

CGI

Lecture 7

cs193i – Internet Technologies
Summer 2004
Stanford University

Administrative Stuff

- HW #2 due today
- HW #3 due August 2
- Midterm should be returned on Monday
- Final
 - Local SCPD students will need to come to campus

The Web Platform

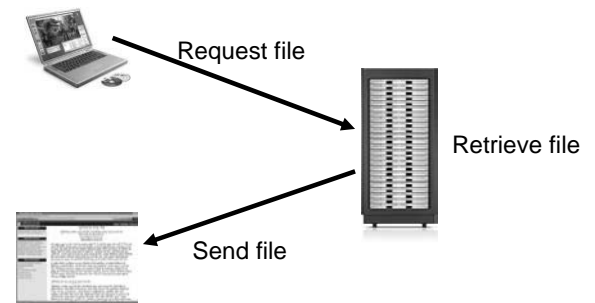
Google
amazon.com

VS

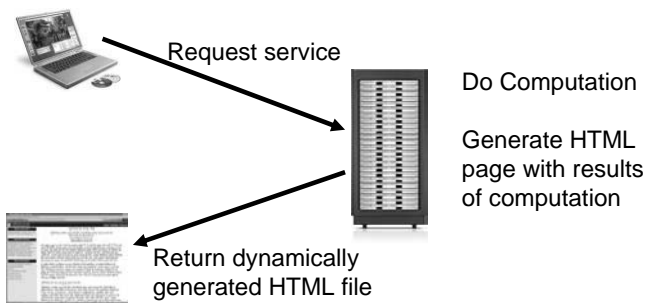


- Web Apps like Google, Amazon, etc... built on the Web "Platform" (as opposed to Win32, Mac, etc...)
- 1990's, Netscape, Sun, etc... touting the Web Platform
- Microsoft was not so happy
- The Browser Wars
- Today, most OS platforms are Web platform enabled (browser + Java, etc...)

Static Pages



Dynamic Pages



Server Side Includes (SSI)

- .shtml files
- Directives embedded in HTML comments
 - `<!--#directive parameter=value parameter=value-->`
 - `<!--#include virtual="header.html"-->`
 - `<!--#flastmod virtual="foo.html" -->`
- Evaluated while page being served
- Can add dynamically generated content to page
- Slow

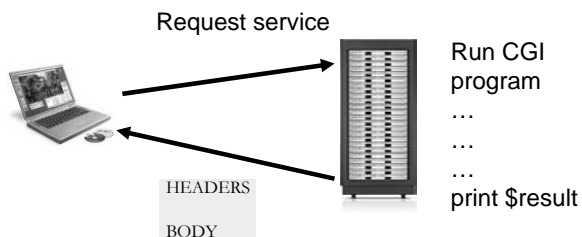
CGI – Common Gateway Interface

- Invented in 1993 by NCSA for HTTPd web server
 - Client requests program to be run on server-side
 - Web server passes parameters to program through UNIX shell environment variables
 - Program spawned as separate process via fork
 - Program's output => Results
 - Server passes back results (usually in form of HTML)
- Good for interfacing external applications with information servers
- See <http://hoohoo.ncsa.uiuc.edu/cgi/>

Competing Technologies

- CGI & Perl (HW #3)
- PHP - PHP Hypertext Preprocessor
 - LAMP Architecture (Linux, Apache, MySQL, PHP/Perl/Python)
- JSP - JavaServer Pages (HW #4)
- ASP - Active Server Pages

