Développement Web

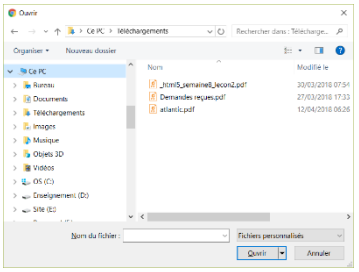
Téléversement d'un fichier

Table des matières

Guy Verghote

Téléversement d'un fichier

1. Le champ input de type file



L'envoi de fichier (upload) s'appuie la mise en place d'une balise input de type 'file'. Ce type permet à l'utilisateur de sélectionner un fichier :

```
<input type='file' id='fichier' accept='.pdf'/>
```

L'attribut accept indique les types de fichiers que le serveur acceptera. La valeur de cet attribut est une liste de valeurs séparées par des virgules, ces valeurs peuvent être :

- Une extension de fichier (".jpg, .png, .doc")
- Un type MIME valide (application/force-download,application/pdf)
- Un type MIME générique
  - audio/* pour des fichiers sonores HTML5
  - video/* pour des fichiers vidéo HTML5
  - image/* pour des fichiers images

L'attribut ne permet absolument pas de garantir que le fichier téléchargé vérifie l'extension ou le type mime. Il ne sert qu'à filtrer les fichiers proposés dans la fenêtre de sélection du fichier.

2. Mise en forme du champ

La mise en forme d'un champ de type file est pris en compte par le navigateur et le résultat est loin d'être joli.

Chrome **Firefox**

Choisir un fichier Aucun fichier choisi Parcourir... Aucun fichier sélectionné.

Pour éviter cela, la solution consiste à cacher le champ de type file et à générer le clic sur ce champ à l'aide d'un bouton plus esthétique et plus significatif.

```
<input type="file" id="fichier" accept="" style='display: none '>
```

Plusieurs représentations sont possibles :

Afficher juste un bouton qui déclenche immédiatement la procédure de téléversement.

Ajouter un fichier PDF de 2 Mo maximum

```
<button class="btn btn-danger w-100 " id="btnAjouter">Ajouter un fichier PDF de 2 Mo maximum</button>
```

Fusionner un bouton et un champ texte pour afficher le nom du fichier téléversé.
Le téléversement s'effectue alors sur le clic du second bouton 'Ajouter'

Parcourir... Choisissez un fichier pdf de 2 Mo max...

```
<label for="nomFichier" class='col-form-label obligatoire'>Fichier txt concernant la course</label>
   <div class="input-group ">
      <button id="btnFichier" class="btn btn-sm btn-outline-secondary">Parcourir...</button>
      <input type="text" id="nomFichier" placeholder = 'Choisissez un fichier .txt' class="form-control" readonly/>
   </div>
   <input type="file" id="fichier" accept="" style='display: none '>
```

3. Les traitements côté serveur

Coté serveur, le fichier est initialement conservé sur le serveur Web dans un espace temporaire (défini dans le fichier php.ini (upload_tmp_dir) sous un nom arbitraire. Les informations relatives au fichier transféré sont accessibles par le tableau associatif (variable super globale) **$_FILE.**
La valeur de la clé d'accès correspond à la valeur de l'attribut id de la balise input de type file correspondant (ici $_FILES['fichier']).
La valeur associée est aussi un tableau associatif un tableau contenant les informations suivantes :

$_FILES['fichier']['tmp_name']	Nom et chemin du fichier temporaire sur le serveur
$_FILES['fichier']['name']	Nom du fichier sur le poste client
$_FILES['fichier']['size']	Taille du fichier en octets
$_FILES['fichier']['type']	Type MIME du fichier : 'text/html' pour un fichier texte, 'image/gif' pour un fichier image au format gif.
$_FILES['fichier']['error']	Code erreur (0 : pas d'erreur, 1 : taille > upload_max_filesize , 2 : taille > max_file_size, …)

