

CGI 4

Survey Example

Presents table of movies/books

Can "vote" for a row (writes vote back to file)

One vote per IP addr -- use \$query->remote_addr

Filenum -- Security, Hidden field

Client specifies a 'filenum'

Security issue vs. letting the client specify a filename

Use a hidden field to maintain continuity

Multiple Vote Buttons

Each row in the table has a "vote" button

How to distinguish which got clicked?

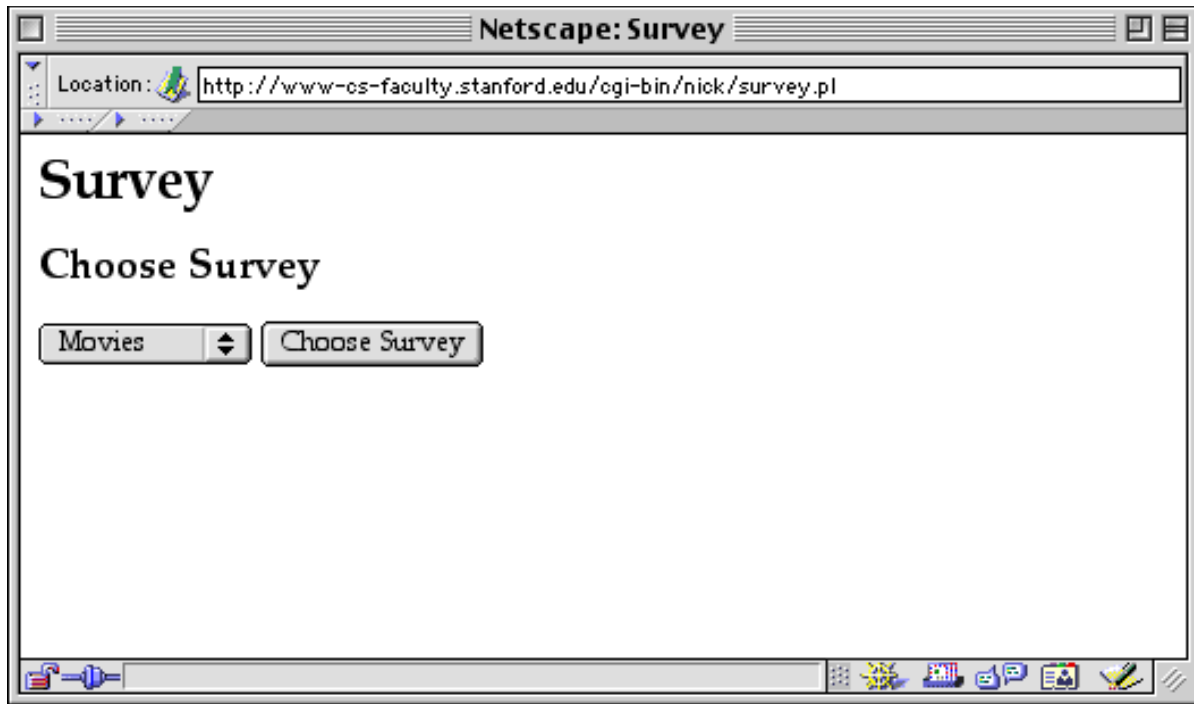
Cannot use value=, since that's the button title

Use the name of the button instead -- v_0, v_1, ...

On server side, look through the bindings to figure out which button was clicked.

