# clas-digital

Generated by Doxygen 1.8.13

# Contents

# Chapter 1

# Hierarchical Index

## 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

# 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)
- const std::string & getPath ()

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

  *checks whether book already has map of words, if not it create them*
- void safeMapOfWords ()

  *safe created word list to file*

### 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()

```
const 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()

```
const 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 getTitle()

```
std::string CBook::getTitle ( )
```

**Returns**

title of book

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

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

  *load all books.*
- std::map< std::string, CBook * > * search (std::string sWord, bool ocr, bool title)

  *search function calling fitting function from search class*
- void createMapWords ()

  *create map of all words (key) and books in which the word occurs (value)*
- void createMapWordsTitle ()

  *create map of all words (key) and book-titles in which the word occurs (value)*

### 4.2.1 Member Function Documentation

#### 4.2.1.1 getMapOfBooks()

```
const 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

#### 4.2.1.3 search()

```
std::map< std::string, CBook * > * CBookManager::search (
            std::string sWord,
            bool ocr,
            bool title )
```

search function calling fitting function from search class

**Returns**

list of all found books

**Parameters**

| | | |
|---|---|---|
| in | *searchOPts* | |

**Returns**

list of all found books

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 convertToLower (std::string &str)
- bool iequals (const char ∗a, const char ∗b)
- bool isLetter (const char s)

  *function checks whether character is a letter with de and fr local*
- bool isWord (const char ∗chWord)

  *checks whether a string is a word*
- void split (std::string str, std::string sDelimitter, std::vector< std::string > &vStr)
- void transform (std::string &str)

  *cuts all non-letter-characters from end and beginning of str*
- 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 convertToLower()

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

**Parameters**

| | | |
|---|---|---|
| in,out | *str* | string to be modified |

#### 4.3.1.3 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 |

#### 4.3.1.4 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 |

#### 4.3.1.5 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

#### 4.3.1.6 isLetter()

```
bool CFunctions::isLetter (
            const char s )
```

function checks whether character is a letter with de and fr local

**Parameters**

| | | |
|---|---|---|
| in | *s* | char to be checked |

#### 4.3.1.7 isWord()

```
bool CFunctions::isWord (
            const char * chWord )
```

checks whether a string is a word

**Parameters**

| | | |
|---|---|---|
| in | *chWord* | string to be checked |

**Returns**

boolean for words/ no word

#### 4.3.1.8 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. |

#### 4.3.1.9 removeSpace()

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

**Parameters**

| | | |
|---|---|---|
| out | *str* | remove of spaces from str |

**Returns**

modified string

#### 4.3.1.10 split()

```
void CFunctions::split (
            std::string str,
            std::string delimiter,
            std::vector< std::string > & vStr )
```

**Parameters**

| | | |
|---|---|---|
| in | *str* | string to be splitet |
| in | *delimitter* | |

#### 4.3.1.11 transform()

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

cuts all non-letter-characters from end and beginning of str

**Parameters**

| | | |
|---|---|---|
| in,out | *string* | to modify |

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)
- const 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 ()
- std::string getTitle ()
- 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()

```
const 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

#### 4.4.2.10 getTitle()

```
std::string CMetadata::getTitle ( )
```

**Returns**

title of book

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

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

## 4.5 CSearch Class Reference

### Public Member Functions

- CSearch (std::string sWord)

  *constructor*
- void normalSearch (std::map< std::string, std::map< std::string, CBook *>> &mapWords, std::map< std::string, CBook *> *mapSR)

### 4.5.1 Member Function Documentation

#### 4.5.1.1 normalSearch()

```
void CSearch::normalSearch (
            std::map< std::string, std::map< std::string, CBook *>> & mapWords,
            std::map< std::string, CBook *> * mapSR )
```

**Parameters**

| in | *mapWords* | map of all words with a list of books in which this where accures |
|---|---|---|
| | | |

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

- src/books/CSearch.hpp
- src/books/Search.cpp

## 4.6 CSearchOptions Class Reference

### Public Member Functions

- CSearchOptions ()
- CSearchOptions (std::string chSearchedWord, int fuzzyness, std::vector< std::string > sCollections, bool onlyTitle, bool onlyOCR, std::string slastName, int from, int to)

  *Constructor.*
- void initialise (std::string chSearchedWord, int fuzzyness, std::vector< std::string > pillar, bool onlyTitle, bool onlyOCR, std::string slastName, int from, int to)

  *initialise search options outside of constructor*
- std::string getSearchedWord () const
- double getFuzzyness () const
- std::vector< std::string > getCollections () const
- bool getOnlyTitle () const
- bool getOnlyOcr () const
- std::string getLastName () const
- int getFrom () const
- int getTo () const

### 4.6.1 Constructor & Destructor Documentation

#### 4.6.1.1 CSearchOptions() [1/2]

```
CSearchOptions::CSearchOptions ( )
```

default constructor.

#### 4.6.1.2 CSearchOptions() [2/2]

```
CSearchOptions::CSearchOptions (
            std::string chSearchedWord,
            int fuzzyness,
            std::vector< std::string > sCollections,
            bool onlyTitle,
            bool onlyOCR,
            std::string slastName,
            int from,
            int to )
```

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *chSearchedWord* | searched word |
| in | *fuzzyness* | value of fuzzyness |
| in | *sCollections* | collections in which to be searched |
| in | *onlyTitle* | search only in title? |
| in | *onlyOCR* | search only in ocr (if exists) |
| in | *slastName* | las name of author |
| in | *from* | date from which books shall be searched |
| in | *to* | date to which books shall be searched |

### 4.6.2 Member Function Documentation

#### 4.6.2.1 getCollections()

```
std::vector< std::string > CSearchOptions::getCollections ( ) const
```

**Returns**

selected pillars

#### 4.6.2.2 getFrom()

```
int CSearchOptions::getFrom ( ) const
```

**Returns**

year from which books shall be searched in

#### 4.6.2.3 getFuzzyness()

```
double CSearchOptions::getFuzzyness ( ) const
```

**Returns**

selected fuzzyness

#### 4.6.2.4 getLastName()

```
std::string CSearchOptions::getLastName ( ) const
```

**Returns**

last name of selected author

#### 4.6.2.5 getOnlyOcr()

```
bool CSearchOptions::getOnlyOcr ( ) const
```

**Returns**

whether search only in ocr (if exists)

