Programming in Perl

Week Twelve

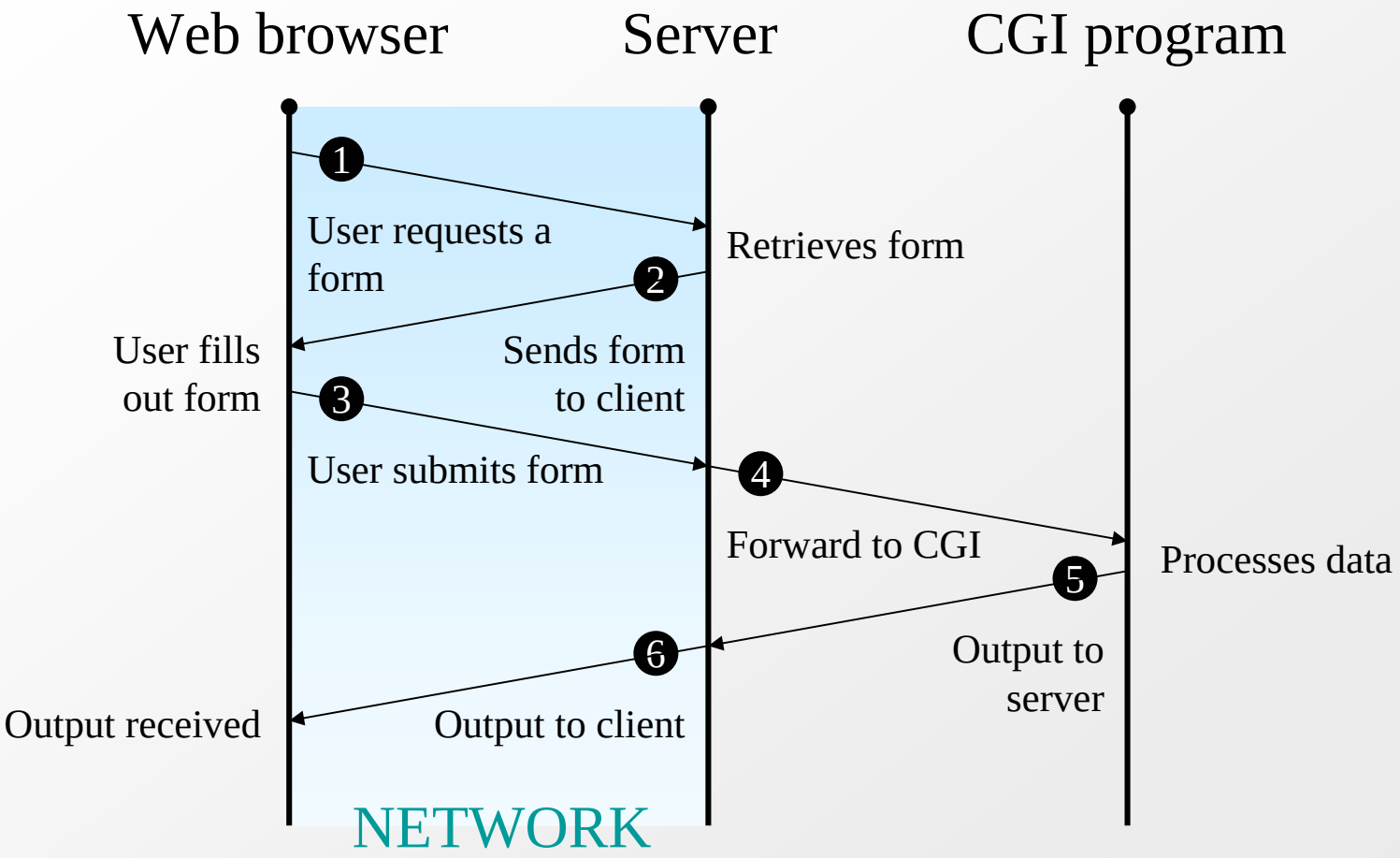
Programming the Web- CGI

Jump to first page

What is a CGI?

- CGI stands for "Common Gateway Interface"
  - It is a gateway that allows you to execute code on the Web server and return the results to the Web browser
  - This allows you to generate Web pages on the fly
- CGI programs can be in C, C++, Java, Visual Basic, Perl or just about any language that can run on your Web server
- Why Perl, then?
  - There are hundreds of CGI scripts written in Perl available
  - There are modules available that automate may things
  - Perl has been called, "the duct tape of the Internet"

How do CGIs work?



