CGI Scripting for Programmers: Introduction

Course Notes

Jon Warbrick
University of Cambridge Computing Service
jon.warbrick@ucs.cam.ac.uk

March 2003

EXAMPLE PROGRAMS

Some of these programs are based on, or adapted from, example programs by Lincon Stein (in 'The Official Guide to Programming with CGI.pm'), Scott Guelich, Shishir Gundavaram and Gunther Birznieks (in 'CGI Programming with Perl') and Rachel Coleman (University of Cambridge MISD).

simple.cgi

```
#!/usr/bin/perl -Tw
use strict;

my $now = localtime();

print "Content-type: text/plain\n";
print "\n";
print "Hello World\n";
print "\n";
print "It is now $now\n";
```

uri_escape()/uri_unescape()

```
sub uri_escape {
  my $text = shift;
  $text =~
   s/([^a-z0-9_.!~*'()-])/sprintf &quot;%%%02X&quot;, ord($1)/egi;
  return $text;
}

sub uri_unescape {
  my $text = shift;
  $text =~ tr/\+/ /;
  $text =~ s/%([a-f0-9][a-f0-9])/chr( hex( $1 ) )/egi;
  return $text;
}
```

simple-html.cgi

```
#!/usr/bin/perl -Tw
use strict;

my $now = localtime();

print "Content-type: text/html; charset=iso-8859-1\n";
print "\n";

print "<html>\n";

print "<head>\n";
print "<title>A first HTML CGI</title>\n";
print "</head>\n";

print "<body>\n";
print "<h1>Hello World</h1>\n";
print "<p>It is $now</p>\n";
print "</body>\n";

print "</html>\n";
```

escape_html()

```
sub escapeHTML {
  my $text = shift;
  $text =~ s/&/&amp;/g;
  $text =~ s/</&lt;/g;
  $text =~ s/>/&gt;/g;
  return $text;
}
```

simple-html2.cgi

```
#!/usr/bin/perl -Tw
use strict;

my $now = localtime();

print "Content-type: text/html; charset=iso-8859-1\n";
print "\n";

print "<html>\n";

print "<head>\n";
print "<title>A first HTML CGI</title>\n";
print "</head>\n";

print "<body>\n";
print "<h1>Hello World</h1>\n";
print "<p>It is ";
print escapeHTML($now);
print "</p>\n";
print "</body>\n";

print "</html>\n";

sub escapeHTML {
  my $text = shift;
  $text =~ s/&/&amp;/g;
  $text =~ s/</&lt;/g;
  $text =~ s/>/&gt;/g;
  return $text;
}
```

register.html

```
<html>

<head>
<title>Mailing list</title>
</head>

<body>
<h1>Mailing list signup</h1>
<p>Please fill in this form to be notified of
future updates</p>

<form action="reg.cgi" method="post">
<p>Name: <input type="text" name="name" /></p>
<p>Email: <input type="text" name="email" /></p>
<p><input type="submit" value="Submit Request" /></p>
</form>

</body>

</html>
```

register2.html

```
<html>

<head>
<title>Mailing list</title>
</head>

<body>
<h1>Mailing list signup</h1>
<p>Please fill in this form to be notified of
future updates</p>

<form action="reg.cgi" method="post">
<p>Name: <input type="text" name="name" /></p>
<p>Email: <input type="text" name="email" /></p>
<p><input type="submit" value="Submit Request" /></p>
</form>

</body>

</html>
```

form-controls.html

```
<html>

<head>
<title>Example form elements</title>
</head>

<body>

<h1>Example form elements</h1>

<form action="environment.cgi" method="post">

<p>

  Name: <input type="text" name="surname" value="Name" />
  <br />
  Password: <input type="password" name="pwd" value="foobar" />

</p>

<hr />

<p>

  <input type="radio" name="drink" value="tea" />Tea
  <input type="radio" name="drink" value="coffee"
                              checked="checked" />Coffee
  <br />
  <input type="checkbox" name="milk"   value="yes" />Milk
  <input type="checkbox" name="sugar" value="yes" />Sugar

</p>

<hr />

<p>

  <input type="submit" name="submit" value="Do Search" />
  <input type="reset"  name="why"    value="Defaults" />
  <input type="button" name="button" value="Click here" />

</p>
```

```
<hr />

<p>

  <input type="image" name="find" value="Finding"
         src="b1.png" alt="[FIND]" />

</p>

<hr />

<p>

  <input type="hidden" name="state" value="New York" />

</p>

<hr />

<p>

  <select name="contact">
    <option selected="selected">Webmaster</option>
    <option value="mailroom">Postmaster</option>
    <option>TimeLord</option>
  </select>

   

  <select name="contact" size="2">
    <option selected="selected">Webmaster</option>
    <option value="mailroom">Postmaster</option>
    <option>TimeLord</option>
  </select>
</p>

<hr />

<p>

  <textarea name="Comments" cols="40" rows="4">
Default text
Foo..
...Bar...
......Buz...
.........Boo...
  </textarea>

</p>

</form>

</body>

</html>
```

