Développement Web

Téléversement d'une image

Table des matières

Guy Verghote

Téléversement d'une image

1. Présentation de l'activité

Dans le cadre de l'étude de la mise en place d'un téléversement de fichier, vous êtes chargé de mettre en place un module web permettant de gérer un ensemble de fichiers image contenant des photos.

Les photos sont stockées dans le répertoire 'data/image'

Le module doit mettre en place une interface permettant :

- La consultation des photos avec une mise en forme de type cadre
- La suppression d'une photo
- L'ajout d'une photo

Lors du téléversement, les images conservent leur nom d'origine avec l'ajout automatique d'un suffixe si un fichier image portant le même nom existe déjà.
Les dimensions (150 sur 150) devront être respectée.

Les fichiers à compléter du module demandé sont stockés dans le répertoire 'uploadimage' de l'application

2. La classes technique InputFileImg

Cette classe dérive de la classe InputFile en venant y ajouter les particularités de fichier image : la largeur, la hauteur et la possibilité de redimensionner l'image.
Pour cela elle nécessite l'installation via composer du composant gumlet/php-image-resize.
Elle surcharge les méthodes checkValidity() et copy() afin de contrôler les dimensions et assurer le redimensionnement le plus approprié.

3. La classe metier FichierImage

Elle centralise les traitements permettant la récupération des fichiers image, l'ajout et la suppression
Elle est configurable à partir de paramètres intégrés directement dans la classe

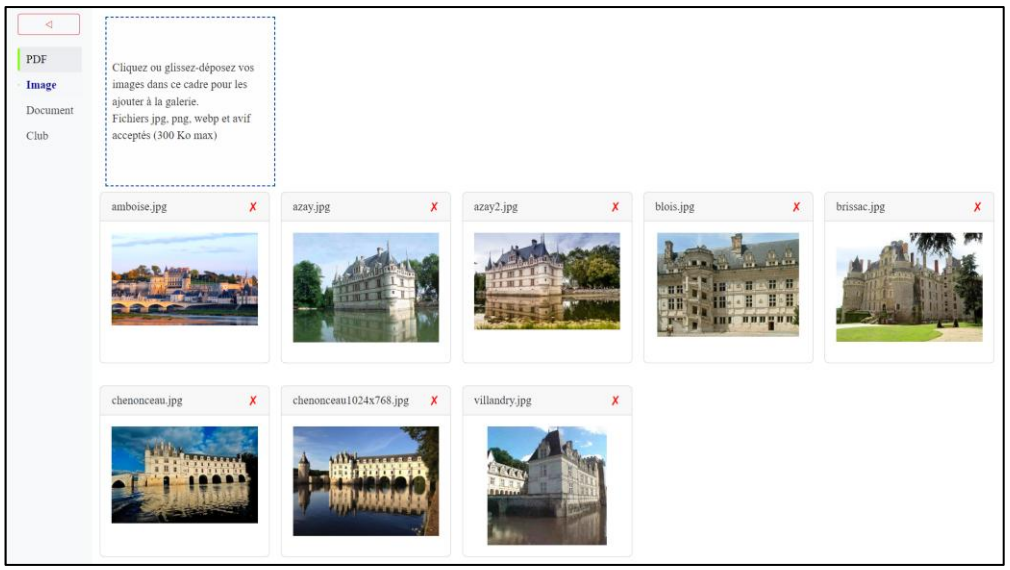
```
private const CONFIG = [
        'repertoire' => '/data/image',
        'extensions' => ["jpg", "png", "webp", "avif"] ,
        'types' => ["image/pjpeg", "image/jpeg", "x-png", "image/png", "image/webp",
"image/avif", "image/heif"],
        'maxSize' => 300 * 1024,
        'require' => true,
        'rename' => true,
        'sansAccent' => true,
        'accept' => '.jpg, .png, .webp, .avif',
        'redimensionner' => true,
        'height'=> 0, // 0 pour ne pas redimensionner
        'width' => 350,
        'label' => 'Fichiers jpg, png, webp et avif acceptés (300 Ko max)',
    ];.
```

Avec cette configuration, le fichier sera renommé avec un suffixe si un fichier de même nom existe déjà, les accents et autres caractères spéciaux seront retirés, l'image sera redimensionnée proportionnellement à sa largeur qui ne dépassera pas 350 px.

Les méthodes getAll() et supprimer($nomFichier) sont strictement identiques à la version 'PDF'
Pour la méthode ajouter(**InputFileImg** $file) seul le type du paramètre change

1. Compléter le script classemetier/fichierimage.pdf

4. Le fichier index.html



La croix permet de supprimer la photo, après une demande de confirmation.
Le style de la zone 'upload' en pointillé sur fond orange est défini dans la feuille de style

```
<div id="cible" class="upload" style="width: 300px; height: 300px;">
   Cliquez ou glissez-déposez vos images dans ce cadre pour les ajouter à la galerie.
   <div id="label"></div>
</div>
<input type="file" id="fichier" style='display:none'>
<div id='lesCartes'></div>
```

Le traitement est très proche du téléversement d'un document. La différence principale repose sur l'utilisation d'un objet **InputFileImg** à la place d'un objet **InputFile.**

Les scripts ajax/ajouter.php et ajax/supprimer.php vont aussi retourner la nouvelle liste de fichiers afin de prendre en compte les éventuelles ajouts ou suppressions des autres utilisateurs.

