Techniques de Programmation Internet
Année Spéciale Informatique

ENSIMAG 2002-2003
James L. Crowley

Séance 6 Annulé

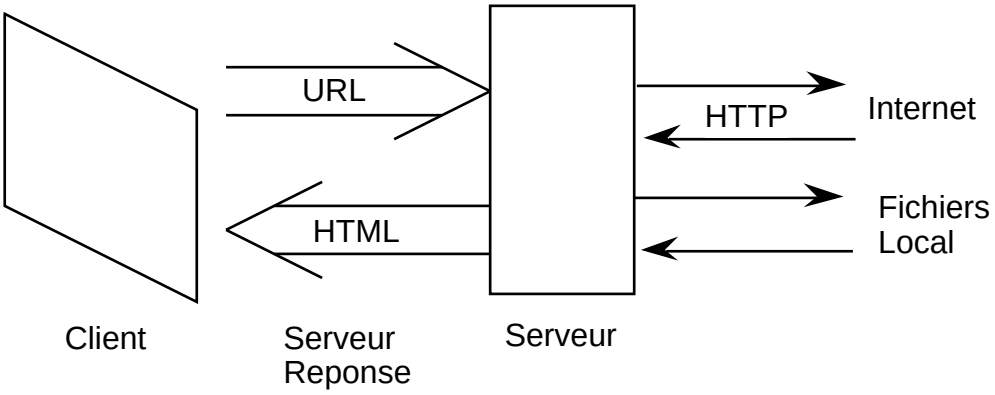
Codage/Decodage des paramètres avec PERL

Plan :

HTTP : Hyper Text Transfert Protocole

(rappele de la séance 1)

Modèle client-serveur pour le transfert des documents hypertextes.
Protocole utilisé par les serveurs WWW depuis 1990.
Échange de messages codés dans un format similaire au type MIME.



URL Syntaxe Definit par RFC 1738 et 1808.

Syntaxe :
method:// [User [: Password] @] machine [: port] / fichier[#ancre|?params]

methods :

file	accès local ou protocole FTP
ftp	protocole FTP
http	protocole HTTP
telnet	session interactive TELNET
gopher	protocole GOPHER
wais	version WAIS du protocole Z39.50
news	protocole NTTP
mailto	adresse de courrier électronique

URI Syntaxe

"%" (Percent) Caractere d'Escape
"/" (Slash) - Indicateur de Hiérarchie.
"#" (Hash) - Identificateur de Fragment (ex les anchors).
"?" (Query) - Delimiteur de interrogation.
"+" utilisé pour les espace.

```
Parametres :   ?name1=value1&name2=value2
```

Codage des paramètres

1. Les caractères non ASCII (code > 128) sont remplacés par %xx ou xx est le code ASCII.
2. Les caractères réservés sont remplacés par leur code ASCII. Ils sont : < > " # % ! $ \ & ( ) + = } [ ] \ : ; ~ ? , / [TAB]
3. L'espace est remplacé par le signe +
4. Les pairs noms/valeur sont transformés par la chaîne de caractères : nom=valeur
5. Les différentes chaînes de caractères sont contaminées en insérant le symbole & entre les paires : nom1=valeur1&nom2=valeur2&..

Il y a deux méthodes de passage des paramètres aux scripts :

GET les données sont transmises par avec la requête à l'URL.
POST les données sont envoyées et lues via STDIN.

Décodage des Paramètres

Dans une scripte, on peut construire une "hash" des paramètres avec le code suivant.

```
#Determine le type de passage
$method = $ENV{'REQUEST_METHOD'};
if ($method eq "GET")
{
    $query = $ENV{'QUERY_STRING'};
}
 elsif  ($method eq "POST")
{
    $size = $ENV{'CONTENT_LENGTH'};
    read (STDIN, $query, $size);
} else
{
    &return_error(500,
       "Server Error", "Unsupported Method");
}

#split le query en key=value.
@key_value_pairs = split(/&/, $query);
```

```
#decode chaque parametre
foreach $key_value (@key_value_pairs)
{
    ($key, $value) = split(/=/, $key_value);
    $value =~ tr/+/ /;
    $value =~
       s/%([\dA-Fa-f][\dA-Fa-f])/pack("C", hex($1))/eg;
    $data{$key} = $value;
    print "data($key) = $data{$key}", "\n";
}
```

Le hash %data contiennent les valeurs des paramètres indexés par les clés.

