IUT Lyon 1
l'excellence technologique

Le langage PHP

(PDO)

Robin FRÈRE robin@widee.fr

Communiquer avec la base de données

- Pour communiquer avec MySQL, PHP offre 3 API :
  - mysql : obsolète en version PHP5.5.0, supprimée sur PHP7

```php
$c = mysql_connect("example.com", "user", "password");
mysql_select_db("database");
$result = mysql_query("SELECT 'Bonjour, cher utilisateur de MySQL !' AS _message FROM DUAL"    );
$row = mysql_fetch_assoc($result);
echo htmlentities($row['_message']);
```

  - mysqli : plus sécurisée que la précédente, la méthode procédurale dédié à MySQL

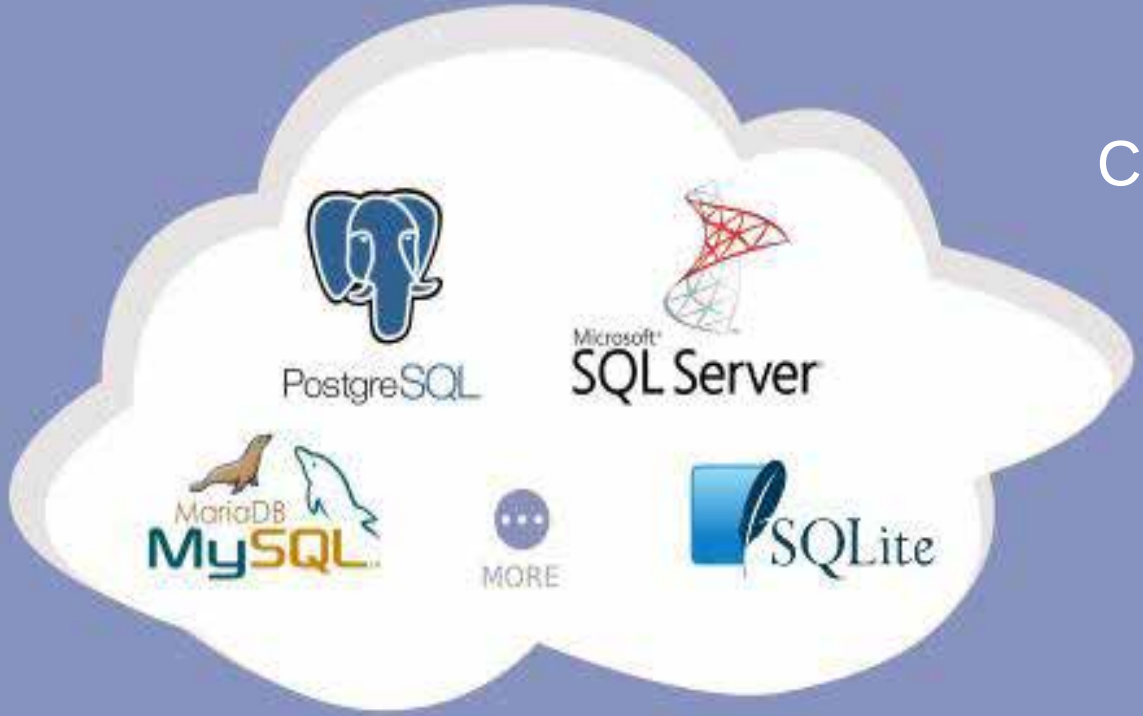
```php
$mysqli = new mysqli("example.com", "user", "password", "database");
$result = $mysqli->query("SELECT 'Bonjour, cher utilisateur de MySQL !' AS _message FROM DUAL"    );
$row = $result->fetch_assoc();
echo htmlentities($row['_message']);
```

  - PDO : La méthode orienté objet qui prend en charge plusieurs type de bases de données

```php
$pdo = new PDO('mysql:host=example.com;dbname=database' , 'user', 'password');
$statement = $pdo->query("SELECT 'Bonjour, cher utilisateur de MySQL !' AS _message FROM
DUAL");
$row = $statement->fetch(PDO::FETCH_ASSOC);
echo htmlentities($row['_message']);
```

- On choisira plutôt PDO pour sa flexibilité et sa sécurité

PDO



Communiquer avec la base de données

L’objet PDO

- Il faut au préalable activer les extensions dans le fichier php.ini :

```
; Charger PDO
extension=php_pdo.dll

; Charger le pilote pour MySQL
extension=php_pdo_mysql.dll

; Pour SQLite
extension=php_pdo_sqlite.dll
extension=php_sqlite3.dll
```

L’objet PDO

- PDO est un objet qui va communiquer avec la base de données
- Le constructeur se charge de la connexion à la BDD :

```
$pdo = new PDO($dsn [, $user [, $password [, $options]]] );
```