CGI Web Application



Just a Perl Program

- Write a standard Perl Program
- Program's output (to stdout) is sent back as HTTP Response
- You must write out everything
 - Headers
 - Blank Space
 - Body

printenv.pl (Client side)

```
#!/usr/pubsw/bin/perl
# $Id: printenv.pl,v 1.1 2004/04/13 04:15:36 morpheus Exp $
# printenv.pl -- demo perl program that prints out
# environment variables.

print "Content-type: text/plain\n\n";
foreach $var (sort(keys(%ENV))) {
    $val = $ENV{$var};
    $val =~ s|\n|\ |g;
    $val =~ s|"|\\"|g;
    print "${var}=\${val}\n\n";
}
```

```
elaine35:/usr/class/cs193i/cgi-bin> telnet cgi.stanford.edu 80
Trying 171.67.16.79...
Connected to cgi.Stanford.EDU (171.67.16.79).
Escape character is '^]'.
GET /class/cs193i/cgi-bin/printenv.pl HTTP/1.0
```

```
HTTP/1.1 200 OK
Date: Wed, 21 Jul 2004 18:00:33 GMT
Server: Apache
Connection: close
Content-Type: text/plain; charset=ISO-8859-1
```

```
DOCUMENT_ROOT="/web/htdocs"
GATEWAY_INTERFACE="CGI/1.1"
KRB5CCNAME="FILE:/tmp/K5tk25842class-cs193i.cgi"
KRBTKFILE="/tmp/tkt25842class-cs193i.cgi"
PATH="/usr/local/bin:/usr/pubsw/bin:/usr/bin:/bin"
QUERY_STRING=""
REMOTE_ADDR="171.64.15.110"
REMOTE_HOST="elaine35.stanford.edu"
REMOTE_PORT="46448"
```

```

REQUEST_METHOD="GET"
REQUEST_URI="/class/cs193i/cgi-bin/printenv.pl"
SCRIPT_FILENAME="/afs/ir/class/cs193i/cgi-bin/printenv.pl"
SCRIPT_NAME="/~class-cs193i/printenv.pl"
SCRIPT_URI="http://cgi.stanford.edu/class/cs193i/cgi-bin/printenv.pl"
SCRIPT_URL="/class/cs193i/cgi-bin/printenv.pl"
SERVER_ADDR="171.67.16.79"
SERVER_ADMIN="webmaster@stanford.edu"
SERVER_NAME="cgi.stanford.edu"
SERVER_PORT="80"
SERVER_PROTOCOL="HTTP/1.0"
SERVER_SOFTWARE="Apache"
Connection closed by foreign host.

```

Client-Side Analysis

- Nothing new:
looks like standard HTTP Request-Response
- But, actually:
Not return printenv.pl file, but rather the output of running that program!!!
- What if we move the printenv.pl file out of the cgi-bin directory???

printenv.pl in WWW directory

```

elaine35:/usr/class/cs193i/cgi-bin> telnet www 80
Trying 171.67.16.81...
Connected to www10.Stanford.EDU (171.67.16.81).
Escape character is '^]'.
GET /class/cs193i/printenv.pl HTTP/1.0

HTTP/1.1 200 OK
Date: Wed, 21 Jul 2004 18:05:09 GMT
Server: Apache
Last-Modified: Fri, 30 Apr 2004 04:42:41 GMT
ETag: "25f4da82-14f-79481240"
Accept-Ranges: bytes
Content-Length: 335
Connection: close
Content-Type: text/plain; charset=ISO-8859-1
Content-Language: en

#!/usr/pubsw/bin/perl
# $Id: printenv.pl,v 1.1 2004/04/13 04:15:36 morpheus Exp $
# printenv.pl -- demo perl program that prints out environment variables.
...

```

What happened?

- Same File Requested
- Different Directory Path
- Different Behaviors!
 - regular directory => returns the file
 - cgi-bin => returns output of the program
- Which Behavior is determined by Server
 - Based on directory, or file extension, ...

