Lecture 4

Some General Guidelines
Using CGI.pm
A Simple Example Using CGI.pm

Assignment Discussion

Complete code below to generate passwords between 6-10 characters with alphabet A-Z (ASCII 65 - 90)

```perl
srand; # initializes random number generator
$len = int(rand(10)) + 6; # rand(n) generates a floating point between 0 - n
$passwd = "";
for ($i=0; $i<$len; $i++) {
    $passwd .= chr( rand(26) + 65 ); # chr returns char given ASCII value
}
print "password generated is $passwd";
```

CS 172 Introduction to Internet Technology
Localizing Variables with my

- In Perl all variables have global scope by default.
- If you need a variable within a particular block only, such as a subroutine, declare it with my.
- This makes it memory efficient as the memory occupied by this variable will be reclaimed at the end of the block.

Why use the strict pragma

- Misspellings are an all-too common source of errors in Perl programs.
- It is easy to misspell a variable name in a language that does not require you to declare variables before they are used.

```perl
use strict;
my @temp = <FH>
#... some code here
while (@tmp) {
  ...
  Misspelt but not detected by Perl
...}
```

A perl program that uses strict pragma must declare variables via my.
Using Strict pragma ...

#!/usr/local/bin/perl
require "parseform2.lib";
use strict;
print "Content-type: text/html\n\n";
my $username;
my $record;
my $uname;
my $ue_string;
my $email;
my $number;
my $DOB;
my $day;
my $month;
my $year;
my @records;

%formdata = &Parse_Form;
$uname = $formdata{username};

All variables, hash and arrays have local scope now - cannot be accessed in other modules.

Hence the Parse_Form subroutine must be changed to return a hash.

What changes are needed?

sub Parse_Form {
  if ($ENV{'REQUEST_METHOD'} eq 'POST') {
    read (STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
    @pairs = split(/&/, $buffer);
    if ($ENV{'QUERY_STRING'}) {
      @getpairs =split(/&/, $ENV{'QUERY_STRING'});
      push(@pairs,@getpairs);
    }
  } elsif ($ENV{'REQUEST_METHOD'} eq 'GET') {
    @pairs = split(/&/, $ENV{'QUERY_STRING'});
  } else { print "Content-type: text/html\n\n";
    print "<p>Use Post or Get";
  }
  foreach $pair (@pairs) {
    ($key, $value) = split (/=/, $pair);
    $key =~ tr/+/ /;
    $key =~ s/%(\[a-fA-F0-9\]\[a-fA-F0-9\])//pack("C", hex($1))/eg;
    $value =~ tr/+/ /;
    $value =~ s/%(\[a-fA-F0-9\]\[a-fA-F0-9\])//pack("C", hex($1))/eg;
    $value =~s/<!--(.|\n)*-->/g;
    if ($formdata{$key}) {
      $formdata{$key} .= ", $value"
    } else {
      $formdata{$key} = $value;
    }
  }
}
Changes needed ...

sub Parse_Form {
  my %formdata;
  ...
  ...

  return %formdata;  #returning the local hash
  %formdata

} 1;

Using CGI.pm

• It has become the standard tool for creating CGI scripts in Perl.
• It provided a simple interface for most common CGI tasks.
• It makes including HTML inside a Perl script more natural and easier to validate.
• The Current Version is 2.76
• The CGI.pm covers the following parts:
  • Handling Input
  • Generating Output
  • Handling Errors
### Code using HTML tags

```perl
#!/usr/local/bin/perl
require "subparseform2.lib";
use strict;
my %formdata = &Parse_Form;
my $name = $formdata{"name"};
print "Content-type: text/html\n
",
"<HTML><HEAD><TITLE>Using HTML Directly</TITLE></HEAD><BODY>",
"$name",
"</BODY></HTML>"
```

Many Web developers start out writing CGI scripts that contain much static code using HTML.

Over the lifetime of a CGI application the HTML often changes much requiring much maintenance effort.

