clas-digital

Generated by Doxygen 1.8.13

Contents

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:

Chapter 3

Class Documentation

3.1 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)
- void handle_accept (session *new_session, const boost::system::error_code &error)

3.1.1 Detailed Description

The Basic Multithreaded HTTPS Server handles all requests.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 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

3.1.3 Member Function Documentation

3.1.3.1 handle_accept()

```
void server::handle_accept (
            session * new_session,
            const boost::system::error_code & error )
```

The asynchronous handle accept callback, registers if there are new clients available and creates a new session from them

Parameters

new_session	The new session to start.
error	The error which is maybe thrown by the system as a result of the accept operation

3.1.3.2 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

3.2 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)

3.2.1 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.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 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

3.2.3 Member Function Documentation

3.2.3.1 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.

3.2.3.2 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

3.2.3.3 socket()

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

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

Returns

socket implementation

3.2.3.4 start()

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

Asynchronous starting the handshake.

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

- src/server/server.cpp