jsonToBatProject

0.2.0

Generated on Wed Feb 28 2024 08:49:55 for jsonToBatProject by Doxygen 1.9.8

Wed Feb 28 2024 08:49:55

Chapter 1

Todo List

Member main (int argc, char ∗argv[])

Remove Debug once getopt is working

Member utils::StartupHandler::getOptions (int argc, char ∗argv[])

Implement functionality for the options.

- Implement/Add more options.
- Shorten function and outsource functionality to other functions.

Member utils::StartupHandler::initEasyLogging ()

Improve easylogging configuration

Chapter 2

Bug List

Member main (int argc, char ∗argv[])
Getopt is not working on Windows.

Member utils::StartupHandler::getOptions (int argc, char ∗argv[])
Global verbose flag is not working.

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 utils Namespace Reference

Namespace for utility functions.

Classes

- class StartupHandler

    Handles startup task for the application.

Variables

- static int verbose = 0

6.1.1 Detailed Description

Namespace for utility functions.

This namespace contains utility functions for the application. Currently, it contains the StartupHandler class.

6.1.2 Variable Documentation

6.1.2.1 verbose

```
int utils::verbose = 0  [static]
```

Definition at line 16 of file StartupHandler.cpp.

Chapter 7

Class Documentation

7.1 utils::StartupHandler Class Reference

Handles startup task for the application.

`#include <StartupHandler.hpp>`

Static Public Member Functions

- static void initEasyLogging ()

  Initialize easylogging.
- static std::optional< std::string > getOptions (int argc, char ∗argv[ ])

  Get options from command line.

Private Member Functions

- StartupHandler ()=default

  Constructor (private)
- StartupHandler (const StartupHandler &)=delete

  Copy constructor (deleted)
- StartupHandler & operator= (const StartupHandler &)=delete

  Assignment operator (deleted)

7.1.1 Detailed Description

Handles startup task for the application.

This class provides functionality for the startup of the application. Currently it initializes easylogging and parses given options.

Note

I think this class should stay static - Simon

Definition at line 28 of file StartupHandler.hpp.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 StartupHandler() [1/2]

```
utils::StartupHandler::StartupHandler ( )  [private], [default]
```

Constructor (private)

This class should not be instantiated.

7.1.2.2 StartupHandler() [2/2]

```
utils::StartupHandler::StartupHandler (
            const StartupHandler &  )  [private], [delete]
```

Copy constructor (deleted)

This class should not be instantiated.

7.1.3 Member Function Documentation

7.1.3.1 getOptions()

```
std::optional< std::string > utils::StartupHandler::getOptions (
            int argc,
            char * argv[] )  [static]
```

Get options from command line.

This function parses the command line options and returns the filename given as an argument. It can hadle short, long and "regular" arguments. Currently, the following options are supported:

- -h, –help: Show help
- -V, –version: Show version
- –verbose: Set verbose flag
- –brief: Unset verbose flag
- –test: Test

  Todo

  Bug Global verbose flag is not working.

Parameters

argc	Number of arguments
argv	Arguments

Returns

Returns either the filename or nothing.

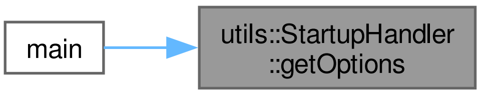
Exceptions

std::invalid_argument	If more than one filename is given.

Definition at line 26 of file StartupHandler.cpp.

References utils::verbose.

Here is the caller graph for this function:



7.1.3.2 initEasyLogging()

```
void utils::StartupHandler::initEasyLogging ( )  [static]
```

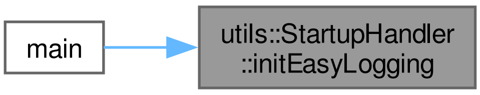
Initialize easylogging.

This function initializes easylogging with the configuration file "$SOURCE/conf/easylogging.conf".

Todo • Improve easylogging configuration

Definition at line 18 of file StartupHandler.cpp.

Here is the caller graph for this function:



7.1.3.3 operator=()

```
StartupHandler & utils::StartupHandler::operator= (
            const StartupHandler & )  [private], [delete]
```

Assignment operator (deleted)

This class should not be instantiated.

