clas-digital

Generated by Doxygen 1.8.13

Contents

Chapter 1

Todo List

Member parse_query (std::map< std::string, std::string > &map, std::string_view const &query)

Fix multiple key collisions and find a way to manage that eg collections=RXBADE&collections=RXFGABAAD

Member server::server (unsigned short port, const char *cert, const char *key)

{Implement some more stuff}

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 http_request Class Reference

Parses and stores informations about an HTTP request.

`#include <httpparser.hpp>`

Public Member Functions

- http_request (const char *asyncReadBuf, size_t bytes)
- void print_request ()
- const std::string_view & GetHeaders (std::string key)
- const std::string_view GetQueryParams (std::string key)
- const std::string_view & GetMethod ()
- const std::string_view & GetURL ()
- const std::string_view & GetPath ()
- const std::string_view & GetQuery ()
- bool IsHealthy ()
- const void * GetBody ()
- unsigned long GetBodySize ()

4.1.1 Detailed Description

Parses and stores informations about an HTTP request.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 http_request()

```
http_request::http_request (
            const char * asyncReadBuf,
            size_t bytes )
```

Creates an http request given a buffer returned by an read operation and the size of the buffer. The class automatically parses query parameters path, url, method body size and pointer to body. While doing so it does not copy anything therefore it is up to the programmer to ensure the buffer given to the http_request constructor remains unchanged while the http_request is in use. If there is an error while parsing the http file the IsHealthy() function will return false

Parameters

in	*asyncReadBuf*	The Read buffer containing the informations about the http request.
in	*bytes*	The length of the buffer.

4.1.3 Member Function Documentation

4.1.3.1 GetBody()

```
const void * http_request::GetBody ( )
```

Returns the body of the http message, this pointer is not owned by the class, therefore it should not be modified at all.

Returns

Returns the immutable body buffer

4.1.3.2 GetBodySize()

```
unsigned long http_request::GetBodySize ( )
```

Returns the body size stored inside the class

Returns

The body size of the class.

4.1.3.3 GetHeaders()

```
const std::string_view & http_request::GetHeaders (
            std::string key )
```

Returns the headers parsed from the http request.

Parameters

key	The name of the http parameter to search for

Returns

The header fitting to the key if there is one

4.1.3.4 GetMethod()

```
const std::string_view & http_request::GetMethod ( )
```

Returns the method used in the Request, 'POST' or 'GET' etc.

Returns

The method used for the message

4.1.3.5 GetPath()

```
const std::string_view & http_request::GetPath ( )
```

Returns the mean path from the url /search?query=hallo will return /search.

Returns

The path extracted from the url.

4.1.3.6 GetQuery()

```
const std::string_view & http_request::GetQuery ( )
```

Returns the whole unparsed query string. Request to /search?query=hallo will return query=hallo

Returns

The whole query string

4.1.3.7 GetQueryParams()

```
const std::string_view http_request::GetQueryParams (
            std::string key )
```

Returns the query parameter parsed from the url given the key to search for.

Parameters

key	The variable name in the query parameter

Returns

The query parameter fitting to a key.

4.1.3.8 GetURL()

```
const std::string_view & http_request::GetURL ( )
```

Returns the URL parsed from the request

Returns

The full url

4.1.3.9 IsHealthy()

```
bool http_request::IsHealthy ( )
```

Returns if the http message parsed is healthy and does not miss something

Returns

Returns if the http message is healthy

4.1.3.10 print_request()

```
void http_request::print_request ( )
```

Prints all the information gathered in the constructor about the http request, mainly used for debug purposes

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

- src/server/httpparser.hpp
- src/server/httpparser.cpp

4.2 server Class Reference

The Basic Multithreaded HTTPS Server handles all requests.

`#include <server.hpp>`

Public Member Functions

- server (unsigned short port, const char *cert, const char *key)
- void run (unsigned int threads=0)

4.2.1 Detailed Description

The Basic Multithreaded HTTPS Server handles all requests.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 server()

```
server::server (
            unsigned short port,
            const char * cert,
            const char * key )
```

Constructs the server from a given port a certificate file path and a key file path.

