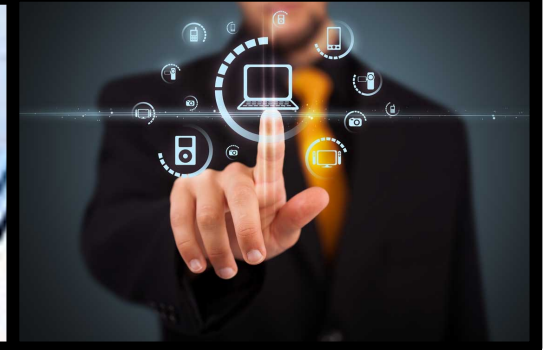
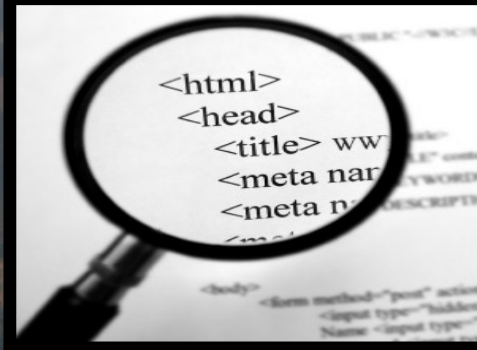


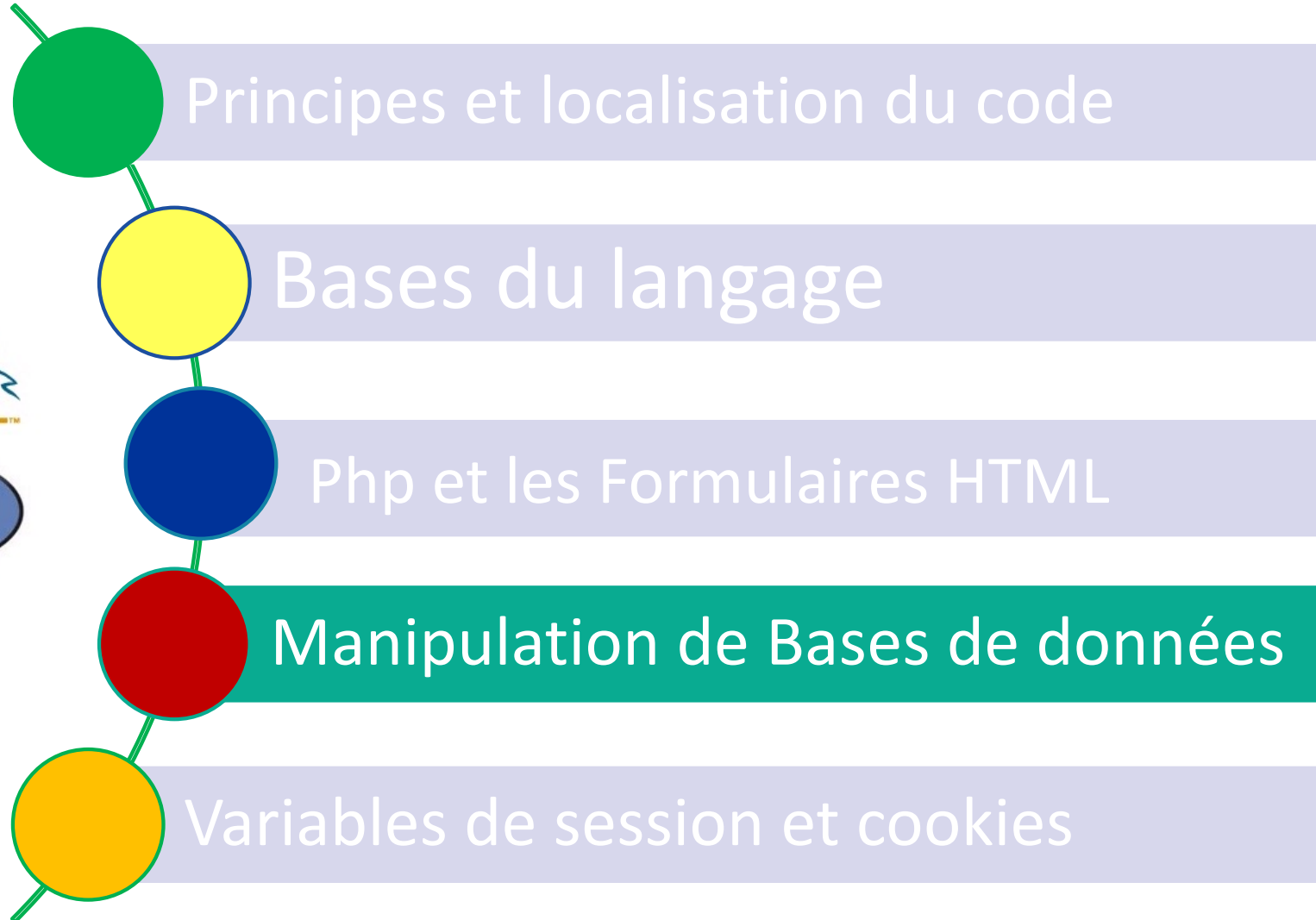
UNIVERSITE DE CORSE
Licence SPI 2ème année
Option Informatique