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

- src/headers/StartupHandler.hpp
- src/sources/StartupHandler.cpp

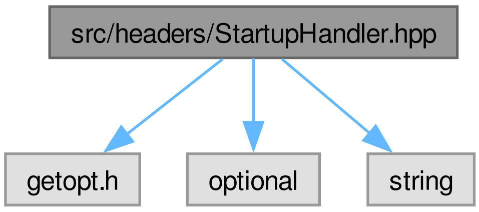
Chapter 8

File Documentation

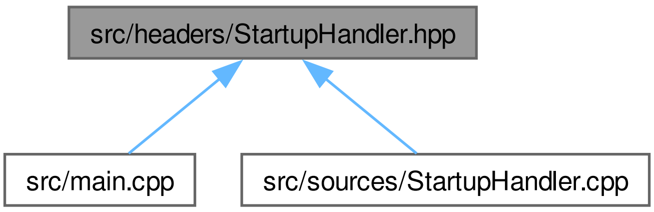
8.1 src/headers/StartupHandler.hpp File Reference

```
#include <getopt.h>
#include <optional>
#include <string>
```

Include dependency graph for StartupHandler.hpp:



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


src/headers/StartupHandler.hpp
src/main.cpp
src/sources/StartupHandler.cpp

Classes

- class utils::StartupHandler

  Handles startup task for the application.

Namespaces

- namespace utils

  Namespace for utility functions.

8.2 StartupHandler.hpp

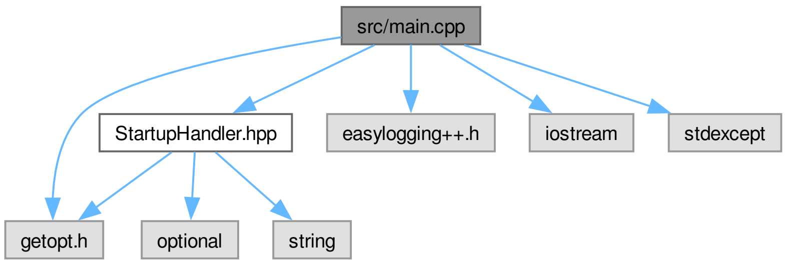
Go to the documentation of this file.

```
#ifdef IS_WINDOWS
#include <xgetopt.h>
#else
#include <getopt.h>
#endif

#include <optional>
#include <string>

namespace utils {
class StartupHandler {
  public:
    static void initEasyLogging();

    static std::optional<std::string> getOptions(int argc, char* argv[]);

  private:
    StartupHandler() = default;

    StartupHandler(const StartupHandler &) = delete;

    StartupHandler &operator=(const StartupHandler &) = delete;

};
} // namespace utils
```

8.3 src/main.cpp File Reference

```
#include "StartupHandler.hpp"
#include "easylogging++.h"
#include <getopt.h>
#include <iostream>
#include <stdexcept>
```

Include dependency graph for main.cpp:



Functions

- INITIALIZE_EASYLOGGINGPP int main (int argc, char ∗argv[ ])

  Main function.

8.3.1 Function Documentation

8.3.1.1 main()

```
INITIALIZE_EASYLOGGINGPP int main (
             int argc,
             char * argv[ ] )
```

Main function.

This is the main function for the application, The application is designed to parse a json file and create a batch file from it. Further more it provides a CLI to help the user to interact with the application.

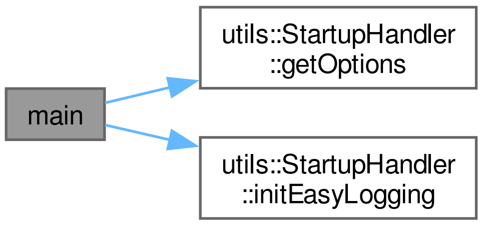
Todo • Remove Debug once getopt is working

Bug Getopt is not working on Windows.

Definition at line 34 of file main.cpp.

References utils::StartupHandler::getOptions(), and utils::StartupHandler::initEasyLogging().

Here is the call graph for this function:


