Cjango-Unchained

Generated by Doxygen 1.8.13

Contents

Chapter 1

Cjango-Unchained

1.1 Introduction

The Cjango-Unchained is a lightweight C++ web app framework. This documentation describes Cjango's internal programming interfaces, including ones which you will use when you write your original callback functions.

Chapter 2

Bug List

Member App::run (int port)

if the number of opened sockets exceeds SOMAXCONN, Cjango gets "apr_socket_recv: Connection reset by peer (54)" error and halts immediately. On Mac OS X (default: 128), check by "sysctl -a | grep somax" and change it by "sudo sysctl -w kern.ipc.somaxconn=2048"

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 6

Class Documentation

6.1 App Class Reference

Public Member Functions

- void add_route (std::string url_pattern, functor f)

  add a <url, function> mapping to the Router instance
- void worker (int clntSock, string strRequest)

  Runs in the work thread spawned to handle given httpRequest.
- void **reload_url_mappings** ()
- void add_monitored_dir (const std::string dir)

  Add a directory to monitor for dynamic loading feature.
- void **set_urls_json_dir** (std::string dir)
- std::string **get_urls_json_dir** () const
- void run (int port)

  starts Http server socket. Runs forever on the given port, setting up listening socket that polls for client connections. If there is a client connection, calls handle_request() to handle the request in a spawned thread.
- int handle_request (int socket)

  handles request for given socket, it calls recv() on the socket and spawns a new worker thread with worker() to handle the received request

Public Attributes

- Router **router**

6.1.1 Member Function Documentation

6.1.1.1 add_monitored_dir()

```
void App::add_monitored_dir (
            const std::string dir )
```

Add a directory to monitor for dynamic loading feature.

Parameters

std::string	dir

Returns

void

6.1.1.2 handle_request()

```
int App::handle_request (
            int clntSock )
```

handles request for given socket, it calls recv() on the socket and spawns a new worker thread with worker() to handle the received request

Parameters

int	clntSock

Returns

void

6.1.1.3 run()

```
void App::run (
            int port )
```

starts Http server socket. Runs forever on the given port, setting up listening socket that polls for client connections. If there is a client connection, calls handle_request() to handle the request in a spawned thread.

Parameters

int	port, port to start the server

Returns

void

Bug if the number of opened sockets exceeds SOMAXCONN, Cjango gets "apr_socket_recv: Connection reset by peer (54)" error and halts immediately. On Mac OS X (default: 128), check by "sysctl -a | grep somax" and change it by "sudo sysctl -w kern.ipc.somaxconn=2048"

6.1.1.4 worker()

```
void App::worker (
             int clntSock,
             string strRequest )
```

Runs in the work thread spawned to handle given httpRequest.

Parameters

int	clntSock, socket to send the response
std::string	strRequest, request in string

Returns

void

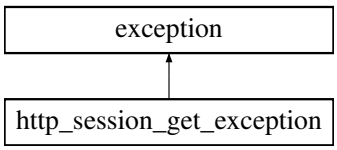
The documentation for this class was generated from the following files:

- app/app.hpp
- app/app.cpp

6.2 http_session_get_exception Class Reference

custom exception class in case HttpSession::get wants to hard-fail when key is not present

Inheritance diagram for http_session_get_exception:



6.2.1 Detailed Description

custom exception class in case HttpSession::get wants to hard-fail when key is not present

The documentation for this class was generated from the following file:

- http_parser/http_session.cpp

6.3 http::HttpRequest Class Reference

Public Member Functions

- **HttpRequest** (std::string path)
- HttpRequest (std::string, std::string, std::string, std::unordered_map< std::string, std::string >, std::unordered_map< std::string, std::string >, std::unordered_map< std::string, std::string >)

