

Wikiprint Book

Title: Trac with FastCGI

Subject: Ecopath Developer Site - TracFastCgi

Version: 2

Date: 2024-04-23 09:26:52

Table of Contents

Trac with FastCGI	3
Simple Apache configuration	3
Simple Cherokee Configuration	4
Simple Lighttpd Configuration	4
Simple LiteSpeed Configuration	7
Simple Nginx Configuration	8

Trac with FastCGI

Since version 0.9, Trac supports being run through the [?FastCGI](#) interface. Like [mod_python](#), this allows Trac to remain resident, and is faster than external CGI interfaces which must start a new process for each request. However, unlike `mod_python`, it is able to support [?SuEXEC](#). Additionally, it is supported by much wider variety of web servers.

Note for Windows: Trac's FCGI does not run under Windows, as Windows does not implement `Socket.fromfd`, which is used by `_fcgi.py`. If you want to connect to IIS, your choice may be [?AJP](#).

Simple Apache configuration

There are two FastCGI modules commonly available for Apache: `mod_fastcgi` and `mod_fcgid`. The `FastCgiIpcDir` and `FastCgiConfig` directives discussed below are `mod_fastcgi` directives; the `DefaultInitEnv` is a `mod_fcgid` directive.

For `mod_fastcgi`, add the following to an appropriate Apache configuration file:

```
# Enable fastcgi for .fcgi files
# (If you're using a distro package for mod_fcgi, something like
# this is probably already present)
<IfModule mod_fastcgi.c>
    AddHandler fastcgi-script .fcgi
    FastCgiIpcDir /var/lib/apache2/fastcgi
</IfModule>
LoadModule fastcgi_module /usr/lib/apache2/modules/mod_fastcgi.so
```

Setting `FastCgiIpcDir` is optional if the default is suitable. Note that the `LoadModule` line must be after the `IfModule` group.

Configure `ScriptAlias` or similar options as described in [TracCgi](#), but calling `trac.fcgi` instead of `trac.cgi`.

You can set up the `TRAC_ENV` as an overall default:

```
FastCgiConfig -initial-env TRAC_ENV=/path/to/env/trac
```

Or you can serve multiple Trac projects in a directory like:

```
FastCgiConfig -initial-env TRAC_ENV_PARENT_DIR=/parent/dir/of/projects
```

But neither of these will work for `mod_fcgid`. A similar but partial solution for `mod_fcgid` is:

```
DefaultInitEnv TRAC_ENV /path/to/env/trac/
```

But this cannot be used in `Directory` or `Location` context, which makes it difficult to support multiple projects.

A better method which works for both of these modules (and for [?lighttpd](#) and CGI as well), because it involves no server configuration settings for environment variables, is to set one of the variables in `trac.fcgi`, e.g.:

```
import os
os.environ['TRAC_ENV'] = "/path/to/projectenv"
```

or

```
import os
os.environ['TRAC_ENV_PARENT_DIR'] = "/path/to/project/parent/dir"
```

Using this method, different projects can be supported by using different `.fcgi` scripts with different `ScriptAliases`, copying and appropriately renaming `trac.fcgi` and adding the above code to create each such script.

See [?this fcgid example config](#) which uses a `ScriptAlias` directive with `trac.fcgi` with a trailing `/` like this:

```
ScriptAlias / /srv/tracsite/cgi-bin/trac.fcgi/
```

Simple Cherokee Configuration

The configuration on Cherokee's side is quite simple. You will only need to know that you can spawn Trac as an SCGI process. You can either start it manually, or better yet, automatically by letting Cherokee spawn the server whenever it is down. First set up an information source in cherokee-admin with a local interpreter.

```
Host:
localhost:4433

Interpreter:
/usr/bin/tracd ?single-env ?daemonize ?protocol=scgi ?hostname=localhost ?port=4433 /path/to/project/
```

If the port was not reachable, the interpreter command would be launched. Note that, in the definition of the information source, you will have to manually launch the spawner if you use a *Remote host* as *Information source* instead of a *Local interpreter*.

After doing this, we will just have to create a new rule managed by the SCGI handler to access Trac. It can be created in a new virtual server, trac.example.net for instance, and will only need two rules. The **default** one will use the SCGI handler associated to the previously created information source. The second rule will be there to serve the few static files needed to correctly display the Trac interface. Create it as *Directory rule* for */chrome/common* and just set it to the *Static files* handler and with a *Document root* that points to the appropriate files: */usr/share/trac/htdocs/*

Simple Lighttpd Configuration

The FastCGI front-end was developed primarily for use with alternative web servers, such as [?lighttpd](#).

lighttpd is a secure, fast, compliant and very flexible web-server that has been optimized for high-performance environments. It has a very low memory footprint compared to other web servers and takes care of CPU load.

