jsonToBatProject

0.2.0

Generated on Thu Feb 29 2024 12:13:30 for jsonToBatProject by Doxygen 1.9.8

Thu Feb 29 2024 12:13:30

Chapter 1

Bug List

Class batch::BatchCreator

HideShell is not implemented correctly

Namespace json

Name to similiar to "Json" namespace from the json library.

Member main (int argc, char ∗argv[])

Initielizes to early for config file to be loaded

Getopt is not working on Windows.

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

Global verbose flag is not working.

Member utils::StartupHandler::initEasyLogging ()

Easylogging conf only recognized when running application from source dir

Chapter 2

Todo List

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 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 batch Namespace Reference

Classes

- class BatchCreator

6.2 cli Namespace Reference

Classes

- class CliHandler

Variables

- static const std::string CLEAR_TERMINAL = "\x1b[2J"
- static const std::string RESET = "\x1b[0m"
- static const std::string BOLD = "\x1b[1m"
- static const std::string DIM = "\x1b[2m"
- static const std::string ITALIC = "\x1b[3m"
- static const std::string UNDERLINE = "\x1b[4m"
- static const std::string BLINKING = "\x1b[5m"
- static const std::string BLACK_FG = "\x1b[30m"
- static const std::string RED_FG = "\x1b[31m"
- static const std::string GREEN_FG = "\x1b[32m"
- static const std::string YELLOW_FG = "\x1b[33m"
- static const std::string BLUE_FG = "\x1b[34m"
- static const std::string MAGENTA_FG = "\x1b[35m"
- static const std::string CYAN_FG = "\x1b[36m"
- static const std::string WHITE_FG = "\x1b[37m"
- static const std::string RED_BG = "\x1b[41m"
- static const std::string GREEN_BG = "\x1b[42m"
- static const std::string YELLOW_BG = "\x1b[43m"
- static const std::string BLUE_BG = "\x1b[44m"
- static const std::string MAGENTA_BG = "\x1b[45m"
- static const std::string CYAN_BG = "\x1b[46m"
- static const std::string WHITE_BG = "\x1b[47m"
- static const std::string ERROR = "\x1b[31m\x1b[1m"

6.2.1 Variable Documentation

6.2.1.1 BLACK_FG

```
const std::string cli::BLACK_FG = "\x1b[30m"  [static]
```

Definition at line 20 of file CliHandler.hpp.

6.2.1.2 BLINKING

```
const std::string cli::BLINKING = "\x1b[5m"  [static]
```

Definition at line 19 of file CliHandler.hpp.

6.2.1.3 BLUE_BG

```
const std::string cli::BLUE_BG = "\x1b[44m"  [static]
```

Definition at line 31 of file CliHandler.hpp.

6.2.1.4 BLUE_FG

```
const std::string cli::BLUE_FG = "\x1b[34m"  [static]
```

Definition at line 24 of file CliHandler.hpp.

6.2.1.5 BOLD

```
const std::string cli::BOLD = "\x1b[1m"  [static]
```

Definition at line 15 of file CliHandler.hpp.

6.2.1.6 CLEAR_TERMINAL

```
const std::string cli::CLEAR_TERMINAL = "\x1b[2J"  [static]
```

Definition at line 13 of file CliHandler.hpp.

6.2.1.7 CYAN_BG

```
const std::string cli::CYAN_BG = "\x1b[46m"  [static]
```

Definition at line 33 of file CliHandler.hpp.

6.2.1.8 CYAN_FG

```
const std::string cli::CYAN_FG = "\x1b[36m"  [static]
```

Definition at line 26 of file CliHandler.hpp.

6.2.1.9 DIM

```
const std::string cli::DIM = "\x1b[2m"  [static]
```

Definition at line 16 of file CliHandler.hpp.

6.2.1.10 ERROR

```
const std::string cli::ERROR = "\x1b[31m\x1b[1m"  [static]
```

Definition at line 35 of file CliHandler.hpp.

6.2.1.11 GREEN_BG

```
const std::string cli::GREEN_BG = "\x1b[42m"  [static]
```

Definition at line 29 of file CliHandler.hpp.

6.2.1.12 GREEN_FG

```
const std::string cli::GREEN_FG = "\x1b[32m"  [static]
```

Definition at line 22 of file CliHandler.hpp.

6.2.1.13 ITALIC

```
const std::string cli::ITALIC = "\x1b[3m"  [static]
```

Definition at line 17 of file CliHandler.hpp.

6.2.1.14 MAGENTA_BG

```
const std::string cli::MAGENTA_BG = "\x1b[45m"  [static]
```

Definition at line 32 of file CliHandler.hpp.

6.2.1.15 MAGENTA_FG

```
const std::string cli::MAGENTA_FG = "\x1b[35m"  [static]
```

Definition at line 25 of file CliHandler.hpp.

6.2.1.16 RED_BG

```
const std::string cli::RED_BG = "\x1b[41m"  [static]
```

Definition at line 28 of file CliHandler.hpp.

6.2.1.17 RED_FG

```
const std::string cli::RED_FG = "\x1b[31m"  [static]
```

Definition at line 21 of file CliHandler.hpp.

6.2.1.18 RESET

```
const std::string cli::RESET = "\x1b[0m"  [static]
```

Definition at line 14 of file CliHandler.hpp.

6.2.1.19 UNDERLINE

```
const std::string cli::UNDERLINE = "\x1b[4m"  [static]
```

Definition at line 18 of file CliHandler.hpp.

6.2.1.20 WHITE_BG

```
const std::string cli::WHITE_BG = "\x1b[47m"  [static]
```

Definition at line 34 of file CliHandler.hpp.

6.2.1.21 WHITE_FG

```
const std::string cli::WHITE_FG = "\x1b[37m"  [static]
```

Definition at line 27 of file CliHandler.hpp.

6.2.1.22 YELLOW_BG

```
const std::string cli::YELLOW_BG = "\x1b[43m"  [static]
```

Definition at line 30 of file CliHandler.hpp.

6.2.1.23 YELLOW_FG

```
const std::string cli::YELLOW_FG = "\x1b[33m"  [static]
```

Definition at line 23 of file CliHandler.hpp.

6.3 clInterface Namespace Reference

6.4 json Namespace Reference

json namespace

Classes

- class JSONData
- class JSONHandler

  JSONHandler class.

6.4.1 Detailed Description

json namespace

The json namespace contains all classes and functions related to the json parsing and handling.

Bug Name to similiar to "Json" namespace from the json library.

6.5 utils Namespace Reference

Namespace for utility functions.

Classes

- class LogAndOut
- class StartupHandler

  Handles startup task for the application.
- class VerboseHandler

Enumerations

- enum class LogLevel {
  INFO , WARNING , ERROR , FATAL ,
  DEBUG , OUT }

Functions

- LogAndOut log ()
- LogAndOut log (const std::string &prefix)
- LogAndOut log (LogLevel level)
- LogAndOut log (LogLevel level, const std::string &prefix)

6.5.1 Detailed Description

Namespace for utility functions.

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

6.5.2 Enumeration Type Documentation

6.5.2.1 LogLevel

```
enum class utils::LogLevel  [strong]
```

Enumerator

INFO	
WARNING	
ERROR	
FATAL	
DEBUG	
OUT	

Definition at line 16 of file LogAndOut.hpp.

6.5.3 Function Documentation

6.5.3.1 log() [1/4]

```
LogAndOut utils::log ( )  [inline]
```

Definition at line 46 of file LogAndOut.hpp.

References INFO.

Here is the caller graph for this function:

6.5.3.2 log() [2/4]

```
LogAndOut utils::log (
            const std::string & prefix )  [inline]
```

Definition at line 47 of file LogAndOut.hpp.

References INFO.

6.5.3.3 log() [3/4]

```
LogAndOut utils::log (
            LogLevel level )  [inline]
```

Definition at line 50 of file LogAndOut.hpp.

6.5.3.4 log() [4/4]

```
LogAndOut utils::log (
            LogLevel level,
            const std::string & prefix )  [inline]
```

Definition at line 51 of file LogAndOut.hpp.

Chapter 7

Class Documentation

7.1 batch::BatchCreator Class Reference

`#include <BatchCreator.hpp>`

Public Member Functions

- BatchCreator (std::shared_ptr< json::JSONData > jsonData)
- std::shared_ptr< std::ofstream > createBatchFile ()

Private Member Functions

- void writeHideShellStart ()
- void writeHideShellEnd ()
- void writeShell ()
- void writeCommands ()
- void writeEnvironmentVariables ()
- void writePathValue ()
- void writeApplication ()

Private Attributes

- std::shared_ptr< json::JSONData > jsonData
- std::shared_ptr< std::ofstream > batchFile = nullptr

7.1.1 Detailed Description

Bug HideShell is not implemented correctly

Definition at line 10 of file BatchCreator.hpp.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 BatchCreator()

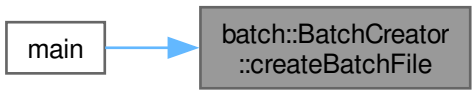
```
batch::BatchCreator::BatchCreator (
            std::shared_ptr< json::JSONData > jsonData )  [explicit]
```

7.1.3 Member Function Documentation

7.1.3.1 createBatchFile()

```
std::shared_ptr< std::ofstream > batch::BatchCreator::createBatchFile ( )
```

Here is the caller graph for this function:



7.1.3.2 writeApplication()

```
void batch::BatchCreator::writeApplication ( )  [private]
```

7.1.3.3 writeCommands()

```
void batch::BatchCreator::writeCommands ( )  [private]
```

7.1.3.4 writeEnvironmentVariables()

```
void batch::BatchCreator::writeEnvironmentVariables ( )  [private]
```

7.1.3.5 writeHideShellEnd()

```
void batch::BatchCreator::writeHideShellEnd ( )  [private]
```

7.1.3.6 writeHideShellStart()

```
void batch::BatchCreator::writeHideShellStart ( )  [private]
```

7.1.3.7 writePathValue()

```
void batch::BatchCreator::writePathValue ( )  [private]
```

7.1.3.8 writeShell()

```
void batch::BatchCreator::writeShell ( )  [private]
```

7.1.4 Member Data Documentation

7.1.4.1 batchFile

```
std::shared_ptr<std::ofstream> batch::BatchCreator::batchFile = nullptr  [private]
```

Definition at line 18 of file BatchCreator.hpp.

