PHP

Les fonctions clés de MySQLi :

MySQLi (MySQL Improved) est une extension de PHP utilisée pour interagir avec des bases de données MySQL.

Voici une recréation du tableau des fonctions clés de MySQLi :

Fonction	Description
`mysqli_connect()`	Ouvre une connexion à une base de données MySQL (mode procédural).
`mysqli_query()`	Exécute une requête SQL.
`mysqli_prepare()`	Prépare une requête SQL pour l'exécution.
`mysqli_execute()`	Exécute une requête préparée.
`mysqli_fetch_assoc()`	Récupère une ligne de résultat sous forme de tableau associatif.
`mysqli_close()`	Ferme une connexion à une base de données.
`$mysqli->query()`	Exécute une requête SQL (mode orienté objet).
`$mysqli->prepare()`	Prépare une requête SQL (mode orienté objet).
`$mysqli->connect_error`	Récupère les erreurs de connexion.
`mysqli_real_escape_string()`	Échappe les caractères spéciaux dans une chaîne pour une utilisation dans une requête SQL.

Exemple de connection :

```
<?php
$host = "localhost";
$username = "root";
$password = "safaa";
$database = "bibliotheque";

$connection = mysqli_connect($host, $username, $password, $database);

if (!$connection) {
    die("Échec de connexion : " . mysqli_connect_error());
}
echo "Connexion réussie.";


?>
```

Techniques de récupération de données avec MySQLi

Lorsqu'une requête SQL est exécutée, les résultats sont retournés sous forme de ressource (ou objet). Pour accéder à ces données, il existe plusieurs fonctions de récupération, chacune adaptée à des besoins spécifiques. Voici les détails et les cas d'utilisation :

Méthode	Structure des données retournées	Avantages	Inconvénients	Quand l'utiliser ?
`mysqli_fetch_assoc()`	Tableau associatif `['colonne' => valeur]`	- Lisibilité grâce aux noms des colonnes - Facile à utiliser	- Moins performant que les index numériques	- Quand vous avez besoin d'accéder aux colonnes par leur nom - Utile pour des boucles itératives simples
`mysqli_fetch_array()`	Tableau associatif et/ou numérique `['colonne' => valeur, 0 => valeur]`	- Flexible : accès par clé associative ou numérique - Utile pour des cas dynamiques	- Plus gourmand en mémoire - Moins performant	- Quand vous avez besoin d'accéder aux données avec des indices numériques et des noms de colonnes
`mysqli_fetch_row()`	Tableau indexé numériquement `[0, 1, 2, ...]`	- Performant et léger - Moins gourmand en mémoire	- Moins lisible : impossible de savoir à quoi correspond chaque index sans documentation	- Quand vous privilégiez les performances - Si les noms des colonnes ne sont pas nécessaires
`mysqli_fetch_all()`	Tableau multidimensionnel `[[ligne1], [ligne2]]`	- Récupère toutes les lignes d'un coup - Simple pour traiter de petits résultats	- Peut être très gourmand en mémoire pour de grandes bases	- Quand vous savez que le résultat est limité en taille - Utile pour exporter ou manipuler des ensembles complets de données

Dans le code que j'ai développé pour pratiquer et utilisé les différentes méthodes de récupération des données de la base de données. Voici un résumé des différentes techniques et comment elles fonctionnent :

1. `mysqli_fetch_assoc()` :

   Cette méthode récupère les résultats sous forme de tableau associatif, ce qui signifie que chaque ligne de résultat peut être accédée via le nom de la colonne.

   Avantage : Cela rend le code plus lisible et plus facile à comprendre.

   Inconvénient : L'accès par nom de colonne est légèrement plus lent que l'accès par index dans certaines situations, bien que cela ne soit généralement pas un problème pour des ensembles de données de taille moyenne.

2. `mysqli_fetch_row()` :

Cette méthode renvoie les résultats sous forme de tableau indexé numériquement, c'est-à-dire que chaque ligne de résultat est accessible via des indices numériques.

Avantage : Elle est rapide et utilise moins de mémoire, car elle ne crée pas un tableau associatif supplémentaire.

Inconvénient : Le code devient moins lisible, car il faut connaître l'ordre des colonnes dans la base de données.

3. `mysqli_fetch_array()` :

   Cette méthode est une combinaison de `fetch_assoc()` et `fetch_row()`, elle permet d'accéder aux données à la fois par leur nom de colonne et par leur index.

   Avantage : Très flexible, elle permet de choisir la méthode d'accès en fonction des besoins.

   Inconvénient : Cette méthode utilise plus de mémoire et peut être un peu moins performante, car elle duplique les résultats sous deux formats (associatif et indexé).

4. `mysqli_fetch_all()` :

   Cette méthode récupère toutes les lignes de résultats à la fois sous forme de tableau.

   Avantage : Plus simple à utiliser pour des petits jeux de données, elle permet de parcourir facilement toutes les lignes avec une boucle `foreach`.

   Inconvénient : Pour de grandes quantités de données, cela peut être inefficace, car toute la table est chargée en mémoire à la fois, ce qui peut entraîner des problèmes de performance.

Exemple de code :

```
<table border="1">
        <tr>
            <th>ID</th>
            <th>Titre</th>
            <th>Auteur</th>
            <th>Catégorie</th>
            <th>Date d'ajout</th>
            <th>Disponibilité</th>
        </tr>
        <?php
        $resultat = mysqli_query($connection, $requete);
        while ($ligne = mysqli_fetch_assoc($resultat)) : ?>
            <tr>
                <td><?= $ligne['id'] ?></td>
                <td><?= $ligne['titre'] ?></td>
                <td><?= $ligne['auteur'] ?></td>
                <td><?= $ligne['categorie'] ?></td>
                <td><?= $ligne['date_ajout'] ?></td>
                <td><?= $ligne['disponibilite'] ? 'Oui' : 'Non' ?></td>
            </tr>
        <?php endwhile; ?>
    </table>
```

Roles de: `include`, `require`, `include_once`, et `require_once` :

les fonctions
`include`, `require`, `include_once` et `require_once` sont utilisées pour inclure et exécuter des fichiers externes dans un script. Leur rôle principal est de permettre la réutilisation du code et d'organiser les projets en plusieurs fichiers

Différence entre `include`, `require`, `include_once`, et `require_once` :

- `include` : Génère un avertissement mais continue l'exécution en cas d'échec.
- `require` : Arrête l'exécution en cas d'échec.
- `include_once` / `require_once` : Assure que le fichier n'est inclus qu'une seule fois.

Safaa Ettalhi