Code erreur : https://www.php.net/manual/fr/features.file-upload.errors.php

```
UPLOAD_ERR_OK
Valeur : 0. Aucune erreur, le téléchargement est correct.
UPLOAD_ERR_INI_SIZE
Valeur : 1. La taille du fichier téléchargé excède la valeur de upload_max_filesize, configurée dans le php.ini.
UPLOAD_ERR_FORM_SIZE
Valeur : 2. La taille du fichier téléchargé excède la valeur de MAX_FILE_SIZE, qui a été spécifiée dans le formulaire HTML.
UPLOAD_ERR_PARTIAL
Valeur : 3. Le fichier n'a été que partiellement téléchargé.
UPLOAD_ERR_NO_FILE
Valeur : 4. Aucun fichier n'a été téléchargé.
UPLOAD_ERR_NO_TMP_DIR
Valeur : 6. Un dossier temporaire est manquant.
UPLOAD_ERR_CANT_WRITE
Valeur : 7. Échec de l'écriture du fichier sur le disque.
```

Attention : Le type mime renvoyé correspond à celui de l'extension. Il n'est donc pas utilisable pour vérifier le véritable type mime du fichier.

Il faut commencer par vérifier le fichier téléversé afin d'éviter toute faille de sécurité ("Ne jamais faire confiance aux données transmises").
Les contrôles doivent porter sur l'extension du fichier, son type mime et sa taille.

La taille maximale, les extensions acceptées et les types mimes acceptés sont définis en fonction de l'application. Ces paramètres peuvent être définis directement dans le code ou placé dans un fichier de configuration.

Le premier contrôle consiste à vérifier qu'un fichier a bien été transmis :

```
if (!isset($_FILES['fichier'])) {
    echo json_encode(['error' => "Aucun fichier transmis"]);
    exit;
}
```

Le second qu'il n'y a pas eu d'erreur lors du transfert :

```
if ($_FILES['fichier']['error'] !== 0) {
    echo json_encode(['error' => "Aucun fichier reçu"]);
    exit;
}
```

On peut ensuite récupérer les données sur le fichier :

```
// récupération des données transmises
$tmp = $_FILES['fichier']['tmp_name'];
$nomFichier = $_FILES['fichier']['name'];
$taille = $_FILES['fichier']['size'];
```

Pour vérifier sa taille, il suffit de la comparer à la taille maximale autorisée :

```
$tailleMax = 2 * 1024 * 1024;
if ($taille > $tailleMax) {
    echo json_encode(['error' => "La taille du fichier dépasse la taille autorisée"]);
    exit;
}
```

Pour vérifier l'extension du fichier :

```
$lesExtensions = ["pdf"];
$extension = strtolower(pathinfo($nomFichier, PATHINFO_EXTENSION));
if (!in_array($extension, $lesExtensions)) {
    echo json_encode(['error' => "Extension du fichier non acceptée"];
    exit;
}
```

Pour vérifier le type mime du fichier :

```
$lesTypes = ["application/pdf"];
$type = mime_content_type($tmp);
if (!in_array($type, $lesTypes)) {
    echo json_encode(['error' => "Type de fichier non accepté"];
    exit;
}
```

Le type MIME (ou MIME type, pour Multipurpose Internet Mail Extensions) est une chaîne de texte qui indique la nature et le format d'un fichier. Il est utilisé pour que les applications (navigateurs, serveurs web, etc.) sachent comment traiter un fichier ou un contenu.
La liste des différents types possibles est normalisée. Chaque type est défini par l'association d'un type général (application, audio, image, son, vidéo, texte) et d'un sous-type qui indique le format exact du fichier.

Les principaux type MIME(Multipurpose Internet Mail Extensions)

```
application/force-download
application/pdf
application/msword
application/vnd.openxmlformats-officedocument.wordprocessingml.document
application/excel,application/vnd.ms-excel
application/x-excel,application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
text/plain ou text/html
application/octet-stream
```

La plupart des fichiers ne contiennent pas explicitement leur type MIME dans leur contenu. Il est déduit à partir de l'extension du fichier (.jpg, .html, etc.), ou détecté via l'analyse du contenu binaire (appelé magic number).

Par exemple, un fichier .jpg ne contient pas une ligne disant image/jpeg, mais il commence par une séquence binaire (FFD8FFE0...) qui identifie un fichier JPEG.
En PHP, il est possible de tester le type mime avec l'outil FileInfo

```
$finfo = finfo_open(FILEINFO_MIME_TYPE);
$type = finfo_file($finfo, $tmp);
finfo_close($finfo);
```

Si le fichier respecte tous ces contrôles, il peut alors être copié sur le serveur en vérifiant préalablement qu'il n'existe pas déjà.

```
// copie sur le serveur
copy($tmp, '../document/' . $nomFichier);
```

L'utilisation de l'instruction copy permet de travailler avec un serveur sur une unité logique (c:) et l'application web sur une autre unité logique (j: dans notre cas)

Lorsque le serveur web et l'application web se trouve sur la même unité logique on peut utiliser :

- La fonction rename($source, $destination) qui déplace et renomme le fichier
- La fonction move_uploaded_file($source, $destination).