Server-Side

- Request from Client
 - If path in special cgi-bin directory, pass to CGI handler
- Headers
At minimum, Content-type (e.g. Content-type: text/html)
- Blank Space
- Body
 - HTML with interspersed output variables
 - Or images, text, pdf, ... depends on Content-type
- Send Results to Client as HTTP Response

Bottom Line

- Perl/CGI Web App Structure
 - CGI runs on server side
 - Put out HTML/Forms to present data and controls for user to take further actions

To Create Your Very Own CGI files

- Sign up for CGI capabilities
<http://cgi.stanford.edu/>
 - Click on "Activate Personal CGI Service" link
- Start Writing CGIs!
- Be careful of Security Issues

Hello World!

```
elaine35:/usr/class/cs193i/cgi-bin> less hello.pl
#!/usr/bin/perl -w
## Hello.pl -- demonstrate a trivial CGI that prints
## out some HTML and the current time on this server.
use strict 'vars';
my($EOL) = "\015\012";

## This is a human-readable str of the current time
my($nowStr);
$nowStr = localtime();

## This line must be included in the header
print "Content-type: text/html$EOL$EOL";
## Write out the HTML content
print "<html><head><title>Hello.pl</title></head>\n";
print "<body bgcolor=white>\n";
print "<h1>Hello.pl</h1>\n";
print "Hello there from CGI-land. It's currently '$nowStr'\n";
print "</body></html>\n";
```

HTML Forms

- Use web page to present choices to user
- `action=url`
 - Specifies URL of CGI that gets data
- `<input name="a-name"...>`
 - Maps response to form element
- `URL?name1=value1&name2=value2...`
 - Data returned to CGI via pairs
 - Funny characters use hexadecimal ASCII representation

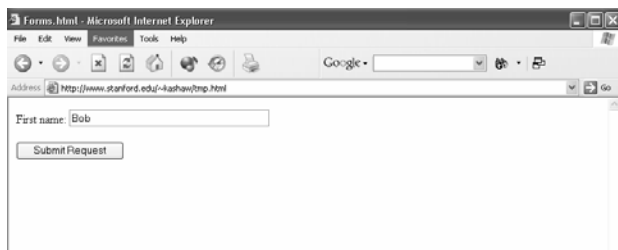
HTML Form Structure

```
<form action=http://cgi.stanford.edu/class/cs193i/cgi-bin/dumpenv.pl method=get>
  <!-- text input -->
  <p> First name: <input type=text name="first-name" size=40 value="Bob">
  <!-- submit button -->
  <p> <input type=submit name=go value="Submit Request">
</form>
```

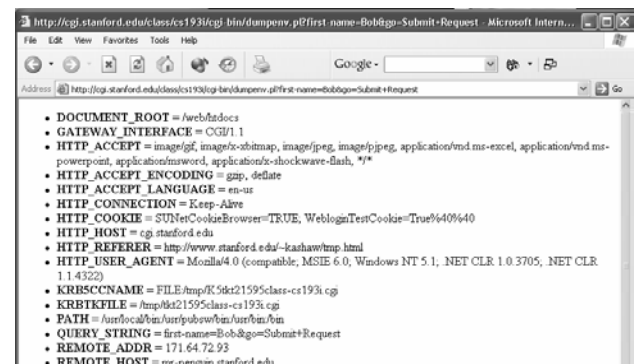
- Form Tag
 - Action Attribute Field
 - Method Attribute Field
- Input Tags Nested in Form
 - Name & Type (what type of input control)
 - Values / Bindings

HTML Form Structure

```
<form action=http://cgi.stanford.edu/class/cs193i/cgi-bin/dumpenv.pl method=get>
  <!-- text input -->
  <p> First name: <input type=text name="first-name" size=40 value="Bob">
  <!-- submit button -->
  <p> <input type=submit name=go value="Submit Request">
</form>
```



