Note: このページのドキュメントは Trac 1.0 に対応しています。それ以前のバージョンについては 0.12/TracStandalone? を参照してください。

Tracd

Tracd は軽量なスタンドアロンの Trac web サーバです。 Tracd は様々な場面で使用することができます。テストや開発用のサーバからロードバランサとして使用されているもう一つのウェブサーバの後段に複数のプロセスを配置

利点

- 依存性が低い: apache その他 web サーバをインストールする必要がありません
- 速い: mod_python バージョンと同じくらい速いでしょう。 (CGI よりはずっと速い)。そして、 バージョン 0.12 以降では、デフォルトで HTTP のバージョン 1.1 が使えるようになりました
- 自動リロード: 開発のために、 Tracd は auto_reload モードを使用しています。そのため、コード (Trac 自身またはプラグインのコード) を更新したときに、自動的にサーバが再起動します

欠点

- 機能が少ない: Tracd に実装されている web サーバはとてもシンプルで、 Apache httpd のように拡張性のある設定ができません
- ネイティブで HTTPS に対応しない: 代わりに sslwrap または stunnel -- tracd と stunnel を使うためのチュートリアル または Apache の mod_proxy を使用します

使用例

単一のプロジェクトをポート 8080 でホストします。 (http://localhost:8080/)

```
$ tracd -p 8080 /path/to/project
```

厳密に言うと、この状態では Trac は localhost のみ ではなく、ネットワーク越しの全員からアクセス可能になっています。 --hostname オプションを使用すると接続元を制限できます。

```
$ tracd --hostname=localhost -p 8080 /path/to/project
```

複数のプロジェクトをホストする場合はこうです (http://localhost:8080/project1/ と http://localhost:8080/project2/)

```
$ tracd -p 8080 /path/to/project1 /path/to/project2
```

Trac は異なるプロジェクト間での URL の一意性を保つために、パスの一番最後の文字列 (訳注: basename) を使用するため、プロジェクト間でパスの一番最後の部分を同じにすることは出来ません。 もし、 /project1/path/to と /project2/path/to を同時に指定した場合、二つ目のプロジェクトだけしか見えなくなります。

複数のプロジェクトを動かすもう一つの方法は、 -e オプションで親ディレクトリを指定し、サブディレクトリに TracEnvironment を配置します。上記の例は以下のように書き換えられます:

```
$ tracd -p 8080 -e /path/to
```

Windows でサーバを終了するには必ず CTRL-BREAK を使用してください。 -- CTRL-C を使用すると Python のプロセスがバックグラウンドで起動したままになるでしょう。

Windows サービスとしてインストールする

オプション 1

Windows のサービスとしてインストールするには、 SRVANY ユーティリティを入手し起動します:

```
 C:\path\to\instsrv.exe tracd C:\path\to\srvany.exe
reg add HKLM\SYSTEM\CurrentControlSet\Services\tracd\Parameters /v Application /d "\"C:\path\to\python.exe\" \"C:\path\to\
net start tracd
```

tracd.exe は使用 しない で下さい。代わりに python.exe を直接登録し、引数に tracd-script.py を使用して下さい。 tracd.exe を使用してしまうと、 python プロセスが SRVANY の制御下ではなくなってしまうため、 net stop tracd を使用しても python プロセスが残留してしまいます。

Windows の起動時に tracd を自動起動させることもできます:

```
sc config tracd start= auto
```

空白には意味があります。そのまま入力して下さい。

一度、Windows サービスがインストールされると、上記の reg add コマンドを使用するよりも、レジストリエディタを起動するほうが簡単かもしれません。指定のパスへの移動:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\tracd\Parameters

3 つの文字列のパラメータが提供されています:

AppDirectory	C: / Python26 /
Application	python.exe
AppParameters	scripts / tracd-script.py -p 8080 ...