  HttpRequest Constructor that represents one HTTP request that can be received by a server.
- std::string get_method () const

  getter for http request's method. Ex: GET
- std::string get_path () const

  getter for http request's path. Ex: /abc.html
- std::string get_scheme () const

  getter for http request's scheme. Ex: HTTP/1.0
- unsigned long get_session_id ()

  getter for http request's session_id, persisted via cookie mechanism
- bool has_session_id ()

  check if http request has a session_id associated with it
- std::unordered_map< std::string, std::string > const & get_meta () const

  getter for http request's meta variables in the request header.
- std::unordered_map< std::string, std::string > const & get_parameters () const

  getter for http request's get/post parameters
- std::unordered_map< std::string, std::string > const & get_cookie () const

  getter for http request's cookie map
- std::shared_ptr< HttpSession > get_session ()

  getter for a session object pointer associated with the current http request If there is no HttpSession object associated with the http request, a new HttpSession object will be created for the request.

Static Public Member Functions

- static unsigned long xorshf96 ()

  Marsaglia's xorshf generator.

Public Attributes

- std::string **method**
- std::string **path**
- std::string **scheme**

Static Public Attributes

- static std::string **session_cookie_key** ="session"

6.3.1 Constructor & Destructor Documentation

6.3.1.1 HttpRequest()

```
http::HttpRequest::HttpRequest (
            std::string method,
            std::string path,
            std::string scheme,
            std::unordered_map< std::string, std::string > meta,
            std::unordered_map< std::string, std::string > params,
            std::unordered_map< std::string, std::string > cookie )
```

HttpRequest Constructor that represents one HTTP request that can be received by a server.

Parameters

method	http request method
path	http request path
scheme	http request version
meta	a map containing the key value pairs of http headers
params	a map containing the key value pairs of http parameters
cookie	a map containing the key value pairs of http header's cookie entries

Returns

HttpRequest

6.3.2 Member Function Documentation

6.3.2.1 get_cookie()

```
std::unordered_map< std::string, std::string > const & http::HttpRequest::get_cookie ( ) const
```

getter for http request's cookie map

Returns

a map of key value pairs for http request cookie

6.3.2.2 get_meta()

```
std::unordered_map< std::string, std::string > const & http::HttpRequest::get_meta ( ) const
```

getter for http request's meta variables in the request header.

Returns

a map of key value pairs for http request header

6.3.2.3 get_method()

```
std::string http::HttpRequest::get_method ( ) const
```

getter for http request's method. Ex: GET

Returns

a string indicating http request's method

6.3.2.4 get_parameters()

```
std::unordered_map< std::string, std::string > const & http::HttpRequest::get_parameters ( )
const
```

getter for http request's get/post parameters

Returns

a map of key value pairs for http request parameters

6.3.2.5 get_path()

```
std::string http::HttpRequest::get_path ( ) const
```

getter for http request's path. Ex: /abc.html

Returns

a path string

6.3.2.6 get_scheme()

```
std::string http::HttpRequest::get_scheme ( ) const
```

getter for http request's scheme. Ex: HTTP/1.0

Returns

a string indicating the current http protocol version of the request

6.3.2.7 get_session()

```
std::shared_ptr< http::HttpSession > http::HttpRequest::get_session ( )
```

getter for a session object pointer associated with the current http request If there is no HttpSession object associated with the http request, a new HttpSession object will be created for the request.

Returns

a shared_ptr of HttpSession object

6.3.2.8 xorshf96()

```
unsigned long http::HttpRequest::xorshf96 ( )  [static]
```

Marsaglia's xorshf generator.

Returns

a pseudo random unsigned long

The documentation for this class was generated from the following files:

- http_parser/http_request.hpp
- http_parser/http_request.cpp

6.4 http::HttpRequestBodyParser Class Reference

Public Member Functions

- HttpRequestBodyParser ()

  HttpRequestBodyParser constructor that contains a default UrlEncodedFormParser for form parsing.
- **HttpRequestBodyParser** (std::vector< UrlEncodedFormParser >)
- std::unordered_map< std::string, std::string > parse (std::istream &, std::string, int)

  given a http request body in istream containing http request parameters, it will parse the content of istream and loads the http request parameters to an unordered_map

Public Attributes

- std::vector< UrlEncodedFormParser > url_encoded_form_parsers

6.4.1 Constructor & Destructor Documentation

6.4.1.1 HttpRequestBodyParser()

```
http::HttpRequestBodyParser::HttpRequestBodyParser ( )
```

HttpRequestBodyParser constructor that contains a default UrlEncodedFormParser for form parsing.

