# COMP284 Scripting Languages
## Lecture 8: Perl (Part 7)
## Handouts



Department of Computer
School of Electrical Engineering, Electronics, an
University of Liverpo

# Contents

# Common Gateway Interface — CGI

The Common Gateway Interface (CGI) is a standard method for web servers to use an external application, a CGI program, to dynamically generate web pages

1. A web client generates a client request, for example, from a HTML form, and sends it to a web server
2. The w
conv
3. The C
the server passes the program's response back to t

# Client requests

In the following we focus on client requests that are generated using HTML forms



```
<!DOCTYPE html>
<html>
<head><title>My HTML Form</title></head>
<body>
<form action
 "http://
 method="
<label>Enter your user name:
    <input type="text" name="username"></label><br>
<label>Enter your full name:
    <input type="text" name="fullname"></label><br>
<input type="submit" value="Click␣for␣response">
</form>
</body>
</html>
```

# Client requests

In the following we focus on client requests that are generated using HTML forms

```
<!DOCTYPE html>
<html>
<head><title>My HTML Form</title></head>
<body>
<form action="h
 method="pos
<label>Enter yo
<label>Enter yo
<input type="su
</form>
</body>
</html>
```

# Encoding of input data

- Input data from an HTML form is sent URL-encoded as sequence of key-value pairs: `key1=value1&key2=value2&...`

  Example:

  `username=dave&fullname=David%20Davidson`

- All cha... ...racters) are enc...
- ASCII characters that are not unreserved charact... using ASCII codes (preceded by %)
  - A space is represented as %20 or +
  - + is represented as %2B
  - % is represented as %25

  Examples:

  `username=cath&fullname=Catherine+O%27Donnell`

# Request methods: GET versus POST

The two main request methods used with HTML forms are `GET` and `POST`:

- `GET`:
  - Form data is appended to the URI in the request
    <
    <
- For



Example:

```
GET /cgi-bin/cgiwrap/ulrich/demo?username=dave&
fullname=David+Davidson HTTP/1.1
Host: cgi.csc.liv.ac.uk
```

# Request methods: GET versus POST

The two main request methods used with HTML forms are `GET` and `POST`:



- `POST`
  - Form data is appended to end of the request (after headers and blank line)
  - For
  - For



Example

```
POST /cgi-bin/cgiwrap/ullrich/demo HTTP/1.1
Host: cgi.csc.liv.ac.uk

username=dave&fullname=David+Davidson
```

# Environment variables: GET

| Env variable | Meaning |
|---|---|
| QUERY_STRING | The query information passed to the program |
| REQUEST_METHOD | The request method that was used |
| PATH_INFO | Extra path information passed to a CGI program |
| PATH_ | ysical |
| SCRIP | |
| SCRIP | |



Example (1):

```
GET http://cgi.csc.liv.ac.uk/cgi-bin/cgiwrap/ullrich/demo/more/dirs?
    username=dave&fullname=David+Davidson
```

```
QUERY_STRING        username=dave&fullname=David+Davidson
REQUEST_METHOD      GET
PATH_INFO           /more/dirs
PATH_TRANSLATED     /users/www/external/docs/more/dirs
SCRIPT_NAME         /cgi-bin/cgiwrap/ullrich/demo
SCRIPT_FILENAME     /users/loco/ullrich/public_html/cgi-bin/demo

STDIN
# empty
```

# Environment variables: GET

| Env variable | Meaning |
|---|---|
| QUERY_STRING | The query information passed to the program |
| REQUEST_METHOD | The request method that was used |
| PATH_INFO | Extra path information passed to a CGI program |
| PATH_ | ysical |
| SCRIP | |
| SCRIP | |



Example (2):

```
GET http://cgi.csc.liv.ac.uk/cgi-bin/cgiwrap/ullrich/demo/more/dirs?
    username=2%60n+d%2Bt+e+s%27t&fullname=Peter+Newton
```

```
QUERY_STRING        username=2%60n+d%2Bt+e+s%27t&fullname=Peter+Newton
REQUEST_METHOD      GET
PATH_INFO           /more/dirs
PATH_TRANSLATED     /users/www/external/docs/more/dirs
SCRIPT_NAME         /cgi-bin/cgiwrap/ullrich/demo
SCRIPT_FILENAME     /users/loco/ullrich/public_html/cgi-bin/demo

STDIN
# empty
```

# Environment variables: POST

| Env variable | Meaning |
|---|---|
| QUERY_STRING | The query information passed to the program |
| REQUEST_METHOD | The request method that was used |
| SCRIPT_NAME | The relative virtual path of the CGI program |
| SCRIP | |





