

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

1 Todo List	1
2 Bug List	3
3 Namespace Index	5
3.1 Namespace List	5
4 Class Index	7
4.1 Class List	7
5 File Index	9
5.1 File List	9
6 Namespace Documentation	11
6.1 utils Namespace Reference	11
6.1.1 Detailed Description	11
6.1.2 Variable Documentation	11
6.1.2.1 verbose	11
7 Class Documentation	13
7.1 utils::StartupHandler Class Reference	13
7.1.1 Detailed Description	13
7.1.2 Constructor & Destructor Documentation	14
7.1.2.1 StartupHandler() [1/2]	14
7.1.2.2 StartupHandler() [2/2]	14
7.1.3 Member Function Documentation	14
7.1.3.1 getOptions()	14
7.1.3.2 initEasyLogging()	15
7.1.3.3 operator=()	16
8 File Documentation	17
8.1 src/headers/StartupHandler.hpp File Reference	17
8.2 StartupHandler.hpp	18
8.3 src/main.cpp File Reference	18
8.3.1 Function Documentation	19
8.3.1.1 main()	19
8.4 main.cpp	20
8.5 src/sources/StartupHandler.cpp File Reference	20
8.6 StartupHandler.cpp	21
Index	23

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:

utils	Namespace for utility functions	11
-----------------------	---	--------------------

Chapter 4

Class Index

4.1 Class List

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

utils::StartupHandler	Handles startup task for the application	13
---------------------------------------	--	----

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

src/ main.cpp	18
src/headers/ StartupHandler.hpp	17
src/sources/ StartupHandler.cpp	20

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 handle 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

<i>argc</i>	Number of arguments
<i>argv</i>	Arguments

Returns

Returns either the filename or nothing.

Exceptions

<code>std::invalid_argument</code>	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]
```

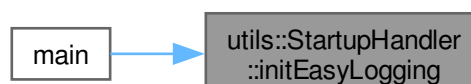
Initialize easylogging.

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](#)

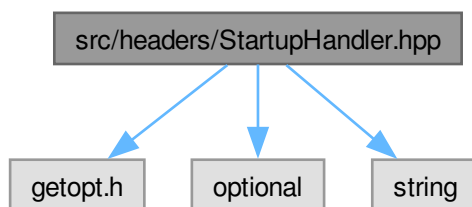
Chapter 8

File Documentation

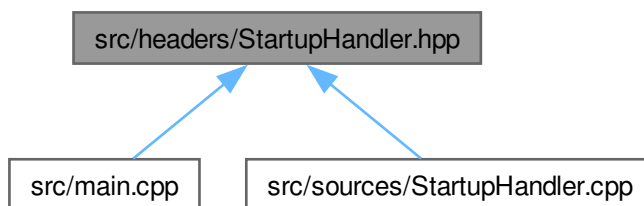
8.1 src/headers/StartupHandler.hpp File Reference

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

Include dependency graph for StartupHandler.hpp:



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



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.](#)

```

00001 #ifndef IS_WINDOWS
00002 #include <xgetopt.h>
00003 #else
00004 #include <getopt.h>
00005 #endif
00006
00007 #include <optional>
00008 #include <string>
00009
00017 namespace utils {
00028 class StartupHandler {
00029 public:
00040     static void initEasyLogging();
00041
00070     static std::optional<std::string> getOptions(int argc, char* argv[]);
00071
00072 private:
00079     StartupHandler() = default;
00080
00087     StartupHandler(const StartupHandler &) = delete;
00088
00095     StartupHandler &operator=(const StartupHandler &) = delete;
00096
00097 };
00098 } // namespace utils

```

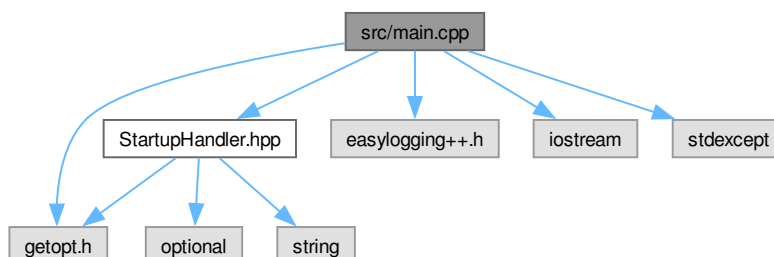
8.3 src/main.cpp File Reference

```

#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.

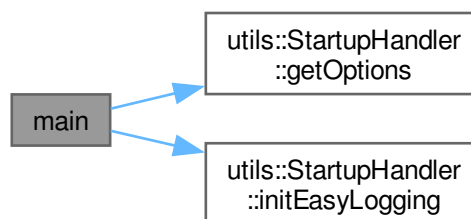
Todo • Remove Debug once getopt is working

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:



