clas-digital

Generated by Doxygen 1.8.13

Contents

Chapter 1

Todo List

Member http_response::SendWithEOM ()

Improve the way the function Sends stuff which is kinda slow at the moment

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

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 CBook Class Reference

Public Member Functions

- CBook (std::string sPath)
- std::string getPath ()

  getter function to return the path to the directory of a book
- std::string getOcrPath ()
- std::string getKey ()
- bool getOcr ()
- std::map< std::string, int > & getMapWords ()
- CMetadata & getMetadata ()
- std::vector< std::string > getCollections ()
- std::string getAuthor ()
- int getDate ()
- void setOcr (bool bOcr)
- void setPath (std::string sPath)
- void createMapWords ()

  checks whether book already has map of words, if not it create them

4.1.1 Constructor & Destructor Documentation

4.1.1.1 CBook()

```
CBook::CBook (
            std::string sPath )
```

Constructor

Parameters

in	*sPath*	Path to book
in	*map*	map of words in book

4.1.2 Member Function Documentation

4.1.2.1 createMapWords()

```
void CBook::createMapWords ( )
```

checks whether book already has map of words, if not it create them

Create a map of all word of this book

4.1.2.2 getAuthor()

```
std::string CBook::getAuthor ( )
```

Returns

lastName, or Name of author

4.1.2.3 getCollections()

```
std::vector< std::string > CBook::getCollections ( )
```

Returns

vector with all collections this book is in

4.1.2.4 getDate()

```
int CBook::getDate ( )
```

Returns

date or -1 if date does not exists or is currupted

4.1.2.5 getKey()

```
std::string CBook::getKey ( )
```

Returns

Key of the book, after extracting it from the path

4.1.2.6 getMapWords()

```
std::map< std::string, int > & CBook::getMapWords ( )
```

Returns

map of all words in book

4.1.2.7 getMetadata()

```
CMetadata & CBook::getMetadata ( )
```

Returns

info.json of book

4.1.2.8 getOcr()

```
bool CBook::getOcr ( )
```

Returns

Boolean, whether book contains ocr or not

4.1.2.9 getOcrPath()

```
std::string CBook::getOcrPath ( )
```

Returns

Path to directory of the book
Path to the ocr.txt file

4.1.2.10 getPath()

```
std::string CBook::getPath ( )
```

getter function to return the path to the directory of a book

Returns

string (Path to directory of the book)
Path to directory of the book

4.1.2.11 setOcr()

```
void CBook::setOcr (
            bool bOcr )
```

Parameters

in	*bool*	indicating whether book has ocr or not

4.1.2.12 setPath()

```
void CBook::setPath (
            std::string sPath )
```

Parameters

in	*sPath*	Path to direcory

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

- src/books/CBook.hpp
- src/books/Book.cpp

4.2 CBookManager Class Reference

Public Member Functions

- std::map< std::string, CBook > & getMapOfBooks ()
- bool initialize ()

  load all books.

4.2.1 Member Function Documentation

4.2.1.1 getMapOfBooks()

```
std::map< std::string, CBook > & CBookManager::getMapOfBooks ( )
```

Returns

map of all book

4.2.1.2 initialize()

```
bool CBookManager::initialize ( )
```

load all books.

Returns

boolean for successful of not

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

- src/books/CBookManager.hpp
- src/books/Bookmanager.cpp

4.3 CFunctions Class Reference

Public Member Functions

- bool compare (const char ∗chT1, const char ∗chT2)
- std::string removeSpace (std::string str)
- void ignoreCase (std::string &str)
- bool iequals (const char ∗a, const char ∗b)
- void createMapOfWords (std::string sPathToOcr, std::map< std::string, int > &mapWords)
- void createMapofWordsFromString (std::string sWords, std::map< std::string, int > &mapWords)
- void loadMapOfWords (std::string sPathToWords, std::map< std::string, int > &mapWords)

4.3.1 Member Function Documentation

4.3.1.1 compare()

```
bool CFunctions::compare (
            const char * chT1,
            const char * chT2 )
```

Parameters

in	*chT1*	first string to compare
in	*chT2*	second string to compare

Returns

Boolean indicating, whether strings compare or not

4.3.1.2 createMapOfWords()

```
void CFunctions::createMapOfWords (
            std::string sPathToOcr,
            std::map< std::string, int > & mapWords )
```