Example

```
POST /cgi-bin/cg
Host: cgi.csc.liv.ac.uk

username=2%60n+d%2Bt+e+s%27t&fullname=Peter+Newton
```



```
QUERY_STRING
# empty
REQUEST_METHOD      POST
SCRIPT_NAME         /cgi-bin/cgiwrap/ullrich/demo
SCRIPT_FILENAME     /users/loco/ullrich/public_html/cgi-bin/demo

STDIN               username=2%60n+d%2Bt+e+s%27t&fullname=Peter+Newton
```

# More environment variables

| Env variable | Meaning |
|---|---|
| HTTP_ACCEPT | A list of the MIME types that the client can accept |
| HTTP_REFERER | The URL of the document that the client points to before accessing the CGI program |
| HTTP_ | |
| REMOT | |
| REMOTE_HOST | The remote hostname o<br>quest |
| SERVER_NAME | The server's hostname |
| SERVER_PORT | The port number of the ho<br>is running |
| SERVER_SOFTWARE | The name and version of the server software |

# CGI programs and Perl

- CGI programs need to process input data from environment variables and STDIN, depending on the request method
  ⇝ preferably the input data would be accessible by the program in a uniform way
- CGI pr
  ⇝ pr
- CGI pr
  ⇝ preferably, there would be an easy way to produ

In Perl all this can be achieved with the use of the
`http://perldoc.perl.org/CGI.html`

# CGI.pm HTML shortcuts

- CGI.pm provides so-called HTML shortcuts that create HTML tags

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| a | address | applet | b | body | br | center | code |
| dd | div | dl | dt | em | font | form | |
| h1 | h2 | h3 | h4 | h5 | h6 | head | header |
| html | | | | | | | strong |
| sup | | | | | | | ul |



- HTML tags have attributes and contents

```
<p align="right">This is a paragraph</p>
```



- HTML shortcuts are given
  - HTML attributes in the form of a hash reference as the first argument
  - the contents as any subsequent arguments

```
p({-align=>right},"This␣is␣a␣paragraph")
```

# CGI.pm HTML shortcuts: Examples

```
Code:     print p();
Output:   <p />

Code:     print p('');
Output:   <p></p>

Code:     print p({-ali
Output:   <p align

Code:     print p({-class=>right_para,-id=>p1},"Text");
Output:   <p class="right_para" id="p1">Text</p>
```

# CGI.pm HTML shortcuts: Nesting vs Start/End

- Nested HTML tags using nested HTML shortcuts

```
Code:   print p(em("Emphasised")."␣Text"), "\n";
Output: <p><em>Emphasised</em> Text</p>
```



- Neste

```
use CGI qw(-ut
print start_
      "␣Text", end_p(), "\n";
Output: <p><em>Emphasised</em> Text</p>
```



The following `start_`*tag*/`end_`*tag* HTML shortcuts are generated automatically by CGI.pm:

```
start_html(), start_form(), start_multipart_form()
  end_html(),   end_form()    end_multipart_form()
```

All others need to be requested by adding `:`*tag* to the CGI.pm import list

# CGI.pm Forms

- HTML forms are created using `start_form` and `end_form`

```
print start_form({-method=>request_method,
                  -action=>uri});
form elements
print end_form;
```

- HTM

| | | |
|---|---|---|
| tex | | ld |
| fil | | st |
| popup_menu | optgroup | |
| image_button | checkbox | |
| radio_group | reset | |



  - optgroup creates an option group within a popup menu
    ⇝ optgroup occurs nested inside popup_menu
  - All other HTML shortcuts for HTML form elements will occur independently of each other within a form

# CGI.pm Forms: Examples

```
print textfield({-name=>'username',
                 -value=>'dave',
                 -size=>100,
                 -maxlength=>500});
```



- -nam
  and is th
- -val
- -size is the size of the text field in characters
- -maxlength is the maximum number of charac
  will accept





Output:

```
<input type="text" name="username"
       value="dave" size="100" maxlength="500" />
```

# CGI.pm Forms: Examples

```
print submit({-name=>'submit',
              -label=>'Click␣for␣response'});
```



- `-name` is an optional argument that allows to distinguish submit buttons from each other
- `-lab`
  show



Output:

```
<input type="submit" name="submit"
       value="Click␣for␣response" />
```

# CGI.pm Forms: Example

```
#!/usr/bin/perl

use CGI qw(-utf8 :all);

print header(-charset=>'utf-8'),
      start_html({-title=>'My␣HTML␣Form',
                  -author=>'u.hustadt@liverpool.ac.uk',


print startform;


print textfield({-name=>'username',
                 -value=>'dave',
                 -size=>100});
print br();
print textfield({-name=>'fullname',
                 -value=>'Please␣enter␣your␣name',
                 -size=>100});
print br();
print submit({-name=>'submit',
              -value=>'Click␣for␣response'});
print end_form, end_html;
```

