Search Engine

Generated by Doxygen 1.9.1

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Namespace Documentation

3.1 custom Namespace Reference

Functions

- size_t wordsCounter (const std::string &s)
- std::string getFileName (std::string s)
- size_t countOccurrences (const std::string &text, const std::string &word)
- std::vector< std::string > getUniqueWords (const std::string &text)
- double round (double num, int precision)
- void print_red (const std::string &msg)
- void print_green (const std::string &msg)
- void print_yellow (const std::string &msg)
- void print_blue (const std::string &msg)

3.1.1 Detailed Description

The namespace defines helper functions for string processing and text output

3.1.2 Function Documentation

3.1.2.1 countOccurrences()

```
size_t custom::countOccurrences (
            const std::string & text,
            const std::string & word )
```

Count occurrences of the word in text

Parameters

text	text
word	a word to count

Returns

number of occurrences word in text

3.1.2.2 getFileName()

```
std::string custom::getFileName (
            std::string s )
```

Receives a path to a file or directory as input, returns its name example:s ="/project/logs/config.txt", getFileName(s) returns config.txt

Parameters

s	path to file or directory

Returns

the name of directory or file

3.1.2.3 getUniqueWords()

```
std::vector< std::string > custom::getUniqueWords (
            const std::string & text )
```

The function generates a list of unique words from the text

Parameters

text	string of one or more words

Returns

list of unique words

3.1.2.4 print_blue()

```
void custom::print_blue (
            const std::string & msg )
```

Colors printing to standard output in blue

Parameters

msg	message to print

3.1.2.5 print_green()

```
void custom::print_green (
            const std::string & msg )
```

Colors printing to standard output in green

Parameters

msg	message to print

3.1.2.6 print_red()

```
void custom::print_red (
            const std::string & msg )
```

olors printing to standard output in red

Parameters

msg	message to print

3.1.2.7 print_yellow()

```
void custom::print_yellow (
            const std::string & msg )
```

Colors printing to standard output in yellow

Parameters

msg	message to print

3.1.2.8 round()

```
double custom::round (
            double num,
            int precision )
```

Rounds the number up to n decimal place

Parameters

num	number to round
precision	quantity with decimal place

Returns

rounded number

3.1.2.9 wordsCounter()

```
size_t custom::wordsCounter (
            const std::string & s )
```

Counts number of words in string

Parameters

s	string

Returns

number of words in string

3.2 format Namespace Reference

Namespaces

- unicode
- utf

3.2.1 Detailed Description

namespace defines two other namespaces for formatting strings in the unicode and utf-8 encodings accordingly

3.3 format::unicode Namespace Reference

Functions

- icu::UnicodeString makeUnicodeString (const std::string &s)
- void toLowerAll (icu::UnicodeString &s)
- icu::UnicodeString deleteExtraSpaces (const icu::UnicodeString &s)
- icu::UnicodeString deletePunctuationMarks (const icu::UnicodeString &s)
- std::string makeUtfString (const icu::UnicodeString &s)
- void convertToPlainText (icu::UnicodeString &s)

3.3.1 Detailed Description

defines functions for working with Unicode strings

3.3.2 Function Documentation

3.3.2.1 convertToPlainText()

```
void format::unicode::convertToPlainText (
            icu::UnicodeString & s )
```

Formatted UnicodeString: deletes whitespaces, punctuation marks, converts all letters to lowercase

Parameters

s	UnicodeString

3.3.2.2 deleteExtraSpaces()

```
icu::UnicodeString format::unicode::deleteExtraSpaces (
            const icu::UnicodeString & s )
```

Deletes whitespaces

Parameters

s	UnicodeString

Returns

UnicodeString

3.3.2.3 deletePunctuationMarks()

```
icu::UnicodeString format::unicode::deletePunctuationMarks (
            const icu::UnicodeString & s )
```

Deletes punctuation marks

Parameters

s	UnicodeStrings

Returns

formatted Unicode string

3.3.2.4 makeUnicodeString()

```
icu::UnicodeString format::unicode::makeUnicodeString (
            const std::string & s )
```

