Wikiprint Book

Title: Trac を FastCGI で使用する

Subject: SilverFrost - TracFastCgi

Version: 3

Date: 12/20/25 13:32:18

SilverFrost 目次

Trac を FastCGI で使用する

FastCGI インターフェースを使用すると mod_python や mod_wsgi と同様に Trac を常駐させることができます。リクエスト毎に新しいプロセスを開始しなければならない CGI インターフェースよりも処理速度は速くなります。加えて、多種多様の Web サーバでサポートされています。

Note: mod_python とは異なり、FastCGI では Apache SuEXEC をサポートしています。つまり、Web サーバを実行しているユーザとは、異なる権限でプログラムを実行することができます。(mod_wsgi でサポートしている WSGIDaemonProcess におけるユーザ / グループと同じ効果を得ることができます)

Windows 向け: Trac の FastCGI は Windows 環境下で動作しません。これは、_fcgi.py で用いられる Socket.fromfd が Windows では実行されないからです。もし IIS へ接続したい場合は、 AJP/ ISAPI に挑戦してみて下さい。

単純な Apache の設定

Apache で利用可能な FastCGI モジュールは 2 種類あります: mod_fastcgi と mod_fcgid (推奨) です。後者の方がよりメンテナンスされています。

次の項目では、FCGI の特定のセットアップに焦点を合わせています。Apache の認証の設定方法については TracModWSGI を参照してください。

CGI モジュールを使用する場合は注意してください。また、必ず Web サーバに cgi-bin フォルダの実行権限を持たせてください。 FastCGI では詳細な権限に関するエラーが出力されますが、mod_fcgid ではこの処理が行われていない場合には、不明瞭なエラーしか出力されません。(Connection reset by peer: mod_fcgid: error reading data from FastCGI server の様な)

mod_fastcgi でのセットアップ

mod_fastcgi では FastCgiIpcDir と FastCgiConfig ディレクティブを使用して Apache の設定ファイルに設定を行います:

```
# 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
```

デフォルトの設定に問題がなければ、 FastCgiIpcDir の設定は必須ではありません。 LoadModule の行は IfModule グループの後になければいけないことに注意して下さい。

ScriptAlias もしくは TracCgi で説明されている類似のオプションを設定しますが、 trac.cgi の代わりに trac.fcgi を指定してください。

すべてデフォルトで TRAC_ENV をセットアップするつもりならば、Apache の設定ファイルに以下を追加します (FastCgiIpcDir より下に):

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

あるいは、以下を追加することで複数の Trac プロジェクトを提供できます:

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

mod_fcgid でのセットアップ

ScriptAlias を設定します。 (詳細は TracCgi を参照してください)、 ただし trac.cgi ■■■■■ trac.fcgi` を呼び出します。 Note: 最後のスラッシュを忘れずに。とても重要です。

```
ScriptAlias /trac /path/to/www/trac/cgi-bin/trac.fcgi/
```

mod_fcgid で Trac environment を設定するには DefaultInitEnv ディレクティブを使用します。このディレクティブは Direcotry や Location コンテキストで使用できないので、複数のプロジェクトを設定する場合、 以下に記述する Environment を設定する別の方法を試してください。

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

Environment を設定する別の方法

Trac environment へのパスを設定するための、より適した方法は、パスを `trac.fcgi`
スクリプト自体に書き込むことです。これによってサーバの環境変数を 設定する必要がなくなり、どちらの FastCgi モジュール (および lighttpd や CGI など) でも、動作するようになります:

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

または

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

プロジェクトごとの `ScriptAlias` **と** `.fcgi` **スクリプトを設定すれば、 起動スクリプトに** `TRAC_ENV`
等を設定する方法を使用して複数のプロジェクトに対応することができます。

この fcgid 設定例 の通り、 ScriptAlias ディレクティブでは末尾の / も含めて設定してください:

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

Cherokee の簡単な設定

Cherokee 側の設定はとても簡単です。 Trac を SCGI プロセスとして起動できるかどうかのみ知っている必要があります。 Cherokee が起動していないときにはいつでも、 Cherokee が Trac を切り離せるようにすることによって、 Trac を手動で起動することも、いっそのこと自動的に起動することもできます。 最初に、ローカルのインタプリタである cherokee-admin で information source を設定します。

```
Host:
localhost:4433

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

もしそのポート番号に到達できなければ、インタプリタコマンドは起動されたことになります。information source の定義において、ローカルインタプリタ の代わりに、 リモートホスト を information source として使用するならば、 spawner を手動で起動する必要があることを覚えておいてください。

そして、 Trac にアクセスするために SCGI ハンドラによって管理される新しいルールを作成しなければなりません。このルールは例えば、trac.example.net という新しい仮想サーバ内で作成し、 2 つのルールのみを必要とします。 デフォルト ルールは以前に作成された information source に関する SCGI ハンドラで使用されます。 2 つ目のルールは Trac のインタフェースを正しく表示するために必要ないくつかの静的ファイルを扱うために設定します。ルールを /common の ディレクトリルール として作成し、 静的ファイル のハンドラを適切なファイルをポイントする ドキュメントルート と共に設定するだけです: $TRAC_LOCAL/htdocs/ ($TRAC_LOCAL はユーザ又はローカル Trac リソース を置いたシステム管理者により定義されたディレクトリです)

Note:
tracd プロセスが起動しない場合や cherokee が 503 エラーページを表示する場合、 python-flup がインストールされていない可能性があります。Python-flup は SCGI の能力を trac に与える従属関係にあります。 debian 基本システムからインストールできます:

