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)
- 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:

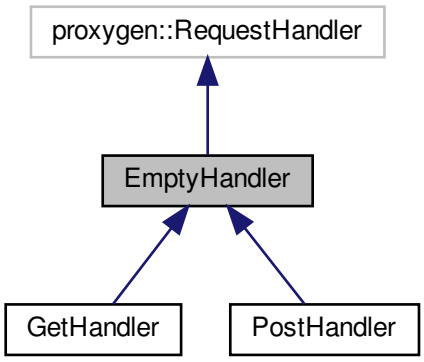
- src/books/CMetadata.hpp
- src/books/Metadata.cpp

4.5 EmptyHandler Class Reference

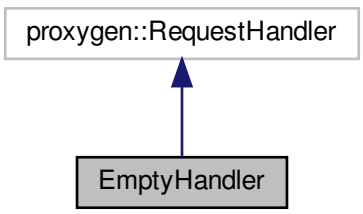
Small class used for setting the default empty method for every request handler.

```
#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.

4.5.1 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

};
```

4.5.2 Member Function Documentation

4.5.2.1 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.

4.5.2.2 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

4.5.2.3 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, and GetHandler.

4.5.2.4 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:

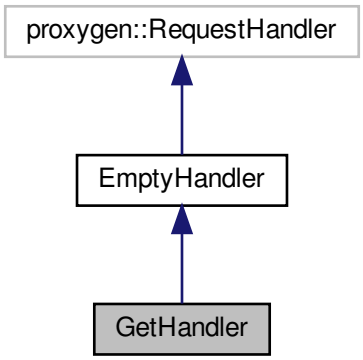
- src/server/BasicHandlers.hpp

4.6 GetHandler Class Reference

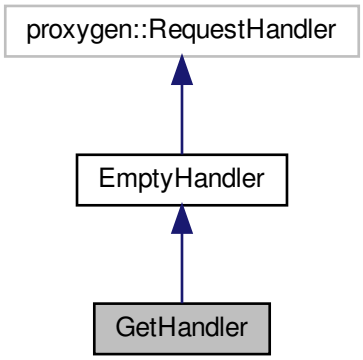
The Basic Get Handler which does almost all of the server disk IO acesses.

`#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.

4.6.1 Detailed Description

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

Most of the function

4.6.2 Member Function Documentation

4.6.2.1 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:

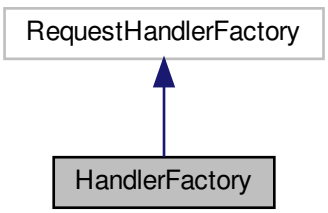
- src/server/BasicHandlers.hpp
- src/server/GetHandler.cpp

4.7 HandlerFactory Class Reference

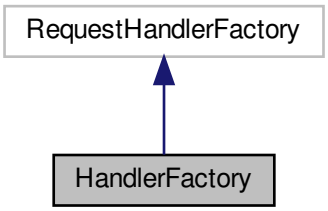
The HandlerFactory is used to instantiate the proxygen server and creates all request handler for every request directed to the server This class redirects every request to the right handler, in order to do so it keeps book of every URI object registered and sends the request to the URI object if there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

4.7.1 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.

