PHILIPPEMILINK

Zeste de savoir

Uploader un fichier volumineux par le web

23 août 2020

Table des matières

Il y a quelques temps, on m'a parlé d'une problématique à laquelle je n'avais pas été encore confronté. Il s'agissait d'uploader des fichiers volumineux vers un serveur par une interface web (en l'occurrence, c'était des photos, et l'upload pouvait facilement peser 500 Mo). Voici les problèmes qui se posaient :

— la taille de maximale du fichier uploadé était limité par la configuration de PHP qui faisait tourner l'application web (les clés de configuration *upload_max_filesize* et *post_max_size*) ;
— on ne voit pas de progression de l'upload. Même si la fibre se développe (lentement), uploader 500 Mo n'est pas une mince affaire!
— et que se passe-t-il si un problème dans l'upload survient lorsque 499 Mo ont déjà bien été envoyés? Je préfère mieux ne pas le savoir… 🍊

On m'a ensuite parlé d'une technique qui consiste à découper (*split*) les fichiers en petits morceaux (*slice* ou *chunk*) en JavaScript (par le client, donc), envoyer ces petits morceaux, et ensuite les ré-assembler sur le serveur (en PHP, pour ma part).

Curieux d'en savoir plus, j'ai demandé à mon moteur de recherche favori. Je vous propose dans ce billet de vous montrer mes découvertes.

1. From scratch…

J'ai donc voulu commencer par construire un système qui fait exactement ce que j'ai décrit dans l'introduction : découpage, upload et ré-assemblage.

i

Le code qui suit est très fortement inspiré de cet article . Celui-ci aussi explique cette méthode.

Pour ce qui est du HTML, que du classique :

```
<form>
    <input type="file" name="file" id="file-input" /><br />
    <input type="submit" value="Upload" id="submit-button" />
</form>

```

1. *From scratch...*

```
<div id="upload-progress"></div>

<script type="text/javascript"
    src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script type="text/javascript" src="upload.js"></script>
```

Le JavaScript maintenant :

```
$(function() {
    var reader = {};
    var file = {};
    var slice_size = 1000 * 1024; // Taille de chaque segment

    function start_upload(event) {
        event.preventDefault();

        reader = new FileReader();
        file = document.querySelector('#file-input').files[0];

        upload_file(0);
    }

    $('#submit-button').on('click', start_upload);

    function upload_file(start) {
        var next_slice = start + slice_size + 1;
        var blob = file.slice(start, next_slice); // on ne voudra
            lire qu'un segment du fichier

        reader.onloadend = function (event) { // fonction à
            exécuter lorsque le segment a fini d'être lu
            if (event.target.readyState !== FileReader.DONE) {
                return;
            }

            $.ajax({
                url: "upload.php",
                type: 'POST',
                dataType: 'json',
                cache: false,
                data: {
                    file_data: event.target.result,
                    file: file.name
                },
                error: function(jqXHR, textStatus, errorThrown) {
                    console.log(jqXHR, textStatus, errorThrown);
                },
                success: function(data) {
```

```
                    var size_done = start + slice_size;
                    var percent_done = Math.floor((size_done /
                        file.size) * 100);

                    if (next_slice < file.size) {

                            $('#upload-progress').html('Uploading File - '
                            + percent_done + '%');

                        upload_file(next_slice); // s'il reste à
                            lire, on appelle récursivement la
                            fonction
                    } else {

                            $('#upload-progress').html('Upload Complete!');
                    }
                }
            });
        };

        reader.readAsDataURL(blob); // lecture du segment
    }
});
```

Et le fichier PHP *upload.php* qui réceptionne les segments :

```
<?php

function decode_chunk($data) {
    $data = explode(';base64,', $data);

    if (!is_array($data) || !isset($data[1])) {
        return false;
    }

    $data = base64_decode($data[1]);
    if (!$data) {
        return false;
    }

    return $data;
}

// $file_path: fichier cible: garde le même nom de fichier, dans le
    dossier uploads
$file_path = 'uploads/' . $_POST['file'];
$file_data = decode_chunk($_POST['file_data']);

```

```php
if (false === $file_data) {
    echo "error";
}

/* on ajoute le segment de données qu'on vient de recevoir
 * au fichier qu'on est en train de ré-assembler: */
file_put_contents($file_path, $file_data, FILE_APPEND);

// nécessaire pour que JavaScript considère que la requête s'est
    bien passée:
echo json_encode([]);
```

Et ça fonctionne! Les fichiers sont correctement uploadés et on voit bien la progression!

2. Mais en fait, des bibliothèques existent !

En me renseignant sur le sujet, je me suis finalement rendu compte que de nombreuses bibliothèques JavaScript existaient pour répondre à ce problème. En voici une liste (bien-sûr non exhaustive !) :

— jQuery file upload plugin
— Resumable.js
— Pupload
— FineUploader

Je ne les ai pas testées, mais visiblement, elles possèdent toutes les fonctionnalités sympas liées à l'upload de fichiers (drag&drop, visualisation de la progression, …).

Lorsque j'ai commencé à creuser un peu ce sujet, j'avais comme objectif de me développer tout un petit système d'upload (visualisation de la progression, reprise en cas d'erreur, mise en pause, drag & drop des fichiers à uploader, …) utilisant cette technique de découpe et ne reposant sur une aucune bibliothèque tierce. Le but était de comprendre comment fonctionnaient tous ces mécanismes.

Autant vous dire que je ne suis pas allé bien loin dans l'élaboration de ce système d'upload. Principalement pour deux raisons :

— je n'avais pas besoin (personnellement) d'un tel système, c'était par pure curiosité ;
— comme j'en ai parlé plus haut, il existe en réalité de nombreuses bibliothèques JavaScript qui implémentent déjà tout ce que je souhaitait faire. Et comme un développeur ne réinvente jamais la roue… 🍊

J'espère, par ce billet, avoir contribué à l'explication de cette technique, que je ne trouve pas excessivement documentée sur Internet. N'hésitez pas à partager les articles/bibliothèques/explications que vous auriez sur le sujet!