Parameters

in	*sPathToOcr*	Path to ocr of a book
out	*mapWords*	map to which new words will be added

4.3.1.3 createMapofWordsFromString()

```
void CFunctions::createMapofWordsFromString (
            std::string sWords,
            std::map< std::string, int > & mapWords )
```

Parameters

in	*sWords*	string of which map shall be created
out	*mapWords*	map to which new words will be added

4.3.1.4 iequals()

```
bool CFunctions::iequals (
            const char * chA,
            const char * chB )
```

iequals: compare two string and ignore case.

Parameters

in	*string*	a
in	*string*	b

Returns

true if strings are equal, false if not

4.3.1.5 ignoreCase()

```
void CFunctions::ignoreCase (
            std::string & str )
```

Parameters

out	*modief*	to ignore case

4.3.1.6 loadMapOfWords()

```
void CFunctions::loadMapOfWords (
            std::string sPathToWords,
            std::map< std::string, int > & mapWords )
```

Parameters

in	*sPathToWords*	path to .txt with all words in book
out	*mapWords*	map to which new words will be added.

4.3.1.7 removeSpace()

```
std::string CFunctions::removeSpace (
            std::string str )
```

Parameters

out	*str*	remove of spaces from str

Returns

modified string

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

- src/books/CFunctions.hpp
- src/books/Functions.cpp

4.4 CMetadata Class Reference

Public Member Functions

- CMetadata (std::string sMetadata)
- nlohmann::json getJson ()
- std::string getMetadata (std::string sSearch)
- std::string getMetadata (std::string sSearch, std::string sFrom)
- std::string getMetadata (std::string sSearch, std::string sFrom1, std::string sFrom2)
- std::string getMetadata (std::string sSearch, std::string sFrom1, std::string sFrom2, int in)
- std::vector< std::string > getCollections ()
- std::string getAuthor ()
- int getDate ()
- std::string getShow ()

4.4.1 Constructor & Destructor Documentation

4.4.1.1 CMetadata()

```
CMetadata::CMetadata (
            std::string sMetadata )
```

Parameters

in	*sMetadata*	path to metadata

4.4.2 Member Function Documentation

4.4.2.1 getAuthor()

```
std::string CMetadata::getAuthor ( )
```

Returns

lastName, or Name of author

4.4.2.2 getCollections()

```
std::vector< std::string > CMetadata::getCollections ( )
```

Returns

vector with all collections this book is in

4.4.2.3 getDate()

```
int CMetadata::getDate ( )
```

Returns

date or -1 if date does not exists or is currupted

4.4.2.4 getJson()

```
nlohmann::json CMetadata::getJson ( )
```

Returns

metadata

4.4.2.5 getMetadata() [1/4]

```
std::string CMetadata::getMetadata (
            std::string sSearch )
```

getter function to return selected metadata string (sSearch: which metadata (f.e. title, date...)

Returns

string

4.4.2.6 getMetadata() [2/4]

```
std::string CMetadata::getMetadata (
            std::string sSearch,
            std::string sFrom )
```

getter function to return selected metadata string (sSearch: which metadata (f.e. title, date...) string (sFrom: from which json (f.e. title -> data -> title)

Returns

string

4.4.2.7 getMetadata() [3/4]

```
std::string CMetadata::getMetadata (
            std::string sSearch,
            std::string sFrom1,
            std::string sFrom2 )
```

getter function to return selected metadata string (sSearch: which metadata (f.e. title, date...) string (sFrom2: from which json (f.e. title -> data -> title) string (sFrom2: in json from which json (f.e. author -> data creators -> author)

Returns

string

4.4.2.8 getMetadata() [4/4]

```
std::string CMetadata::getMetadata (
            std::string sSearch,
            std::string sFrom1,
            std::string sFrom2,
            int in )
```

getter function to return selected metadata string (sSearch: which metadata (f.e. title, date...) string (sFrom2: from which json (f.e. title -> data -> title) string (sFrom2: in json from which json (f.e. author -> data creators -> author) int (index: in case of list: which element from list)

Returns

string

4.4.2.9 getShow()

```
std::string CMetadata::getShow ( )
```

Returns

string with Auhtor + first 6 words 15 words of title + date

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

- src/books/CMetadata.hpp
- src/books/Metadata.cpp

4.5 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 (const std::string key)
- const std::string_view GetQueryParams (const std::string key)
- const std::string_view & GetMethod () const
- const std::string_view & GetURL ()
- const std::string_view & GetPath ()
- const std::string_view & GetQuery ()
- bool IsHealthy ()
- const void * GetBody ()
- unsigned long GetBodySize ()

4.5.1 Detailed Description

Parses and stores informations about an HTTP request.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 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.

