TP2 – Gestion de fichiers PHP (Upload)

Ce TP de programmation visent à confirmer un socle de connaissances de base, à réutiliser les concepts, des notions et des modèles vus en première année de BTS. Ce sont des étapes indispensables pour aborder d'autres apprentissages. Ce TP est conduit de manière non-guidée pour vous placer le plus souvent dans une situation d'investigation et de renforcement de vos connaissances.

Les objectifs de ce TP sont de continuer le travail du TP précèdent : la gestion des données issues d'un formulaire, de gérer l'envoi de fichier, et la lecture de celui-ci côté serveur. Vous allez devoir rédiger un compte-rendu sous forme de grille de tests unitaire pour gérer l'avancement de votre TP.

Présentation

L'objectif de ce deuxième TP est de réaliser une application web permettant l'envoi de fichier (ici de configuration), et d'en lire certains éléments.

Vous travaillerez avec vos logiciels préférés et les fichiers devront être déposés dans votre répertoire personnel, sur le serveur web de la section.

Etape 1 : Page du formulaire

Vous allez devoir réaliser une page HTML « upload.html » structurée (avec conteneur HTML5) qui contient tout le nécessaire pour envoyer un fichier (balise form, input type file, etc…). Dès l'instant où votre formulaire propose aux visiteurs d'envoyer un fichier, il faut ajouter l'attribut **enctype="multipart/form-data"** à la balise **<form>** .

```
<form action="cible_envoi.php" method="post" enctype="multipart/form-data">
    <p>Formulaire d'envoi de fichier</p>
</form>
```

Grâce à **enctype** , le navigateur du visiteur sait qu'il s'apprête à envoyer des fichiers.

Les éléments <input> dont l'attribut type vaut **"file"** permettent à un utilisateur de sélectionner un ou plusieurs fichiers depuis leur appareil et de les *uploader* vers un serveur.

Documentation : https://developer.mozilla.org/fr/docs/Web/HTML/Element/Input/file

Attention : Vous n'acceptez qu'en upload que des fichiers de types « json » ou « xml »

Exemple :

```
<label for="avatar">Choisissez une image d'avatar:</label>

<input type="file"
       id="avatar" name="avatar"
       accept="image/png, image/jpeg">

```

Etape 2 : Script de traitement

Votre application devra comporter un script de traitement « traitement.php » qui s'occupera de traiter le fichier uploadé.

Pour traiter votre fichier « uploadé » vous allez devoir manipuler un des tableaux superglobaux disponibles dans PHP **: $_FILES**… Mais le traitement des fichiers uploader est particulier…
En fait, au moment où la page PHP s'exécute, le fichier a été envoyé sur le serveur mais il est stocké dans un dossier temporaire. C'est à vous de décider si vous acceptez définitivement le fichier ou non.

Vous pouvez par exemple vérifier si le fichier a la bonne extension (si vous demandiez une image et qu'on vous envoie un « .txt », vous devrez refuser le fichier). Dans notre cas, nous devons vérifier si c'est bien un fichier JSON ou XML.

Si le fichier est bon, vous l'accepterez grâce à la fonction PHP : **move_uploaded_file** , et ce, d'une manière définitive.

Comment faire ? Pour chaque fichier envoyé, une variable **$_FILES['nom_du_champ']** est créée.

Remarque : Dans notre exemple précèdent, la variable s'appellera **$_FILES['avatar']** .

Cette variable est un tableau qui contient plusieurs informations sur le fichier :

Variable	Signification
$_FILES['monfichier']['name']	Contient le nom du fichier envoyé par le visiteur.
$_FILES['monfichier']['type']	Indique le type du fichier envoyé. Si c'est une image gif par exemple, le type sera image/gif .
$_FILES['monfichier']['size']	Indique la taille du fichier envoyé. **Attention** : cette taille est en octets. Il faut environ 1 000 octets pour faire 1 Ko, et 1 000 000 d'octets pour faire 1 Mo. **Attention** : la taille de l'envoi est limitée par PHP. Par défaut, impossible d'uploader des fichiers de plus de 8 Mo.
$_FILES['monfichier']['tmp_name']	Juste après l'envoi, le fichier est placé dans un répertoire temporaire sur le serveur en attendant que votre script PHP décide si oui ou non il accepte de le stocker pour de bon. Cette variable contient l'emplacement temporaire du fichier (c'est PHP qui gère ça).
$_FILES['monfichier']['error']	Contient un code d'erreur permettant de savoir si l'envoi s'est bien effectué ou s'il y a eu un problème et si oui, lequel. La variable vaut 0 s'il n'y a pas eu d'erreur.

Travail à faire : Pour réaliser la procédure veuillez écrire le script de traitement suivant :

1. Vérifier que le fichier a bien été envoyé via la fonction **isset()**
2. Vérifier si la taille du fichier ne dépasse pas 100Ko.
3. Vérifier si l'extension de fichier est bien un « .json » ou un « .xml », en utilisant la variable **$_FILES['monfichier']['name']**
4. Enfin seulement si toutes les étapes sont respectées, veuillez copier de manière définitive le fichier sur le serveur

Si tout est bon, on accepte le fichier en appelant **move_uploaded_file()**. Cette fonction prend deux paramètres :

- le nom temporaire du fichier : on l'a avec **$_FILES['monfichier']['tmp_name']**;

- le chemin qui est le nom sous lequel sera stocké le fichier de façon définitive. On peut utiliser le nom d'origine du fichier **$_FILES['monfichier']['name']** ou générer un nom au hasard.

