NAME

CGI::Push - Simple Interface to Server Push

SYNOPSIS

```
use CGI::Push qw(:standard);

do_push(-next_page=>\&next_page,
-last_page=>\&last_page,
-delay=>0.5);

sub next_page {
my($q,$counter) = @_;
return undef if $counter >= 10;
return start_html('Test'),
h1('Visible'),"\n",
"This page has been called ", strong($counter)," times",
end_html();
}

sub last_page {
my($q,$counter) = @_;
return start_html('Done'),
h1('Finished'),
strong($counter - 1),' iterations.',
end_html;
}
```

DESCRIPTION

CGI::Push is a subclass of the CGI object created by CGI.pm. It is specialized for server push operations, which allow you to create animated pages whose content changes at regular intervals.

You provide CGI::Push with a pointer to a subroutine that will draw one page. Every time your subroutine is called, it generates a new page. The contents of the page will be transmitted to the browser in such a way that it will replace what was there beforehand. The technique will work with HTML pages as well as with graphics files, allowing you to create animated GIFs.

Only Netscape Navigator supports server push. Internet Explorer browsers do not.

USING CGI::Push

CGI::Push adds one new method to the standard CGI suite, *do_push()*. When you call this method, you pass it a reference to a subroutine that is responsible for drawing each new page, an interval delay, and an optional subroutine for drawing the last page. Other optional parameters include most of those recognized by the CGI *header()* method.

You may call *do_push()* in the object oriented manner or not, as you prefer:

```
use CGI::Push;
$q = new CGI::Push;
$q->do_push(-next_page=>\&draw_a_page);

-or-

use CGI::Push qw(:standard);
do_push(-next_page=>\&draw_a_page);
```

Parameters are as follows:

-next_page

```
do_push(-next_page=>\&my_draw_routine);
```

This required parameter points to a reference to a subroutine responsible for drawing each new page. The subroutine should expect two parameters consisting of the CGI object and a counter indicating the number of times the subroutine has been called. It should return the contents of the page as an **array** of one or more items to print. It can return a false value (or an empty array) in order to abort the redrawing loop and print out the final page (if any)

```
sub my_draw_routine {
my($q,$counter) = @_;
return undef if $counter > 100;
return start_html('testing'),
h1('testing'),
"This page called $counter times";
}
```

You are of course free to refer to create and use global variables within your draw routine in order to achieve special effects.

-last_page
: This optional parameter points to a reference to the subroutine responsible for drawing the last page of the series. It is called after the -next_page routine returns a false value. The subroutine itself should have exactly the same calling conventions as the -next_page routine.

-type
: This optional parameter indicates the content type of each page. It defaults to ''text/html''. Normally the module assumes that each page is of a homogeneous MIME type. However if you provide either of the magic values ''heterogeneous'' or ''dynamic'' (the latter provided for the convenience of those who hate long parameter names), you can specify the MIME type — and other header fields — on a per-page basis. See ''heterogeneous pages'' for more details.

-delay
: This indicates the delay, in seconds, between frames. Smaller delays refresh the page faster. Fractional values are allowed.

  If not specified, -delay will default to 1 second

-cookie, -target, -expires, -nph
: These have the same meaning as the like-named parameters in *CGI::header()*.

  If not specified, -nph will default to 1 (as needed for many servers, see below).

Heterogeneous Pages

Ordinarily all pages displayed by CGI::Push share a common MIME type. However by providing a value of ''heterogeneous'' or ''dynamic'' in the *do_push()* -type parameter, you can specify the MIME type of each page on a case-by-case basis.

If you use this option, you will be responsible for producing the HTTP header for each page. Simply modify your draw routine to look like this:

```
sub my_draw_routine {
my($q,$counter) = @_;
return header('text/html'), # note we're producing the header here
start_html('testing'),
h1('testing'),
"This page called $counter times";
}
```

You can add any header fields that you like, but some (cookies and status fields included) may not be interpreted by the browser. One interesting effect is to display a series of pages, then, after the last page, to redirect the browser to a new URL. Because*r edirect()* does b<not> work, the easiest way is with a -refresh header field, as shown below:

```
sub my_draw_routine {
my($q,$counter) = @_;
return undef if $counter > 10;
return header('text/html'), # note we're producing the header here
start_html('testing'),
h1('testing'),
"This page called $counter times";
}

sub my_last_page {
       return header(-refresh=>'5; URL=http://somewhere.else/finished.html',
-type=>'text/html'),
start_html('Moved'),
h1('This is the last page'),
'Goodbye!'
hr,
end_html;
}
```

Changing the Page Delay on the Fly

If you would like to control the delay between pages on a page-by-page basis, call *push_delay()* from within your draw routine. *push_delay()* takes a single numeric argument representing the number of seconds you wish to delay after the current page is displayed and before displaying the next one. The delay may be fractional. Without parameters, *push_delay()* just returns the current delay.

INSTALLING CGI::Push SCRIPTS

CGI::Push SCRIPTS Server push scripts must be installed as no-parsed-header (NPH) scripts in order to work correctly on many servers. On Unix systems, this is most often accomplished by prefixing the script's name with "nph-". Recognition of NPH scripts happens automatically with WebSTAR and Microsoft IIS. Users of other servers should see their documentation for help.

Apache web server from version 1.3b2 on does not need server push scripts installed as NPH scripts: the -nph parameter to *do_push()* may be set to a false value to disable the extra headers needed by an NPH script.

AUTHOR INFORMATION

Address bug reports and comments to: lstein@cshl.org

BUGS

This section intentionally left blank.

SEE ALSO

CGI::Carp, CGI