Master 2 RADI 2004 / 2005

Applications documentaires

Pratique des « encodings » de documents html

Julien Van Den Bossche
contact@julienvdb.com
http://www.julienvdb.com/universite/master/applidoc

Sommaire

1. Introduction

Le but de ce devoir est de transcoder des documents, dont l'encoding n'est pas connu au départ, vers l'encoding UTF8. Je vais vous expliquer comment j'ai procéder pour réaliser cette tache et je vous montrerai les problèmes qui se posent par rapport a cet encoding dans les différents navigateurs Internet.
Je finirai par un bilan sur le travail effectué et sur l'encoding UTF8.

2. Pré requis

La notion de jeu de caractères (charset en anglais) est une notion primordiale pour tout développeur web, et même, de façon plus générale, pour tout programmeur soucieux de préserver une certaine interopérabilité. Pourtant, cette notion n'est connue que par l'intermédiaire de la ligne <meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />, et certains s'imaginent que la simple modification de cette ligne permet de changer de jeu de caractère. Il n'en est rien. Je vais donc tenter d'éclaircir ici certains points.

Vous savez tous qu'un ordinateur est une machine fonctionnant grâce à des calculs binaires (1 ou 0), pour la simple mais bonne raison qu'il est bien plus facile de représenter un nombre en base binaire qu'en base 10 (qui nécessiterait de distinguer dix états physiques). Par voie de conséquence, les fichiers sont eux aussi stockés sous la forme d'une suite de bits, chaque bit ayant la valeur 0 ou 1. Ce sont ces bits que l'on voit lorsqu'on édite un fichier à l'aide d'un éditeur hexadécimal, car chaque valeur hexadécimale (base 16) correspond à 4 bits ($16 = 2^4$). Dans ce cadre, le jeu de caractère représente la façon de représenter chaque caractère (et donc un texte) dans cette base binaire. La plupart des jeux de caractères étant codés sur un octet, soit huit bits, chaque caractère du fichier texte correspondrait à une paire de chiffres dans un éditeur hexadécimal. On comprend alors que cette notion est tout ce qu'il y a de plus concret. En effet, tout comme ouvrir un fichier généré par un programme dans un autre programme peut poser quelques difficultés pour des raisons d'encodage de l'information, tenter de visionner un fichier texte (ou une page HTML, qui n'est jamais qu'un fichier texte amélioré) utilisant un certain jeu de caractère à l'aide d'un éditeur ou d'un navigateur gérant un autre jeu de caractère peut amener des résultats étranges, comme une substitution d'un caractère par un autre, ou peut-être un message d'erreur, si aucun caractère du jeu utilisé par le programme ne correspond à la paire hexadécimale renseignée dans le fichier.

L'UTF8 utilise un nombre d'octets variable pour représenter les caractères.
On utilise ainsi entre 1 et 6 octets pour représenter un caractère, en introduisant une convention pour différencier les octets isolés, représentant à eux seuls un caractère, ou les octets groupés, devant être considérés à plusieurs pour pouvoir en extraire l'information. Pour conserver la compatibilité avec l'ASCII et les jeux étendus, il a été décidé que les octets isolés commenceraient par un 0. Cette norme permet de représenter un nombre impressionnant de caractères différents, sans compter le fait qu'elle permet un autre système de représentation, l'agrégation de glyphes. Un peu plus de 90 000 caractères ont été attribués pour l'instant.

3. Langage utilisé

J'ai décidé d'utiliser le langage PHP car il comporte tous les outils nécessaires pour travailler correctement avec les expressions régulières. La manipulation des fichiers se fait assez naturellement.
Des fonctions pour l'encoding sont mises à notre disposition.
De plus ce langage est gratuit et se retrouve sur n'importe quelle plateforme.

4. Transformation des fichiers

La première étape a été d'ouvrir un fichier et d'en récupérer son encoding dans la balise meta. Il a donc fallu la détecter dans l'ensemble du code puis en extraire la valeur de l'encoding. Pour ce faire j'ai utilisé des expressions régulières qui sont en fait un masque qui permet de reconnaître une expression.

Exemple :

"(<meta http-equiv=\"Content-Type\" content=\")([^\">]) "*

Une fois trouver l'expression j'ai extrait la valeur du charset avec une autre expression régulière.

Exemple :

"(.)(charset=)(.*)"*

Ensuite il a fallu changer la valeur du charset dans le nouveau fichier que je vais créer à partir de l'original.
Pour ce faire j'utilise encore le même style d'expression régulière et j'utilise la fonction *eregi_replace* qui va permettre de remplacer en expression dans un texte par une autre en fonction du masque défini.
Je change en fait toute la balise *meta* et j'ajoute la nouvelle plus l'ancienne ainsi que l'url de la page en commentaire.
L'étape suivante a été d'encoder le texte en UTF8. Pour ce faire j'ai utilisé la fonction *utf8_encode* que propose PHP.

Pour dé-baliser les pages nous avons décidé de ne pas enlever les balises qui mettaient en forme le texte au niveau disposition. Nous avons donc garder les balises de structures du document HTML : <HTML><HEAD><BODY>…
Nous avons garder les balises de saut de ligne (
) les balises des divisions (<DIV>) ainsi que les balises des tableaux (<TABLE><TR><TH><TD>).
Tout le reste a été enlevé : <B><SPAN…><FONT…>…
Nous avons enlevé tous les scripts.