# Making it work

For CGI programs to work on our systems you must proceed as follows:

1. Your home directory must be 'world executable'
2. You must have a directory
   `$HOME/public_html/cgi-bin/`
   You e and exec
3. You
   `$HOME/public_html/cgi-bin/`
   and must be executable by everyone
4. The CGI script can then be accessed using the URL
   `http://cgi.csc.liv.ac.uk/cgi-bin/cgiwrap/<user>/<script>`
   or `http://cgi.csc.liv.ac.uk/cgi-bin/cgiwrapd/<user>/<script>`

   where `<user>` is your user name
   and `<script>` is the filename of the script
   (`cgiwrapd` provides debugging output, but does not reveal all errors)

# Accessing and processing data

- Perl provides a hash %ENV that stores the information stored in environment variables
- Processing %ENV is done in the standard way for hashes



```
print "The request method used is ",


foreach $key (
  print "Th
}
```



Output:



```
The request method used is GET
The value of SCRIPT_NAME is /cgi-bin/cgiwrap/ullrich/demo
The value of SERVER_NAME is cgi.csc.liv.ac.uk
The value of SERVER_ADMIN is root@localhost
The value of HTTP_ACCEPT_ENCODING is gzip,deflate
The value of HTTP_CONNECTION is keep-alive
The value of REQUEST_METHOD is GET
```

# Accessing and processing data

- `CGI.pm` provides the `param` routine to access the input data of HTML forms
- For a sequence of key-value pairs

  *key1*=*value1*&*key2*=*value2*&*key3*=*value3*&...

  repres

  will ret

  *value1* *value2* *value3*

  while `param()` returns the list `'key1'`
- The values returned by `param` have alrea
- `param('key')` returns the empty string if *value* is empty
- `param('key')` returns `undef` if *key* is not among the key-value pairs of the request
- This does not depend on whether the request method is GET or POST

# Accessing and processing data

- `CGI.pm` provides the `param` routine to access the input data of HTML forms



```
print "The value of username is ",
      param('username'), br(), br(), "\n";
print "The v
```



```
foreach $key (
   print "The value of $key is ",param($key), br(), "\n";
}
```

Output:



```
The value of username is dave
The value of fullname is David Davidson

The value of submit is Click for response
The value of username is dave
The value of fullname is David Davidson
```

# Accessing and processing data: UFT-8

- The pragma -utf8 in

```
use CGI qw(-utf8 :all);
```

makes CGI.pm treat all param() values as UTF-8 strings

- Alternatively, specific param() values can be decoded using the decode subro

```
use Encod
my $fulln
```

- With

```
binmode(STDOUT,":encoding(utf-8)");
print header(-charset=>'utf-8');
```

we ensure that the web page we produce is sent to the brower using UTF-8 encoding

# Accessing and processing data: Security

- Do not trust any data accessed via `param` (beware code injection)
  Example:

```
print "The value of username is ",param('username'),"\n";
```

  together with input

```
<script>window.location="http://malware_site/"</script>
```

  for us

- Check

```
if (param                    ~
  print "Not a valid user name"
} else {
  print "The value of username is ",param('usern
}
```

  or sanitise the input using the CGI.pm routine `escapeHTML`:

```
  print "The value of username is ",
        escapeHTML(param('username')),"\n";
```

  or even better, do both

# CGI.pm Scripts: Example (Part 1)

```
use CGI qw(-utf-8 :all *table);
binmode(STDOUT, ":encoding(utf-8)");

print header(-charset=>'utf-8'), "\n",
      start_html({-title=>'Form Processing',
                  -author=>'u.hustadt@liverpool.ac.uk'});

if (!defined(p
  # This branch is e
  print start
  print textf

  print br(), "\n";
  print textfield({-name=>'fullname',
                   -value=>'(lease enter your name'
                   -size=>100}), "\n";
  print br(), "\n";
  print submit({-name=>'submit',
                -value=>'Click for response'}), "\n";
  print end_form;
} else {
  # This branch is executed if the client request is generated
  # by the form
```

# CGI.pm Scripts: Example (Part 2)

```
  # (We are in the else-branch now)

  print start_table({-border=>1});
  print caption("Inputs");
  foreach $key (param()) {
    print Tr(td('PARAM'),td($key),td(escapeHTML(param($key))));
   }
  foreach $ke
    print Tr
  }
  print end_t
}
print end_html;
```

# CGI.pm Scripts: Example (Part 3)

Page produced on the first visit

# Revision

Read

- Chapter 11: Perl Modules

of

R. L. Sch

Learning Perl.

O'Reilly, 2011



- `http://perldoc.perl.org/CGI.html`