Table of Contents

Tracd

Tracd is a lightweight standalone Trac web server. In most cases it's easier to setup and runs faster than the CGI script.

Pros

- Fewer dependencies: You don't need to install apache or any other web-server.
- Fast: Should be almost as fast as the mod_python version (and much faster than the CGI).
- Automatic reloading: For development, Tracd can be used in *auto_reload* mode, which will automatically restart the server whenever you make a change to the code (in Trac itself or in a plugin).

Cons

- Fewer features: Tracd implements a very simple web-server and is not as configurable or as scalable as Apache HTTPD.
- No native HTTPS support: sslwrap can be used instead,

or stunnel -- a tutorial on how to use stunnel with tracd or Apache with mod_proxy.

Usage examples

A single project on port 8080. (http://localhost:8080/)

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

With more than one project. (http://localhost:8080/project1/ and http://localhost:8080/project2/)

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

You can't have the last portion of the path identical between the projects since Trac uses that name to keep the URLs of the different projects unique. So if you use `/project1/path/to` and `/project2/path/to`, you will only see the second project.

An alternative way to serve multiple projects is to specify a parent directory in which each subdirectory is a Trac project, using the `-e` option. The example above could be rewritten:

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

Using Authentication

Tracd provides support for both Basic and Digest authentication. The default is to use Digest; to use Basic authentication, replace `–auth` with `–basic-auth` in the examples below, and omit the realm.

```
//Support for Basic authentication was added in version 0.9.//
```

If the file `/path/to/users.htdigest` contains user accounts for project1 with the realm “mycompany.com”, you'd use the following command-line to start tracd:

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

Note: the project “name” passed to the `–auth` option is the base name of the project environment directory.

Of course, the digest file can be be shared so that it is used for more than one project:

```
$ 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
```

Another way to share the digest file is to specify “*” for the project name:

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

How to set up an htdigest password file

If you have Apache available, you can use the htdigest command to generate the password file. Type 'htdigest' to get some usage instructions, or read this page from the Apache manual to get precise instructions. You'll be prompted for a password to enter for each user that you create. For the name of the password file, you can use whatever you like, but if you use something like `users.htdigest` it will remind you what the file contains. As a suggestion, put it in your <projectname>/conf folder along with the trac.ini file.

Note that you can start tracd without the –auth argument, but if you click on the *Login* link you will get an error.

Generating Passwords Without Apache

If you don't have Apache available, you can use this simple Python script to generate your passwords:

```
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: If you use the above script you must use the –auth option to tracd, not –basic-auth, and you must set the realm in the –auth value to 'trac' (without the quotes). Example usage (assuming you saved the script as trac-digest.py):

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

Tips

Serving static content

If `tracd` is the only webserver used for the project, it can also be used to distribute static content (tarballs, Doxygen documentation, etc.)

This static content should be put in the `$TRAC_ENV/htdocs` folder, and is accessed by URLs like `<project_URL>/chrome/site/...`.

Example: given a `$TRAC_ENV/htdocs/software-0.1.tar.gz` file, the corresponding relative URL would be `/<project_name>/chrome/site/software-0.1.tar.gz`, which in turn can be written using the relative link syntax in the Wiki: `software-0.1.tar.gz`

The development version of Trac supports a new `htdocs:` TracLinks syntax for the above. With this, the example link above can be written simply `htdocs:software-0.1.tar.gz`.

See also: TracInstall, TracCgi, TracModPython, TracGuide

Translation:

- Russian (перевод на Русский)