4.5.3 Member Function Documentation

4.5.3.1 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

4.5.3.2 GetBodySize()

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

Returns the body size stored inside the class

Returns

The body size of the class.

4.5.3.3 GetHeaders()

```
const std::string_view & http_request::GetHeaders (
            const 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

4.5.3.4 GetMethod()

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

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

Returns

The method used for the message

4.5.3.5 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.

4.5.3.6 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

4.5.3.7 GetQueryParams()

```
const std::string_view http_request::GetQueryParams (
            const 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.

4.5.3.8 GetURL()

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

Returns the URL parsed from the request

Returns

The full url

4.5.3.9 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

4.5.3.10 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

4.6 http_response Class Reference

This class constructs a http reponse and sends it so the client.

`#include <httpparser.hpp>`

Classes

- struct MimeTypes
- struct StatusCodes

Public Types

- enum Errors { Errors::StatusAlreadyDefined, Errors::StatusNotSet }

Public Member Functions

- http_response (ssl_socket &sock)
- http_response & status (const std::string hdr)
- http_response & header (const std::string key, const std::string value)
- http_response & body (const std::string &bdy)

  Setting the body of the returned http message.
- void SendWithEOM ()

  Send the composed message back to the server and deletes itself afterwards.

4.6.1 Detailed Description

This class constructs a http reponse and sends it so the client.

4.6.2 Member Enumeration Documentation

4.6.2.1 Errors

`enum http_response::Errors [strong]`

The Error codes thrown by various http_response functions

Enumerator

StatusAlreadyDefined	The status was already defined when the user tried to set it.
StatusNotSet	The status was not set before the user tried to set the headers.

4.6.3 Constructor & Destructor Documentation

4.6.3.1 http_response()

```
http_response::http_response (
            ssl_socket & sock )
```

Creates the http response.

4.6.4 Member Function Documentation

4.6.4.1 body()

```
http_response & http_response::body (
            const std::string & bdy )
```

Setting the body of the returned http message.

Parameters

bdy	The body to set in the response

Returns

Returns a reference to itself for function chaining

4.6.4.2 header()

```
http_response & http_response::header (
            const std::string key,
            const std::string value )
```

Sets a new header for the pending response.

Parameters

in	*key*	The header field to write
in	*value*	The value field which will get set after the header field

Returns

A reference to the class itself, this enables function chaining eg response.header("Content-Length","10").header("hdr2","someVal");

4.6.4.3 SendWithEOM()

```
void http_response::SendWithEOM ( )
```

Send the composed message back to the server and deletes itself afterwards.

Returns

Returns nothing as the instance will be deleted hereafter

Todo Improve the way the function Sends stuff which is kinda slow at the moment

4.6.4.4 status()

```
http_response & http_response::status (
            const std::string hdr )
```

Sets the status code for the response

Parameters

statusCode	The statusCode for the response

Returns

A reference to the class itself, used for function chaining

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

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

4.7 http_response::MimeTypes Struct Reference

```
#include <httpparser.hpp>
```

Static Public Attributes

- static constexpr char html [ ] = "text/html"

  The body has the html type;.
- static constexpr char javascript [ ] = "text/javascript"

  The body is a javascript document.
- static constexpr char json [ ] = "application/json"

  The body is a json text file.
- static constexpr char jpg [ ] = "image/jpeg"

  The body has the jpg format.
- static constexpr char txt [ ] = "text/plain"

  The body has the txt format.

4.7.1 Detailed Description

The mime types used to tell the browser what kind of content to expect

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

- src/server/httpparser.hpp

4.8 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)
- std::map< std::string_view, std::function< void(http_request &, http_response &)> > & operator[ ] (std::string method)

  The function will be used to set the various callback functions in the server.

