Mémo PHP

Quelle que soit la requête du CRUD on procède toujours de la même manière

- ligne 1 : créer l'objet PDO pour se connecter à notre BDD
- ligne 2 : Construire la requête SQL avec des paramètres nommés :param1, :param2, etc., pour se proétger contre l'injection SQL
- ligne 3 : Préparer la requête en injectant la requête SQL
- ligne 4 : Exécuter la requête en passant les valeurs des paramètres nommés
- ligne 5 : (pour un SELECT) : Récupérer les données avec fetchAll (pour plusieurs résultats) ou fetch (un seul résultat)
- ligne 6 : (pour un SELECT) : Fermer la connexion avec closeCursor

```
$pdo = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'user', 'password');
$sql = 'requête SQL du CRUD' ;                    // SELECT, INSERT INTO, UPDATE ou DELETE
$request = $pdo->prepare($sql) ;
$request->execute(['param1' => $value1, 'param2' => $value2]) ;
$data = $request->fetchAll(PDO::FETCH_ASSOC) ;    // Seulement pour SELECT
$request->closeCursor() ;                         // Libère le curseur de la requête pour éviter les erreurs
```

Remarque : pour plus de détails voir III

Les bases de SQL

La **bases de données** (BDD) sont gérées par des SGBD (**S**ystèmes de **G**estion de **B**ases de **D**onnées) qui sont des programmes qui se chargent du stockage des données. Nous utiliserons le SGBD **MySQL**.
Pour communiquer avec MySQL, on utilise le langage SQL.

CRUD :

- Create (INSERT INTO en sql)
- read (SELECT en sql)
- update (UPDATE en sql)
- Delete (DELETE en sql)

Table : utilisateur — Champs

Entrée →

id	nom	prenom	age	sex
1	Dupont	René	50	M
2	Dalton	Julie	35	F
3	Mbappe	Kylian	26	M
4	Habdou	Haicha	15	F
5	Dupont	Jean	60	M
6	Idji	Loanna	10	F
7	Ulloa	Léa	26	F

I) Lire la BDD avec SELECT

```
SELECT * FROM utilisateur
```

- SELECT : demande à MySQL d'afficher ce que contient une table.
- * : indique quels champs MySQL doit récupérer dans la table (* pour récupérer tous les champs).
- FROM utilisateur : indique à MySQL dans quelle table (ici utilisateur) on va récupérer les données.

```
SELECT nom, prenom FROM utilisateur
```

- nom, prenom : pour récupérer uniquement les champs nom et le prenom.

Critères de sélection : clause WHERE

```
SELECT * FROM utilisateur WHERE id=5;
```

On peut combiner plusieurs conditions avec AND et OR : `SELECT * FROM utilisateur WHERE age>25 AND sex="M"`

Ordonné : clause ORDER BY

```
SELECT * FROM utilisateur ORDER BY nom ASC          (ou DESC)
```

On peut combiner avec la clause WHERE : `SELECT * FROM utilisateur WHERE sex="F" ORDER BY nom ASC`

Limité : clause LIMIT

```
SELECT * FROM utilisateur LIMIT 10 : affiche les 10 premiers
SELECT * FROM utilisateur LIMIT 5, 10 : affiche 10 lignes à partir du 6ème
```

On peut combiner avec la clause WHERE et ORDER BY :

```
SELECT * FROM utilisateur WHERE sex="F" ORDER BY nom ASC LIMIT 2
```

Attention : Il est impératif de placer les mots-clés dans l'ordre : WHERE, puis ORDER BY et enfin LIMIT.

II) Modifier la BDD avec INSERT INTO, UPDATE ET DELETE

Ajouter des données : INSERT INTO

INSERT INTO utilisateur (nom, prenom, age, sex) VALUES (‘Rabbit','Roger', 16, ‘M')
Attention : On ne saisit pas le champ id, car il a la propriété AUTO_INCREMENT.

Modifier des données : UPDATE

UPDATE utilisateur SET prenom= ‘jeanne’, sex=’F’ WHERE id=5
Attention : si on oublie la clause WHERE, toutes les entrées seront modifiées.

Supprimer des données : DELETE

DELETE FROM utilisateur WHERE nom='Dupont'
Attention : si on oublie le WHERE, la table sera entièrement vidée.

III) Communiquer avec la BDD en PHP

Lire des données dans la BDD en PHP

Pour communiquer avec la BDD en PHP on va utiliser un objet PDO

```
$pdo = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'user', 'password');
```

On injecte la requête SQL dans l’objet PDO

```
$request = $pdo->query(‘SELECT * FROM utilisateur’) ;
```

On récupère les données

```
$users = $request->fetchall(PDO::FETCH_ASSOC) ;    // FETCH_ASSOC pour récupérer un tableau associatif
```

On peut afficher les utilisateurs un par un avec un foreach

```
foreach ($users as $user) {
  echo $user[‘nom’] . ‘ ‘ . $user[‘prenom’] . ‘<br>’ ;
}
```

Remarque : pour récupérer plusieurs lignes, on utilise fetchall() et pour récupérer une seule ligne on utilise fetch().

