

Websourcebrowser

Source code at a glance

EuroPython 2008

Stefan Schwarzer, SSchwarzer.com
info@sschwarzer.com

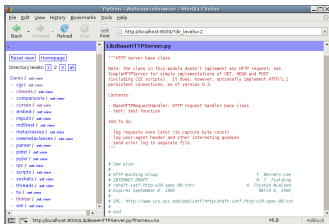
Vilnius, Lithuania, 2008-07-09

Motivation

- **Contribution to an existent project**
 - Open source projekt (Python, TurboGears, ...)
 - Maintain an employer's code
 - ...
- **File managers and editors have their limitations**
 - **Either** directory structure **or** file visible
 - In general, not usable via shell login

Features

- Easy to use, minimal requirements
- Webservicebrowser is a webserver (think pydoc -p)
- Usable with every webbrowser, locally or remotely
- View directories and files side-by-side
- Syntax highlighting if Pygments is installed
- Cross-platform (tested on Linux and Windows)
- Open source (MIT license)



```
libbaseHTTPServer.py
"""HTTP server base class.
Note: the class in this module doesn't implement any HTTP request, see
SimpleHTTPServer for simple implementations of GET, HEAD and POST
(including CGI scripts). It does, however, optionally implement HTTP/1.1
persistent connections, see it's version 0.3.
Contents:
- BaseHTTPServerHandler: HTTP request handler base class
- test: Test function
See To Do:
- log requests even later (to capture byte count)
- log user-agent header and other interesting goodies
- split error log to separate files
...
# See also:
# HTTP monitoring scripts
# HTTPSEC scripts
# HTTPD: conf-ftp-usb-001.txt
# Apache: httpd.conf
# URL: http://www.ccs.ucl.ac.uk/pub/ietf/http/whrf-conf-ftp-usb-001.txt
#
#
"""
```

Web interface

Python - Webservice - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Back Forward Reload Stop Print http://localhost:8000/?dir_levels=2

Lib/BaseHTTPServer.py

Reset view Homepage

Directory levels: 1 2 3 all

Demo / set view

- > cgi / set view
- > classes / set view
- > comparisons / set view
- > curses / set view
- > embed / set view
- > imputil / set view
- > mdStest / set view
- > metaclasses / set view
- > newmetaclasses / set view
- > parser / set view
- > pdist / set view
- > pysvr / set view
- > rpc / set view
- > scripts / set view
- > sockets / set view
- > threads / set view
- > tix / set view
- > tkinter / set view
- > xml / set view

```
"""HTTP server base class.

Note: the class in this module doesn't implement any HTTP request; see
SimpleHTTPServer for simple implementations of GET, HEAD and POST
(including CGI scripts). It does, however, optionally implement HTTP/1.1
persistent connections, as of version 0.3.

Contents:

- BaseHTTPRequestHandler: HTTP request handler base class
- test: test function

XXX To do:

- log requests even later (to capture byte count)
- log user-agent header and other interesting goodies
- send error log to separate file
"""

# See also:
#
# HTTP Working Group                                T. Berners-Lee
# INTERNET-DRAFT                                    R. T. Fielding
# <draft-ietf-http-v10-spec-00.txt>                   H. Frystyk Nielsen
# Expires September 8, 1995                           March 8, 1995
#
# URL: http://www.ics.uci.edu/pub/ietf/http/draft-ietf-http-v10-spec-00.txt
#
# and
```