7.1.4.2 jsonData

```
std::shared_ptr<json::JSONData> batch::BatchCreator::jsonData  [private]
```

Definition at line 17 of file BatchCreator.hpp.

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

- src/headers/BatchCreator.hpp

7.2 cli::CliHandler Class Reference

```
#include <CliHandler.hpp>
```

7.2.1 Detailed Description

Definition at line 8 of file CliHandler.hpp.

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

- src/headers/CliHandler.hpp

7.3 json::JSONData Class Reference

```
#include <JSONData.hpp>
```

Public Member Functions

- void setOutputFile (std::string &newOutputfile)
  Set's the output file.
- void setHideShell (bool newHideShell)
  Set's the hide shell flag.
- void setApplication (const std::string &newApplication)
  Set's the application.
- void addCommand (const std::string &command)
  Add a command to the commands vector.
- void addEnvironmentVariable (const std::string &name, const std::string &value)
  Add an environment variable to the environmentVariables vector.
- void addPathValue (const std::string &pathValue)
  Add a path value to the pathValues vector.
- const std::string & getOutputFile () const
  Get the output file.
- bool getHideShell () const
  Get the hide shell flag.
- const std::optional< std::string > & getApplication () const
  Get the application.
- const std::vector< std::string > & getCommands () const
  Get the commands.
- const std::vector< std::tuple< std::string, std::string > > & getEnvironmentVariables () const
  Get the environment variables.
- const std::vector< std::string > & getPathValues () const
  Get the path values.

Private Attributes

- std::string outputfile
- bool hideShell
- std::optional< std::string > application
- std::vector< std::string > commands
- std::vector< std::tuple< std::string, std::string > > environmentVariables
- std::vector< std::string > pathValues

Static Private Attributes

- static const int8_t suffixLength = 4

7.3.1 Detailed Description

Definition at line 10 of file JSONData.hpp.

7.3.2 Member Function Documentation

7.3.2.1 addCommand()

```
void json::JSONData::addCommand (
            const std::string & command )
```

Add a command to the commands vector.

Parameters

command	The command

Exceptions

std::invalid_argument	if the command is empty

Definition at line 40 of file JSONData.cpp.

References commands, LOG_ERROR, and LOG_INFO.

7.3.2.2 addEnvironmentVariable()

```
void json::JSONData::addEnvironmentVariable (
            const std::string & name,
            const std::string & value )
```

Add an environment variable to the environmentVariables vector.

The environment variable is added as a tuple with the name and value as it's elements.

Parameters

name	The name of the environment variable
value	The value of the environment variable

Exceptions

std::invalid_argument	if the name or the value is empty

Definition at line 49 of file JSONData.cpp.

References environmentVariables, LOG_ERROR, and LOG_INFO.

7.3.2.3 addPathValue()

```
void json::JSONData::addPathValue (
            const std::string & pathValue )
```

Add a path value to the pathValues vector.

Parameters

pathValue	The path value

Exceptions

std::invalid_argument	if the pathValue is empty

Definition at line 60 of file JSONData.cpp.

References LOG_ERROR, LOG_INFO, and pathValues.

7.3.2.4 getApplication()

```
const std::optional< std::string > & json::JSONData::getApplication ( ) const  [inline]
```

Get the application.

Returns

The application

Definition at line 90 of file JSONData.hpp.

References application.

7.3.2.5 getCommands()

```
const std::vector< std::string > & json::JSONData::getCommands ( ) const  [inline]
```

Get the commands.

Returns

The commands

Definition at line 98 of file JSONData.hpp.

References commands.

7.3.2.6 getEnvironmentVariables()

```
const std::vector< std::tuple< std::string, std::string > > & json::JSONData::getEnvironment↩
Variables ( ) const  [inline]
```

Get the environment variables.

Returns

The environment variables

Definition at line 107 of file JSONData.hpp.

References environmentVariables.

7.3.2.7 getHideShell()

```
bool json::JSONData::getHideShell ( ) const  [inline]
```

Get the hide shell flag.

Returns

The hide shell flag

Definition at line 82 of file JSONData.hpp.

References hideShell.

7.3.2.8 getOutputFile()

```
const std::string & json::JSONData::getOutputFile ( ) const  [inline]
```

Get the output file.

Returns

The output file

Definition at line 74 of file JSONData.hpp.

References outputfile.

7.3.2.9 getPathValues()

```
const std::vector< std::string > & json::JSONData::getPathValues ( ) const  [inline]
```

Get the path values.

Returns

The path values

Definition at line 115 of file JSONData.hpp.

References pathValues.

7.3.2.10 setApplication()

```
void json::JSONData::setApplication (
            const std::string & newApplication )
```

Set's the application.

Parameters

application	The application

Definition at line 32 of file JSONData.cpp.

References application, and LOG_INFO.

7.3.2.11 setHideShell()

```
void json::JSONData::setHideShell (
            bool newHideShell )  [inline]
```

Set's the hide shell flag.

Parameters

hideShell	The hide shell flag

Definition at line 29 of file JSONData.hpp.

References hideShell.

7.3.2.12 setOutputFile()

```
void json::JSONData::setOutputFile (
            std::string & newOutputfile )
```

Set's the output file.

Note

If the output file does not end with .bat, the function will append .bat to the output file.

Parameters

outputfile	The output file

Exceptions

std::invalid_argument	if the outputfile is empty
std::invalid_argument	if the outputfile is already set

Definition at line 8 of file JSONData.cpp.

References LOG_ERROR, LOG_INFO, LOG_WARNING, outputfile, and suffixLength.

7.3.3 Member Data Documentation

7.3.3.1 application

```
std::optional<std::string> json::JSONData::application  [private]
```

Definition at line 122 of file JSONData.hpp.

7.3.3.2 commands

```
std::vector<std::string> json::JSONData::commands  [private]
```

Definition at line 123 of file JSONData.hpp.

7.3.3.3 environmentVariables

```
std::vector<std::tuple<std::string, std::string> > json::JSONData::environmentVariables  [private]
```

Definition at line 124 of file JSONData.hpp.

7.3.3.4 hideShell

```
bool json::JSONData::hideShell  [private]
```

Definition at line 121 of file JSONData.hpp.

7.3.3.5 outputfile

```
std::string json::JSONData::outputfile  [private]
```

Definition at line 120 of file JSONData.hpp.

7.3.3.6 pathValues

```
std::vector<std::string> json::JSONData::pathValues  [private]
```

Definition at line 125 of file JSONData.hpp.

7.3.3.7 suffixLength

```
const int8_t json::JSONData::suffixLength = 4  [static], [private]
```

Definition at line 126 of file JSONData.hpp.

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

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

7.4 json::JSONHandler Class Reference

JSONHandler class.

`#include <JSONHandler.hpp>`

Public Member Functions

- JSONHandler (const std::string &filename)
  Constructor.
- std::shared_ptr< JSONData > getJSONData ()
  Retrieve the JSONData object.

Private Member Functions

- std::shared_ptr< Json::Value > parseFile (const std::string &filename) const
  Parse a file.
- void assignOutputFile () const
  Assigns the output file to the JSONData object.
- void assignHideShell () const
  Assigns the hide shell value to the JSONData object.
- void assignApplication () const
  Assigns the application to the JSONData object.
- void assignEntries () const
  Assigns the entries to the JSONData object.
- void assignCommand (const Json::Value &entry) const
  Assigns a command to the JSONData object.
- void assignEnvironmentVariable (const Json::Value &entry) const
  Assigns an environment variable to the JSONData object.
- void assignPathValue (const Json::Value &entry) const
  Assigns a path value to the JSONData object.
- std::shared_ptr< JSONData > createJSONData ()
  Creates a JSONData object.

Private Attributes

- std::shared_ptr< Json::Value > root
- std::shared_ptr< JSONData > data

7.4.1 Detailed Description

JSONHandler class.

The JSONHandler class is responsible for parsing a json file and creating a JSONData object from it when requested. It assigns all necessary values to the JSONData object. Most of the error handling is done in the JSONData object.

Definition at line 29 of file JSONHandler.hpp.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 JSONHandler()

```
json::JSONHandler::JSONHandler (
            const std::string & filename )  [explicit]
```

Constructor.

The constructor calls the parseFile function to parse the file and adds it to the corresponding member variable.

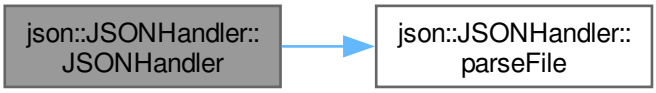
Parameters

filename	The filename to parse

Definition at line 9 of file JSONHandler.cpp.

References LOG_INFO, parseFile(), and root.

Here is the call graph for this function:



7.4.3 Member Function Documentation

7.4.3.1 assignApplication()

```
void json::JSONHandler::assignApplication ( ) const  [private]
```

Assigns the application to the JSONData object.

