cheroot

Release 6.3.1.dev1+g9d73226.d20180520

May 20, 2018

Contents

CHAPTER 1

History

1.1 v6.3.0

17 May 2018

- PR #87: Add `cheroot` command and runpy launcher to launch a WSGI app from the command-line.

1.2 v6.2.4

19 Apr 2018

- Fix missing `resolve_peer_creds` argument in *`cheroot.wsgi.Server`* being bypassed into *`cheroot.server.HTTPServer`*.
- PR #85: Revert conditional dependencies. System packagers should honor the dependencies as declared by cheroot, which are defined intentionally.

1.3 v6.2.3

14 Apr 2018

- PR #85: Skip installing dependencies from backports namespace under Python 3.

1.4 v6.2.2

14 Apr 2018

- #84 (CherryPy #1704): Fix regression, causing `ModuleNotFoundError` under cygwin.

1.5 v6.2.1

10 Apr 2018

- PR #83: Fix regression, caused by inverted check for Windows OS.
- Add more URLs to distribution metadata

1.6 v6.2.0

10 Apr 2018

- PR #37: Implement PEERCRED lookup over UNIX-socket HTTP connection.
  - Discover connected process' PID/UID/GID
  - Respect server switches: `peercreds_enabled` and `peercreds_resolve_enabled`
  - `get_peer_creds` and `resolve_peer_creds` methods on connection
  - `peer_pid`, `peer_uid`, `peer_gid`, `peer_user` and `peer_group` properties on connection
  - `X_REMOTE_PID`, `X_REMOTE_UID`, `X_REMOTE_GID`, `X_REMOTE_USER` (`REMOTE_USER`) and `X_REMOTE_GROUP` WSGI environment variables when enabled and supported
  - Per-connection caching to reduce lookup cost

1.7 v6.1.2

08 Apr 2018

- #81: Fix regression introduced by PR #80.
  - Restore *`storing bound socket`* in Windows broken by use of `socket.AF_UNIX`

1.8 v6.1.1

07 Apr 2018

- PR #80: Fix regression introduced by 68a5769.
  - Get back support for `socket.AF_UNIX` in stored bound address in *`cheroot.server.HTTPServer.bind_addr`*

1.9 v6.1.0

05 Apr 2018

- PR #67: Refactor testsuite to completely rely on pytest.
  - Integrate pytest-testmon and pytest-watch
  - Stabilise testing

- CherryPy #1664 via PR #66: Implement input termination flag support as suggested by @mitsuhiko in his wsgi.input_terminated Proposal.
- #73: Fix SSL error bypassing.
- #77 via PR #78: Fix WSGI documentation example to support Python 3.
- PR #76: Send correct conditional HTTP error in helper function.
- CherryPy #1404 via PR #75: Fix headers being unsent before request closed. Now we double check that they've been sent.
- Minor docs improvements.
- Minor refactoring.

1.10 v6.0.0

04 Dec 2017

- Drop support for Python 2.6, 3.1, 3.2, and 3.3.
- Also drop built-in SSL support for Python 2.7 earlier than 2.7.9.

1.11 v5.11.0

04 Dec 2017

- CherryPy #1621: To support webtest applications that feed absolute URIs to getPage but expect the scheme/host/port to be ignored (as cheroot 5.8 and earlier did), provide a `strip_netloc` helper and recipe for calling it in a subclass.

1.12 v5.10.0

23 Nov 2017

- Minor refactorings of cheroot/server.py to reduce redundancy of behavior.
- Delinting with fewer exceptions.
- Restored license to BSD.

1.13 v5.9.2

23 Nov 2017

- #61: Re-release without spurious files in the distribution.

1.14 v5.9.1

17 Nov 2017

- #58: Reverted encoding behavior in wsgi module to correct regression in CherryPy tests.

1.15 v5.9.0

16 Nov 2017

- CherryPy #1088 and PR #53: Avoid using SO_REUSEADDR on Windows where it has different semantics.
- `cheroot.tests.webtest` adopts the one method that was unique in CherryPy, now superseding the implementation there.
- Substantial cleanup around compatibility functions (_compat module).
- License unintentionally changed to MIT. BSD still declared and intended.