Analyse des Données en PERL

PERL contient trois opérateurs puissants pour l'analyse de données textuelles :

Match : Trouver l'existence d'un motif dans une chaîne
Substitute : Remplacer un motif par un autre dans une chaîne
Split : découper une chaîne en deux parties à chaque coté d'un motif.

Ces expressions sont rendu puissante, mais illisible, par l'existence des caractères spéciales : [,], (,), *, ., ^, -, |, \, ?

Match

Syntaxe :

```
$chaine =~ m/motif/;
```

Trouver l'existence de la chaîne "motif" dans la chaîne de caractères $chaîne

Exemples : s6.1.html - Détecter le Navigateur employé.

```
#!/usr/local/bin/perl
# script s6.1.cgi
print "Content-type: text/html\n\n";

#$browser = $ENV{'HTTP_USER_AGENT'};
$browser = "Mozilla";

if ($browser =~ m/Mozilla/)
{
   print "I see you are using Netscape.  Good Choice\n";
}  else
{
    print "I recommend that you switch to Netscape\n";
}
```

Match est sensible au minuscules-majuscule. Pour rendre la recherche insensible au majuscule, utilise l'option "i". Exemple :

```
    if ($query = ~m/help/i)
    { print "Voulez-vous de l\'aide?", "\n"  };
```

Substitute

Remplacer un motif par un autre dans une chaîne

Syntaxe :

```
$chaine =~ s/veiux/nouveau/g;
```

La commande "s" permettre de remplacer la première instance d'un motif par un autre. Avec l'option "g", cette commande remplace TOUS les instances.

Exemple. Le script y2k.cgi prepare pour l'année 2000 en remplacent chaque "y" par un " k".

<A HREF="s6.2.html">s6.2.html</A>

```
#!/usr/local/bin/perl
#file s6.2.cgi
print "Content-type: text/html", "\n\n";
print "<HTML>" , "\n";
print  "<TITLE>Preparation pour Y2K</TITLE>" , "\n";
print "<BODY>" , "\n";
$query = $ENV{'QUERY_STRING'};

($field, $value) = split(/=/, $query);
$value =~ tr/+/ /;

print qq(Votre donnee est : "$value".), "<br>\n";

$value =~ s/y/k/gi;

print qq(Pour l'année 2000 il faut : "$value".), "<br>\n";

print "<br>", "\n";
print "</BODY>" , "\n";
print "</HTML>" , "\n";
exit(0);
```

Inspection du résultat de Match et Substitute

PERL affecte trois variables avec le résultat de Match et Substitut :

- $& contient le motif qui etait matchaient.
- $` contient le chaîne avant le match.
- $' contient le reste de la chaîne.

Exemple : <A HREF="s6.3.html">s6.3.html</A>

```
$query = "string=Please+HELP+me!";

($field, $value) = split(/=/, $query);
$value =~ tr/+/ /;

print "field = $field.<br>value=  $value. <br>", "\n";

$value =~ m/help/i;

print qq(Avant le mot help il y a "$`"), "\n";
print qq(Le mot help : "$&"), "\n";
print qq(Apres le mot help il y a "$'"), "\n";
```

Decoupage avec Split

Syntaxe :

```
@resultat = split(/motif/,$donnee);
```

Le résultat est de type array!

Exemple classique :

```
($field, $value) = split(/=/, $query);
```

Par exemple. la commande suivante découpe une chaîne en mots.
Chaque mot est une chaîne dans une array.

@mots = split(/ /, $query);

On peut ensuite découper chaque mot en lettre

```
foreach $mot (@mots)
{
    print qq(Pour le mot "$mot" les lettres sont : );
    @lettres = split(/./, $mot);
}
```

Construction de Motifs de recherche.

Les caractères spéciales sont [,], (,), *, ., ^, -, |, \, ?

Le Joker

Dans les motifs pour «match», «substitute» et «split», le caractère "." est un joker. Il match tout ou rien.

Exemple, `/p.p/` select les `pap, pop, p%p, pp,` etc.

Escape

Pour inclure un caractère spécial dans un motif, il faut le précéder par \.

exemple : Pour matcher un "." lui-même, il faut précéder avec "\"

```
$query = "3.14159";
($decimal, $fraction) = split (/\./, $query);
print "Decimal : $decimal,  Fraction = $fraction\n";
```

D'autre caractère spéciales sont

- "\ " space
- "\t" tab
- "\n" newline
- "\r" return
- "\f" formfeed

Match avec un groupe de caractères. .

Les caractères “[" et "]" permettent de définir une range.

Exemples :

[0-9] pour les numéros
[a-z] pour les lettres minuscules
[A-Z] pour les lettres majuscules.
[a-zA-Z] pur les miniscule et majuscules.