On gardera le même nom de fichier que celui d'origine. Comme **$_FILES['monfichier']['name']** contient le chemin entier vers le fichier d'origine (ex : C:\dossier\config.json) , par exemple), il nous faudra extraire le nom du fichier. On peut utiliser pour cela la fonction **basename** qui renverra juste « config.json ».

Votre appel de fonction devra ressembler à cela :

```
15                  move_uploaded_file($_FILES['monfichier']['tmp_name'], 'uploads/' .
    basename($_FILES['monfichier']['name']));
```

Attention : Il faut au préalable avoir créé le répertoire uploads, et lui avoir correctement attribué les droits nécessaires pour y copier un fichier.

Remarque : Vous allez devoir faire appel à des fonctions de manipulation de chaines de caractères, vous les avez peut-être vues l'année dernière en C++, celles-ci existent aussi en PHP, et elles portent souvent le même nom… veuillez consulter la documentation suivante : https://www.php.net/manual/fr/ref.strings.php

Documentation :

- https://www.php.net/manual/fr/function.move-uploaded-file.php
- https://www.php.net/manual/fr/ref.strings.php
- https://www.php.net/manual/fr/ref.filesystem.php

Etape 3 : Lecture du fichier uploadé

La manipulation de fichier en php est assez aisée car il existe des fonctions particulières permettant de lire tout le contenu en variable. Vous retrouvez également les fonctions de manipulation « bas niveau » que vous avez vus en C ou C++. (fopen, fread, fwrite, etc…).

Dans notre cas, vous avez deux cas de figures :

- Lecture d'un fichier XML
- Lecture d'un fichier JSON

Le type de fichier étant déterminé avant lors du contrôle du fichier, nous pouvons nous en servir pour déterminer comment lire le fichier…

a) Le fichier JSON

Rien de plus simple en JSON : il suffit de lire tout le contenu du fichier via la fonction « **file_get_contents** » dans une variable et de travailler avec cette variable en décodant la chaine JSON. Comme par exemple pour obtenir le nom de la première personne:

```php
// chemin d'accès à votre fichier JSON
$file = 'noms.json';
// mettre le contenu du fichier dans une variable
$data = file_get_contents($file);
// décoder le flux JSON
$obj = json_decode($data);
// accéder à l'élément approprié
echo $obj[0]->name;
```

```json
[{
  "name" : "Alex",
  "age" : "25",
  "address" : "Paris"
}, {
  "name" : "Emily",
  "age" : "18",
  "address" : "Toulouse"
}, {
  "name" : "Thomas",
  "age" : "22",
  "address" : "Lile"
}]
```

Résultat attendu : Alex

Dans notre cas, la seule difficulté sera d'aller chercher l'information dans le tableau « obj »...

Travail à faire : Continuer le traitement, en testant avec le fichier de configuration : **config1.json** et le fichier **config2.json** et afficher toutes les adresses IP « **interfaces** » de chaque machines « **nodes** ». Copiez le résultat obtenu pour chacun des fichiers.

Bonus : A partir des informations du fichier de configuration, déterminer les adresses **réseau** de chaque interfaces.

Rappel : Adresse réseau = adresse machine **ET** masque

Documentation :

- https://www.php.net/manual/fr/function.json-decode.php
- https://fr.wikipedia.org/wiki/JavaScript_Object_Notation
- https://www.php.net/manual/fr/function.file-get-contents.php
- https://www.php.net/manual/fr/ref.filesystem.php

b) Le fichier XML

Pour le fichier XML c'est un peu plus compliqué car il existe plusieurs façons de faire qui peuvent dépendre des fonctionnalités attendues (recherche d'information, mise à jour de données, etc). Dans notre cas, on se contentera uniquement d'utiliser de lire le fichier XML et de l'exploiter comme le JSON sous forme de tableau objet.

Pour cela nous avons besoin de la fonction **simplexml_load_file**, que nous chargeons dans une variable **$xml**, puis nous allons chercher le nom de la première personne :

```php
// chemin d'accès à votre fichier JSON
$file = 'noms.xml';
// mettre le contenu du fichier dans une variable
$xml = simplexml_load_file($file);
// accéder à l'élément approprié
echo $xml->row[0]->name;
```

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <row>
    <name>Alex</name>
    <age>25</age>
    <address>Paris</address>
  </row>
  <row>
    <name>Emily</name>
    <age>18</age>
    <address>Toulouse</address>
  </row>
  <row>
    <name>Thomas</name>
    <age>22</age>
    <address>Lile</address>
  </row>
</root>
```

Résultat attendu : Alex

Documentation :

- https://www.php.net/manual/fr/function.simplexml-load-file.php
- https://www.php.net/manual/fr/ref.filesystem.php

Travail à faire : Mettre à jour votre traitement, en testant avec le fichier de configuration : **config1.xml** et le fichier **config2.xml** et afficher toutes les adresses IP « **interfaces** » de chaque machines « **nodes** ». Copiez le résultat obtenu pour chacun des fichiers. Puis comparer avec la lecture des fichiers JSON.