Returns

HttpRequestBodyParser

6.4.2 Member Function Documentation

6.4.2.1 parse()

```
std::unordered_map< std::string, std::string > http::HttpRequestBodyParser::parse (
            std::istream & input_stream,
            std::string content_type,
            int content_leng )
```

given a http request body in istream containing http request parameters, it will parse the content of istream and loads the http request parameters to an unordered_map

Returns

an unordered_map containing the parameters for HttpRequest

6.4.3 Member Data Documentation

6.4.3.1 url_encoded_form_parsers

```
std::vector<UrlEncodedFormParser> http::HttpRequestBodyParser::url_encoded_form_parsers
```

parser that parses application/x-www-form-urlencoded

The documentation for this class was generated from the following files:

- http_parser/http_request_body_parser.hpp
- http_parser/http_request_body_parser.cpp

6.5 http::HttpRequestLine Class Reference

Public Member Functions

- HttpRequestLine (std::string action, std::string uri, std::string protocolVersion, std::unordered_map< std::string, std::string > parameters)

  HttpRequestLine constructor that represents the first line of a http request as well as the paramters of a get HttpRequest.
- HttpRequestLine (std::string action, std::string uri, std::string protocolVersion)

  HttpRequestLine constructor that represents the first line of a http request.

Public Attributes

- std::string **action**
- std::string **uri**
- std::unordered_map< std::string, std::string > **parameters**
- std::string **protocolVersion**

6.5.1 Constructor & Destructor Documentation

6.5.1.1 HttpRequestLine() [1/2]

```
http::HttpRequestLine::HttpRequestLine (
            std::string action,
            std::string uri,
            std::string protocolVersion,
            std::unordered_map< std::string, std::string > parameters )
```

HttpRequestLine constructor that represents the first line of a http request as well as the paramters of a get HttpRequest.

Parameters

action	method of a http request
uri	path of a http request
parameters	a map containing the parameters of a http get request
protocolVersion	version of the http protocol for the request

6.5.1.2 HttpRequestLine() [2/2]

```
http::HttpRequestLine::HttpRequestLine (
            std::string action,
```

```
            std::string uri,
            std::string protocolVersion )
```

HttpRequestLine constructor that represents the first line of a http request.

Parameters

action	method of a http request
uri	path of a http request
protocolVersion	version of the http protocol for the request

The documentation for this class was generated from the following files:

- http_parser/http_request_line.hpp
- http_parser/http_request_line.cpp

6.6 http::HttpRequestParser Class Reference

Public Member Functions

- **HttpRequestParser** (HttpRequestBodyParser)
- HttpRequest parse_request_line_and_headers (std::istream &input_stream)

  given an input_stream containing a http request, parses the request line and headers
- std::unordered_map< std::string, std::string > parse_body (std::istream &, std::string, int)

  helper method that parses a http request's body
- HttpRequest parse (std::istream &input_stream)

  given an input_stream containing a http request, parses the request line, headers, and body

Public Attributes

- HttpRequestBodyParser **body_parser**
- HttpStreamReader **reader**
- UrlEncodedFormParser **url_encoded_form_parser**

6.6.1 Member Function Documentation

6.6.1.1 parse()

```
http::HttpRequest http::HttpRequestParser::parse (
            std::istream & input_stream )
```

given an input_stream containing a http request, parses the request line, headers, and body

Parameters

input_stream	a stream containing a http request string

Returns

a http request object representing the data from the input_stream

6.6.1.2 parse_body()

```
std::unordered_map< std::string, std::string > http::HttpRequestParser::parse_body (
            std::istream & input_stream,
            std::string content_type,
            int content_leng )
```

helper method that parses a http request's body

Parameters

input_stream	a stream containing http request's body
content_type	the content type of the http request as specified in its headers
content_leng	the content length of the http request as specified in its headers