After Submit Button

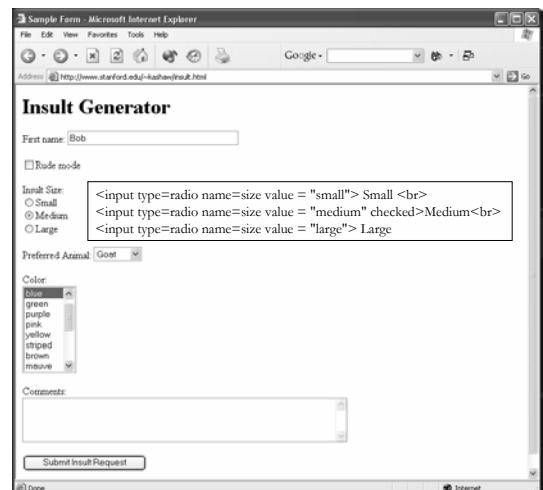
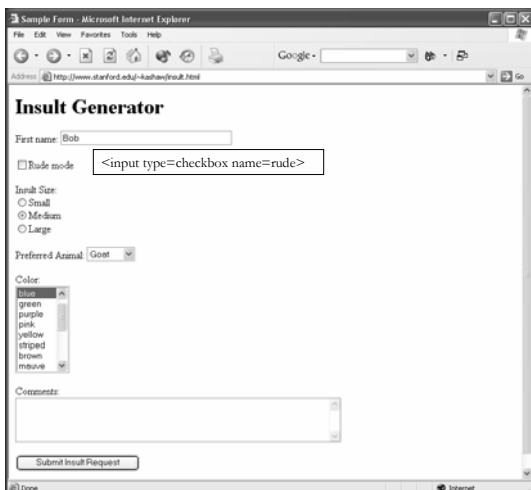
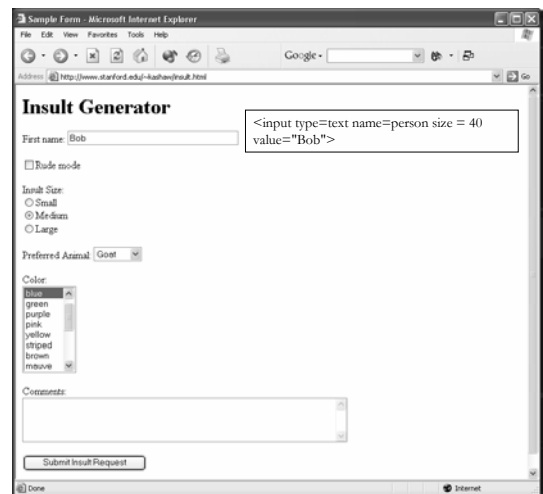
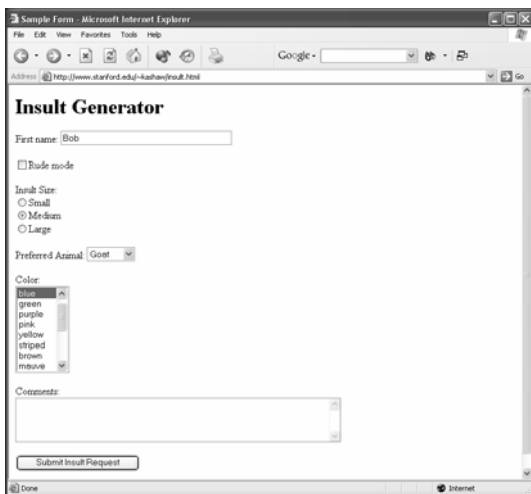


