

Scripturi CGI în Perl

O trecere în revistă a facilităților oferite de Perl în ceea ce privește programarea CGI

– Sabin Corneliu Buraga

Cine este Perl?

Fiind gratuit și posedând numeroase mijloace de documentare *online*, Perl poate fi folosit în special pentru dezvoltarea rapidă de aplicații de administrare a sistemului de operare și destinate Web-ului, reprezentând un mediu ideal pentru conceperea script-urilor CGI (vezi subcapitolul 2). Mai mult, anumite servere Web (*e.g.* Apache) includ interpretoare Perl interne. Mediul Perl se poate obține de pe Internet, via FTP sau HTTP, prin intermediul locațiilor *CPAN* (*Comprehensive Perl Archive Network*). Principala sursă este `ftp://ftp.funet.fi`, dar se pot folosi și alte locații, listate la `http://www.perl.com/CPAN/`. De asemenea, orice distribuție actuală de Linux include interpretorul și manualele standard Perl. Pentru alte platforme, Perl poate fi obținut de la adresele amintite mai sus. Spre deosebire de alte limbaje, Perl este un limbaj *interpretat*, în sensul că instrucțiunile Perl nu sunt convertite în cod executabil (nu se generează un fișier executabil, spre a fi rulat independent de interpretorul Perl). Vom spune despre programele Perl că sunt *scripturi*, un limbaj de tip script fiind destinat să prelucreze, să automatizeze și să integreze facilitățile oferite de un anumit sistem (*e.g.* sistem de operare, server Web, navigator Web, aplicație de birou). Alte limbaje de tip script sunt *bash*, *Python*, *Tcl/Tk* ori *JavaScript*.

Având în vedere că Perl este un interpretor (similar *shell*-ului *bash*), prima linie a unui fișier sursă Perl va trebui să fie următoarea:

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

Această linie indică încărcătorului sistemului de operare locația interpretorului Perl (s-ar putea în unele cazuri să difere de directorul `/usr/bin`, dați `whereis perl` pentru a vedea unde a fost instalat). Ea va fi ignorată în alte medii diferite de UNIX (Linux), fiind considerată simplu comentariu.

Pentru a putea fi executat, fișierul memorând scriptul Perl va trebui să aibă permisiunea de execuție setată prin comanda `chmod`. Orice

script Perl va trebui să poată fi citit și executat de ceilalți utilizatori, deci în mod uzual vom folosi `chmod 755 nume_script.pl.cgi`.

Scripturi CGI în Perl

Înainte de a trimite spre navigator pagini Web sau alte tipuri de informații (multimedia, arhive, documente XML etc.), orice script CGI trebuie să afișeze la ieșirea standard câmpul HTTP `Content-type`. Desigur, nici scripturile CGI concepute în Perl nu fac excepție. Astfel, cel mai simplu script este următorul:

```
#!/usr/bin/perl

# trimitem antetul HTTP
print "Content-type:
text/html\n\n";

# trimitem codul HTML,
# (folosim facilitatea
"here")
print <<HTML;
<html>
  <head>
    <title>Salut!</title>
  </head>
  <body bgcolor="white"
    text="blue">
    <p>Salut!</p>
  </body>
</html>
HTML
```

Ca orice alt script CGI conceput în alt limbaj, un script CGI scris în Perl va avea acces la variabilele de mediu furnizate de serverul Web. Acest lucru se poate rezolva prin accesarea tabloului asociativ `ENV`:

```
#!/usr/bin/perl

# trimitem antetul HTTP
print "Content-type: text/plain\n\n";

print "Variabilele de mediu disponibile:\n";

foreach $variabila (sort keys %ENV) {
  print "$variabila: $ENV{$variabila}\n";
}
```

Modulul CGI

Pentru realizarea comodă de scripturi CGI în Perl, este pus la dispoziție modulul *CGI*, utilizat în special pentru generarea și procesarea formularelor și a *cookie*-urilor. Acest modul oferă un obiect generic CGI pentru acces la variabile de mediu, pentru procesarea lor și stocarea rezultatelor. Modulul CGI poate fi utilizat pentru preluarea datelor transmise atât prin metoda GET, cât și prin metoda POST, fără a concepe programe separate pentru fiecare metodă în parte. Un alt avantaj este dat de posibilitatea de a depana scripturile CGI rulându-le direct de la *prompt*-ul UNIX, în loc de a fi invocat prin intermediul serverului Web.