---

### Code using CGI.pm

Object Oriented Version

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td><code>#!/usr/local/bin/perl -w</code></td>
</tr>
<tr>
<td>2:</td>
<td><code>use strict;</code></td>
</tr>
<tr>
<td>3:</td>
<td><code>use CGI qw( :standard);</code></td>
</tr>
<tr>
<td>4:</td>
<td><code>my $q = new CGI;</code></td>
</tr>
<tr>
<td>5:</td>
<td><code>my $name = $q-&gt;param(&quot;name&quot;);</code></td>
</tr>
<tr>
<td>6:</td>
<td><code>print $q-&gt;header(&quot;text/html&quot;),</code></td>
</tr>
<tr>
<td>7:</td>
<td><code>$q-&gt;start_html(&quot;Welcome&quot;),</code></td>
</tr>
<tr>
<td>8:</td>
<td><code>$q-&gt;p(&quot;Hi $name!&quot;)</code>,</td>
</tr>
<tr>
<td>9:</td>
<td><code>$q-&gt;end_html;</code></td>
</tr>
</tbody>
</table>

- Line 4: creates a CGI object.
- Param() allows you to access parameters submitted to the CGI script.
- Subsequently all CGI functions are preceded with a $q->.
Code using CGI.pm
Standard Version

1: #!/usr/local/bin/perl -w
2: use strict;
3: use CGI qw(:standard);
4: my $name = param("name");
5: print header("text/html"),
6: start_html("Welcome"),
   p("Hi $name!"),
7: end_html;

• Line 3 requests
  CGI.pm to export the
  standard functions.
  Which are then used
directly.
• CGI.pm provides
  several predefined
  group of functions like
  :standard.

Handling Input with CGI.pm

• Accessing parameters is done with param, which allows you
  to access the parameters submitted via a GET or POST
  request.
• CGI.pm provides many methods to get information about
  your environment. Some of these are similar to obtaining
  them through the Perl’s %ENV hash.

my $query = new CGI;
my $path = $query->path_info;       # using CGI.pm method
my $path = $ENV{PATH_INFO};         # using %ENV hash

CS 172 Introduction to Internet Technology
Getting the names Parameters

If your CGI program was invoked with the parameter list as in:

   http://goanna.cs.rmit.edu.au/~cha/cgi-bin/test.cgi?
   name=john&drink=orange&topping=cheese&topping=garlic

The param() method will return the parameter names as a list which in this case is name drink cheese.

To get the value(s) of a named parameter such as drink or topping you pass the param method a single argument as in

   $drink = $query->param('drink');
   @toppings = $query->param('topping');  #use an array for multivalues

Generating Output with CGI.pm

• CGI.pm provides a very elegant solution for outputting both headers and HTML with Perl.

• It allows you to embed HTML into your code.

• Every HTML element can be generated via a corresponding method in CGI.pm.

• The CGI.pm automatically generate the closing tags when needed.

   print “<H2>Current Time</H2>”;
   print $query->h2("Current Time");
Creating Forms

- The CGI.pm provides a full set of methods for creating Web fill-out forms which return strings containing the HTML code for creating the requested form elements.
- You are responsible for printing these.
- You are free to add formatting tags around the form elements.

```perl
$query = new CGI;
print $query->header("text/html");
print $query->startform("GET","http://.../cgibin/supply.cgi");
print "<P>Type of Gear Box : ",
$query->popup_menu(-name=>'Type',-values=>\@Parts);
print "Quantity ? ",$query->textfield('Quantity');
print "$<p>",$query->submit;
print $query->endform;
print $query->end_html;
```

Standard HTML elements

Creating a Text Field

```perl
textfield('NAME', 'INITIAL-VALUE', 'WINDOW-SIZE', 'MAX-CHARACTERS')
```

```perl
print "Quantity ? ",$query->textfield('Quantity');
```