Convert std::string from utf-8 to unicode

Parameters

s	string

Returns

UnicodeString

3.3.2.5 makeUtfString()

```
std::string format::unicode::makeUtfString (
            const icu::UnicodeString & s )
```

Converts Unicode string to utf-8 std::string

Parameters

s	UnicodeString

Returns

std::string

3.3.2.6 toLowerAll()

```
void format::unicode::toLowerAll (
            icu::UnicodeString & s )
```

Converts all characters to lowercase

Parameters

s	UnicodeString

3.4 format::utf Namespace Reference

Functions

- void formatString (std::string &s)
- void toLowerCase (std::string &s)
- void deletePunctuationMarks (std::string &s)
- void deleteExtraSpaces (std::string &s)

3.4.1 Detailed Description

Defines functions for working with utf-8 strings

3.4.2 Function Documentation

3.4.2.1 deleteExtraSpaces()

```
void format::utf::deleteExtraSpaces (
            std::string & s )
```

Delete all extra spaces

Parameters

s	reference to string

3.4.2.2 deletePunctuationMarks()

```
void format::utf::deletePunctuationMarks (
            std::string & s )
```

Delete all punctuation marks in s

Parameters

s	reference to string

3.4.2.3 formatString()

```
void format::utf::formatString (
            std::string & s )
```

Delete punctuation marks, extra spaces and lowercase all letters

Parameters

s	string to format

Returns

reference to format string

3.4.2.4 toLowerCase()

```
void format::utf::toLowerCase (
            std::string & s )
```

Lowercase all letters in s

Parameters

s	reference to string

Chapter 4

Class Documentation

4.1 ConverterJSON Class Reference

`#include <converter_json.h>`

Public Member Functions

- **ConverterJSON** (const PathType &jsons_dir)
- **ConverterJSON** (PathType conf_p, PathType req_p)
- const json & getConfig () const
- RequestsList getRequests () const
- PathsList getTextDocuments () const
- int getResponsesLimit () const
- void putAnswers (const AnswersLists &answers) const
- void updateConfig (const PathType &path="")
- void updateRequests (const PathType &path="")
- PathType getConfigPath () const
- PathType getRequestsPath () const

Static Public Member Functions

- static json openJson (const PathType &path)
- static int writeJsonToFile (json &json_obj, const std::string &path)
- static PathType findFile (const std::string &file_name, const PathType &dir=".")

4.1.1 Detailed Description

Class for working with json files

4.1.2 Member Function Documentation

4.1.2.1 findFile()

```
PathType ConverterJSON::findFile (
            const std::string & file_name,
            const PathType & dir = "." )  [static]
```

Searches for file_name in directory tree with root in dir

Parameters

file_name	name of the file
dir	start dir

Returns

absolute path to a file or empty string

4.1.2.2 getConfig()

```
const json& ConverterJSON::getConfig ( ) const  [inline]
```

Config getter

Returns

json object config.json

4.1.2.3 getConfigPath()

```
PathType ConverterJSON::getConfigPath ( ) const  [inline]
```

config_path getter

Returns

config_path member

4.1.2.4 getRequests()

```
RequestsList ConverterJSON::getRequests ( ) const
```

Method for receiving requests from the requests.json file

Returns

a list of requests from the requests.json file

4.1.2.5 getRequestsPath()

```
PathType ConverterJSON::getRequestsPath ( ) const  [inline]
```

requests_path getter

Returns

requests_path member

4.1.2.6 getResponsesLimit()

```
int ConverterJSON::getResponsesLimit ( ) const
```

The method reads the max_responses field to determine the limit number of responses per request

Returns

max_responses

4.1.2.7 getTextDocuments()

```
PathsList ConverterJSON::getTextDocuments ( ) const
```

Returns

a list with the paths to documents to search in config.json

4.1.2.8 openJson()

```
json ConverterJSON::openJson (
            const PathType & path )  [static]
```

Creates json object from a file under path. Before creating check if file exists, perms to read and file's extension(must be ∗.json)

Parameters

path	path to the file

Returns

json object

4.1.2.9 putAnswers()

```
void ConverterJSON::putAnswers (
            const AnswersLists & answers ) const
```