2. En vous inspirant de la version 'PDF', compléter les scripts 'ajax/ajouter.php' et 'ajax/supprimer.php' et réaliser les tests fonctionnels sous Postman

Url : http://upload/uploadimage/ajax/ajouter.php
Exemple : {"error":{"fichier":"La taille du fichier (354648) dépasse la taille autorisée (307200)"}}

Url : http://upload/uploadimage/ajax/supprimer.php
Exemple : placer un fichier test.txt dans le répertoire data/image et essayer de le supprimer
{"error":{"global":"Extension 'txt' non autorisée pour suppression"}}

5. Le script index.php

Il doit récupérer les paramètres du téléversement, et la liste des photos pour alimenter les variables Javascript 'lesParametres' et 'lesPhotos'.

```
// récupération des paramètres du téléversement
$lesParametres = json_encode(FichierImage::getConfig(), JSON_UNESCAPED_SLASHES |
JSON_UNESCAPED_UNICODE);

// récupération des fichiers PDF
$lesFichiers = json_encode(FichierImage::getAll(), JSON_UNESCAPED_SLASHES |
JSON_UNESCAPED_UNICODE);

$head = <<<EOD
<script>
    const lesFichiers = $lesFichiers;
    let lesParametres = $lesParametres;
</script>
EOD;];
```

3. Compléter le script index.php

6. Le script index.js

Il permet d'alimenter l'interface et de déclencher les demandes d'ajout ou de suppression.

6.1. Les procédures événementielles

Au niveau de l'ajout, les procédures événementielles vont permettre de lancer la sélection d'un fichier et son contrôle côté client. Le glisser déposer est mis en place

```
// Déclencher le clic sur le champ de type file lors d'un clic dans la zone cible
cible.onclick = () => fichier.click();

// // ajout du glisser déposer dans la zone cible
cible.ondragover = (e) => e.preventDefault();
cible.ondrop = (e) => {
   e.preventDefault();
   controlerFichier(e.dataTransfer.files[0]);
};

// Lancer la fonction controlerFichier si un fichier a été sélectionné dans l'explorateur
fichier.onchange = () => {
   if (fichier.files.length > 0) {
      controlerFichier(fichier.files[0]);
   }
};
```

6.2. La fonction controlerFichier

```
function controlerFichier(file) {
   // Efface les erreurs précédentes
   effacerLesErreurs();
   // Vérification de taille et d'extension
   if (!fichierValide(file, lesParametres)) {
      return;
   }
   // si le redimensionnement est demandé, on ne vérifie pas les dimensions
   if (lesParametres.redimensionner) {
      ajouter(file);
   } else {
      // sinon on vérifie les dimensions
       verifierDimensionsImage(file, lesParametres, () => ajouter(file));    }
}
```

Comme pour le téléversement d'un document, elle fait appel à la fonction ficherValide() du module formulaire.js pour vérifier la taille et l'extension.

Pour une image il est possible de vérifier les dimensions en appelant la fonction verifierDimensionImage(). Cette fonction est forcément asynchrone car il faut attendre le chargement de l'image pour pouvoir récupérer les dimensions de l'image et les contrôler par rapport aux dimensions fixées. Elle utilise donc une fonction de rappel en cas de succès. En cas d'erreur le message est affiché sous le champ 'fichier'.

L'appel de la fonction verifierDimensionImage n'est à faire que si l'image ne doit pas être redimensionnée (propriété redimensionner = false)

Il est intéressant d'étudier le code de cette fonction qui prévoit 4 cas:

```
export function verifierDimensionsImage(file, dimensions, { onSuccess, onErreur } = {}) {
   const img = new Image();
   img.src = URL.createObjectURL(file);
   img.onload = () => {
      // Cas 1 : Les deux dimensions (largeur et hauteur) sont renseignées

      // Cas 2 : Seule la largeur est renseignée

      // Cas 3 : Seule la hauteur est renseignée

      // Cas 4 : Aucune dimension n'est renseignée (pas de vérification nécessaire)

   img.onerror = () => {
      onErreur?.("Le fichier n'est pas une image valide.");
      URL.revokeObjectURL(img.src); // Nettoyer aussi en cas d'erreur de chargement
   };
}
```

La fonction verifierDimensionImage réalise le contrôle des dimensions de l'image. Pour cela elle charge l'image en mémoire (dans un objet Image) afin d'accéder à ses propriétés width et height.