```
sudo apt-get install python-flup
```

Lighttpd の簡単な設定

FastCGI フロントエンドは最初 lighttpd のような、 Apache 以外の Web サーバのために開発されました。

lighttpd はセキュアで高速で、規格に準拠したとても柔軟な Web サーバで、高いパフォーマンスの環境で最適化されます。 他の Web サーバに比べて CPU や、メモリの占有率がとても少ないです。

trac.fcgi(0.11 以前) / fcgi_frontend.py (0.11) を lighttpd で使用するためには、 lighttpd.conf に以下の行を追加します:

```
#var.fcgi_binary="/usr/bin/python /path/to/fcgi_frontend.py" # 0.11 if installed with easy_setup, it is inside the egg dir
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")
                    )
                  )
                )
```

動かしたい Trac のインスタンス毎に fastcgi.server のエントリを追加する必要があります。別の方法として、上記の TRAC_ENV の代わりに TRAC_ENV_PARENT_DIR を使用でき、 lighttpd.conf に設定する代わりに trac.fcgi ファイルに bin-environment (上記の Apache の設定に書かれています) の2つのうちのどちらかを設定します。

Note: lighttpd には 'SCRIPT_NAME' と 'PATH_INFO' に関するバグがあります。例として、 fastcgi.server の uri が '/trac' とするところを '/' としてしまいます。 (本家チケット 2418 参照) このバグは lighttpd 1.4.23 以降で修正されています。 fastcgi.server のパラメータとして、 "fix-root-scriptname" => "enable" を追加する必要があるでしょう。

lighttpd で2つのプロジェクトを動かすには、 lighttpd.conf に以下の設定を追加します:

```
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: 各フィールドの値が異なることに注意して下さい。もし、 .fcgi スクリプト内の環境変数を設定する方が好ましいならば、 trac.fcgi スクリプトを 例えば、 first.fcgi や second.fcgi というようにコピー / リネームして、上記設定の中でこれらのスクリプトを参照するようにしてください。 両方のプロジェクトが同じ trac.fcgi スクリプトから起動しているとしても、 異なるプロセスになることに注意して下さい。

Note: server.modules をロードする順番はとても重要です。もし、 mod_auth が mod_fastcgi より 先に mod_auth がロードされない設定になっていない場合、サーバはユーザ認証に失敗します。

認証のために lighttpd.conf の 'server.modules' 中で mod_auth を有効にして、 auth.backend と認証方法を選択して下さい:

```
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: パスワードファイルがない場合、 lighttpd (確認したバージョンは 1.4.3) が停止するので注意して下さい。

Note: バージョン 1.3.16 以前では lighttpd は 'valid-user' をサポートしていないので注意してください。

条件付の設定は静的リソースをマッピングするときに便利です。例として FastCGI を経由せずに直接イメージファイルや CSS を参照するときなどです:

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

# Set up 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")
                    )
                  )
                 )
}
```

複数のプロジェクトのそれぞれにエイリアスを作れば、複数のプロジェクトを動かすのは技術的には簡単です。 fastcgi.server を条件ブロックの中で宣言しラッピングします。 複数のプロジェクトをハンドルするもう一つの方法があります。 TRAC_ENV_PARENT_DIR を TRAC_ENV の代わりに使用し、グローバルの認証機構を使用します。サンプルを見てみましょう:

```
#  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/htpassword/file/trac.htpasswd"
 $HTTP["url"] =~ "^/projects/.*/login$" {
   auth.require = ("/" =>
                    (
                      "method"  => "basic",
                      "realm"   => "trac",
                      "require" => "valid-user"
                    )
                  )
 }
```

lighttpd では環境変数の LC_TIME を上書きして、日付/時間のフォーマットを変更することも出来ます。

```
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")
                    )
                  )
                )
```