Note: もし AppDirectory が上記のように設定されているならば、実行ファイル と スクリプトのパス、パラメータの値は、設定したフォルダへの相対パスになります。ここ一点に絞って言うと、これにより Python をアップデートするときに変更箇所が限定されるため少しだけ楽になります。 (ドキュメントでは、 /full/path/to/htpasswd とフルパス指定するよう書いてあるかもしれませんが、これは、 .htpasswd ファイルについても同様のことが言えます。このファイルを Python ディレクトリ配下に配置したいと思わないかもしれません。)

Windows 7 ユーザは、 srvany.exe は使用できないかもしれません。代わりに WINSERV ユーティリティを使用して、起動してください:

```
"C:\path\to\winserv.exe" install tracd -displayname "tracd" -start auto "C:\path\to\python.exe" c:\path\to\python\scripts\

net start tracd
```

オプション 2

Trac Hacks より WindowsServiceScript を入手して下さい。 Trac のサービスを インストール、削除、開始、停止などできます。

オプション 3

cygwin の cygrunsrv.exe を使用することもできます:

```
$ cygrunsrv --install tracd --path /cygdrive/c/Python27/Scripts/tracd.exe --args '--port 8000 --env-parent-dir E:\IssueTra
$ net start tracd
```

認証を使用する

Tracd は基本認証とダイジェスト認証の両方に対応しています。ダイジェスト認証の方がより安全です。以降の例ではダイジェスト認証を使用しています。基本認証を使用 --auth を --basic-auth に置き換えて下さい。

認証を使用する際の一般的なフォーマット:

```
$ tracd -p port --auth="base_project_dir,password_file_path,realm" project_path
```

オプションについて:

base_project_dir: 下記の通りプロジェクトのベースのディレクトリを特定する:

- 複数のプロジェクトを立てているとき: project_path への 相対パス
- 1 つのみプロジェクトを立てているとき (-s オプション使用): プロジェクトのディレクトリの名前

絶対パスを使用しないで下さい。 Note: このパラメータは、 Windows の環境であっても大文字小文字を区別します。

- password_file_path: パスワードファイルへのパス
- realm: realm の名前 (なんでも指定できます)
- project_path: プロジェクトへのパス
- --auth 上記の例では、ダイジェスト認証を使用しています。基本認証を使用する際は `--auth` を `--basic-auth` に置き換えてください。基本認証は "realm" を必要としませんが、コマンドとしては、最後の引用句に空のレルム名前が直接続くことになるので、2つ目のコンマは必要になります

例:

```
$ tracd -p 8080 \
 --auth="project1,/path/to/passwordfile,mycompany.com" /path/to/project1
```

もちろん、パスワードファイルは 1 つ以上のプロジェクトで共有することができます。

```
$ tracd -p 8080 \
 --auth="project1,/path/to/passwordfile,mycompany.com" \
 --auth="project2,/path/to/passwordfile,mycompany.com" \
 /path/to/project1 /path/to/project2
```

パスワードファイルを共有するもう一つの方法として、プロジェクトの名前を指定するところで、 "*" を使用します:

```
$ tracd -p 8080 \
 --auth="*,/path/to/users.htdigest,mycompany.com" \
 /path/to/project1 /path/to/project2
```

基本認証: htpasswd パスワードを使用する

このセクションでは、 `tracd` と Apache の .htpasswd ファイルの使用方法について記述します。

> Note: htpasswd のフォーマットを解読するために、(少なくとも Python 2.6 は) fcrypt パッケージをインストールする必要があります。 Trac のソースコードでは、まず `import crypt` を試みますが、 Python 2.6 にそのようなパッケージはありません。 SHA-1 パスワードのみ、このモジュールなしで対応します。(Trac 1.0 から)

Apache の `htpasswd` コマンドを使用して、 .htpasswd ファイルを作成します。 (Apache を使用せずにこれらのファイルを作成する方法については 下記 を参照して下さい):

```
$ sudo htpasswd -c /path/to/env/.htpasswd username
```

そしてユーザを追加します:

```
$ sudo htpasswd /path/to/env/.htpasswd username2
```

そして、 `tracd` をこのように起動します:

```
$ tracd -p 8080 --basic-auth="projectdirname,/fullpath/environmentname/.htpasswd,realmname" /fullpath/environmentname
```

例:

```
$ tracd -p 8080 --basic-auth="testenv,/srv/tracenv/testenv/.htpasswd,My Test Env" /srv/tracenv/testenv
```

Note: いくつかのプラットフォーム (OpenBSD) では、htpasswd に "-m" をパラメータで渡す必要があるかもしれません。

ダイジェスト認証: htdigest パスワードファイルの設定方法

もし、 Apache がインストールされているなら、パスワードファイルを生成するのに、 htdigest コマンドを使用することができます。 'htdigest' とタイプして使用方法を見るか、詳細な使用方法を見るために Apache のマニュアルの このページ を読んでください。ユーザを作成するたびに、パスワードを入力するように求められます。パスワードファイルの名前には好きな名前をつけることができますが、`users.htdigest` というような名前にしておけば、ファイルに何が含まれているかを覚えておけるでしょう。このファイルは <projectname>/conf フォルダに trac.ini ファイルと一緒に置いておくとよいでしょう。