parse_form_data()

```
sub parse_form_data {
  my ($query, %form_data, $name, $value, $name_value,
      @name_value_pairs);
  @name_value_pairs = split(/&/,$ENV{QUERY_STRING} )
                                         if $ENV{QUERY_STRING};
  if ( $ENV{REQUEST_METHOD} and
       $ENV{REQUEST_METHOD} eq 'POST' and
       $ENV{CONTENT_LENGTH} ) {
```

```
    $query = "";
    if (read(STDIN, $query, $ENV{CONTENT_LENGTH}) ==
                                      $ENV{CONTENT_LENGTH}) {
      push @name_value_pairs, split(/&/,$query);
    }
  }
  foreach $name_value ( @name_value_pairs ) {
    ($name,$value) = split /=/, $name_value;
    $name = uri_unescape($name);
    $value = "" unless defined $value;
    $value = uri_unescape($value);
    $form_data{$name} = $value;
  }
  return %form_data;
}
```

clock.html

```
<html>
<head>
<title>A virtual clock</title>
</head>

<body>

<form action='clock.cgi'>
<p>Your name: <input type='text' name='name' /></p>
<p>Show:
<input type='checkbox' checked='checked' name='time' />time
<input type='checkbox' checked='checked' name='weekday' />weekday
<input type='checkbox' checked='checked' name='day' />day
<input type='checkbox' checked='checked' name='month' />month
<input type='checkbox' checked='checked' name='year' />year
</p>
<p>Time style
<input type='radio' name='type' value='12-hour' />12-hour
<input type='radio' name='type' value='24-hour'
                                          checked='checked' />24-hour
</p>
<p>
<input type='submit' name='show' value='Show' />
<input type='reset' value='Reset' />
</p>
</form>

</body>

</html>
```

clock.cgi

```
#!/usr/bin/perl -wT
use strict;

use POSIX 'strftime';

use vars '%query';

%query = parse_form_data();

print "Content-type: text/html; charset=iso-8859-1\n";
print "\n";
print "<html>\n";
print "<head>\n";
print "<title>A virtual clock</title>\n";
print "</head>\n";
```

```
print "<body>\n";
print_time();
print "</body>\n";
print "</html>\n";

sub print_time {

  my ($format, $current_time);

  $format = '';
  if ($query{time}) {
    if ($query{type} eq '12-hour') {
      $format = '%r ';
    }
    else {
      $format = '%T ';
    }
  }
  $format .= '%A, ' if $query{weekday};
  $format .= '%d '  if $query{day};
  $format .= '%B '  if $query{month};
  $format .= '%Y '  if $query{year};

  $current_time = strftime($format,localtime);

  if ($query{name}) {
      print "Welcome ";
      print escapeHTML($query{name});
      print "! ";
  }
  print "It is <b>";
  print escapeHTML($current_time);
  print "</b><hr />\n";

}


sub parse_form_data {
  my ($query, %form_data, $name, $value, $name_value,
      @name_value_pairs);
  @name_value_pairs = split(/&/,$ENV{QUERY_STRING} )
                                        if $ENV{QUERY_STRING};
  if ( $ENV{REQUEST_METHOD} and
       $ENV{REQUEST_METHOD} eq 'POST' and
       $ENV{CONTENT_LENGTH} ) {
    $query = "";
    if (read(STDIN, $query, $ENV{CONTENT_LENGTH}) ==
                                      $ENV{CONTENT_LENGTH}) {
      push @name_value_pairs, split(/&/,$query);
    }
  }
  foreach $name_value ( @name_value_pairs ) {
    ($name,$value) = split /=/, $name_value;
    $name = uri_unescape($name);
    $value = "" unless defined $value;
    $value = uri_unescape($value);
    $form_data{$name} = $value;
  }
  return %form_data;
}

sub escapeHTML {
  my $text = shift;
  $text =~ s/&/&amp;/g;
  $text =~ s/</&lt;/g;
  $text =~ s/>/&gt;/g;
  return $text;
}
```

```
sub uri_unescape {
  my $text = shift;
  $text =~ tr/\+/ /;
  $text =~ s/%([a-f0-9][a-f0-9])/chr( hex( $1 ) )/egi;
  return $text;
}
```