For using trac.fcgi (prior to 0.11) / fcgi_frontend.py (0.11) with lighttpd add the following to your lighttpd.conf:

```
#var.fcgi_binary="/path/to/fcgi_frontend.py" # 0.11 if installed with easy_setup, it is inside the egg directory
var.fcgi_binary="/path/to/cgi-bin/trac.fcgi" # 0.10 name of prior fcgi executable
fastcgi.server = ( "/trac" =>

    ( "trac" =>
        ( "socket" => "/tmp/trac-fastcgi.sock",
          "bin-path" => fcgi_binary,
          "check-local" => "disable",
          "bin-environment" =>
            ( "TRAC_ENV" => "/path/to/projenv" )
        )
    )
)
```

Note that you will need to add a new entry to fastcgi.server for each separate Trac instance that you wish to run. Alternatively, you may use the TRAC_ENV_PARENT_DIR variable instead of TRAC_ENV as described above, and you may set one of the two in trac.fcgi instead of in lighttpd.conf using bin-environment (as in the section above on Apache configuration).

For using two projects with lighttpd add the following to your lighttpd.conf:

```
fastcgi.server = ( "/first" =>
    ( "first" =>
        ( "socket" => "/tmp/trac-fastcgi-first.sock",
          "bin-path" => fcgi_binary,
          "check-local" => "disable",
          "bin-environment" =>
            ( "TRAC_ENV" => "/path/to/projenv-first" )
        )
    ),
)
```

```

    "/second" =>
    ( "second" =>
    ( "socket" => "/tmp/trac-fastcgi-second.sock",
      "bin-path" => fcgi_binary,
      "check-local" => "disable",
      "bin-environment" =>
        ( "TRAC_ENV" => "/path/to/projenv-second" )
      )
    )
  )
)

```

Note that field values are different. If you prefer setting the environment variables in the `.fcgi` scripts, then copy/rename `trac.fcgi`, e.g., to `first.fcgi` and `second.fcgi`, and reference them in the above settings. Note that the above will result in different processes in any event, even if both are running from the same `trac.fcgi` script.

Note It's very important the order on which server.modules are loaded, if `mod_auth` is not loaded **BEFORE** `mod_fastcgi`, then the server will fail to authenticate the user.

For authentication you should enable `mod_auth` in `lighttpd.conf` 'server.modules', select `auth.backend` and `auth` rules:

```

server.modules          = (
...
  "mod_auth",
...
)

auth.backend            = "htpasswd"

# Separated password files for each project
# See "Conditional Configuration" in
# http://trac.lighttpd.net/trac/file/branches/lighttpd-merge-1.4.x/doc/configuration.txt

$HTTP["url"] =~ "^/first/" {
  auth.backend.htpasswd.userfile = "/path/to/projenv-first/htpasswd.htaccess"
}
$HTTP["url"] =~ "^/second/" {
  auth.backend.htpasswd.userfile = "/path/to/projenv-second/htpasswd.htaccess"
}

# Enable auth on trac URLs, see
# http://trac.lighttpd.net/trac/file/branches/lighttpd-merge-1.4.x/doc/authentication.txt

auth.require = ( "/first/login" =>
  ( "method"  => "basic",
    "realm"   => "First project",
    "require" => "valid-user"
  ),
  "/second/login" =>
  ( "method"  => "basic",
    "realm"   => "Second project",
    "require" => "valid-user"
  )
)

```

Note that `lighttpd` (I use version 1.4.3) stopped if password file doesn't exist.

Note that `lighttpd` doesn't support 'valid-user' in versions prior to 1.3.16.