Exemple :

```
if ($query =~ m/[0-9]/) {print "numero $& detecte\n";}
```

Detection d'une exception

Noter une teste pour une exception avec "^".

exemple :

```
if ($query =~ m/[^0-9]/) {print "non-numéro $& detecté\n";}
```

Les raccourcisses des classes.

on peut utiliser

```
    \d      pour    [0-9]
    \D      pour    [^0-9]
    \W      pour    [^a-zA-Z0-9_]
    \s      pour    [\ \t\n\r\f]
                    #space, tab, newline, return, formfeed
    \S      pour    [^\ \t\n\r\f]
```

Suivre le symbole par {m, n} a fin de spécifier un minimum
de "m" et un maximum de "n".

Si on note que {m}, la valeur de "n" est par defaut, à l'infini.

Exemple : \d{5} permet de rechercher au mois 5 entières.

```
if ($query =~ m/\d{5}/) {print qq(5 numeros "$&"
detecte'\n);}
```

Quantification

Le "+" permet les correspondences multiples (minimum 1).

if ($query =~ m/1+/g) {print "numero $& detecte avant $'\n";}

Disjonction

Les alternatives peuvent être listé avec "|"

exemple : Pour reconnaître les numéros en style français ou américain
on utilise \.|\,

```
$query = "3,14159";
($decimal, $fraction) = split (/\.|\,/, $query);
print "Decimal = $decimal,  Fraction = $fraction\n";
```

Parentheses

Tout qui est matché dans les parenthèses est stocké dans une variable nommée $n ou "n" est un numéro.

Par exemple : m/(\d)/; affecte le résultat à $1.

exemple :

```
$query = "3.14159";
if ($query =~ m/(\d)\.(\d+)/)
{ print '$1 = ', $1, ' $2 = ', $2;}
```

résultat :

```
$1 = 3 $2 = 14159
```

Scripte PERL de Decodage

Dans notre scripte, nous avons décodés les paramètres avec :

```
$value =~ s/%([\dA-Fa-f][\dA-Fa-f])/pack("C", hex($1))/eg;
```

s - Substitute s/vieux/nouveau/
% - Les codes ascii commencent par "%"
() - affecter les résultats à la variable $1
[] - range de valeurs possibles
\d = Un numéro base 10.
A-Fa-f = les numéros base 16.

[\dA-Fa-f][\dA-Fa-f] - Deux numéros en base 16.
%[\dA-Fa-f][\dA-Fa-f] - Une code ascii en base 16.
%([\dA-Fa-f][\dA-Fa-f]) - Affecter un codes ascii en base 16 at $1.

Pack(-, -) - Compose un caractère avec les deux arguments.
"C" - la valeur ascii --
hex($1) = la valeur en base 16 du $1.
pack("C", hex($1)) - Créer une valeur ascii pour $1.
s/-/pack("C", hex($1))/e - exécuter la deuxième partie de la expression.
s/-/-/eg; Fait une substitution globale.

Les Valeurs Cachées (Hidden fields)

Dans la construction d'un page, on peut avoir besoins de stocker l'information acquis d'une forme précédente. Par exemple, on peut avoir besoin de conserver une hash %data. Ors, CGI est une protocole sans mémoire. Que faire?

Il faut utiliser les valeurs cachées.

Utilisez la commande (HTML) INPUT avec type HIDDEN.

```
<INPUT TYPE=hidden NAME=$name VALUE=$value>
```

Le résultat name=value apparaître dans les paramètres d'une appelle.

Par exemple, la scripte suivante encode un "hash" dans une page html sous forme de valeurs cachées.

```
foreach $key (keys %data)
{
 print "<INPUT TYPE=hidden NAME=$key VALUE=$data{$key}>\n";
}
```