Modalități de utilizare Putem folosi modulul CGI prin intermediul a două paradigme de programare: *funcțională (procedurală)* și *obiectuală*. Cele două paradigme nu diferă decât prin modul de acces la funcționalitățile modulului: via funcții în primul caz și via metode în al doilea.

Următorul exemplu folosește paradigma procedurală:

```
#!/usr/bin/perl

# utilizam modulul CGI in forma standard
use CGI qw/:standard/;

# trimitem antetul HTTP
```

Listing 1. Controrizarea accesului la o pagină web (contor.pl.cgi)

```
#!/usr/bin/perl

$contor = '/home/httpd/contor.data';
$lacat = '/home/httpd/contor.LOCK';

print "Content-type: text/html";
&incrementeaza;
print $accesari;

sub incrementeaza {
    # citim numarul de accesari din fisier
    open(CONTOR, $contor) ||
        die "Eroare la deschiderea contorului.\n";
    $accesari = <CONTOR>;
    $accesari++;
    close(CONTOR);
    # scriere concurenta
    # verificam existenta fisierului lacat
    # exista, deci alta instanta a scriptului
    # actualizeaza continutul contorului
    while (-e $lacat) {
        sleep 2; # asteptam 2 secunde
    }
    # putem scrie in fisier, cream lacatul
    open(LACAT, ">$lacat") ||
        die "Eroare la crearea lacatului.\n";
    close(LACAT);
    # scriem valoarea incrementata
    open(CONTOR, ">$contor") ||
        die "Eroare la actualizarea contorului";
    print CONTOR "$accesari\n";
    close(CONTOR);
    # stergem lacatul
    unlink($lacat);
}
```

```
print header();

# afisam antetul paginii Web
print start_html(-title => "Un salut");

# afisam diferite elemente HTML
print h1('Salut!'),
    p('Un paragraf...');

# afisam finalul de document
print end_html();

Același script, din perspectiva orientată-obiect, este:
```

```
#!/usr/bin/perl

# utilizam modulul CGI
use CGI;

# instantiem obiectul CGI
$c = new CGI;

# trimitem antetul HTTP
print $c->header();

# afisam antetul paginii Web
print $c->start_html(-title => "Un salut");

# afisam diferite elemente HTML
print $c->h1('Salut!'),
    $c->p('Un paragraf...');

# afisam finalul de document
print $c->end_html();
```

Inițializarea obiectului CGI Metoda `new()` poate instanția obiectul CGI în diverse moduri, oferind posibilitatea de a citi parametrii de intrare via un descriptor de fișier prin construcția:

```
$c = new CGI(HANDLER);

Vor fi citite perechi nume=valoare delimitate de caractere new line sau un șir de interogare codificat conform specificațiilor CGI. O altă metodă de a inițializa obiectul CGI este cea prin intermediul tablourilor asociative care vor conține nume de câmpuri și valori:
```

```
$c = new CGI({'culoare' => 'verde',
             'nume' => 'Sabin',
             'prieteni' => [qw/Victor Stefan/]});
```

O altă manieră este următoarea (pasăm ca argument un șir de interogare URI codificat):

```
$c = new CGI('culoare=verde&nume=Sabin');
```

Preluarea parametrilor Cele mai uzuale utilizări ale modulului CGI sunt cele în care sunt implicate formularele Web ale căror valori de câmpuri trebuie prelucrate comod. Pentru a prelua toți parametrii pasați scriptului ne putem sluji de un tablou, apelând metoda `param()`:

```
@parametri = $c->param;
```

Dacă dorim să preluăm valoarea unui anumit parametru vom folosi una dintre construcțiile:

```
@prieteni = $c->param('prieteni');  
$culoare = $c->param('culoare');
```

În prima variantă, rezultatul este preluat de un tablou, deoarece câmpul prieteni poate conține elemente multiple ale unui marcator <select>.