Returns

a map containing key value pairs of the http request's cookie

6.6.1.3 parse_request_line_and_headers()

```
http::HttpRequest http::HttpRequestParser::parse_request_line_and_headers (
            std::istream & input_stream )
```

given an input_stream containing a http request, parses the request line and headers

Parameters

input_stream	a stream containing a http request string

Returns

HttpRequest with populated http request headers and an empty body

The documentation for this class was generated from the following files:

- http_parser/http_request_parser.hpp
- http_parser/http_request_parser.cpp

6.7 http::HttpResponse Class Reference

Public Member Functions

- HttpResponse (std::string)

  HttpResponse constructor with a 200 status code.
- HttpResponse (std::string, HttpRequest &)

  HttpResponse constructor with a 200 status code and content type text/html.
- HttpResponse (std::string, std::string)

  HttpResponse constructor with a 200 status code.
- HttpResponse (int)

  HttpResponse constructor with content type text/html.
- std::string to_string ()

  returns a well-formated string version of http response compliant with http/1.0 protocol
- void set_cookie (std::string, std::string)

  given a key value pair, inserts the pair to request's cookie

Static Public Member Functions

- static HttpResponse render_to_response (std::string)

  given a file path, generate a http response. The generated HttpResponse will have text/html as its content-type and a status code of 200.
- static HttpResponse render_to_response (std::string, std::string)

  given a file path and content type of the file, generate a http response The generated HttpResponse will have text/html as its content-type and a status code of 200.
- static HttpResponse render_to_response (std::string, HttpRequest &)

  given a file path and a http request, generate a http response. The generated HttpResponse will have text/html as its content-type and a status code of 200.
- static HttpResponse render_to_response (std::string, std::string, HttpRequest &)

  given a file path, content type of the file, and a http request, create a http response
- static std::string get_template (std::string path)

  given a file path, generate a string containing file's data

Public Attributes

- std::string **content**
- int **status_code** = 200
- std::string **reason_phrase** = "OK"
- std::string **http_version** = "HTTP/1.0"
- std::string **content_type** = "text/html"
- std::unordered_map< std::string, std::string > **headers**

Static Public Attributes

- static std::unordered_map< int, std::string > **code_to_reason**
- static std::string **templates_root**

6.7.1 Constructor & Destructor Documentation

6.7.1.1 HttpResponse() [1/4]

```
http::HttpResponse::HttpResponse (
            std::string content )
```

HttpResponse constructor with a 200 status code.

Parameters

content	body of the http response

Returns

HttpResponse

6.7.1.2 HttpResponse() [2/4]

```
http::HttpResponse::HttpResponse (
            std::string content,
            http::HttpRequest & request )
```

HttpResponse constructor with a 200 status code and content type text/html.

Parameters

content	body of the http response
request	a HttpRequest object which corresponds to the current http response

Returns

HttpResponse

6.7.1.3 HttpResponse() [3/4]

```
http::HttpResponse::HttpResponse (
            std::string content,
            std::string content_type )
```

HttpResponse constructor with a 200 status code.

Parameters

content	body of the http response
content_type	content type of the http response

Returns

HttpResponse

6.7.1.4 HttpResponse() [4/4]

```
http::HttpResponse::HttpResponse (
            int status_code )
```

HttpResponse constructor with content type text/html.

Parameters

status_code	status code of the http response

Returns

HttpResponse

6.7.2 Member Function Documentation

6.7.2.1 get_template()

```
std::string http::HttpResponse::get_template (
            std::string path )  [static]
```

given a file path, generate a string containing file's data

Parameters

path	path to a file whose content will be returned as a string path should be relative to HttpResponse::templates_root

Returns

a string representing the file's content

6.7.2.2 render_to_response() [1/4]

```
http::HttpResponse http::HttpResponse::render_to_response (
            std::string path )  [static]
```

given a file path, generate a http response. The generated HttpResponse will have text/html as its content-type and a status code of 200.

