

CGI 4

Basic Java Section

Optional introduction to the Java language

Thu may 17, Terman Aud, 3:15-5:05

Broadcast Thursday, May 17 at 5:45-7:35 on Channel E1.

Also, you can see the link the Sun Java tutorial, which also covers Java basics.

The Fundamental Issue of CGI

Applications...

The one key thing you need to absorb to understand server-side web programming.

1. GUI Applications

Users are used to this...

State in local memory

GUI displays that state

Edit operations change the state -- there's only one copy of the state

Going "back" shows the same state

2. Web Applications

User View

See page 1

The user makes a choice -- Submits a form or click a link

See page 2

The user makes a choice -- Submits a form or click a link

See page 3

Illusion

The user thinks of page1, page2, page3, as all being edits on one, unified state.

The user thinks that the choices made to lead to page2 are somehow remembered in page3

Server View

Request 1-> page 1

time passes, other users make requests

Request2 -> page 2

time passes, other users make requests

Request3 -> page 3

Server -- No Continuity

When Request2 comes in, there is nothing in HTTP that formally connects it to Request1

The server will need to do extra work to create the illusion that page1, page2, page3, all go together

Server -- Breadcrumbs

When returning page2, the server puts extra information in page2 as a reminder about what was going on.

Then, when the user does a click on page2 (making request3), the server can recover the information to see what was going on, and so produce page3 with the right context.

Dbsearch continuity illusion

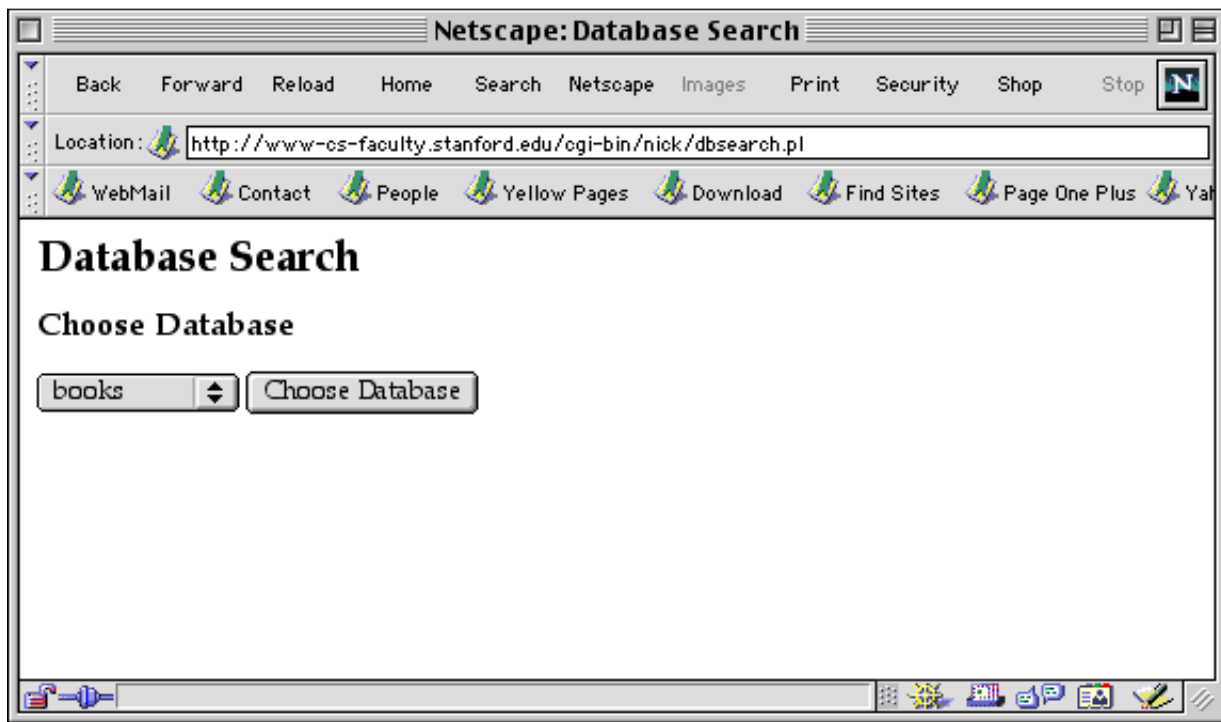
The user chooses a database on the first page