Varianta procedurală este:

```
$o_culoare = param('culoare');
```

Atunci când dorim să asignăm o nouă valoare unui parametru, vom scrie, de exemplu:

```
$c->param(-name=>'culoare', -value=>'rosu');
```

Mai pot fi folosite și metodele:

- `append()` adaugă un nou parametru;
- `delete()` șterge un parametru;
- `delete_all()` elimină toți parametrii;
- `save()` salvează starea unui formular (șirul de interogare), utilizând un descriptor de fișier;
- `url()` furnizează URL-ul scriptului.

Procesarea antetului HTTP Așa cum am văzut, înainte de a genera cod-sursă HTML, un script CGI trebuie să trimită obligatoriu antetul HTTP. Acest lucru se realizează prin intermediul metodei sau funcției `header()`:

```
# trimite Content-type: image/gif  
print $c->header('image/gif');
```

Metoda `header()` poate fi folosită și pentru a seta alte câmpuri HTTP:

```
print $c->header(  
  # Content-type  
  -type => 'image/png',  
  # codul de stare HTTP  
  -status => '402 Payment Required',  
  # timpul de expirare  
  -expires => '+3d',  
  # parametru-utilizator  
  -Cost => '$ 0.01');
```

Pentru atributul `-expires` pot fi specificate valori de timp precum `now` (acum), `+30s` (după 30 de secunde), `+15m` (15 minute), `+5h` (5 ore) sau `+3M` (3 luni). Sunt acceptate și valori negative.

Pot fi, de asemenea, trimise câmpuri definite de utilizatori, în exemplul de mai sus `Cost`. Acest lucru permite folosirea unor protocoale noi, fără a trebui să actualizăm modulul CGI (e.g. putem expedia câmpuri specifice protocolului SOAP – Simple Object Access Protocol).

Mai mult, se poate folosi metoda `redirect()` pentru a redirecta navigatorul către altă locație:

```
# redirectare in functie de limba  
if ($limba eq 'ro')  
  print $c->redirect('/ro/index.html');  
else  
  print $c->redirect('/en/index.html');
```

De asemenea, se pot invoca diverse metode care să ofere valorile variabilelor de mediu specifice HTTP. Astfel, putem apela metode precum:

- `auth_type()` furnizează tipul autentificării (e.g. Basic);
- `query_string` returnează șirul de interogare CGI;

- `remote_addr()` furnizează adresa IP a calculatorului client care a invocat scriptul;
 - `remote_host()` ca mai sus, dar se returnează numele simbolic al calculatorului client;
 - `request_method()` furnizează metoda HTTP utilizată (GET, POST sau HEAD);
 - `server_name()` returnează numele serverului Web pe care rulează scriptul;
 - `server_port()` returnează portul de acces folosit în comunicația dintre server și client;
 - `user_agent()` identifică numele și versiunea agentului-utilizator (navigatorului, de cele mai multe ori) folosit pe calculatorul client.
- Crearea de marcatori HTML** Putem folosi următoarele metode pentru generarea antetului și finalului unui document HTML:
- `start_html()` construiește antetul unei pagini Web:

```
print $c->start_html(  
  -title => 'Facultatea de Informatica',  
  -author => 'busaco@infoiasi.ro',  
  -meta => {'keywords' => 'CS, Romania, Iasi'},  
  -style => {'src' => 'styles/fcs.css'},  
  -bgcolor => 'white',  
  -text => 'navy');
```

Pot fi utilizate și atributele:

- `script` definește funcții JavaScript încorporate în antetul unui document HTML;
 - `onLoad` specifică instrucțiunile JavaScript rulate la apariția evenimentului de încărcare a paginii;
 - `onUnload` definește codul JavaScript invocat la apariția evenimentului de părăsire a paginii Web.
- `end_html()` termină o pagină HTML.