#### 4.6.2.6 getOnlyTitle()

```
bool CSearchOptions::getOnlyTitle ( ) const
```

**Returns**

whether search only in title

#### 4.6.2.7 getSearchedWord()

```
std::string CSearchOptions::getSearchedWord ( ) const
```

**Returns**

searched word

#### 4.6.2.8 getTo()

```
int CSearchOptions::getTo ( ) const
```

**Returns**

year to which books shall be searched

#### 4.6.2.9 initialise()

```
void CSearchOptions::initialise (
            std::string chSearchedWord,
            int fuzzyness,
            std::vector< std::string > pillar,
            bool onlyTitle,
            bool onlyOCR,
            std::string slastName,
            int from,
            int to )
```

initialise search options outside of constructor

**Parameters**

| | | |
|---|---|---|
| in | *chSearchedWord* | searched word |
| in | *fuzzyness* | value of fuzzyness |
| in | *sCollections* | collections in which to be searched |
| in | *onlyTitle* | search only in title? |
| in | *onlyOCR* | search only in ocr (if exists) |
| in | *slastName* | las name of author |
| in | *from* | date from which books shall be searched |
| in | *to* | date to which books shall be searched |

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

- src/books/CSearchOptions.hpp
- src/books/SearchOptions.cpp

## 4.7 debug::empty Struct Reference

This structure does not do anything with its constructor arguments it also does not print them.

`#include <debug.hpp>`

### Public Member Functions

- template<typename ... Args>
  empty (Args...)
  *This function does not do anything at all.*

### 4.7.1 Detailed Description

This structure does not do anything with its constructor arguments it also does not print them.

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

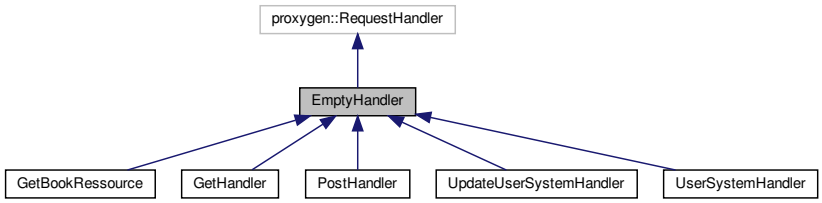
- src/util/debug.hpp

## 4.8 EmptyHandler Class Reference

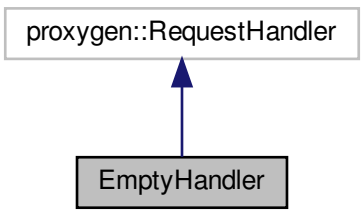
Small class used for setting the default empty method for every request handler.

`#include <BasicHandlers.hpp>`

Inheritance diagram for EmptyHandler:




Collaboration diagram for EmptyHandler:



### Public Member Functions

- virtual void onRequest (std::unique_ptr< proxygen::HTTPMessage > headers) noexcept override

  *Dummy function for the proxygen virtual function onRequest.*
- virtual void onBody (std::unique_ptr< folly::IOBuf > body) noexcept override

  *Dummy empty handler for the on body proxygen virtual function.*
- virtual void onUpgrade (proxygen::UpgradeProtocol proto) noexcept override

  *Dummy empty handler for the proxygen function onUpgrade.*
- virtual void requestComplete () noexcept override

  *The empty handler for the proxygen function requestComplete.*
- virtual void onError (proxygen::ProxygenError err) noexcept override

  *The dummy function for the proxygen function onError.*
- virtual void onEgressPaused () noexcept override

  *The dummy function for the proxygen function inEgressPaused.*
- virtual void onEgressResumed () noexcept override

  *The dummy function for the proxygen function onEgressResumed.*
- virtual void onEOM () noexcept override

  *The dummy function for the end of message function.*

### Public Attributes

- std::shared_ptr< User > _user

  *The user this specific request is from.*

## 4.8.1 Detailed Description

Small class used for setting the default empty method for every request handler.

This Handler is just used to provide an default empty method for the pure virtual class Request Handler, therefore it does not do anything at all expect implementing these empty methods Usage is as follows

```
class MyRequestHandler : public EmptyHandler
{
        public:
    void onRequest(std::unique_ptr<proxygen::HTTPMessage> headers)
        noexcept override;  //This is okay now just defining the function you use in your handler

};
```

## 4.8.2 Member Function Documentation

### 4.8.2.1 onBody()

```
virtual void EmptyHandler::onBody (
            std::unique_ptr< folly::IOBuf > body )  [inline], [override], [virtual], [noexcept]
```

Dummy empty handler for the on body proxygen virtual function.

**Parameters**

| | |
|---|---|
| *body* | The body provided by proxygen for this message, can be called multiple times for the same request if there is a lot of data |

Reimplemented in PostHandler, and UpdateUserSystemHandler.

#### 4.8.2.2 onError()

```
virtual void EmptyHandler::onError (
            proxygen::ProxygenError err )  [inline], [override], [virtual], [noexcept]
```

The dummy function for the proxygen function onError.

**Parameters**

| | |
|---|---|
| *err* | The error that occured |

#### 4.8.2.3 onRequest()

```
virtual void EmptyHandler::onRequest (
            std::unique_ptr< proxygen::HTTPMessage > headers )  [inline], [override], [virtual],
[noexcept]
```

Dummy function for the proxygen virtual function onRequest.

**Parameters**

| | |
|---|---|
| *headers* | The HTTP Message provided by proxygen |

Reimplemented in PostHandler, GetHandler, GetBookRessource, and UserSystemHandler.

#### 4.8.2.4 onUpgrade()

```
virtual void EmptyHandler::onUpgrade (
            proxygen::UpgradeProtocol proto )  [inline], [override], [virtual], [noexcept]
```

Dummy empty handler for the proxygen function onUpgrade.

