif (ou les deux)
 - d'un objet
- Obtenir directement l'ensemble des lignes de la réponse sous forme :
 - d'un tableau de tableau
 - d'un tableau d'objet

VII.1 ITERATION SUR L'OBJET PDO_STATEMENT

Ceci se fait en utilisant la méthode « fetch() » sur l'objet PDO_Statement. Le paramètre « fetch_style » détermine sous quel forme la ligne va être renvoyée. Les principales valeurs possibles sont :

- `PDO::FETCH_NUM` : retourne un tableau
- `PDO::FETCH_ASSOC` : retourne un tableau associatif le nom des clés du tableau correspondent aux noms de la colonne de la requête
- `PDO::FETCH_BOTH` (défaut) : retourne un tableau indexé et associatif
- `PDO::FETCH_OBJ` : retourne un objet anonyme avec les noms de propriétés correspondent aux noms des colonnes de la requête
- `PDO::FETCH_CLASS` : retourne une nouvelle instance de la classe demandée

Pour les deux derniers cas, il est possible d'utiliser la méthode « fetchObject() ».



Définition de la méthode `fetch()`

```
mixed PDOStatement::fetch ([ int $fetch_style [,  
int $cursor_orientation = PDO::FETCH_ORI_NEXT [,  
int $cursor_offset = 0 ]]] )
```

(extrait manuel Php)

Définition de la méthode `fetchObject()`

```
mixed PDOStatement::fetchObject ([ string  
$class_name = "stdClass" [, array $ctor_args ] ] )
```

(extrait manuel Php)

VII.2 OBTENTION D'UN TABLEAU

Utiliser la méthode « `fetchAll()` ». Cette méthode renvoie un tableau contenant toutes les lignes du résultat de la requête.


Le paramètre « `fetch_Style` » peut prendre les valeurs suivantes :

- `PDO::FETCH_COLUMN` : Retourne le numéro de la colonne demandée (indexée à partir de 0)
- `PDO::FETCH_CLASS` : Retourne une instance de la classe désirée. Les colonnes sélectionnées sont liées aux attributs de la classe. Pour tout noms de colonnes ou alias de la requête ne correspondant pas à des noms de propriété de la classe, la méthode magique « `__set()` » de la classe est appelée avant le constructeur.
- `PDO::FETCH_FUNC` : Retourne la valeur de retour de la fonction précisée en deuxième argument. Les paramètres de la fonction doivent correspondre aux noms de colonnes ou alias de la requête.

Définition de la méthode `fetchAll()`

```
array PDOStatement::fetchAll ([ int $fetch_style [,  
mixed $fetch_argument [, array $ctor_args =  
array() ]]] )
```

(extrait manuel Php)

	Auteur	Centre de Créteil	Formation	Date Mise à jour	Page 9
	Didier Bonneau	GRN 164	DWWW	02/12/2022	cours php - l'api pdo.docx

VIII EXEMPLE COMPLET DE CODE

Soit une petite base de données mySql appelée « tp_personne » servant à enregistrer des personnes. Seuls les noms, prénoms seront mémorisés. Les deux champs ne peuvent pas être nuls.

Pour des manipulations sur les exceptions, un trigger sur le champ « nom » sera créer permettant de ne pas accepter des noms inférieur à trois caractères :

VIII.1 CREATION DE LA BASE

Sous phpMyAdmin créez la base de données, avec le code donné à gauche.

Ajouter quelques données à cette base.

Structure de la table « personnes »

```
CREATE TABLE IF NOT EXISTS personnes
( Id int(11) NOT NULL AUTO_INCREMENT,
  nom varchar(50) NOT NULL,
  prenom varchar(50) NOT NULL,
  PRIMARY KEY (id))
ENGINE=InnoDB DEFAULT
CHARSET=latin1 AUTO_INCREMENT ;
```

Définition du Trigger »

```
CREATE
  DEFINER = 'root'@'localhost'
  TRIGGER tp_personne.testNom
  BEFORE INSERT
  ON tp_personne.personnes
  FOR EACH ROW
  BEGIN
    IF (SELECT LENGTH(NEW.nom)) < 3 THEN
      SET NEW.nom = null;
    END IF;
  END
```

VIII.2 EXTRAIT DES SOURCES PHP

Dans ce qui suit, seuls des extraits de sources seront donnés.

Connexion à la base

```
// création d'un objet PDO
$PDO_connect = new PDO
('mysql:host=localhost;dbname=tp_personne', 'root', '');
// positionnement des attributs pour la gestion des erreurs
$PDO_connect->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
```

Recherche d'une personne

Extrait du contrôleur des actions après recherche du nom « didier » :

```
switch($action){
    case 'find':
        // récupération des paramètres venant du formulaire
        $nom = $_POST['nom'];
        $prenom = $_POST['prenom'];
        $reqSql = 'SELECT * FROM personne WHERE nom=\"'. $nom . '\"';
        $pdoStat_pers = $pdo_connect->query($reqSql);
        if ($pdoStat_pers->rowCount() == 1){
            // récupération des résultats -> array()
            $stab_pers = $pdoStat_pers->fetch();
```

Par défaut le « fetch_style » prend la valeur « PDO::FETCH_BOTH ». On obtient donc un tableau dans lequel les valeurs peuvent être lues grâce aux indices (0, 1, 2) ou par les clés ('id', 'nom', 'prenom')

Explications

- Nom du driver : mysql
- Adresse du serveur : localhost
- Nom de la base de données : tp_personne
- Nom de l'utilisateur : root
- Mot de passé de l'utilisateur : vide

Amélioration : utilisation de la méthode « quote() » de PDO

```
$reqSql = 'SELECT * FROM personne WHERE nom =
$_PDO_connect->quote( $nom );
```

Résultat par var_dump(\$stab_pers)

```
array
  'id' => string '1' (length=1)
  0 => string '1' (length=1)
  'nom' => string 'didier' (length=6)
  1 => string 'didier' (length=6)
  'prenom' => string 'didier' (length=6)
  2 => string 'didier' (length=6)
```