- $dsn (Data Source Name) en général le nom du pilote suivi d’une syntaxe spécifique :
  - MySQL : `mysql:dbname=testdb;host=127.0.0.1`
  - SQLite : `sqlite:theDBFile.db`
  - ODBC : `odbc:DSN_NAME`
  - `...`
- $user : le nom d’utilisateur de la BDD
- $password : le mot de passe de connexion à la BDD
- $options : paramètres complémentaires appliqués à l’objet PDO

L’objet PDO

Exemple de connexion avec MySQL :

```php
<?php
/* Connexion à une base MySQL avec l'invocation de pilote */
$dsn = 'mysql:dbname=testdb;host=127.0.0.1';
$user = 'dbuser';
$password = 'dbpass';

try {
    $dbh = new PDO($dsn, $user, $password);
} catch (PDOException $e) {
    echo 'Connexion échouée : ' . $e->getMessage();
}

echo 'Connexion à la base réussie';
```

PDO->exec

- Exécute une requête SQL
- Retourne le nombre de ligne affectées par la requête
- Ne retourne aucun résultat (il faudra préférer PDO::query() ou PDO::prepare())
- Retourne false en cas d'erreur (ATTENTION ! peut retourner un équivalent à false tel que 0)

```php
<?php
$dbh = new PDO('odbc:sample', 'db2inst1', 'ibmdb2');

/* Effacement de toutes les lignes de la table FRUIT */
$count = $dbh->exec("DELETE FROM fruit");

/* Retourne le nombre de lignes effacées */
echo "Retourne le nombre de lignes effacées :\n";
echo "Effacement de $count lignes.\n";
```

PDO->query

- Exécute une requête SQL
- Retourne un objet de type PDOStatement
- Retourne false en cas d'erreur

```php
<?php

$dsn = 'mysql:dbname=testdb;host=127.0.0.1';
$user = 'dbuser';
$password = 'dbpass';

$conn = new PDO($dsn, $user, $password);

$sql =  'SELECT name, color, calories FROM fruit ORDER BY name';

foreach  ($conn->query($sql) as $row) {
    echo $row['name'] . "\t";
    echo $row['color'] . "\t";
    echo $row['calories'] . "\n";
}
```

PDO->quote

- Protège une chaîne de caractères pour l’utiliser dans une requête SQL PDO
- Evite les injection SQL dans le code

```php
<?php
$conn = new PDO('sqlite:/home/lynn/music.sql3');

$string = $conn->quote($_GET[‘name’]);

$sql =  ”SELECT * FROM fruit WHERE name = '{$_GET[‘name’]}' ”; // faille injection SQL

$sql =  ”SELECT * FROM fruit WHERE name = $string ”; // requête protégée
```

PDO->prepare

- Prépare une requête SQL à l’exécution et retourne un objet
- Il est conseillé d’utiliser cette méthode pour sécuriser les requêtes lorsqu’on insère des variables
- Retourne un objet de type PDOStatement
- Retourne false en cas d’erreur

```php
$statement = $pdo->prepare("SELECT * FROM users WHERE id = :id");
```

- Dans l’exemple, **:id** sera la variable qui sera injecté lors de la requête d'exécution
- Dans le cas d’une requête préparée, les variables seront automatiquement nettoyées pour éviter les injections SQL
- Plus rapide à être réalisée que PDO::query() en cas d’utilisation de variables

PDO->prepare

- Exemple complet :

```php
$pdo = new PDO("sqlite:sqlite.db");

$statement = $pdo->prepare("SELECT * FROM users
          WHERE name = :varName AND email = :varMail");

$statement->execute( // On exécute la requête préparée
   [
       'varName' => 'john',            // un tableau contenant les variables
       'varMail' => 'john.doe@myemail.com'
   ]
);

// on récupère les informations grâce à l'objet PDOStatement
$users = $statement->fetchAll();
var_dump($users);
```

PDO->setAttribute

- Configurer un attribut PDO
- Méthode qui prend en premier paramètre l'attribut et en second la valeur
- Les paramètres attendus sont des constantes de PDO
- Liste des attributs disponibles : https://www.php.net/manual/fr/pdo.setattribute.php

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

Dans l'exemple on modifie le comportement de $pdo pour afficher les erreurs dans PDO comme une exception.

- Les attributs peuvent être définis lors de l'implémentation de l'objet :

```php
$pdo = new PDO($dsn, $user, $password, [ PDO::ATTR_ERRMODE
        => PDO::ERRMODE_EXCEPTION
    ]);
```

L’objet PDOStatement

- Représente une requête préparée et, une fois exécutée, le jeu de résultats associé.
- L’objet a pu être créé par PDO::prepare() et PDO::query()
- Les méthodes disponibles permettront de récupérer les données et autres opérations

PDOStatement->execute

- Méthode qui exécute une requête préparée
- Retourne true en cas de succès ou false si il y une erreur

```
$sth = $dbh->prepare('SELECT nom, couleur, calories
                      FROM fruit
                      WHERE calories < :calories AND couleur = :couleur');

$sth->execute([
      'calories' => $calories,
      'couleur' => $couleur
      ]);
```