Conditional configuration is also useful for mapping static resources, i.e. serving out images and CSS directly instead of through FastCGI:

```
# Aliasing functionality is needed
server.modules += ("mod_alias")

# Setup an alias for the static resources
alias.url = ("/trac/chrome/common" => "/usr/share/trac/htdocs")

# Use negative lookahead, matching all requests that ask for any resource under /trac, EXCEPT in
# /trac/chrome/common, and use FastCGI for those
$HTTP["url"] =~ "^/trac(?!/chrome/common)" {
# Even if you have other fastcgi.server declarations for applications other than Trac, do NOT use += here
fastcgi.server = ("/trac" =>
    ("trac" =>
        ("socket" => "/tmp/trac-fastcgi.sock",
            "bin-path" => fcgi_binary,
            "check-local" => "disable",
            "bin-environment" =>
                ("TRAC_ENV" => "/path/to/projenv")
            )
        )
    )
}
```

The technique can be easily adapted for use with multiple projects by creating aliases for each of them, and wrapping the fastcgi.server declarations inside conditional configuration blocks. Also there is another way to handle multiple projects and it's to use TRAC_ENV_PARENT_DIR instead of TRAC_ENV and use global auth, let's see an example:

```
# This is for handling multiple projects
alias.url = ( "/trac/" => "/path/to/trac/htdocs/" )

fastcgi.server += ("/projects" =>
    ("trac" =>
        (
            "socket" => "/tmp/trac.sock",
            "bin-path" => fcgi_binary,
            "check-local" => "disable",
            "bin-environment" =>
                ("TRAC_ENV_PARENT_DIR" => "/path/to/parent/dir/of/projects/" )
            )
        )
    )
)

#And here starts the global auth configuration
auth.backend = "htpasswd"
auth.backend.htpasswd.userfile = "/path/to/unique/htpasswd/file/trac.htpasswd"
$HTTP["url"] =~ "^/projects/.*/login$" {
    auth.require = ("/" =>
        (
            "method" => "basic",
            "realm" => "trac",
            "require" => "valid-user"
        )
    )
}
```

Changing date/time format also supported by lighttpd over environment variable LC_TIME

```
fastcgi.server = ("/trac" =>
    ("trac" =>
        ("socket" => "/tmp/trac-fastcgi.sock",
            "bin-path" => fcgi_binary,
            "check-local" => "disable",
```

```

        "bin-environment" =>
        ( "TRAC_ENV" => "/path/to/projenv",
          "LC_TIME" => "ru_RU" )
      )
    )
  )

```

For details about languages specification see [?TracFaq](#) question 2.13.

Other important information like [?this updated TracInstall page](#), [and this](#) are useful for non-fastcgi specific installation aspects.

If you use trac-0.9, read [?about small bug](#)

Relaunch lighttpd, and browse to `http://yourhost.example.org/trac` to access Trac.

Note about running lighttpd with reduced permissions:

If nothing else helps and trac.fcgi doesn't start with lighttpd settings `server.username = "www-data"`, `server.groupname = "www-data"`, then in the bin-environment section set `PYTHON_EGG_CACHE` to the home directory of www-data or some other directory accessible to this account for writing.

Simple LiteSpeed Configuration

The FastCGI front-end was developed primarily for use with alternative webserver, such as [?LiteSpeed](#).

LiteSpeed web server is an event-driven asynchronous Apache replacement designed from the ground-up to be secure, scalable, and operate with minimal resources. LiteSpeed can operate directly from an Apache config file and is targeted for business-critical environments.

Setup

- 1) Please make sure you have first have a working install of a Trac project. Test install with [?tracd?](#) first.
- 2) Create a Virtual Host for this setup. From now on we will refer to this vhost as [TracVhost?](#). For this tutorial we will be assuming that your trac project will be accessible via:

```
http://yourdomain.com/trac/
```

- 3) Go to [?TracVhost?](#) ? External Apps? tab and create a new ?External Application?.

```

Name: MyTracFCGI
Address: uds://tmp/lshhttpd/mytracfcgi.sock
Max Connections: 10
Environment: TRAC_ENV=/fullpath/to/mytracproject/ <--- path to root folder of trac project
Initial Request Timeout (secs): 30
Retry Timeout (secs): 0
Persistent Connection      Yes
Connection Keepalive Timeout: 30
Response Bufferring: No
Auto Start: Yes
Command: /usr/share/trac/cgi-bin/trac.fcgi <--- path to trac.fcgi
Back Log: 50
Instances: 10

```

- 4) Optional. If you need to use htpasswd based authentication. Go to [?TracVhost?](#) ? Security? tab and create a new security ?Realm?.

```

DB Type: Password File
Realm Name: MyTracUserDB          <--- any name you wish and referenced later
User DB Location: /fullpath/to/htpasswd <--- path to your htpasswd file

```

If you don't have a htpasswd file or don't know how to create the entries within one, go to <http://sherylcanter.com/encrypt.php>, to generate the user:password combos.