**Parameters**

| | |
|---|---|
| *proto* | The new protocol to follow from there on |

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

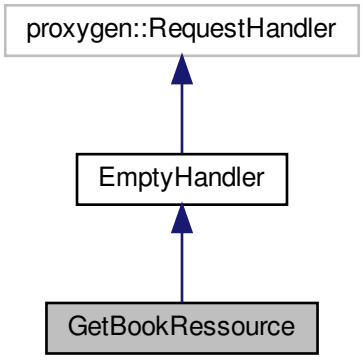
- src/server/BasicHandlers.hpp

## 4.9 GetBookRessource Class Reference

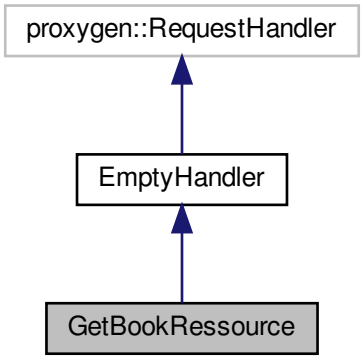
Returns either all the books in the server or all the files in one book or a specific ressource from a specific book.

`#include <URIObjects.hpp>`

Inheritance diagram for GetBookRessource:




Collaboration diagram for GetBookRessource:



### Public Member Functions

- void onRequest (std::unique_ptr< proxygen::HTTPMessage > headers) noexcept override

  *Sends back specific informations regarding the user profile and profile changes.*

### Additional Inherited Members

#### 4.9.1 Detailed Description

Returns either all the books in the server or all the files in one book or a specific ressource from a specific book.

#### 4.9.2 Member Function Documentation

##### 4.9.2.1 onRequest()

```
void GetBookRessource::onRequest (
            std::unique_ptr< proxygen::HTTPMessage > headers )  [override], [virtual], [noexcept]
```

Sends back specific informations regarding the user profile and profile changes.

**Parameters**

| | |
|---|---|
| *headers* | The headers provided by the proxygen library |

Reimplemented from EmptyHandler.

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

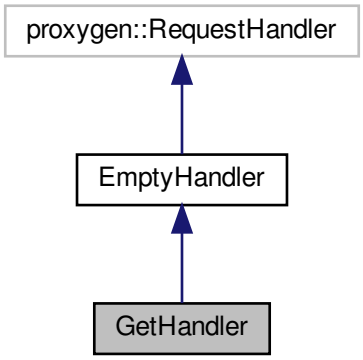
- src/server/URIObjects.hpp
- src/server/URIObjects.cpp

## 4.10 GetHandler Class Reference

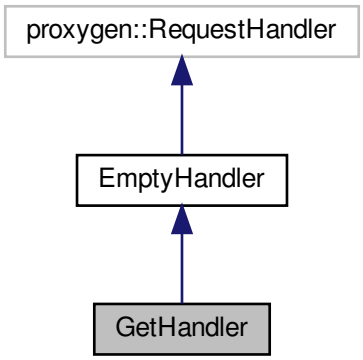
The Basic Get Handler which does almost all of the server disk IO acesses.

```
#include <BasicHandlers.hpp>
```

Inheritance diagram for GetHandler:




Collaboration diagram for GetHandler:




### Public Member Functions

- void onRequest (std::unique_ptr< proxygen::HTTPMessage > headers) noexcept override

  *Tries to satisfy a ressource request, do access right check and provide either an error response or the ressource response.*

### Additional Inherited Members

#### 4.10.1 Detailed Description

The Basic Get Handler which does almost all of the server disk IO acesses.

Most of the function

### 4.10.2 Member Function Documentation

#### 4.10.2.1 onRequest()

```
void GetHandler::onRequest (
            std::unique_ptr< proxygen::HTTPMessage > headers )  [override], [virtual], [noexcept]
```

Tries to satisfy a ressource request, do access right check and provide either an error response or the ressource response.

**Parameters**

| | |
|---|---|
| *headers* | The HTTP headers for this request provided by proxygen |

Reimplemented from EmptyHandler.

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

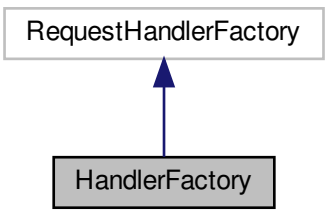
- src/server/BasicHandlers.hpp
- src/server/GetHandler.cpp

## 4.11 HandlerFactory Class Reference

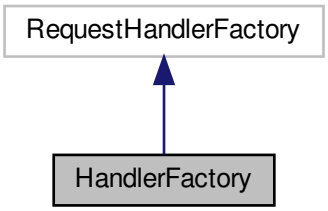
The HandlerFactory is used to instantiate the proxygen server and creates all request handler for every request directed to the server This class redirects every request to the right handler, in order to do so it keeps book of every URI object registered and sends the request to the URI object if there is any else the request goes to the default handler.

```
#include <HandlerFactory.hpp>
```

Inheritance diagram for HandlerFactory:



Collaboration diagram for HandlerFactory:




### Public Member Functions

- void onServerStart (folly::EventBase ∗) noexcept override
- void onServerStop () noexcept override

  *Handles the cleanup and behaviour if the server is stopped, in this case it does nothing at all because no cleanup is needed.*
- RequestHandler ∗ onRequest (RequestHandler ∗, HTTPMessage ∗hdr) noexcept override
- template<typename T >
  RequestHandler ∗ onRequest (std::map< std::string, EmptyHandler ∗(∗)()> &mp, HTTPMessage ∗hdr)

  *This function handles all requests to either getMap or postMap depending on the first parameter It creates the right RequestHandler and sets the user parameter in the handler class to the user that does the request.*

## 4.11.1 Detailed Description

The HandlerFactory is used to instantiate the proxygen server and creates all request handler for every request directed to the server This class redirects every request to the right handler, in order to do so it keeps book of every URI object registered and sends the request to the URI object if there is any else the request goes to the default handler.

## 4.11.2 Member Function Documentation

### 4.11.2.1 onRequest() [1/2]

```
RequestHandler* HandlerFactory::onRequest (
            RequestHandler * ,
            HTTPMessage * hdr )  [inline], [override], [noexcept]
```

As soon as an request is received this function gets called by the proxygen server and expects to return an Request handler. This function does nothing more than to select the correct request handler and return a new instance of him to the proxygen server.