Method writes answers to the file answers.json in json format

Parameters

answer	a data array containing answers to queries to the database of indexed documents

4.1.2.10 updateConfig()

```
void ConverterJSON::updateConfig (
            const PathType & path = "" )
```

Overwrites the current config file according to the path

Parameters

path	path to json file

4.1.2.11 updateRequests()

```
void ConverterJSON::updateRequests (
            const PathType & path = "" )
```

Overwrites the current requests file according to the path

Parameters

path	path to json file

4.1.2.12 writeJsonToFile()

```
int ConverterJSON::writeJsonToFile (
```

```
json & json_obj,
const std::string & path )  [static]
```

If file path exists overwriting it by file, create new file in path otherwise

Parameters

file	file to write
path	path to new file

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

- /home/maxnet/search_engine/include/converter_json.h
- /home/maxnet/search_engine/src/converter_json.cpp

4.2 DocRelevance Struct Reference

`#include <search_server.h>`

Public Member Functions

- **DocRelevance** (const std::pair< size_t, size_t > &pair)
- bool **operator**> (const DocRelevance &right) const

Public Attributes

- size_t **doc_id**
- size_t **relevance**

4.2.1 Detailed Description

Structure for forming the relevance of documents

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

- /home/maxnet/search_engine/include/search_server.h

4.3 FileReader Class Reference

`#include <file_reader.h>`

Public Member Functions

- **FileReader** (const FileReader &other)=delete
- FileReader & **operator=** (const FileReader &right)=delete
- **FileReader** (const std::string &file_path) noexcept
- void open (const std::string &file_path)
- bool is_open () const
- PathType getPath () const
- std::string getExtension () const
- Text getText ()
- Text getFormattedText ()

Static Public Member Functions

- static bool isReadable (const std::string &file_path)
- static bool isWriteable (const std::string &file_path)

4.3.1 Detailed Description

Wrapper over standard file reading stream

4.3.2 Member Function Documentation

4.3.2.1 getExtension()

```
std::string FileReader::getExtension ( ) const  [inline]
```

Extension getter

Returns

extension of opened file

4.3.2.2 getFormattedText()

```
Text FileReader::getFormattedText ( )
```

Returns file's text formatted as unicode: deletes whitespaces, punctuation marks, converts all letters to lowercase

Returns

utf-8 text

4.3.2.3 getPath()

```
PathType FileReader::getPath ( ) const  [inline]
```

Path getter

Returns

path to opened file

4.3.2.4 getText()

```
Text FileReader::getText ( )
```

Returns the contents of the file without formatting

Returns

text of the file

4.3.2.5 is_open()

```
bool FileReader::is_open ( ) const  [inline]
```

Checks if fstream opened

Returns

true if fstream opened, false otherwise

4.3.2.6 isReadable()

```
bool FileReader::isReadable (
            const std::string & file_path )  [static]
```

Checks permission to read from a file. Only owner rights are checked

Parameters

file_path	path to file

Returns

true if the file is readable by the owner false otherwise

4.3.2.7 isWriteable()

```
bool FileReader::isWriteable (
            const std::string & file_path )  [static]
```

Checks permission to write to a file. Only owner rights are checked

Parameters

file_path	path to file

Returns

true if the file is writeable by the owner false otherwise

4.3.2.8 open()

```
void FileReader::open (
            const std::string & file_path )
```

Opens new filestream, if there is already an open fstream, it will be closed

Parameters

file_path	path to file

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

- /home/maxnet/search_engine/include/file_reader.h
- /home/maxnet/search_engine/src/file_reader.cpp

4.4 InvertedIndex Class Reference

```
#include <inverted_index.h>
```

Public Member Functions

- InvertedIndex ()=default
- InvertedIndex (const InvertedIndex &other)
- InvertedIndex & **operator=** (const InvertedIndex &right)
- **InvertedIndex** (InvertedIndex &&other) noexcept
- InvertedIndex & **operator=** (InvertedIndex &&right) noexcept
- void updateDocumentBase (const PathsList &input_docs)
- const Frequency & getWordCount (const std::string &word) const

