

**Basic Perl CGI Programming**

EIW - Perl CGI

1

---

---

---

---

---

---

---

---

**Issues**

- How and when your program is invoked.
- Generating Response
  - HTTP Headers
  - HTML (or whatever document type you want)
- Getting at form fields (query string).

EIW - Perl CGI

2

---

---

---

---

---

---

---

---

**Remember!**

- Your Perl CGI program is run whenever a request is received.
- Every request starts a new copy of your program.
- A response is not complete until your program has terminated

EIW - Perl CGI

3

---

---

---

---

---

---

---

---

## The Response

- To operate as a CGI program, your Perl program must:
  - Send a "Content-type:" HTTP response header back to the client (print it out).
  - Send back other HTTP response headers (optional).
  - Send back a blank line
    - this signals the end of the HTTP response headers.
  - Create some kind of document and send it back to the client.
    - typically HTML

EIW - Perl CGI

4

---

---

---

---

---

---

---

---

## Simple Example

```
#!/usr/bin/perl

# send required HTTP response header and blank line.
print "Content-type: text/html\n\n";

# This saves lots of quoting and individual calls to
# print!

print<<EOHTML;
<HTML><HEAD> <TITLE>Simple Perl CGI</TITLE></HEAD>
<BODY>
<H1>I am generated by a Perl CGI program!</H1>
</BODY>
</HTML>

EOHTML
```

EIW - Perl CGI

5

---

---

---

---

---

---

---

---

## Another Example

```
#!/usr/bin/perl

# send required HTTP response header and blank line.
print "Content-type: text/html\n\n";

# send HTML document head
print "<HTML><HEAD> <TITLE>Powers of 2</TITLE></HEAD>\n";
print "<BODY>\n";

# Create a table
print "<TABLE BORDER=2>\n";
for ($i=0;$i<10;$i++) {
    print "<TR><TD>$i</TD><TD> . 2**$i . </TD></TR>\n";
}
print "</TABLE>\n";

# close out the document body
print "</BODY></HTML>\n";
```

EIW - Perl CGI

6

---

---

---

---

---

---

---

---

## Perl localtime function

- So far the examples have not been dynamic
  - always the same document.
- We can use the **localtime** function to include the current date and time on the web page.

```
$time = localtime();
```

---

---

---

---

---

---

---

---

## localtime example

```
#!/usr/bin/perl
# Create document that knows what time it is

$time = localtime(); # time is a string date/time

# send HTTP header.
print "Content-type: text/html\n\n";

# send a document that includes the time
print <<EODOC
<HTML><HEAD> <TITLE>Time and Date</TITLE></HEAD>
<BODY>
<H1>It is now: $time</H1>
</BODY>
</HTML>

EODOC
```

---

---

---

---

---

---

---

---

## Forms & CGI

- Many CGI programs are sent form field names and values from the browser as part of the request.
- The document sent back depends on the values of the form fields (the query string).

---

---

---

---

---

---

---

---

## Getting the form fields

- We could do this:
  - get the request method from an *environment variable*.
  - get the query string from a different *environment variable* (GET) or from STDIN (POST).
  - Split the query up and do URL decoding on each field name and value.

EIW - Perl CGI 10

---

---

---

---

---

---

---

---

## -OR-

- We can use a Perl CGI library that does all the work for us!
- There are many Perl CGI libraries (modules), we will look at the most basic module: **CGI.pm**
- We won't worry about *environment variables*, etc...

EIW - Perl CGI 11

---

---

---

---

---

---

---

---

## Telling Perl you want to use **CGI.pm**

```
use CGI ':standard';
```

- sometimes you will see this as:

```
use CGI qw/:standard/;
```
- There are a few *flavors* of this module, we will look at *standard* (which is based on perl subroutines).
  - you can also use the *Object Oriented* version...

EIW - Perl CGI 12

---

---

---

---

---

---

---

---

## CGI.pm

- Complete documentation is available at:

<http://search.cpan.org/author/JHI/perl-5.8.0/lib/CGI.pm>

The suggested book "Perl and CGI for the WWW" covers only the basics of using CGI.pm

EIW - Perl CGI

13

---

---

---

---

---

---

---

---

## Getting Form Fields

- CGI.pm provides the function `param()`
- If you don't pass any parameters you get back a *list* of all the field *names*.

```
@names = param();
```

EIW - Perl CGI

14

---

---

---

---

---

---

---

---

## Sample Query String

```
foo.pl?size=large&color=red&x=3
```

```
@names = param();
```

```
##@names is now:
```

```
# ("size", "color", "x")
```

EIW - Perl CGI

15

---

---

---

---

---

---

---

---

## Getting the value of a field

- If you pass `param` a parameter, it assumes the parameter is a field *name*. `param` will return the *value* of that field (from the query string).

```
$size = param('size');  
$clr = param('color');
```

EIW - Perl CGI

16

---

---

---

---

---

---

---

---

## Typical CGI Program

- Usually you know what form fields to expect.
  - you know the names, but not the values.
- You should always remember that your CGI can receive any query string, not just one created by your own forms...