**Parameters**

| | |
|---|---|
| *hdr* | The header of the message received, is used to set identify the user the message was sent from |

**Returns**

Returns the correct request handler for the requested URI

#### 4.11.2.2 onRequest() [2/2]

```
template<typename T >
RequestHandler* HandlerFactory::onRequest (
            std::map< std::string, EmptyHandler *(*)()> & mp,
            HTTPMessage * hdr )  [inline]
```

This function handles all requests to either getMap or postMap depending on the first parameter It creates the right RequestHandler and sets the user parameter in the handler class to the user that does the request.

**Parameters**

| | |
|---|---|
| *mp* | The map to use for looking up the URI function mapping |
| *hdr* | The http message received by the user |

**Returns**

The request handler which is about to handle the received request

#### 4.11.2.3 onServerStart()

```
void HandlerFactory::onServerStart (
            folly::EventBase *  )  [inline], [override], [noexcept]
```

Initialises the HandlerFactory and gets called by the proxygen server as the server starts up. Reserve all the post URI Objects as well as all the GET URI Objects.

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

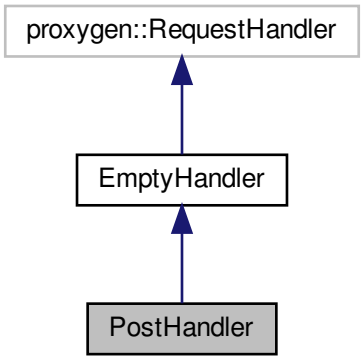
- src/server/HandlerFactory.hpp

## 4.12 PostHandler Class Reference

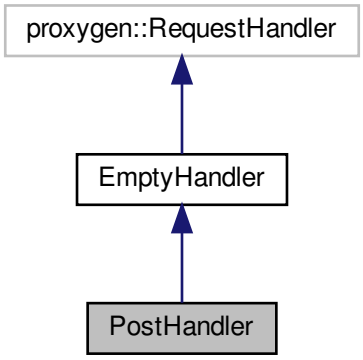
Handles the basic posts to the server mainly does the login and not much else.

`#include <BasicHandlers.hpp>`

Inheritance diagram for PostHandler:




Collaboration diagram for PostHandler:




### Public Member Functions

- void onBody (std::unique_ptr< folly::IOBuf > body) noexcept override

  *Proxygen callback for body data tries to parse user name and password from the data.*
- void onRequest (std::unique_ptr< proxygen::HTTPMessage > headers) noexcept override

  *The Function determines if the user wants to login or logout and handles the request accordingly.*

### Additional Inherited Members

### 4.12.1 Detailed Description

Handles the basic posts to the server mainly does the login and not much else.

### 4.12.2 Member Function Documentation

#### 4.12.2.1 onBody()

```
void PostHandler::onBody (
            std::unique_ptr< folly::IOBuf > body )  [override], [virtual], [noexcept]
```

Proxygen callback for body data tries to parse user name and password from the data.

**Parameters**

| | |
|---|---|
| *body* | The data send with the post request |

Reimplemented from EmptyHandler.

#### 4.12.2.2 onRequest()

```
void PostHandler::onRequest (
            std::unique_ptr< proxygen::HTTPMessage > headers )  [override], [virtual], [noexcept]
```

The Function determines if the user wants to login or logout and handles the request accordingly.

**Parameters**

| | |
|---|---|
| *headers* | The http message passed by proxygen to our handler |

Reimplemented from EmptyHandler.

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

- src/server/BasicHandlers.hpp
- src/server/PostHandler.cpp

## 4.13 debug::print Struct Reference

This structure prints everything in order given to the constructor of the class and an endline at the end of all prints.

```
#include <debug.hpp>
```

### Public Member Functions

- template<typename ... Args, typename T >
  print (T t1, Args... args)

  *Overloaded constructor prints all arguments in order to stdout.*
- template<typename T >
  print (T t1)

  *Prints the last argument of the constructor together with a newline.*

## 4.13.1 Detailed Description

This structure prints everything in order given to the constructor of the class and an endline at the end of all prints.

## 4.13.2 Constructor & Destructor Documentation

### 4.13.2.1 print() [1/2]

```
template<typename ...  Args, typename T >
debug::print::print (
            T t1,
            Args...  args )  [inline]
```

Overloaded constructor prints all arguments in order to stdout.

**Parameters**

| | |
|---|---|
| *t1* | The argument to print now |
| *args* | The arguments to print next |

### 4.13.2.2 print() [2/2]

```
template<typename T >
debug::print::print (
            T t1 )  [inline]
```

Prints the last argument of the constructor together with a newline.

**Parameters**

| | |
|---|---|
| *t1* | The last parameter to print |

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

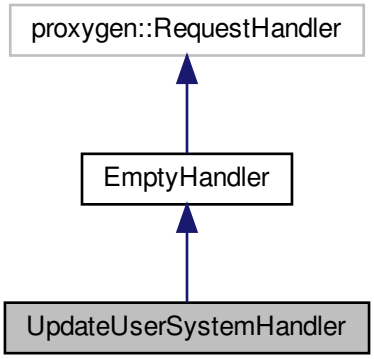
- src/util/debug.hpp

## 4.14 UpdateUserSystemHandler Class Reference

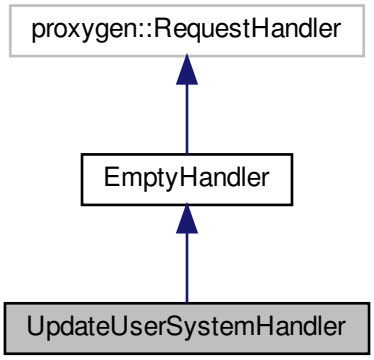
Handles all changes to the user table like create delete and change access.

`#include <URIObjects.hpp>`

Inheritance diagram for UpdateUserSystemHandler:




Collaboration diagram for UpdateUserSystemHandler:




### Public Member Functions

- void onBody (std::unique_ptr< folly::IOBuf > body) noexcept override

  *Only needs the body which contains the data to change the access rights and create the user. The supported actions are create delete and change user rights.*

### Additional Inherited Members

#### 4.14.1 Detailed Description

Handles all changes to the user table like create delete and change access.

#### 4.14.2 Member Function Documentation

##### 4.14.2.1 onBody()

```
void UpdateUserSystemHandler::onBody (
            std::unique_ptr< folly::IOBuf > body )  [override], [virtual], [noexcept]
```

Only needs the body which contains the data to change the access rights and create the user. The supported actions are create delete and change user rights.

