Envoyer un fichier

Dans ce chapitre, nous allons ajouter la possibilité à un utilisateur d'ajouter des images pour chacun des produits.

Sur un aspect plus théorique, l'envoi de fichier est une fonctionnalité très sensible en matière de sécurité des applications. Pour en assurer la sécurisation, nous allons devoir transférer le fichier envoyé dans un dossier temporaire afin qu'il subisse plusieurs vérifications avant d'être utilisé. Seulement une fois les vérifications effectuées, le fichier pourront être déplacés dans un dossier de l'application.

1. Modification de la vue « form_product.html.twig »

Pour commencer, nous devons modifier la vue pour y ajouter un nouveau champ.
Placez le morceau de code suivant dans le formulaire :

```
<div class="mb-3">
        <label for="productImage" class="form-label">Image</label>
        <input type="file" class="form-control" id="productImage" name="productImage">
</div>
```

Puis ajoutons l'attribut « enctype » dans la balise du formulaire :

```
<form action="{{ page }}" method="POST" enctype="multipart/form-data">
```

L'attribut « enctype », indique à PHP que toutes les informations relatives à notre fichier doivent être stockées sous la forme d'un tableau au travers de la variable globale « $_FILES ».

Un nouveau champ devrait maintenant s'afficher, il vous permet de télécharger un fichier qui pour le moment n'est pas pris en compte dans le traitement du formulaire.

2. Ajouter l'envoi de fichier dans le contrôleur

Nous allons maintenant devoir écrire le code nous permettant de manipuler le fichier que nous enverrons dans le formulaire.

Dans la partie qui permet de traiter le formulaire, ajouter le code suivant :

```
$uploads = [
    'extensions' => ['png', 'jpg'],
    'path' => 'uploads/',
    'state' => false
];
$file_name = null;
```

```
if (isset($_FILES["productImage"])) {

    if (!empty($_FILES["productImage"]['name'])) {
        $file_upload = explode(".", $_FILES["productImage"]['name']);

        // Vérification de l'extension
        if (in_array($file_upload[count($file_upload) - 1], $uploads['extensions'])) {

            // Nettoyage des accents
            $file_name = strtr($file_upload[0],
'ÀÁÂÃÄÅÇÈÉÊËÌÍÎÏÒÓÔÕÖÙÚÛÜÝàáâãäåçèéêëìíîïðòóôõöùúûüýÿ',
'AAAAAACEEEEIIIIOOOOOUUUUYaaaaaaceeeeiiiioooooouuuuyy');
            // Nettoyage des caractères spéciaux
            $file_name = preg_replace('/([^.a-z0-9]+)/i', '_', $file_name);
            $file_name = $file_name . '.' . $file_upload[count($file_upload) - 1];

            $file_path = $uploads['path'] . $file_name;

            if (!file_exists($file_path)) {
                // Déplacement du fichier
                move_uploaded_file($_FILES['productImage']['tmp_name'], $file_path);
                $uploads['state'] = true;
            } else {
                $msg = "Une image avec ce nom existe déjà !";
            }

        } else {
            $msg = "L'extension du fichier n'est pas acceptée !";
        }
    } else {
        $msg = "Veuillez choisir un fichier !";
    }
}

if (!empty($label) && !empty($description) && !empty($category)) {
    if ($uploads['state']) {
        $form = [
            'state' => 'success',
            'message' => 'Votre produit a bien été ajouté !'
        ];
    } else {
        $form = [
            'state' => 'danger',
            'message' => $msg
        ];
    }
```

```
    saveProduct($db, $label, $description, $price, $category, $file_name);
} else {
    $form = [
        'state' => 'danger',
        'message' => 'Votre produit n\'a pas été ajouté car les champs obligatoires n\'on pas été remplis !'
    ];
}
```

Explications :

- $uploads est une variable de type tableau que nous allons utiliser pour paramétrer l'envoi de fichier. Ce tableau comprend la clé « extensions » permettant de lister l'ensemble des extensions des fichiers qui sont autorisées à être envoyées. Ici, nous avons limité aux simples extensions « jpg » et « png ». La clé « path » permet de renseigner à quel endroit nous stockerons le fichier. Attention, le fichier n'est pas stocké en base de données, mais physiquement dans un dossier de l'application. Enfin la clé « state » pour définir si l'envoi de fichier s'est bien déroulé ou non. Par défaut, nous considèrerons que l'envoi de fichier est incorrect et ainsi nous pourrons à notre guise passer l'état en vrai.
- $file_name est par défaut défini à « null »
- isset($_FILES["productImage"]) cette ligne vérifie qu'une image est bien envoyée.
- !empty($_FILES["productImage"]['name']) cette ligne vérifie que le fichier envoyé depuis le formulaire n'est pas vide, dans le cas contraire une erreur lui est renvoyée.
- $file_upload = explode(".", $_FILES["productImage"]['name']); cette ligne permet de couper la chaine de caractères à chaque fois qu'un point sera présent. Prenons l'exemple d'une image avec le nom « mon_image.png », la fonction « explode » retournerait un tableau avec une première valeur « mon_image » et une seconde valeur « png ».
- La fonction « in_array » de PHP permet de vérifier la présence d'une valeur dans un tableau. Dans le cas présent, nous vérifions que l'extension du fichier envoyé est bien autorisée.
- $file_name = strtr($file_upload[0], 'ÀÁÂÃÄÅÇÈÉÊËÌÍÎÏÒÓÔÕÖÙÚÛÜÝàáâãäåçèéêëìíîïðòóôõöùúûüýÿ', 'AAAAAACEEEEIIIIOOOOOUUUUYaaaaaaceeeeiiiioooooouuuuyy'); cette instruction remplace tous les caractères accentués par des caractères non accentués. Plus particulièrement la liste du haut remplace la liste du bas.
- $file_name = preg_replace('/([^.a-z0-9]+)/i', '_', $file_name); cette ligne se charge de remplacer les éventuels caractères spéciaux tels que des espaces par des underscore « _ ».
- !file_exists($file_path) l'instruction de la condition s'attarde à vérifier qu'un fichier ayant le même nom n'existe pas. Si aucun fichier du même nom n'est trouvé, l'envoi de fichier est autorisé à continuer.
- Si toutes les conditions sont vérifiées, le déplacement du fichier vers le dossier « uploads » situé dans le dossier « public » est autorisé. Le déplacement s'effectue