UE Programmation WEB Back End
CH1.4 – Manipulation de Bases de données



Evelyne VITTORI
vittori_e@univ-corse.fr

CH1- Fondamentaux Php



CH1.4 – Manipulation de BD

Objectifs

- Découvrir les API d'interaction avec une BD MySQL
- Etudier l'API PDO
 - Etapes de mise en œuvre
 - Mises à jour
 - Recherche de données
- Exercice d'application:
TPN°1 (non à rendre)



Manipuler une base de données avec php

Trois APIs php permettent d'accéder à une BD:

- **mysql**

Obsolète donc déconseillée ...

- fonctions d'accès procédural à une base de données MySQL

- **mysqli**

- classes permettant la manipulation « orientée objet » d'une BD MySQL

- **Php Data Objects (PDO)**

- manipulation OO d'une BD d'un SGBD quelconque: MySQL, SQLite, PostgreSQL,...

Notre approche dans ce cours

Etapes de manipulation d'une base de données

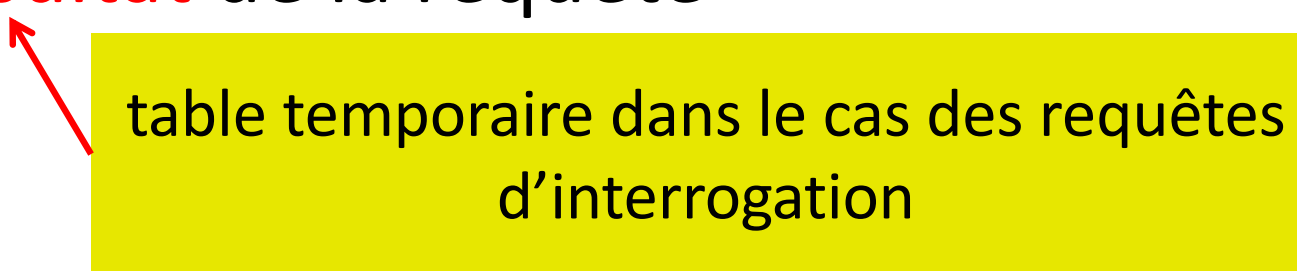
1. Établir la **connexion** à la Base de données
2. Demander au serveur d'exécuter une **requête** SQL
(Recherche, Mise à jour...)
3. Récupérer le **résultat** de la requête


table temporaire dans le cas des requêtes d'interrogation
4. **Fermer** la connexion

Classes de l'API PDO

PDO

```
exec(requete:String) : int  
prepare(requete:String) : PDOStatement  
query(requete:String) : PDOStatement
```

PDOStatement



```
columnCount() : int  
rowCount() : int  
fetch(...): tuple (tableau ou objet)  
execute(paramètres)
```