引数 `--auth` なしで tracd をスタートできることに注意して下さい。ただし、 ログイン (英語版では Login) リンクをクリックするとエラーになります。

Apache 以外の環境でパスワードを生成する

基本認証は online HTTP Password generator を用いて完成させることができます。これは `SHA-1` もサポートしています。生成した password-hash をシステムの .htpasswd ファイルにコピーします。Windows 版の Python は htpasswd のデフォルトのハッシュタイプである "crypt" モジュールに対応していないので注意してください。MD5 パスワードハッシュには対応しているため、MD5 を使用するとよいでしょう。

簡単な Python スクリプトで digest 認証のパスワードファイルを生成できます:

```
from optparse import OptionParser
# The md5 module is deprecated in Python 2.5
try:
   from hashlib import md5
except ImportError:
   from md5 import md5
realm = 'trac'

# build the options
usage = "usage: %prog [options]"
parser = OptionParser(usage=usage)
parser.add_option("-u", "--username",action="store", dest="username", type = "string",
                  help="the username for whom to generate a password")
parser.add_option("-p", "--password",action="store", dest="password", type = "string",
                  help="the password to use")
parser.add_option("-r", "--realm",action="store", dest="realm", type = "string",
                  help="the realm in which to create the digest")
(options, args) = parser.parse_args()

# check options
if (options.username is None) or (options.password is None):
  parser.error("You must supply both the username and password")
if (options.realm is not None):
  realm = options.realm

# Generate the string to enter into the htdigest file
kd = lambda x: md5(':'.join(x)).hexdigest()
print ':'.join((options.username, realm, kd([options.username, realm, options.password])))
```

Note: 上記のスクリプトを使用する場合、 `--auth` の引数に trac を指定し、レルムを設定しなければなりません。使用例 (上記スクリプトを trac-digest.py として保存したとします):

```
 $ python trac-digest.py -u username -p password >> c:\digest.txt
$ tracd --port 8000 --auth=proj_name,c:\digest.txt,trac c:\path\to\proj_name
```

md5sum を使用する

`md5sum` ユーティリティを使用するとダイジェスト認証のパスワードファイルを作成することができます:

```
user=
realm=
password=
path_to_file=
echo ${user}:${realm}:$(printf "${user}:${realm}:${password}" | md5sum - | sed -e 's/\s\+-//') > ${path_to_file}
```

リファレンス

これはリマインダとして、オンラインヘルプです。 (`tracd --help`):

```
Usage: tracd [options] [projenv] ...

Options:
 --version             show program's version number and exit
 -h, --help            show this help message and exit
 -a DIGESTAUTH, --auth=DIGESTAUTH
                       [projectdir],[htdigest_file],[realm]
 --basic-auth=BASICAUTH
                       [projectdir],[htpasswd_file],[realm]
 -p PORT, --port=PORT  the port number to bind to
 -b HOSTNAME, --hostname=HOSTNAME
                       the host name or IP address to bind to
 --protocol=PROTOCOL   http|scgi|ajp|fcgi
 -q, --unquote         unquote PATH_INFO (may be needed when using ajp)
 --http10              use HTTP/1.0 protocol version instead of HTTP/1.1
 --http11              use HTTP/1.1 protocol version (default)
 -e PARENTDIR, --env-parent-dir=PARENTDIR
                       parent directory of the project environments
 --base-path=BASE_PATH
                       the initial portion of the request URL's "path"
 -r, --auto-reload     restart automatically when sources are modified
 -s, --single-env      only serve a single project without the project list
 -d, --daemonize       run in the background as a daemon
 --pidfile=PIDFILE     when daemonizing, file to which to write pid
 --umask=MASK          when daemonizing, file mode creation mask to use, in
                       octal notation (default 022)
 --group=GROUP         the group to run as
 --user=USER           the user to run as
```

tracd を起動させたターミナルウィンドウを閉じる場合には、tracd がハングアップしないように -d オプションを使用してください。

Tips

静的なリソースを扱う

もし、 `tracd` **が単一のプロジェクトのみを扱う Web サーバだとしたら、 静的なリソースを割り当てるのに使用することができます。 (tar アーカイブ, Doxygen ドキュメントなど)**

