Concevez votre site web avec PHP et MySQL

30 hours Medium License

Écrire des données

Nous avons vu dans le chapitre précédent que l'on pouvait facilement récupérer des informations de notre base de données. Nous avons également pu constater que le langage SQL est très puissant car il propose de nombreux critères de sélection et de tri (`WHERE` , `ORDER BY` , etc.).

Il est maintenant temps de découvrir comment on peut ajouter et modifier des données dans la base. Pour cela, nous allons aborder de nouvelles requêtes SQL fondamentales et qu'il vous faut connaître : `INSERT` , `UPDATE` et `DELETE` .

INSERT : ajouter des données

Votre mission, si vous l'acceptez : ajouter une nouvelle entrée à la table « jeux_video » sur laquelle nous avons travaillé dans le chapitre précédent.

Mouahahahah, mais c'est facile. Tu utilises phpMyAdmin et hop ! c'est fait !
… Quoi, j'ai dit quelque chose de mal ?

Non, non.
C'est vrai que phpMyAdmin permet de rajouter de nouvelles entrées dans la table. Mais ce qui nous intéresse ici, c'est de le faire au moyen d'un script PHP et d'une requête SQL.

Tout d'abord, je vous rappelle à quoi ressemble la table « jeux_video » :

ID	nom	possesseur	console	prix	nbre_joueurs_max	commentaires
1	Super Mario Bros	Florent	NES	4	1	Un jeu d'anthologie !
2	Sonic	Patrick	Megadrive	2	1	Pour moi, le meilleur jeu au monde !
3	Zelda : ocarina of time	Florent	Nintendo 64	15	1	Un jeu grand, beau et complet comme on en voit rarement de nos jours
4	Mario Kart 64	Florent	Nintendo 64	25	4	Un excellent jeu de kart !
5	Super Smash Bros Melee	Michel	GameCube	55	4	Un jeu de baston délirant !
...	...	...	...	...	...	...

La requête `INSERT INTO` permet d'ajouter une entrée

Pour rajouter une entrée, vous aurez besoin de connaître la requête SQL. En voici une par exemple qui ajoute un jeu :

php

```
INSERT INTO jeux_video(ID, nom, possesseur, console, prix, nbre_joueurs_max, commentaires) VALUES('',
'Battlefield 1942', 'Patrick', 'PC', 45, 50, '2nde guerre mondiale')
```

Les nombres (tels que 45 et 50 ici) n'ont pas besoin d'être entourés d'apostrophes. Seules les chaînes de caractères les nécessitent.

Étudions un peu cette requête.

- D'abord, vous devez commencer par les mots-clés `INSERT INTO` qui indiquent que vous voulez insérer une entrée.
- Vous précisez ensuite le nom de la table (ici `jeux_video`), puis listez entre parenthèses les noms des champs dans lesquels vous souhaitez placer des informations.
- Enfin – et c'est là qu'il ne faut pas se tromper – vous inscrivez `VALUES` suivi des valeurs à insérer **dans le même ordre que les champs que vous avez indiqués**.

Vous remarquerez que pour le premier champ (ID), j'ai laissé des apostrophes vides. C'est voulu : le champ a la propriété `auto_increment`, MySQL mettra donc le numéro d'ID lui-même. On pourrait même se passer du champ ID dans la requête :

php

```
INSERT INTO jeux_video(nom, possesseur, console, prix, nbre_joueurs_max, commentaires) VALUES('Battlefield
1942', 'Patrick', 'PC', 45, 50, '2nde guerre mondiale')
```

C'est encore plus simple ! Le champ ID sera de toute façon automatiquement rempli par MySQL, il est donc inutile de le lister.

Enfin, si vous le désirez, sachez que vous n'êtes pas obligés de lister les noms des champs en premier ; cette requête marche tout aussi bien (mais elle est moins claire) :

php

```
INSERT INTO jeux_video VALUES('', 'Battlefield 1942', 'Patrick', 'PC', 45, 50,
      '2nde guerre mondiale')
```

Il faut lister les valeurs pour tous les champs sans exception (ID compris) dans le bon ordre.