main
utils::StartupHandler ::getOptions
utils::StartupHandler ::initEasyLogging

8.4 main.cpp

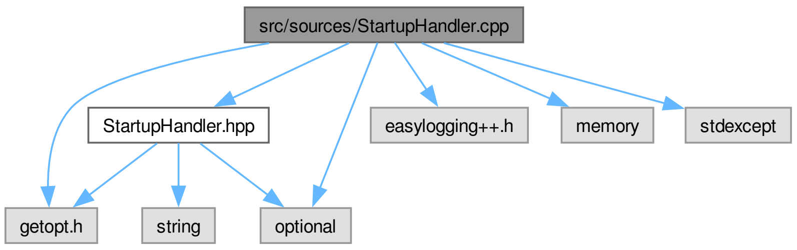
Go to the documentation of this file.

```
#include "StartupHandler.hpp"
#include "easylogging++.h"

#ifdef IS_WINDOWS
#include <xgetopt.h>
#else
#include <getopt.h>
#endif

#include <iostream>
#include <stdexcept>

INITIALIZE_EASYLOGGINGPP

int main(int argc, char *argv[]) {
  //Debug
#ifdef IS_LINUX
  std::cout « "Linux\n\n" « std::endl;
#elif IS_WINDOWS
  std::cout « "Windows\n\n" « std::endl;
#else
  std::cout « "Unknown OS\n\n" « std::endl;
#endif
  utils::StartupHandler::initEasyLogging();

  if (argc <= 1) {
    LOG(WARNING) « "No arguments provided, exiting!";
    std::cout « "No arguments provided, exiting!\n";
    return 1;
  }

  std::cout « "Hello, World!" « std::endl;
  std::optional<std::string> filename;

  try {
    filename = utils::StartupHandler::getOptions(argc, argv);
  } catch (const std::invalid_argument &e) {
    LOG(WARNING) « "Caught invalid argument: " « e.what();
    std::cout « "Invalid argument: " « e.what() « std::endl;
  }

  if (!filename.has_value()) {
    LOG(ERROR) « "No filename given! Exiting...";
    std::cerr « "No filename given!\nExiting...\n";
    return 1;
  }

  LOG(INFO) « "Filename received: " « filename.value();
  std::cout « "Filename: " « filename.value() « std::endl;
  LOG(INFO) « "Further processing...";
  std::cout « "Further processing..." « std::endl;
  LOG(INFO) « "Application exiting!";
  return 0;
}
```

8.5 src/sources/StartupHandler.cpp File Reference

```
#include "StartupHandler.hpp"
#include "easylogging++.h"
#include <getopt.h>
#include <memory>
#include <optional>
#include <stdexcept>
```

Include dependency graph for StartupHandler.cpp:



Namespaces

- namespace utils

  Namespace for utility functions.

Variables

- static int utils::verbose = 0

8.6 StartupHandler.cpp

Go to the documentation of this file.

```
#include "StartupHandler.hpp"
#include "easylogging++.h"

#ifdef IS_WINDOWS
#include <xgetopt.h>
#else
#include <getopt.h>
#endif

#include <memory>
#include <optional>
#include <stdexcept>

namespace utils {

static int verbose = 0;

void StartupHandler::initEasyLogging()
{
    el::Configurations conf("conf/easylogging.conf");
    el::Loggers::reconfigureLogger("default", conf);
    el::Loggers::reconfigureAllLoggers(conf);
    LOG(INFO) « "Easylogging initialized!";
}

std::optional<std::string> StartupHandler::getOptions(int argc, char* argv[])
{
    LOG(INFO) « "Parsing options...";
    static const struct option long_options[] = {
        /* These options set a flag. */
        {"verbose", no_argument, &verbose, 1},
        {"brief", no_argument, &verbose, 0},
        {"help", no_argument, nullptr, 'h'},
        {"version", no_argument, nullptr, 'V'},
        {"test", required_argument, nullptr, 0},
        nullptr
    };

    do {
        int optIndex = -1;
        std::unique_ptr<struct option> opt = nullptr;
        auto result = getopt_long(argc, argv, "hV", long_options, &optIndex);

        if (result == -1) {
            break;
        }

        switch (result) {
            case '?':
                LOG(INFO) « "Unknown option given";
                std::cout « "Not know\n";
                break;

            case 'h':
                LOG(INFO) « "Help option given";
                std::cout « "long h\n";
                break;

            case 'V':
                LOG(INFO) « "Version option given";
                std::cout « "long V\n";

            case '0':
                opt = std::make_unique<struct option>(long_options[optIndex]);
                LOG(INFO) « "Option " « opt->name « " given";

                if (opt->has_arg == required_argument) {
                    LOG(INFO) « "Argument: " « optarg;
                }

                break;

            default:
                std::cout « "I shouldnt have been here!\n";
                break;
        }
    } while (true);

    LOG(INFO) « "Parsing options done";
    std::optional<std::string> filename = {};
    LOG(INFO) « "Parsing other arguments...";

    while (optind < argc) {
        if (filename.has_value()) {
            LOG(ERROR) « "Only one filename can be given!";
            throw std::invalid_argument("Only one filename can be given!\n");
        }

        LOG(INFO) « "Filename set to: " « argv[optind];
        filename = std::string(argv[optind++]);
    }

    return filename;
}
} // namespace utils
```

Index