この静的なリソースは `$TRAC_ENV/htdocs` **フォルダに置き、** `<project_URL>/chrome/site/...` **という URL でアクセスします。**

例: ファイル名が `$TRAC_ENV/htdocs/software-0.1.tar.gz` **だったとき、 対応する URL は** `/<project_name>/chrome/site/software-0.1.tar.gz` **となります。 代わりに** `htdocs:software-0.1.tar.gz` **(TracLinks のシンタックス) や** `[/<project_name>/chrome/site/software-0.1.tar.gz]` **(相対リンクのシンタックス) で記述することができます。**

> TracLinks **における** *htdocs:* **のサポートは Trac のバージョン 0.10 で追加されました。**

tracd をプロキシの背後で使用する

ある状況において tracd を Apache もしくは他のウェブサーバの背後で使用するときについてです。

この状況において、間違ったホストやプロトコルにリダイレクトされてしまったなどの経験があるかもしれません。この場合 (そして、この場合に限って) `[trac] use_base_url_for_redirect` **を** `true` **に設定することによって、リダイレクトを行なう際 Trac に** `[trac] base_url` **の値を強制的に使用させることができます。**

もし、`tracd` **に接続するために AJP プロトコルを使用しているならば (flup をインストールしているならば可能です) 、ダブルクオテーションの問題にぶつかったことがあるかもしれません。その際は、** `--unquote` **パラメータを追加することを考えてください。**

TracOnWindowsIisAjp, TracNginxRecipe **も参照してください。**

プロキシ背後の tracd の認証

`--basic-auth` **を使用する代わりに、tracd のインスタンスに外部認証を提供しても有効です。この方法については** #9206 **で議論されています。**

下記は Apache 2.2 、mod_proxy、mod_authnz_ldap を使用した場合の設定例です。

まず Apache のネームスペースに tracd を定義します。

```
<Location /project/proxified>
       Require ldap-group cn=somegroup, ou=Groups,dc=domain.com
       Require ldap-user somespecificusertoo
       ProxyPass http://localhost:8101/project/proxified/
       # Turns out we don't really need complicated RewriteRules here at all
       RequestHeader set REMOTE_USER %{REMOTE_USER}s
</Location>
```

HTTP_REMOTE_USER ヘッダを有効な認証ソースとして認識させるためには単一ファイルのプラグインが必要です。HTTP_FOO_BAR のような HTTP ヘッダは Foo-Bar に変換されます。remote-user-auth.py のようなファイル名をつけ、proxified/plugins ディレクトリ内に配置してください:

```
from trac.core import *
from trac.config import BoolOption
from trac.web.api import IAuthenticator

class MyRemoteUserAuthenticator(Component):

   implements(IAuthenticator)

   obey_remote_user_header = BoolOption('trac', 'obey_remote_user_header', 'false',
              """Whether the 'Remote-User:' HTTP header is to be trusted for user logins
               (''since ??.??').""")

   def authenticate(self, req):
       if self.obey_remote_user_header and req.get_header('Remote-User'):
           return req.get_header('Remote-User')
       return None
```

TracIni にパラメータを追加します:

```
...
[trac]
...
obey_remote_user_header = true
...
```

tracd を起動します:

```
tracd -p 8101 -r -s proxified --base-path=/project/proxified
```

もしプラグインをすべてのプロジェクトにインストールしたい場合、 共有pluginsディレクトリ に格納し、共有 trac.ini でプラグインのコンポーネントを有効に設定しなければいけません。

共有 config (例えば /srv/trac/conf/trac.ini):

```
[components]
remote-user-auth.* = enabled
[inherit]
plugins_dir = /srv/trac/plugins
[trac]
obey_remote_user_header = true
```

各プロジェクトの config (例えば /srv/trac/envs/myenv):

```
[inherit]
file = /srv/trac/conf/trac.ini
```

/ (root) とは異なるベースパスで起動する

Tracd は /<project> とは異なるベース URL でプロジェクトを提供することをサポートします。変更するためのパラメータは以下の通りです。

```
 $ tracd --base-path=/some/path
```

See also: TracInstall, TracCgi, TracModPython, TracGuide, Windows 上での tracd.exe の実行