PHP : Créer et modifier un enregistrement

Quand on développe un site web dynamique (comme un blog, un gestionnaire de livres ou une base de données d'élèves), on a souvent besoin de **formulaires** pour **ajouter** ou **modifier** des données.

Objectif du cours

À la fin de ce cours, tu seras capable de :

- Utiliser **un seul formulaire** pour gérer **l'ajout** et la **modification** d'un enregistrement
- Lire un paramètre dans l'URL (`$_GET['id']`) pour passer en **mode édition**
- Pré-remplir un formulaire avec les données existantes
- Traiter le formulaire avec **PDO** pour faire un `INSERT` ou un `UPDATE`
- Appliquer les **bonnes pratiques de sécurité et de lisibilité**

Introduction

Plutôt que de créer deux pages distinctes (`ajouter.php` et `modifier.php`), il est possible d'utiliser **une seule page** qui :

- Affiche un **formulaire vide** pour l'ajout
- Affiche un **formulaire pré-rempli** pour l'édition
- Détecte automatiquement l'action à faire : `INSERT` ou `UPDATE`

Ce que cette page va faire

Voici un paragraphe clair et pédagogique qui explique **avec des mots** les différentes étapes de fonctionnement de la page `livre_form.php` :

La page `livre_form.php` est conçue pour gérer **à la fois l'ajout et la modification** d'un livre. Elle commence par établir une **connexion sécurisée à la base de données** avec PDO. Ensuite, elle vérifie s'il y a un paramètre `id` dans l'URL (`$_GET['id']`). Si cet ID est présent, cela signifie qu'on est en **mode édition** : on exécute une requête `SELECT` pour récupérer les données du livre à modifier, et on remplit les variables `$titre`, `$auteur` et `$annee` avec les valeurs existantes. Ces valeurs serviront à **pré-remplir le formulaire**. Si aucun ID n'est fourni, on reste en **mode ajout**, avec un formulaire vide prêt à recevoir de nouvelles données.

Ensuite, la page vérifie si le formulaire a été envoyé (en testant la méthode `POST`). Si c'est le cas, elle récupère les données saisies par l'utilisateur, les nettoie (avec `trim()`) et les valide (par exemple, le titre et l'auteur ne doivent pas être vides). Si un ID est présent dans le formulaire (caché dans un champ `hidden`), la page effectue une requête `UPDATE` : elle **modifie** les données existantes. Sinon, elle effectue un `INSERT` : elle **ajoute** un nouveau livre dans la base. Dans les deux cas, les requêtes SQL sont préparées avec des paramètres pour éviter les failles de sécurité (comme les injections SQL). Enfin, un message de confirmation est affiché à l'utilisateur, et on peut ajouter une redirection si on le souhaite.

En résumé, cette page s'adapte automatiquement selon le contexte (ajout ou modification), ce qui permet de **réutiliser le même code** pour deux actions différentes, tout en restant clair, sécurisé et maintenable.

La structure de la base

Voici un exemple de table pour stocker des livres :

```sql
CREATE TABLE livres (
  id INT AUTO_INCREMENT PRIMARY KEY,
  titre VARCHAR(100) NOT NULL,
  auteur VARCHAR(50) NOT NULL,
  annee INT
);
```