Parameters

path	path to a file whose content will be in the http response's body. path should be relative to HttpResponse::templates_root

Returns

HttpResponse

6.7.2.3 render_to_response() [2/4]

```
http::HttpResponse http::HttpResponse::render_to_response (
            std::string path,
            std::string content_type )  [static]
```

given a file path and content type of the file, generate a http response The generated HttpResponse will have text/html as its content-type and a status code of 200.

Parameters

path	path to a file whose content will be in the http response's body. path should be relative to HttpResponse::templates_root
content_type	content type of the http response whose value is set in http response's header

Returns

HttpResponse

6.7.2.4 render_to_response() [3/4]

```
http::HttpResponse http::HttpResponse::render_to_response (
            std::string file,
            http::HttpRequest & request )  [static]
```

given a file path and a http request, generate a http response. The generated HttpResponse will have text/html as its content-type and a status code of 200.

Parameters

path	path to a file whose content will be in the http response's body. path should be relative to HttpResponse::templates_root
request	a HttpRequest object which corresponds to the current http response

Returns

HttpResponse

6.7.2.5 render_to_response() [4/4]

```
http::HttpResponse http::HttpResponse::render_to_response (
            std::string path,
            std::string content_type,
            http::HttpRequest & request )  [static]
```

given a file path, content type of the file, and a http request, create a http response

Parameters

path	path to a file whose content will be in the http response's body. path should be relative to HttpResponse::templates_root
content_type	content type of the http response whose value is set in http response's header
request	a HttpRequest object which corresponds to the current http response

Returns

HttpResponse

6.7.2.6 set_cookie()

```
void http::HttpResponse::set_cookie (
            std::string key,
            std::string value )
```

given a key value pair, inserts the pair to request's cookie

Parameters

key	cookie key
value	cookie value

The documentation for this class was generated from the following files:

- http_parser/http_response.hpp
- http_parser/http_response.cpp

6.8 http::HttpSession Class Reference

Public Member Functions

- HttpSession ()

  HttpSession constructor that initializes a reader-writer lock.
- std::string get (std::string)

  given a key, return its value from the session map. This method is thread safe
- void set (std::string, std::string)

  given a (key, value pair), insert it to the session map. This method is thread safe

6.8.1 Member Function Documentation

6.8.1.1 get()

```
std::string http::HttpSession::get (
            std::string key )
```

given a key, return its value from the session map. This method is thread safe

Parameters

key	key used to retrieve value stored in the session map

Returns

if key is found, the value corresponding to the key. If key is not found, an empty string

The documentation for this class was generated from the following files:

- http_parser/http_session.hpp
- http_parser/http_session.cpp

6.9 http::HttpStreamReader Class Reference

Public Member Functions

- std::string get_next_line (std::istream &input_stream)

*given an input stream, returns the current line deliminted by '
'*

- std::string to_string (std::istream &input_stream)

  given an input stream, converts it to string
- std::string **read_util** (std::istream &input_stream, int character)
- std::string read (std::istream &input_stream, int length)

  given an input stream and number of length bytes we want to read from the input stream, converts number of length bytes + 1 from input stream to a string
- void eat_white_space (std::istream &input_stream)

  removes white spaces from input stream

Static Public Attributes

- static const int **carraige_return** = '\r'
- static const int **line_feed** = '\n'

6.9.1 Member Function Documentation

6.9.1.1 eat_white_space()

```
void http::HttpStreamReader::eat_white_space (
            std::istream & input_stream )
```

removes white spaces from input stream

Parameters

input_stream	is the input stream we are currently processing

6.9.1.2 get_next_line()

```
std::string http::HttpStreamReader::get_next_line (
            std::istream & input_stream )
```

given an input stream, returns the current line deliminted by '
'

Parameters

input_stream	is the input stream we are currently processing

Returns

a string representing the next line in the input_stream

6.9.1.3 read()

```
std::string http::HttpStreamReader::read (
            std::istream & input_stream,
            int length )
```

given an input stream and number of length bytes we want to read from the input stream, converts number of length bytes + 1 from input stream to a string

Parameters

input_stream	is the input stream we are currently processing
length	number of bytes we would like to read from the input stream