Pour dé-baliser le source je n'ai pas eu besoin de redéfinir des expressions régulières car PHP possède une fonction pour supprimer les balises que l'on souhaite ou plutôt supprimer toutes les balises et garder celles que l'on souhaite.

strip_tags($texte_encode_utf8,'<html><head><body><table><tr><th><td><h1> ');

Pour retirer les scripts j'ai construit une expression régulière :

"<script[^>]>(.|\n)*</script>(\r\n)?"*

Pour le reste du travail il a fallu écrire mes lignes de texte dans des fichiers de sortie.

5. Comportement des navigateurs

- ✓ Sur les documents en français, encodés par PHP en UTF8, seul Firefox parvient à afficher les caractères correctement. Netscape et Internet Explorer n'arrivent pas à afficher correctement les accents.

- ✓ Sur le finois on retrouve quelques problèmes avec Internet explorer et Netscape. Firefox arrive à afficher la page correctement.

  Exemple sur le même texte :
  IE et Netscape : *Uutiset tÃ¤nÃ¤Ã¤n*
  Firefox : *Uutiset tänään*

- ✓ Sur le Grec aucun des trois navigateurs ne réussi à afficher les pages comme il devrait. Dans ce cas non avons effectué un encodage de iso-8859-7 vers UTF8

- ✓ Sur le russe on retrouve le même problème et l'encodage en UTF8 par PHP n'est pas satisfaisant pour les trois navigateurs. Dans ce cas non avons effectué un encodage de windows-1251 vers UTF8

- ✓ L'arabe n'est lui non plus pas bien encodé donc les navigateur ne le lise pas correctement.
  Dans ce cas non avons effectué un encodage de windows-1256 vers UTF8

- ✓ L'hébreux n'est lui non plus pas bien encodé donc les navigateur ne le lise pas correctement.
  Dans ce cas non avons effectué un encodage de windows-1255 vers UTF8

- ✓ Le chinois n'est pas bien géré par les trois navigateurs une fois encodé en UTF8.

- ✓ Le japonais n'est pas bien géré une fois encodé.

- ✓ Le coréen n'est pas correctement reconnu une fois encodé en UTF8 par les trois navigateurs.

- ✓ L'hindi est bien géré par Internet explorer dans son encoding original et dans son encodage en UTF8. En revanche il n'est pas bien lu par Netscape et Firefox tant dans son encoding original qu'encodé en UTF8.

Tableau récapitulatif

Langue	Encoding original	Encodé en	Internet Explorer	FireFox	Netscape
Français	iso-8859-1	UTF8	Moyen	Bon	Moyen
Finnois	iso-8859-1	UTF8	Moyen	Bon	Moyen
Grec	iso-8859-7	UTF8	Mauvais	Mauvais	Mauvais
Russe	koi8-r	UTF8	Mauvais	Mauvais	Mauvais
Russe	windows-1251	UTF8	Mauvais	Mauvais	Mauvais
Arabe	windows-1256	UTF8	Mauvais	Mauvais	Mauvais
Hébreu	windows-1255	UTF8	Mauvais	Mauvais	Mauvais
Chinois	gb2312	UTF8	Mauvais	Mauvais	Mauvais
Chinois	Big5	UTF8	Mauvais	Mauvais	Mauvais
Japonais	EUC-JP	UTF8	Mauvais	Mauvais	Mauvais
Japonais	Shift_JIS	UTF8	Mauvais	Mauvais	Mauvais
Coréen	euc-kr	UTF8	Mauvais	Mauvais	Mauvais
Hindi	x-user-defined	UTF8	Bon	Mauvais	Mauvais

On se retrouve avec des navigateurs qui ne visionnent pas très bien les documents transcodés en UTF8 par PHP. Un petit avantage tout de même au client FireFox qui semble être le mieux.

Quand on visionne les sources la différence est la même que dans les navigateurs.
Les chaînes de caractères ne sont pas bien encodées à priori.
Donc on peut penser que c'est PHP qui ne possède pas de fonction correcte pour encoder vers l'UTF8.

6. Difficultés rencontrées

Après avoir effectué ce travail, je suis un peu déçu par PHP car il n'encode pas correctement en UTF8. On se retrouve comme on a pu le voir avec des textes qui ont été très mal encodés.

Concernant les éditeurs de texte, j'avais l'habitude de travailler avec Emacs mais ce dernier ne gère pas l'utf8. J'ai donc utilisé le bloc note sous Windows qui permet de gérer l'utf8.

Certains navigateurs ne veulent pas se mettre en UTF8 alors que c'est stipulé dans le charset.

La récupération de l'encoding dans l'entête http est difficile et ne fonctionne pas très bien. Les informations sont soit jamais envoyés, soit erronés.

Concernant le débalisage, on se retrouve avec des pages presque vierge car le source HTML est mal fait : il y a des balises ouvertes mais pas fermées et il y a non respect du DOM de la W3C.

7. Utilisation du transcoder

Cet outil est disponible sur http://wwww.julienvdb.com/universite/master/applidoc.

Vous avez le choix entre exécuter une liste d'url provenant d'un fichier texte ou bien vous pouvez choisir de rentrer une url. Le fichier à charger doit contenir une url par ligne.

Une fois le travail effectué, vous avez accès au fichier de synthèse.

8. Conclusion

Ce devoir a su me montrer que le problème d'encoding est assez complexe. Pourtant le standard UTF8 semblerait le mieux adapté mais on voit bien qu'il n'est pas pris en charge par tous les logiciels (l'éditeur Emacs par exemple).
Les navigateurs Internet ne s'occupent pas forcément de la balise *meta* ce qui engendre des problèmes (exemple : mon fichier de synthèse est encodé en utf8 mais IE choisit l'encoding ISO pour le lire donc on retrouve des problèmes sur l'affichage).
PHP ne comporte peut être pas les meilleures librairies pour traiter les problèmes d'encoding.