clock2.cgi

```
#!/usr/bin/perl -wT
use strict;

use POSIX 'strftime';

use vars '%query';

%query = parse_form_data();

print "Content-type: text/html; charset=iso-8859-1\n";
print "\n";
print "<html>\n";
print "<head>\n";
print "<title>A virtual clock</title>\n";
print "</head>\n";
print "<body>\n";
print_time() if %query;
print_form();
print "</body>\n";
print "</html>\n";

sub print_time {

  my ($format, $current_time);

  $format = '';
  if ($query{time}) {
    if ($query{type} eq '12-hour') {
      $format = '%r ';
    }
    else {
      $format = '%T ';
    }
  }
  $format .= '%A, ' if $query{weekday};
  $format .= '%d '  if $query{day};
  $format .= '%B '  if $query{month};
  $format .= '%Y '  if $query{year};

  $current_time = strftime($format,localtime);

  if ($query{name}) {
      print "Welcome ";
      print escapeHTML($query{name});
      print "!";
  }
  print "It is <b>";
  print escapeHTML($current_time);
  print "</b><hr />\n";

}

sub print_form {
  print "<form action=''>\n";
  print "<p>Your name: ";
  textbox ('name');
  print "</p>\n";
  print "<p>Show:\n";
```

```
  checkbox('time');
  checkbox('weekday');
  checkbox('day');
  checkbox('month');
  checkbox('year');
  print "</p>\n";
  print "<p>Time style\n";
  radio('type','12-hour');
  radio('type','24-hour');
  print "</p>\n";
  print "<p>\n";
  print "<input type='submit' name='show' value='Show' />\n";
  print "<input type='reset' value='Reset' />\n";
  print "</p>\n";
  print "</form>\n";
}

sub textbox {
  my ($name) = @_;
  $name = escapeHTML($name);
  print "<input type='text' name='$name' />\n";
}

sub checkbox {
  my ($name) = @_;
  $name = escapeHTML($name);
  print "<input type='checkbox' name='$name' />$name\n";
}

sub radio {
  my ($name,$value) = @_;
  $name = escapeHTML($name);
  $value = escapeHTML($value);
  print "<input type='radio' name='$name' value='$value' />$value\n";
}

sub parse_form_data {
  my ($query, %form_data, $name, $value, $name_value,
      @name_value_pairs);
  @name_value_pairs = split(/&/,$ENV{QUERY_STRING} )
                                        if $ENV{QUERY_STRING};
  if ( $ENV{REQUEST_METHOD} and
       $ENV{REQUEST_METHOD} eq 'POST' and
       $ENV{CONTENT_LENGTH} ) {
    $query = "";
    if (read(STDIN, $query, $ENV{CONTENT_LENGTH}) ==
                                      $ENV{CONTENT_LENGTH}) {
      push @name_value_pairs, split(/&/,$query);
    }
  }
  foreach $name_value ( @name_value_pairs ) {
    ($name,$value) = split /=/, $name_value;
    $name = uri_unescape($name);
    $value = "" unless defined $value;
    $value = uri_unescape($value);
    $form_data{$name} = $value;
  }
  return %form_data;
}

sub escapeHTML {
  my $text = shift;
  $text =~ s/&/&amp;/g;
  $text =~ s/</&lt;/g;
  $text =~ s/>/&gt;/g;
  return $text;
}

sub uri_unescape {
```

```
  my $text = shift;
  $text =~ tr/\+/ /;
  $text =~ s/%([a-f0-9][a-f0-9])/chr( hex( $1 ) )/egi;
  return $text;
}
```