La fonction remane renvoie une erreur si le fichier $destination existe déjà, alors que la fonction move_uploaded_file ecrase le fichier existant.

Le fichier temporaire est supprimé automatiquement à la fin du script.

Si l'on veut garantir l'unicité du nom du fichier on peut choisir de renommer le fichier en cas de doublon. Il suffit pour cela de lui ajouter un suffixe :

Par exemple : test.pdf devient test(1).pdf si test.pdf existe déjà :

```
$nom = pathinfo($nomFichier, PATHINFO_FILENAME);
$i = 1;
while (file_exists( REP_DOCUMENT . $nomFichier)) {
   $nomFichier = "$nom(" . $i++ . ").$extension";
}
```

Le traitement sur le fichier peut être différent, en fonction de l'application et ne nécessite pas forcément d'être conservé côté serveur.
On peut par exemple, utiliser un fichier Excel, ou un fichier Csv pour alimenter les données d'une table. Dans ce cas les contrôles sur le fichier sont plus importants, il faut notamment vérifier sa structure.

3.1. Utilisation de la classe InputFile pour simplifier la procédure de téléversement

Afin de simplifier le travail du programmeur, nous avons mis en place une classe **InputFile** permettant de prendre en charge l'ensemble des traitements (contrôle, et téléversement)

Il suffit alors d'instancier un objet de cette classe et d'appeler la méthode **checkValidity()** pour réaliser l'ensemble des contrôles.

Le constructeur prend en paramètre un tableau contenant l'ensemble des règles de validation.

Exemple du contenu du tableau :

```
[
   'repertoire' => '/data/document',      → répertoire dans lequel le fichier sera téléversé
   'extensions' => ['pdf'],               → tableau des extensions autorisées
   'types' => ["application/pdf"],        → tableau des types mime autorisés
   'maxSize' => 1024 * 1024,              → taille maximale autorisé en octet (soit 1 Mo ici)
   'require' => true,                     → le fichier est obligatoire
   'rename' => false,                     → le fichier conservera son nom
   'sansAccent' => false,                 → les accents seront aussi conservés
   'accept' => '.pdf'                     → utilisé pour renseigner l'attribut 'accept' de la balise file
   'label' => 'Fichier PDF (1 Mo max)'    → utilisé pour renseigner le label associé à la balise file
]
```

En cas d'erreur, la méthode **getValidationMessage()** permet de récupérer le message d'erreur associé
En l'absence d'erreur, il faut appeler la méthode **copy()** afin de recopier le fichier téléversé dans son répertoire de stockage.

Important : Pour pouvoir utiliser la classe InputFile, le champ input de type file doit est transmis en utilisant le nom 'fichier' ($_FILES['fichier'])

3.2. Mise en place du téléversement

Afin d'éviter de répéter l'ensemble de la procédure dans chaque application, une classe est souvent utilisée.
Cette classe encapsule le tableau des paramètres dans une propriété privée de la classe CONFIG et le rend accessible depuis l'extérieur de la classe à partir de la méthode **getConfig()**.

Elle contient une méthode qui va prendre en charge la totalité de la procédure.

Exemple sur le téléversement d'un fichier PDF : la classe **FichierPDF**

```
class FichierPDF  {
   private const CONFIG = [
      'repertoire' => '/data/pdf',
      'extensions' => ['pdf'],
      'types' => ["application/pdf"],
      'maxSize' => 512 * 1024, // 512 Ko
      'require' => true,
      'rename' => false,
      'sansAccent' => true,
      'accept' => '.pdf',
      'label' => 'Fichier PDF à téléverser (512 Ko max)',
   ];

   private const DIR = RACINE . self::CONFIG['repertoire'];

   public static function ajouter(): array    {
      // instanciation et paramétrage d'un objet InputFile : y compris le $_FILES[]
      $file = new InputFile(self::CONFIG);
      // vérifie la validité du fichier en teant compte des paramètres de configuration
      if ($file->checkValidity()) {
         // copie du fichier en prenant en compte les paramètres de configuration
         if ($file->copy()) {
            return [
               'success' => true,
               'message' => "Le fichier a été téléversé avec succès"
            ];
         } else {
            return [
               'success' => false,
               'message' => "Le fichier n'a pas pu être téléversé"
            ];
         }
      } else {
         return [
            'success' => false,
            'message' => $file->getValidationMessage()
         ];
      }
   }
   …
}
```