Creating a Textarea Field

```perl
textarea('NAME', 'INITIAL-VALUE', 'ROWS', 'COLUMNS')
```

Creating a Password Field

```perl
password_field('NAME', 'INITIAL-VALUE', 'WINDOW-SIZE', 'MAX-CHARACTERS')
```
Creating a Group of Related Checkboxes

```php
print $query->checkbox_group(
    -name=>'group_name',
    -values=>['eenie','meenie','minie','moe'],
    -default=>['eenie','moe'],
    -linebreak=>'true'
);
```

Creating a Pop-up Menu

The following method creates a selection menu:

```php
popup_menu('NAME','OPTION-NAMES','SELECTED-OPTION');
```

```php
print $query->popup_menu(-name=>'menu_name',
    -values=>['eenie','meenie','minie','moe'],-default=> 'moe');
```

The second argument can be a reference to an array argument as in the following:

```php
print "<P>Type of Gear Box : ",
$query->popup_menu(-name=>'Type', -values => 
@$Parts );
```
A simple example using CGI.pm

- Let's create a simple example using CGI.pm, where reconditioned vehicle parts availability can be checked.
- It is assumed that the users have already registered.
- The currently available parts (and quantity) are dynamic in nature and is generated by a CGI program based on the contents of the file parts.txt.

File parts.txt

Honda Accord [89 model] Gear Box : (34)
Toyota Camri [94 model] Gear Box : (20)
BMW [89 model] Gear Box : (24)
Peugot 404 [92 model] Gear Box : (12)
A simple Design

Client

Connect to URL

HTML (Form)

Submit form

CGI generated HTML (Form)

Submit form

CGI generated HTML

Server

GetID.html

check.cgi

supply.cgi

Initial HTML form

```html
<html>
  <head>
    <title>Reconditioned Vehicle Parts System</title>
  </head>
  <body>
    <form action="http://goanna.cs.rmit.edu.au/~charles/cgi-bin/check.cgi" method="post">
      <p><b>Name:</b> <input type="text" name="name" value="" size="20" maxlength="40"></p>
      <p><b>Registration Number:</b> <input type="text" name="regno" value="" size="20" maxlength="40"></p>
      <p><input type="submit" name="order" value="Check Parts Available"></p>
    </form>
  </body>
</html>
```
CGI Script for Dynamic List Creation

```perl
#!/usr/local/bin/perl
use strict;
use CGI qw(:all);
my @Parts;
my $query;
open(PARTS,"<parts.txt"); @Parts = <PARTS>; close(PARTS);
$query = new CGI;
print $query ->header("text/html");
print "<H3><P>Welcome ", $query->param('name'), " to the Vehicle Parts System;
print "Please Select the Type and Quantity Required <H3>";
print "$query->startform("GET","http://www.cs.rmit.edu.au/~charles/cgi-bin/supply.cgi");
print "<P>Type of Gear Box : ",
print "$query->popup_menu(-name=>'Type','-values=>\@Parts);print "Quantity ? ",$query->textfield('Quantity');
print "$<P">","$query->submit;print $query->endform;
```
foreach $line(@Parts)
{
    if ($line eq $Type) {
        ($Discard, $Available) = split(/\(/,$line);
        ($Available,$Discard) = split(/\)/,$Available);
        if ($Quantity < $Available ) {
            print "<p> All $Quantity Items Can be supplied immediately";
        }
        else {
            $Balance = $Quantity - $Available;
            print "<p> We can only supply $Available now",
            "<p> Place an order for the remaining $Balance items";
        }
    } else {
        print "<p> Please Enter Numeric value for Quantity";
    }
}
print $query ->end_html;

Where do I get more Information about CGI.pm Library?

http://stein.cshl.org/WWW/software/CGI/cgi_docs.html
CS172 - Tutorial 4

1. Modify the program developed last week to incorporate:

   (a) the strict pragma
   (b) CGI.pm modules

   Modularize your code by breaking them into subroutines