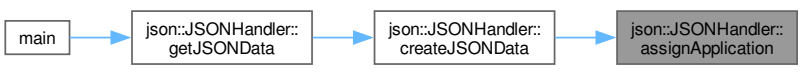
Note

How should error handling be done? Value can be empty, but what about null vs ""?

Definition at line 52 of file JSONHandler.cpp.

References data, LOG_INFO, and root.

Here is the caller graph for this function:



7.4.3.2 assignCommand()

```
void json::JSONHandler::assignCommand (
            const Json::Value & entry ) const  [private]
```

Assigns a command to the JSONData object.

The function takes a Json::Value object and assigns the command to the JSONData object

Parameters

entry	The entry to assign

Note

Error handling is done in the JSONData object

Definition at line 79 of file JSONHandler.cpp.

References data, and LOG_INFO.

Here is the caller graph for this function:

7.4.3.3 assignEntries()

```
void json::JSONHandler::assignEntries ( ) const  [private]
```

Assigns the entries to the JSONData object.

The function loops through the entries and calls the corresponding function to assign the entry to the JSONData object

Exceptions

std::runtime_error	If the entry type is unknown

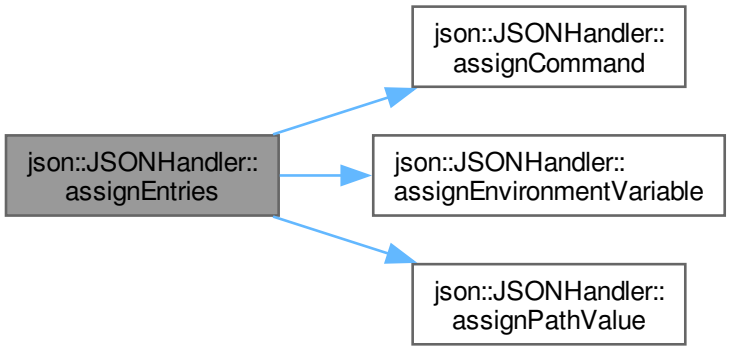
Note

Other error handling is done in the JSONData object

Definition at line 58 of file JSONHandler.cpp.

References assignCommand(), assignEnvironmentVariable(), assignPathValue(), LOG_ERROR, LOG_INFO, and root.

Here is the call graph for this function:



Here is the caller graph for this function:

7.4.3.4 assignEnvironmentVariable()

```
void json::JSONHandler::assignEnvironmentVariable (
            const Json::Value & entry ) const  [private]
```

Assigns an environment variable to the JSONData object.

The function takes a Json::Value object and assigns a tuple of the environment variable to the JSONData object

Parameters

entry	The entry to assign

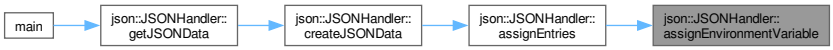
Note

Error handling is done in the JSONData object

Definition at line 85 of file JSONHandler.cpp.

References data, and LOG_INFO.

Here is the caller graph for this function:



7.4.3.5 assignHideShell()

```
void json::JSONHandler::assignHideShell ( ) const  [private]
```

Assigns the hide shell value to the JSONData object.

Note

There is no real error handling for this value, it defaults to false

Definition at line 46 of file JSONHandler.cpp.

References data, LOG_INFO, and root.

Here is the caller graph for this function:

7.4.3.6 assignOutputFile()

```
void json::JSONHandler::assignOutputFile ( ) const  [private]
```

Assigns the output file to the JSONData object.

Note

Error handling is done in the JSONData object

Definition at line 40 of file JSONHandler.cpp.

References data, LOG_INFO, and root.

Here is the caller graph for this function:

7.4.3.7 assignPathValue()

```
void json::JSONHandler::assignPathValue (
            const Json::Value & entry ) const  [private]
```

Assigns a path value to the JSONData object.

The function takes a Json::Value object and assigns the path value to the JSONData object

Parameters

entry	The entry to assign

Note

Error handling is done in the JSONData object

Definition at line 92 of file JSONHandler.cpp.

References data, and LOG_INFO.

Here is the caller graph for this function:

7.4.3.8 createJSONData()

```
std::shared_ptr< JSONData > json::JSONHandler::createJSONData ( )  [private]
```

Creates a JSONData object.

The function creates the JSONData object and calls all the necessary methods to assign the values to the object.

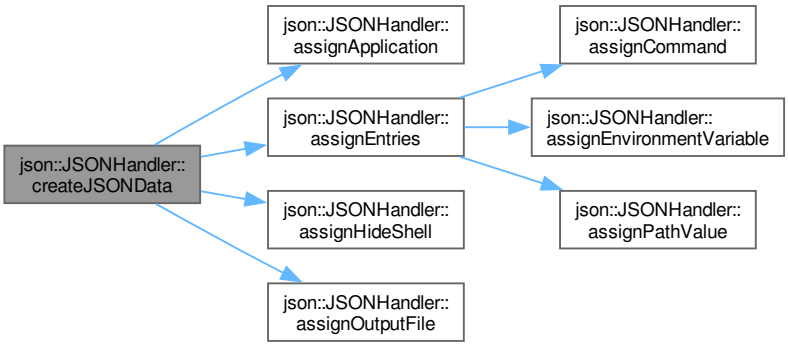
Returns

std::shared_ptr<JSONData> The JSONData object

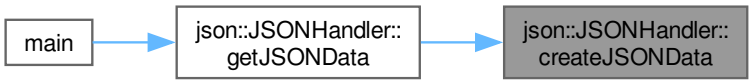
Definition at line 30 of file JSONHandler.cpp.

References assignApplication(), assignEntries(), assignHideShell(), assignOutputFile(), data, and LOG_INFO.

Here is the call graph for this function:



Here is the caller graph for this function:



7.4.3.9 getJSONData()

```
std::shared_ptr< JSONData > json::JSONHandler::getJSONData ( )
```

Retrieve the JSONData object.

The function takes the necesarry steps to create a JSONData object and then returns it

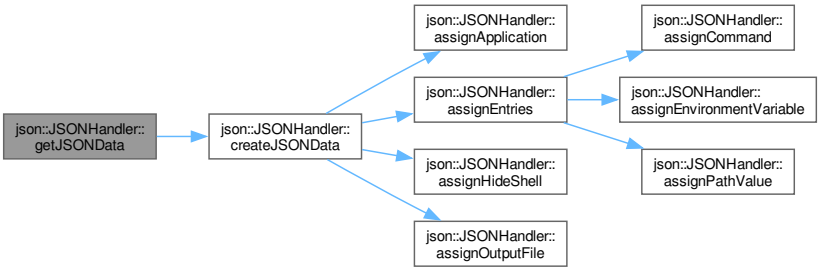
Returns

std::shared_ptr<JSONData> The JSONData object

Definition at line 25 of file JSONHandler.cpp.

References createJSONData(), and LOG_INFO.

Here is the call graph for this function:



Here is the caller graph for this function:

7.4.3.10 parseFile()

```
std::shared_ptr< Json::Value > json::JSONHandler::parseFile (
            const std::string & filename ) const  [private]
```

Parse a file.

The function takes a filename and parses the file into a Json::Value object.

Parameters

filename	The filename to parse

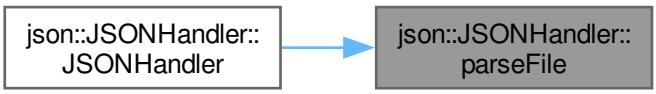
Returns

std::shared_ptr<Json::Value> The parsed file

Definition at line 15 of file JSONHandler.cpp.

References LOG_INFO.

Here is the caller graph for this function:



7.4.4 Member Data Documentation

7.4.4.1 data

```
std::shared_ptr<JSONData> json::JSONHandler::data  [private]
```

Definition at line 158 of file JSONHandler.hpp.

7.4.4.2 root

```
std::shared_ptr<Json::Value> json::JSONHandler::root  [private]
```

Definition at line 157 of file JSONHandler.hpp.

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

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

7.5 utils::LogAndOut Class Reference

```
#include <LogAndOut.hpp>
```

Public Types

- typedef std::ostream &(* Manipulator) (std::ostream &)

Public Member Functions

- LogAndOut (LogLevel newLevel)
- LogAndOut (const std::string &newPrefix, LogLevel newLevel)
- template<typename T >
  LogAndOut & operator<< (const T &val)
- LogAndOut & operator<< (Manipulator manipulator)
- ~LogAndOut ()

Private Attributes

- std::string prefix
- LogLevel level
- std::ostringstream buffer

7.5.1 Detailed Description

Note

7.5.2 Bugs and Quirks

- Automatically makes new line for cout – Nevermind?!

Definition at line 24 of file LogAndOut.hpp.

7.5.3 Member Typedef Documentation

7.5.3.1 Manipulator

```
typedef std::ostream &(* utils::LogAndOut::Manipulator) (std::ostream &)
```

Definition at line 36 of file LogAndOut.hpp.

7.5.4 Constructor & Destructor Documentation

7.5.4.1 LogAndOut() [1/2]

```
utils::LogAndOut::LogAndOut (
            LogLevel newLevel )  [inline]
```

Definition at line 26 of file LogAndOut.hpp.

7.5.4.2 LogAndOut() [2/2]

```
utils::LogAndOut::LogAndOut (
            const std::string & newPrefix,
            LogLevel newLevel )  [inline]
```

Definition at line 27 of file LogAndOut.hpp.