1.16 v5.8.3

11 Aug 2017

- Improve HTTP request line validation:
  - Improve HTTP version parsing
- Fix HTTP CONNECT method processing:
  - Respond with `405 Method Not Allowed` if `proxy_mode is False`
  - Validate that request-target is in authority-form
- Improve tests in `test.test_core`
- PR #44: Fix EPROTOTYPE @ Mac OS

1.17 v5.8.2

07 Aug 2017

- Fix PR #39 regression. Add HTTP request line check: absolute URI path must start with a forward slash ("/").

1.18 v5.8.1

05 Aug 2017

- CI improvements:
  - Add basic working Circle CI v2 config
- Fix URI encoding bug introduced in PR #39
  - Improve `cheroot.test.helper.Controller` to properly match unicode

1.19 v5.8.0

01 Aug 2017

- CI improvements:
  - Switch to native PyPy support in Travis CI

  - Take into account **PEP 257** compliant modules
  - Build wheel in Appveyor and store it as an artifact
- Improve urllib support in *cheroot._compat*
- #38 via PR #39: Improve URI parsing:
  - Make it compliant with **RFC 7230**, **RFC 7231** and **RFC 2616**
  - Fix setting of `environ['QUERY_STRING']` in WSGI
  - Introduce `proxy_mode` and `strict_mode` argument in `server.HTTPRequest`
  - Fix decoding of unicode URIs in WSGI 1.0 gateway

1.20 v5.7.0

24 Jun 2017

- CI improvements:
  - Don't run tests during deploy stage
  - Use VM based build job env only for pyenv envs
  - Opt-in for beta trusty image @ Travis CI
  - Be verbose when running tests (show test names)
  - Show xfail/skip details during test run
- #34: Fix `_handle_no_ssl` error handler calls
- #21: Fix `test_conn` tests:
  - Improve setup_server def in HTTP connection tests
  - Fix HTTP streaming tests
  - Fix HTTP/1.1 pipelining test under Python 3
  - Fix `test_readall_or_close` test
  - Fix `test_No_Message_Body`
  - Clarify `test_598` fail reason
- #36: Add GitHub templates for PR, issue && contributing
- #27: Default HTTP Server header to Cheroot version str
- Cleanup _compat functions from server module

1.21 v5.6.0

20 Jun 2017

- Fix all **PEP 257** related errors in all non-test modules.

  `cheroot/test/*` folder is only one left allowed to fail with this linter.
- CherryPy #1602 and PR #30: Optimize chunked body reader loop by returning empty data is the size is 0.
- CherryPy #1486: Reset buffer if the body size is unknown

- CherryPy #1131: Add missing size hint to SizeCheckWrapper

1.22 v5.5.2

18 Jun 2017

- PR #32: Ignore "unknown error" and "https proxy request" SSL errors.

  Ref: sabnzbd/sabnzbd#820

  Ref: sabnzbd/sabnzbd#860

1.23 v5.5.1

18 Jun 2017

- Make Appveyor list separate tests in corresponding tab.
- PR #29: Configure Travis CI build stages.

  Prioritize tests by stages.

  Move deploy stage to be run very last after all other stages finish.
- PR #31: Ignore "Protocol wrong type for socket" (EPROTOTYPE) @ OSX for non-blocking sockets.

  This was originally fixed for regular sockets in CherryPy #1392.

  Ref: https://forums.sabnzbd.org/viewtopic.php?f=2&t=22728&p=112251

1.24 v5.5.0

02 May 2017

- #17 via PR #25: Instead of a read_headers function, cheroot now supplies a `HeaderReader` class to perform the same function.

  Any *`HTTPRequest`* object may override the header_reader attribute to customize the handling of incoming headers.

  The server module also presents a provisional implementation of a `DropUnderscoreHeaderReader` that will exclude any headers containing an underscore. It remains an exercise for the implementer to demonstrate how this functionality might be employed in a server such as CherryPy.
- PR #26: Configured TravisCI to run tests under OS X.

1.25 v5.4.0

19 Mar 2017

- PR #22: Add "ciphers" parameter to SSLAdapter.

1.26 v5.3.0

12 Mar 2017

- PR #8: Updated style to better conform to **PEP 8**.

  Refreshed project with jaraco skeleton.

  Docs now built and deployed at RTD.

1.27 v5.2.0

02 Mar 2017

- #5: Set `Server.version` to Cheroot version instead of CherryPy version.
- PR #4: Prevent tracebacks and drop bad HTTPS connections in the `BuiltinSSLAdapter`, similar to `pyOpenSSLAdapter`.
- #3: Test suite now runs and many tests pass. Some are still failing.