clock3.cgi

```
#!/usr/bin/perl -wT
use strict;

use POSIX 'strftime';

use vars '%query';

%query = parse_form_data();

print "Content-type: text/html; charset=iso-8859-1\n";
print "\n";
print "<html>\n";
print "<head>\n";
print "<title>A virtual clock</title>\n";
print "</head>\n";
print "<body>\n";
print_time() if %query;
print_form();
print "</body>\n";
print "</html>\n";

sub print_time {

  my ($format, $current_time);

  $format = '';
  if ($query{time}) {
    if ($query{type} eq '12-hour') {
      $format = '%r ';
    }
    else {
      $format = '%T ';
    }
  }
  $format .= '%A, ' if $query{weekday};
  $format .= '%d '  if $query{day};
  $format .= '%B '  if $query{month};
  $format .= '%Y '  if $query{year};

  $current_time = strftime($format,localtime);

  if ($query{name}) {
      print "Welcome ";
      print escapeHTML($query{name});
      print "! ";
  }
  print "It is <b>";
  print escapeHTML($current_time);
  print "</b><hr />\n";

}

sub print_form {
  print "<form action=''>\n";
  print "<p>Your name: ";
  textbox ('name');
  print "</p>\n";
  print "<p>Show:\n";
  checkbox('time');
```

```
  checkbox('weekday');
  checkbox('day');
  checkbox('month');
  checkbox('year');
  print "</p>\n";
  print "<p>Time style\n";
  radio('type','12-hour');
  radio('type','24-hour');
  print "</p>\n";
  print "<p>\n";
  print "<input type='submit' name='show' value='Show' />\n";
  print "<input type='reset' value='Reset' />\n";
  print "</p>\n";
  print "</form>\n";
}

sub textbox {
  my ($name) = @_;
  $name = escapeHTML($name);
  print "<input type='text' name='$name'";
  if ($query{$name}) {
    print " value='$query{$name}'\n";
  }
  print " />\n";
}

sub checkbox {
  my ($name) = @_;
  $name = escapeHTML($name);
  print "<input type='checkbox' name='$name'";
  if ($query{$name}) {
    print " checked='checked'";
  }
  print " />$name\n";
}

sub radio {
  my ($name,$value) = @_;
  $name = escapeHTML($name);
  $value = escapeHTML($value);
  print "<input type='radio' name='$name' value='$value'";
  if ($query{$name} eq $value) {
    print " checked='checked'";
  }
  print " />$value\n";
}

sub parse_form_data {
  my ($query, %form_data, $name, $value, $name_value,
      @name_value_pairs);
  @name_value_pairs = split(/&/,$ENV{QUERY_STRING} )
                                        if $ENV{QUERY_STRING};
  if ( $ENV{REQUEST_METHOD} and
       $ENV{REQUEST_METHOD} eq 'POST' and
       $ENV{CONTENT_LENGTH} ) {
    $query = "";
    if (read(STDIN, $query, $ENV{CONTENT_LENGTH}) ==
                                      $ENV{CONTENT_LENGTH}) {
      push @name_value_pairs, split(/&/,$query);
    }
  }
  foreach $name_value ( @name_value_pairs ) {
    ($name,$value) = split /=/, $name_value;
    $name = uri_unescape($name);
    $value = "" unless defined $value;
    $value = uri_unescape($value);
    $form_data{$name} = $value;
  }
  return %form_data;
```

```
}

sub escapeHTML {
  my $text = shift;
  $text =~ s/&/&amp;/g;
  $text =~ s/</&lt;/g;
  $text =~ s/>/&gt;/g;
  return $text;
}

sub uri_unescape {
  my $text = shift;
  $text =~ tr/\+/ /;
  $text =~ s/%([a-f0-9][a-f0-9])/chr( hex( $1 ) )/egi;
  return $text;
}
```

random.cgi

```
#!/usr/bin/perl -Tw
use strict;

my ($docroot, $pict_dir, @pictures, $num_pictures,
    $lucky_one, $buffer);

$docroot  = "/var/www/html";
$pict_dir = "cgi-course-examples/pictures";

chdir "$docroot/$pict_dir"
            or die "Failed to chdir to picture directory: $!";
@pictures = glob('*.png');
$num_pictures = $#pictures;
$lucky_one = $pictures[rand($num_pictures-1)];
die "Failed to find a picture" unless $lucky_one;

print "Content-type: image/png\n";
print "\n";

binmode STDOUT;
open (IMAGE, $lucky_one)
  or die "Failed to open image $lucky_one: $!";
while (read(IMAGE, $buffer, 4096)) {
  print $buffer;
}

close IMAGE;
```

random2.cgi

```
#!/usr/bin/perl -Tw
use strict;
my ($docroot, $pict_dir, @pictures, $num_pictures,
    $lucky_one, $buffer);
$docroot  = "/var/www/html";
$pict_dir = "cgi-course-examples/pictures";

chdir "$docroot/$pict_dir"
            or die "Failed to chdir to picture directory: $!";

@pictures = glob('*.png');
$num_pictures = $#pictures;

$lucky_one = $pictures[rand($num_pictures-1)];

die "Failed to find a picture" unless $lucky_one;
```

```
print "Location: /$pict_dir/$lucky_one\n”;
print “\n";
```

random2a.cgi

```
#!/usr/bin/perl -Tw
use strict;
my ($docroot, $pict_dir, @pictures, $num_pictures,
    $lucky_one, $buffer);
$docroot  = "/var/www/html";
$pict_dir = "cgi-course-examples/pictures";

chdir "$docroot/$pict_dir"
            or die "Failed to chdir to picture directory: $!";

@pictures = glob('*.png');
$num_pictures = $#pictures;

$lucky_one = $pictures[rand($num_pictures-1)];

die "Failed to find a picture" unless $lucky_one;

print "Location: http://www.example.com/$pict_dir/$lucky_one\n”;
print “\n";
```