6.9.1.4 to_string()

```
std::string http::HttpStreamReader::to_string (
            std::istream & input_stream )
```

given an input stream, converts it to string

Parameters

input_stream	is the input stream we are currently processing

Returns

the entire input stream as a string

The documentation for this class was generated from the following files:

- http_parser/http_stream_reader.hpp
- http_parser/http_stream_reader.cpp

6.10 MSocket Class Reference

Public Member Functions

- **MSocket** (int sock)

- int **socket** ()
- void **close** ()
- int **set_reusable** ()
- int **set_nonblocking** ()
- int **bind** (int port)
- int **listen** (int maxconn)
- std::shared_ptr< MSocket > **accept** ()
- int **recv** (void *buf, size_t len)
- int **send** (void *buf, size_t len)
- int **send** (std::string str)

The documentation for this class was generated from the following files:

- app/msocket.hpp
- app/msocket.cpp

6.11 Router Class Reference

Public Member Functions

- **Router** (URLmap routes)
- int **nr_patterns** () const
- void **erase_all_patterns** ()
- void add_route (std::string url_pattern, functor f)

  Add a mapping from url_pattern to a callback into patterns_list and pattern_to_callback. if the given url pattern is already set, this will overwrite the pattern by the new callback.
- void **set_static_dir** (const std::string dir)
- std::string **get_static_dir** () const
- std::string resolve (http::HttpRequest) const

  Called in get_http_response() for mapping a given request's path to an url pattern. If no pattern matches the given request's path, RouterException::INVALID_URL will be returned. If starting from "/static/", RouterException::STAT↩ IC_FILE_SERVED will be returned. Both enums are handled in get_http_response().
- http::HttpResponse get_http_response (http::HttpRequest)

  find and execute a callback corresponding to the given request

6.11.1 Member Function Documentation

6.11.1.1 add_route()

```
void Router::add_route (
            std::string url_pattern,
            functor f )
```

Add a mapping from url_pattern to a callback into patterns_list and pattern_to_callback. if the given url pattern is already set, this will overwrite the pattern by the new callback.

Parameters

url_pattern	(e.g. "/diary/[0-9]{4}")

Returns

void

6.11.1.2 get_http_response()

```
http::HttpResponse Router::get_http_response (
            http::HttpRequest request )
```

find and execute a callback corresponding to the given request

Returns

an HttpResponse object built by the callback

6.11.1.3 resolve()

```
std::string Router::resolve (
            http::HttpRequest request ) const
```

Called in get_http_response() for mapping a given request's path to an url pattern. If no pattern matches the given request's path, RouterException::INVALID_URL will be returned. If starting from "/static/", RouterException::STATIC_FILE_SERVED will be returned. Both enums are handled in get_http_response().

Returns

a corresponding url pattern such as "diary/[0-9]{4}/[0-9]{2}/"

The documentation for this class was generated from the following files:

- routing/router.hpp
- routing/router.cpp

6.12 Selector Struct Reference

Public Member Functions

- **Selector** (int sock)
- void **set_server** (int sock)
- void **fd_zero** ()
- void **fd_set2** (int sock)
- bool **fd_isset** (int sock, int m=0)
- void **fd_clr** (int sock)
- bool **server_isset** (int m=0)
- int **select** ()
- void **report** (int m=0)

Public Attributes

- int **servSock** = -1
- fd_set **allSocks**
- fd_set **retSocks**
- struct timeval **timeout**
- int **maxSock** = -1

The documentation for this struct was generated from the following file:

- app/selector.hpp

6.13 http::UrlEncodedFormParser Class Reference

Public Member Functions

- bool can_parse_content_type (std::string content_type)

  checks if cotent_type in question can be parsed by the current parser
- char **charToInt** (char)
- char **strToBin** (char *pString)
- std::string **urlDecode** (const std::string &str)
- std::unordered_map< std::string, std::string > get_parameter (std::istream &input_stream, int content_leng)

  given current input stream which contains get or post params encoded as a single string, parses the stream into a map
- std::vector< std::string > split (std::string str, char delimiter)

  splits a string by a character delimiter and returns the result as a vector

Public Attributes

- HttpStreamReader **input_stream_reader**
- std::string **supported_content_type** = "application/x-www-form-urlencoded"