**Parameters**

| | |
|---|---|
| *body* | The body which contains an array of json with the commands that should be executed |

**Returns**

200 Ok to the client if everything worked

Reimplemented from EmptyHandler.

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

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

## 4.15 URIFile Class Reference

The class contains basic information about the URI file.

```
#include <BasicHandlers.hpp>
```

### Public Member Functions

- URIFile (std::string path, int accessRights=0)

  *The constructor tells the URI Object has which file path on the disk and the access rights needed to access it.*
- URIFile (const URIFile &fl)

  *Copy constructur, careful never ever use that!!! It was just created because std::unordered_map needs a copy constructor.*
- URIFile (URIFile &&mvConst)

  *The move constructor constructs the new object by moving all the data out of the other URIFile object.*
- bool doAccessCheck (int acc) const

  *Performs an access check on this file with the given access rights.*
- const std::string & getPath () const

  *Returns a const reference to the path the file points to.*
- const std::string & getMimeType () const

  *Returns the mime type of the given file path in html reprensentation.*
- std::unique_ptr< folly::IOBuf > getBuffer ()

  *Returns a unqiue ptr to the clone of the IOBuf so that in can be send to the client in a timely manner.*
- std::unique_ptr< folly::IOBuf > & getBufferReference ()

  *This function is mainly used if one wants to move the content of the buffer to a file, eg. when the URI File object is short lived.*

## 4.15.1 Detailed Description

The class contains basic information about the URI file.

## 4.15.2 Constructor & Destructor Documentation

### 4.15.2.1 URIFile() [1/3]

```
URIFile::URIFile (
            std::string path,
            int accessRights = 0 )
```

The constructor tells the URI Object has which file path on the disk and the access rights needed to access it.

**Parameters**

| | |
|---|---|
| *path* | The path to the file to load |
| *accessRights* | The access rights needed to access the file, the accessRights must be a power of two! |

### 4.15.2.2 URIFile() [2/3]

```
URIFile::URIFile (
            const URIFile & fl )  [inline]
```

Copy constructur, careful never ever use that!!! It was just created because std::unordered_map needs a copy constructor.

**Parameters**

| | |
|---|---|
| *fl* | The file to create this file from |

#### 4.15.2.3 URIFile() [3/3]

```
URIFile::URIFile (
            URIFile && mvConst )  [inline]
```

The move constructor constructs the new object by moving all the data out of the other URIFile object.

**Parameters**

| | |
|---|---|
| *mvConst* | The object to move the data away from |

### 4.15.3 Member Function Documentation

#### 4.15.3.1 doAccessCheck()

```
bool URIFile::doAccessCheck (
            int acc ) const
```

Performs an access check on this file with the given access rights.

**Parameters**

| | |
|---|---|
| *acc* | The access rights trying to access the file, can be any positive integer |

#### 4.15.3.2 getBuffer()

```
std::unique_ptr< folly::IOBuf > URIFile::getBuffer ( )
```

Returns a unqiue ptr to the clone of the IOBuf so that in can be send to the client in a timely manner.

```
URIFile file("web/index.html",0);
ResponseBuilder(downstream_)
    .status(200,"Ok")
    .header("Content-Type",file.getMimeType())
    .body(file.getBuffer());    //Dont move the buffer as it would remove the data from the buffer inside
       the URIFile class
```

**Returns**

A unique ptr to a clone of the IOBuf which holds the file data

#### 4.15.3.3 getBufferReference()

```
std::unique_ptr< folly::IOBuf > & URIFile::getBufferReference ( )
```

This function is mainly used if one wants to move the content of the buffer to a file, eg. when the URI File object is short lived.

```
URIFile file("web/index.html",0);
ResponseBuilder(downstream_)
    .status(200,"Ok")
    .header("Content-Type",file.getMimeType())
    .body(std::move(file.getBufferReference()));
    return; //The URIFile object is short lived, no reason to copy the whole buffer so just move the loaded
       file buff away
```

**Returns**

The reference to a buffer

#### 4.15.3.4 getMimeType()

```
const std::string & URIFile::getMimeType ( ) const
```

Returns the mime type of the given file path in html reprensentation.

```
//This is how to use it
URIFile file("web/index.html",0);
file.getMimeType(); //Will be "text/html"
```

**Returns**

A const reference to the detected mime type

#### 4.15.3.5 getPath()

```
const std::string & URIFile::getPath ( ) const
```

Returns a const reference to the path the file points to.

**Returns**

A const reference to the path the file points to

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

- src/server/BasicHandlers.hpp
- src/server/GetHandler.cpp

## 4.16 User Class Reference

The basic user class this represents a basic user and stores email password and access rights for this user as well as the current session.

`#include <user_system.hpp>`

### Public Member Functions

- User ()
- User (const char *email, const char *pass, int access)
- std::string toJSON () const
- int GetAccessRights () const
- void SetAccessRights (int acc)
- const std::string & GetEmail () const
- const std::string & GetPassword () const
- const std::string & GetSessid () const

  *Returns the current session id of the user logged in at the moment.*
- void SetSessionId (std::string sessid)

  *Sets the session id to a new session id.*
- bool DoesMatch (std::string email, std::string passwd) const

### Static Public Member Functions

- static bool AccessCheck (const std::shared_ptr< User > &usr, int accRequired)

  *Check if the user has the necessary access rights to access this ressource.*

### 4.16.1 Detailed Description

The basic user class this represents a basic user and stores email password and access rights for this user as well as the current session.

### 4.16.2 Constructor & Destructor Documentation

#### 4.16.2.1 User() [1/2]

```
User::User ( )
```

Default constructor for the User class.

#### 4.16.2.2 User() [2/2]

```
User::User (
            const char * email,
            const char * pass,
            int access )
```

Constructs a user with a given email, password and access rights.

**Parameters**

| | |
|---|---|
| *email* | The email the user got |
| *pass* | The password the user will use to login |
| *access* | The access rights the user got. |

### 4.16.3 Member Function Documentation

#### 4.16.3.1 AccessCheck()

```
bool User::AccessCheck (
            const std::shared_ptr< User > & usr,
            int accRequired )  [static]
```

Check if the user has the necessary access rights to access this ressource.