avec la fonction « move_uploaded_file() » et l'autorisation est effectuée en passant l'état de la variable « uploads » à vrai.
- Pendant la vérification des éléments du formulaire, il faut ajouter une condition qui vérifie si l'état de l'envoi de fichier est vrai. Selon le cas un message d'erreur ou de succès est renvoyé. Le message d'erreur varie en fonction de la variable « $msg ».
- Pour finir, il faut ajuster l'appel à la fonction « saveProduct » qui ajoute le résultat en base de données. En effet, ici, nous devons lui ajouter le nom de l'image. Si aucune image n'est définie, nous lui enverrons une valeur « null »

3. Créer le dossier de téléchargement

Pour stocker l'ensemble des fichiers du site e-commerce, nous allons créer un dossier « uploads » dans le dossier « public ».

Nous allons définir les permissions sur « 777 ».
Pour cela utilisez la commande : chmod -R 777 uploads/

Enfin, vérifiez que la commande a bien changé les permissions avec : ls -l

Attention :
Sur un serveur de production, il ne faut jamais laisser les permissions en 777. Vous devez correctement configurer votre serveur web pour que les permissions soient en 755.

4. Modification de la base de données

Afin d'assigner une image à chacun de nos produits, nous allons devoir stocker en base de données le chemin vers cette dernière.

Travail à faire :
- Dans PhpMyAdmin, ajouter une nouvelle colonne « image » permettant de stocker une chaine de 255 caractères pouvant être « null ».

Dans le fichier « ProductModel.php » quelques adaptations sont nécessaires afin de prendre en compte le nouveau champ de la table « shop_products » :

```
function getOneProduct($db, $idProduct) {
    $query = $db->prepare("SELECT id, label, `description`, price, idCategory, image
FROM shopcourse_products WHERE id = :id");
    $query->execute([
      'id' => $idProduct
    ]);
    $product = $query->fetch();

    return $product;
  }
```

```
    function getAllProduct($db) {
        $query = $db->prepare("SELECT id, label, `description`, price, idCategory, image
FROM shopcourse_products");
        $query->execute();
        $products = $query->fetchAll();

        return $products;
    }

    function saveProduct($db, $label, $description, $price, $category, $image) {
        $query = $db->prepare("INSERT INTO shopcourse_products (label, `description`,
price, idCategory, image) VALUES (:label, :descr, :price, :category, :image)");
        return $query->execute([
            'label' => $label,
            'descr' => $description,
            'price' => $price,
            'category' => $category,
            'image' => $image
        ]);
    }
```

Vous pouvez dès à présent tester l'envoi de fichier. Tout devrait fonctionner correctement. Dans la base de données, vous pourrez voir apparaitre le nom du fichier que vous avez envoyé.

5. Afficher l'image d'un produit

Nous affichons à plusieurs reprises les produits dans le site e-commerce. Au tout début nous avions choisi d'afficher une image fake par défaut. Maintenant que nous avons un système d'envoi de fichier, nous allons utiliser le nouveau champ de la base de données.

Pour cela nous allons modifier nos vues à deux endroits.

D'abord le fichier « home.html.twig » en modifiant le code de cette manière :

```
<div class="col-md-4">
  {% if product.image %}
      <img src="uploads/{{ product.image }}" class="img-fluid rounded-start" alt="...">
  {% else %}
      <img src="https://fakeimg.pl/350x200/?text=Image" class="img-fluid rounded-start"
alt="...">
  {% endif %}
</div>
```

Explications :

Le morceau de code ci-dessus permet de conditionner l'affichage de l'image. Si une image est définie pour un produit alors elle sera affichée, sinon nous afficherons l'image par défaut.

Travail à faire :

- Faire les mêmes adaptations pour la vue Twig « show_product.html.twig »

6. Conclusion

Pour conclure ce chapitre riche en contenu, voici une petite mise en pratique pour vous faire la main.

- Nous avons actuellement ajouté l'envoi de fichier depuis la page d'ajout d'un produit. Le problème c'est que nous allons également avoir besoin de cette fonctionnalité dans la modification d'un produit. Pour cela, vous allez réaliser une fonction générique qui pourra être réutilisée à plusieurs endroits de votre application.
    - Pour faciliter la gestion de ces fonctions, vous allez créer le dossier « service » dans le dossier « src ».
    - Dans le dossier « service » vous allez créer un fichier que vous nommerez « Upload.php »
    - En vous appuyant de ce que nous venons de faire, vous devez proposer une fonction capable d'envoyer des fichiers.
    - Avant de passer à la suite, assurez-vous que tout fonctionne correctement.
    - La fonction prendra en paramètre le nom du champ dans le formulaire, la liste des extensions et le chemin où stocker le fichier.
- Adaptez et intégrez votre fonction pour qu'un utilisateur puisse modifier l'image de son produit. Attention, dans le cas d'une modification, une image peut déjà avoir été définie, celle-ci ne doit pas être écrasée ni supprimée.

À vous de jouer !