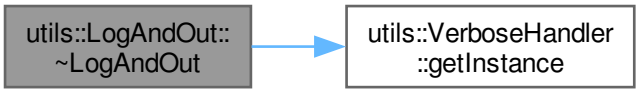
7.5.4.3 ~LogAndOut()

```
utils::LogAndOut::~LogAndOut ( )
```

Definition at line 6 of file LogAndOut.cpp.

References cli::BLACK_FG, buffer, cli::CYAN_FG, utils::DEBUG, cli::ERROR, utils::ERROR, utils::FATAL, utils::VerboseHandler::getInstance(), utils::INFO, cli::ITALIC, level, utils::OUT, prefix, cli::RED_BG, cli::RESET, utils::WARNING, and cli::YELLOW_FG.

Here is the call graph for this function:



7.5.5 Member Function Documentation

7.5.5.1 operator<<() [1/2]

```
template<typename T >
LogAndOut & utils::LogAndOut::operator<< (
            const T & val )  [inline]
```

Definition at line 31 of file LogAndOut.hpp.

References buffer.

7.5.5.2 operator<<() [2/2]

```
LogAndOut & utils::LogAndOut::operator<< (
            Manipulator manipulator )
```

Definition at line 38 of file LogAndOut.cpp.

References buffer.

7.5.6 Member Data Documentation

7.5.6.1 buffer

```
std::ostringstream utils::LogAndOut::buffer  [private]
```

Definition at line 44 of file LogAndOut.hpp.

7.5.6.2 level

```
LogLevel utils::LogAndOut::level  [private]
```

Definition at line 43 of file LogAndOut.hpp.

7.5.6.3 prefix

```
std::string utils::LogAndOut::prefix  [private]
```

Definition at line 42 of file LogAndOut.hpp.

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

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

7.6 utils::StartupHandler Class Reference

Handles startup task for the application.

```
#include <StartupHandler.hpp>
```

Public Member Functions

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

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)

7.6.1 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 26 of file StartupHandler.hpp.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 StartupHandler() [1/2]

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

Copy constructor (deleted)

This class should not be instantiated.

7.6.2.2 StartupHandler() [2/2]

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

Constructor (private)

This class should not be instantiated.

7.6.3 Member Function Documentation

7.6.3.1 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.

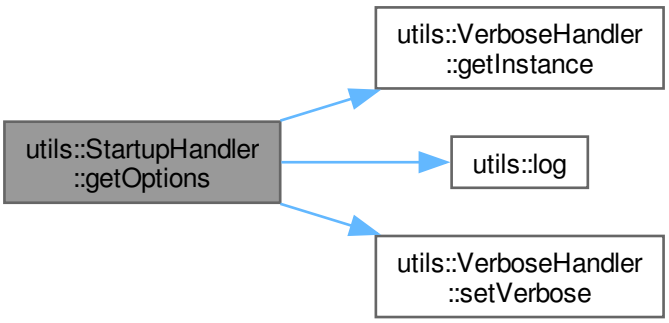
Exceptions

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

Definition at line 19 of file StartupHandler.cpp.

References utils::FATAL, utils::VerboseHandler::getInstance(), utils::log(), LOG_ERROR, LOG_INFO, and utils::VerboseHandler::setVerbose().

Here is the call graph for this function:



Here is the caller graph for this function:

7.6.3.2 initEasyLogging()

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

Initialize easylogging.

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

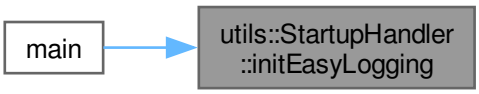
Todo • Improve easylogging configuration

Bug Easylogging conf only recognized when running application from source dir

Definition at line 12 of file StartupHandler.cpp.

References LOG_INFO.

Here is the caller graph for this function:



7.6.3.3 operator=()

```
StartupHandler & utils::StartupHandler::operator= (
            const StartupHandler &  )  [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

7.7 utils::VerboseHandler Class Reference

```
#include <Verbose.hpp>
```

Public Member Functions

- bool isVerbose () const
- void setVerbose (bool verbose)

Static Public Member Functions

- static VerboseHandler & getInstance ()

Private Member Functions

- VerboseHandler ()=default
- VerboseHandler (const VerboseHandler &)=delete
- VerboseHandler & operator= (const VerboseHandler &)=delete

Private Attributes

- bool verboseFlag = false

7.7.1 Detailed Description

Definition at line 6 of file Verbose.hpp.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 VerboseHandler() [1/2]

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

7.7.2.2 VerboseHandler() [2/2]

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

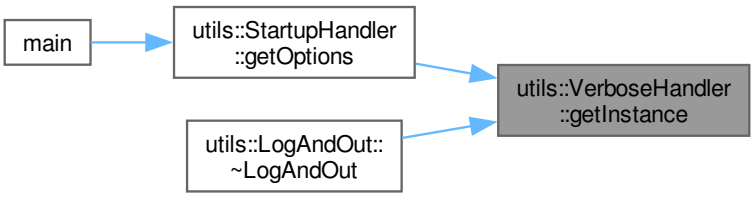
7.7.3 Member Function Documentation

7.7.3.1 getInstance()

```
static VerboseHandler & utils::VerboseHandler::getInstance ( )  [inline], [static]
```

Definition at line 8 of file Verbose.hpp.

Here is the caller graph for this function:



7.7.3.2 isVerbose()

```
bool utils::VerboseHandler::isVerbose ( ) const
```

Definition at line 5 of file Verbose.cpp.

References verboseFlag.

7.7.3.3 operator=()

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

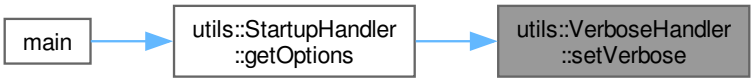
7.7.3.4 setVerbose()

```
void utils::VerboseHandler::setVerbose (
            bool verbose )
```

Definition at line 7 of file Verbose.cpp.

References LOG_INFO, and verboseFlag.

Here is the caller graph for this function:



7.7.4 Member Data Documentation

7.7.4.1 verboseFlag

```
bool utils::VerboseHandler::verboseFlag = false  [private]
```

Definition at line 22 of file Verbose.hpp.

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

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

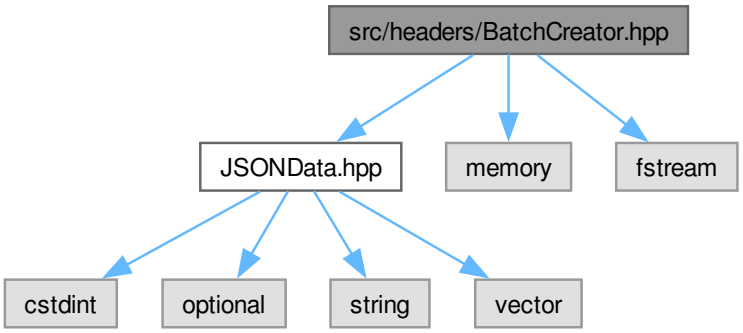
Chapter 8

File Documentation

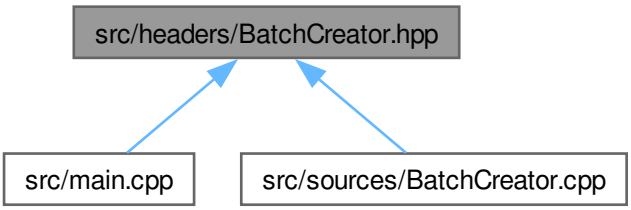
8.1 src/headers/BatchCreator.hpp File Reference

```
#include "JSONData.hpp"
#include <memory>
#include <fstream>
```

Include dependency graph for BatchCreator.hpp:



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



Classes

- class batch::BatchCreator

Namespaces

- namespace batch

8.2 BatchCreator.hpp

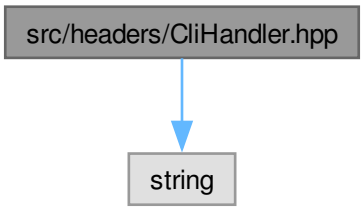
Go to the documentation of this file.