4.8.1 Detailed Description

The Basic Multithreaded HTTPS Server handles all requests.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 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.

Parameters

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

4.8.3 Member Function Documentation

4.8.3.1 operator[]()

```
std::map< std::string_view, std::function< void(http_request &, http_response &)> > & server↩
::operator[] (
            std::string method )
```

The function will be used to set the various callback functions in the server.

Parameters

method	Contains the method which should be used for the callback

Returns

A reference to the method map where one can set the callback directly eg. server["GET"]["/search"] = callback;

4.8.3.2 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

4.9 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)

Static Public Attributes

- static std::map< std::string_view, std::function< void(http_request &, http_response &)> > _getMap
  The map for getter callbacks.
- static std::map< std::string_view, std::function< void(http_request &, http_response &)> > _postMap
  The map for all post callbacks.

4.9.1 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.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 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

4.9.3 Member Function Documentation

4.9.3.1 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.

4.9.3.2 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

4.9.3.3 socket()

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

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

Returns

socket implementation

4.9.3.4 start()

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

Asynchronous starting the handshake.

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

- src/server/server.cpp

4.10 http_response::StatusCodes Struct Reference

```
#include <httpparser.hpp>
```

Static Public Attributes

- static constexpr char Continue [ ] = "HTTP/1.1 100 Continue\r\n"
  Continue HTTP Header.
- static constexpr char Ok [ ] = "HTTP/1.1 200 OK\r\n"
  Ok HTTP Header.
- static constexpr char NotFound [ ] = "HTTP/1.1 404 Not found\r\n"
  Not found HTTP header.
- static constexpr char Unauthorized [ ] = "HTTP/1.1 401 Unauthorized\r\n"
  Unauthorized http header.

4.10.1 Detailed Description

The status codes used to answer to the client

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

- src/server/httpparser.hpp

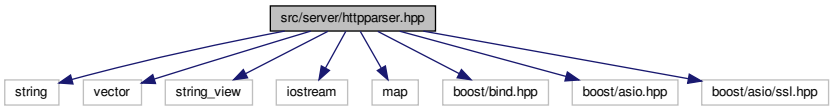
Chapter 5

File Documentation

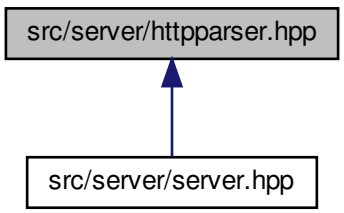
5.1 src/server/httpparser.hpp File Reference

```
#include <string>
#include <vector>
#include <string_view>
#include <iostream>
#include <map>
#include <boost/bind.hpp>
#include <boost/asio.hpp>
#include <boost/asio/ssl.hpp>
```

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.
- class http_response

  This class constructs a http reponse and sends it so the client.
- struct http_response::StatusCodes
- struct http_response::MimeTypes

Typedefs

- typedef boost::asio::ssl::stream< boost::asio::ip::tcp::socket > **ssl_socket**

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

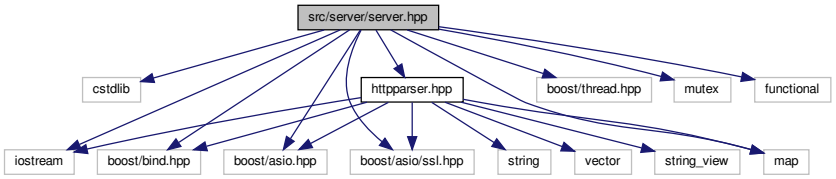
Returns

The calculated byte

5.2 src/server/server.hpp File Reference

```
#include <cstdlib>
#include <iostream>
#include <boost/bind.hpp>
#include <boost/asio.hpp>
#include <boost/asio/ssl.hpp>
#include <boost/thread.hpp>
#include <mutex>
#include <map>
#include <functional>
#include "httpparser.hpp"
```

Include dependency graph for server.hpp:



Classes

- class server

  The Basic Multithreaded HTTPS Server handles all requests.

5.2.1 Detailed Description

Classes for receiving encrypted ssl connections and for sending and receving data from these connections running.

This header defines the basic interface for starting a multithreaded ssl encrypted server which reads https requests