6.13.1 Member Function Documentation

6.13.1.1 can_parse_content_type()

```
bool http::UrlEncodedFormParser::can_parse_content_type (
            std::string content_type )
```

checks if cotent_type in question can be parsed by the current parser

Parameters

content_type	a string representing the content type of a http request

Returns

: a boolean indicating whether content_type is supported

6.13.1.2 get_parameter()

```
std::unordered_map< std::string, std::string > http::UrlEncodedFormParser::get_parameter (
            std::istream & input_stream,
            int content_leng )
```

given current input stream which contains get or post params encoded as a single string, parses the stream into a map

Parameters

input_stream	is the input stream we are currently processing which contains get or post params
content_leng	length of the input_stream

Returns

a map containing http request's parameters

6.13.1.3 split()

```
std::vector< std::string > http::UrlEncodedFormParser::split (
            std::string str,
            char delimiter )
```

splits a string by a character delimiter and returns the result as a vector

Parameters

str	string to be split
delimiter	a character delimiter to split the string by

Returns

the split string in a vector

The documentation for this class was generated from the following files:

- http_parser/url_encoded_form_parser.hpp
- http_parser/url_encoded_form_parser.cpp

Chapter 7

File Documentation

7.1 http_parser/http_request.cpp File Reference

HttpRequest class implementation.

```
#include "http_request.hpp"
#include <stdio.h>
#include <stdlib.h>
#include "../app/externs.hpp"
```

Variables

- std::string **http_logger_name** = "http_request"

7.1.1 Detailed Description

HttpRequest class implementation.

7.2 http_parser/http_request.hpp File Reference

HttpRequest class declaration.

```
#include <unordered_map>
#include <iostream>
#include <fstream>
#include <string>
#include <memory>
#include "http_session.hpp"
#include <pthread.h>
```

Classes

- class http::HttpRequest

Functions

- std::ostream & http::operator<< (std::ostream &Str, HttpRequest const &v)

  overloading << for HttpRequest object that enables HttpRequest to be written to a output stream

7.2.1 Detailed Description

HttpRequest class declaration.

7.2.2 Function Documentation

7.2.2.1 operator<<()

```
std::ostream & http::operator<< (
            std::ostream & Str,
            HttpRequest const & v )
```

overloading << for HttpRequest object that enables HttpRequest to be written to a output stream

Returns

std::ostream& enabling HttpRequest to be written to a output stream

7.3 http_parser/http_request_body_parser.cpp File Reference

HttpRequestBodyParser class implementation.

```
#include "http_request_body_parser.hpp"
#include "../app/externs.hpp"
```

7.3.1 Detailed Description

HttpRequestBodyParser class implementation.

7.4 http_parser/http_request_body_parser.hpp File Reference

HttpRequestBodyParser class declaration.

```
#include <vector>
#include <unordered_map>
#include <iostream>
#include <fstream>
#include "url_encoded_form_parser.hpp"
```

Classes

- class http::HttpRequestBodyParser

7.4.1 Detailed Description

HttpRequestBodyParser class declaration.

7.5 http_parser/http_request_line.cpp File Reference

HttpRequestLine class implementation.

```
#include "http_request_line.hpp"
```

7.5.1 Detailed Description

HttpRequestLine class implementation.

7.6 http_parser/http_request_line.hpp File Reference

HttpRequestLine class declaration.

```
#include <unordered_map>
#include <string>
```

Classes

- class http::HttpRequestLine

7.6.1 Detailed Description

HttpRequestLine class declaration.

7.7 http_parser/http_request_parser.cpp File Reference

HttpRequestParser class implementation.

```
#include "http_request_parser.hpp"
#include "../app/externs.hpp"
#include <algorithm>
#include <string>
#include <iostream>
```

7.7.1 Detailed Description

HttpRequestParser class implementation.