5) Go to [PythonVhost?](#) ? Contexts? and create a new ?FCGI Context?.

```
URI: /trac/ <--- URI path to bind to python fcgi app we created
Fast CGI App: [VHost Level] MyTractFCGI <--- select the trac fcgi extapp we just created
Realm: TracUserDB <--- only if (4) is set. select realm created in (4)
```

6) Modify /fullpath/to/mytracproject/conf/trac.ini

```
#find/set base_url, url, and link variables
base_url = http://yourdomain.com/trac/ <--- base url to generate correct links to
url = http://yourdomain.com/trac/ <--- link of project
link = http://yourdomain.com/trac/ <--- link of graphic logo
```

7) Restart LiteSpeed, ?lswwwctrl restart?, and access your new Trac project at:

```
http://yourdomain.com/trac/
```

Simple Nginx Configuration

1) Nginx configuration snippet - confirmed to work on 0.6.32

```
server {
    listen      10.9.8.7:443;
    server_name trac.example;

    ssl         on;
    ssl_certificate      /etc/ssl/trac.example.crt;
    ssl_certificate_key  /etc/ssl/trac.example.key;

    ssl_session_timeout 5m;

    ssl_protocols  SSLv2 SSLv3 TLSv1;
    ssl_ciphers    ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;
    ssl_prefer_server_ciphers  on;

    # (Or ``^/some/prefix/(.*)``.
    if ($uri ~ ^/(.*)) {
        set $path_info /$1;
    }

    # You can copy this whole location to ``location [/some/prefix]/login``
    # and remove the auth entries below if you want Trac to enforce
    # authorization where appropriate instead of needing to authenticate
    # for accessing the whole site.
    # (Or ``location /some/prefix``.)
    location / {
        auth_basic          "trac realm";
        auth_basic_user_file /home/trac/htpasswd;

        # socket address
        fastcgi_pass         unix:/home/trac/run/instance.sock;

        # python - wsgi specific
        fastcgi_param        HTTPS on;

        ## WSGI REQUIRED VARIABLES
        # WSGI application name - trac instance prefix.
        # (Or ``fastcgi_param SCRIPT_NAME /some/prefix``.)
        fastcgi_param        SCRIPT_NAME      "";
        fastcgi_param        PATH_INFO        $path_info;
```



```

    ## WSGI NEEDED VARIABLES - trac warns about them
    fastcgi_param  REQUEST_METHOD    $request_method;
    fastcgi_param  SERVER_NAME       $server_name;
    fastcgi_param  SERVER_PORT       $server_port;
    fastcgi_param  SERVER_PROTOCOL   $server_protocol;

    # for authentication to work
    fastcgi_param  AUTH_USER          $remote_user;
    fastcgi_param  REMOTE_USER        $remote_user;
}
}

```

2) Modified trac.fcgi:

```

#!/usr/bin/env python
import os
sockaddr = '/home/trac/run/instance.sock'
os.environ['TRAC_ENV'] = '/home/trac/instance'

try:
    from trac.web.main import dispatch_request
    import trac.web._fcgi

    fcgiserv = trac.web._fcgi.WSGIServer(dispatch_request,
        bindAddress = sockaddr, umask = 7)
    fcgiserv.run()

except SystemExit:
    raise
except Exception, e:
    print 'Content-Type: text/plain\r\n\r\n',
    print 'Oops...'
    print
    print 'Trac detected an internal error:'
    print
    print e
    print
    import traceback
    import StringIO
    tb = StringIO.StringIO()
    traceback.print_exc(file=tb)
    print tb.getvalue()

```

3) reload nginx and launch trac.fcgi like that:

```
trac@trac.example ~ $ ./trac-standalone-fcgi.py
```

The above assumes that:

- There is a user named 'trac' for running trac instances and keeping trac environments in its home directory.
- /home/trac/instance contains a trac environment
- /home/trac/htpasswd contains authentication information
- /home/trac/run is owned by the same group the nginx runs under
 - and if your system is Linux the /home/trac/run has setgid bit set (chmod g+s run)
 - and patch from ticket #T7239 is applied, or you'll have to fix the socket file permissions every time

Unfortunately nginx does not support variable expansion in fastcgi_pass directive. Thus it is not possible to serve multiple trac instances from one server block.

If you worry enough about security, run trac instances under separate users.

Another way to run trac as a FCGI external application is offered in ticket #T6224

See also: [TracGuide](#), [TracInstall](#), [ModWSGI](#), [CGI](#), [ModPython](#), [?TracNginxRecipe](#)