Use \$query->param() to get an array of the binding names

Survey Example




Netscape: Survey

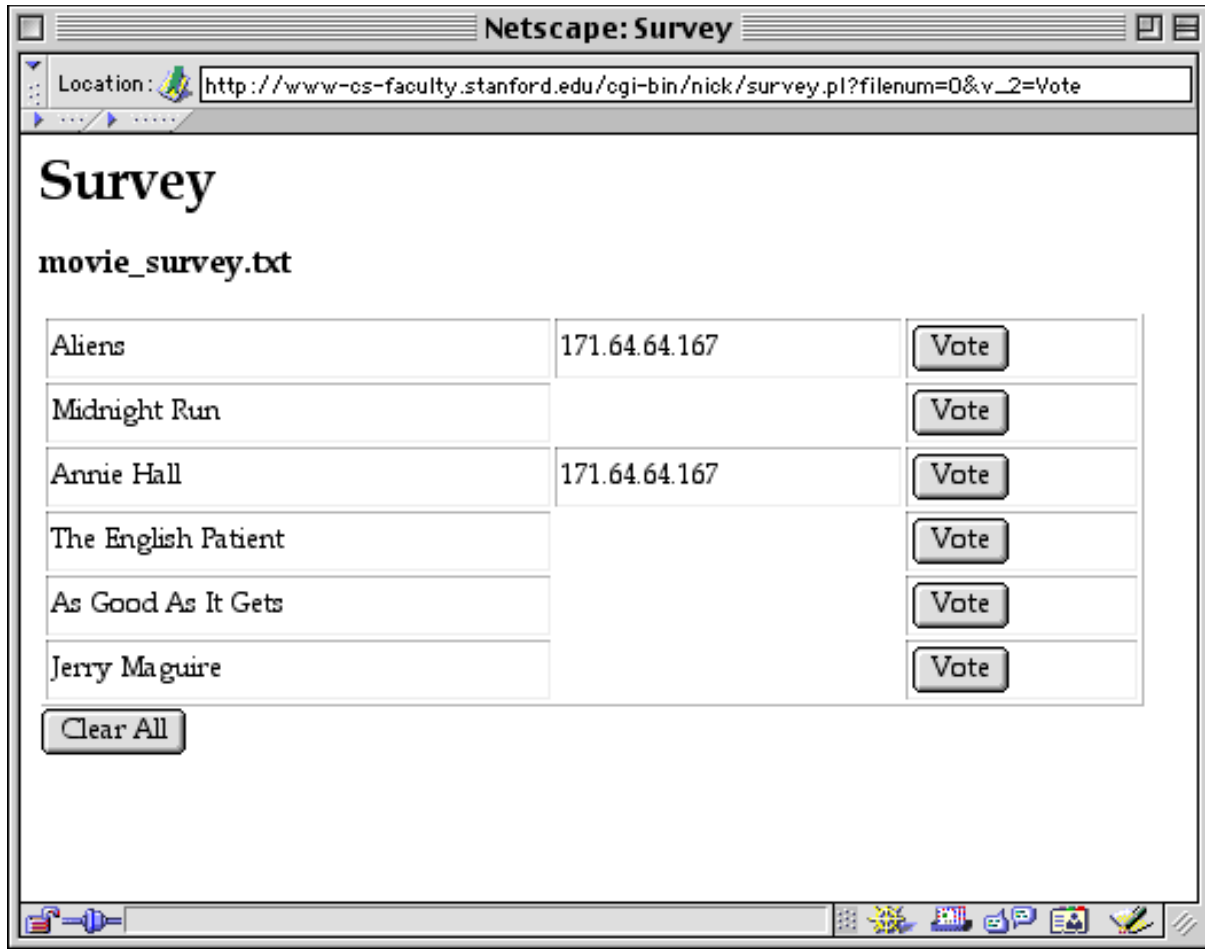
Location: <http://www-cs-faculty.stanford.edu/cgi-bin/nick/survey.pl?filenum=0&choose=Choose+Sur>

Survey

movie_survey.txt

Aliens	<input type="button" value="Vote"/>
Midnight Run	<input type="button" value="Vote"/>
Annie Hall	<input type="button" value="Vote"/>
The English Patient	<input type="button" value="Vote"/>
As Good As It Gets	<input type="button" value="Vote"/>
Jerry Maguire	<input type="button" value="Vote"/>





Survey Code

```
#!/usr/bin/perl -w
use strict 'vars';

## Survey CGI -- lets people vote for the row they like.
## Supports multiple files -- uses a hidden field to track
## which file the user is on.

use CGI;

#### HTML start and end
my $html_start = <<EOT;
<html><head>
<title>Survey</title>
</head>
<body bgcolor=white>
<h1>Survey</h1>
EOT

my $html_end = "</body></html>\n";
```

```

#### Start Output
## $| = 1;          ## sets STDOUT to unbuffered (optional)
print "Content-type: text/html\n\n";
print $html_start;

my $query = new CGI;

my $filenum = $query->param("filenum");
my $filename;

# Encode which file as a number index into our filanem array.
# This is very simple and effective strategy to limit what we pass to open()
my @filenames = ("movie_survey.txt", "book_survey.txt");

my @lines;      ## global var -- the lines of text

if (defined($filenum)) {
    $filename = $filenames[$filenum]; ## note: filename from OUR array
}

# Security note: we do not send a string from the client
# side into an open() call. In perl, open() can do all sorts
# of things, such as "rm *", so we must be careful what we pass it.

#### Choose Form
my $chooseForm = <<EOT;
<h2>Choose Survey</h2>
<p>
<form method=get>
<select name=filenum>
<option value=0>Movies
<option value=1>Books
</select>
<input type=submit name=choose value="Choose Survey">
</form>
EOT

#### Response logic
if (!defined($filename)) { ## no file -> choose form
    print $chooseForm;
    print $html_end;
    exit(0);
}
else {
    ReadFile(); ## loads the global @lines array
}

