Transfert de fichiers vers le serveur

Ce TP doit permettre gérer le téléchargement des images des films vers le serveur.

Activité 1 [7 min]

/apps/templates/add-film.tpl

Il est possible, à partir du navigateur client, de télécharger (téléverser en français) un fichier vers le serveur.
Il faut, pour cela, mettre en place un formulaire avec une balise *input* de type *file*.
L'objectif (à la fin de l'activité) est d'ajouter, pour chaque film, la possibilité de **télécharger une image poster**.
L'image devra être placée au final dans le dossier serveur :

`I:/_xampp/htdocs/dev.fande.fr/assets/img/films`.

Voici un schéma qui illustre le principe :

On suppose que le client va choisir de télécharger un fichier nommé poster.png

Une fois téléchargé, le fichier est placé par le serveur APACHE dans un **dossier temporaire** du serveur.
Toutes les informations relatives au fichier transféré sont disponibles dans la variable globale **$_FILES**.

Complétez le code du template `add-film.tpl` comme suit :

```
<form action="{$smarty.const.ROOT}add-film-save" method="POST" enctype="multipart/form-data" >
  ...
  <div class="form-group">
    <label>Fichier poster </label>
    <input type="file" name="poster"  />
  </div>
  <button type="submit" class="btn btn-default">Ajouter</button>
  ...
</form>
```

Remarques:

- assurez-vous que votre formulaire de téléchargement de fichier contienne *enctype="multipart/form-data"*, sinon, le fichier se sera pas téléchargé.
- c'est la balise de type="file" qui permet de choisir le fichier local (sur le poste navigateur) à transférer (un bouton 'parcourir …' permet de rechercher le fichier en local)

Dans ce code d'illustration, c'est la commande `add-film-save` qui devra traiter la réception du fichier.
Quand le transfert est effectué, automatiquement, le serveur met à votre disposition un tableau associatif $_FILES (avec 5 clés) qui vous permet de manipuler le fichier transféré. Voici l'organisation de ce tableau dans le cas où la balise input file a comme attribut name= "poster" :

$_FILES["poster"]["name"] :: le nom original du fichier,
$_FILES["poster"]["type"] :: le type MIME du fichier (exemple: *"image/png"*)
$_FILES["poster"]["size"] :: la taille, en octets, du fichier téléchargé
$_FILES["poster"]["tmp_name"] :: le nom temporaire du fichier qui sera chargé sur la machine serveur
$_FILES["poster"]["error"] :: le code d'erreur associé au téléchargement de fichier (0=pas de problème)

Activité 2 [15 min]

/apps/controllers.php

Dans le TP4 précédent, le traitement de la commande `add-film-save` a directement été écrit dans le routeur `index.php`. Beaucoup de modifications vont maintenant être ajoutées, et il faut remettre en place une organisation plus classique avec une fonction contrôleur. La fonction contrôleur doit recevoir 2 paramètres : les données à ajouter et un tableau sur les informations du fichier uploadé.
En conséquence, ajoutez une nouvelle fonction `addfilmsave_action()` dans le contrôleur.
Dans le routeur, écrivez (ou modifiez) :

```
elseif ( 'add-film-save' == $command  )
{
  addfilmsave_action( $_POST, $_FILES );
}
```

Modifiez le code de cette fonction du contrôleur associée en écrivant (tout le reste du code déjà écrit dans la fonction sera mis en commentaire) :

```
function addfilmsave_action( $datas, $file ) {
  debug($file, 'Informations de débogage');
}
```

Répondez aux questions suivantes (**les réponses dépendent totalement de la manipulation que vous venez de faire sur VOTRE machine**) :

- Quel est le nom du fichier reçu?

  .....

- Quel est le type du fichier reçu?

  .....

- Quel est le nom temporaire du fichier sur le serveur

  .....

- Quelle est la taille du fichier en octets?

  .....

Pour enregistrer le fichier, nous avons besoin de connaitre le chemin absolu vers notre projet.
Ajouter la constante ROOT_DIR dans le fichier des constantes :

```
define('ROOT_DIR', __DIR__ );
```

Modifiez le code de traitement de `addfilmsave_action()` :

```
$dir2save = ROOT_DIR . '/assets/img/films';
debug( $dir2save, "dossier d'enregistrement" );
$image_source = $file['poster']['tmp_name'];
$image_dest = "$dir2save/{$file['poster']['name']}";
if (move_uploaded_file( $image_source, $image_dest )) {
    debug( "Le fichier est valide; il a été téléchargé avec succès. \n" );
} else {
    debug( " PROBLÈME pendant le téléchargement du fichier!!\n" );
}
```

Testez et répondez aux questions (en recherchant dans la doc du PHP) :

- Que fait l'instruction `move_uploaded_file`?

  .....

- Où est enregistré, au final, le fichier sur le serveur?

  .....

Activité 3 – évolution 1 - [8 min]

/apps/controllers.php

Coté serveur, dans votre script, il faut faire plusieurs tests avant de placer l'image dans son dossier final.
La première vérification consiste à s'assurer que le format de l'image est bien de type **.png**; tout autre format sera rejeté.

