Atelier : Mise en place d'un système d'upload sécurisé

Facile, 2h

Contexte

Tu es un développeur web qui a été engagé par une entreprise pour ajouter une fonctionnalité d'upload de fichiers sur un site web.

Le but est de permettre à des candidats d'uploader leurs CV sur la plateforme afin que les recruteurs puissent les télécharger. Cependant, il faut que les candidats puissent uploader uniquement des fichiers PDF et que ces fichiers soient stockés dans un dossier sécurisé.

Les noms des fichiers uploadés doivent être de la forme suivante : `cv_<nom>_<prenom>_<uniqid>.pdf`.

Architecture du projet

La première étape est de mettre en place une architecture de projet.

Afin de sécuriser les fichiers et d'empêcher leur accès direct, tu vas créer un dossier `uploads` à la racine du projet dans lequel les fichiers seront stockés.

Tu vas également créer un dossier `public`, qui contiendra les fichiers accessibles depuis le navigateur et un dossier `src` qui contiendra les fichiers PHP utilisés par l'application.

Ces fichiers seront les suivants :

- `pubic/index.php` : la page d'accueil du site web permettant aux candidats d'uploader leur CV.
- `src/Upload.php` : la class PHP qui permettra d'uploader les fichiers.

Voici l'arborescence de ton projet :

```
.
├── uploads
└── public
    └── index.php
└── src
    └── Upload.php
```

Fichier `index.php`

Une fois les dossiers `uploads` et `public` créés, place le code suivant dans le fichier `index.html` :

```php
<?php
// Activation des sessions pour stocker les messages d'erreur
session_start();
// Upload du fichier
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    require_once __DIR__ . '/../src/Upload.php';


    $upload = new Upload();
    if ($upload->upload()) {
        $message = 'Votre CV a bien été uploadé.';
    } else {
        $message = $_SESSION['message'];
    }
}


// Suppression du message d'erreur de la session pour qu'il ne s'affiche qu'une fois
unset($_SESSION['message']);
?>


<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Upload de CV</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/water.css@2/out/light.css">
</head>


<body>
    <h1>Upload de CV</h1>
    <hr/>
    <h3>Candidat</h3>
```

```
    <p>Merci d'uploader votre CV par ce formulaire</p>
    <form action="" method="post" enctype="multipart/form-data">
        <label for="lastname">Votre nom</label>
        <input type="text" name="lastname" id="lastname" required>
        <label for="firstname">Votre prénom</label>
        <input type="text" name="firstname" id="firstname"
required>
        <label for="cv">Votre CV</label>
        <input type="file" name="cv" id="cv"
accept="application/pdf" required>
        <input type="submit" value="Envoyer">
    </form>


    <?php if ($message): ?>
        <div>
            <?= htmlspecialchars($message, ENT_QUOTES); ?>
        </div>
    <?php endif; ?>


</body>
</html>
```

Ce fichier est assez simple. Il contient un formulaire qui permet aux candidats d'uploader leur CV.

Il contient également de quoi gérer l'affiche de message d'erreur ou de succès et l'upload du fichier.

Si la méthode HTTP utilisée est `POST`, on instancie la classe `Upload` et on appelle la méthode `upload` qui permet d'uploader le fichier.

Fichier `Upload.php`

Maintenant, tu vas créer la classe PHP qui permettra d'uploader les fichiers.

Commence par créer le fichier `Upload.php` dans le dossier `src`.

Dans un premier temps, tu vas créer la classe `Upload` et la méthode `upload` qui permettra d'uploader le fichier.

Dans cette classe, tu vas également définir les propriétés suivantes :

- `$uploadDirectory` : le dossier dans lequel les fichiers seront stockés.

- `$allowedExtensions` : les extensions autorisées.
- `$allowedMimeTypes` : les types MIME autorisés.

```php
<?php


class Upload {


    private string $uploadDirectory = __DIR__ . '/../uploads/';
    private array $allowedExtensions = ['pdf'];
    private array $allowedMimeTypes = ['application/pdf'];


    public function upload() : bool {
    // TODO
    }
}
```

Notre méthode `upload` retourne un type `bool` pour indiquer si l'upload s'est bien passé ou non.

Ensuite, tu vas effectuer quelques vérifications pour t'assurer que le formulaire a bien été rempli.

Vérification que les champs attendus sont présents

Dans la méthode `upload`, tu vas vérifier que les champs attendus sont bien présents dans la requête HTTP.

```php
// Check if all fields are filled
if (empty($_FILES['cv']) || empty($_POST['lastname']) ||
empty($_POST['firstname'])) {
    $_SESSION['message'] = 'Merci de remplir tous les champs';
    return false;
}
```

Si ce n'est pas le cas, on crée un message d'erreur et on retourne `false` pour indiquer que l'upload a échoué.

L'utilisation de `return false;` ici nous permet de ne pas exécuter le reste du code de la méthode `upload` si les champs ne sont pas remplis. Cela se nomme un `Early return`. Cela nous permet d'éviter d'avoir un code trop imbriqué.

