

# CGI 3

---

## Recall

HTML Form syntax  
Bindings sent to CGI

## dumpenv.pl

Prints out environment vars  
Use to see QUERY\_STRING when invoked as a CGI

```
#!/usr/bin/perl -w
# Print out the values of all the environment variables
# in an HTML <ul>.
# Call from the shell or invoke as a CGI script.

#
# HTTP header section
#
print "content-type: text/html\r\n\r\n";

#
# HTML header
#
$header = <<EOT;
<html>
<head><title>DumpEnv</title></head>
<body bgcolor=white>
EOT

$trailer = <<EOT;
</body>
</html>
EOT

# Print an HTML <ul> for all the environment vars

print $header;
my(@keys) = sort(keys %ENV); ## extract the keys from ENV hash table
print "<ul>\n";
foreach $key (@keys) {
    print "<li><b>$key</b> = $ENV{$key}\n";
}
print "</ul>\n";
print $trailer;
```

## CGI.pm

It has lots of features, but for now we'll just use it to extract bindings. Modern Perl installations have CGI.pm installed already by default  
use CGI;

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

```
$param = $query->param('param-name');
```

Will be undef if there is no such binding.

Use the defined() operator to check it

## <<EOT; Perl Syntax

The following syntax puts the raw text into the \$string...

```
$string = <<EOT;
```

```
This here can be any old thing.
```

```
Including $variable interpolation.
```

```
*** yeee haaa ***
```

```
EOT
```

## moviesearch.pl Points

Uses CGI.pm to extract the value of the "target" binding

If the target string is present, opens the local file "movies.txt", searches it, and prints the results to a table.

If the target string is not present, generates a search form

## Form/action trick

Returns a form when invoked without a "target" binding

The form does not contain an action=url, so the submit will send the bindings back to the source of the form, which is moviesearch.pl itself.

Pro: In this way, the form and the script do not have to know each-others filename, and it's easier to keep the field name's consistent since it's all in the one file.

Con: It is more structured to maintain the form in its own HTML file that is separate from the perl script file, especially if the two are maintained by separate people.

# moviesearch.pl Code

```
#!/usr/bin/perl -w
# Implements a simple search on a local tab-delimited database
# and returns the results in an HTML table.
# When called with no arguments, returns a simple search form.
# A submit of the search form returns a table of all the
# records which contain the target search string.
# (this is a simplified version of the dbsearch.pl example)

use CGI;

##
## HTML Header
##
$header = <<EOT;
<html><head>
<title>Movie Search</title>
</head>
<body bgcolor=white>
<h1>Movie Search</h1>
EOT

$strailer = "</body></html>\n";

##
## Start Output
##
$| = 1;          ## set STDOUT to be unbuffered
print "Content-type: text/html\r\n\r\n";
print $header;
$query = new CGI; ## this allocates the CGI module state

my($target, $filename);

$filename = "movies.txt";

##
## Search Form
##
$searchForm = <<EOT;
<h2>$filename</h2>
<p>

<form method=get>
Search string:<input type=text name=target size=20><br>

<p>
<input type=submit name=search value="Submit Search">

</form>
EOT
```

```

## If the target is not defined, just give them the target form
if (!defined($query->param('target'))) {
    print $searchForm;
    print $trailer;
    exit(0);
}
else {
    $target = $query->param('target');

    print "<p>Search for '<b>$target</b>' in the database '<b>$filename</b>'\n";

    open(DATA, "$filename") || ReportError("Cannot open '$filename'");
    ## Security: do not pass a string from the client to open()

    ## Make a table out of all the rows which matched...
    print "<table border=1 width = 100%>\n";

    my($labels);
    $labels = <DATA>;          # grab the labels from the first line
    chop($labels);
    TableRow($labels, "th");

    my($count) = 0;
    while (defined($line = <DATA>)) {
        $target = quotemeta($target); ## insert \'s
        if ($line =~ /$target/i) {    # Case insensitive search for target
            chop($line);
            TableRow($line, "td");
            $count++;
        }
    }

    close(DATA);

    print"</table>\n";

    # print count with the pluralization correct
    if ($count == 1) { print "<p>1 matching record\n"; }
    else { print "<p>$count matching records\n"; }

    print "<hr>$searchForm\n";

    print $trailer;
    exit(0);
}

# Helper which prints out one row of a table.
# The text should be tab delimited, and the type
# should be "th" for table header, or "td" for
# a table row (the default).
sub TableRow {

```

```
my($text, $type) = @_;  
  
my(@fields, $elem);  
  
@fields = split(/\t/, $text, -1);  
  
print "<tr>\n";  
foreach $elem (@fields) {  
    print "<$type>$elem</$type>";  
}  
print "\n</tr>\n";  
}  
  
sub ReportError {  
    my($err) = @_;  
  
    print "<h1>Error</h1><p>$err\n</html>\n";  
    exit(0);  
}
```

