Collecte automatique ou programmée de données et de renseignements sur les sites Web

À force de développer les interfaces graphiques, l'interaction et l'ergonomie, on en vient à oublier, pire, à concevoir plus difficilement, la création d'automates qui feraient le travail à notre place de façon régulière et autonome. Il est ahurissant de constater quotidiennement le temps perdu en travaux répétitifs sur un ordinateur censé pourtant favoriser un accroissement de productivité.

Ne nous y trompons pas, il ne s'agit nullement ici de programmer des "aspirateurs" de sites tels que *Wget* ou *Websnake* : ce serait quelque peu réducteur. Ceux-ci se voient initialement développés afin de rapatrier des hiérarchies entières de documents statiques diffusés par ces sites, principalement en vue d'une consultation locale pour des raisons d'économie (limitation des durées de communication) ou pour la gestion de sites miroirs. Si tel est le cas, nous vous invitons à vous pencher sur l'utilitaire *Wget* (fourni en standard sur les CD de la Red Hat 5.X), qui remplit parfaitement cette fonction. À titre d'anecdote, voici ci-dessous la commande permettant de télécharger les images recto des pochettes de CD proposés par la Fnac sur son site (réfléchissez à deux fois avant de lancer cette commande, car il y a environ 37 000 fichiers, représentant un peu plus de 600 Mo) :

```
wget -b -o pochettes_CD.log -t 10 -T
600 -r -L -np http://www.fnacdi-
rect.fr/images/disques/recto
```

Définition du problème

Non, le problème se révèle un tout petit peu plus complexe, puisque l'idée se résume essentiellement à simuler l'envoi d'un formulaire, présenté normalement au format Html à un utilisateur, comme s'il avait été rempli et renvoyé manuellement par celui-ci. Un peu plus complexe, mais pas la mer à boire tout de même.

Les différentes étapes sont, tout d'abord, l'accès au formulaire à l'aide d'un navigateur quelconque, puis la sauvegarde du source de la page Html, suivie de l'analyse de son contenu et plus particulièrement de la zone délimitée par les balises <FORM></FORM> (attention, plusieurs couples de ces balises peuvent coexister au sein de la même page, ce qui ne présente aucune erreur ni incohérence d'un point de vue logique ou syntaxique) et enfin, la détermination des différentes variables présentes, de leurs valeurs et caractéristiques possibles et même de leur ordre d'apparition dans le document.

L'ordre d'apparition des couples "variable=valeur" dans la requête à rédiger ne devrait pas se voir pris en compte par les scripts CGI, mais il n'en demeure pas moins que bon nombre de prétendus spécialistes du Web ne respectent pas ce principe.

En guise d'application typique, vous pouvez regarder le source de notre script de conversion de devises, qu'il suffit d'invoquer en prenant en compte pour seul paramètre le montant de dollars à transposer en francs.

Œil de Lynx

En fait, il s'agit juste d'un appel à *Lynx*, navigateur Web en mode texte (employé sur les terminaux privés de capacités graphiques et aussi sur les serveurs Videotex qui assurent un relais vers le Web pour de simples Minitels en mode quatre-vingts colonnes), de façon non interactive (implicite avec le paramètre "-post_data") que l'on a "encapsulé" dans un script Perl et dont la sortie est redirigée sur un descripteur de fichier ("LYNX") afin d'en analyser le contenu pour y trouver une ligne contenant "United States Dollars = X,XXX.XX FRF"

Le paramètre facultatif "-nolist" évite que *Lynx* conclut sur un récapitulatif de tous les liens trouvés dans le document reçu.

Pour voir le formulaire Html correspondant, il vous suffit de consulter la page http://www.xe.net/currency.

Vous aurez le loisir de voir dans le source que la méthode d'envoi des données est de type "POST" Si elle avait été de type "GET", le paramètre communiqué à *Lynx* aurait tout simplement été "-get_data" plutôt que "-post_data" et n'aurait impliqué aucune autre modification. Les trois tirets sont propres à *Lynx* et lui indiquent la fin des données à soumettre au serveur. *Lynx* autorisant l'écriture des valeurs à envoyer sur plusieurs lignes, la présence de ces tirets lui indique leur limite.