Taille maximale du fichier

Pour vérifier la taille maximale du fichier, tu peux utiliser la fonction `filesize` de PHP.

Lorsque tu upload un fichier en PHP, celui-ci est stocké dans un dossier temporaire. Tu peux récupérer le chemin de ce fichier temporaire avec la variable `$_FILES['cv']['tmp_name']`.

Vu que le fichier est présent sur le serveur, tu peux utiliser la fonction `filesize` pour récupérer sa taille en octets.

Ajoute ce code à la suite dans la méthode `upload` :

```
// Check file size
if (filesize($_FILES['cv']['tmp_name']) > 1000000) {
    $_SESSION['message'] = 'Le fichier est trop volumineux (1Mo max)';
    return false;
}
```

Vérification de l'extension du fichier

Maintenant, tu vas vérifier que le fichier uploadé est bien un fichier PDF.

Pour cela, tu peux utiliser la fonction `pathinfo` de PHP qui permet de récupérer des informations sur un chemin de fichier.

Toujours dans la méthode `upload`, ajoute le code suivant :

```
// Check file extension
$extension = pathinfo($_FILES['cv']['name'], PATHINFO_EXTENSION);
if (!in_array($extension, $this->allowedExtensions)) {
    $_SESSION['message'] = 'Le fichier doit être un PDF';
    return false;
}
```

Dans la classe, tu as une propriété `$allowedExtensions` qui contient les extensions autorisées. Ensuite, tu récupères l'extension du fichier uploadé avec la fonction `pathinfo` et tu vérifies qu'elle est bien présente dans le tableau des extensions autorisées.

Si ce n'est pas le cas, tu refuses le fichier.

Vérification du type MIME du fichier

Tu vas maintenant vérifier que le type MIME du fichier uploadé est bien un PDF. Pour cela, tu peux utiliser la fonction `mime_content_type` de PHP.

Toujours dans la méthode `upload`, ajoute le code suivant :

```
// Check file mime type
$mime = mime_content_type($_FILES['cv']['tmp_name']);
if (!in_array($mime, $this->allowedMimeTypes)) {
    $_SESSION['message'] = 'Le fichier doit être un PDF';
    return false;
}
```

Attention : cette vérification seule ne suffit pas, car le type MIME d'un fichier peut être manipulé par un attaquant. Il est donc important de vérifier également l'extension du fichier.

Nom du fichier

Tu vas maintenant générer un nom de fichier unique et sécurisé pour le fichier uploadé.

Dans un premier temps, tu vas devoir filtrer les champs nom et prénom pour éviter les caractères spéciaux, puis tu vas générer un nom de fichier unique à partir de la fonction `random_bytes`.

```
// Create secure filename (the extension is already checked)
$lastname = preg_replace('/[^a-z]/i', '', $_POST['lastname']);
$firstname = preg_replace('/[^a-z]/i', '', $_POST['firstname']);
$uniqId = bin2hex(random_bytes(8));
$filename = 'cv_' . $lastname . '_' . $firstname . '_' . $uniqId . '.' . $extension;
```

Tu as utilisé la fonction `preg_replace` pour remplacer tous les caractères qui ne sont pas des lettres par une chaîne vide.

Ensuite, tu as utilisé la fonction `bin2hex` pour convertir les octets générés par la fonction `random_bytes` en une chaîne de caractères hexadécimaux.

Tu te retrouves donc avec un nom de fichier qui ressemble à ça : `cv_nom_prenom_1234567890abcdef.pdf`.

Upload du fichier

Tu vas maintenant uploader le fichier dans le dossier `uploads`.

```
// Move file to uploads directory
```

```php
$destination = $this->uploadDirectory . $filename;
if (!move_uploaded_file($_FILES['cv']['tmp_name'], $destination))
{
    $_SESSION['message'] = 'Erreur lors de l\'upload';
    return false;
}
```

Tu as utilisé la fonction `move_uploaded_file` pour déplacer le fichier temporaire vers le dossier `uploads` avec le nom de fichier que tu as généré précédemment.

Si l'upload échoue, tu crées un message d'erreur et tu retournes `false`.

Fin du fichier

Si le traitement du fichier s'est bien passé, tu peux simplement retourner `true`.

```php
return true;
```

Fichier complet

A envoyer

Tester l'upload

Tu peux maintenant tester l'upload de fichier. Pour cela tu dois lancer un terminal et te placer dans le dossier `public` de ton projet.

Ensuite, tu peux lancer un serveur PHP avec la commande suivante :

```
php -S 0.0.0.0:8000
```

Tu peux ensuite accéder à ton site web à l'adresse `http://localhost:8000` ou `http://<ton-ip>:8000` depuis la machine hôte si tu es sur une machine virtuelle.

Tu dois maintenant pouvoir uploader un fichier PDF et voir le message de succès s'afficher. Les fichiers uploadés sont stockés dans le dossier `uploads`.

Aller plus loin

S'il te reste du temps, tu peux essayer d'implémenter une page qui permet aux recruteurs de télécharger les CV des candidats.