4. Les traitements côté client

Le document téléversé (un objet file) doit être contrôlé puis envoyé vers le serveur.
La variable globale **leFichier** va stocker cet objet

```
let leFichier = null; // contient le fichier téléversé
```

Un champ de type 'file' comporte une propriété files qui retourne un objet FileList contenant un tableau d'objet file. Il est effectivement possible de sélectionner plusieurs fichiers depuis un champ de type 'file' en utilisant l'attribut multiple

Un objet file possède plusieurs propriétés :
- name : le nom du fichier avec son extension,
- size : la taille en bytes du fichier
- type : le type MIME (ex: application/pdf) du fichier

La vérification peut porter sur la taille du fichier et son extension.
Le contrôle du type mime n'est pas réellement efficace car la propriété type retourne toujours le type correspondant à l'extension du fichier, or un pirate peut changer l'extension du fichier sans problème.

Les paramètres du téléversement auront été transmis par le point de terminaison (index.php)

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

$head = <<<HTML
<script>
    const lesParametres = $lesParametres;
</script>
HTML;

 // chargement de l'interface
require RACINE . "/include/interface.php";
```

L'attribut accept du champ de type 'file' est alimenté à l'aide du tableau lesParametres :

```
const fichier = document.getElementById('fichier');
fichier.accept = lesParametres.accept;
```

La fonction **fichierValide**(file, controle) de la bibliothèque formulaire.js automatise la phase de contrôle.
'contrôle' est un objet contenant dans les propriétés suivantes :
- taille : pour indiquer éventuellement la taille à ne pas dépasser
- lesExtensions : tableau contenant les extensions autorisées
- reponse : message indiquant la raison de l'échec du contrôle

Cette méthode retourne un booléen indiquant la réussite ou l'échec des contrôles.
La raison de l'échec est retournée dans la propriété 'reponse' de l'objet contrôle (c'est une façon de retourner un second résultat lors de l'appel d'une fonction)

```
// Déclencher le clic sur le champ de type file lors d'un clic sur le bouton btnFichier
btnFichier.onclick = () => fichier.click();

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

btnAjouter.onclick = () => {
    if (leFichier === null) {
       afficherErreurSaisie('fichier', 'Veuillez sélectionner ou faire glisser un fichier');
   } else {
       ajouter();
   }
}
```

Il est possible de mettre en place le glisser déposer au niveau du champ nomfichier

```
nomFichier.ondragover =  (e) => e.preventDefault();
nomFichier.ondrop =  (e) => {
     e.preventDefault();
     controlerFichier(e.dataTransfer.files[0]);
}
```

La fonction de contrôle prend la forme suivante :

```
function controlerFichier(file) {
   effacerLesErreurs();
   if (fichierValide(file, lesParametres)) {
       nomFichier.textContent = file.name;
       leFichier = file;
   } else {
       leFichier = null;
       nomFichier.textContent = '';
   }
}
```

La fonction alimente la variable leFichier si l'objet file respecte les contrôles.
D'autres contrôles peuvent venir s'insérer à la suite, un contrôle d'unicité par exemple si on dispose d'un tableau contenant la liste des documents téléversés

4.1. L'envoi du fichier : la fonction ajouter()

Le transfert du fichier sélectionné vers le serveur se réalise à l'aide d'un appel Ajax synchrone afin d'éviter des clics répétés sur le bouton 'ajouter'.
Il faut passer en paramètre l'objet file représentant le fichier sélectionné.
Cet objet file ne peut être transmis qu'en utilisant un objet FormData.