8.4 main.cpp

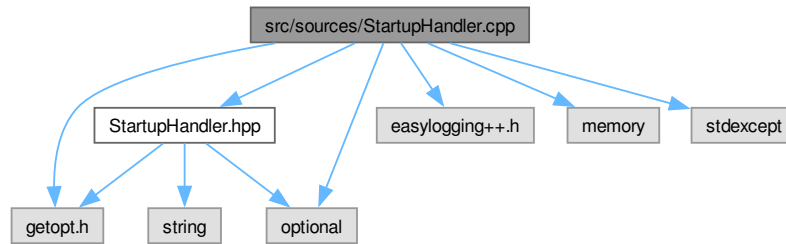
[Go to the documentation of this file.](#)

```
00001 #include "StartupHandler.hpp"
00002 #include "easylogging++.h"
00003
00004 #ifdef IS_WINDOWS
00010 #include <xgetopt.h>
00011 #else
00012 #include <getopt.h>
00013 #endif
00014
00015 #include <iostream>
00016 #include <stdexcept>
00017
00018 INITIALIZE_EASYLOGGINGPP
00019
00034 int main(int argc, char *argv[]) {
00035     //Debug
00036     #ifdef IS_LINUX
00037         std::cout << "Linux\n\n" << std::endl;
00038     #elif IS_WINDOWS
00039         std::cout << "Windows\n\n" << std::endl;
00040     #else
00041         std::cout << "Unknown OS\n\n" << std::endl;
00042     #endif
00043     utils::StartupHandler::initEasyLogging();
00044
00045     if (argc <= 1) {
00046         LOG(WARNING) << "No arguments provided, exiting!";
00047         std::cout << "No arguments provided, exiting!\n";
00048         return 1;
00049     }
00050
00051     std::cout << "Hello, World!" << std::endl;
00052     std::optional<std::string> filename;
00053
00054     try {
00055         filename = utils::StartupHandler::getOptions(argc, argv);
00056     } catch (const std::invalid_argument &e) {
00057         LOG(WARNING) << "Caught invalid argument: " << e.what();
00058         std::cout << "Invalid argument: " << e.what() << std::endl;
00059     }
00060
00061     if (!filename.has_value()) {
00062         LOG(ERROR) << "No filename given! Exiting...";
00063         std::cerr << "No filename given!\nExiting...\n";
00064         return 1;
00065     }
00066
00067     LOG(INFO) << "Filename received: " << filename.value();
00068     std::cout << "Filename: " << filename.value() << std::endl;
00069     LOG(INFO) << "Further processing...";
00070     std::cout << "Further processing..." << std::endl;
00071     LOG(INFO) << "Application exiting!";
00072     return 0;
00073 }
```

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.](#)

```

00001 #include "StartupHandler.hpp"
00002 #include "easylogging++.h"
00003
00004 #ifdef IS_WINDOWS
00005 #include <xgetopt.h>
00006 #else
00007 #include <getopt.h>
00008 #endif
00009
00010 #include <memory>
00011 #include <optional>
00012 #include <stdexcept>
00013
00014 namespace utils {
00015
00016 static int verbose = 0;
00017
00018 void StartupHandler::initEasyLogging()
00019 {
00020     el::Configurations conf("conf/easylogging.conf");
00021     el::Loggers::reconfigureLogger("default", conf);
00022     el::Loggers::reconfigureAllLoggers(conf);
00023     LOG(INFO) << "Easylogging initialized!";
00024 }
00025
00026 std::optional<std::string> StartupHandler::getOptions(int argc, char* argv[])
00027 {
00028     LOG(INFO) << "Parsing options...";
00029     static const struct option long_options[] = {
00030         /* These options set a flag. */
00031         {"verbose", no_argument, &verbose, 1},
00032         {"brief", no_argument, &verbose, 0},
00033         {"help", no_argument, nullptr, 'h'},
00034         {"version", no_argument, nullptr, 'V'},
00035         {"test", required_argument, nullptr, 0},
00036         nullptr
00037     };
00038

```

```

00039     do {
00040         int optIndex = -1;
00041         std::unique_ptr<struct option> opt = nullptr;
00042         auto result = getopt_long(argc, argv, "hV", long_options, &optIndex);
00043
00044         if (result == -1) {
00045             break;
00046         }
00047
00048         switch (result) {
00049             case '?':
00050                 LOG(INFO) << "Unknown option given";
00051                 std::cout << "Not know\n";
00052                 break;
00053
00054             case 'h':
00055                 LOG(INFO) << "Help option given";
00056                 std::cout << "long h\n";
00057                 break;
00058
00059             case 'V':
00060                 LOG(INFO) << "Version option given";
00061                 std::cout << "long V\n";
00062
00063             case 'O':
00064                 opt = std::make_unique<struct option>(long_options[optIndex]);
00065                 LOG(INFO) << "Option " << opt->name << " given";
00066
00067                 if (opt->has_arg == required_argument) {
00068                     LOG(INFO) << "Argument: " << optarg;
00069                 }
00070
00071                 break;
00072
00073             default:
00074                 std::cout << "I shouldnt have been here!\n";
00075                 break;
00076         }
00077     } while (true);
00078
00079     LOG(INFO) << "Parsing options done";
00080     std::optional<std::string> filename = {};
00081     LOG(INFO) << "Parsing other arguments...";
00082
00083     while (optind < argc) {
00084         if (filename.has_value()) {
00085             LOG(ERROR) << "Only one filename can be given!";
00086             throw std::invalid_argument("Only one filename can be given!\n");
00087         }
00088
00089         LOG(INFO) << "Filename set to: " << argv[optind];
00090         filename = std::string(argv[optind++]);
00091     }
00092
00093     return filename;
00094 }
00095 } // namespace utils

```

Index

Bug List, [3](#)

getOptions
 utils::StartupHandler, [14](#)

initEasyLogging
 utils::StartupHandler, [15](#)

main
 main.cpp, [19](#)
main.cpp
 main, [19](#)

operator=
 utils::StartupHandler, [15](#)

src/headers/StartupHandler.hpp, [17](#), [18](#)
src/main.cpp, [18](#), [20](#)
src/sources/StartupHandler.cpp, [20](#), [21](#)
StartupHandler
 utils::StartupHandler, [14](#)

Todo List, [1](#)

utils, [11](#)
 verbose, [11](#)
utils::StartupHandler, [13](#)
 getOptions, [14](#)
 initEasyLogging, [15](#)
 operator=, [15](#)
 StartupHandler, [14](#)

verbose
 utils, [11](#)