errors.cgi

```
#!/usr/bin/perl -Tw
use strict;

my ($file, $buffer);
$file = '/var/www/msg.txt';
if ((localtime(time))[1] % 2 == 0) {
  error (403, "Forbidden",
         "You may not access this document at the moment");
}
elsif (!-r $file) {
  error(404, "Not found",
        "The document requested was not found");
}
else {
  unless (open (TXT, $file)) {
    error (500, "Internal Server Error",
           "An Internal server error occurred");
  }
  else {
    print "Content-type: text/plain\n";
    print "\n";
    while (read(TXT, $buffer, 4096)) {
      print $buffer;
    }
    close TXT;
  }
}

sub error {
  my ($code,$msg,$text) = @_;
  print "Status: $code $msg\n";
  print "Content-type: text/html; charset=iso-8859-1\n";
  print "\n";
  print "<html><head><title>$msg</title></head>\n";
  print "<body><h1>$msg</h1>\n";
  print "<p>$text</p></body></html>\n";
}
```

simple-html3.cgi

```
#!/usr/bin/perl -Tw
use strict;

my $now = localtime();

print "Content-type: text/html; charset=iso-8859-1\n";
print "Cache-control: max-age=30\n";
print "\n";

print "<html>\n";

print "<head>\n";
print "<title>A first HTML CGI</title>\n";
print "</head>\n";

print "<body>\n";
print "<h1>Hello World</h1>\n";
print "<p>It is ";
print escapeHTML($now);
print "</p>\n";
print "</body>\n";

print "</html>\n";

sub escapeHTML {
  my $text = $_[0];
  $text =~ s/&/&amp;/g;
  $text =~ s/</&lt;/g;
  $text =~ s/>/&gt;/g;
  return $text;
}
```

bottomless.cgi

```
#!/usr/bin/perl -Tw
use strict;

print "Content-type: text/html; charset=iso-8859-1\n";
print "\n";

print "<html>\n";

print "<head>\n";
print "<title>A Bottomless document tree</title>\n";
print "<meta name='robots' content='index,nofollow' />\n";
print "</head>\n";

print "<body>\n";
print "<h1>A Bottomless document tree</h1>\n";
print "<p>Here we have a <a href='tar/pit.html'>relative\n";
print "link</a>.</p>\n";
print "</body>\n";

print "</html>\n";
```

cgi.cgi

```
#!/usr/bin/perl -Tw
use strict;

use CGI;

my $q = new CGI;
```

```
print
  $q->header,
  $q->start_html (-title=>"Great rings of power"),
  $q->center(
    $q->h1("Ring allocation"),
    $q->p("Allocation of the Great Rings of power"),
    $q->table({border=>1},
      $q->Tr({align=>"center"},
        [ $q->th( [ 'Elves', 'Dwarf Lords', 'Mortal Men' ] ),
          $q->td( [ '3',     '7',           '9'          ] )
        ]
      )
    )
  ),
  $q->end_html;
```

clock-cgi.cgi

```
#!/usr/bin/perl -wT
use strict;

use POSIX 'strftime';
use CGI;

my $q = new CGI;

print $q->header,
    $q->start_html (-title=>"A virtual clock");

print_time() if $q->param();
print_form();

print $q->end_html;

sub print_time {

  my ($format, $current_time);

  $format = '';
  $format = ($q->param('type') eq '12-hour') ? '%r ' : '%T '
                                                  if $q->param('time');
  $format .= '%A, ' if $q->param('weekday');
  $format .= '%d '  if $q->param('day');
  $format .= '%B '  if $q->param('month');
  $format .= '%Y '  if $q->param('year');

  $current_time = strftime($format,localtime);

  if ($q->param('name')) {
      print "Welcome ";
      print $q->escapeHTML($q->param('name'));
      print "! ";
  }
  print "It is <b>";
  print $q->escapeHTML($current_time);
  print "</b><hr />\n";

}

sub print_form {
  print $q->start_form,
        $q->p(
          "Your name: ",
          $q->textfield(-name=>'name'),
        ),
        $q->p(
          "Show:",
```

```
          $q->checkbox(-name=>'time',     -checked=>1),
          $q->checkbox(-name=>'weekday', -checked=>1),
          $q->checkbox(-name=>'day',      -checked=>1),
          $q->checkbox(-name=>'month',    -checked=>1),
          $q->checkbox(-name=>'year',     -checked=>1),
        ),
        $q->p(
          "Time style",
          $q->radio_group(-name=>'type', -values=>['12-hour','24-hour']),
        ),
        $q->p(
          $q->submit(-name=>'Show'),
          $q->reset(-name=>'Reset'),
        ),
     $q->end_form;
}
```

fatal.cgi

```
#!/usr/bin/perl -Tw
use strict;

use CGI::Carp qw(fatalsToBrowser);

my $now = localtome();

print "Content-type: text/plain\n";
print "\n";
print "Hello World\n";
print "\n";
print "It is now $now\n";
```