1.28 v5.1.0

22 Jan 2017

- Removed the WSGI prefix from classes in *`cheroot.wsgi`*. Kept aliases for compatibility.
- #1: Corrected docstrings in *`cheroot.server`* and *`cheroot.wsgi`*.
- PR #2: Fixed `ImportError` when pkg_resources cannot find the cheroot distribution.

1.29 v5.0.1

14 Jan 2017

- Fix error in `parse_request_uri` created in 68a5769.

1.30 v5.0.0

14 Jan 2017

- Initial release based on cherrypy.cherrypy.wsgiserver 8.8.0.

CHAPTER 2

cheroot._compat module

Compatibility code for using Cheroot with various versions of Python.

cheroot._compat.**assert_native**(*n*)
Check whether the input is of nativ `str` type.

Raises: TypeError: in case of failed check

cheroot._compat.**bton**(*b*, *encoding='ISO-8859-1'*)
Return the byte string as native string in the given encoding.

cheroot._compat.**ntob**(*n*, *encoding='ISO-8859-1'*)
Return the native string as bytes in the given encoding.

cheroot._compat.**ntou**(*n*, *encoding='ISO-8859-1'*)
Return the native string as unicode with the given encoding.

2.1 Indices and tables

- genindex
- modindex
- search

CHAPTER 3

cheroot.errors module

Collection of exceptions raised and/or processed by Cheroot.

exception cheroot.errors.**FatalSSLAlert**
Bases: Exception

Exception raised when the SSL implementation signals a fatal alert.

exception cheroot.errors.**MaxSizeExceeded**
Bases: Exception

Exception raised when a client sends more data then acceptable within limit.

Depends on `request.body.maxbytes` config option if used within CherryPy

exception cheroot.errors.**NoSSLError**
Bases: Exception

Exception raised when a client speaks HTTP to an HTTPS socket.

cheroot.errors.**plat_specific_errors**(**errnames*)
Return error numbers for all errors in errnames on this platform.

The 'errno' module contains different global constants depending on the specific platform (OS). This function will return the list of numeric values for a given list of potential names.

3.1 Indices and tables

- genindex
- modindex
- search

CHAPTER 4

cheroot.makefile module

Socket file object.

class cheroot.makefile.**BufferedWriter**(*raw*, *buffer_size=8192*)
Bases: _pyio.BufferedWriter

Faux file object attached to a socket object.

write(*b*)
Write bytes to buffer.

cheroot.makefile.**MakeFile**(*sock*, *mode='r'*, *bufsize=8192*)
File object attached to a socket object.

class cheroot.makefile.**MakeFile_PY2**(**args*, ***kwargs*)
Bases: object

Faux file object attached to a socket object.

flush()
Write all data from buffer to socket and reset write buffer.

read(*size=-1*)
Read data from the socket to buffer.

readline(*size=-1*)
Read line from the socket to buffer.

recv(*size*)
Receive message of a size from the socket.

send(*data*)
Send some part of message to the socket.

write(*data*)
Sendall for non-blocking sockets.

cheroot.makefile.**MakeFile_PY3**(*sock*, *mode='r'*, *bufsize=8192*)
File object attached to a socket object.

4.1 Indices and tables

- genindex
- modindex
- search

CHAPTER 5

cheroot.server module

A high-speed, production ready, thread pooled, generic HTTP server.

For those of you wanting to understand internals of this module, here's the basic call flow. The server's listening thread runs a very tight loop, sticking incoming connections onto a Queue:

```
server = HTTPServer(...)
server.start()
while True:
    tick()
    # This blocks until a request comes in:
    child = socket.accept()
    conn = HTTPConnection(child, ...)
    server.requests.put(conn)
```

Worker threads are kept in a pool and poll the Queue, popping off and then handling each connection in turn. Each connection can consist of an arbitrary number of requests and their responses, so we run a nested loop:

```
while True:
    conn = server.requests.get()
    conn.communicate()
    ->  while True:
            req = HTTPRequest(...)
            req.parse_request()
            ->  # Read the Request-Line, e.g. "GET /page HTTP/1.1"
                req.rfile.readline()
                read_headers(req.rfile, req.inheaders)
            req.respond()
            ->  response = app(...)
                try:
                    for chunk in response:
                        if chunk:
                            req.write(chunk)
                finally:
                    if hasattr(response, "close"):
```

(continues on next page)

(continued from previous page)

```
                        response.close()
                if req.close_connection:
                    return
```

And now for a trivial doctest to exercise the test suite

```
>>> 'HTTPServer' in globals()
True
```

class cheroot.server.**HTTPRequest**(*server*, *conn*, *proxy_mode=False*, *strict_mode=True*)

Bases: object

An HTTP Request (and response).

