More Server-Side Perl

CS174

Chris Pollett

Nov 1, 2006.

Outline

- CGI.pm shortcuts
- Headers and Footers
- Handling Form data
- flock
- Handling Cookies

CGI.pm

- CGI.pm is a Perl module with a bunch of functions and classes defined in it to make server side programming easy.
- To use CGI.pm, you add the following line at the start of your Perl program:

  use CGI “:standard”;

- The string “:standard” says that we don’t want to load the whole module and that we don’t want to mess with the OO way to interact with CGI.pm.

Common CGI.pm functions

- CGI.pm adds several functions which correspond to HTML tags.
- For instance the function br corresponds to the break tag. The line:

  print br;

  outputs
- This kind of shortcut command may take parameters. So:

  print h1(“This is the real stuff”);

  outputs

  <h1>This is the real stuff</h1>

More CGI.pm functions

- HTML tags usually have both contents and attributes. Attributes can be provided to CGI.pm shortcuts as name value pairs using following kind of syntax:

```
print textarea(-name => “Description”,
               -rows => “2”,
               -cols => “3”);
```

  This outputs:

```
<textarea name=“Description” rows=“2” cols=“3”></textarea>
```

- You can also have functions which take attributes and contents:

```
print a({-href => “fruit.html”}, “Press here for fruit descriptions”);
```

Even More CGI.pm functions

- In general, shortcut functions can save considerably on outputting pages and forms.
- For instance, to output lists one can use the following syntax:

  print ol(li({-type =>"square"}, ["milk","bread", "cheese"]));

  To get:

  <li type="square">milk

  <li type="square">bread

  <li type="square">cheese

- A similar strategy works unordered lists. There are also command like this for tables (see book).

Radio Groups

- To output radio groups one can do:

```
print radio_group( -name=> ‘colors’,
    -values => [‘blue’, ‘green’, ‘yellow’, ‘red’],
    -default => ‘blue’
);
```

- This outputs:

```
<input type=“radio” name=“colors” values=“blue”
    checked=“checked” /> blue
<input type=“radio” name=“colors” values=“green” /> green
<input type=“radio” name=“colors” values=“yellow” /> yellow
<input type=“radio” name=“colors” values=“red” /> red
```

Headers and Footers

- Recall if we were not using CGI.pm we had to manually print out the Content-type header to get anything to appear over at the client.
- This can be easily accomplished using the single line:

  print header;

  This outputs

  Content-Type: text/html; charset=ISO-8859-1

  ---blank line ---

- It is also easy to output the head of an HTML document with:

  print start_html("Title of doc");

  This outputs

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US" xml:lang="en-
    US"><head><title>title</title>
</head><body>
```

- To close the document we could use end_html to output </body></html>

More Complicated Headers

- The header function can take parameters which can output more complicated headers:

```
print header(-type=>'image/gif',
  -status=>'402 Payment Required',
  -expires=>'+3d',#how long browser should #cache
  this cgi response
  -cookie=>$my_cookie, #what cookie to set
  -charset=>'UTF-7',
  -attachment=>'foo.gif',
  -Cost=>'$0.02' #random other header
);
```

- Similarly, we can add parameters to start_html: -title, -author, -style, -dtd

Handling Form Data

- Form data sent to your script is easily available using the param function.
- For instance,

  my $name = param(“formvar”);

  Stores the value sent by the form for the variable formvar into the $name perl variable.

Flock

- Consider the following scenario which might happen if we were trying to maintain a counter:

  #Have a file with a count currently at 27

  1. CGI1 started by Client1 reads the file containing the count into its variable $counter (gets 27)
  2. CGI2 started by Client2 reads the file containing the count into its variable $counter (gets 27)
  3. CGI1 does $counter++ (so 28) writes file back out
  4. CGI2 does $counter++ (so 28) writes file back out.
- So we had two hits but it was only counted as one hit.
- To prevent this we want to use locking.
- This can be done using the Perl flock function.

Flock Example

```
use Fcntl qw(:DEFAULT :flock);
open(TAX_DATA, "+<taxdata") or
   die "TAX_DATA could not be opened";
 flock(TAX_DATA, LOCK_EX) or
    die "TAX_DATA could not be locked";
chomp($tax = <TAX_DATA>); #now update $tax
seek(TAX_DATA, 0, 0) or
   die "TAX_DATA could not be rewound";
print TAX_DATA $tax or
  die "TAX_DATA could not be rewritten";
close TAX_DATA;
```

CGI.pm and Cookies

- We saw how we could set a cookie using the header command.
- We can create a cookie using:

```
$mycookie = cookie (
        -name => a_name,
    -value => a_value,
    -expires => a_time
);
```

- We can also use cookie to get the value returned froma client for a cookie:

```
$age = cookie("age");
```