**Parameters**

| | |
|---|---|
| *usr* | The user the message is one can be a nullptr as well! |
| *accRequired* | The access rights required to access this ressource |

**Returns**

true if the user has enough access rights false otherwise

#### 4.16.3.2 DoesMatch()

```
bool User::DoesMatch (
            std::string email,
            std::string passwd ) const
```

Checks if the given password and email matches the users credentials.

**Parameters**

| | |
|---|---|
| *email* | The email to check against |
| *passwd* | The Password to check against |

**Returns**

Returns true if the given credentials matches the user credentials

#### 4.16.3.3 GetAccessRights()

```
int User::GetAccessRights ( ) const
```

Getter for the access rights of the user.

#### 4.16.3.4 GetEmail()

```
const std::string & User::GetEmail ( ) const
```

The getter for the email the user uses.

#### 4.16.3.5 GetPassword()

```
const std::string & User::GetPassword ( ) const
```

The getter for the password of the user.

#### 4.16.3.6 GetSessid()

```
const std::string & User::GetSessid ( ) const
```

Returns the current session id of the user logged in at the moment.

**Returns**

The session id for the user

#### 4.16.3.7 SetAccessRights()

```
void User::SetAccessRights (
            int acc )
```

Setter for the access rights of the user.

**Parameters**

| | |
|---|---|
| *acc* | The new access rights for the user. |

#### 4.16.3.8 toJSON()

```
std::string User::toJSON ( ) const
```

Returns the user information as json file. User information means: email and access rights.

**Returns**

A string in the json format containing email and access rights of the user

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

- src/login/user_system.hpp
- src/login/user.cpp

## 4.17 UserHandler Class Reference

The user handler got a list of all available users and manages creating and deleting new users.

`#include <user_system.hpp>`

### Public Member Functions

- UserHandler (std::string filePath)
  *Loads the user table from the specified path and initialises it.*
- bool AddUser (std::string email, std::string password, int access)
  *Adds a user to the map of current users.*
- void SetAccessRights (std::string email, int newAccess)
- std::string toJSON ()
- void RemoveUser (std::string email)
- std::string DoLogin (std::string email, std::string password)
- std::shared_ptr< User > GetUserBySessid (std::string x)
  *Returns a shared ptr to the User associated with the session id if it exists.*
- std::shared_ptr< User > GetUserByName (std::string email)
  *Returns a shared pointer to the user associated with the given email returns a nullptr otherwise.*
- void RemoveSession (std::string x)
  *Removes the session by the given id.*

### Static Public Member Functions

- static UserHandler & GetUserTable ()
  *Returns the global user table used to manage all users in the server.*

### 4.17.1 Detailed Description

The user handler got a list of all available users and manages creating and deleting new users.

### 4.17.2 Constructor & Destructor Documentation

#### 4.17.2.1 UserHandler()

```
UserHandler::UserHandler (
            std::string filePath )
```

Loads the user table from the specified path and initialises it.

**Parameters**

| | |
|---|---|
| *filePath* | The path to the saved user table |

### 4.17.3 Member Function Documentation

#### 4.17.3.1 AddUser()

```
bool UserHandler::AddUser (
            std::string email,
            std::string password,
            int access )
```

Adds a user to the map of current users.

**Parameters**

| | |
|---|---|
| *email* | The email with which the user gets created |
| *password* | The password the user accounts has got |
| *access* | The access rights the user has |

#### 4.17.3.2 DoLogin()

```
std::string UserHandler::DoLogin (
            std::string email,
            std::string password )
```

Checks if a given set of email and password matches an existing user and returns the user if the password and login matches.

**Parameters**

| | |
|---|---|
| *email* | The email address of the user. |
| *password* | The Password of the user |

**Returns**

returns either the user if the password/email matches an user or zero if there is no user with this password and/or email.

#### 4.17.3.3 GetUserByName()

```
std::shared_ptr< User > UserHandler::GetUserByName (
            std::string email )
```

Returns a shared pointer to the user associated with the given email returns a nullptr otherwise.

**Parameters**

| | |
|---|---|
| *email* | The email of the user which should be found |

**Returns**

A shared pointer to the user associated with the given email

#### 4.17.3.4 GetUserBySessid()

```
std::shared_ptr< User > UserHandler::GetUserBySessid (
            std::string x )
```

Returns a shared ptr to the User associated with the session id if it exists.

**Parameters**

| | |
|---|---|
| *x* | The session id |

**Returns**

The user associated with the session id

#### 4.17.3.5 GetUserTable()

```
static UserHandler& UserHandler::GetUserTable ( )  [inline], [static]
```

Returns the global user table used to manage all users in the server.

**Returns**

A reference to the global user table

#### 4.17.3.6 RemoveSession()

```
void UserHandler::RemoveSession (
            std::string x )
```

Removes the session by the given id.

**Parameters**

| | |
|---|---|
| *x* | The session id to remove |

**Returns**

#### 4.17.3.7 RemoveUser()

```
void UserHandler::RemoveUser (
            std::string email )
```

Removes a user from the user table and deletes all files and all folder associated with him.

**Parameters**

| | |
|---|---|
| *email* | The email from the user who is about to be removed from the map of users |

#### 4.17.3.8 SetAccessRights()

```
void UserHandler::SetAccessRights (
            std::string email,
            int newAccess )
```

Set the access rights for a specific user and save the changes instantly to disk.

**Parameters**

| | |
|---|---|
| *email* | The email of the user who gets the access rights changed |
| *newAccess* | The new access rights the user gets granted |

#### 4.17.3.9 toJSON()

```
std::string UserHandler::toJSON ( )
```

Converts the complete user table to a string formatted in json style. The json contains only email name and access rights.