```
00001 #ifndef BATCHCREATOR_HPP
00002 #define BATCHCREATOR_HPP
00003
00004 #include "JSONData.hpp"
00005 #include <memory>
00006 #include <fstream>
00007
00008 namespace batch {
00010     class BatchCreator {
00011     public:
00012         explicit BatchCreator(std::shared_ptr<json::JSONData> jsonData);
00013
00014         std::shared_ptr<std::ofstream> createBatchFile();
00015
00016     private:
00017         std::shared_ptr<json::JSONData> jsonData;
00018         std::shared_ptr<std::ofstream> batchFile = nullptr;
00019
00020         void writeHideShellStart();
00021
00022         void writeHideShellEnd();
00023
00024         void writeShell();
00025
00026         void writeCommands();
00027
00028         void writeEnvironmentVariables();
00029
00030         void writePathValue();
00031
00032         void writeApplication();
00033
00034
00035     };
00036 } // namespace batch
00037
00038
00039 #endif // BATCHCREATOR_HPP
```

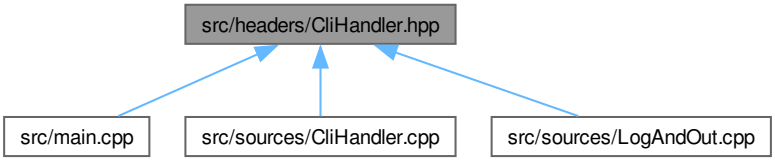
8.3 src/headers/CliHandler.hpp File Reference

`#include <string>`

Include dependency graph for CliHandler.hpp:



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



Classes

- class cli::CliHandler

Namespaces

- namespace cli

Variables

- static const std::string cli::CLEAR_TERMINAL = "\x1b[2J"
- static const std::string cli::RESET = "\x1b[0m"
- static const std::string cli::BOLD = "\x1b[1m"
- static const std::string cli::DIM = "\x1b[2m"
- static const std::string cli::ITALIC = "\x1b[3m"
- static const std::string cli::UNDERLINE = "\x1b[4m"
- static const std::string cli::BLINKING = "\x1b[5m"
- static const std::string cli::BLACK_FG = "\x1b[30m"
- static const std::string cli::RED_FG = "\x1b[31m"
- static const std::string cli::GREEN_FG = "\x1b[32m"
- static const std::string cli::YELLOW_FG = "\x1b[33m"
- static const std::string cli::BLUE_FG = "\x1b[34m"
- static const std::string cli::MAGENTA_FG = "\x1b[35m"
- static const std::string cli::CYAN_FG = "\x1b[36m"
- static const std::string cli::WHITE_FG = "\x1b[37m"
- static const std::string cli::RED_BG = "\x1b[41m"
- static const std::string cli::GREEN_BG = "\x1b[42m"
- static const std::string cli::YELLOW_BG = "\x1b[43m"
- static const std::string cli::BLUE_BG = "\x1b[44m"
- static const std::string cli::MAGENTA_BG = "\x1b[45m"
- static const std::string cli::CYAN_BG = "\x1b[46m"
- static const std::string cli::WHITE_BG = "\x1b[47m"
- static const std::string cli::ERROR = "\x1b[31m\x1b[1m"

8.4 CliHandler.hpp

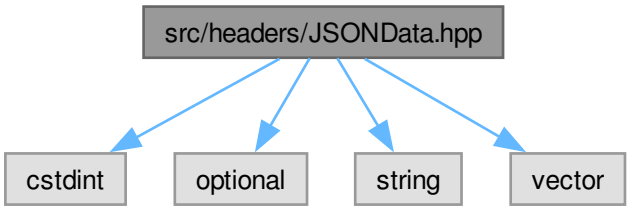
Go to the documentation of this file.

```
00001 #ifndef CLIHANDLER_HPP
00002 #define CLIHANDLER_HPP
00003
00004 #include <string>
00005 namespace cli {
00006
00007
00008 class CliHandler {};
00009
00010 #if IS_WIN32
00011 // Not supported rn
00012 #else
00013 const static std::string CLEAR_TERMINAL = "\x1b[2J";
00014 const static std::string RESET = "\x1b[0m";
00015 const static std::string BOLD = "\x1b[1m";
00016 const static std::string DIM = "\x1b[2m";
00017 const static std::string ITALIC = "\x1b[3m";
00018 const static std::string UNDERLINE = "\x1b[4m";
00019 const static std::string BLINKING = "\x1b[5m";
00020 const static std::string BLACK_FG = "\x1b[30m";
00021 const static std::string RED_FG = "\x1b[31m";
00022 const static std::string GREEN_FG = "\x1b[32m";
00023 const static std::string YELLOW_FG = "\x1b[33m";
00024 const static std::string BLUE_FG = "\x1b[34m";
00025 const static std::string MAGENTA_FG = "\x1b[35m";
00026 const static std::string CYAN_FG = "\x1b[36m";
00027 const static std::string WHITE_FG = "\x1b[37m";
00028 const static std::string RED_BG = "\x1b[41m";
00029 const static std::string GREEN_BG = "\x1b[42m";
00030 const static std::string YELLOW_BG = "\x1b[43m";
00031 const static std::string BLUE_BG = "\x1b[44m";
00032 const static std::string MAGENTA_BG = "\x1b[45m";
00033 const static std::string CYAN_BG = "\x1b[46m";
00034 const static std::string WHITE_BG = "\x1b[47m";
00035 const static std::string ERROR = "\x1b[31m\x1b[1m";
00036 #endif
00037 } // namespace cli
00038
00039 #endif // CLIHANDLER_HPP
```

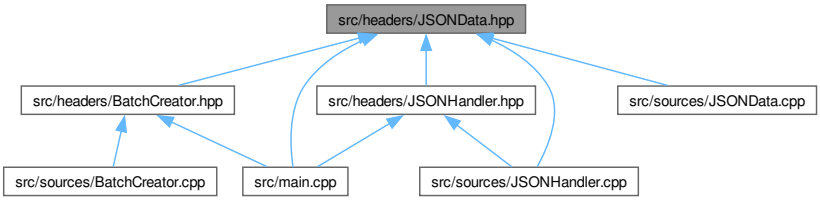
8.5 src/headers/JSONData.hpp File Reference

```
#include <cstdint>
#include <optional>
#include <string>
#include <vector>
```

Include dependency graph for JSONData.hpp:



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



Classes

- class json::JSONData

Namespaces

- namespace json

  json namespace

8.6 JSONData.hpp

Go to the documentation of this file.

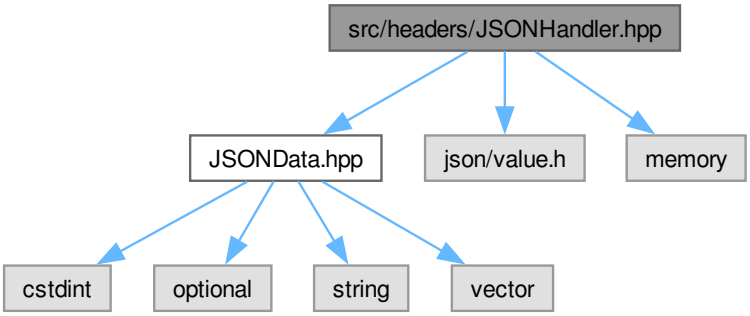
```
00001 #ifndef JSONDATA_HPP
00002 #define JSONDATA_HPP
00003
00004 #include <cstdint>
00005 #include <optional>
00006 #include <string>
00007 #include <vector>
00008
00009 namespace json {
00010     class JSONData {
00011     public:
00023         void setOutputFile(std::string &newOutputfile);
00024
00029         void setHideShell(bool newHideShell) {
00030             this->hideShell = newHideShell;
00031         }
00032
00037         void setApplication(const std::string &newApplication);
00038
00045         void addCommand(const std::string &command);
00046
00059         void addEnvironmentVariable(const std::string &name,
00060                                     const std::string &value);
00061
00068         void addPathValue(const std::string &pathValue);
00069
00074         [[nodiscard]] const std::string &getOutputFile() const {
00075             return outputfile;
00076         }
00077
00082         [[nodiscard]] bool getHideShell() const {
00083             return hideShell;
00084         }
00085
00090         [[nodiscard]] const std::optional<std::string> &getApplication() const {
00091             return application;
00092         }
00093
00098         [[nodiscard]] const std::vector<std::string> &getCommands() const {
00099             return commands;
00100         }
```

```
00101
00106         [[nodiscard]] const std::vector<std::tuple<std::string, std::string»
00107         &getEnvironmentVariables() const {
00108             return environmentVariables;
00109         }
00110
00115         [[nodiscard]] const std::vector<std::string> &getPathValues() const {
00116             return pathValues;
00117         }
00118
00119     private:
00120         std::string outputfile;
00121         bool hideShell;
00122         std::optional<std::string> application;
00123         std::vector<std::string> commands;
00124         std::vector<std::tuple<std::string, std::string» environmentVariables;
00125         std::vector<std::string> pathValues;
00126         const static int8_t suffixLength = 4;
00127     };
00128 } // namespace json
00129
00130 #endif // JSONDATA_HPP
```

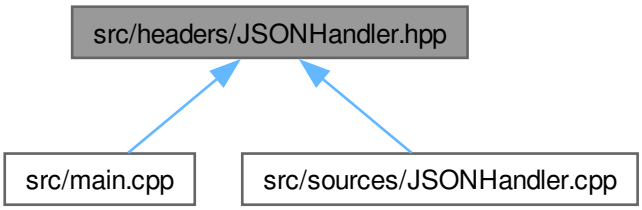
8.7 src/headers/JSONHandler.hpp File Reference

```
#include "JSONData.hpp"
#include "json/value.h"
#include <memory>
```

Include dependency graph for JSONHandler.hpp:



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



Classes

- class json::JSONHandler

  JSONHandler class.

Namespaces

- namespace json

  json namespace

8.8 JSONHandler.hpp

Go to the documentation of this file.

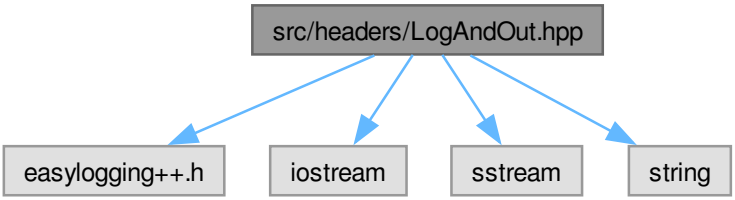
```
00001 #ifndef JSONHANDLER_HPP
00002 #define JSONHANDLER_HPP
00003
00004 #include "JSONData.hpp"
00005 #include "json/value.h"
00006 #include <memory>
00007
00017 namespace json {
00018
00029     class JSONHandler {
00030     public:
00040         explicit JSONHandler(const std::string &filename);
00041
00051         std::shared_ptr<JSONData> getJSONData();
00052
00053     private:
00064         [[nodiscard]] std::shared_ptr<Json::Value> parseFile(const std::string &filename) const;
00065
00072         void assignOutputFile() const;
00073
00080         void assignHideShell() const;
00081
00089         void assignApplication() const;
00090
00102         void assignEntries() const;
00103
00116         void assignCommand(const Json::Value &entry) const;
00117
00130         void assignEnvironmentVariable(const Json::Value &entry) const;
00131
00144         void assignPathValue(const Json::Value &entry) const;
00145
00155         std::shared_ptr<JSONData> createJSONData();
00156
00157         std::shared_ptr<Json::Value> root;
00158         std::shared_ptr<JSONData> data;
00159     };
00160 } // namespace json
00161
00162 #endif // JSONHANDLER_HPP
```

8.9 src/headers/LogAndOut.hpp File Reference

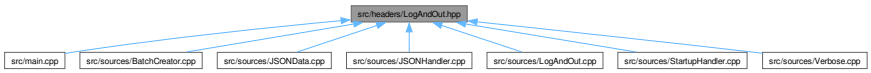
```
#include <easylogging++.h>
#include <iostream>
#include <sstream>
```