Remarques:

- il faut tester la valeur de la clé **$_FILES["poster"]["type"]** et s'assurer qu'elle est bien égale à ' image/png'
- cette information correspond au type mime du fichier; regardez sur http://fr.wikipedia.org/wiki/Type_MIME les types mime des différents formats de fichier

Si le format n'est pas correct, aucun traitement n'est effectué.
Ajoutez ce test dans `addfilmsave_action()` :

```
if ( .. à compléter ... ) {
  $dir2save = ROOT_DIR . '/assets/img/films';
  debug( $dir2save, "dossier d'enregistrement" );
  ...
}
```

Activité 4 – évolution 2 - [20 min]

/apps/controllers.php

La taille du fichier doit être modifiée; la **largeur** du fichier final (celui qui est déposé dans `/assets//img/films`) doit être de **800** pixels.
Le rapport largeur/hauteur de l'image initiale doit cependant être **conservé**.
Par exemple, si l'image d'origine est en 973 x 603, alors :

- le ratio largeur / hauteur original est : R = 973/603
- la largeur finale sera : **800** (pixels) cette valeur est imposée
- la hauteur finale sera : 603/973 x 800 = 800 / R = 495 (pixels)

Intégrez ce traitement dans `addfilmsave_action()` :
Remarques:

- la première partie du travail consiste à obtenir la dimension de l'image téléchargée; l'écriture list($width, $height) = getimagesize($image_source) permet de récupérer les dimension de l'image téléchargée
- voici (une grande partie) du code qui permet de redimensionner l'image; les variables $new_width et $new_height doivent, au préalable, avoir été renseignées :

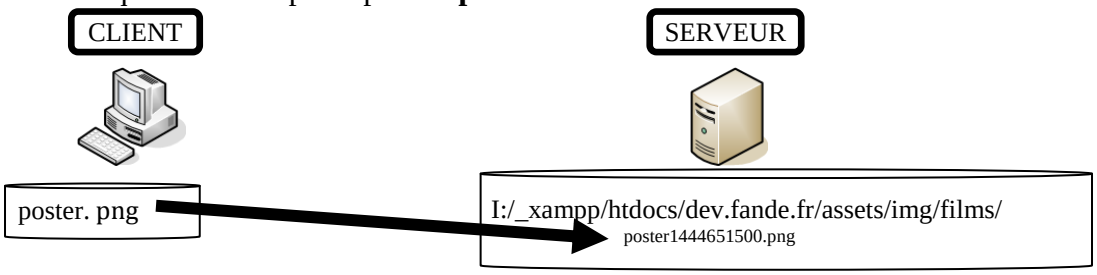
```
// lire l'image d'origine
$img = imagecreatefrompng( $image_source );
// à compléter ...
$img2 = ImageCreateTrueColor( $new_width, $new_height );
imagecopyResampled( $img2, $img, 0, 0, 0, 0, $new_width, $new_height, $width, $height );
imagejpeg( $img2, $image_source );
// effacer les zones mémoire
imagedestroy($img);
imagedestroy($img2);
```

Intégrez ce code dans la fonction `addfilmsave_action()` en complétant le texte des zones de commentaire.

Activité 5 – évolution 3 - [20 min]

/apps/controllers.php

Le fichier doit être renommé si un autre fichier porte déjà le même nom. Le principe simple que l'on va appliquer consiste à ajouter, à la fin du nom naturel du fichier, un marqueur spécifique et unique. Souvent le TIMESTAMP est utilisé.
QUESTION : recherchez la fonction du PHP qui permet d'obtenir cette information.
Illustration: supposons que le fichier que vous devez transférer se nomme poster.png;
alors le fichier transféré sera nommé poster1444651500.png
Voici un schéma qui illustre le principe **complet** :



QUESTION : recherchez la fonction du PHP qui permet d'obtenir uniquement le nom d'un fichier à partir de son nom complet (à partir de poster.png on obtiendrai poster).
Intégrez ce code de renommage dans la fonction `addfilmsave_action()`.

Activité 6 – évolution 4 - [5 min]

fande / lib / user.inc.php

Vous devez regrouper l'ensemble du traitement développé dans la page `upload.php` à l'intérieur d'une fonction nommée `uploadImage($file)` que vous placerez dans la librairie `apps/lib/functions.php`.

Voici la structure de la fonction:

```
/*
 * Enregistre une image téléchargée dans le dossier assets/img/films.
 * Le format de l'image .png est vérifié.
 * l'image est renommée si une image portant le même nom est déjà existante.
 * L'image est redimensionnée à 800 pixels de large (la hauteur est proportionnelle).
 *
 * @param {string}    $file
 *                    le nom de l'élément file dans le formulaire.
 * @return array
 *         un tableau avec 2 éléments :
 *             - le statut de l'opération: true si une erreur, false sinon.
 *             - le nom final de l'image telle qu'elle est enregistrée sur le disque.
 *
 * @code
 * list( $error, $image ) = uploadImage( $file['poster'] );
 * @endcode
 */
function uploadImage( $file )
{
        … à compléter
}
```

CONSEILS:

La fonction retourne une structure de type tableau pour informer de l'état de son traitement.

Voici ce qu'il faut écrire à la fin de fonction `uploadImage`:

```
return array( $error, $filename );
```

ici `$error` est une variable qui contient `true` ou `false` suivant les résultat des opérations, et `$filename` représente le nom final du fichier placé sur le disque.

Activité 7 [10 min.]

fande / lib / user.inc.php

Lorsque vous allez effacer un film, son image associée doit aussi être supprimée du dossier `/img/films`.

Donnez le code de la fonction `deleteImage($image)` (qui sera également placée dans `/apps/lib/functions.php`) qui permet de supprimer une image. Voici la structure de la fonction:

```
/*
 * Effacer une image dans le dossier /assets/img/films.
 * @param {string}    $image
 *                    le nom du fichier image à effacer.
 * @code
 * deleteImage('poster.png'); // effacer l'image /img/films/poster.png
 * @endcode
 */
function deleteImage( $image )
{
        … à compléter ici
}
```

ATTENTION: si la fonction est utilisée avec un nom d'image qui n'existe pas physiquement dans le dossier, aucune erreur ne doit être déclenchée ou affichée.