Le fichier `livre_form.php` (tout-en-un)

```php
<?php
// Connexion PDO (à adapter selon ton environnement)
$pdo = new PDO("mysql:host=localhost;dbname=bibliotheque;charset=utf8", "root", "");

// Initialisation des variables
$mode = "ajouter";
$id = null;
$titre = "";
$auteur = "";
$annee = "";

// Étape 1 : détection du mode (édition ou ajout)
if (isset($_GET['id'])) {
    $id = (int) $_GET['id'];

    // Récupérer le livre
    $stmt = $pdo->prepare("SELECT * FROM livres WHERE id = ?");
    $stmt->execute([$id]);
    $livre = $stmt->fetch(PDO::FETCH_ASSOC);

    if ($livre) {
        $mode = "modifier";
        $titre = $livre['titre'];
        $auteur = $livre['auteur'];
        $annee = $livre['annee'];
    } else {
        die("Livre introuvable.");
    }
}

// Étape 2 : traitement du formulaire
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Sécurité : on nettoie les données
```

```php
    $titre = trim($_POST['titre']);
    $auteur = trim($_POST['auteur']);
    $annee = !empty($_POST['annee']) ? (int) $_POST['annee'] : null;

    // Vérification des champs obligatoires
    if ($titre === '' || $auteur === '') {
        die("Le titre et l'auteur sont obligatoires.");
    }

    // INSERT ou UPDATE ?
    if (!empty($_POST['id'])) {
        // 🔁 Mise à jour
        $id = (int) $_POST['id'];
        $stmt = $pdo->prepare("UPDATE livres SET titre = ?, auteur = ?, annee = ? WHERE id = ?
        $stmt->execute([$titre, $auteur, $annee, $id]);
        echo "✅ Livre modifié avec succès.";
    } else {
        // ➕ Insertion
        $stmt = $pdo->prepare("INSERT INTO livres (titre, auteur, annee) VALUES (?, ?, ?)");
        $stmt->execute([$titre, $auteur, $annee]);
        echo "✅ Livre ajouté avec succès.";
    }

    // Redirection éventuelle après traitement (bonne pratique)
    // header("Location: liste.php");
    // exit;
}
?>
```

Formulaire HTML (à inclure après le code PHP)

```php
<h2><?= ucfirst($mode) ?> un livre</h2>
<form method="POST">
  <?php if ($mode === "modifier"): ?>
    <input type="hidden" name="id" value="<?= htmlspecialchars($id) ?>">
  <?php endif; ?>

  <label>
    Titre : <input type="text" name="titre" value="<?= htmlspecialchars($titre) ?>" required>
  </label><br>

  <label>
    Auteur : <input type="text" name="auteur" value="<?= htmlspecialchars($auteur) ?>" require
  </label><br>

  <label>
    Année : <input type="number" name="annee" value="<?= htmlspecialchars($annee) ?>">
  </label><br>

  <button type="submit">
    <?= $mode === "modifier" ? "Enregistrer les modifications" : "Ajouter" ?>
  </button>
</form>
```

```
23
24  <a href="liste.php">⬅️ Retour à la liste</a>
```

Étapes clés expliquées

Étape	Pourquoi ?
Vérifie `$_GET['id']`	Pour détecter si on veut modifier un enregistrement
Prépare une requête `SELECT`	Pour charger les données existantes
Affiche un formulaire avec les champs pré-remplis	Pour permettre la modification
Utilise `$_SERVER['REQUEST_METHOD']`	Pour séparer l'affichage du traitement
Nettoie et valide les données (`trim()`, `required`)	Pour éviter les erreurs ou les injections
Utilise `PDO` et des **requêtes préparées**	Pour sécuriser contre les injections SQL
Ajoute un champ `<input type="hidden" name="id">`	Pour savoir s'il faut faire `UPDATE` ou `INSERT`

Bonnes pratiques

- ✅ Utilise **PDO avec des requêtes préparées**
- ✅ Vérifie que les champs obligatoires sont bien remplis (`required`, `trim`)
- ✅ Évite les doublons de code avec un **formulaire réutilisable**
- ✅ Protège l'affichage avec `htmlspecialchars()`
- ✅ Prévoyez une **redirection après enregistrement** pour éviter les doublons (ex: si l'utilisateur recharge la page)

Résumé – Ce qu'il faut retenir

1. Tu peux gérer **ajout + modification** sur une même page PHP
2. Utilise `GET` pour charger les données à modifier, et `POST` pour enregistrer
3. L'**ID** permet de distinguer entre `INSERT` et `UPDATE`
4. Les **requêtes préparées avec PDO** sont indispensables pour la sécurité
5. Un formulaire bien structuré doit être **pré-rempli** si on est en mode édition