RFC 7578 : Returning Values from Forms: multipart/form-data

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 15 juillet 2015

Date de publication du RFC : Juillet 2015

https://www.bortzmeyer.org/7578.html

Voici la nouvelle définition du type de données Internet `multipart/form-data`, qui est notamment envoyé par les navigateurs Web une fois qu'on a rempli un formulaire. Elle remplace la définition du RFC 2388 [1].

Il y a plusieurs façons de représenter les données d'un formulaire rempli (cf. annexe B de notre RFC). (Et il faut aussi se rappeler qu'il n'y a pas que les formulaires en HTML : le format décrit dans ce RFC peut servir à d'autres types de formulaires.) Les deux plus fréquentes sont le format par défaut des formulaires Web, `application/x-www-form-urlencoded`, et le `multipart/form-data` qui fait l'objet de notre RFC. La sagesse générale des développeurs Web (citée par exemple dans la norme HTML 4 <http://www.w3.org/TR/html401/interact/forms.html#form-content-type>) est d'utiliser `application/x-www-form-urlencoded` **sauf** s'il faut transmettre des fichiers ou des données binaires, auquel cas on préfère `multipart/form-data`. En HTML, cela se choisit par l'attribut `enctype`. Voici un exemple de formulaire HTML pour envoyer un fichier :

```
<form action="http://www.example.com/register"
       enctype="multipart/form-data"
       method="post">
   <p>
   What is your name? <input type="text" name="submit-name"/>
   What files are you sending? <input type="file" name="thefile"/>
   <input type="submit" value="Send"/>
 </form>
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc2388.txt`

Si l'utilisateur « Larry » envoie ainsi le fichier `file1.txt`, le navigateur encodera les données du formulaire ainsi :

```
Content-Type: multipart/form-data; boundary=---------------------------147689181032659200515465868892
Content-Length: 359

-----------------------------147689181032659200515465868892
Content-Disposition: form-data; name="submit-name"

Larry
-----------------------------147689181032659200515465868892
Content-Disposition: form-data; name="thefile"; filename="file1.txt"
Content-Type: text/plain

   ... contents of file1.txt ...