lotr.cgi

```
#!/usr/bin/perl -Tw
use strict;

use CGI;
use DBI;

use vars '$q', '$dbh';

$q = CGI->new;

print $q->header,
      $q->start_html (-title=>"The characters");

my %attr = ( RaiseError => 1,
             PrintError => 0,
             AutoCommit => 1,
           );
my $dbh = DBI->connect("DBI:SQLite:dbname=lotr",
               "user", "pwd", \%attr);

do_list() if $q->param;
do_form();

$dbh->disconnect;
print $q->end_html;

sub do_list {

    my $race = $q->param('race');
    my $select = '';
    $select = 'AND race.id = ' . $dbh->quote($race)
                                        if ($race =~ /^\d$/);
```

```
    my $sth = $dbh->prepare ("SELECT characters.name, race.name
                              FROM characters, race
                              WHERE characters.race = race.id
                              $select
                              ORDER BY characters.name");
    $sth->execute;
    my $results = $sth->fetchall_arrayref;
    print $q->center(
            $q->h1("Characters"),
            $q->table({border=>1},
              $q->Tr({align=>"center"},
                [ $q->th( [ 'Name', 'Race' ] ),
                  map { $q->td($_) } @$results
                ]
              )
            )
          );

}

sub do_form {

    my $sth = $dbh->prepare ("SELECT name, id
                              FROM race
                              ORDER BY name");
    $sth->execute;

    my @values = ('*');
    my %labels = ('*' => 'All');
    while ( my ($name, $race) = $sth->fetchrow_array) {
        push @values,$race;
        $labels{$race}=$name;
    }

    print $q->center(
              $q->start_form,
              $q->p(
                "Chose a Middle Earth race: ",
                $q->br,
                $q->popup_menu(-name=>'race',
                               -values=>\@values,
                               -labels=>\%labels),
                $q->submit,
              ),
              $q->end_form,
          );

}
```

loan.cgi

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

# This program from 'The Official Guide to Programming with
# CGI.pm', Lincon Stein, 1998

# script: loan.cgi
use CGI qw/:standard :html3/;

# this defines the contents of the fill out forms
# on each page.
@PAGES = ('Personal
Information','References','Assets','Review','Confirmation');
%FIELDS = ('Personal Information' =>
['Name','Address','Telephone','Fax'],
```

```
        'References'           => ['Personal Reference 1','Personal
Reference 2'],
        'Assets'               => ['Savings Account','Home','Car']
        );
# accumulate the field names into %ALL_FIELDS;
foreach (values %FIELDS) {
    grep($ALL_FIELDS{$_}++,@$_);
}


# figure out what page we're on and where we're heading.
$current_page = calculate_page(param('page'),param('go'));
$page_name = $PAGES[$current_page];

print_header();
print_form($current_page)         if $FIELDS{$page_name};
print_review($current_page)       if $page_name eq 'Review';
print_confirmation($current_page) if $page_name eq 'Confirmation';
print end_html;

# CALCULATE THE CURRENT PAGE
sub calculate_page {
    my ($prev,$dir) = @_;
    return 0 if $prev eq ''; # start with first page
    return $prev + 1 if $dir eq 'Submit Application';
    return $prev + 1 if $dir eq 'Next Page';
    return $prev - 1 if $dir eq 'Previous Page';
}

# PRINT HTTP AND HTML HEADERS
sub print_header {
    print header,
    start_html("Your Friendly Family Loan Center");
}

# PRINT ONE OF THE QUESTIONNAIRE PAGES
sub print_form {
    my $current_page = shift;
    print "Please fill out the form completely and accurately.",
       start_form,
       hr;
    draw_form(@{$FIELDS{$page_name}});
    print hr;
    print submit(-name=>'go',-value=>'Previous Page')
      if $current_page > 0;
    print submit(-name=>'go',-value=>'Next Page'),
       hidden(-name=>'page',-value=>$current_page,-override=>1),
       end_form;
}

# PRINT THE REVIEW PAGE
sub print_review {
    my $current_page = shift;
    print "Please review this information carefully before submitting it.
",
       start_form;
    my (@rows);
    foreach $page ('Personal Information','References','Assets') {
      push(@rows,th({-align=>left},em($page)));
      foreach $field (@{$FIELDS{$page}}) {
          push(@rows,
             TR(th({-align=>left},$field),
                td(param($field)))
             );
          print hidden(-name=>$field);
      }
    }
    print table({-border=>1},caption($page),@rows),
```

```
        hidden(-name=>'page',-value=>$current_page,-override=>1),
        submit(-name=>'go',-value=>'Previous Page'),
        submit(-name=>'go',-value=>'Submit Application'),
        end_form;
}

# PRINT THE CONFIRMATION PAGE
sub print_confirmation {
    print "Thank you. A loan officer will be contacting you shortly.",
       p,
       a({-href=>'../source.html'},'Code examples');
}


