

clas-digital

Generated by Doxygen 1.8.13

Contents

1	Todo List	1
2	Class Index	3
2.1	Class List	3
3	Class Documentation	5
3.1	http_request Class Reference	5
3.1.1	Detailed Description	5
3.1.2	Constructor & Destructor Documentation	5
3.1.2.1	http_request()	5
3.1.3	Member Function Documentation	6
3.1.3.1	GetHeaders()	6
3.1.3.2	GetMethod()	6
3.1.3.3	GetPath()	6
3.1.3.4	GetQuery()	7
3.1.3.5	GetQueryParams()	7
3.1.3.6	GetURL()	7
3.1.3.7	IsHealthy()	7
3.1.3.8	print_request()	8
3.2	server Class Reference	8
3.2.1	Detailed Description	8
3.2.2	Constructor & Destructor Documentation	8
3.2.2.1	server()	8
3.2.3	Member Function Documentation	9
3.2.3.1	run()	9
3.3	session Class Reference	9
3.3.1	Detailed Description	9
3.3.2	Constructor & Destructor Documentation	9
3.3.2.1	session()	9
3.3.3	Member Function Documentation	10
3.3.3.1	handle_handshake()	10
3.3.3.2	handle_read()	10
3.3.3.3	socket()	10
3.3.3.4	start()	11

Chapter 1

Todo List

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:

http_request	Parses and stores informations about an HTTP request	5
server	The Basic Multithreaded HTTPS Server handles all requests	8
session	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	9

Chapter 3

Class Documentation

3.1 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](#) ()

3.1.1 Detailed Description

Parses and stores informations about an HTTP request.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 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	<i>asyncReadBuf</i>	The Read buffer containing the informations about the http request.
in	<i>bytes</i>	The length of the buffer.

3.1.3 Member Function Documentation

3.1.3.1 GetHeaders()

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

Returns the headers parsed from the http request.

Parameters

<i>key</i>	The name of the http parameter to search for
------------	----------------------------------------------

Returns

The header fitting to the key if there is one

3.1.3.2 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

3.1.3.3 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.

3.1.3.4 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

3.1.3.5 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.

3.1.3.6 GetURL()

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

Returns the URL parsed from the request

Returns

The full url

3.1.3.7 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

3.1.3.8 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

3.2 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)

3.2.1 Detailed Description

The Basic Multithreaded HTTPS Server handles all requests.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 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

<i>port</i>	The port to let the server listen to
<i>cert</i>	The certificate file path to open
<i>key</i>	The key file path to open

3.2.3 Member Function Documentation

3.2.3.1 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

<i>threads</i>	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

3.3 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)

3.3.1 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.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 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

<i>io_service</i>	The io_service to read from and write to
<i>context</i>	The ssl context used to encrypt the connection

3.3.3 Member Function Documentation**3.3.3.1 handle_handshake()**

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

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

Parameters

<i>error</i>	The error returned by the system if the handshake fails.
--------------	----------------------------------------------------------

3.3.3.2 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

<i>error</i>	The error returned from the read function
<i>bytes_transferred</i>	The number of bytes transferred into the buffer

3.3.3.3 socket()

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

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

Returns

socket implementation

3.3.3.4 start()

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

Asynchronous starting the handshake.

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

- src/server/server.cpp