Static Public Attributes

- static const Frequency nfound

4.4.1 Detailed Description

The class is a dictionary containing unique words from indexed documents

4.4.2 Constructor & Destructor Documentation

4.4.2.1 InvertedIndex() [1/2]

```
InvertedIndex::InvertedIndex ( )  [default]
```

Constructs empty dictionary

4.4.2.2 InvertedIndex() [2/2]

```
InvertedIndex::InvertedIndex (
            const InvertedIndex & other )  [inline]
```

Copy constructor without copying of mutex member

Parameters

other	another instance of InvertedIndex class

4.4.3 Member Function Documentation

4.4.3.1 getWordCount()

```
const Frequency & InvertedIndex::getWordCount (
            const std::string & word ) const
```

Method determines the number of occurrences of a word in the loaded document base

Parameters

word	the word whose occurrence frequency is to be determined

Returns

const ref to a list with word frequency or nfound

4.4.3.2 updateDocumentBase()

```
void InvertedIndex::updateDocumentBase (
            const PathsList & input_docs )
```

Update or fill in the database of documents on which we will then search

Parameters

input_docs	paths to documents

4.4.4 Member Data Documentation

4.4.4.1 nfound

```
const Frequency InvertedIndex::nfound  [static]
```

Default return value for queries that are not in the dictionary

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

- /home/maxnet/search_engine/include/inverted_index.h
- /home/maxnet/search_engine/src/inverted_index.cpp

4.5 RelativeIndex Struct Reference

```
#include <search_server.h>
```

Public Member Functions

- bool **operator==** (const RelativeIndex &other) const

Public Attributes

- size_t **doc_id**
- double **rank**

4.5.1 Detailed Description

Structure represents an relevant document and its relevance for the query

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

- /home/maxnet/search_engine/include/search_server.h

4.6 ScreenWriter Class Reference

Public Member Functions

- **ScreenWriter** (ArgsList args)
- void **operator()** ()

Static Public Member Functions

- template<class... Args>
  static ConverterPtr makeConverter (Args &&... args)
- static ConverterPtr handMakeConverter ()
- static ArgsList commandParser (const std::string &cmd)

4.6.1 Member Function Documentation

4.6.1.1 commandParser()

```
ArgsList ScreenWriter::commandParser (
            const std::string & cmd )  [static]
```

Turns a string into a sequence of commands

Parameters

cmd	string containing commands

Returns

queue containing commands

4.6.1.2 handMakeConverter()

```
ConverterPtr ScreenWriter::handMakeConverter ( )  [static]
```

An alternative way to construct ConverterJSON through dialogue with the user

Returns

unique_ptr to created ConverterJSON

4.6.1.3 makeConverter()

```
template<class...  Args>
ConverterPtr ScreenWriter::makeConverter (
            Args &&...  args )  [static]
```

Method constructs ConverterJSON

Template Parameters

Args	variadic parameters

Parameters

args	arguments will be passed to the constructor ConverterJSON can be (), (string), (string, string)

Returns

unique_ptr to a ConverterJson

The method must be defined in the declaration file because it requires instantiation

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

- /home/maxnet/search_engine/include/screen_writer.h
- /home/maxnet/search_engine/src/screen_writer.cpp

4.7 SearchServer Class Reference

```
#include <search_server.h>
```

Public Member Functions

- **SearchServer** (const SearchServer &other)=default
- SearchServer (const InvertedIndex &idx)
- AnswersLists search (const std::vector< std::string > &queries_input)

4.7.1 Detailed Description

Class performs search for relevant documents

4.7.2 Constructor & Destructor Documentation

4.7.2.1 SearchServer()

```
SearchServer::SearchServer (
            const InvertedIndex & idx )  [inline], [explicit]
```

Parameters

idx	pointer to the docs database

4.7.3 Member Function Documentation

4.7.3.1 search()

```
AnswersLists SearchServer::search (
            const std::vector< std::string > & queries_input )
```

Search query processing method (async)

Parameters

queries_input	search queries taken from the file requests.json

Returns

returns a sorted list of relevant responses for given requests

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

- /home/maxnet/search_engine/include/search_server.h
- /home/maxnet/search_engine/src/search_server.cpp