```
function ajouter() {
   effacerLesErreurs();
   // transfert du fichier vers le serveur dans le répertoire sélectionné
   const formData = new FormData();
   formData.append('fichier', leFichier);
   appelAjax({
      url: 'ajax/ajouter.php',
      data: formData,
      success: (data) => {
            afficher(data);
      }
   });
```

Ici l'appel retourne la liste des fichiers.
Cela permet de mettre à jour l'interface en affichant tous les fichiers
Avantages : si plusieurs personnes ajoutent des fichiers, ils peuvent voir les ajouts des autres.

5. Cas particulier pour les fichiers image

Le téléversement d'une image n'est qu'un cas particulier de téléversement. Lorsqu'on téléverse une image, il est possible d'ajouter un contrôle portant sur les dimensions de l'image et éventuellement de prendre en charge le redimensionnement de l'image source.

5.1. Mise en forme du champ

On utilise souvent une zone permettant de glisser une image ou de télécharger une image. Cette zone pouvant servir à prévisualiser l'image si le téléversement n'est pas déclenché immédiatement.


```
<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'>
```

La classe 'upload' est définie dans la feuille de style.

5.2. Les traitements côté serveur

Lors du téléversement d'une image, en plus des contrôles vus précédemment, il est possible de contrôler les dimensions et aussi de redimensionner l'image.

La classe **InputFileImg** qui dérive de la classe InputFile prend en charge ces possibilités

Cette classe comporte 3 propriétés supplémentaires : Height, Width et Redimensionner (true/false)

Le redimensionnement est pris en charge par le composant Gumlet\ImageResize

Ce composant s'intalle à l'aide de composer : composer require gumlet/image-resize
Rappel : l'interpréteur PHP doit être défini dans les paramètres de l'IDE et 'composer' doit être installé et configuré sur l'IDE.

Les paramètres que doit respecter le fichier téléversé sont stockés dans un tableau

```
[
   'repertoire' => '/data/image',
   'extensions' => ["jpg", "png", "webp", "avif"] ,
   'types' => ["image/pjpeg", "image/jpeg", "x-png", "image/png", "image/webp",  "image/avif", "image/heif"],
   'maxSize' => 150 * 1024,
   'require' => true,
   'rename' => false,
   'sansAccent' => false,
   'redimensionner' => true,
   'height'=> 150,
   'width' => 150
]
```

Il suffit alors d'instancier et de paramétrer un objet de cette classe et d'appeler la méthode **checkValidity()** pour réaliser l'ensemble des contrôles.

En cas d'erreur, la méthode **getValidationMessage()** permet de récupérer le message d'erreur associé
En l'absence d'erreur, il faut appeler la méthode **copy()** afin de recopier le fichier téléversé dans son répertoire de stockage.

Comme pour le téléversement d'un document, une classe est souvent utilisée pour prendre en charge le traitement.
Cette classe encapsule le tableau des paramètres dans une propriété privée de la classe CONFIG et le rend accessible depuis l'extérieur de la classe à partir de la méthode **getConfig()**.

Elle contient une méthode qui va prendre en charge la totalité de la procédure.

Téléversement d'un fichier

Exemple sur le téléversement d'un fichier image : la classe FichierImage

```
class FichierImage {
    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,
        'width' => 350,
        'label' => 'Fichiers jpg, png, webp et avif acceptés (300 Ko max)',
    ];

    private const DIR = RACINE . self::CONFIG['repertoire'];

    public static function getConfig(): array  {
        return self::CONFIG;
    }

    public static function ajouter(): array   {
        // instanciation et paramétrage d'un objet InputFile : y compris le $_FILES[]
        $file = new InputFileImg(self::CONFIG);
        // vérifie la validité du
        if ($file->checkValidity()) {
            // copie du fichier en procédant éventuellement au redimensionnement
            if ($file->copy()) {
                return [
                    'success' => true,
                    'message' => "Le fichier a été téléversé avec succès"
                ];
            } else {
                return [
                    'success' => false,
                    'message' => "Le fichier n'a pas pu être téléversé"
                ];
            }
        } else {
            return [
                'success' => false,
                'message' => $file->getValidationMessage()
            ];
        }
    }
    …
}
```

5.3. Les traitements côté client

Les contrôles sont exactement les mêmes que pour un fichier document.
Il est éventuellement possible de vérifier les dimensions de l'image en utilisant la fonction verifierDimensionsImage(file, lesParametres, onSuccess = {}) du module formulaire.js

```
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));
    }
}
```

La fonction ajouter() reste identique, elle peut varier au niveau de traitement en cas de succès

```
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);
        }
    });
}
```