Application en PHP

Utilisons cette requête SQL au sein d'un script PHP. Cette fois, au lieu de faire appel à `query()` (que l'on utilisait dans le chapitre précédent pour récupérer des données), on va utiliser `exec()` qui est prévue pour exécuter des modifications sur la base de données :

php

```
<?php
try
{
    $bdd = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'root', '');
}
catch(Exception $e)
{
        die('Erreur : '.$e->getMessage());
}

// On ajoute une entrée dans la table jeux_video
$bdd->exec('INSERT INTO jeux_video(nom, possesseur, console, prix, nbre_joueurs_max, commentaires)
VALUES(\'Battlefield 1942\', \'Patrick\', \'PC\', 45, 50, \'2nde guerre mondiale\')');
```

```
13
14 echo 'Le jeu a bien été ajouté !';
15 ?>
```

Que fait ce code ? Il ajoute une entrée dans la BDD pour le jeu « Battlefield 1942 », appartenant à « Patrick », qui fonctionne sur « PC », qui coûte 45 euros, etc.

La présence de multiples apostrophes rend la requête un peu difficile à lire et à écrire à cause des antislashs `\` que l'on doit rajouter devant. De plus, cette requête insère toujours les mêmes données. Comme on l'a vu dans le chapitre précédent, si on veut rendre une partie de la requête variable, le plus rapide et le plus sûr est de faire appel aux requêtes préparées.

Insertion de données variables grâce à une requête préparée

Si on choisit d'utiliser une requête préparée (ce que je vous recommande si vous souhaitez insérer des variables), le fonctionnement est en fait exactement le même que dans le chapitre précédent :

php

```
1  <?php
2  $req = $bdd->prepare('INSERT INTO jeux_video(nom, possesseur, console, prix, nbre_joueurs_max,
   commentaires) VALUES(:nom, :possesseur, :console, :prix, :nbre_joueurs_max, :commentaires)');
3  $req->execute(array(
4      'nom' => $nom,
5      'possesseur' => $possesseur,
6      'console' => $console,
7      'prix' => $prix,
8      'nbre_joueurs_max' => $nbre_joueurs_max,
9      'commentaires' => $commentaires
10     ));
11
12 echo 'Le jeu a bien été ajouté !';
13 ?>
```

Désormais je ne mets plus l'étape de la connexion à MySQL avec PDO dans mes codes pour les simplifier. Bien entendu, il faut toujours se connecter au préalable si on veut que la requête fonctionne.

Pour plus de clarté, j'ai utilisé ici des marqueurs nominatifs. Comme vous le voyez, j'ai créé l'array sur plusieurs lignes : c'est autorisé, et c'est surtout bien plus lisible.

Les variables telles que `$nom` et `$possesseur` doivent avoir été définies précédemment. Généralement, on récupèrera des variables de `$_POST` (issues d'un formulaire) pour insérer une entrée dans la base de données. Nous découvrirons un cas pratique dans le TP suivant.

UPDATE : modifier des données

Vous venez de rajouter « Battlefield » dans la BDD et tout s'est bien passé.
Mais… vous vous rendez compte avec stupeur que « Battlefield » se joue en fait à 32 joueurs maximum (au lieu de 50) et qu'en plus son prix a baissé : on le trouve à 10 euros (au lieu de 45).

La requêteUPDATEpermet de modifier une entrée

No problemo amigo !
Avec une petite requête SQL, on peut arranger ça. En effet, en utilisant `UPDATE` vous allez pouvoir modifier l'entrée qui

pose problème :

```php
UPDATE jeux_video SET prix = 10, nbre_joueurs_max = 32 WHERE ID = 51
```

Comment ça marche ?

- Tout d'abord, le mot-clé `UPDATE` permet de dire qu'on va modifier une entrée.
- Ensuite, le nom de la table (`jeux_video`).
- Le mot-clé `SET` , qui sépare le nom de la table de la liste des champs à modifier.
- Viennent ensuite les champs qu'il faut modifier, séparés par des virgules. Ici, on modifie le champ « prix », on lui affecte la valeur « 10 » (`prix = 10`), puis on fait de même pour le champ `nbre_joueurs_max` . Les autres champs ne seront pas modifiés.
- Enfin, le mot-clé `WHERE` est tout simplement indispensable. Il nous permet de dire à MySQL quelle entrée il doit modifier (sinon, toutes les entrées seraient affectées !). On se base très souvent sur le champ ID pour indiquer **quelle entrée** doit être modifiée. Ici, on suppose que « Battlefield » a été enregistré sous l'ID no 51.