A single HTTP connection may consist of multiple request/response pairs.

chunked_write = False

If True, output will be encoded with the "chunked" transfer-coding.

This value is set automatically inside send_headers.

close_connection = False

Signals the calling Connection that the request should close. This does not imply an error! The client and/or server may each request that the connection be closed.

conn = None

The HTTPConnection object on which this request connected.

ensure_headers_sent()

Ensure headers are sent to the client if not already sent.

header_reader = <cheroot.server.HeaderReader object>

A HeaderReader instance or compatible reader.

inheaders = {}

A dict of request headers.

outheaders = []

A list of header tuples to write in the response.

parse_request()

Parse the next HTTP request start-line and message-headers.

read_request_headers()

Read self.rfile into self.inheaders. Return success.

read_request_line()

Read and parse first line of the HTTP request.

Returns: bool: True if the request line is valid or False if it's malformed.

ready = False

When True, the request has been parsed and is ready to begin generating the response. When False, signals the calling Connection that the response should not be generated and the connection should close.

respond()

Call the gateway and write its iterable output.

send_headers()

Assert, process, and send the HTTP response message-headers.

You must set self.status, and self.outheaders before calling this.

server = None
The HTTPServer object which is receiving this request.

simple_response(*status*, *msg=''*)
Write a simple response back to the client.

write(*chunk*)
Write unbuffered data to the client.

class cheroot.server.**HTTPConnection**(*server*, *sock*, *makefile=<function MakeFile_PY3>*)
Bases: object

An HTTP connection (active socket).

RequestHandlerClass
alias of *HTTPRequest*

close()
Close the socket underlying this connection.

communicate()
Read each request and respond appropriately.

get_peer_creds()
Return the PID/UID/GID tuple of the peer socket for UNIX sockets.

This function uses SO_PEERCRED to query the UNIX PID, UID, GID of the peer, which is only available if the bind address is a UNIX domain socket.

Raises: NotImplementedError: in case of unsupported socket type RuntimeError: in case of SO_PEERCRED lookup unsupported or disabled

linger = False

peer_gid
Return the group id of the connected peer process.

peer_group
Return the group of the connected peer process.

peer_pid
Return the id of the connected peer process.

peer_uid
Return the user id of the connected peer process.

peer_user
Return the username of the connected peer process.

peercreds_enabled = False

peercreds_resolve_enabled = False

rbufsize = 8192

remote_addr = None

remote_port = None

resolve_peer_creds()
Return the username and group tuple of the peercreds if available.

Raises: NotImplementedError: in case of unsupported OS RuntimeError: in case of UID/GID lookup unsupported or disabled

ssl_env = None

wbufsize = 8192

class cheroot.server.**HTTPServer**(*bind_addr*, *gateway*, *minthreads=10*, *maxthreads=-1*, *server_name=None*, *peercreds_enabled=False*, *peercreds_resolve_enabled=False*)

Bases: object

An HTTP server.

ConnectionClass
The class to use for handling HTTP connections.

alias of *HTTPConnection*

bind(*family*, *type*, *proto=0*)
Create (or recreate) the actual socket object.

bind_addr
Return the interface on which to listen for connections.

For TCP sockets, a (host, port) tuple. Host values may be any IPv4 or IPv6 address, or any valid hostname. The string 'localhost' is a synonym for '127.0.0.1' (or '::1', if your hosts file prefers IPv6). The string '0.0.0.0' is a special IPv4 entry meaning "any active interface" (INADDR_ANY), and '::' is the similar IN6ADDR_ANY for IPv6. The empty string or None are not allowed.

For UNIX sockets, supply the filename as a string.

Systemd socket activation is automatic and doesn't require tempering with this variable.

clear_stats()
Reset server stat counters..

error_log(*msg=''*, *level=20*, *traceback=False*)
Write error message to log.