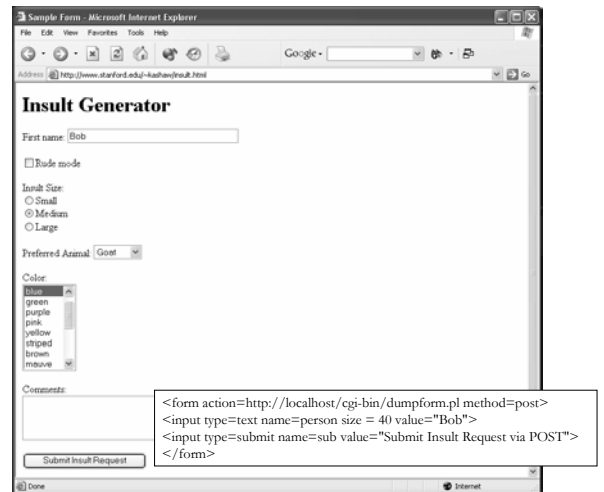
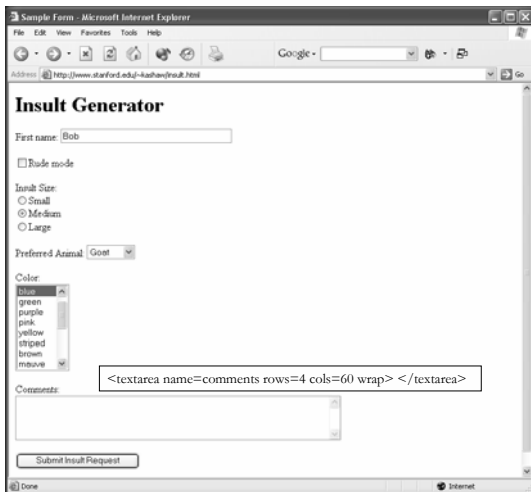
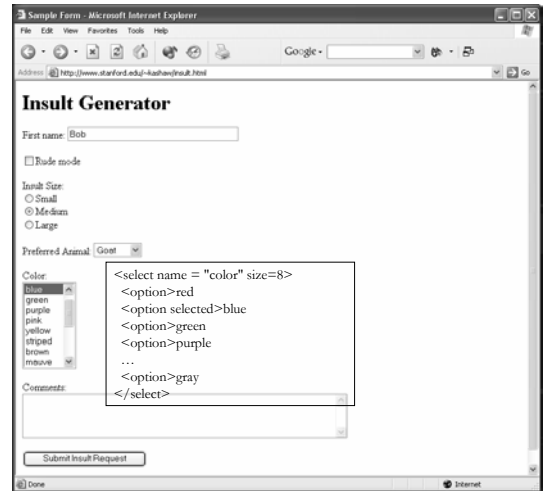
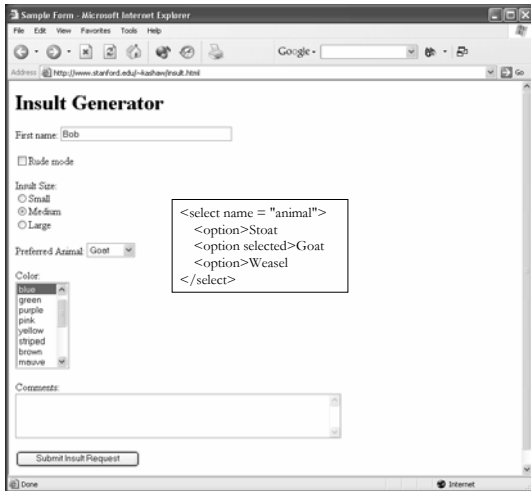
Input Tag Types

- `<input type=text name=first-name value="Bob">`
- `type=checkbox`
- `type=radio`
- `type=submit`
- `type=image`
- `type=hidden` (we'll see later!)
- `type=reset`

More Input Fields

- `<select>`
- `<textarea>`



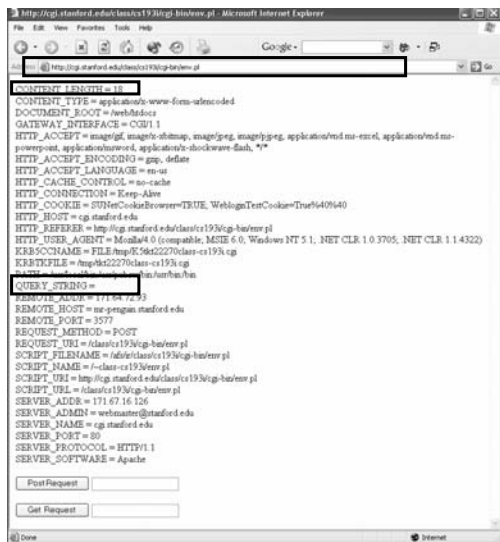


Getting Input Parameters

- Input can be submitted via GET or POST
<form ... method=xxx>
- Handle input parameters through CGI.pm Perl Module

Passing in Parameters

- GET Method
 - Bindings show up as UNIX Environment Variables
 - QUERY_STRING Environment variable is the query part (after the ?)
- POST Method
 - Passed in Content part of the HTTP Request
 - Shows up in CGI Program's stdin



