G. Hunault
Licence Professionnelle

Angers, septembre 2005

Développement *Web*

1. Perl

Le service de bioinformatique veut convertir des fichiers de codons en fichiers d'acides aminés. En supposant que les seules conversions possibles sont

codon	acide aminé
`"atg"`	`"M"`
`"att"`	`"I"`
`"gca"`	`"A"`
`"cca"`	`"P"`

écrire un script perl qui lit le nom d'un fichier de codons et produit le fichier des acides aminés correspondant. Par exemple à la suite de l'exécution de

```
perl conversion.pl fic0032.codon
```

on doit disposer d'un fichier `fic0032.aa` avec les codons convertis en acides aminés. Ainsi les lignes 1 et 2 du fichier `fic0032.codon` qui sont

```
ccagcaccagcaatgattatgatt
atggcaatggca
```

seraient converties en

```
PAPAMIMI
MAMA
```

2. Php et Mysql

La base de données Mysql nommée VEHICULES contient entre autres les champs VH (pour véhicule), DST (pour distance en kilomètres) et CNS (pour consommation absolue en litres d'essence). On voudrait savoir quels véhicules ont une consommation relative (rapport CNS/DST rapportée à 100 km) supérieure à la moyenne des consommations relatives. Donner les instructions Php qui effectuent cette recherche. On supposera que le serveur est LAGAFFE, que l'utilisateur est GASTON et que le mot de passe est SIESTE.

3. Xml

On dispose d'un fichier XML nommé `septembre2005.xml` dont le texte est fourni ci-dessous.

Ce document est-il bien formé ? est-il valide ? Expliquer pourquoi.

En admettant que ce document soit correct, écrire une transformation `XSL` qui compte le nombre d'ateliers et de personnes (pour notre exemple, une personne est soit un ouvrier soit un technicien).

```
<xml>
<USINE poste="1">
<Atelier numero=1>
     <ouvrier nom="EINSTEIN" prénom ="Albert" />
     <ouvrier nom="BOSE"     prénom ="Kurt" />
     <technicien nom="LAVIGNE"  prénom ="Mars" />
</Atelier><Atelier numero=3>
     <ouvrier nom="CURIE" prénom ="Joliot" />
     <ouvrier nom="CURIE" prénom ="Marie" />
     <technicien nom="RADIO" prénom ="Actif" />
     <technicien nom="CALCAIRE" prénom ="Végétal" />
</Atelier>
<Atelier numero=2>
     <ouvrier nom="FERMAT" prénom ="Pierre" />
</Atelier>
</USINE>
</xml>
```

Esquisse de SOLUTION

1. Perl

```
# CONVERSION DE  CODONS EN ACIDES AMINES (gH)

use File::Basename;

#  test des paramètres

if ($ARGV[0] eq "") {
     print " syntaxe : perl conversion.pl nom_de_fichier \n" ;
     exit(-1) ;
}  ; # fin de test sur les arguments

$fichier = $ARGV[0] ; # récupération du nom du fichier

# préparation du fichier de sortie

($nom,$chemin,$suffixe) = fileparse($fichier,"\.codon") ;

$sortie = "$nom.aa" ;

# voici une autre solution si on connait pas File::Basename :
# utiliser le texte avant le dernier point, soit :

$sortie = substr($fichier,0,rindex($fichier,".")).".aa" ;

open( FS ,">$sortie")
  || die "\n impossible d'écrire dans le fichier nommé $sortie \n\n" ;

# table de correspondance entre codon et acide aminé

%dcod=(
    "atg"=>"M",
    "att"=>"I",
    "gca"=>"A",
    "cca"=>"P"
) ; # fin de tableau de décodage
```

```
# ouverture du fichier d'entrée

open( FIC ,"<$fichier")
  || die "\n impossible d'ouvrir le fichier nommé $fichier \n\n" ;

# parcours du fichier avec conversion par paquets de 3

while (chomp($lig=<FIC>)) {
  $lng  = length($lig)/3 ;
  $nlig = "" ;
  for ($idc=0;$idc<$lng;$idc++) {
     $nlig .= $dcod{substr($lig,3*$idc,3)} ;
  } ; # fin de pour
  print FS "$nlig\n" ;
} ; # fin de tant que

# fermeture des fichiers et message de fin

close(FIC) ;
close(FS) ;
print " vous pouvez utiliser $sortie \n"
```

2. Php et Mysql

```
# CONNEXION AU SERVEUR
  mysql_connect("LAGAFFE","GASTON","SIESTE") ;
# OUVERTURE DE LA BASE
  mysql_select_db("VEHICULES") ;
```