4.7.2 Member Function Documentation

4.7.2.1 onRequest()

```
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

4.7.2.2 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:

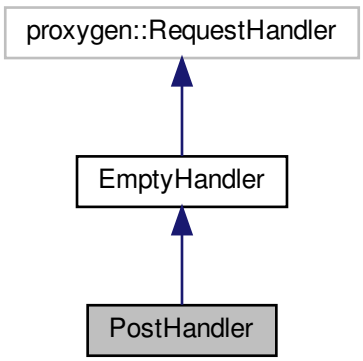
- src/server/HandlerFactory.hpp

4.8 PostHandler Class Reference

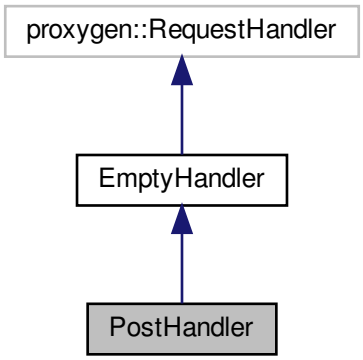
Handles the basic posts to the server mainly does the login and not much else.

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

Inheritance diagram for PostHandler:



Collaboration diagram for PostHandler:



Public Member Functions

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

  Handles all post requests which are not handled by other handlers.
- 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.

4.8.1 Detailed Description

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

4.8.2 Member Function Documentation

4.8.2.1 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.

4.8.2.2 onRequest()

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

Handles all post requests which are not handled by other handlers.

Parameters

headers	The headers provided by proxygen

Reimplemented from EmptyHandler.

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

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

4.9 URIFile Class Reference

The class contains basic information about the URI file.

Public Member Functions

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

  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 () const

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

4.9.1 Detailed Description

The class contains basic information about the URI file.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 URIFile() [1/3]

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

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!

4.9.2.2 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

4.9.2.3 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

4.9.3 Member Function Documentation

4.9.3.1 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

4.9.3.2 getBuffer()

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

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

Returns

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

4.9.3.3 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

4.9.3.4 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 file:

- src/server/GetHandler.cpp

Chapter 5

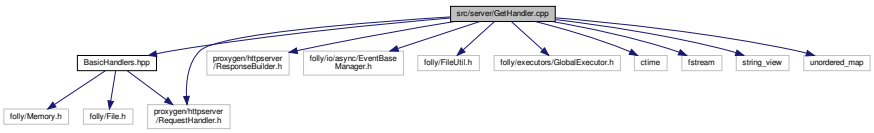
File Documentation

5.1 src/server/GetHandler.cpp File Reference

Implements the interface of the GetHandler class and all the default response functions like SendErrorNotFound etc.

```
#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:



Classes

- class URIFile

  The class contains basic information about the URI file.

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**

5.1.1 Detailed Description

Implements the interface of the GetHandler class and all the default response functions like SendErrorNotFound etc.

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

5.1.2 Function Documentation

5.1.2.1 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.

```
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
```

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

5.1.2.2 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.

```
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
```

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

5.1.3 Variable Documentation

5.1.3.1 fileAccess

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

Initial value:

```
{
    {"/",URIFile("web/index.html",0)},
    {"/favicon.ico",URIFile("web/favicon.png",0)}
}
```

5.2 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 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.

5.2.1 Detailed Description

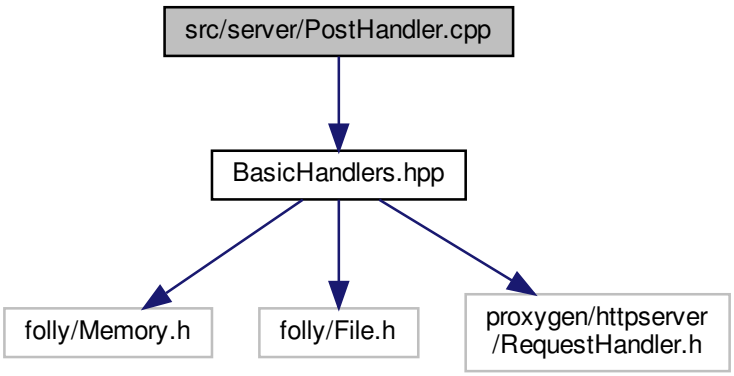
This file contains the basic static handler factory used to instantiate the proxygen server The Handler Factory selects based on the request the appropriate request handler class

5.3 src/server/PostHandler.cpp File Reference

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

```
#include "BasicHandlers.hpp"
```

Include dependency graph for PostHandler.cpp:



5.3.1 Detailed Description

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