Un premier problème se pose : comment alimenter la propriété 'src' de l'objet Image ?
La méthode window.URL.createObjectURL(file)permet de créer un objet URL à partir du fichier sélectionné (file) par l'utilisateur. Cet objet URL représente le contenu du fichier et peut être utilisé comme source pour des éléments HTML tels que les balises <img> ou <video>. Cela permet d'afficher l'image sélectionnée sur l'interface web sans avoir à la télécharger sur le serveur.

```
// création d'un objet image
let img = new Image();
// chargement de l'image
img.src = window.URL.createObjectURL(file);
```

Un second problème se pose : il faut attendre le chargement en mémoire de l'image pour pouvoir accéder aux propriétés width et height.
Pour cela il faut programmer l'événement load de l'objet Image

Le fait de devoir charger l'image en mémoire peut déclencher une erreur de chargement si le fichier transmis n'est pas une image. L'événement 'error' se déclenche et on peut le traiter par sa propriété onerror pour indiquer que le fichier n'est pas valide. Cela permet en quelque sorte de vérifier le type mime du fichier côté client.

6.3. La fonction afficher

La fonction afficher() génère l'affichage des images placées dans des conteneurs de type carte à partir de la lecture du tableau passé en paramètre

```
function afficher(data) {
    // On vide le contenu précédent
    listePhoto.innerHTML = '';

    // Création d'un conteneur flex
    let conteneur = document.createElement('div');
    conteneur.style.display = 'flex';
    conteneur.style.flexWrap = 'wrap';
    conteneur.style.gap = '1rem'; // Espacement entre les cartes
    conteneur.style.justifyContent = 'flex-start';

    // Boucle sur les fichiers
    for (const nomFichier of data) {
        const carte = creerCartePhoto(nomFichier);
        // Limite de taille et comportement responsif
        carte.style.flex = '0 1 300px'; // Ne grandit pas, peut rétrécir, base 300px
        carte.style.maxWidth = '100%';
        conteneur.appendChild(carte);
    }
    listePhoto.appendChild(conteneur);
}
```

Elle utilise le mode flex pour assurer un affichage, adapté selon l'écran, des cartes avec un largeur fixée à 300px.

Téléversement d'une image

La fonction creerCartePhoto(nomFichier) génère une balise de class 'card'

```
unction creerCartePhoto(nomFichier) {
    // Création de la carte principale (div avec classe Bootstrap "card")
    const carte = document.createElement('div');
    carte.classList.add("card", "mb-3"); // "mb-3" ajoute une marge inférieure
    carte.id = nomFichier; // Utilisation du nom du fichier comme ID unique

    // Création de l'entête de la carte (section supérieure)
    const entete = document.createElement('div');
    entete.classList.add("card-header");

    // Création du bouton ✗ pour supprimer l'image
    const btnSupprimer = creerBoutonSuppression(() => confirmer(() =>
supprimer(nomFichier)));
    btnSupprimer.classList.add('float-end'); // Positionné à droite

    // Création d'un élément pour afficher le nom du fichier dans l'entête
    const nom = document.createElement('div');
    nom.classList.add('float-start'); // Positionné à gauche
    nom.innerText = nomFichier;

    // Assemblage de l'entête : bouton et nom
    entete.appendChild(btnSupprimer);
    entete.appendChild(nom);

    // Ajout de l'entête dans la carte
    carte.appendChild(entete);

    // Création du corps de la carte contenant l'image
    const corps = document.createElement('div');
    corps.classList.add("card-body");
    corps.style.height = '250px'; // Hauteur fixe pour uniformiser les cartes

    // Création de l'image à afficher
    const img = document.createElement('img');
    img.src = lesParametres.repertoire + '/' + nomFichier; // Chemin complet de l'image
    img.alt = ""; // Texte alternatif vide pour accessibilité
    img.style.maxWidth = '100%';
    img.style.maxHeight = '100%';
    img.style.width = 'auto';
    img.style.height = 'auto';
    img.style.objectFit = 'contain';

    // Insertion de l'image dans le corps de la carte
    corps.appendChild(img);

    // Ajout du corps dans la carte
    carte.appendChild(corps);

    // Retour de la carte complète prête à être insérée dans le DOM
    return carte;
}
```

6.4. La fonction ajouter

La fonction ajouter() réalise un appel au script ajax/ajouter.php qui va contrôler le fichier, l'ajouter dans le répertoire concerné et renvoyer la liste des images contenues dans le répertoire
Il faut forcément utiliser un objet formData pour assurer le transfert d'un objet File.

```
function ajouter(file) {
    let formData = new FormData();
    formData.append('fichier', file);
    appelAjax({
        url: 'ajax/ajouter.php',
        data: formData,
        success: (data) => {
            afficherToast("La photo a été ajoutée dans la bibliothèque");
            // Mise à jour de l'interface
            afficher(data);
        }
    });
}
```

6.5. La fonction supprimer

La démarche reste la même.

```
function supprimer(nomFichier) {
    effacerLesErreurs();
    appelAjax({
        url: 'ajax/supprimer.php',
        data: {  nomFichier: nomFichier},
        success: (data) => {
                afficherToast("La photo a été supprimée de la bibliothèque");
                // mise à jour de l'interface
                afficher(data);
        }
    });
}
```

6.6. Programme principal

Il alimente l'interface et met en place comme dans les autres activité le menu vertical.

```
configurerFormulaire();
fichier.accept = lesParametres.accept;
label.innerText = lesParametres.label;
afficher(lesFichiers);
```

4. Compléter le script index.js et réaliser les tests fonctionnels de l'application