**Returns**

The string containing the UserTable in json format

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

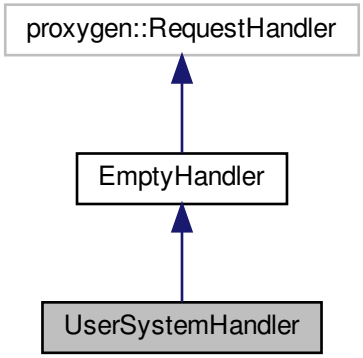
- src/login/user_system.hpp
- src/login/usertable.cpp

## 4.18 UserSystemHandler Class Reference

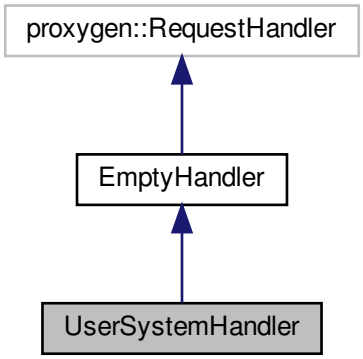
Handles all read accesses to the user system.

`#include <URIObjects.hpp>`

Inheritance diagram for UserSystemHandler:




Collaboration diagram for UserSystemHandler:



### Public Member Functions

- void onRequest (std::unique_ptr< proxygen::HTTPMessage > headers) noexcept override

  *Sends back specific informations regarding the user profile and profile changes.*

### Additional Inherited Members

## 4.18.1 Detailed Description

Handles all read accesses to the user system.

## 4.18.2 Member Function Documentation

#### 4.18.2.1 onRequest()

```
void UserSystemHandler::onRequest (
            std::unique_ptr< proxygen::HTTPMessage > headers )  [override], [virtual], [noexcept]
```

Sends back specific informations regarding the user profile and profile changes.

**Parameters**

| | |
|---|---|
| *headers* | The http request provided by the proxygen library |

Reimplemented from EmptyHandler.

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

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

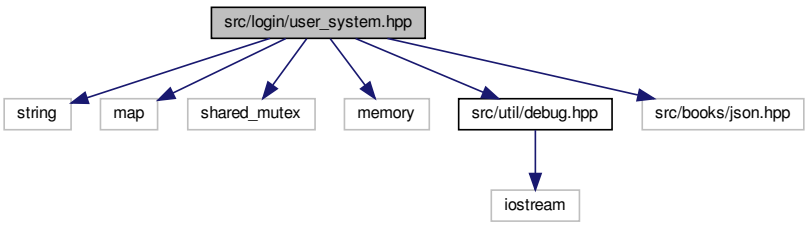
# Chapter 5

# File Documentation

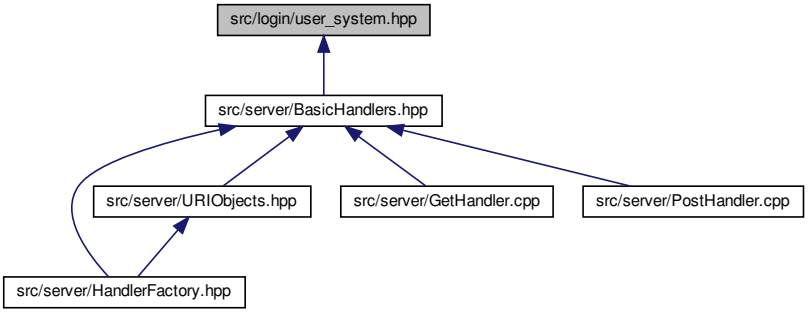
## 5.1 src/login/user_system.hpp File Reference

```
#include <string>
#include <map>
#include <shared_mutex>
#include <memory>
#include "src/util/debug.hpp"
#include "src/books/json.hpp"
```

Include dependency graph for user_system.hpp:




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



### Classes

- class User

  *The basic user class this represents a basic user and stores email password and access rights for this user as well as the current session.*
- class UserHandler

  *The user handler got a list of all available users and manages creating and deleting new users.*

### Enumerations

- enum AccessRights { USR_READ = 1, USR_WRITE = 2, USR_ADMIN = 4 }

  *Defines the basic a access rights a user can have at the moment.*

### 5.1.1 Detailed Description

This file defines the interface for the basic user class

### 5.1.2 Enumeration Type Documentation

#### 5.1.2.1 AccessRights

```
enum AccessRights
```

Defines the basic a access rights a user can have at the moment.

**Enumerator**

| | |
|---|---|
| USR_READ | The user has read access means, he can access all books for reading. |
| USR_WRITE | The user has write access he upload new books and change existing ones. |
| USR_ADMIN | The user is an admin he can create new users and give all users new rights, he can delete users as well. |

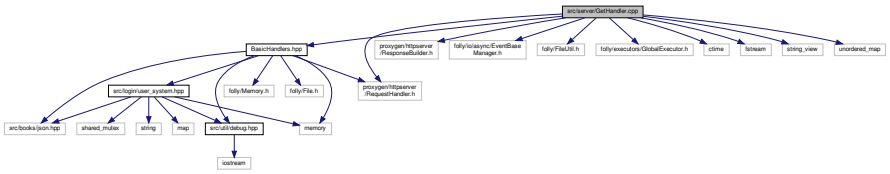
## 5.2 src/server/GetHandler.cpp File Reference

Implements the interface of the GetHandler class and the interface of the URIFile class.

```
#include "BasicHandlers.hpp"
#include <proxygen/httpserver/RequestHandler.h>
#include <proxygen/httpserver/ResponseBuilder.h>
#include <folly/io/async/EventBaseManager.h>
#include <folly/FileUtil.h>
#include <folly/executors/GlobalExecutor.h>
#include <ctime>
```

```
#include <fstream>
#include <string_view>
#include <unordered_map>
```

Include dependency graph for GetHandler.cpp:




## Functions

- void SendErrorNotFound (proxygen::ResponseHandler *rsp, std::string message)

  *Sends an 404 not found message to the client with the given message.*
- void SendAccessDenied (proxygen::ResponseHandler *rsp, std::string message)

  *Sends an 401 access denied message to the client with the given message.*

## Variables

- std::unordered_map< std::string, URIFile > **fileAccess**

### 5.2.1 Detailed Description

Implements the interface of the GetHandler class and the interface of the URIFile class.

Implements the interface of the gethandler class and the default response functions also hosts the file Map which mappes almost all files to a specific URI with the given access rights so one can do access checks on files

### 5.2.2 Function Documentation

#### 5.2.2.1 SendAccessDenied()

```
void SendAccessDenied (
            proxygen::ResponseHandler * rsp,
            std::string message = "<center><h1>Access denied</h1></center>" )
```

Sends an 401 access denied message to the client with the given message.

