

# Capítulo 2: web dinámica



Perl CGI.pm



- CGI: Common Gateway Interface
  - Un servidor web es comúnmente usado como una entrada a una aplicación
    - Acceso a documentos “legacy”
    - Interacción con base de datos
  - CGI es un acuerdo entre los implementadores de servidores HTTP
    - La idea es integrar scripts y programas de acceso a los datos
  - Uso típico en formularios HTML para construir aplicaciones con bases de datos
- Un CGI se ejecuta siempre en tiempo-real
  - Genera información dinámica
    - A diferencia de los documentos estáticos



- Un programa CGI es un ejecutable
  - Permite a todo el mundo ejecutar un programa en el servidor
  - Típicamente residen en un directorio “cgi-bin”
  - Puede ser escrito en cualquier lenguaje que se pueda ejecutar en el servidor
    - C/C++, Fortran, Perl, etcétera
  - CGI en C (por ejemplo) necesitan ser compilados
    - Típicamente existe el directorio “cgi-src”
  - Muchos prefieren escribir CGI scripts
    - Más fácil para debugear, modificar y mantener



- CGI y Perl
  - Usaremos módulo CGI.pm
    - Define objetos CGI: contiene valores del requerimiento y valores de estado
    - Se pueden examinar los parámetros que recibe el script
    - Permite crear formularios con valores iniciales
    - Provee funciones que generan HTML
    - Soporte para subir archivos, css, cookies, etc
  - Estilo de programación
    - Orientado a Objeto: se crea uno o varios objetos y se invocan sus métodos
    - Orientado a la función: se invocan las funciones directamente



- Orientado al objeto

```
#!/usr/bin/perl -w

use CGI;                                # load CGI routines
$q = new CGI;                           # create new CGI object
print $q->header,                        # create the HTTP header
      $q->start_html('hello world'),    # start the HTML
      $q->h1('hello world'),           # level 1 header
      $q->end_html;                     # end the HTML
```

- Orientado a la función

```
#!/usr/bin/perl -w

use CGI qw/:standar/;                  # load CGI routines

print header,                          # create the HTTP header
      start_html('hello world'),      # start the HTML
      h1('hello world'),              # level 1 header
      end_html;                       # end the HTML
```



- Invocación de rutinas

- Hay rutinas que aceptan hasta 20 argumentos
- Se utilizan un “-” y nombre:

```
print $q->header(-type=>'image/gif',-expires=>'+3d');
```

- Se pueden invocar con sólo un argumento
  - En estos casos se puede obviar el nombre del argumento

```
print $q->header('text/html');
```

- Para el caso de “header” el argumento es el document-type
- Los argumentos pueden ser variables escalares, referencia a un arreglo o hash

```
$q->param(-name=>'veggie',-value=>'tomato');
```

```
$q->param(-name=>'veggie',-value=>['tomato','tomahto','potato','potahto']);
```



- Generación dinámica de HTML

Code	Generated HTML
----	-----
<code>h1()</code>	<code>&lt;h1&gt;</code>
<code>h1('some','contents');</code>	<code>&lt;h1&gt;some contents&lt;/h1&gt;</code>
<code>h1({-align=&gt;left});</code>	<code>&lt;h1 align="LEFT"&gt;</code>
<code>h1({-align=&gt;left},'contents');</code>	<code>&lt;h1 align="LEFT"&gt;contents&lt;/h1&gt;</code>

- Se pueden generar cabeceras HTTP no estándares:

```
print $q->header(-type => 'text/html',  
  -cost => 'Three smackers',  
  -annoyance_level => 'high',  
  -complaints_to => 'bit bucket');
```

```
HTTP/1.0 200 OK  
Cost: Three smackers  
Annoyance-level: high  
Complaints-to: bit bucket  
Content-type: text/html
```



- Crear un nuevo objeto

```
$query = new CGI;
```

- Se asigna la entrada de los métodos GET o POST en objeto \$query

- Se puede crear a partir de un archivo con parámetros

```
$query = new CGI(INPUTFILE);
```

- Las líneas deben tener forma TAG=VALUE
- Este tipo de archivo se puede crear con el método “save” de \$query





- También se puede inicializar con valores

```
$query = new CGI( {'dinosaur'=>'barney',  
                  'song'=>'I love you',  
                  'friends'=>[qw/Jessica George Nancy/]}  
);
```

```
$query = new CGI('dinosaur=barney&color=purple');
```

- Obtener los nombres de los parámetros enviados al script

```
@names = $query->param
```

- Retorna los nombres de los parámetros como una lista



- Obtener los valores de los parámetros

```
@values = $query->param('foo');
```

```
$value = $query->param('foo');
```