Your first CGI

```
#!/usr/bin/perl
my $time = localtime(time);
print "Content-type: text/html\n\n";
print <<end_of_block;
<HTML>
    <HEAD>
        <TITLE>Hello World</TITLE>
    </HEAD>
    <BODY>
    <H1>Greetings, Terrans!</H1>
    <P>The Time is $time
    </BODY>
</HTML>
end_of_block
```

This line specifies the MIME type of the message to the client. The two newlines are important!!

This is the HTML content of the message in a "here-document"

How do CGIs get information?

- The URL specification describes a "query string"

  `http://www.perl.org/cgi-bin/order?flavor=grape&size=small`

  - This URL is generated when a "submit" button is pressed from within a form
- The `cgi-bin` is a location for CGIs specified by the server
  - It can be called anything, but is traditionally `cgi-bin`
- The CGI program to execute is next followed by a '?' and the for contents of the form
  - When used from a `<FORM>` tag the `name=value` pairs are the field and its value
  - Multiple values are separated with a '&'
- You can generate this URL yourself

Use the CGI.pm Module To Process the URL

```
#!/usr/bin/perl
# cgi-bin/ice_cream;
use CGI qw(param);
my $favorite = param('flavor');
print "Content-type: text/html\n\n";
print <<end_of_block
<HTML>
    <HEAD>
        <TITLE>Hello World</TITLE>
    </HEAD>
    <BODY>
    <H1>Hello, World!</H1>
    <P>Your favorite flavor is $favorite.
    </BODY>
</HTML>
end_of_block
```

This line imports the param function from CGI.pm

This line extracts the value of the "flavor" variable from the URL

The value of "flavor" is inserted here

`CGI.pm` can do more work that parameter passing!

- `CGI.pm` has lots of functions that do things for you
- The functions are grouped together to make it easier to import them.
  - `:cgi` - import all argument-handling methods, like `param()`
  - `:form` - import all fill-out form generation methods, like `textfield()`
  - `:html2` - import all methods that generate HTML 2.0 standard elements
  - `:html3` - import all methods that generate HTML 3.0 elements
  - `:netscape` - import all methods that generate Netscape-specific HTML extensions
  - `:html` - `:html2` + `:html3` + `:netscape`
  - `:standard` - `:html2` + `:form` + `:cgi`
  - `:all`

ice_cream again

```
#!/usr/bin/perl -w
# cgi-bin/ice_cream V2
use CGI qw(:standard);
print header, start_html('Hello World'), h1('Hello, World!');
my $favorite = param("flavor");
print p("Your favorite flavor is $favorite.");
print end_html;
```

Web page to drive `ice_cream`

```
<!-- ice_cream.html -->
<HTML>
    <HEAD>
    <TITLE>Hello Ice Cream</TITLE>
    </HEAD>
    <BODY>
    <H1>Hello Ice Cream</TITLE>
    <FORM ACTION="http//www.somewhere.org/cgi-bin/ice_cream">
    What's your flavor? <INPUT NAME="favorite" VALUE="mint">
    <P>
    <INPUT TYPE="submit">
    </FORM>
    </BODY>
<HTML>
```

But wait! A CGI can generate HTML!

```
#!/usr/bin/perl -w
# cgi-bin/ice_cream V3
use CGI qw(:standard);
my $favorite = param("flavor");
print header, start_html('Hello Ice Cream'),
    h1('Hello Ice Cream');
if ($favorite) {
    print p("Your favorite flavor is $favorite.");
} else {
    print hr, start_form;    # hr() emits a horizontal rule
    print p("Please select a flavor: ",
        textfield("flavor", "mint"));
    print end_form, hr;
}
print end_html;
```

If the value is set display the new page, else display the form

The Last ice_cream

```
#!/usr/bin/perl -w
# cgi-bin/ice_cream V4
use strict; # enforce variable declarations and quoting
use CGI qw(:standard);

print header, start_html("Ice Cream Stand"),
    h1("Ice Cream Stand");

if (param()) { # the form has already been filled out
    my $who = param("name");
    my $flavor = param("flavor");
 my $scoops = param("scoops");
 my $taxrate = 1.0743;
 my $cost = sprintf("%.2f", $taxrate * (1.00 + $scoops *
   0.25));
 print p("Ok, $who, have $scoops scoops of $flavor for
   \$$cost.");
```