On ne peut pas utiliser directement une instruction comme

```
select VH , CNS/DST from VEHICULES where CNS/DST > avg(CNS/DST) ;
```

car en Mysql on ne peut imbriquer les calculs. Or le calcul avec avg induit une requête imbriquée. Il faut donc procéder en deux temps et commencer par calculer la moyenne avant de faire un deuxieme "select" pour extraire les lignes concernées.

```
# calcul de la moyenne

$rq   = " select avg(CNS/DST) from VEHICULES " ;
$er   = mysql_query(" $rq ") ;
$lr   = mysql_fetch_array($er) ;
$res  = $lr["avg(CNS/DST)"] ;
$moy  = 100.0 * $res ;
print " moyenne des consommations relatives = $moy l/km\n" ;

# affichage des véhicules dont la consommation relative
# est supérieure à la moyenne des consommations relatives

$rq  = " select VH , CNS, DST from VEHICULES where 100.0*(CNS/DST) > $moy" ;
$rq  = " select VH , CNS, DST , 100.0*(CNS/DST) as Cr " ;
$rq .= "        from VEHICULES " ;
$rq .= "        where 100.0*(CNS/DST) > $moy order by Cr desc " ;

$er   = mysql_query(" $rq ") ;
print " Véhicule Consommation Distance Moyenne relative\n " ;

while ($ligr=mysql_fetch_array($er)) {

    # récupération des résultats

    $veh =  $ligr["VH"] ;
    $cns =  $ligr["CNS"] ;
    $dst =  $ligr["DST"] ;
    $mr  =  100.0*($cns/$dst) ;

    # formatage

    $f_veh =  sprintf("%-12s",$veh) ;
    $f_cns =  sprintf("%8d",$cns) ;
    $f_dst =  sprintf("%8d",$dst) ;
    $f_mr  =  sprintf("   %6.2f",$mr) ;

    # affichage

    print "$f_veh $f_cns $f_dst $f_mr \n" ;

 } ; # fin tant que
```

3. Xml

Le document XML n'est pas correct, car ce n'est pas la façon d'écrire `xml` (il faut un point d'interrogation). De plus, il n'y a pas d'*encoding* spécifié pour la langue utilisée alors qu'il y a une balise accentuée. Donc en ligne 1, au lieu de `<xml>` il faut mettre

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

Du coup, il faut enlever `</xml>` comme dernière ligne. Le document sera alors "presque" bien formé car en XML tous les attributs, même numériques doivent être mis entre guillemets. Il faut donc corriger

```
numero=1
```

en

```
numero="1"
```

etc.

Voici le vrai document bien formé :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

   <USINE poste="1">
   <Atelier numero="1">
        <ouvrier nom="EINSTEIN" prénom ="Albert" />
        <ouvrier nom="BOSE"     prénom ="Kurt" />
        <technicien nom="LAVIGNE"  prénom ="Mars" />
   </Atelier><Atelier numero="3">
        <ouvrier nom="CURIE" prénom ="Joliot" />
        <ouvrier nom="CURIE" prénom ="Marie" />
        <technicien nom="RADIO" prénom ="Actif" />
        <technicien nom="CALCAIRE" prénom ="Végétal" />
   </Atelier>
   <Atelier numero="2">
        <ouvrier nom="FERMAT" prénom ="Pierre" />
   </Atelier>
   </USINE>
```

Quant à savoir s'il est valide, cette question n'a aucun sens car aucune grammaire (DTD ou XSD) n'est proposée. Or, on valide forcément par rapport à une grammaire.

Pour compter, il suffit ici d'utiliser la fonction `count` de XSL soit le texte

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

   <xsl:output method="text" indent="no" encoding="ISO-8859-1" />

   <xsl:template match="/">

       Comptage des ateliers et des personnes :

       il y a <xsl:value-of select="count(//Atelier)" /> ateliers
       et     <xsl:value-of select="count(//ouvrier|//technicien)" /> personnes

       à savoir
          <xsl:value-of select="count(//ouvrier)" /> ouvriers
       et <xsl:value-of select="count(//technicien)" /> techniciens.

   </xsl:template>

</xsl:stylesheet>
```

qui produit pour notre exemple :

```
  Comptage des ateliers et des personnes :

   il y a 3 ateliers  et 8 personnes
   à savoir 5 ouvriers et 3 techniciens.
```

Si on veut avoir un "s" facultatif en fonction du nombre d'ateliers, on peut écrire

```
il y a <xsl:value-of select="count(//Atelier)" />
atelier<xsl:if test="(count(//Atelier)>1)">s</xsl:if>.
```

à condition de bien coller le test au mot atelier.