Objet Résultat
d'une requête

Connexion à une base de données

- La connexion à une base de données est réalisée par la création d'un objet de la classe

PDO `$bdd = new PDO (dsn, user, password) ;`

- constructeur à 3 paramètres:

- le Data Source Name (dsn): désigne le sgbd (« pilote ») et la base de données considérée
- le nom d'utilisateur (*root* par défaut)
- le mot de passe (chaine vide par défaut)

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

Exécution d'une requête SQL

- 3 types d'exécution :

de type String

- **Mises à jour:** `exec($requete)` de la classe PDO

- Pour les requêtes de maj du schéma: CREATE TABLE, ALTER TABLE, DROP TABLE
- Pour les requêtes de maj des données: INSERT, UPDATE, DELETE qui retournent un entier (nombre de tuples traités)

- **Interrogation:** `query($requete)` de la classe PDO


Pour les requêtes (SELECT) qui retournent un **résultat (de type PDOStatement)** (tuples du résultat de la requête)

- **Requêtes paramétrées:** `prepare($requete)` de la classe PDO puis `execute()` de PDOStatement

Requêtes de mise à jour du schéma

- la méthode **exec** () de la classe **PDO** prend comme argument une chaîne (**String**) indiquant la requête SQL à exécuter :

```
$bdd = new  
PDO('mysql:host=localhost;dbname=etudiants', 'root', '');  
  
$requete="CREATE TABLE t_prof(pro_id VARCHAR(25),  
                                pro_salaire FLOAT)" ;  
  
$bdd->exec($requete);
```



La méthode renvoie 0 dans le cas d'une requête de création de table

Requête de mise à jour des données

- la méthode **exec** () renvoie le nombre de lignes impactées par la requête

```
$requete="INSERT INTO t_prof VALUES ('Titi',2000)" ;  
$nb=$bdd->exec($requete) ;  
echo "<h3> Resultat de la requete ".$nb . "<h3>";
```

Il est aussi possible d'utiliser la méthode query de la classe PDO qui renvoie un objet PDOStatement

Requête de mise à jour à partir d'une saisie dans un formulaire

```
<form method="post" action="ajoutProf.php">  
Nom : <input type="text" name="nom"><br>  
Salaire : <input type="text" name="salaire"><br><br>  
<input type="submit" value="Enregistrer">  
</form>
```

Tableau \$_POST

nom	salaire
Toto	2000

Nouveau professeur

Nom :
Salaire :

Enregistrer

Requête de mise à jour à partir d'une saisie dans un formulaire

```
//Récupération des données du formulaire
$nom=$_POST["nom"];
$salaire=$_POST["salaire"];

//Expression de la requête
$requete="INSERT INTO t_prof VALUES ('$nom',$salaire) " ;
$nb=$bdd->exec($requete);

echo "<h3> Resultat de la requete ".$nb . "<h3>";
```

Requêtes d'Interrogation

- **query()** de la classe **PDO** prend comme argument une chaîne (**String**) indiquant la requête SQL à exécuter et renvoie un résultat de type PDOStatement :

```
$requete="SELECT et_nom, etu_note FROM  
          t_etudiant" ;  
$result=$bdd->query($requete) ;  
if (!$result)  
{  
    echo "<script> alert('lecture  
          impossible') </script>";  
}
```