```
#include <string>
```

Include dependency graph for LogAndOut.hpp:



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



Classes

- class utils::LogAndOut

Namespaces

- namespace utils

  Namespace for utility functions.

Macros

- #define LOG_INFO utils::log(utils::LogLevel::INFO, "-- ")
- #define LOG_ERROR utils::log(utils::LogLevel::ERROR)
- #define LOG_WARNING utils::log(utils::LogLevel::WARNING)
- #define LOG_DEBUG utils::log(utils::LogLevel::DEBUG)
- #define OUTPUT utils::log(utils::LogLevel::OUT, "Output: ")

Enumerations

- enum class utils::LogLevel {
  utils::INFO , utils::WARNING , utils::ERROR , utils::FATAL ,
  utils::DEBUG , utils::OUT }

Functions

- LogAndOut utils::log ()
- LogAndOut utils::log (const std::string &prefix)
- LogAndOut utils::log (LogLevel level)
- LogAndOut utils::log (LogLevel level, const std::string &prefix)

8.9.1 Macro Definition Documentation

8.9.1.1 LOG_DEBUG

```
#define LOG_DEBUG utils::log(utils::LogLevel::DEBUG)
```

Definition at line 13 of file LogAndOut.hpp.

8.9.1.2 LOG_ERROR

```
#define LOG_ERROR utils::log(utils::LogLevel::ERROR)
```

Definition at line 11 of file LogAndOut.hpp.

8.9.1.3 LOG_INFO

```
#define LOG_INFO utils::log(utils::LogLevel::INFO, "-- ")
```

Definition at line 10 of file LogAndOut.hpp.

8.9.1.4 LOG_WARNING

```
#define LOG_WARNING utils::log(utils::LogLevel::WARNING)
```

Definition at line 12 of file LogAndOut.hpp.

8.9.1.5 OUTPUT

```
#define OUTPUT utils::log(utils::LogLevel::OUT, "Output:  ")
```

Definition at line 14 of file LogAndOut.hpp.

8.10 LogAndOut.hpp

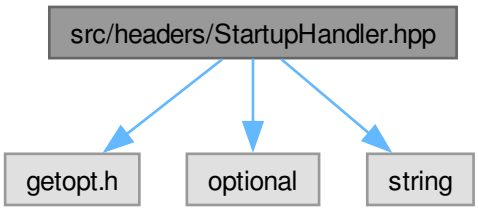
Go to the documentation of this file.

```
00001 #ifndef LOGANDOUT_HPP
00002 #define LOGANDOUT_HPP
00003
00004 #include <easylogging++.h>
00005 #include <iostream>
00006 #include <sstream>
00007 #include <string>
00008 namespace utils {
00009
00010 #define LOG_INFO utils::log(utils::LogLevel::INFO, "-- ")
00011 #define LOG_ERROR utils::log(utils::LogLevel::ERROR)
00012 #define LOG_WARNING utils::log(utils::LogLevel::WARNING)
00013 #define LOG_DEBUG utils::log(utils::LogLevel::DEBUG)
00014 #define OUTPUT utils::log(utils::LogLevel::OUT, "Output: ")
00015
00016 enum class LogLevel { INFO, WARNING, ERROR, FATAL, DEBUG,
00017   OUT, };
00018
00024 class LogAndOut {
00025 public:
00026   LogAndOut(LogLevel newLevel) : level(newLevel) {}
00027   LogAndOut(const std::string &newPrefix, LogLevel newLevel)
00028
00029       : prefix(newPrefix), level(newLevel) {}
00030
00031   template <typename T> LogAndOut &operator«(const T &val) {
00032     buffer « val;
00033     return *this;
00034   }
00035
00036   typedef std::ostream &(*Manipulator)(std::ostream &);
00037   LogAndOut &operator«(Manipulator manipulator);
00038
00039   ~LogAndOut();
00040
00041 private:
00042   std::string prefix;
00043   LogLevel level;
00044   std::ostringstream buffer;
00045 };
00046 inline LogAndOut log() { return LogAndOut(LogLevel::INFO); }
00047 inline LogAndOut log(const std::string &prefix) {
00048   return LogAndOut(prefix, LogLevel::INFO);
00049 }
00050 inline LogAndOut log(LogLevel level) { return LogAndOut(level); }
00051 inline LogAndOut log(LogLevel level, const std::string &prefix) {
00052   return LogAndOut(prefix, level);
00053 }
00054 } // namespace utils
00055
00056 #endif // LOGANDOUT_HPP
```

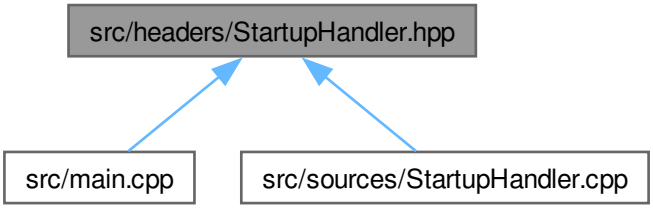
8.11 src/headers/StartupHandler.hpp File Reference

```
#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.

8.12 StartupHandler.hpp

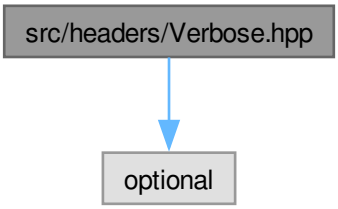
Go to the documentation of this file.

```
00001 #ifndef STARTUPHANDLER_HPP
00002 #define STARTUPHANDLER_HPP
00003
00004 #include <getopt.h>
00005 #include <optional>
00006 #include <string>
00007
00015 namespace utils {
```

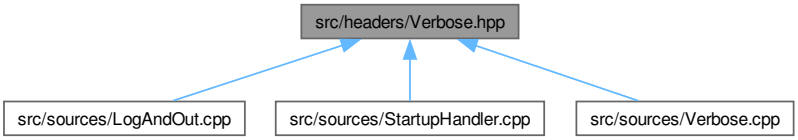
```
00026     class StartupHandler {
00027     public:
00042         static void initEasyLogging();
00043
00072         static std::optional<std::string> getOptions(int argc, char *argv[]);
00073
00080         StartupHandler(const StartupHandler &) = delete;
00081
00088         StartupHandler &operator=(const StartupHandler &) = delete;
00089
00090     private:
00097         StartupHandler() = default;
00098
00099
00100     };
00101 } // namespace utils
00102 #endif // STARTUPHANDLER_HPP
```

8.13 src/headers/Verbose.hpp File Reference

`#include <optional>`
Include dependency graph for Verbose.hpp:



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



Classes

- class utils::VerboseHandler

Namespaces

- namespace utils

  Namespace for utility functions.

8.14 Verbose.hpp

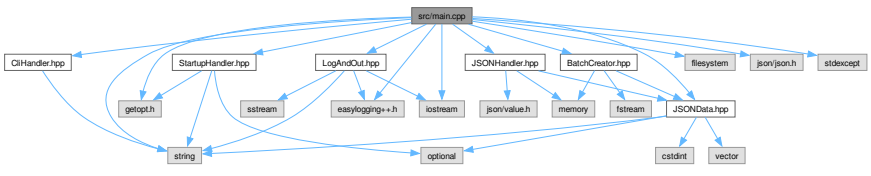
Go to the documentation of this file.

```
00001 #ifndef VERBOSE_HANDLER_HPP
00002 #define VERBOSE_HANDLER_HPP
00003
00004 #include <optional>
00005 namespace utils {
00006 class VerboseHandler {
00007 public:
00008   static VerboseHandler &getInstance() {
00009     static VerboseHandler instance;
00010     return instance;
00011   }
00012
00013   bool isVerbose() const;
00014
00015   void setVerbose(bool verbose);
00016
00017 private:
00018   VerboseHandler() = default;
00019   VerboseHandler(const VerboseHandler &) = delete;
00020   VerboseHandler &operator=(const VerboseHandler &) = delete;
00021
00022   bool verboseFlag = false;
00023 };
00024 }
00025
00026 #endif
```

8.15 src/main.cpp File Reference

```
#include "BatchCreator.hpp"
#include "CliHandler.hpp"
#include "JSONData.hpp"
#include "JSONHandler.hpp"
#include "LogAndOut.hpp"
#include "StartupHandler.hpp"
#include <easylogging++.h>
#include <filesystem>
#include <getopt.h>
#include <iostream>
#include <json/json.h>
#include <stdexcept>
#include <string>
```

Include dependency graph for main.cpp:



Macros

- #define FL std::flush

Functions

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

8.15.1 Macro Definition Documentation

8.15.1.1 FL

```
#define FL std::flush
```

Note

Temporary till logger is fixed

Definition at line 17 of file main.cpp.