Args: msg (str): error message level (int): logging level traceback (bool): add traceback to output or not

gateway = None
A Gateway instance.

interrupt
Flag interrupt of the server.

max_request_body_size = 0
The maximum size, in bytes, for request bodies, or 0 for no limit.

max_request_header_size = 0
The maximum size, in bytes, for request headers, or 0 for no limit.

maxthreads = None
The maximum number of worker threads to create.

(default -1 = no limit)

minthreads = None
The minimum number of worker threads to create (default 10).

nodelay = True
If True (the default since 3.1), sets the TCP_NODELAY socket option.

peercreds_enabled = False
If True, peer cred lookup can be performed via UNIX domain socket.

peercreds_resolve_enabled = False
If True, username/group will be looked up in the OS from peercreds.

protocol = 'HTTP/1.1'
The version string to write in the Status-Line of all HTTP responses.

For example, "HTTP/1.1" is the default. This also limits the supported features used in the response.

ready = False
Internal flag which indicating the socket is accepting connections.

request_queue_size = 5
The 'backlog' arg to socket.listen(); max queued connections.

(default 5).

runtime()
Return server uptime.

safe_start()
Run the server forever, and stop it cleanly on exit.

server_name = None
The name of the server; defaults to `self.version`.

shutdown_timeout = 5
The total time to wait for worker threads to cleanly exit.

Specified in seconds.

software = None
The value to set for the SERVER_SOFTWARE entry in the WSGI environ.

If None, this defaults to `'%s Server' % self.version`.

ssl_adapter = None
An instance of ssl.Adapter (or a subclass).

You must have the corresponding SSL driver library installed.

start()
Run the server forever.

stop()
Gracefully shutdown a server that is serving forever.

tick()
Accept a new connection and put it on the Queue.

timeout = 10
The timeout in seconds for accepted connections (default 10).

version = 'Cheroot/6.3.1.dev1+g9d73226.d20180520'
A version string for the HTTPServer.

class cheroot.server.**SizeCheckWrapper**(*rfile*, *maxlen*)
Bases: object

Wraps a file-like object, raising MaxSizeExceeded if too large.

close()
Release resources allocated for rfile.

next()
Generate next file chunk.

read(*size=None*)
Read a chunk from rfile buffer and return it.

Args: size (int): amount of data to read

Returns: bytes: Chunk from rfile, limited by size if specified.

readline(*size=None*)
Read a single line from rfile buffer and return it.

Args: size (int): minimum amount of data to read

Returns: bytes: One line from rfile.

readlines(*sizehint=0*)
Read all lines from rfile buffer and return them.

Args: sizehint (int): hint of minimum amount of data to read

Returns: list[bytes]: Lines of bytes read from rfile.

class cheroot.server.**KnownLengthRFile**(*rfile*, *content_length*)
Bases: object

Wraps a file-like object, returning an empty string when exhausted.

close()
Release resources allocated for rfile.

next()
Generate next file chunk.

read(*size=None*)
Read a chunk from rfile buffer and return it.

Args: size (int): amount of data to read

Returns: bytes: Chunk from rfile, limited by size if specified.

readline(*size=None*)
Read a single line from rfile buffer and return it.

Args: size (int): minimum amount of data to read

Returns: bytes: One line from rfile.

readlines(*sizehint=0*)
Read all lines from rfile buffer and return them.

Args: sizehint (int): hint of minimum amount of data to read

Returns: list[bytes]: Lines of bytes read from rfile.

class cheroot.server.**ChunkedRFile**(*rfile*, *maxlen*, *bufsize=8192*)
Bases: object

Wraps a file-like object, returning an empty string when exhausted.

This class is intended to provide a conforming wsgi.input value for request entities that have been encoded with the 'chunked' transfer encoding.

close()
Release resources allocated for rfile.

read(*size=None*)
Read a chunk from rfile buffer and return it.

Args: size (int): amount of data to read

Returns: bytes: Chunk from rfile, limited by size if specified.

read_trailer_lines()
Read HTTP headers and yield them.

Returns: Generator: yields CRLF separated lines.

readline(*size=None*)
Read a single line from rfile buffer and return it.

Args: size (int): minimum amount of data to read

Returns: bytes: One line from rfile.

readlines(*sizehint=0*)
Read all lines from rfile buffer and return them.