Pentru fiecare element HTML poate fi invocată metoda purtând același nume cu al marcatorului dorit. De exemplu, metode de generare a elementelor vide (e.g. <hr> sau
):

```
print $c->hr;
```

De asemenea, se pot invoca metode de generare a elementelor având *tag*-uri de început și de sfârșit (e.g. , <h3> sau <a>):

```
print $c->h3("Sunt ", $c->em("inclinat"), " sa cred...");
```

Metode pentru generarea elementelor de formular HTML

Metodă	Descriere
<code>startform()</code>	marchează începutul unui formular
<code>endform()</code>	finalizează un formular
<code>textfield()</code>	crează un câmp de tip text
<code>textarea()</code>	generează un element <textarea>
<code>password_field()</code>	crează un câmp de tip password
<code>filefield()</code>	crează un câmp de tip file , utilizat pentru acțiunea de upload
<code>popup_menu()</code>	crează un meniu cu opțiuni (posibil multiple)
<code>scrolling_list()</code>	crează o listă cu opțiuni (posibil multiple)
<code>checkbox()</code>	generează un singur câmp de tip checkbox
<code>checkbox_group()</code>	generează un grup de butoane de bifare de tip checkbox
<code>radio_group()</code>	generează un grup de butoane de tip radio
<code>submit()</code>	crează un buton de tip submit
<code>reset()</code>	crează un buton de tip reset
<code>hidden()</code>	generează un câmp ascuns, de tip hidden
<code>button()</code>	generează un buton generic (în mod obișnuit folosit în conjuncție cu un script JavaScript)

Un alt exemplu, care generează o listă neordonată de hiperlegături, este:

```
use CGI qw/:standard/;

print h2("Despre noi..."),
      ul(
        li(a({href=>"http://www.infoiasi.ro/~stanasa/"}),
           "Stefan")),
        li(a({href=>"http://www.cs.tuiasi.ro/mituc/"}),
           "Victor")),
        li(a({href=>"http://www.infoiasi.ro/~busaco/"}),
           "Sabin"));
```

Pentru a genera cele mai multe marcaje HTML vom invoca funcțiile/metodele corespunzătoare scrise cu minuscule, cu următoarele excepții:

- pentru a construi elemente <tr> se va folosi Tr() ori TR(), deoarece tr() este funcție Perl standard;
- pentru a genera <param> (subelement al marcatorelor <applet> sau <object>), vom invoca PARAM();

- pentru a genera <select> se va utiliza funcția Select() în loc de funcția standard select().

Vom atașa atribute elementelor HTML, prin pasarea fiecărei funcții corespunzătoare unui element a unui tablou asociativ conținând valorile acestor atribute. Un exemplu:

```
use CGI qw/:standard/;

# generam o imagine

print img(src => 'fig.gif',
          align => 'right',
          alt => 'Figura');
```

Un alt exemplu, în care vom afișa un tablou al cărui conținut este generat prin program:

```
#!/usr/bin/perl

use CGI qw/:standard/;
```