Comme vous le constatez, *Lynx* s'avère excellent pour ce genre d'opération ; toutefois, s'il faut le relancer à chaque requête et que, dans la foulée, il convient de filtrer le résultat obtenu dans un script, autant gagner du temps et économiser des ressources tout en gagnant la possibilité d'améliorer nos requêtes sur des points particuliers que *Lynx* ne permet pas de personnaliser. Pour cela, il faut et il suffit d'employer une bibliothèque de fonctions spécialisées, tâche que remplit à merveille *LWP* pour Perl.

Attention, *LWP* repose sur un nombre important d'autres bibliothèques, ce qui rend son installation un rien rébarbative. Elles servent une grande diversité d'applications, touchant essentiellement aux services Internet (telnet, SMTP, POP, etc.).

Lynx reste pour la suite de nos développements un outil privilégié, en raison de sa richesse. En effet, parmi sa multitude d'options, une partie se trouve consacrée à l'analyse du flux de données qui transitent entre le client (le navigateur Web) et le serveur. Ce sont "-dump" et "-source", pour les plus simples, qui affichent le source Html du document sollicité, plutôt que son contenu mis en forme, "-trace" (éventuellement associée avec "-tlog") qui donne tout le détail des échanges et "-mime_header", option très intéressante qui présente tous les en-têtes renvoyés par le serveur et qui précèdent le document.

Automatisation

Revenons une seconde sur l'étape consistant à répertorier les variables contenues dans un formulaire, leurs valeurs possibles (pour les menus et les divers boutons à cocher, radio ou image) et éventuellement, les contraintes auxquelles elles sont soumises, au moins du côté du navigateur

(option "MAXLENGTH" ou vérifications/ modifications en JavaScript). Comme il risque de s'avérer fastidieux, et même hasardeux, de tout relever manuellement pour ensuite remettre l'ensemble en forme dans le cadre d'une requête *Lynx*, par exemple pour ressembler à :

```
variable_1=valeur_1&variable_2=valeu
r_2&etc.
```

une astuce toute simple consiste à remplacer dans le document Html sauvé sur votre disque dur (en local donc), l'URL associée à la variable "ACTION" dans la balise "FORM" par quelque chose du genre :

```
http://localhost/cgi-bin/test-cgi
```

Ainsi, en l'ouvrant dans votre navigateur et en remplissant les différents champs puis en le soumettant enfin, le script test-cgi vous retournera une valeur pour QUERY_STRING (si la méthode est de type "GET") correctement formulée, que vous pourrez utiliser telle quelle dans vos programmes.

Trouver les coordonnées d'une entreprise introuvable

Traitons un cas concret : la recherche des coordonnées d'une société localisée quelque part en France.

Facile, nous direz-vous ! Pour cela, il suffit d'aller sur www.pageszoom.com, www.annuaire.laposte.fr ou bien encore sur www.annu.com. Hum ! Hum ! Exact, sauf que... dans chacun de ces annuaires et mis à part le premier d'entre eux, vraiment en dessous de tout, car il fait l'impasse sur des demandes simples (règlement de compte personnel) vous êtes obligé d'entrer au moins une référence géographique, ville ou département. Quel plaisir de saisir tour à tour 37000 noms de communes ou alors, dans le meilleur des cas, près d'une centaine de numéros de départements si l'on n'a aucune idée de la situation géographique de l'entité recherchée !

En revanche, il suffit d'un "petit" script pour résoudre notre problème.

Notez que si ce script, pour des besoins didactiques, utilise *LWP*, il pourrait tout aussi bien, et probablement plus facilement, faire uniquement appel à *Lynx*.

A l'inverse, celui-ci se révélera probablement insuffisant, si vous voulez développer un utilitaire équivalent, pour chercher l'information sur les sites de la Poste ou de France Télécom.

A cet égard, nous vous invitons à plancher sur ce problème, car il représente un beau défi (nous sommes convaincus qu'il y a moyen d'en venir à bout).

Lire la documentation :

man LWP::UserAgent

man HTTP::Response

man HTTP::Request

pour en savoir plus sur ces trois bibliothèques de fonctions et toutes les options qu'elles offrent. HTTP::Request a notamment l'énorme avantage sur *Lynx* d'autoriser la définition d'en-têtes spécifiques, comme par exemple 'Referer', qui est parfois employé par les serveurs pour "tenter" de contrôler la provenance de la requête.