7.8 http_parser/http_request_parser.hpp File Reference

HttpRequestParser class declaration.

```
#include "url_encoded_form_parser.hpp"
#include "http_request_body_parser.hpp"
#include "http_request.hpp"
#include "http_request_line.hpp"
#include <unordered_map>
#include <iostream>
#include <fstream>
```

Classes

- class http::HttpRequestParser

7.8.1 Detailed Description

HttpRequestParser class declaration.

7.9 http_parser/http_response.cpp File Reference

HttpResponse class implementation.

```
#include "http_response.hpp"
#include <fstream>
#include <streambuf>
#include "../app/externs.hpp"
```

Functions

- std::string get_reason_phrase (int status_code)

  helper method that translates http status code to status message

7.9.1 Detailed Description

HttpResponse class implementation.

7.9.2 Function Documentation

7.9.2.1 get_reason_phrase()

```
std::string get_reason_phrase (
            int status_code )
```

helper method that translates http status code to status message

Parameters

status_code	valid http response status code defined in http/1.0 protocol

Returns

the corresponding status message

7.10 http_parser/http_response.hpp File Reference

HttpResponse class declaration.

```
#include <unordered_map>
#include <string>
#include "http_request.hpp"
```

Classes

- class http::HttpResponse

Functions

- std::ostream & http::operator<< (std::ostream &Str, HttpResponse const &v)

  overloads << function for debugging purposes: convenience of printing out http response

7.10.1 Detailed Description

HttpResponse class declaration.

7.11 http_parser/http_session.cpp File Reference

HttpSession class implementation.

```
#include "http_session.hpp"
```

Classes

- class http_session_get_exception

  custom exception class in case HttpSession::get wants to hard-fail when key is not present

Variables

- http_session_get_exception **sessionex**

7.11.1 Detailed Description

HttpSession class implementation.

7.12 http_parser/http_session.hpp File Reference

HttpSession class declaration. This class is thread-safe.

```
#include <unordered_map>
#include <iostream>
#include <fstream>
#include <string>
#include <mutex>
#include <exception>
#include <pthread.h>
```

Classes

- class http::HttpSession

7.12.1 Detailed Description

HttpSession class declaration. This class is thread-safe.

7.13 http_parser/http_stream_reader.cpp File Reference

HttpStreamReader class implementation.

```
#include "http_stream_reader.hpp"
#include "../app/externs.hpp"
```

7.13.1 Detailed Description

HttpStreamReader class implementation.

7.14 http_parser/http_stream_reader.hpp File Reference

HttpStreamReader class declaration.

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
```

Classes

- class http::HttpStreamReader

7.14.1 Detailed Description

HttpStreamReader class declaration.

7.15 http_parser/url_encoded_form_parser.cpp File Reference

UrlEncodedFormParser class implementation.

```
#include "url_encoded_form_parser.hpp"
#include "../app/externs.hpp"
#include <ctype.h>
#include <stdlib.h>
```

7.15.1 Detailed Description

UrlEncodedFormParser class implementation.

7.16 http_parser/url_encoded_form_parser.hpp File Reference

UrlEncodedFormParser class declaration.

```
#include "http_stream_reader.hpp"
#include <string>
#include <unordered_map>
#include <iostream>
#include <fstream>
#include <exception>
#include <vector>
#include <sstream>
```

Classes

- class http::UrlEncodedFormParser

7.16.1 Detailed Description

UrlEncodedFormParser class declaration.

7.17 routing/router.hpp File Reference

Router class definition and related consts/aliases.

```
#include <stdio.h>
#include <functional>
#include <unordered_map>
#include <vector>
#include <string>
#include <stdexcept>
#include <algorithm>
#include <cjango>
```

Classes

- class Router

Typedefs

- using **functor** = std::function< http::HttpResponse(http::HttpRequest)>
- using **URLmap** = std::unordered_map< std::string, functor >

Enumerations

- enum **RouterException** { **INVALID_URL**, **STATIC_FILE_SERVED** }

Variables

- const std::string **route_logger_name** = "route"

7.17.1 Detailed Description

Router class definition and related consts/aliases.

Author

caprice-j

Date

2017.04.17

Version

0.1

Warning

this example may include some wrong descriptions

Index