Université de Technologie de Compiègne

Génie Informatique

Rapport de TD

LO17

Steve LAGACHE & Romain PELLERIN

Chargé de TD : ?

Printemps 2016 (P16)

Dernière mise à jour : 14 mars 2016

Table des matières

1 TD1 : Structuration

L'objectif de ce TD était de s'initier au "*parsing*" en Perl. Nous avions un peu plus de 300 pages HTML contenant toute un article issu d'un même site scientifique. Il nous fallait extraire des données "uniques" comme le numéro de l'article ou sa rubrique, et des données présentes une (parfois zéro) ou plusieurs fois, comme les paragraphes du texte principal ou les images présentes.

1.1 Méthodologie employée

Pour la totalité des éléments à récupérer, nous avons utilisé des *regex* (sauf pour le nom du fichier que nous récupérerions dans `$ARGV`).

Pour produire notre XML, nous avons décidé de tout afficher sur la sortie standard. Il s'agira ensuite de rédiriger cette sortie standard dans un fichier grâce à l'opérateur `>` d'Unix.

1.1.1 Repérer un élémént d'intérêt

Pour chaque élément unique qu'il nous fallait récupérer, nous avons essayer de trouver la structure de balises HTML voisinante la plus proche possible qui soit unique. Pour cela, nous avons beaucoup fait appel à la structure `.*?` (n'importe quel caractère zéro, une ou plusieurs fois) dans son mode *lazy*. Par exemple, pour le numéro, la structure HTML autour était suffisant simple pour ne pas utiliser "n'importe quel caractère" :

```
$body=~/<span class="style95" style="color:inherit">(\d+)<\/span
   ><\/a>/;
```

Tandis que nous en avons eu besoin pour la partie contact :

```
$body =~ /Pour en savoir plus, contacts :.*?<p class="style44"><
   span class="style85">(.*?)<\/span>/s;
```

1.1.2 Lecture du contenu d'un fichier

Nous avons été capables de récuperer le contenu d'un fichier passé en argument grâce à une boucle sur l'opérateur "diamant". Fondalement, cet opérateur permet de boucler sur chaque ligne de "l'*input*". Nous avons donc simplement récupéré chaque ligne du fichier et avons concaténé ces lignes à une variable `$body` initialisée à une chaîne de caractères vide.

1.1.3 Lecture du contenu de plusieurs fichiers

Pour pouvoir récupérer toutes les lignes de chaque fichier tout en dissociant les fichiers, nous avons du opter pour une technique légèrement différente.

1. Nous créons une tableau.
2. Chaque "case" du tableau sera une variable similaire à `$body` dont nous avons parlé au-dessus : elle contiendra toutes les lignes **d'un seul fichier**.
3. Grâce à l'opérateur diamant, nous bouclons sur toutes les lignes de l'*input*.
4. Nous sommes capable de savoir à tout instant quel fichier nous sommes en train de lire grâce à la variable `$ARGV`.

5. Une fois que nous avons récupéré tous les différents documents dans chaque case du tableau, nous commençons à boucler sur ce tableau de la même manière que nous le faisions pour un seul fichier.

```
@htmls;
while (<>) {
  $fichier = $ARGV;
  $fichier=~s/.*\///g;
  if (!defined(@htmls{$fichier})) {
    $htmls{$fichier} = $_;
  }
  else {
    $htmls{$fichier} .= $_;
  }
}

print "<corpus>\n";
while (($fichier,$html) = each(%htmls)) {
  print "<bulletin>\n";
  ...
  print "</bulletin>\n";
  }
print "</corpus>\n";
```

1.2 Commandes Unix

1.2.1 Génération du XML

Voici la commande Unix utilisée pour récupérer la sortie standard de notre programme et la rediriger dans un fichier XML. Il faut également préciser que nous avons rendu notre programme exécutable grâce à `chmod`. De plus, nous utilisons le script `convert.pl` pour convertir les entités HTML en caractère Unicode.

```
./td1.pl BULLETINS/*.htm | perl convert.pl > output.xml
```

1.2.2 Vérification du nombre d'images

Nous avons également utilisé quelques commandes Unix supplémentaires dans le but de vérifier que nous récupérerions bien le nombre exactes d'images présentes dans les articles. Par exemple, nous avons utilisé `grep` dans son mode *regex*.

```
./td1.pl BULLETINS/*.htm | perl convert.pl > output.xml && { grep "
   <image>" output.xml } | wc -l && echo "Images parsées en Perl"
   && { grep -aoE "streaming.+?\.jpg" BULLETINS/*.htm && grep -aoE
   "<img.*?www\.bulletins-electroniques\.com\/Resources_fm\/
   actualites.*?\.jpg" BULLETINS/*.htm } | wc -l && echo "Images
   dans les fichiers d'origine";
```

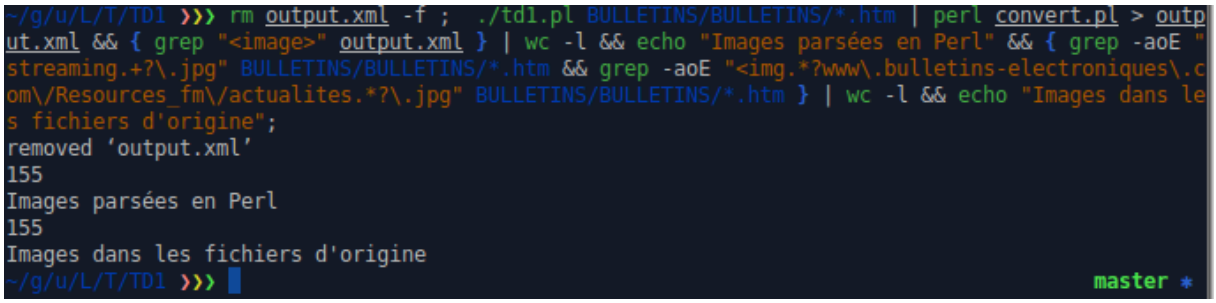

Figure 1.1 – Résultat de la commande

2 Conclusion