Args: sizehint (int): hint of minimum amount of data to read

Returns: list[bytes]: Lines of bytes read from rfile.

class cheroot.server.**Gateway**(*req*)
Bases: object

Base class to interface HTTPServer with other systems, such as WSGI.

respond()
Process the current request. Must be overridden in a subclass.

cheroot.server.**get_ssl_adapter_class**(*name='builtin'*)
Return an SSL adapter class for the given name.

5.1 Indices and tables

- genindex
- modindex
- search

CHAPTER 6

cheroot.testing module

6.1 Indices and tables

- genindex
- modindex
- search

CHAPTER 7

cheroot.wsgi module

This class holds Cheroot WSGI server implementation.

Simplest example on how to use this server:

```
from cheroot import wsgi

def my_crazy_app(environ, start_response):
    status = '200 OK'
    response_headers = [('Content-type','text/plain')]
    start_response(status, response_headers)
    return [b'Hello world!']

addr = '0.0.0.0', 8070
server = wsgi.Server(addr, my_crazy_app)
server.start()
```

The Cheroot WSGI server can serve as many WSGI applications as you want in one instance by using a PathInfoDispatcher:

```
path_map = {
    '/': my_crazy_app,
    '/blog': my_blog_app,
}
d = wsgi.PathInfoDispatcher(path_map)
server = wsgi.Server(addr, d)
```

class cheroot.wsgi.**Gateway**(*req*)

Bases: *cheroot.server.Gateway*

A base class to interface HTTPServer with WSGI.

classmethod gateway_map()

Create a mapping of gateways and their versions.

Returns:

dict[tuple[int,int],class]: map of gateway version and corresponding class

get_environ()
Return a new environ dict targeting the given wsgi.version.

respond()
Process the current request.

From **PEP 333**:

> The start_response callable must not actually transmit the response headers. Instead, it must store them for the server or gateway to transmit only after the first iteration of the application return value that yields a NON-EMPTY string, or upon the application's first invocation of the write() callable.

start_response(*status*, *headers*, *exc_info=None*)
WSGI callable to begin the HTTP response.

write(*chunk*)
WSGI callable to write unbuffered data to the client.

This method is also used internally by start_response (to write data from the iterable returned by the WSGI application).

class cheroot.wsgi.**Gateway_10**(*req*)
Bases: *cheroot.wsgi.Gateway*

A Gateway class to interface HTTPServer with WSGI 1.0.x.

get_environ()
Return a new environ dict targeting the given wsgi.version.

version = (1, 0)

class cheroot.wsgi.**Gateway_u0**(*req*)
Bases: *cheroot.wsgi.Gateway_10*

A Gateway class to interface HTTPServer with WSGI u.0.

WSGI u.0 is an experimental protocol, which uses unicode for keys and values in both Python 2 and Python 3.

get_environ()
Return a new environ dict targeting the given wsgi.version.

version = ('u', 0)

class cheroot.wsgi.**PathInfoDispatcher**(*apps*)
Bases: object

A WSGI dispatcher for dispatch based on the PATH_INFO.

class cheroot.wsgi.**Server**(*bind_addr*, *wsgi_app*, *numthreads=10*, *server_name=None*, *max=-1*, *request_queue_size=5*, *timeout=10*, *shutdown_timeout=5*, *accepted_queue_size=-1*, *accepted_queue_timeout=10*, *peercreds_enabled=False*, *peercreds_resolve_enabled=False*)
Bases: *cheroot.server.HTTPServer*

A subclass of HTTPServer which calls a WSGI application.

numthreads
Set minimum number of threads.

wsgi_version = (1, 0)
The version of WSGI to produce.

cheroot.wsgi.**WSGIGateway**
alias of *cheroot.wsgi.Gateway*

cheroot.wsgi.**WSGIGateway_10**
alias of *cheroot.wsgi.Gateway_10*

cheroot.wsgi.**WSGIGateway_u0**
alias of *cheroot.wsgi.Gateway_u0*

cheroot.wsgi.**WSGIPathInfoDispatcher**
alias of *cheroot.wsgi.PathInfoDispatcher*

cheroot.wsgi.**WSGIServer**
alias of *cheroot.wsgi.Server*

7.1 Indices and tables

- genindex
- modindex
- search

Cheroot is the high-performance, pure-Python HTTP server used by CherryPy.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index

V

W