The Last `ice_cream`

```
} else { # first time through, so present clean form
    print hr(); # draw a horizontal rule before the form
    print start_form();
    print p("What's your name? ", textfield("name"));

    print p("What flavor: ", popup_menu("flavor",
        ['mint','cherry','mocha']));
    print p("How many scoops? ", popup_menu("scoops",
        [ 1..3 ]));
    print p(submit("order"), reset("clear"));
    print end_form(), hr();
}
print end_html;
```

CGI.pm "widgets"

- CGI.pm can generate any HTML form
  - Text Fields

    ```
    textfield("name")
    ```
  - Password fields

    ```
    Password_field("password")
    ```
  - Buttons

    ```
    submit("order"), reset("clear")
    ```
  - Image Maps

    ```
    image_button('arizona', '/image/arizona', 'MIDDLE')
    ```

Fancier Calling Sequences

- `CGI.pm` widgets have a more general calling sequence

```
print scrolling_list(
    -NAME => 'flavors',
    -VALUES => [ qw(mint chocolate cherry vanilla peach) ],
    -LABELS => {
        mint => "Mighty Mint",
        chocolate => "Cherished Chocolate",
        cherry => "Cheery Cherry",
        vanilla => "Very Vanilla",
        peach => "Perfectly Peachy",
    },
    -SIZE => 3,
    -MULTIPLE=> 1, # 1 for true, 0 for false
);
```

Guestbook

```
#!/usr/bin/perl -w
use 5.004;
use strict;             # enforce declarations and quoting
use CGI qw(:standard); # import shortcuts
use Fcntl qw(:flock);  # imports LOCK_EX, LOCK_SH, LOCK_NB

sub bail {              # function to handle errors gracefully
    my $error = "@_";
    print h1("Unexpected Error"), p($error), end_html;
    die $error;
}

my(
    $CHATNAME, # name of guestbook file
    $MAXSAVE,  # how many to keep
    $TITLE,    # page title and header
    $cur,      # new entry in the guestbook
    @entries,  # all cur entries
    $entry,    # one particular entry
);
```

Guestbook

```
$TITLE = "Simple Guestbook";
$CHATNAME = "/usr/tmp/chatfile"; # wherever makes sense on your system
$MAXSAVE = 10;

print header, start_html($TITLE), h1($TITLE);

$cur = CGI->new();                          # current request
if ($cur->param("message")) {               # good, we got a message
    $cur->param("date", scalar localtime); # set to the current time
 @entries = ($cur);                         # save message to array
}
```

Guestbook

NAME1=VALUE1
NAME2=VALUE2
NAME3=VALUE3

```
if ( -e $CHATNAME ) {
   # open the file for read-write (preserving old contents)
   open(CHANDLE, "+< $CHATNAME") || bail("cannot open $CHATNAME: $!");
} else {
   # open a new file for read-write
   open(CHANDLE, "+> $CHATNAME") || bail("cannot create $CHATNAME:
   $!");
}

# get exclusive lock on the guestbook (LOCK_EX == exclusive lock)
flock(CHANDLE, LOCK_EX) || bail("cannot flock $CHATNAME: $!");

# grab up to $MAXSAVE old entries, newest first
while (!eof(CHANDLE) && @entries < $MAXSAVE) {
    $entry = CGI->new(\*CHANDLE); # pass the filehandle by reference
    push @entries, $entry;
}
seek(CHANDLE, 0, 0) || bail("cannot rewind $CHATNAME: $!");
foreach $entry (@entries) {
    $entry->save(\*CHANDLE); # pass the filehandle by reference
}
truncate(CHANDLE, tell(CHANDLE)) ||
                                    bail("cannot truncate $CHATNAME: $!");
close(CHANDLE) || bail("cannot close $CHATNAME: $!");
```

Guestbook

```
print hr, start_form;          # hr() emits html horizontal rule: <HR>
print p("Name:", $cur->textfield(
    -NAME => "name"));
print p("Message:", $cur->textfield(
    -NAME => "message",
    -OVERRIDE => 1,            # clears previous message
    -SIZE => 50));
print p(submit("send"), reset("clear"));
print end_form, hr;

print h2("Prior Messages");
foreach $entry (@entries) {
    printf("%s [%s]: %s",
    $entry->param("date"),
    $entry->param("name"),
    $entry->param("message"));
    print br();
}
print end_html;
```

CGI.pm

- There is lots more to CGI.pm
  - Check boxes
  - Radio groups
  - Hidden fields
  - Cookies
  - Frames
- Limited support Cascading Style Sheets
- Read the CGI.pm manual on www.perldoc.com