PDOStatement->bindParam

- Lie un paramètre à un nom de variable spécifique
- Utile dans les requêtes préparées

```php
$calories = 150;
$couleur = 'rouge';

$sth = $dbh->prepare('SELECT nom, couleur, calories
                      FROM fruit
                      WHERE calories < :calories AND couleur = :couleur');

$sth->bindParam('calories', $calories, PDO::PARAM_INT); // PDO::PARAM_INT est une constante qui définit le format de la variable attendue

$sth->bindParam('couleur', $couleur, PDO::PARAM_STR, 12); // le dernier paramètre défini la longueur de la chaîne attendue

$sth->execute();
```

PDOStatement->fetch

- Permet de récupérer **une** ligne depuis un jeu de résultats associé à PDOStatement
- Prend en paramètre le style de récupération (voir fetch_style)

```php
<?php
$sth = $dbh->prepare("SELECT nom, couleur FROM fruit");
$sth->execute();

$result1 = $sth->fetch();
print_r($result1); // affiche le premier résultat

$result2 = $sth->fetch();
print_r($result2); // affiche le second résultat
```

PDOStatement->fetchAll

- Retourne un tableau contenant toutes les lignes du jeu d'enregistrement
- Prend en paramètre le style de récupération (idem fetch() voir fetch_style)

```php
<?php
$sth = $dbh->prepare("SELECT nom, couleur FROM fruit");
$sth->execute();

$result = $sth->fetchAll();
print_r($result); // affiche le tableau de tous les résultats
```

Fetch_style

- Par défaut les éléments retournés par fetch() ou fetchAll() sont des tableaux indexé numériquement **et** avec les noms de colonnes

```
[0] => Array
        (
            [nom] => apple
            [0] => apple
            [couleur] => red
            [1] => red
        )
```

Fetch_style

- Les options disponibles :
  - `PDO::FETCH_ASSOC` Résultats sous forme de tableau associatif
  - `PDO::FETCH_BOTH` (par défaut) Résultat sous forme de tableau associatif **et** numériquement indexé
  - `PDO::FETCH_LAZY` Combine `PDO::FETCH_BOTH` et `PDO::FETCH_OBJ`
  - `PDO::FETCH_NUM` Résultat sous forme de tableau indexé par le numéro de la colonne commençant par 0
  - `PDO::FETCH_OBJ` Résultat sous forme d'objet
  - … d'autres options disponibles : https://www.php.net/manual/fr/pdostatement.fetch.php
- Possibilité de définir les paramètres avec PDOStatement::setFetchMode :

```php
$stmt->setFetchMode(PDO::FETCH_NUM);
```

- Possibilité de déclarer lors de l'instanciation de PDO :

```php
$pdo = new PDO('odbc:sample', 'db2inst1', 'ibmdb2',
          [PDO::ATTR_DEFAULT_FETCH_MODE  => PDO::FETCH_OBJ]);
```

PDOException

- On récupère les erreurs de PDO à l’aide de cet Objet

```php
try {
	$PDO = new PDO( '...' );
}
catch( PDOException $Exception ) {
	echo $Exception->getMessage( ) . ‘ code erreur : ’ . $Exception->getCode( );
	die();
}
```

PDO - les transactions

- On peut utiliser PDO pour réaliser des transactions :

```php
$pdo = new PDO('...');

$pdo->beginTransaction();
$pdo->exec("UPDATE posts SET title = 'titre de mon article' WHERE id = 1" );
$pdo->exec("UPDATE posts SET content = 'Contenu de mon article' WHERE id =
1");
$pdo->exec("SELECT * FROM posts WHERE id = 1" );

if($update_BDD ){
   $pdo->commit(); // Les Changements seront enregistrés dans la BDD
}else{
   $pdo->rollBack(); // Annule les opérations précédentes
}
```

Les bonnes pratiques

- Toujours vérifier l'existence et la validité des variables :
  - l'existence en testant avec **isset()**, **empty()**
  - la validité : **gettype(), is_long(), is_double(), is_string()** …
  - tester avec les expressions régulière **preg_match()**
- Filtrer les données :
  - **htmlspecialchars()** : convertit les caractères spéciaux du HTML **& ‘ “ < >** en entité HTML
  - **htmlentities()** : convertir tous les caractères éligibles en leur entité
  - **strip_tags()** : Supprime les balises HTML et PHP
  - **[add|strip]slashes()** : ajoute/supprime un antislash devant les caractères spéciaux **\ ‘ “**
  - **quotemeta()** : ajoute un antislash devant les métas-caractères **. \ + * ? [^] ($)**
  - **trim()** : supprime les espaces surnuméraires

Stockage des mots de passe

- Les mots de passe ne doivent jamais être stockées de manière claire
- Préférer les méthodes de hachage à sens unique
- Il est déconseillé d’utiliser md5(), sha1 et sha256() car si elles sont rapides il est devenu facile de les déchiffrer.
- Une fonction pratique dans PHP : password_hash()