**Parameters**

| | |
|---|---|
| *rsp* | The downstream_ Response Builder ever RequestHandler has got |
| *message* | The message to set the body to, the format of the body will always be html<br><br>`SendAccessDenied(downstream_); //Can be used like this in every handler inheriting from proxygen::RequestHandler or EmptyHandler SendAccessDenied(downstream_, "<h1>My special error</h1>"); //Or specify a string to send a specific error message back` |

#### 5.2.2.2 SendErrorNotFound()

```
void SendErrorNotFound (
            proxygen::ResponseHandler * rsp,
            std::string message = "<center><h1>Not found!</h1></center>" )
```

Sends an 404 not found message to the client with the given message.

**Parameters**

| | |
|---|---|
| *rsp* | The downstream_ Response Builder ever RequestHandler has got |
| *message* | The message to set the body to, the format of the body will always be html<br><br>`SendErrorNotFound(downstream_); //Can be used like this in every handler inheriting from proxygen::RequestHandler or EmptyHandler SendErrorNotFound(downstream_, "<h1>My special error</h1>"); //Or specify a string to send a specific error message back` |

### 5.2.3 Variable Documentation

#### 5.2.3.1 fileAccess

```
std::unordered_map<std::string,URIFile> fileAccess
```

**Initial value:**

```
{
    {"/",URIFile("web/guest_index.html",0)},
    {"/favicon.ico",URIFile("web/favicon.png",0)},
    {"/home",URIFile("web/index.html",1)},
    {"/search",URIFile("web/Search.html",1)},
    {"/administration",URIFile("web/Administration.html",4)},
    {"/uploadbook",URIFile("web/UploadBook.html",2)},
    {"/managebooks",URIFile("web/ManageBooks.html",2)}
}
```

## 5.3 src/server/HandlerFactory.hpp File Reference

```
#include <folly/Memory.h>
#include <folly/executors/GlobalExecutor.h>
#include <folly/executors/CPUThreadPoolExecutor.h>
#include <folly/init/Init.h>
#include <folly/io/async/EventBaseManager.h>
#include <folly/portability/GFlags.h>
#include <folly/portability/Unistd.h>
#include <proxygen/httpserver/HTTPServer.h>
#include <proxygen/httpserver/RequestHandlerFactory.h>
#include <list>
```

```
#include <string>
#include <map>
#include "src/server/BasicHandlers.hpp"
#include "src/server/URIObjects.hpp"
```

Include dependency graph for HandlerFactory.hpp:

## Classes

- class HandlerFactory

  *The HandlerFactory is used to instantiate the proxygen server and creates all request handler for every request directed to the server This class redirects every request to the right handler, in order to do so it keeps book of every URI object registered and sends the request to the URI object if there is any else the request goes to the default handler.*

## Functions

- template<typename T >
  EmptyHandler * **CreateHandler** ()

### 5.3.1 Detailed Description

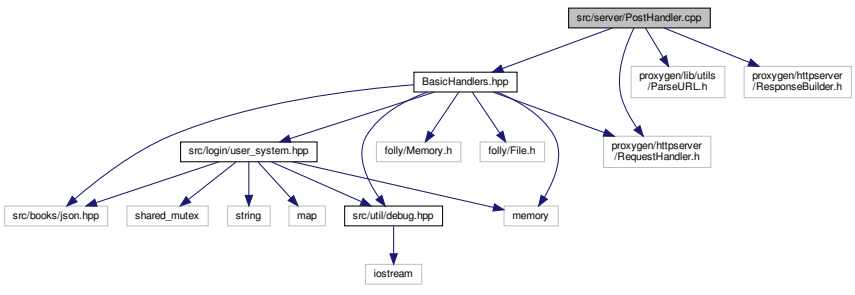
This file contains the basic static handler factory used to instantiate the proxygen server The Handler Factory selects based on the request the appropriate request handler class

## 5.4 src/server/PostHandler.cpp File Reference

Implements the interface for the PostHandler class and handles all user logins.

```
#include "BasicHandlers.hpp"
#include <proxygen/lib/utils/ParseURL.h>
#include <proxygen/httpserver/RequestHandler.h>
#include <proxygen/httpserver/ResponseBuilder.h>
```

Include dependency graph for PostHandler.cpp:



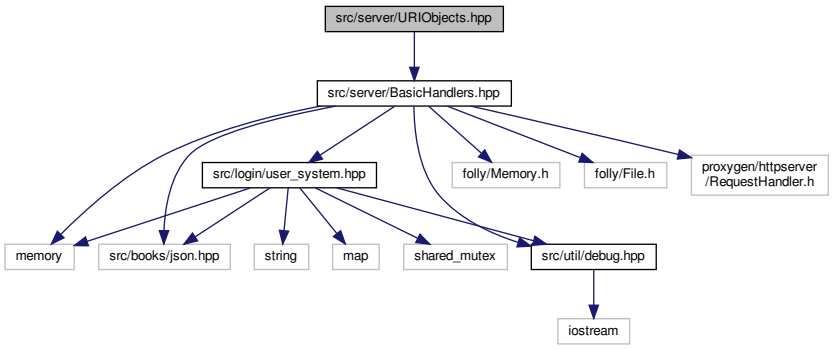
### 5.4.1 Detailed Description

Implements the interface for the PostHandler class and handles all user logins.

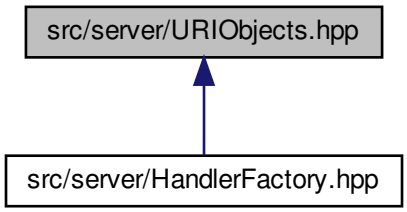
## 5.5 src/server/URIObjects.hpp File Reference

```
#include "src/server/BasicHandlers.hpp"
```

Include dependency graph for URIObjects.hpp:




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




### Classes

- class UserSystemHandler

  *Handles all read accesses to the user system.*
- class UpdateUserSystemHandler

  *Handles all changes to the user table like create delete and change access.*
- class GetBookRessource

  *Returns either all the books in the server or all the files in one book or a specific ressource from a specific book.*

### 5.5.1 Detailed Description

Defines the basic URI Objects /search and /getprofileinfo etc.