objet de type
PDOStatement

Requêtes d'Interrogation

Objet PDOStatement

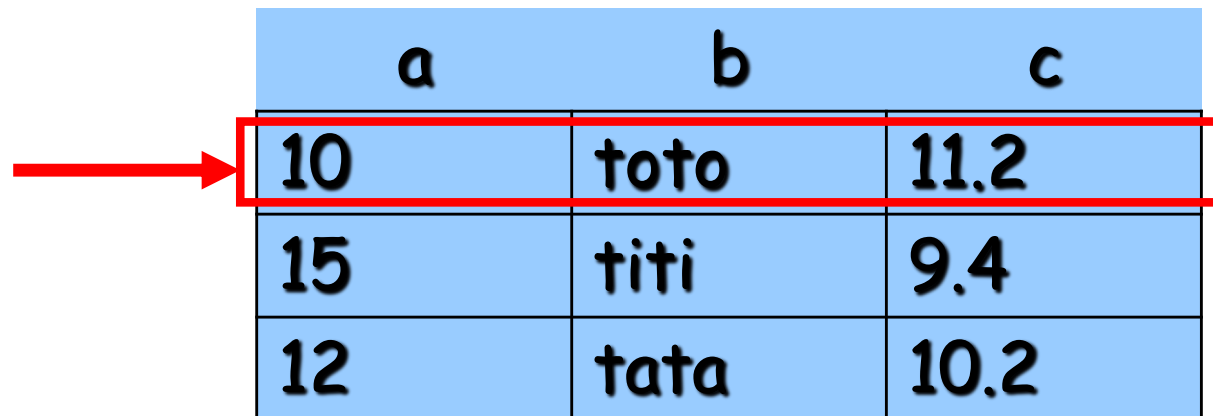
- Un objet PDOStatement représente le **résultat d'une requete** (table temporaire ou jeu de résultat)
- Plusieurs méthodes permettent de le manipuler

PDOStatement
<code>columnCount() : int</code> <code>rowCount() : int</code> <code>fetch(...): tuple (tableau ou objet)</code> <code>execute(paramètres)</code>

Objet PDO Statement

Principe de manipulation

- Un PDOStatement possède un curseur (ou pointeur) de lecture.
- Le curseur peut-être positionné:
 - sur une ligne de la table (un tuple)
 - avant la première ligne (lors de la création de l'objet PDOStatement)
 - après la dernière ligne



A diagram illustrating a database table with three columns labeled 'a', 'b', and 'c'. The table contains three rows of data. A red arrow points to the first row, which is also highlighted with a red border, indicating the current position of the cursor.

a	b	c
10	toto	11.2
15	titi	9.4
12	tata	10.2

Accéder au résultat d'une requête avec la méthode `fetch()`

- La méthode **`fetch()`** renvoie une ligne de la table résultat sous la forme d'un tableau ou d'un objet
 - *Elle positionne le curseur sur une ligne*
- `fetch(PDO::FETCH_ASSOC)`
 - renvoie un tableau indicé par les noms de colonne
- `fetch(PDO::FETCH_BOTH)`
 - renvoie un tableau indicé par les noms et les numéros de colonne
- `fetch(PDO::FETCH_OBJ)`
 - renvoie un objet ayant comme attributs les noms de colonnes

constantes de la
classe PDO

Comment accéder à une ligne du résultat?

Tableau indicé par les noms de colonnes

```
$ligne=$result->fetch(PDO::FETCH_ASSOC) ;
```

The diagram illustrates how to access data from a fetched row using associative array notation. A table with three columns (a, b, c) and three rows is shown. The first row's data (10, toto, 11.2) is highlighted with red boxes. A red arrow points from the label `$ligne['a']` to the value 10, and another red arrow points from the label `$ligne['c']` to the value 11.2.

a	b	c
10	toto	11.2
15	titi	9.4
12	tata	10.2

Comment accéder à une ligne du résultat?

Objet avec attributs (noms de colonnes)

```
$ligne=$result->fetch(PDO::FETCH_OBJ) ;
```

The diagram illustrates how to access data from a PDO result object. It shows a table with three columns: 'a', 'b', and 'c'. The first row of data is highlighted with red boxes around the values '10', 'toto', and '11.2'. A red arrow points from the text '\$ligne->a' to the '10' cell. Another red arrow points from the text '\$ligne->c' to the '11.2' cell. This demonstrates that the fetched object has attributes corresponding to the column names.

a	b	c
10	toto	11.2
15	titi	9.4
12	tata	10.2

Boucle de parcours d'un résultat avec la méthode fetch()

- Lors du premier appel, la méthode fetch() renvoie la **première ligne** de la table résultat.
- Elle retourne **false** si le dernier tuple a déjà été lu ou si la table est vide.
- Chaque appel fait avancer le curseur au tuple suivant

```
while ($ligne=$result->fetch(...)) {  
    // Traitement de chaque tuple  
  
}
```