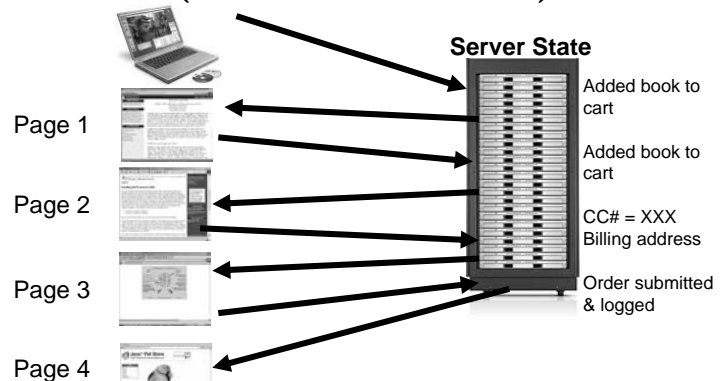
Get vs. Post

- GET
 - Attr/Val pairs attached after ?
 - + CGI operations can be bookmarked
 - - What happens if user refreshes, or clicks back button? Double Submit!
 - Use only for idempotent operations

Get vs. Post

- POST
 - Attr/Val pairs attached as Request Body
 - + CGI operations cannot be bookmarked
 - - If user refreshes, or clicks back button, browser may display warning
 - Can use for non-idempotent operations
 - Or idempotent ops with LONG URLs

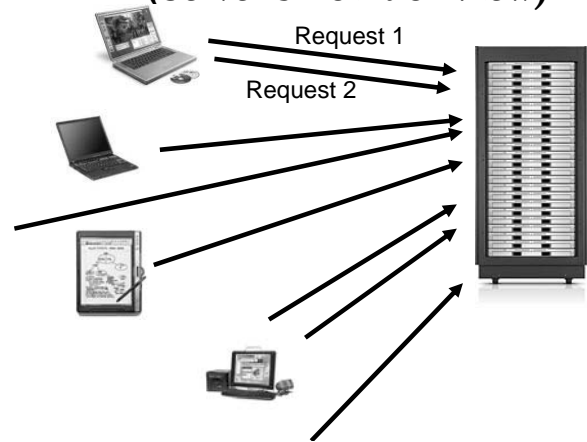
Continuity Problem (User's Point of View)



The Illusion of Continuity

- User thinks that choices made on page 1 are remembered on page 3
- However
 - HTTP is Stateless
 - Requests from same user do not necessarily come in adjacent requests

Continuity Problem (Server's Point of View)



Continuity Problem Resolution

- Back Button Problem
 - Serial Number Solution – track submitted orders
- Reconcile Double Submits
 - Add record example
 - May be intentional

Store State Somewhere

- HTTP is stateless
- Server Side?
 - Makes Server Really Complicated
 - State per client!
- Client Side?

“Post-It Notes”

- Server puts little notes on the client side
- When client submits the next form, it also (unknowingly) submits these little notes
- Server reads the notes, remembers who the client is

Technique: Hidden Fields

- `<input type=hidden name=myVar value=\"theVal\">`
- + simple way to store state on client side
- - what if the client (user)
 - closes browser, returns to your site 30 seconds later?
 - bookmarks your page?
 - enters your site through 3rd party links?

Technique: HTTP Cookies

- `http://wp.netscape.com/newsref/std/cookie_spec.html`
- Server can store bite sized information on client side, telling it which URLs this state is valid for
- When client requests one of those URLs, it transmits the "cookie" to the server
- + Site will remember who you are
- - Privacy?