```

```

## clear -> erase all votes
if (defined($query->param("clear"))) {
    my $line;
    foreach $line (@lines) {
        my($text, $votes) = ParseRow($line);
        $line = $text . "\n";
    }
    WriteFile();
}

## check for vote -> update that row
my $num = FindVoteSubmit();
if ($num != -1) {
    my ($text, $votes) = ParseRow($lines[$num]);
    my $ip = $query->remote_addr;
    ## (above example of getting HTTP header field from CGI obj)
    ## add this ip as a vote, if it has not already voted
    if (index($votes, $ip) == -1) {
        $votes = $votes . " $ip";
        $lines[$num] = $text . "\t" . "$votes" . "\n";
        WriteFile();
    }
}

## In any case, now print out the table
## Use hidden field to remember which file we're on
print "<h3>$filename</h3>\n";
print "<form method=get>\n";
print "<input type=hidden name=filenum value=$filenum>\n";
print "<table border=1 width = 100%>\n";

my $line;
my $i;
for ($i=0; $i<scalar(@lines); $i++) {
    $line = $lines[$i];
    my ($text, $votes) = ParseRow($line);

    ## trick: use v_NNN as button name, where NNN is row index
    my $buttonName = "v_" . $i;

print <<EOT;
<tr><td>$text</td><td>$votes</td>
<td><input type=submit name=$buttonName value=Vote></td></tr>
EOT

}

print "</table>\n";
print "<input type=submit name=clear value=\"Clear All\" ></form>\n";

print $html_end;
exit(0);
}

```

```

## Returns the number of a v_NNN submission,
## or -1 if none found.
sub FindVoteSubmit {
    my @params = $query->param();
    my $param;
    foreach $param (@params) {
        if ($param =~ /^v_(\d+)/) {
            return $1;
        }
    }
    return(-1);
}

## Given one line of text, parse it into text and vote strings
## and return them as an array length 2
sub ParseRow {
    my($line) = @_;
    chomp($line);
    my ($text, $votes) = split(/\t/, $line);
    if (!defined($votes)) { $votes = ""; }
    return($text, $votes);
}

## Reads $filename into @lines array
sub ReadFile {
    open(DATA, "$filename") || ReportError("Cannot open '$filename'");
    ## Security: do not pass a string from the client to open()

    @lines = <DATA>;    ## read whole file

    close(DATA);
}

## Writes @lines array to $filename
sub WriteFile {
    open(DATA, ">$filename") || ReportError("Cannot open '$filename' for writing");
    my($line);
    foreach $line (@lines) {
        print DATA $line;
    }
    close(DATA);
}

## Helper that prints an error and exits
sub ReportError {
    my($err) = @_;

    print "\n<h1>Error</h1><p>$err\n</html>\n";
    exit(0);
}

```

Many Forms / Rowindex Alternative

Problem: which vote button was clicked

Above: use button name

Alternate: have many little forms, one per row.

Each little form has the filename hidden field, a submit button, and a 2nd hidden field called "rowindex"

In some ways this is cleaner, but it involves a lot of repetition

Security

Strings from the "client side" -- not trusted

open() problems

In Perl, open() can run a program, such as with the filename "rm -rf * |"

In any case, open() could be used to return an unintended file on the server

Therefore: careful what we pass to open

Filenum solution

Simple: the client can only index into our array

Effective, but possibly inconvenient, since array must be kept up to date

Filename field problems

What if we put hidden "filename" field in the form

Is that safe?

No, the client could just edit the form however they like, and then submit it

Filename syntax checking

Could get filename from client side, but check it syntactically (e.g. must be a word, followed by .txt)...

```
if (!$fname =~ m/^[w\-\_]+\\.txt$/) {
    ReportError("Filename must not contain funny chars, but got '$fname'");
}
```

Get/Post/Back button

Browser behavior varies with the back button

Survey seq: clear, vote0, vote1 -- what happens when we hit the back button?

Mostly: if the GET method is used, the back button does not do a submit, it just shows the old HTML

If the POST method is used, the browser may ask the user if they would like to do a submit

In general, getting your web app to behave perfectly for all scenarios where the user uses the back button is impossible.