-----------------------------147689181032659200515465868892--
```

Alors que l'encodage pour du `application/x-www-form-urlencoded` aurait été `submit-name=Larry&the` (ce qui ne convient pas pour le fichier). Cette différence entre les formats des données des formulaires est bien expliquée dans cette réponse sur StackOverflow <http://stackoverflow.com/a/4526286/15625>. Et si vous voulez expérimenter vous-même avec netcat, voyez cette autre réponse <http://stackoverflow.com/a/28380690/15625>.

Vu de l'utilisateur, les choses se passent comme cela (section 1 de notre RFC) : on présente à l'utilisateur un formulaire L'utilisateur le remplit, il sélectionne (par exemple en cliquant) le contrôle qui sert à dire « j'ai fini, traite-moi ce formulaire ». Le contenu est envoyé à une application sur le serveur. (Petit souvenir personnel : les formulaires sont apparus avec la version 2 de Mosaic et, pendant un certain temps, il était difficile de compter dessus car tout le monde n'avait pas encore mis à jour. Mais ça semblait formidable d'avoir un formulaire en Motif développé avec juste trois lignes d'HTML.)

Quel est le format de ce contenu qu'on envoie ? Il faut se rappeler que les formulaires ne sont pas spécifiques au Web, à HTML et à HTTP : le format peut être utilisé dans toute application qui a une notion de formulaire. Notre RFC ne spécifie pas le formulaire, ni le traitement sur le serveur de réception, mais uniquement le format de transport du formulaire rempli.

Il travaille donc sur une vision abstraite d'un formulaire, composé d'une séquence de champs, chaque champ ayant une valeur, fournie par l'utilisateur (certains champs peuvent avoir une valeur par défaut). Les champs ont un nom, qui est une chaîne de caractères Unicode (mais il est conseillé de se limiter à ASCII si on veut maximiser les chances d'interopérabilité). Il est également conseillé que les noms des champs soient uniques.

La section 4 de notre RFC détaille le format complet. Un message `multipart/form-data` suit de près le format des autres `multipart/` MIME (RFC 2046, section 5.1). Comme son nom l'indique, il est composé de plusieurs parties séparées par une frontière, une ligne commençant par deux tirets. La valeur complète de cette ligne-frontière est donnée dans le champ `Content-Type:`. Il ne faut évidemment pas que cette ligne apparaisse dans les données envoyées.

Pour chaque partie, il est obligatoire d'avoir un champ `Content-Disposition:` (RFC 2183) avec un paramètre `name` qui indique le nom original du champ. Si la partie concerne un fichier qui a été envoyé, le RFC recommande que le champ `Content-Disposition:` contienne un paramètre `filename`, suggérant au serveur un nom pour le fichier. (« Suggérant » car, pour des raisons de sécurité, le serveur ne doit pas utiliser ce nom aveuglément, cf. aussi la section 7.)

Problème toujours intéressant, le jeu de caractères utilisé, et son encodage. On peut mettre un paramètre `charset` au `Content-Type:` de chaque partie MIME. Mais il est plus fréquent d'avoir un tel paramètre qui soit global. En HTML, la convention est qu'un champ caché du formulaire, nommé `_charset_`, fixe l'encodage par défaut. (Rappelez-vous que, dans les protocoles IETF et W3C, `charset` est mal nommé : c'est un encodage, plus qu'un jeu de caractères.) D'autre part, lorsque le transport du `multipart/form-data` est « *"8-bit clean"* », ce qui est le cas de HTTP, l'utilisation du `Content-Transfer-Encoding:` est déconseillée.

La section 5 rassemble un ensemble de conseils pratiques pour la bonne transmission et interprétation des données des formulaires. D'abord, il est fortement déconseillé (même si, légalement parlant, ce n'est pas systématiquement interdit) d'éviter les caractères autres que ceux d'ASCII pour les noms des champs dans un formulaire. Ce n'est pas grave en pratique puisque ces noms, contrairement aux étiquettes affichées dans le formulaire, ne sont pas visibles à l'utilisateur. Voici un exemple en HTML :

```
<p>Envoyez une copie de votre carte d'identité
<input type="file" name="idscan"/></p>
```

Le texte d'accompagnement comprend des caractères non-ASCII mais pas le paramètre `name`.

Toujours d'un point de vue pratique, quelques conseils de sécurité en section 7 :

— Les données envoyées peuvent être des données personnelles, voire sensibles. C'est d'autant plus vrai que certains navigateurs Web ont une fonction d'auto-remplissage qui met automatiquement des informations personnelles dans les formulaires, ce qui fait qu'un utilisateur distrait peut envoyer davantage d'informations qu'il ne le voulait. L'auteur du formulaire et de son traitement doit donc faire attention à ces données.
— Le contenu du fichier peut être dangereux (*"malware"*).

Le type `multipart/form-data` est, comme les autres types MIME, enregistré à l'IANA `<https://www.iana.org/assignments/media-types/media-types.xml#multipart>`.

Quels sont les changements depuis le RFC 2388 ? Le format a été spécifié à l'origine dans le RFC 1867 puis dans la norme HTML 3.2 `<http://www.w3.org/TR/REC-html32-19970114>`. L'annexe A de notre RFC détaille les changements, parmi lesquels :

— Les noms des champs non-ASCII doivent désormais être en UTF-8 et non plus encodés selon la méthode du RFC 2047 (mais, de toute façon, il est recommandé de les éviter). Cf. la section 5.1
— Lorsque plusieurs fichiers sont envoyés (pour un même champ du formulaire), l'ancienne recommandation (RFC 2388) était d'utiliser un `multipart/mixed`, la nouvelle est de mettre chaque fichier dans une partie MIME spécifique (en utilisant le même nom de champ).
— La convention du champ `_charset_` (encodage par défaut) est désormais documentée.