Cookie Syntax

- On HTTP response, the server writes:
`Set-Cookie: NAME=VALUE;
expires=DATE; path=PATH;
domain=DOMAIN_NAME; secure`
- On HTTP requests, the client looks through cookie database, finds all cookies that match the current URL (domain+path), and writes:
`Cookie: NAME1=OPAQUE_STRING1;
NAME2=OPAQUE_STRING2; ...`

Cookie Example

Client requests a document, and receives in the response:

```
Set-Cookie: CUSTOMER=WILE_E_COYOTE; path=/  
expires=Wednesday, 09-Nov99 23:12:40 GMT
```

When client requests a URL in path "/" on this server, it sends:

```
Cookie: CUSTOMER=WILE_E_COYOTE
```

Client requests a document, and receives in the response:

```
Set-Cookie: PART_NUMBER=ROCKET_LAUNCHER_0001; path=/
```

When client requests a URL in path "/" on this server, it sends:

```
Cookie: CUSTOMER=WILE_E_COYOTE;  
PART_NUMBER=ROCKET_LAUNCHER_0001
```

Client receives:

```
Set-Cookie: SHIPPING=FEDEX; path=/foo
```

Cookie Example

When client requests a URL in path "/" on this server, it sends:

```
Cookie: CUSTOMER=WILE_E_COYOTE;  
PART_NUMBER=ROCKET_LAUNCHER_0001
```

When client requests a URL in path "/foo" on this server, it sends:

```
Cookie: CUSTOMER=WILE_E_COYOTE;  
PART_NUMBER=ROCKET_LAUNCHER_0001; SHIPPING=FEDEX
```

Some Details

<<EOT; Perl Syntax

- Puts raw text into specified string

```
$string = <<EOT;  
Stuff  
More Stuff  
EOT
```

- << EOT marks start of data
- EOT on line by itself with no whitespace marks end

```
#!/usr/bin/perl -wT
# 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";

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

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

# Emit an HTML <ul> for all the environment vars
# set up for the CGI
print $header

print "<ul>\n";
## iterate over the keys, but sort them first
foreach $key (sort (keys %ENV)) {
    print "<li><b>$key</b> = $ENV{$key}\n";
}
}
```

```
elaine35:/usr/class/cs193i/cgi-bin> telnet cgi.stanford.edu 80
Trying 171.67.16.79...
Connected to cgi1.Stanford.EDU (171.67.16.79).
Escape character is '^]'.
GET /class/cs193i/cgi-bin/dumpenv.pl HTTP/1.0

HTTP/1.1 200 OK
Date: Wed, 21 Jul 2004 18:22:46 GMT
Server: Apache
Connection: close
Content-Type: text/html; charset=ISO-8859-1

<ul>
<li><b>DOCUMENT_ROOT</b> = /web/htdocs
<li><b>GATEWAY_INTERFACE</b> = CGI/1.1
...
</ul>
</body>
</html>
```

CGI.pm Module

- Object Oriented or Function-Oriented
- Enables easy parsing of inputs

```
use CGI;
$query = new CGI;
@names = $query->param; ## all variable names
$value = $query->param('color'); ## may be undef
@values = #query->param("sizes"); ## multi-binding
```

<http://jan.netcomp.monash.edu.au/ecommerce/CGI-pm.html>

```
use CGI;
my $q = new CGI;
print $q->header("text/html");

# Print out all the key/value pairs...
print "<html><head><title>Form Bindings</title></head>";
print "<body bgcolor=white>\n";
print "<h1>Your Key/Value Bindings...</h1>\n";
print '<table border=1 width="100%">'; ## note use of ' to hide " in string

my(@vars, $var, $val);
@vars = $q->param;
foreach $var (sort @vars) {
    $val = $q->param($var);
    print "<tr>\n"; ## one <tr> for each row
    print "<td><b>$var</b></td>\n"; ## one <td> for each elt
    print "<td>$val</td>\n";
    print "</tr>\n";
}
print "</table>\n";
print "</body></html>\n";
```

CGI Handling Methods

- param
- delete
- delete_all
- save
- url
- cookie
- ...

Form / HTML Methods

- start_html
- end_html
- startform
- textfield
- textarea
- password_field
- filefield
- popup_menu
- scrolling_list
- ...
- submit
- hidden
- ...

Environment Variable Methods

- user_agent
- path_info
- remote_host
- referer
- request_method
- ...