Boucle de parcours d'un résultat avec la méthode fetch()

Suite code de la page 13

```
//lecture des resultats
echo "<table border=\"1\">";
echo "<tr><th> Nom </th> <th> Note</th></tr>";
while ($ligne=$result->fetch(PDO::FETCH_ASSOC))
{
    echo " <tr> <td>". $ligne['etu_nom'] . "</td>"
        . "<td>". $ligne['etu_note'] . "</td>"
        . "</tr>";
}
echo "</table>";
//fermeture de l'objet result
$result->closeCursor();
//fermeture de la connexion à la base de données
$bdd=null;
```

Requête paramétrée


- La méthode **prepare()** de l'objet **PDO** crée un PDOStatement qui comporte des paramètres spécifiés par des ? :

```
$ps = $bdd->prepare ("SELECT * FROM t_etudiant WHERE  
etu_nom = ? ");
```

- Pour exécuter une requête paramétrée, il faut invoquer la méthode **execute** sur l'objet PDOStatement

```
$nom="toto";  
$ps->execute (array ($nom) );
```

tableau des
paramètres effectifs



- Le résultat de la requête est affecté à l'objet PDOStatement lui-même.

Exemple de requête préparée

```
$nom=$_POST["nom"];
$salaire=$_POST["salaire"];

$req = "INSERT INTO t_utilisateur VALUES (?, ?)";
$r=$bdd->prepare($req);
$res=$r->execute(array($nom, $salaire));

if ($res)
    echo "<p>Le prof ". $nom ." a bien été ajouté</p>";
else
    echo "<p>Une erreur s'est produite</p>";
```




Tableau des paramètres effectifs

Exemple de requête préparée (autre écriture)

```
$nom=$_POST["nom"];  
$salaire=$_POST["salaire"];  
$req = "INSERT INTO t_utilisateur VALUES (?, ?)";
```

```
$r=$bdd->prepare($req);
```

Association des paramètres effectifs

```
$r->bindParam(1, $nom);  
$r->bindParam(2, $salaire);
```

```
$res=$r->execute();
```

```
if ($res)  
    echo "<p>Le prof ". $nom ." a bien été ajouté</p>";  
else  
    echo "<p>Une erreur s'est produite</p>";
```

Problème des caractères spéciaux

- Les caractères spéciaux apparaissant dans les zones de saisies d'un formulaire (*input-text* ou *textarea*) doivent être « échappés » (précédés du caractère \) avant d'être utilisés dans l'expression d'une requête (chaîne de caractères).

J'ai froid
C'est l'hiver



J\'ai froid \nC\'est l\'hiver

- Plusieurs possibilités:
 - Appel d'une fonction spécifique
 - \$exemple= **addslashes**(\$_POST["exemple"],""\n");
 - Utiliser des requêtes préparées

Solution la plus simple

Code erreurs renvoyés par MySQL

- `$rows = $bdd->exec(...requête MAJ ...)`
 `if ($rows ==1)`
 `// insertion OK`
 `else //erreur`
 `$code= $rows->errorCode();`
 `If ($code==...)`
- Exemples:
 - 1586: insertion impossible (duplicate key)
 - 1216: violation contrainte clé étrangère
- Pour une liste complète des codes erreurs:
 - <https://dev.mysql.com/doc/refman/8.0/en/server-error-reference.html>

Exercice d'application : TP1



- Création d'un site de gestion des droits de gestion d'utilisateurs.

Base de données Securite

table t_utilisateur

Attributs	Types	Commentaires
login	varchar 20	identifiant de l'utilisateur
mdp	varchar 15	mot de passe
nom	varchar 20	nom
categorie	varchar 20	categorie (admin ou user)

Menu gestion

- [Affichage utilisateurs](#)
- [Ajout utilisateur](#)
- [Recherche utilisateur](#)
- [Suppression utilisateur](#)