使用言語指定の詳細については TracFaq の 2.13 の質問を参照して下さい。

静的なリソースのマッピングに関する事 の様な他の重要な情報は、fastcgi 以外でもインストールの詳細をつかむのに有効です。]

lighttpd を再起動し、ブラウザに `http://yourhost.example.org/trac` を入力して、 Trac にアクセスして下さい。

制限された権限で lighttpd を起動するにあたって気をつけること:

もし、 trac.fcgi が lighttpd の設定で `server.username = "www-data"` や `server.groupname = "www-data"` を設定しても起動せずどうしようもないときは、 `bin-environment` セクションの `PYTHON_EGG_CACHE` を `www-data` のホームディレクトリまたは `www-data` アカウントで書き込みが可能なディレクトリに設定して下さい。 (訳注: debian 系 Linux に限定した話だと思われます。 `www-data` は lighttpd を起動するユーザに適宜読み替えてください。)

LiteSpeed の簡単な設定

FastCGI フロントエンドは最初 LiteSpeed のような、 Apache 以外の Web サーバのために開発されました。

LiteSpeed Web サーバはイベント駆動、非同期型であり、 Apache に代わるものとしてセキュアで拡張可能になるようにゼロからデザインされています。そして、最低限のリソースで操作できます。 LiteSpeed は Apache の設定ファイルから直接操作でき、ビジネスに不可欠な環境をターゲットにしています。

1. 最初に Trac プロジェクトをインストールして動作することを確認して下さい。最初のインストールでは、 "tracd" を使用します
1. このセットアップでは仮想ホストを作成します。以下、この仮想ホストのことを TracVhost と呼びます。このチュートリアルで、先ほど作ったプロジェクトが以下の URL 経由でアクセスできると仮定します:

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

1. "TracVhost → External Apps" タブへ移動し、新しい "External Application" を作成します

```
Name: MyTracFCGI
Address: uds://tmp/lshttpd/mytracfcgi.sock
Max Connections: 10
Environment: TRAC_ENV=/fullpathto/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
```

1. (非必須) htpasswd ベースの認証を使用するならば、 "TracVhost → Security" タブへ移動し、新しいセキュリティ "Realm" を作成することができます

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

もし、 htpasswd ファイルを持っていない、もしくは作り方を知らない場合は、 http://sherylcanter.com/encrypt.php にアクセスし、ユーザ名:パスワード の一対を生成して下さい。

1. "PythonVhost → Contexts" へ移動し、新しい "FCGI Context" を作成します

```
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)
```

1. /fullpathto/mytracproject/conf/trac.ini を修正します

```
#find/set base_rul, 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
```

1. LiteSpeed を "lswsctrl restart" で再起動し、新しい Trac プロジェクトに以下の URL でアクセスします:

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

Nginx 簡単な設定

Nginx は FastCGI プロセスを生成することはできませんが、伝達することができます。そのため、 Trac を独立して FastCGI サーバを開始する必要があります。

1. 基本認証を行う Nginx の設定 - 0.6.32 で動作することを確認しました

```
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;
   }

   # it makes sense to serve static resources through Nginx
   location /chrome/ {
        alias /home/trac/instance/static/htdocs/;
   }

   # 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;
       fastcgi_param  QUERY_STRING       $query_string;

       # For Nginx authentication to work - do not forget to comment these
       # lines if not using Nginx for authentication
       fastcgi_param  AUTH_USER          $remote_user;
       fastcgi_param  REMOTE_USER        $remote_user;

       # for ip to work
       fastcgi_param REMOTE_ADDR         $remote_addr;

       # For attchments to work
```

```
            fastcgi_param    CONTENT_TYPE     $content_type;
            fastcgi_param    CONTENT_LENGTH   $content_length;
        }
    }
```

1. trac.fcgi の変更:

```
#!/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()
```

1. nginx をリロードし、 trac.fcgi をこのように起動します:

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

上記設定は以下の条件だと仮定します:

- trac のインスタンスを実行するためのユーザ名を 'trac' とします。ホームディレクトリに trac Environment をおきます
- Trac environment は /home/trac/instance に配置します
- /home/trac/htpasswd に認証情報が含まれています
  /home/trac/run は nginx を起動しているグループが所有しています
  - Linux を使用しているならば、 /home/trac/run に (chmod g+s run) を設定します
  - 本家チケット 7239 のパッチを適用し、ソケットファイルのパーミッションをそのつど修正しなければいけません

残念ですが、 nginx は fastcgi_pass ディレクティブ内の変数展開をサポートしていません。 したがって、 1 つのサーバーブロックから複数の trac インスタンスを起動することができません。

セキュリティ面で不安があるならば、 各 Trac インスタンスを別々のユーザで起動してください。

Trac を FCGI の外部アプリケーションとして起動するもう一つの方法は、 本家チケット 6224 を参照して下さい。

See also: TracGuide, TracInstall, ModWSGI, CGI, ModPython, TracNginxRecipe