Todo {Implement some more stuff}

Parameters

port	The port to let the server listen to
cert	The certificate file path to open
key	The key file path to open

4.2.3 Member Function Documentation

4.2.3.1 run()

```
void server::run (
            unsigned int threads = 0 )
```

Runs the server with the given number of threads, if 0 is specified runs on as many threads as there are cores in the system.

Parameters

threads	The number of threads to run the server on.

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

- src/server/server.hpp
- src/server/server.cpp

4.3 session Class Reference

Handles a TCP Session uses async reads and writes to perform requests The session uses asynchronous read and write operations to communicate with the client. Only used for HTTPS clients at the moment may be used for other purposes later on.

Public Member Functions

- session (boost::asio::io_service &io_service, boost::asio::ssl::context &context)
- ssl_socket::lowest_layer_type & socket ()
- void start ()
- void handle_handshake (const boost::system::error_code &error)
- void handle_read (const boost::system::error_code &error, size_t bytes_transferred)

4.3.1 Detailed Description

Handles a TCP Session uses async reads and writes to perform requests The session uses asynchronous read and write operations to communicate with the client. Only used for HTTPS clients at the moment may be used for other purposes later on.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 session()

```
session::session (
            boost::asio::io_service & io_service,
            boost::asio::ssl::context & context )
```

Creates a new TCP session and pushes the work into the io_service.

Parameters

io_service	The io_service to read from and write to
context	The ssl context used to encrypt the connection

4.3.3 Member Function Documentation

4.3.3.1 handle_handshake()

```
void session::handle_handshake (
            const boost::system::error_code & error )
```

The asynchronous called function that handles the ssl handshake and startes the first asynchronous read on the connection

Parameters

error	The error returned by the system if the handshake fails.

4.3.3.2 handle_read()

```
void session::handle_read (
            const boost::system::error_code & error,
            size_t bytes_transferred )
```

Asynchronous read operation used to read data from the client.

Parameters

error	The error returned from the read function
bytes_transferred	The number of bytes transferred into the buffer

4.3.3.3 socket()

```
ssl_socket::lowest_layer_type & session::socket ( )
```

Returns the implementation of the socket used for system specific functions and APIs.

Returns

socket implementation

4.3.3.4 start()

```
void session::start ( )
```

Asynchronous starting the handshake.

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

- src/server/server.cpp

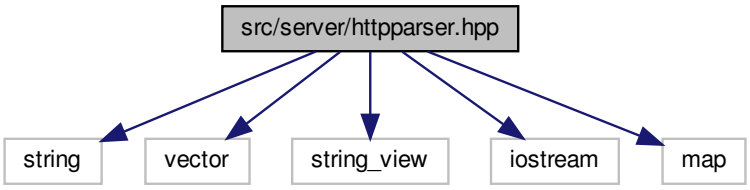
Chapter 5

File Documentation

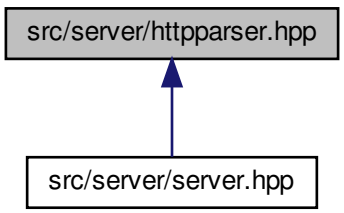
5.1 src/server/httpparser.hpp File Reference

```
#include <string>
#include <vector>
#include <string_view>
#include <iostream>
#include <map>
```

Include dependency graph for httpparser.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class http_request

  Parses and stores informations about an HTTP request.

Functions

- void parse_query (std::map< std::string, std::string > &map, std::string_view const &query)
- unsigned char tochar (char hi, char lo)

5.1.1 Detailed Description

Classes for parsing requests and constructing reponses in the http format

This header defines the interface to the classes http_request and http_reponse

5.1.2 Function Documentation

5.1.2.1 parse_query()

```
void parse_query (
            std::map< std::string, std::string > & queryMap,
            std::string_view const & query )
```

Parses a given string in the xhttp html format and puts the results in the given map.

Parameters

out	*map*	The map to put the results from the parsing into
in	*query*	The string to parse into the map.

Todo Fix multiple key collisions and find a way to manage that eg collections=RXBADE&collections=RXFGABAAD

5.1.2.2 tochar()

```
unsigned char tochar (
            char hi,
            char lo )
```

Transform two hexadecimal digits into the character they represent eg tochar('2','0') = 0x20.

Parameters

in	*hi*	The high byte of the character
in	*lo*	The low byte of the character

Returns

The calculated byte