Voici un exemple

```
<!-- s6.5.html -->

<HTML>
<TITLE>Script s6.5.cgi </TITLE>
<BODY   BGCOLOR=#FFFFFF >

<H1>Demonstration de valeurs caches. </H1>
<FORM ACTION ="/cgi-bin/jlc/s6.5.cgi" METHOD="get">
<H1>Veuillez completer ce formulaire</H1>
<P>Nom: <input name="name" size="48">
<p>
<SELECT NAME="langue" SIZE=1>
<OPTION>English <OPTION SELECTED>Francais <OPTION>Espanol
<OPTION>Italiano<OPTION>Dansk <OPTION>Svensk <OPTION>Suomi
</SELECT>
<br>
<INPUT TYPE=SUBMIT Value="Valider">
```

```
<INPUT TYPE=RESET Value="Reinitialiser">

</FORM>
</BODY>
</HTML>

<!--s6.5.cgi -->
#!/usr/local/bin/perl
#s6.5.cgi

$query = $ENV{'QUERY_STRING'};
@key_value_pairs = split(/&/, $query);
foreach $key_value (@key_value_pairs)
{
    ($key, $value) = split(/=/, $key_value);
    $value =~ tr/+/ /;
    $value =~ s/%([\dA-Fa-f][\dA-Fa-f])/pack("C",
hex($1))/eg;
    $data{$key} = $value;
}

print "Content-type: text/html", "\n\n";
print "<HTML><br>\n";
print "<TITLE>Page Dynamique s6.6.html </TITLE><br>\n";
print "<BODY  BGCOLOR=#FFFFFF>\n";
print "<H1>Demonstration Des Valeurs Caches. </H1><br>\n";
print qq(<FORM ACTION ="/cgi-bin/jlc/s6.6.cgi"
METHOD="get"><br>\n);
print qq(Les commandes suivants ont ete execute<p><PRE>\n);

#Create Hidden Variables
foreach $key (keys %data)
{
    print qq(<INPUT TYPE=HIDDEN NAME="$key"
VALUE="$data{$key}">);
    print qq(&lt;INPUT TYPE=HIDDEN NAME="$key"
VALUE="$data{$key}"&gt;\n);
}
print qq(</PRE><p>);
print qq(<INPUT TYPE=SUBMIT Value="Valider">\n);
print qq(<INPUT TYPE=RESET Value="Reinitialiser"><br>\n);
print "<HR><br><br>\n";
print "</BODY><br>\n";
print "</HTML><br>\n";
exit(0);
```

```
<!--  s6.6.cgi -->

#!/usr/local/bin/perl
#s6.6.cgi

$query = $ENV{'QUERY_STRING'};
@key_value_pairs = split(/&/, $query);
foreach $key_value (@key_value_pairs)
{
    ($key, $value) = split(/=/, $key_value);
    $value =~ tr/+/ /;
    $value =~ s/%([\dA-Fa-f][\dA-Fa-f])/pack("C",
hex($1))/eg;
    $data{$key} = $value;
}

#Print message based on Hidden Variables
print  "Content-type: text/html", "\n\n";
print qq(<HTML>\n);
print qq(<TITLE>Page Dynamique s6.6.html </TITLE>\n);
print qq(<BODY   BGCOLOR=#FFFFFF>\n);
print qq(<H1>Demonstration Des Valeurs Cachés. </H1>);
print qq(<br>\n);

if ($data{"langue"} eq "English")
   {print qq(Hello $data{"name"}\n); }
if ($data{"langue"} eq "Francais")
   {print qq(Bonjour $data{"name"}\n); }
if ($data{"langue"} eq "Espanole")
   {print qq(Buenas Dias $data{"name"}\n); }
if ($data{"langue"} eq "Italiano")
   {print qq(Bonjiorno $data{"name"}\n); }

print qq(<br>\n);
print qq(<INPUT TYPE=SUBMIT Value="restart demo">\n);
print qq(<HR><br>\n);
print qq(</BODY>\n);
print qq(</HTML>\n);

exit(0);
```

Cookies

Cookies permet d'identifier un utilisateur.

Un scripte CGI peut envoyer un "cookie" avec une identification unique à une utilisateur. Le scripte peut consulter les cookies d'un utilisateur afin d'adapter les réponses aux commandes.

Les cookies sont lus par une scripte CGI avec les variables de l'environnement.

Les cookies sont stockés par un script CGI par la commande html : Set-Cookie

Envois des Cookies.

La commande html Set-Cookie doit être exécuté AVANT la commande Content-Type.

En PERL, on écrit :

```
print "Set-Cookie:$name=$value\n";
```

La taille maximum de "value" est de 4K octet.
la valeur du cookie est déterminé par le scripte

Par exemple, soit qu'un utilisateur a choisi la française comme langue.
Dans une $query, nous avons :

.....&language=Francaise&....

Dans votre hash %data vous trouvez :

```
   "Francaise" pour  $data {"language"};
```

Vous pouvez stocker ceci dans le navigateur de l'utilisateur avec :

```
print "Set-Cookie:language=$data{"language"}\n";
print FILE "Content-type: text/html", "\n\n";
print FILE "<HTML>" , "\n";
...
print FILE "</HTML>" , "\n";
```

Le cookie est stocké temporairement, il est effacé par le navigateur quand l'utilisateur termine son execution du navigateur.