8.15.2 Function Documentation

8.15.2.1 main()

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

Main function.

Bug Initielizes to early for config file to be loaded

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.

Bug Getopt is not working on Windows.

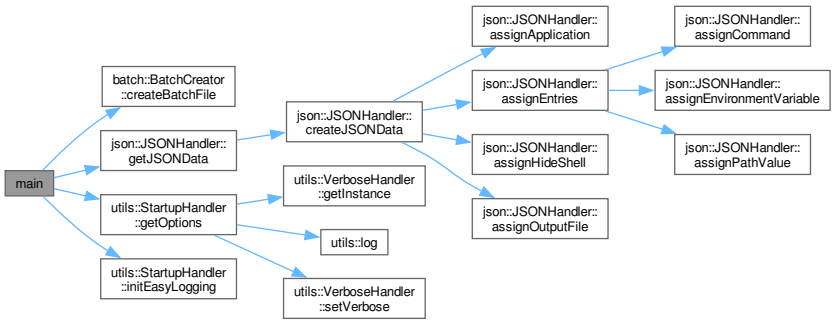
Note

maybe close in creator? But this leaves possibility to add more stuff - why?

Definition at line 34 of file main.cpp.

References cli::CLEAR_TERMINAL, batch::BatchCreator::createBatchFile(), json::JSONHandler::getJSONData(), utils::StartupHandler::getOptions(), cli::GREEN_FG, utils::StartupHandler::initEasyLogging(), cli::ITALIC, LOG_ERROR, LOG_INFO, LOG_WARNING, OUTPUT, cli::RESET, and cli::UNDERLINE.

Here is the call graph for this function:



8.16 main.cpp

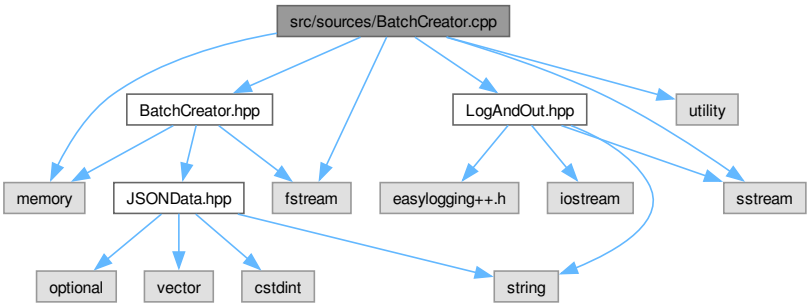
Go to the documentation of this file.

```
00001 #include "BatchCreator.hpp"
00002 #include "CliHandler.hpp"
00003 #include "JSONData.hpp"
00004 #include "JSONHandler.hpp"
00005 #include "LogAndOut.hpp"
00006 #include "StartupHandler.hpp"
00007
00008 #include <easylogging++.h>
00009 #include <filesystem>
00010 #include <getopt.h>
00011 #include <iostream>
00012 #include <json/json.h>
00013 #include <stdexcept>
00014 #include <string>
00015
00017 #define FL std::flush
00018
00020 INITIALIZE_EASYLOGGINGPP
00021
00034 int main(int argc, char *argv[]) {
00035   std::cout « cli::CLEAR_TERMINAL;
00036   utils::StartupHandler::initEasyLogging();
00037   LOG_INFO « cli::ITALIC « "Starting Application..." « cli::RESET « "\n\n";
00038   if (argc <= 1) {
00039     LOG_WARNING « "No arguments provided, exiting!\n";
00040     return 1;
00041   }
00042
00043   std::optional<std::string> filename;
00044
00045   try {
00046     filename = utils::StartupHandler::getOptions(argc, argv);
00047   } catch (const std::invalid_argument &e) {
00048     LOG_ERROR « "Caught invalid argument: " « e.what() « std::endl;
00049     return 1;
00050   }
00051
00052   if (!filename.has_value()) {
00053     LOG_ERROR « "No filename provided, exiting!\n";
00054     return 1;
00055   }
00056
00057   OUTPUT « cli::UNDERLINE « "Processing file: " « cli::ITALIC
00058          « filename.value() « cli::RESET « "\n";
00059
00060   // Initialize the JSONHandler with the file(name)
00061   json::JSONHandler jsonHandler(filename.value());
00062   // Get a JSONData object from the JSONHandler
00063   std::shared_ptr<json::JSONData> jsonData = jsonHandler.getJSONData();
00064   // Print the outputfile as a test
00065   const std::string outputfile = jsonData->getOutputFile();
00066   OUTPUT « "- Creating Outputfile: " « cli::ITALIC « outputfile « cli::RESET
00067          « "\n";
00068
00069   batch::BatchCreator batchCreator(jsonData);
00072   std::shared_ptr<std::ofstream> batchFile = batchCreator.createBatchFile();
00073
00074   std::filesystem::path path = std::filesystem::current_path();
00075   OUTPUT « "- Batch file created at: " « cli::ITALIC « path.string() « "/"
00076          « outputfile « cli::RESET « "\n\n";
00077
00078   batchFile->close();
00079
00080   OUTPUT « cli::GREEN_FG « "Done! Exiting application...\n" « cli::RESET;
00081   return 0;
00082 }
```

8.17 src/sources/BatchCreator.cpp File Reference

```
#include "BatchCreator.hpp"
#include "LogAndOut.hpp"
#include <fstream>
#include <memory>
#include <sstream>
```

#include <utility>
Include dependency graph for BatchCreator.cpp:



8.18 BatchCreator.cpp

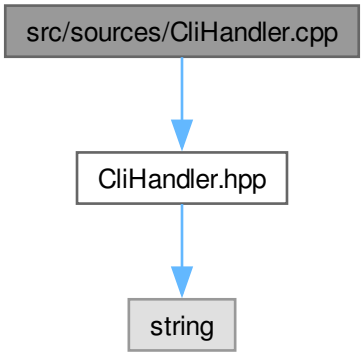
Go to the documentation of this file.

```
00001 #include "BatchCreator.hpp"
00002 #include "LogAndOut.hpp"
00003 #include <fstream>
00004 #include <memory>
00005 #include <sstream>
00006 #include <utility>
00007
00008 namespace batch {
00009
00010 BatchCreator::BatchCreator(std::shared_ptr<json::JSONData> jsonData)
00011     : jsonData(std::move(jsonData)) {
00012   LOG_INFO << "Initializing BatchCreator...\n";
00013 }
00014
00015 std::shared_ptr<std::ofstream> BatchCreator::createBatchFile() {
00016   LOG_INFO << "Creating batch file...\n";
00017   this->batchFile = std::make_shared<std::ofstream>();
00018   this->batchFile->open(this->jsonData->getOutputFile());
00019   if (this->jsonData->getHideShell()) {
00020     this->writeHideShellStart();
00021   }
00022   this->writeShell();
00023   this->writeCommands();
00024   this->writeEnvironmentVariables();
00025   this->writePathValue();
00026   this->writeApplication();
00027   if (this->jsonData->getHideShell()) {
00028     this->writeHideShellEnd();
00029   }
00030   LOG_INFO << "Batch file created! Returning it...\n";
00031   return this->batchFile;
00032 }
00033
00034 void BatchCreator::writeHideShellStart() {
00035   LOG_INFO << "Writing hide shell start...\n";
00036   *this->batchFile << "@ECHO OFF\n";
00037 }
00038
00039 void BatchCreator::writeHideShellEnd() {
00040   LOG_INFO << "Writing hide shell end...\n";
00041   *this->batchFile << "@ECHO ON\n";
00042 }
00043
00045 void BatchCreator::writeShell() {
00046   if (this->jsonData->getHideShell()) {
00047     LOG_INFO << "Writing closing shell after finish...\n";
00048     *this->batchFile << R"(START C:\Windows\System32\cmd.exe /C ")";
00049     return;
00050   }
00051   LOG_INFO << "Writing keeping shell open after finish...\n";
00052   *this->batchFile << R"(START C:\Windows\System32\cmd.exe /K ")";
00053 }
```

```
00054
00055 void BatchCreator::writeCommands() {
00056   for (const auto &command : this->jsonData->getCommands()) {
00057     LOG_INFO « "Writing command: " « command « "\n";
00058     *this->batchFile « "CALL " « command « " && ^\n";
00059   }
00060 }
00061
00062 void BatchCreator::writeEnvironmentVariables() {
00063   for (const auto &envVar : this->jsonData->getEnvironmentVariables()) {
00064     LOG_INFO « "Writing environment variable: " « std::get<0>(envVar) « "="
00065              « std::get<1>(envVar) « "\n";
00066     *this->batchFile « "SET " « std::get<0>(envVar) « "="
00067                      « std::get<1>(envVar) « " && ^\n";
00068   }
00069 }
00070
00071 void BatchCreator::writePathValue() {
00072   std::stringstream additionalPaths;
00073   for (const auto &pathValue : this->jsonData->getPathValues()) {
00074     LOG_INFO « "Writing additional path value: " « pathValue « "\n";
00075     additionalPaths « pathValue « ";";
00076   }
00077   LOG_INFO « "Adding additional paths: " « additionalPaths.str() « "\n";
00078   *this->batchFile « "SET PATH=" « additionalPaths.str() « "%PATH%";
00079 }
00080
00081 void BatchCreator::writeApplication() {
00082   if (!this->jsonData->getApplication().has_value()) {
00083     LOG_INFO « "No application provided, writing to close shell...\n";
00084     *this->batchFile « "\"\r\n";
00085     return;
00086   }
00087   LOG_INFO « "Writing application: "
00088            « this->jsonData->getApplication().value() « "\n";
00089   *this->batchFile « " && ^\n"
00090                    « this->jsonData->getApplication().value() « "\"\r\n";
00091 }
00092 } // namespace batch
```