Attention : quand on injecte des variables dans une requête, on utilise des **requêtes préparées** pour se protéger de **l’injection SQL.**

```
$pdo = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'user', 'password');
$request = $pdo->prepare(‘SELECT * FROM utilisateur WHERE nom=:nom’) ;          // paramètre nommé (:nom)
$request->execute([‘non’ => ‘Jean’] ;                    // on donne une valeur au paramètre nommé (:nom)
$user = $request->fetch(PDO::FETCH_ASSOC) ;        // ici on utilise fetch() car on ne récupère qu’une seule ligne
```

Modifier la BDD en PHP

Pour modifier la BDD on tuilise des requêtes préparées.

```
$pdo = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'user', 'password');
$sql = ‘……’ // requête INSERT INTO, UDAPTE ou DELETE avec paramètres nommés
$request = $pdo->prepare($sql) ;
$request->execute([‘param1’ => $value1, ‘param2’ => $value2] ;
```

IV) Les fonctions SQL

Les fonctions scalaires

SELECT UPPER(nom) FROM utilisateur : renvoie le nom en majuscule
Autres exemples : LOWER(), LENGTH(), CEIL(), ...

On peut utiliser un alias : SELECT UPPER(nom) as nom_maj FROM utilisateur

Les fonctions d’agrégations : Elles ne renvoient qu’une seule valeur
SELECT COUNT(*) FROM utilisateur : renvoie le nombre de ligne
avec alias : SELECT COUNT(*) as nombre_utilisateur FROM utilisateur

Autres exemples : MAX(), MIN(), AVG(), ...

IV) Jointure

Une jointure permet de faire le lien entre les tables à l'aide d'une requête SQL, afin de relier les informations des deux tables entre elles.

Table : message

id	id_utilisateur	message
1	2	coucou
2	2	ca va ?
3	1	oui et toi ?
4	2	ca va merci.

Table : utilisateur

id	nom	prenom	age	sex
1	Dupont	René	50	M
2	Dalton	Julie	35	F
3	Mbappe	Kylian	26	M
4	Habdou	Haicha	15	F

Les jointure internes : INNER JOIN
elles ne sélectionnent que les données qui ont une correspondance entre les deux tables.

```
SELECT * FROM message m
INNER JOIN utilisateur u
ON m.id_utilisateur = u.id
```

id	id_utilisateur	message	id	nom	prenom	age	sex
1	1	coucou	1	Doe	John	50	M
2	2	ca va ?	2	Dalton	Julie	35	F
3	1	oui et toi ?	1	Doe	John	50	M
4	2	ca va merci.	2	Dalton	Julie	35	F

Les jointures externes permettent de récupérer toutes les données, même celles sans correspondance dans l'autre table.

La jointure externes LEFT JOIN

```
SELECT * FROM message m
LEFT JOIN utilisateur u
ON m.id_utilisateur = u.id
```

id	id_utilisateur	message	id	nom	prenom	age	sex
1	1	coucou	1	Doe	John	50	M
2	2	ca va ?	2	Dalton	Julie	35	F
3	1	oui et toi ?	1	Doe	John	50	M
4	2	ca va merci.	2	Dalton	Julie	35	F
5	4	ca va merci.	NULL	NULL	NULL	NULL	NULL

La jointure externes RIGHT JOIN

```
SELECT * FROM message m
RIGHT JOIN utilisateur u
ON m.id_utilisateur = u.id
```

id	id_utilisateur	message	id	nom	prenom	age	sex
1	1	coucou	1	Doe	John	50	M
3	1	oui et toi ?	1	Doe	John	50	M
2	2	ca va ?	2	Dalton	Julie	35	F
4	2	ca va merci.	2	Dalton	Julie	35	F
NULL	NULL	NULL	3	Mbappé	Kylian	26	M

V) Aller plus loin avec SQL

Le groupement de données : GROUP BY et HAVING
GROUP BY : clause utilisée avec une fonction d'agrégat pour obtenir des informations sur des groupes de données.
HAVING : est l'équivalent de WHERE, mais HAVING filtre après l'agrégation (c.a.d après un GROUP BY), contrairement à WHERE qui filtre avant.

Récupérer la moyenne d'âge des hommes et des femmes en ne comptant que ceux ayant plus de 18 ans.
SELECT AVG(age) AS age_moyen, sex FROM utilisateur WHERE age>18 GROUP BY sex
Résultat : M → 45,3 F → 35,5

Récupérer la moyenne d'age des hommes et des femmes en ne comptant que les moyennes supérieures à 18 ans.
SELECT AVG(age) AS age_moyen, sex FROM utilisateur GROUP BY sex HAVING AVG(age) > 18
Résultat : M → 45,3 (le filtre HAVING est appliqué après l'agrégation)

Les dates en SQL
Les types de dates :

- DATE : stocke une date au format AAAA-MM-JJ (Année-Mois-Jour) ;
- TIME : stocke un moment au format HH:MM:SS (Heures:Minutes:Secondes) ;
- DATETIME : stocke la combinaison d'une date et d'un moment de la journée au format AAAA-MM-JJ HH:MM:SS.
- YEAR : stocke une année, soit au format AA, soit au format AAAA.

```
SELECT * FROM chat WHERE date = '2010-04-02'
```

SELECT * FROM chat WHERE date >= '2010-04-02 15:28:22'
SELECT * FROM chat WHERE date >= '2010-04-02 00:00:00' AND date <= '2010-04-18 00:00:00'

Les fonctions de gestion des dates
NOW() : permet d'enregistrer la date actuelle au format AAAA-MM-JJ HH:MM:SS
INSERT INTO chat(pseudo, message, date) VALUES('jean', 'Coucou', NOW())
Autres exemples : CURDATE(), CURTIME(), DAY(), MONTH(), YEAR(), DATE_FORMAT, DATE_ADD, DATE_SUB