Listing 2. Memorarea preferințelor utilizatorilor

```
#!/usr/bin/perl

use CGI;

# constante utilizate in cadrul formularului
# culori de fundal dorite
@culori=qw/gray coral bisque beige gold green lime linen
         orchid seashell sienna silver wheat/;
# dimensiunea fontului
@marime=("<implicit>", 1..7);

# instantiem obiectul CGI
$c = new CGI;

# preluam vechile preferinte din cookie-ul "preferinte"
%preferinte = $c->cookie('preferinte');

# preluam noile preferinte ale utilizatorului prin
# inspectarea valorilor transmise prin formular
$preferinte{'culoare'} = $c->param('culoare')
  if $c->param('culoare');
$preferinte{'nume'} = $c->param('nume')
  if $c->param('nume');
$preferinte{'marime'} = $c->param('marime')
  if $c->param('marime');

# alegem culoarea implicita 'silver' daca nu exista
$preferinte{'culoare'} = 'silver'
  unless $preferinte{'culoare'};

# modificam parametrii cookie-ului astfel incit
# sa fie persistent si sa reflecte noile preferinte
$un_cookie = $c->cookie(-name=>'preferinte',
                      -value=> \%preferinte,
                      -expires=>'60d');

# trimitem cookie-ul
print $c->header(-cookie=>$un_cookie);

# generam titlul paginii, incluzind numele utilizatorului
$title = $preferinte{'nume'} ?
  "Pagina lui $preferinte{nume}!" :
  "Pagina utilizatorului";

# vom crea pagina HTML, oferind posibilitatea
# de a schimba preferintele de
# culoare, nume de utilizator si marimea fontului
print $c->start_html(-title=>$title,
                    -bgcolor=>$preferinte{'culoare'});

# stabilim marimea fontului
print "<basefont size=$preferinte{marime}>\n"
  if $preferinte{'marime'} > 0;

print <<END;
<h3 align="center">$title</h3>
<hr>
<p align="justify">
Modificati modul de aparitie al paginii
completand formularul de mai jos.
Preferintele dumneavoastra vor fi
valabile timp de maxim 60 de zile.</p>
END
;

# vom crea formularul de preferinte
print join("\n",
  "<hr>",
  $c->start_form,
  "Prenumele d-voastra: ",
  $c->textfield(-name=>'nume',
              -default=>$preferinte{'nume'},
              -size=>30),
  "<br>",
  "Culoarea de fundal preferata: ",
  $c->popup_menu(-name=>'culoare',
               -values=>\@culori,
               -default=>$preferinte{'culoare'}),
  "Marimea fontului: ",
  $c->popup_menu(-name=>'marime',
               -values=>\@marime,
               -default=>$preferinte{'marime'}),
  "<br>",
  $c->submit(-label=>'Memoreaza preferintele'),
  "<hr>");
```

```
print header;

@valori = (1..33);
@antete = ('n', 'n/2', 'n' . sup('2'));
@rinduri = th(\@antete);

foreach $rind (@valori) {
    push(@rinduri,
        Tr({-align=>'center'},
            td($rind), td($rind/2), td($rind**2)));
}
print table({-border=>'1', -width=>'50%'},
            @rinduri);
```

Crearea formularelor Web Modulul CGI permite generarea formularelor HTML într-o manieră simplă și flexibilă, putând fi utilizate metodele prezentate în tabelul *Metode pentru generarea elementelor de formular HTML*.

Alte funcții utile oferite de modulul CGI sunt:

- `escape()` convertește un șir de caractere în codificarea utilizată de URI-urile CGI;
- `unescape()` convertește un șir codificat CGI în reprezentarea sa normală;

```
use CGI qw/escape unescape/;
```

```
$sir = escape('~/:#?');
print unescape($sir);
```

- `escapeHTML()` convertește un șir de caractere, substituind orice caracter HTML ilegal prin entitatea corespunzătoare;
- `unescapeHTML()` convertește un șir de caractere conținând entități HTML în șir obișnuit.

```
use CGI qw/escapeHTML unescapeHTML/;
```

```
$sir = escapeHTML('Un sir mai <mic>...');
print unescapeHTML($sir);
```

Posibilități de depanare Înainte de a fi operaționale efectiv pe Web, scripturile CGI trebuie atent verificate și depanate. Modulul CGI oferă suport și pentru aceste activități importante, programatorul putând apela un script Perl direct de la *prompt*-ul sistemului. Parametrii care în mod normal trebuiau preluați via variabila de mediu `QUERY_STRING` sau de la intrarea standard pot fi furnizați ca argumente în linia de comandă:

```
(infoiasi)$ ./preferinte.pl nume=Sabin culoare=verde
```

Mai mult, scriptul poate fi apelat fără argumente, iar valorile vor fi citite de la intrarea standard (fiecare construcție de forma `nume=valoare` fiind urmată de un caracter linie nouă).