# CREATE A GENERIC QUESTIONNAIRE
sub draw_form {
    my (@fields) = @_;
    my (%fields);
    grep ($fields{$_}++,@fields);
    my (@hidden_fields) = grep(!$fields{$_},keys %ALL_FIELDS);
    my (@rows);
    foreach (@fields) {
     push(@rows,
          TR(th({-align=>left},$_),
           td(textfield(-name=>$_,-size=>50))
           )
          );
    }
    print table(@rows);

    foreach (@hidden_fields) {
        print hidden(-name=>$_);
    }
}
```

cookie.cgi

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

# This program from 'The Official Guide to Programming with
# CGI.pm', Lincon Stein, 1998

use CGI qw(:standard :html3);

# Some constants to use in our form.
@colors=qw/aqua black blue fuschia gray green lime maroon navy olive
    purple red silver teal white yellow/;
@sizes=("<default>",1..7);

# recover the "preferences" cookie.
%preferences = cookie('preferences');

# If the user wants to change the background color or her
# name, they will appear among our CGI parameters.
foreach ('text','background','name','size') {
    $preferences{$_} = param($_) || $preferences{$_};
}

# Set some defaults
$preferences{'background'} = $preferences{'background'} || 'silver';
$preferences{'text'} = $preferences{'text'} || 'black';

# Refresh the cookie so that it doesn't expire.
$the_cookie = cookie(-name=>'preferences',
                     -value=>\%preferences,
                     -path=>'/',
                     -expires=>'+30d');
```

```
print header(-cookie=>$the_cookie);

# Adjust the title to incorporate the user's name, if provided.
$title = $preferences{'name'} ?
    "Welcome back, $preferences{name}!" : "Customizable Page";

# Create the HTML page.  We use several of the HTML 3.2
# extended tags to control the background color and the
# font size.  It's safe to use these features because
# cookies don't work anywhere else anyway.
print start_html(-title=>$title,
                 -bgcolor=>$preferences{'background'},
                 -text=>$preferences{'text'}
            );

print basefont({-size=>$preferences{size}}) if $preferences{'size'} > 0;

print h1($title);

# Create the form
print hr,
    start_form,

    "Your first name: ",
    textfield(-name=>'name',
              -default=>$preferences{'name'},
              -size=>30),br,

    table(
       TR(
          td("Preferred"),
          td("Page color:"),
          td(popup_menu(-name=>'background',
                    -values=>\@colors,
                    -default=>$preferences{'background'})
           ),
          ),
       TR(
          td(''),
          td("Text color:"),
          td(popup_menu(-name=>'text',
                    -values=>\@colors,
                    -default=>$preferences{'text'})
           )
          ),
       TR(
          td(''),
          td("Font size:"),
          td(popup_menu(-name=>'size',
                    -values=>\@sizes,
                    -default=>$preferences{'size'})
           )
          )
       ),

    submit(-label=>'Set preferences'),
    end_form,
    end_html;
```

template.ttml

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>
<title>Congratulations!!</title>
```

```
</head>

<body>

<h1>Congratulations [% name FILTER html %]</h1>

<p>Congratulations [% name FILTER html %], we are pleased
to tell you that you have just been allocated
$[% value FILTER html %] in our prize draw. All you need
to do is contact us at our address below to claim your prize.
</p>

<p>
[% FOREACH line = address -%]
[% line FILTER html %]<br />
[% END -%]
</p>

</body>
</html>
```

template.cgi

```
#!/usr/bin/perl -wT
use strict;

use Template;
use CGI;

my $q = CGI->new;

my $data = { name => 'Jon Warbrick',
             value => "1,000,000",
             address => ['123, The Street', 'Anytown', 'Aynwhere',
                         'ZZ1 1ZZ']
           };

my $tt = Template->new or
      die "Failed to create new template: " . Template->error();

my $html;
$tt->process("template.ttml",$data,\$html) || die $tt->error();

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

sendmail.pl

```
#!/usr/bin/perl -Tw
use strict;

$ENV{PATH} = $ENV{BASH_ENV} = '';

my $from    = 'jw35@cam.ac.uk';
my $to      = 'jon.warbrick@ucs.cam.ac.uk';
my @message = ("From: $from",
               "To: $to",
               "Subject: A test message",
               "",
               "Hello World!");

open(SENDMAIL, "|/usr/sbin/sendmail -oi -t")
  or die "Failed to open sendmail: $!\n";

foreach my $line (@message) {
  print SENDMAIL "$line\n";
```

```
}

close SENDMAIL or warn $! ? "Error closing sendmail pipe: $!\n"
                          : "Error $? from sendmail pipe";
```