- Si el parámetro no tiene valor, retornará un string vacío
  - Ejemplo: "name1=&name2=" o "name1&name2"
- Si el parámetro no existe retornará “undef” en contexto escalar
  - Arreglo vacío en contexto de arreglo



- Asignar valores a los parámetros

```
$query->param('foo','an','array','of','values');
```

```
$query->param(-name=>'foo',-values=>['an','array','of','values']);
```

```
$query->param(-name=>'foo',-value=>'the value');
```

```
$query->append(-name=>'foo',-values=>['yet','more','values']);
```

- Borrar parámetros

```
$query->delete('foo','bar','baz');
```

```
$query->delete_all();
```



- Obtener lista de parámetros como un hash

```
$params = $q->Vars;  
print $params->{'address'};  
  
@foo = split("\0", $params->{'foo'});  
  
%params = $q->Vars;
```

- Recuperar errores

```
my $error = $q->cgi_error;  
if ($error) {  
    print $q->header(-status=>$error),  
          $q->start_html('Problems'),  
          $q->h2('Request not processed'),  
          $q->strong($error);  
    exit 0;  
}
```



- Crear cabecera estándar HTTP

```
print header;  
  
print header('image/gif');  
  
print header('text/html','204 No response');  
  
print header(-type=>'image/gif',  
             -nph=>1,  
             -status=>'402 Payment required',  
             -expires=>'+3d',  
             -cookie=>$cookie,  
             -charset=>'utf-7',  
             -attachment=>'foo.gif',  
             -Cost=>'$2.00');
```



- Muchos browser no guardan en caché el resultado de un CGI
  - Cada vez que se recarga la página se ejecuta el script
  - Para que guarde en caché se puede utilizar -expires

+30s

+10m

+1h

-1d

now

+3M

+10y

Thursday, 25-Apr-1999 00:40:33 GMT

30 seconds from now

ten minutes from now

one hour from now

yesterday (i.e. "ASAP!")

immediately

in three months

in ten years time

at the indicated time & date



- Generando header HTML

```
print start_html(-title=>'Secrets of the Pyramids',  
                 -author=>'fred@capricorn.org',  
                 -base=>'true',  
                 -target=>'_blank',  
                 -meta=>{'keywords'=>'pharaoh secret mummy',  
                         'copyright'=>'copyright 1996 King Tut'},  
                 -style=>{'src'=>'/styles/style1.css'},  
                 -BGCOLOR=>'blue');
```

- Genera el header HTML y <body>
- Se puede agregar un atributo de idioma

```
print $q->start_html(-lang=>'fr-CA');
```



- Incluir Javascript

```
$query = new CGI;
print header;
$JSCRIPT=<<END;
// Ask a silly question
function riddle_me_this() {
    var r = prompt("What walks on four legs in the morning, " +
        "two legs in the afternoon, " +
        "and three legs in the evening?");
    response(r);
}
// Get a silly answer
function response(answer) {
    if (answer == "man")
        alert("Right you are!");
    else
        alert("Wrong! Guess again.");
}
END
print start_html(-title=>'The Riddle of the Sphinx',
    -script=>$JSCRIPT);
```





- Incluir Javascript
  - Usando el tag “script” con “type” y “src”

```
print $q->start_html(-title=>'The Riddle of the Sphinx',  
    -script=>{-type=>'JAVASCRIPT',  
    -src=>'/javascript/sphinx.js'}  
);
```

```
print $q->(-title=>'The Riddle of the Sphinx',  
    -script=>{-type=>'PERLSCRIPT',  
    -code=>'print "hello world!\n;""}  
);
```



- Crear elementos estándares HTML

```
print $q->blockquote(  
    "Many years ago on the island of",  
    $q->a({href=>"http://crete.org/"}, "Crete"),  
    "there lived a Minotaur named",  
    $q->strong("Fred."),  
    ),  
    $q->hr;
```

- El código HTML generado:

```
<blockquote>  
Many years ago on the island of  
<a href="http://crete.org/">Crete</a> there lived  
a minotaur named <strong>Fred.</strong>  
</blockquote>  
<hr>
```



- Propiedad distributiva de los atributos

```
print ul(  
    li({-type=>'disc'},[ 'Sneezy','Doc','Sleepy','Happy' ])  
);
```

– Genera:

```
<ul>  
    <li type="disc">Sneezy</li>  
    <li type="disc">Doc</li>  
    <li type="disc">Sleepy</li>  
    <li type="disc">Happy</li>  
</ul>
```



- Funciones especiales
  - Por colisiones de nombres con funciones Perl, comienzan con mayúscula:

Select

Tr

Link

Delete

Accept

Sub