L'exploitation d'un tel méta-annuaire passera par un Intranet, qui fera office d'intermédiaire entre les utilisateurs et les différents annuaires cités plus haut, en présentant son propre formulaire ; celui-ci invoquera un script CGI dérivé de l'embryon proposé ici.

Intéressons-nous à la question pendant quelques instants

La première opération consiste à remplacer les espaces par le caractère "+" dans le nom passé en paramètre.

La création de l'objet "$requete" stipule que la méthode pour le passage des données au script CGI est de type "POST" (par opposition à "GET") et que l'URL du script s'écrit "http://www.annu.com/annu/cgi-bin/www".

Dans la boucle qui passera en revue les quatre-vingt-quinze départements, nous donnons nous-mêmes la valeur correcte pour le champ "Content-length" appartenant à l'en-tête de la requête envoyée au script CGI. Les différents champs obligatoires dans l'en-tête seront automatiquement remplis et insérés à votre place par la méthode "request", si vous ne les avez pas définis explicitement avant son appel.

La variable d'instance "content" (dans l'expression "$reponse->content") contient le résultat délivré par le script CGI via son serveur Web dans sa représentation "source". Cela explique la présence des deux substitutions dans la procédure "traite_reponse" : la première sert à faciliter la délimitation de chaque bloc de lignes constituant les coordonnées d'une personne ou société trouvée, et la substitution suivante élimine toutes les balises Html pour ne conserver que les informations brutes. Ensuite, comme chaque bloc commence par son numéro d'ordre, seul sur une ligne, et finit par une séquence de quatre tirets, eux aussi isolés sur une ligne, l'extraction des coordonnées se trouve de ce fait simplifiée. Il suffit d'y supprimer les lignes vides et de remplacer le "é" en "e" (cette dernière opération se révèle inutile sur les systèmes affichant correctement les caractères codés selon la norme ISO 8859-1). Notez que, au besoin, les librairies *LWP* et associées contiennent toutes les méthodes requises pour connaître le détail complet des échanges entre client et serveur.

Allons plus loin

Il existe quatre optimisations possibles. La première consiste à lancer les requêtes pour tous les départements en parallèle plutôt qu'en séquence. La seconde se résume ainsi : présenter des résultats à l'utilisateur au fur et à mesure de leur réception. En outre, on peut proposer une recherche Région Parisienne ou Province. Enfin, une amélioration d'ordre esthétique passera par la détection des coordonnées partageant la même adresse pour les regrouper en une seule entité avec la liste des différents services et/ou numéros de téléphone et/ou de télécopieur.

Attention, le nom à rechercher doit répondre à la syntaxe attendue par un script CGI quant à la représentation des caractères. Mis à part les lettres, les chiffres et quelques symboles de ponctuation, ceux-ci sont piochés dans la norme ISO 8859-1 (adoptée avec bonheur et discernement par Linux, et accessoirement, par pure chance et hasard, sur les dernières versions de Winchose) et codés en hexadécimal sous la forme %HH.

Conversion automatique de devises

```
#!/usr/bin/perl

$somme = $ARGV[0];
$devise_depart =
'USD+United+States+Dollars';
$devise_arrivee =
'FRF+France+Francs';

open (LYNX, "/usr/bin/lynx
post_data -nolist
http://www.xe.net/cgi-
bin/ucc/convert<<FIN_REQUETE\nti-
mezone=Canada%2FEastern&Amount=$so
mme&From=$devise_depart&To=$devi-
se_arrivee\n---\nFIN_REQUETE\n| );

while (<LYNX>)
{
  if (/United States
Dollars[^\d]+([\d, ]+) FRF/){
    ($change = $1) =~ tr/ ./  /;
    print "Cela fait $change
Francs Francais\n";
    last;
  }
}
close (LYNX);
```