http://localhost:8000/Lib/BaseHTTPServer.py?frames=no

MLB Adblock

Source code in ELinks text browser

```
Python - Webservicebrowser - ELinks
Doc - Python (1/11)

Doc
Reset view Homepage
Directory levels: 1 2 3 all
up
c-api / set view
> abstract.rst
> concrete.rst
> exceptions.rst
> index.rst
> init.rst
> intro.rst
> memory.rst
> newtypes.rst
> refcounting.rst
> utilities.rst
> veryhigh.rst
data / set view
> refcounts.dat
distutils / set view
> apiref.rst
> builtdist.rst
> commandref.rst
> configfile.rst
> examples.rst
> extending.rst
> index.rst
> introduction.rst
> packageindex.rst
> setupscript.rst
> sourcedist.rst
> uploading.rst
documenting / set view
> fromlatex.rst
> index.rst
> intro.rst
> markup.rst
> rest.rst
> sphinx.rst
> style.rst
extending / set view
http://localhost:8000/Doc/distutils/commandref.rst?frames-no - /Doc/distutils/commandref.rst [-----]

Doc/distutils/commandref.rst
.. _reference:
*****
Command Reference
*****

.. % \section{Building modules: the \protect\command{build} command family}
.. % \label{build-cmds}
.. % \subsubsection{\protect\command{build}}
.. % \label{build-cmd}
.. % \subsubsection{\protect\command{build\_py}}
.. % \label{build-py-cmd}
.. % \subsubsection{\protect\command{build\_ext}}
.. % \label{build-ext-cmd}
.. % \subsubsection{\protect\command{build\_clib}}
.. % \label{build-clib-cmd}

.. _install-cmd:
Installing modules: the :command:`install` command family
=====

The install command ensures that the build commands have been run and then runs
the subcommands :command:`install_lib`, :command:`install_data` and
:command:`install_scripts` .

.. % \subsubsection{\protect\command{install\_lib}}
.. % \label{install-lib-cmd}

.. _install-data-cmd:
:command:`install_data`
=====

This command installs all data files provided with the distribution.

.. _install-scripts-cmd:
```

<http://webservicebrowser.sschwarzer.net>

Webservicebrowser

[Login](#) [Settings](#) [Help/Guide](#) [About Trac](#)

[Wiki](#)

[Timeline](#)

[Roadmap](#)

[Browse Source](#)

[View Tickets](#)

[Search](#)

[Start Page](#) [Index by Title](#) [Index by Date](#) [Last Change](#)

Webservicebrowser

Purpose

Webservicebrowser makes it easy to visually scan trees of source code. In particular, you can view the directory tree and the source code of a file side by side, so you can quickly change to a different file. Get [release 0.3](#) (alpha).

Features

- Platform-independent, any web browser can be used to view the source code
- Includes a small webserver, so the source code and the browser to view it can be on different computers
- By default, they are on the same computer :-)
- If [Pygments](#) is installed on the server side, which may be the local computer, source code is automatically highlighted
- Image files supported by the browser are displayed
- Binary files are displayed as hexdumps
- Written in portable [Python](#)
- [Open Source](#) ([MIT license](#))

Website contents

- Documentation in [ManPage](#) style (see also [InstallationInstructions](#) for the installation)
- [Download](#)

```
webservicebrowser-0.3/browser.py

py_path = os.path.abspath(sys.modules[__name__].__file__)
# assume the CSS file is in the same directory as the Python file
css_path = os.path.join(os.path.dirname(py_path), config.CSS_FILE_NAME)
return css_path

class SourceBrowserHandler(BaseHTTPServer.BaseHTTPRequestHandler):
    #protocol_version = "HTTP/1.1"
    server_version = "webservicebrowser 0.x"
    sys_version = ""

    # deal with builtin commands

    def handle_builtin(self, url):
        """
        Handle a special command which is built into Webservicebrowser.

        Currently, there are the following commands:

        - the CSS file name: return the CSS file of Webservicebrowser
          to the web client

        - raw_data: returns the raw data of a file to the web client
        """
        # determine builtin command/file: skip to string after the
        # special directory
        builtin_url = url[len(config.SPECIAL_DIR):]
        parse_result = urlparse.urlparse(builtin_url)
        builtin = parse_result[2]
        query_string = cgi.parse(os.parse_result[4])

        query_string = {}
        query_string["frames"] = 0
```

Ideas for further development

- Improve navigation/orientation
- File-specific views (e. g. list contents of archive files)
- Speed improvements (caching)
- Make better suited for public websites (multithreading)
- JavaScript, maybe AJAX (optional, Webservicebrowser should still work in text browsers)
- Turn import/include statements into links
- Search function
- Outline (“table of contents”) of source code files

Wanted :-)

- Security audit (continuous activity)
- User interface ideas
- Bug reports
- Patches
- More wiki pages
- **Feedback**, e. g. **at this conference** ;-)

What about a sprint?

Talk to me!