Net-SMTP.pl

```
#!/usr/bin/perl -Tw
use strict;

use Net::SMTP;

my $from    = 'jw35@cam.ac.uk';
my $to      = 'jon.warbrick@ucs.cam.ac.uk';
my @message = ("From: $from",
               "To: $to",
               "Subject: A test message",
               "",
               "Hello World!");
eval {
  my $smtp = Net::SMTP->new('ppsw.cam.ac.uk', Debug => 1)
                               or die "connect";
  $smtp->mail($from)           or die "mail";
  $smtp->to($to)               or die "to";
  $smtp->data()                or die "data";
  foreach my $line (@message) {
    $smtp->datasend("$line\n") or die "datasend";
  }
  $smtp->dataend()             or die "dataend";
  $smtp->quit()                or die "quit";
};
if ($@) {
  die "Message not sent: $@ failed\n";
}
```

upload.html

```
<html>
<head>
<title>Upload Example</title>
</head>

<body>
<h1>Upload Example</h1>

<p>Upload a file:</p>

<form method="post" action="upload.cgi"
      enctype="multipart/form-data">
<p>Save as: <input type="text" name="save_as" /></p>
<p><input type="file" name="upload" value="" size="60" /></p>
<p><input type="submit" name="submit" value="Upload File" /></p>
</form>

</body>
</html>
```

upload.cgi

```
#!/usr/bin/perl -Tw
use strict;

use CGI;

$CGI::DISABLE_UPLOADS = 0;
```

```
$CGI::POST_MAX        = 1024 * 1024;

use vars '$q';

$q = new CGI;

print $q->header,
      $q->start_html('File upload'),
      $q->h1('File upload');

print_results();

print $q->end_html;

sub print_results {

  my $length;
  my $file = $q->param('upload');
  if (!$file) {
    print "No file uploaded.";
    return;
  }
  print $q->p(
          $q->b('Save as:'),
          $q->escapeHTML($q->param('save_as')),
        ),
        $q->p(
          $q->b('Uploaded file name:'),
          $q->escapeHTML($file)
        ),
        $q->p(
          $q->b('File MIME type:'),
          $q->escapeHTML($q->uploadInfo($file)->{'Content-Type'})
     );
  my $fh = $q->upload('upload');
  while (<$fh>) {
    $length += length($_);
  }
  print $q->p(
          $q->b('File length:'),
          $length
        );
}
```

PROTOCOL REFERENCE

CGI Environment Variables

Extracted from `http://hoohoo.ncsa.uiuc.edu/cgi/interface.html`

The following environment variables are not request-specific and are set for all requests:

- `SERVER_SOFTWARE`

  The name and version of the information server software answering the request (and running the gateway). Format: name/version

- `SERVER_NAME`

  The server's hostname, DNS alias, or IP address as it would appear in self-referencing URLs.

- `GATEWAY_INTERFACE`

The revision of the CGI specification to which this server complies. Format: CGI/revision

---

The following environment variables are specific to the request being fulfilled by the gateway program:

- `SERVER_PROTOCOL`

  The name and revision of the information protocol this request came in with. Format: protocol/revision

- `SERVER_PORT`

  The port number to which the request was sent.

- `REQUEST_METHOD`

  The method with which the request was made. For HTTP, this is "GET", "HEAD", "POST", etc.

- `PATH_INFO`

  The extra path information, as given by the client. In other words, scripts can be accessed by their virtual pathname, followed by extra information at the end of this path. The extra information is sent as PATH_INFO. This information should be decoded by the server if it comes from a URL before it is passed to the CGI script.

- `PATH_TRANSLATED`

  The server provides a translated version of PATH_INFO, which takes the path and does any virtual-to-physical mapping to it.

- `SCRIPT_NAME`

  A virtual path to the script being executed, used for self-referencing URLs.

- `QUERY_STRING`

  The information which follows the ? in the URL which referenced this script. This is the query information. It should not be decoded in any fashion. This variable should always be set when there is query information, regardless of command line decoding.

- `REMOTE_HOST`

  The hostname making the request. If the server does not have this information, it should set REMOTE_ADDR and leave this unset.

- `REMOTE_ADDR`

  The IP address of the remote host making the request.

- `AUTH_TYPE`

  If the server supports user authentication, and the script is protects, this is the protocol-specific authentication method used to validate the user.

- `REMOTE_USER`

  If the server supports user authentication, and the script is protected, this is the username they have authenticated as.

- `REMOTE_IDENT`

If the HTTP server supports RFC 931 identification, then this variable will be set to the remote user name retrieved from the server. Usage of this variable should be limited to logging only.

- `CONTENT_TYPE`

  For queries which have attached information, such as HTTP POST and PUT, this is the content type of the data.

- `CONTENT_LENGTH`

  The length of the said content as given by the client.

In addition to these, the header lines received from the client, if any, are placed into the environment with the prefix `HTTP_` followed by the header name. Any - characters in the header name are changed to _ characters. The server may exclude any headers which it has already processed, such as Authorization, Content-type, and Content-length. If necessary, the server may choose to exclude any or all of these headers if including them would exceed any system environment limits.