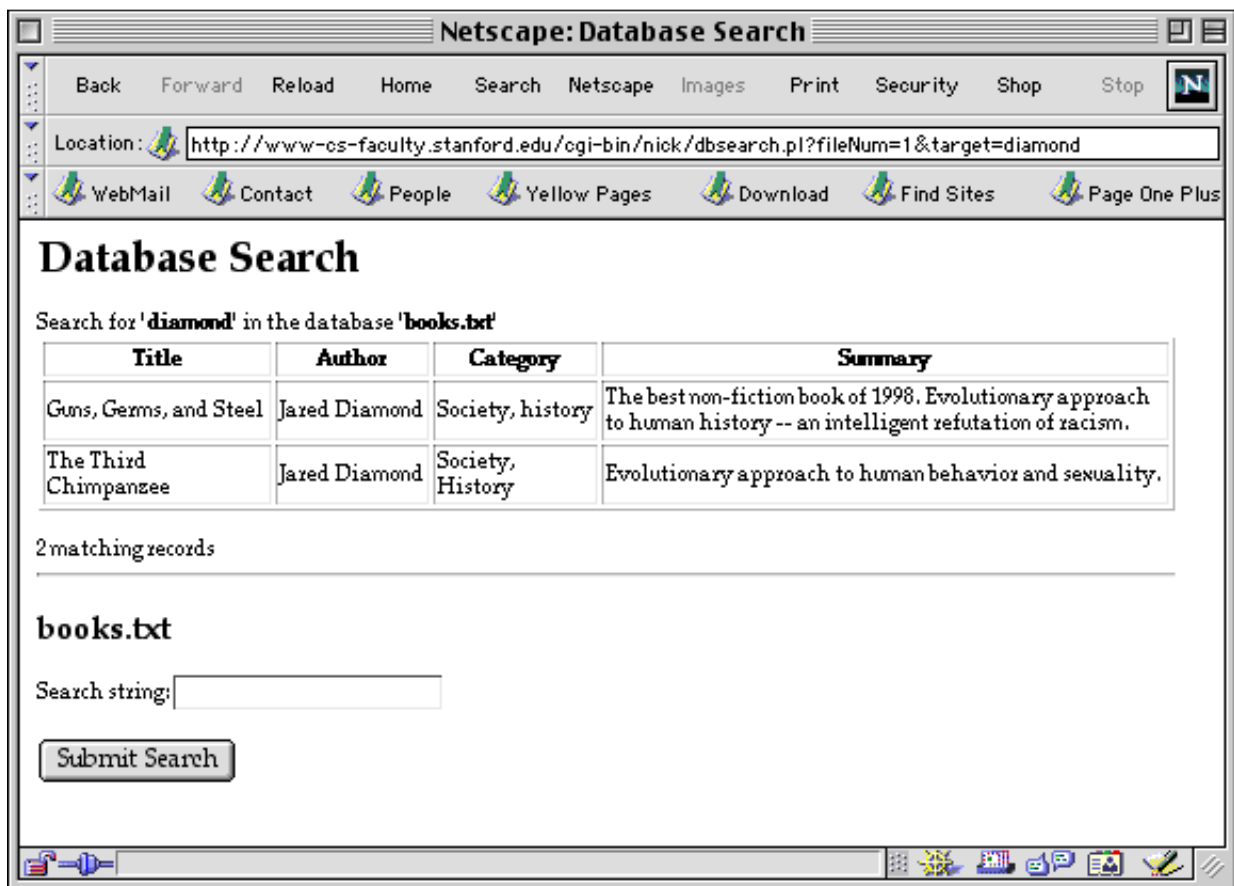
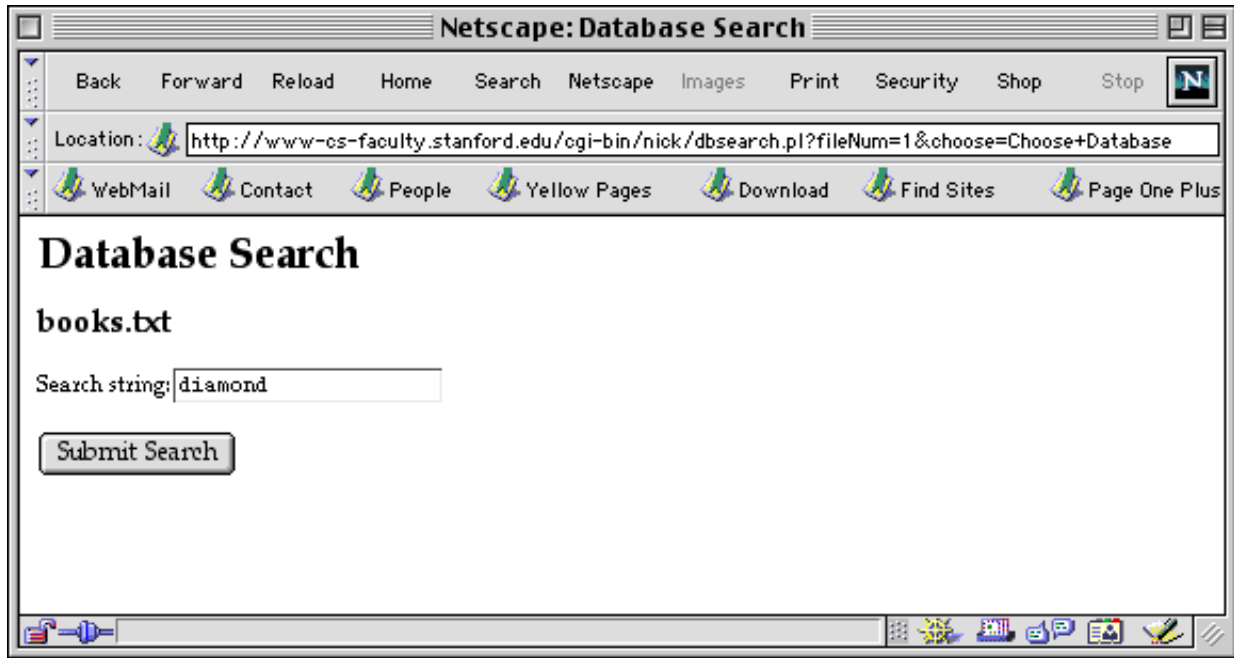
The later pages appear to remember this choice