Si vous voulez, vous pouvez vous baser sur le nom du jeu au lieu de l'ID pour effectuer votre sélection :

```php
UPDATE jeux_video SET prix = '10', nbre_joueurs_max = '32' WHERE nom = 'Battlefield 1942'
```

Dernière minute ! Florent vient de racheter tous les jeux de Michel. Il va falloir modifier ça tout de suite.

Euh, il va falloir modifier chaque entrée une à une ?

Non ! Il n'est pas question de passer des heures à modifier chaque entrée une à une pour ça ! En réfléchissant environ 0,5 seconde, vous allez trouver tout seuls la requête SQL qui permet de faire ce qu'on souhaite.

C'est bon, vous avez trouvé ? Allez, je vous donne la réponse dans le mille :

```php
UPDATE jeux_video SET possesseur = 'Florent' WHERE possesseur = 'Michel'
```

Traduction : « **Dans la table`jeux_video`, modifier toutes les entrées dont le champ`possesseur`est égal à Michel, et le remplacer par Florent**. »

Qu'il y ait 1, 10, 100 ou 1 000 entrées, cette requête à elle seule suffit pour mettre à jour toute la table ! Si c'est pas beau, le SQL… ;-)

Application en PHP

De la même manière, en PHP on fait appel à `exec()` pour effectuer des modifications :

```php
<?php
$bdd->exec('UPDATE jeux_video SET prix = 10, nbre_joueurs_max = 32 WHERE nom = \'Battlefield 1942\'');
?>
```

Notez que cet appel renvoie le nombre de lignes modifiées. Essayez de récupérer cette valeur dans une variable et de l'afficher, par exemple comme ceci :

```php
<?php
$nb_modifs = $bdd->exec('UPDATE jeux_video SET possesseur = \'Florent\' WHERE possesseur = \'Michel\'');
```

```
3 echo $nb_modifs . ' entrées ont été modifiées !';
4 ?>
```

Cela affichera quelque chose comme : `13 entrées ont été modifiées !`

Avec une requête préparée

Si vous insérez des données variables, par exemple envoyées par l'utilisateur, je vous recommande là encore de faire appel à une requête préparée :

php

```
1 <?php
2 $req = $bdd->prepare('UPDATE jeux_video SET prix = :nvprix, nbre_joueurs_max = :nv_nb_joueurs WHERE nom =
  :nom_jeu');
3 $req->execute(array(
4     'nvprix' => $nvprix,
5     'nv_nb_joueurs' => $nv_nb_joueurs,
6     'nom_jeu' => $nom_jeu
7     ));
8 ?>
```

DELETE : supprimer des données

Enfin, voilà une dernière requête qui pourra se révéler utile : `DELETE`. Rapide et simple à utiliser, elle est quand même un poil dangereuse : après suppression, il n'y a aucun moyen de récupérer les données, alors faites bien attention !

Voici comment on supprime par exemple l'entrée de « Battlefield » :

php

```
1 DELETE FROM jeux_video WHERE nom='Battlefield 1942'
```

Il n'y a rien de plus facile :

- `DELETE FROM` : pour dire « supprimer dans » ;
- `jeux_video` : le nom de la table ;
- `WHERE` : indispensable pour indiquer quelle(s) entrée(s) doi(ven)t être supprimée(s).

Si vous oubliez le `WHERE`, toutes les entrées seront supprimées. Cela équivaut à vider la table.

Je vous laisse essayer cette requête en PHP. Vous pouvez là encore passer par `exec()` si vous voulez exécuter une requête bien précise, ou bien utiliser une requête préparée si votre requête dépend de variables.

En résumé

- On utilise différents mots-clés en fonction du type de modification que l'on souhaite effectuer :
  - `INSERT INTO` : ajout d'une entrée ;
  - `UPDATE` : modification d'une ou plusieurs entrées ;
  - `DELETE` : suppression d'une ou plusieurs entrées.
- Comme pour la sélection de données, on utilise les requêtes préparées pour personnaliser nos requêtes en fonction de variables.

- Lorsqu'on utilise `UPDATE` ou `DELETE`, il faut penser à filtrer avec un `WHERE` sinon toute la table sera affectée par l'opération !

Que pensez-vous de ce cours ?

Lire des données

TP : un mini-chat

The author

Mathieu Nebra

Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms :o)

Check out this course via