Tracd

Tracd は軽量なスタンドアロンの Trac Web サーバです。ほとんどのケースでは CGI よりセットアップが簡単で、処理速度も速くなります。

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)。
- 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。

認証を使用する

Using tracd with Apache .htpasswd files:

To create a .htpasswd file using htpasswd:

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

then for additional users:

```
sudo htpasswd /var/www/html/.htpasswd-users username2
```

then for starting the tracd:

```
tracd -p 8080 --basic-auth=environmentname,/fullpath/environmentname/.htpasswd,/fullpath/environmentname /fullpath/environ
```

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。デフォルトはダイジェスト認証です;
基本認証を使用するためには、以降の例で使用する `--auth` を `--basic-auth` に置き換えて下さい。(ダイアログに使用する "realm"
を指定しなければなりません。 BASICAUTH には カンマ を末尾に指定した空の文字列を指定することができます。)

基本認証への対応はバージョン 0.9 以降で追加されました。

認証を使用するための一般的なコマンドは以下の通りです:

```
$ 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 はプロジェクトのベースディレクトリ; Note: これはプロジェクト名ではありません。そして Windows の環境においても、大文字と小文字を区別します。
- password_file_path パスワードファイルのパス
- realm レルム
- project_path プロジェクトのパス

使用例:

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

もちろん、ダイジェストファイルは複数のプロジェクト間で共有することが出来ます:

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

ダイジェストファイルを共有するもうひとつの方法は、プロジェクトの名前に "*" を指定することです:

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

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

もし Apache が使用できない場合でも、簡単な Python スクリプトでパスワードを生成できます:

```
from optparse import OptionParser
import md5

# 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")
(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")

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

Note: もし tracd を --basic-auth ではなくて、 --auth オプションを使用して起動するときに、上記のスクリプトを使わないといけないとしたら --auth の値に 'trac' を(シングルクォートなしで) 指定し、レルムを設定しなければなりません。(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
```

Note: --basic-auth を使用したければ、 .htpasswd ファイルを作成するのに Apache サーバの htpasswd ツールを使用する必要があります。他の部分についてはダイジェスト認証を行う場合とほとんど同じ方法です。しかしレルムには必ず empty 値 (すなわち、パス後のカンマのみ) を指定するようにしてください。 Windows で使用する場合は必ず -m オプションを使用してください。 (*nix ではテストしなかったので、その場合は不明です) Apache がない環境では、 htpasswd.py を使うとよいでしょう。 (このスクリプトは crypt か fcrypt モジュールが必要です; ソースのコメントに詳細が書いてあります。)

md5sum ユーティリティを使用するとダイジェストパスワードを作成することができます:

```
echo -e "${user}:trac:${password}\c" | md5sum - >>to-file
```

'to-file' は手で編集が必要です。行末の " -" を削除し、行頭に "${user}:trac:" を追加してください。詳細は trac-digest-corrected.sh を参照してください。 (訳注: realm が trac に固定されていますので、実際には改造しないと使えないでしょう)

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。 Wiki には、相対リンクシンタックスを使用して、 [/<project_name>/chrome/site/software-0.1.tar.gz] と書くことができます。 (訳注: [/chrome/site/software-0.1.tar.gz] が正しい)

Trac の開発バージョンでは新しく htdocs: に対応します。 TracLinks は 上記のようなシンタックスになります。これによって、上記の例のリンクはただ単に htdocs:software-0.1.tar.gz と書くことができます。

Apache の書き換え規則を使用する

Apache のビハインドで tracd を使用すると、いくつかの状況で不正なホストまたはプロトコルに URL をリダイレクトされる問題が発生するかもしれません。この場合 (この場合だけ) [trac] use_base_url_for_redirect を true に設定することができます。これによって Trac がやむを得ず [trac] base_url の値を使用するためリダイレクトを行います。

検索パスとは別のベースパス (/)

Tracd は、プロジェクト毎に異なるベース URL 、および /<project> をサポートします。コマンドは以下の通りです。

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

See also: TracInstall, TracCgi, TracModPython, TracGuide