Modifier un produit

Maintenant que l'ajout de produit est fonctionnel, nous allons nous consacrer à la modification des produits.

1. Adapter le formulaire

Pour réaliser le système de modification, nous baserons sur le formulaire qui a été créé pour ajouter un produit. En effet, cela parait être la meilleure solution pour éviter l'écriture de code redondant.

Nous allons voir qu'avec quelques modifications nous pouvons facilement rendre compatible le formulaire pour assurer les deux cas.

Dans « form_product.html.twig », apportez les modifications pour obtenir le résultat suivant :

```
<form action="{{ page }}" method="POST">
    <div class="mb-3">
      <label for="productLabel" class="form-label">Libellé</label>
      <input type="text" class="form-control" id="productLabel" name="productLabel"
value="{{ product.label }}">
    </div>
    <div class="mb-3">
      <label for="productDescription" class="form-label">Description</label>
      <textarea class="form-control" id="productDescription"
name="productDescription" rows="3">{{ product.description }}</textarea>
    </div>
    <div class="mb-3">
      <label for="productPrice" class="form-label">Prix</label>
      <input type="number" class="form-control" id="productPrice"
name="productPrice" step="0.01" value="{{ product.price }}">
    </div>
    <div class="mb-3">
      <select class="form-select" name="productCategory">
        < !—VOTRE CODE PHP -->
      </select>
    </div>
    <button type="submit" class="btn btn-primary"
name="btnPostProduct">Ajouter</button>
  </form>
```

Explication :

- La variable « {{ page }} » fait référence à une nouvelle valeur que nous allons devoir faire passer depuis le contrôleur. Cette valeur correspond à la cible du formulaire.

- Pour chaque champ du formulaire, nous ajoutons l'attribut « value ». Cela permet d'insérer la valeur dans le champ lorsque nous souhaitons modifier un produit. Dans le cas d'un ajout, le champ sera automatiquement vide.
- Pour uniformiser le nommage des variables, nous devons modifier ici le nom du bouton en « btnPostProduct ».

Info :
Le code « *< !—VOTRE CODE PHP -->* » correspond au code que vous avez réalisé au cours d'un précédent chapitre.

2. Mettre à jour le produit dans la base de données

Pour mettre à jour le produit en base de données, nous allons créer une nouvelle fonction dans le fichier « ProductModel.php ». Celle-ci se nommera « updateOneProduct ».

Voici le code de la fonction :

```
function updateOneProduct($db, $id, $label, $description, $price, $category) {
    $query = $db->prepare("UPDATE shop_products SET label=:label,
`description`=:descr, price=:price, idCategory=:category WHERE id=:id");
    return $query->execute([
      'id' => $id,
      'label' => $label,
      'descr' => $description,
      'price' => $price,
      'category' => $category,
    ]);
}
```

Explications :

- En SQL, pour mettre à jour une donnée, il faudra utiliser le mot-clé « UPDATE » suivi du nom de la table et d'un autre mot-clé « SET ».
- Après le mot-clé « SET » il faut écrire la liste des colonnes que la requête doit modifier. Chaque colonne est séparée par un « ; ».
- La requête est constituée de valeurs dynamiques. Ces valeurs seront remplacées par les variables que nous lui passerons en paramètre.
- Enfin, nous devons renseigner l'id du produit à modifier, c'est pour cela que la requête se termine avec l'instruction SQL « WHERE id= :id ».

3. Le contrôleur

Nous avons réalisé l'ensemble du code nécessaire pour permettre la modification d'un produit. Il nous reste plus qu'à ordonner à notre contrôleur d'effectuer les actions nécessaires.

Exercice pratique :

Créer une nouvelle route « updateProduct » qui répondra grâce à la fonction « updateProductController » du fichier contrôleur « UpdateProductControlle.php ».

Dans le contrôleur que vous venez de créer et plus particulièrement dans la fonction, vous allez insérer le code suivant :

```
function updateProductController($twig, $db) {

    include_once '../src/model/ProductModel.php';

    if (isset($_GET['product'])) {
        $product = getOneProduct($db, $_GET['product']);

        if (isset($_POST['btnPostProduct'])) {
            $label = $_POST['productLabel'];
            $description = $_POST['productDescription'];
            $price = $_POST['productPrice'];

            if(empty($price)) {
                $price = 0.00;
            }

            $category = $_POST['productCategory'];

            if (!empty($label) && !empty($description) && !empty($category)) {
                updateOneProduct($db,$product['id'], $label, $description, $price, $category);
            }
        }

        echo $twig->render('form_product.html.twig', [
            'product' => $product,
            'page' => '?page=updateProduct&product='.$product['id']
        ]);
    } else {
        echo $twig->render('home.html.twig', []);
    }
}
```

Explication :

- La première portion de code permet de récupérer le produit en base de données en fonction de l'id que nous avons passé dans l'URL. Il est récupéré depuis le mot-clé « product ». Cela impose donc à ce que l'URL contienne « product=1 » par exemple pour obtenir le produit dont l'id équivaut à 1.
- La deuxième portion de code s'exécute lorsque le formulaire est envoyé. De cette manière nous effectuons ici quelques vérifications avant d'utiliser la fonction « updateOneProduct » permettant de modifier l'enregistrement en base de données.

Les vérifications sont ici les mêmes que pour le formulaire d'ajout. N'hésitez donc pas à consulter le chapitre sur l'ajout d'un produit pour en savoir un peu plus.

- Enfin, pour la dernière portion de code, nous passons à notre vue la variable « $product » et l'URL de destination pour notre formulaire.

Vous pouvez maintenant tester que tout fonctionne. Pour cela rendez-vous sur la page de votre site et ajouter à la fin : « ?page=updateProduct&product=1 »

Attention :
Lorsque vous modifiez votre produit, la modification n'est pas directe. Pensez donc à recharger votre page.
Pour ce faire, pensez à cliquer dans votre barre d'adresse et à faire entrer pour rafraichir votre page.

⇨ Ne revoyez pas une nouvelle fois le formulaire

4. Modifier le AddProductController

Puisque nous avons modifié le comportement de notre vue, nous devons modifier le comportement du contrôleur permettant l'ajout d'un produit.

Dans « AddProductController.php » faites les modifications selon l'extrait de code suivant :

```
if (isset($_POST['btnPostProduct'])) {
        ...
}

echo $twig->render('form_product.html.twig', [
   'form' => $form,
   'page' => '?page=addProduct'
]);
```

Vérifiez que l'ajout fonctionne toujours.

Exercice pratique :

- Apporter les modifications nécessaires pour que le formulaire de modification d'un produit puisse afficher des messages de succès et d'erreur.
- Faire en sorte que si l'id d'un produit n'est pas renseigné dans l'URL, un message s'affiche en indiquant à l'utilisateur que le produit n'existe pas et lui propose de retourner sur la page d'accueil. Actuellement, nous affichons à l'utilisateur la page d'accueil.
- Modifier le texte du bouton en « Modifier » lorsque l'utilisateur est sur la page de modification et en « Ajouter » lorsqu'il est sur la page d'ajout.
- Faire de même pour le titre de la page.