Exemple de scripturi Perl

În cele ce urmează vom furniza două exemple de scripturi concepute în limbajul Perl, ilustrând facilitățile puse la dispoziție de modulul CGI.

Contorizarea accesului la o pagină Web Ne propunem să memorăm într-un fișier numărul de vizite ale utilizatorilor unei anumite pagini Web. Din cauza faptului că actualizarea conținutului acestui fișier se poate realiza concurrent (mai multe persoane pot încărca pagina concomitent), vom utiliza un fișier lacăt care să indice dacă putem reactualiza conținutul fișierului de contorizare. Codul sursă al scriptului, stocat în fișierul `contor.pl.cgi` este prezentat în *Listing 1*.

Acest script va fi invocat la fiecare încărcare a unui document HTML, folosind directiva SSI exec:

```
<html>
<head>
  <title>...</title>
</head>
<body>
  ...
  <p>Sunteti vizitatorul cu numarul:
  <!--exec cgi="/cgi-bin/contor.pl.cgi" -->
</body>
</html>
```

Memorarea preferințelor utilizatorilor În cadrul acestui script vom putea observa capacitățile oferite de modulul CGI în ceea ce privește manipularea *cookie*-urilor. Utilizatorul va putea să introducă prin intermediul unui formular interactiv numele său, culoarea de fundal a paginii și mărimea fontului implicit. Aceste preferințe vor fi stocate în *cookie*-uri pe mașina client a utilizatorului până la următoarea vizită sau maxim 60 de zile. La invocarea scriptului se verifică dacă preferințele există și se modifică înfățișarea paginii în concordanță cu acestea (vezi *Listing 2*).

În loc de concluzii...

Unde cum am văzut în cele prezentate mai sus, modulul CGI oferă o serie de funcționalități utile dezvoltatorilor de pagini HTML generate în mod dinamic de diverse scripturi CGI concepute în limbajul Perl. Cititorii interesați pot experimenta diferite alte aplicații Web pe care să încerce să le implementeze complet în Perl, ținând cont că unelele de dezvoltare Perl sunt disponibile gratuit și sunt independente de platforma folosită.

Într-un viitor articol vom vedea cum pot fi prelucrate comod documentele XML folosind limbajul Perl.

Sabin Corneliu Buraga este doctorand în Computer Science la Facultatea de Informatică, Universitatea „Al. I. Cuza” din Iași și poate fi contactat la busaco@infoiasi.ro sau <http://www.infoiasi.ro/~busaco/>. ■ 98

Resurse

- S. Asbury et al., *CGI How-To*, Macmillan Computer Publishing, 1996
- S. Buraga, *Tehnologii Web*, Editura Matrix Rom, București, 2001: <http://www.infoiasi.ro/~busaco/books/web.html>
- S. Buraga, *Cookie-uri*, NET Report, vol.9, 10 (97), oct. 2000
- S. Buraga, Ș. Andrei, *CGI-uri versus servlet-uri (partea I)*, NET Report, vol.10, 12 (111), dec. 2001
- S. Buraga, Ș. Andrei, *CGI-uri versus servlet-uri (partea II)*, NET Report, vol. 11, 01 (112), ian. 2002
- Ș. Trăușan-Matu et al., *Prelucrarea documentelor folosind XML și Perl*, Editura Matrix Rom, București, 2001
- L. Wall et al., *Programming Perl (Third Edition)*, O'Reilly & Associates, Cambridge, 2000
- ***, CPAN: <http://www.perl.com/CPAN>