Technique: hidden fields

Our first technique will be to use hidden fields in the HTML -- use them to store state in a form -- so the state is available later when the form is submitted.

Dbsearch example...





```

#!/usr/bin/perl -w
## dbsearch.pl

# 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.

use CGI;

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

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

##
## Start Output
##
$| = 1;          ## set STDOUT to be unbuffered

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

## HTML header
print $header;

$query = new CGI; ## this allocates the CGI module state

my(@filenames, $fileNum, $target, $filename);

# List of filenames we are willing to open. This is the simplest way
# to make sure that the client only opens the server files we really intend.
@filenames = ("movies.txt", "books.txt");
$fileNum = -1;
$filename = "";

if (defined($query->param('fileNum')) {
    $fileNum = $query->param('fileNum');
    $filename = $filenames[$fileNum];
}

# Security note: we do not open an arbitrary filename
# we got "off the net" since that could be misused to
# return files we don't want returned or to run processes
# on the server with names like "|rm *". That file opens

```

```

# can run processes is a Perl feature.
# Allternate security idea: allow the filename to come
# from the client, but only allow simple characters
# and not things like ".." or "/" which might allow them to
# move around in the directory structure...
# note: [^...] inverts the set
# if ($fileName =~ m"^[^\\w\d\\.\\-_|\\.\\.]" ) { error }
# Anything but letters, numbers, etc. is an error

```

```

##
## Choose Form
##
## (uses the "<<" trick to put the text in the file)
$chooseForm = <<EOT;
<h2>Choose Database</h2>
<p>
<form method=get>
<select name=fileNum>
<option value=0>movies
<option value=1>books
</select>
<input type=submit name=choose value="Choose Database">
</form>
EOT

```

```

##
## Search Form
##
## Uses a hidden field to remember its fileNum
$searchForm = <<EOT;
<h2>${filename}</h2>
<p>

<form method=get>
<input type=hidden name=fileNum value="${fileNum}">
Search string:<input type=text name=target size=20><br>

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

</form>
EOT

```

```

## If there is no fileNum, give 'em the chooser
if (!defined($query->param('fileNum')) ) {
    print $chooseForm;
    print $trailer;
    exit(0);
}

```

```

## Otherwise, if there is fileNum but no target
elseif (!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'");

    ## 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);
        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);
}

```

```

## TableRow and ReportError omitted

```