Vous pouvez forcer le stockage de la cookie par l'affection d'une date d'expiration en format jjj,dd-mmm-yyyy hh:mm:ss GMT. Ceci peut géner certaine utilisateurs.

Par exemple :

```
print "Set-Cookie:language=$data{"language"};
    expires=Thu, 23-Mar-2000  00:00:00 GMT\n";
```

Lecture des Cookie

Si le navigateur contient les cookies avec le nom de votre domaine, ils sont transmis dans une requête et stocké dans une variable de l'environnement HTTP_COOKIE

La scripte suivant construit un hash avec les cookies.

```
if ($ENV{'HTTP_COOKIE'})
{
    @cookies = split(/;/, $ENV{'HTTP_COOKIE'});
    foreach $cookie (@cookies)
    {
        ($name, $value) = split(/=/, $cookie);
        $cookie-hash($name) = $value;
    }
)
```

Voici un exemple

```
<!-- s5.7.html -->
<HTML>
<TITLE>Demonstration de Cookies.</TITLE>
<BODY   BGCOLOR=#FFFFFF >
<FORM ACTION ="/cgi-bin/jlc/s6.7.cgi" METHOD="get">
<H1>Demonstration de Cookies. </H1>

<H2>Veuillez completer ce formulaire</H2>
<P>Nom: <input name="name" size="48">
<p>
```

```
<SELECT NAME="language" SIZE=1>
<OPTION>English <OPTION SELECTED>Francais <OPTION>Espanol
<OPTION>Italiano<OPTION>Dansk <OPTION>Svensk <OPTION>Suomi
</SELECT>
<br>
<INPUT TYPE=SUBMIT Value="Valider">
<INPUT TYPE=RESET Value="Reinitialiser">

</FORM>
</BODY>
</HTML>


<!-- s6.7.cgi -->
#!/usr/local/bin/perl
#s5.7.cgi

$query = $ENV{'QUERY_STRING'};
@key_value_pairs = split(/&/, $query);
foreach $key_value (@key_value_pairs)
{
    ($key, $value) = split(/=/, $key_value);
    $value =~ tr/+/ /;
    $value =~ s/%([\dA-Fa-f][\dA-Fa-f])/pack("C",
hex($1))/eg;
    $data{$key} = $value;
}

#Set Cookies
print qq(Set-
Cookie:name=$data{"name"}&language=$data{"language"}\n);

#print page
print qq(Content-type: text/html\n\n);
print qq(<HTML>\n);
print qq(<HEAD>\n);
print qq(<TITLE>Page Dynamique s6.8.html </TITLE>\n);
print qq(</HEAD>\n);
print qq(<BODY BGCOLOR=#FFFFFF>\n);
print qq(<H1>Demonstration des Cookies </H1>\n);
print qq(<FORM ACTION ="/cgi-bin/jlc/s6.8.cgi"
METHOD="get">\n);
print qq(<br>\n);
print qq(<INPUT TYPE=SUBMIT Value="Valider">\n);
print qq(<HR><br>\n);
```

```
print qq(</BODY>\n);
print qq(</HTML>\n);
exit(0);

<!-- s5.8.cgi -->

#!/usr/local/bin/perl
#s6.8.cgi

if ($ENV{'HTTP_COOKIE'})
{
    @cookies = split(/&/, $ENV{'HTTP_COOKIE'});
    foreach $cookie (@cookies)
    {
       ($name, $value) = split(/=/, $cookie);
       $cookieHash{$name} = $value;
    }
}

#Print message based on Cookies
print qq(Content-type: text/html\n\n);
print qq(<HTML>\n);
print qq(<TITLE>Page Dynamique s6.8.html </TITLE>\n);
print qq(<BODY BGCOLOR=#FFFFFF>\n);
print qq(<H1>Demonstration des cookies </H1>\n);
print qq(<P>\n);
print qq(http_cookie: $ENV{'HTTP_COOKIE'}<p>\n);

if ($cookieHash{"language"} eq "English")
    {print qq(Hello $cookieHash{"name"}\n); }
if ($cookieHash{"language"} eq "Francais")
    {print qq(Bonjour $cookieHash{"name"}\n); }
if ($cookieHash{"language"} eq "Espanole")
    {print qq(Buenas Dias$cookieHash{"name"}\n); }
if ($cookieHash{"language"} eq "Italiano")
     {print qq(Bonjiorno $cookieHash{"name"}\n); }

print qq(<HR><br>\n);
print qq(</BODY>\n);
print qq(</HTML>\n);
exit(0);
```