À l'inverse, le seul caractère non ASCII présent dans les réponses délivrées par www.annu.com semble être le é de Télécopie et Numéris, que nous remplaçons par un e à l'aide de s/\351/e/g.
Soyons clairs : bien que très intéressant, un automate de ce genre qui semble de prime abord simple à programmer, révèle tout de même plusieurs cas de figure à prendre en considération. Parmi quelques exemples très communs, nous citerons le découpage de la réponse en plusieurs pages Html contenant un bouton suite – lequel oblige à soumettre une autre requête au serveur, plus ou moins différente de celle de base (voilà pourquoi le source présenté ici se cantonne à n'afficher au plus que les dix premières coordonnées trouvées dans chaque département) et l'absence de réponse, plus fréquente encore. Autrement dit, il se révèle indispensable de tester manuellement une majorité de cas et de sauver les pages Html correspondant aux résultats, afin de pouvoir les analyser en détail avant de se lancer dans l'écriture d'un client Web automatique.

SSL

Un cas particulier reste celui de l'accès à des sites sécurisés avec le protocole SSL. Normalement, nous pourrions nous abstenir d'en parler puisque, dans le cas de *Lynx* par exemple, les modifications du code développées pour utiliser ce protocole, ou même le code exécutable correspondant, ne devraient pas être accessibles depuis la France. D'autant que le site officiel de *Lynx* se montre assez intransigeant sur ce sujet et ne propose un envoi de ceux-ci que par la poste et uniquement sur le territoire américain. Malgré tout, il arrive qu'un quidam, moins à cheval sur les lois ou moins averti, dépose sur un site FTP public tout ou partie du code en question, disponible alors sous la désignation lynx-ssl-2.8-1.i386.rpm (mais bon, nous ne vous avons rien dit, hein ?).
Notez que vous ne serez en mesure d'installer cette archive qu'à condition d'avoir préalablement récupéré et installé la librairie SSL (archive SSLeay-0.8.1 4.glibc.i386.rpm par exemple, mais... chut !).
Il convient de préciser que l'on retrouve sur les serveurs très sollicités par le genre d'automates présentés ici des parades similaires à celles apparues avec l'essor du Minitel ; la plus simple consiste à changer la présentation des données communiquées à l'utilisateur, afin de rendre caduque, au moins temporairement, l'analyse contenue dans le script.
Si vous désirez approfondir ce sujet, sachez qu'il existe un ouvrage consacré à ce domaine : *Programmation De Clients Web Avec Perl*, par Clinton Wong, chez O'REILLY.

Yannick Cadin
Yannick@kommando.com

Recherche de coordonnées sur toute la France

```
#!/usr/local/bin/perl

use LWP::UserAgent;
use HTTP::Request;
use HTTP::Response;

($nom_cherche = $ARGV[0]) =~ s/ /+/g;
$formulaire =
"country=france&type=0&cv=0&num=0&nom=$nom_cherche&ville=&dep=&rub=";
$agent = new LWP::UserAgent;
$requete = new HTTP::Request ('POST', 'http://www.annu.com/annu/cgi-
bin/www');

for ($dept = 1; $dept <= 95; $dept++)
{
  $formulaire =~ s/dep=[^&]*&/dep=$dept&/;
  $requete->header ('Content-length' => length ($formulaire));
  $requete->content ($formulaire);
  $reponse = $agent->request ($requete);

  if ($reponse->is_success){
    $total_reponses += traite_reponse ($reponse->content);
  } else {
    print STDERR "Echec de la requete pour le departement $dept\n";
    $debug && print $reponse->error_as_HTML;
  }
}
$total_reponses && print "\n$total_reponses reponse(s) trouvee(s)\n";


sub traite_reponse
{
  my ($reponse) = @_;
  my ($trouves, $element, $coordonnees);

  if ($reponse =~ /Nombre de personnes trouv.es  : (\d+)/) {
    $trouves = ($1 > 10) ? 10 : $1;
    $reponse =~ s/<HR>/----/g;
    $reponse =~ s/<[^>]*>//g;

    for ($element = 1; $element <= $trouves; $element++){
      if ($reponse =~ /\n$element\n(.*?)\n----\n/s){
        $coordonnees = $1;
        $coordonnees =~ s/\n{2,}/\n/g;
        $coordonnees =~ s/\351/e/g;
        print "$coordonnees";
      }else{
        die "Probleme d'analyse de la reponse\n";
      }
    }
  }else{
    $trouves = 0;
  }
  return $trouves;
```