EIW - Perl CGI

17

---

---

---

---

---

---

---

---

## Generic CGI

- We can write a Generic CGI that doesn't care what form fields are sent, it just prints out all the names and values in an HTML table.
- This can be useful for testing forms sent via POST (to make sure they are sending what you want).

EIW - Perl CGI

18

---

---

---

---

---

---

---

---

## Perl Generic CGI

```
#!/usr/bin/perl
use CGI ':standard';

# get a list of all the field names received.
@names = param();

# tell the browser we are sending HTML
print "Content-type: text/html\n\n";

# print HTML document head
print<<EOHEAD;
<HTML>
<HEAD><TITLE>Generic</TITLE></HEAD>
<BODY>
EOHEAD
```

EIW - Perl CGI

19

---

---

---

---

---

---

---

---

## Perl Generic CGI (cont.)

```
# build a table that displays all the fields received
print "<TABLE BORDER=2>\n";
for ($i=0;$i<=#names;$i++) {
    $fieldname=$names[$i];
    $val = param($fieldname);
    print "<TR><TD>$fieldname</TD><TD>$val</TD></TR>\n";
}
print "</TABLE>\n";

# Finish off the document
print "</BODY></HTML>\n";
```

EIW - Perl CGI

20

---

---

---

---

---

---

---

---

## A CGI Login Program

- We create a login form
  - fields "name" and "password"
- CGI that receives the queries from our form
  - our program is the form ACTION
- We compare the name and password to a list of valid entries.
  - send back an error or send back a welcome

EIW - Perl CGI

21

---

---

---

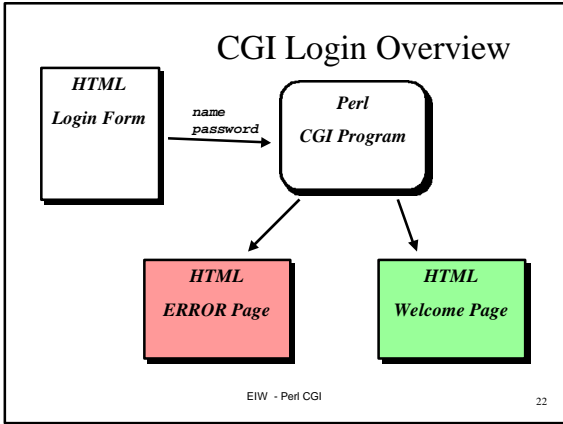
---

---

---

---

---




---

---

---

---

---

---

---

---

### Perl Login CGI

We need the following code:

1. send back HTTP header
2. get the name and password
3. find out if the name,password are valid
  - A. generate welcome page
  - B. generate error page

EIW - Perl CGI 23

---

---

---

---

---

---

---

---

### Steps 1 and 2

```

#!/usr/bin/perl
#
# Login CGI

use CGI ':standard';

# send required HTTP response header
print "Content-type: text/html\n\n";

# Get the name and password from the query
$user = param('name');
$pw = param('password');
  
```

EIW - Perl CGI 24

---

---

---

---

---

---

---

---

## Checking the input

- How do we know if the name and password are correct?
  - in a real system we would have a database with names,passwords.
  - for now we will just "hard-code" the right values.

EIW - Perl CGI

25

---

---

---

---

---

---

---

---

## Verifying Name,Password

```
# Make sure the name and password are correct

if (($user eq "dave") && ($pw eq "eiw")) {
    print "<H1>Welcome $user, you are valid.</H1>\n";
} else {
    print "<H1>You are not valid!</H1>\n";
}

print "</BODY></HTML>\n";
```

EIW - Perl CGI

26

---

---

---

---

---

---

---

---

## What to send back.

- The code just sends back a single line that says welcome or error.
  - this is not realistic!
- In general we want to send an entire document.
- We could use print statements... but there is a better way.

EIW - Perl CGI

27

---

---

---

---

---

---

---

---

## Perl subroutine `sendfile()`

```
sub sendfile() {  
    my($filename) = $_[0];  
    open(F,$filename);  
    while (<F>) {  
        print;  
    }  
    close(F);  
}
```

EIW - Perl CGI

28

---

---

---

---

---

---

---

---

## Using our `sendfile()` subroutine

- We must include the definition of `sendfile()` in the Perl program.
- We can now consider creating a welcome page as a file: `welcome.html`
- We can also create a copy of the login form in the file: `form.html`

EIW - Perl CGI

29

---

---

---

---

---

---

---

---

## Revised Perl Code

```
# Make sure the name and password are correct  
if (($user eq "dave") && ($pw eq "eiw")) {  
    sendfile("welcome.html");  
} else {  
    print "<H1>You are not valid!</H1>\n";  
    sendfile("form.html");  
}
```

EIW - Perl CGI

30

---

---

---

---

---

---

---

---