8.19 src/sources/CliHandler.cpp File Reference

`#include "CliHandler.hpp"`
Include dependency graph for CliHandler.cpp:



Namespaces

- namespace clInterface

8.20 CliHandler.cpp

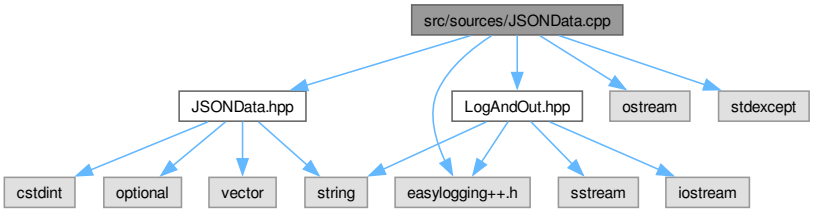
Go to the documentation of this file.

```
00001 //
00002 // Created by simon on 28.02.24.
00003 //
00004
00005 #include "CliHandler.hpp"
00006
00007 namespace clInterface {
00008 } // clInterface
```

8.21 src/sources/JSONData.cpp File Reference

```
#include "JSONData.hpp"
#include "LogAndOut.hpp"
#include <easylogging++.h>
#include <ostream>
#include <stdexcept>
```

Include dependency graph for JSONData.cpp:



Namespaces

- namespace json

  json namespace

8.22 JSONData.cpp

Go to the documentation of this file.

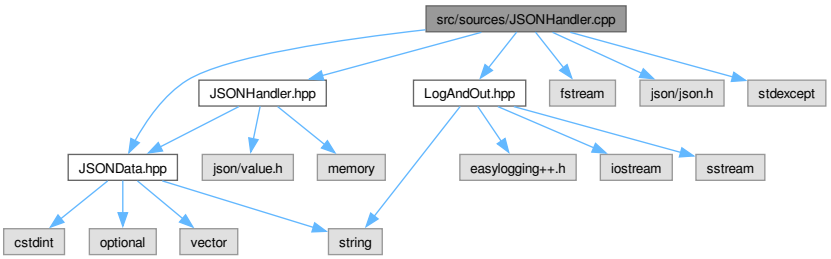
```
00001 #include "JSONData.hpp"
00002 #include "LogAndOut.hpp"
00003 #include <easylogging++.h>
00004 #include <ostream>
00005 #include <stdexcept>
00006
00007 namespace json {
00008 void JSONData::setOutputFile(std::string &newOutputfile) {
00009   LOG_INFO << "Setting outputfile to...";
00010   if (newOutputfile.empty()) {
00011     LOG_ERROR << "Tried to set empty outputfile!";
00012     throw std::invalid_argument("Outputfile cannot be empty");
00013   }
00014
00015   if (!this->outputfile.empty()) {
00016     LOG_ERROR << "Outputfile already set!";
00017     throw std::invalid_argument("Outputfile already set");
00018   }
```

```
00019
00020   if (newOutputfile.find(".bat") == std::string::npos ||
00021       newOutputfile.find(".bat") !=
00022           newOutputfile.size() - JSONData::suffixLength) {
00023     newOutputfile += ".bat";
00024     LOG_WARNING « "Outputfile does not have .bat suffix, adding it now: "
00025                 « newOutputfile;
00026   }
00027
00028   this->outputfile = newOutputfile;
00029   LOG_INFO « "Outputfile set to: " « this->outputfile « "\n";
00030 }
00031
00032 void JSONData::setApplication(const std::string &newApplication) {
00033   if (newApplication.empty()) {
00034     return;
00035   }
00036   LOG_INFO « "Setting application to: " « newApplication « "\n";
00037   this->application.emplace(newApplication);
00038 }
00039
00040 void JSONData::addCommand(const std::string &command) {
00041   if (command.empty()) {
00042     LOG_ERROR « "Tried to add empty command to data object!";
00043     throw std::invalid_argument("Command cannot be empty");
00044   }
00045   LOG_INFO « "Adding command: " « command « "\n";
00046   this->commands.push_back(command);
00047 }
00048
00049 void JSONData::addEnvironmentVariable(const std::string &name,
00050                                       const std::string &value) {
00051   if (name.empty() || value.empty()) {
00052     LOG_ERROR « "Tried to add invalid environment variable to data object!";
00053     LOG_INFO « "Envirement variables have to have a name and a value!";
00054     throw std::invalid_argument("Name and value cannot be empty");
00055   }
00056   LOG_INFO « "Adding environment variable: " « name « "=" « value « "\n";
00057   this->environmentVariables.emplace_back(name, value);
00058 }
00059
00060 void JSONData::addPathValue(const std::string &pathValue) {
00061   if (pathValue.empty()) {
00062     LOG_ERROR « "Tried to add empty path value to data object!";
00063     throw std::invalid_argument("Path value cannot be empty");
00064   }
00065   LOG_INFO « "Adding path value: " « pathValue « "\n";
00066   this->pathValues.push_back(pathValue);
00067 }
00068 } // namespace json
```

8.23 src/sources/JSONHandler.cpp File Reference

```
#include "JSONHandler.hpp"
#include "JSONData.hpp"
#include "LogAndOut.hpp"
#include <fstream>
#include <json/json.h>
#include <stdexcept>
```

Include dependency graph for JSONHandler.cpp:



Namespaces

- namespace json

  json namespace

8.24 JSONHandler.cpp

Go to the documentation of this file.

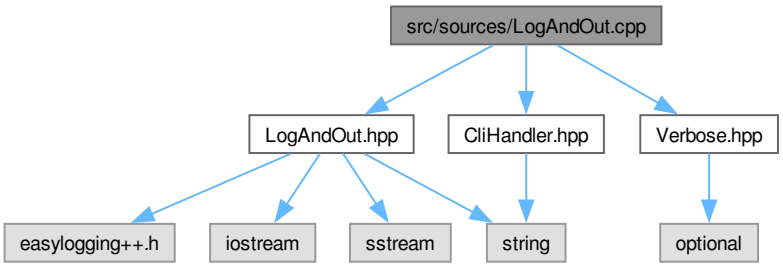
```
00001 #include "JSONHandler.hpp"
00002 #include "JSONData.hpp"
00003 #include "LogAndOut.hpp"
00004 #include <fstream>
00005 #include <json/json.h>
00006 #include <stdexcept>
00007
00008 namespace json {
00009 JSONHandler::JSONHandler(const std::string &filename) {
00010   LOG_INFO « "Initializing JSONHandler with filename: " « filename « "\n";
00011   this->root = parseFile(filename);
00012 }
00013
00014 std::shared_ptr<Json::Value>
00015 JSONHandler::parseFile(const std::string &filename) const {
00016   LOG_INFO « "Parsing file: " « filename « "\n";
00017   std::ifstream file(filename);
00018   Json::Value newRoot;
00019   Json::Reader reader;
00020   reader.parse(file, newRoot);
00021   LOG_INFO « "File parsed\n";
00022   return std::make_shared<Json::Value>(newRoot);
00023 }
00024
00025 std::shared_ptr<JSONData> JSONHandler::getJSONData() {
00026   LOG_INFO « "Creating JSONData object for return...\n";
00027   return this->createJSONData();
00028 }
00029
00030 std::shared_ptr<JSONData> JSONHandler::createJSONData() {
00031   LOG_INFO « "Creating JSONData object...\n";
00032   this->data = std::make_shared<JSONData>();
00033   this->assignOutputFile();
00034   this->assignHideShell();
00035   this->assignApplication();
00036   this->assignEntries();
00037   return this->data;
00038 }
00039
00040 void JSONHandler::assignOutputFile() const {
00041   LOG_INFO « "Assigning outputfile...\n";
00042   std::string outputFile = this->root->get("outputfile", "").asString();
00043   this->data->setOutputFile(outputFile);
00044 }
00045
00046 void JSONHandler::assignHideShell() const {
00047   LOG_INFO « "Assigning hide shell...\n";
00048   bool hideShell = this->root->get("hideshell", false).asBool();
00049   this->data->setHideShell(hideShell);
00050 }
00051
00052 void JSONHandler::assignApplication() const {
00053   LOG_INFO « "Assigning application...\n";
00054   std::string application = this->root->get("application", "").asString();
00055   this->data->setApplication(application);
00056 }
00057
00058 void JSONHandler::assignEntries() const {
00059   LOG_INFO « "Assigning entries...\n";
00060   for (const auto &entry : this->root->get("entries", "")) {
00061     std::string entryType = entry.get("type", "").asString();
00062
00063     if (entryType == "EXE") {
00064       LOG_INFO « "Calling function to assign command...\n";
00065       this->assignCommand(entry);
00066     } else if (entryType == "ENV") {
00067       LOG_INFO « "Calling function to assign environment variable...\n";
00068       this->assignEnvironmentVariable(entry);
00069     } else if (entryType == "PATH") {
00070       LOG_INFO « "Calling function to assign path value...\n";
00071       this->assignPathValue(entry);
```

```
00072     } else {
00073       LOG_ERROR << "Unknown entry type: " << entryType << "\n";
00074       throw std::invalid_argument("Unknown entry type");
00075     }
00076   }
00077 }
00078
00079 void JSONHandler::assignCommand(const Json::Value &entry) const {
00080   LOG_INFO << "Assigning command...\n";
00081   std::string command = entry.get("command", "").asString();
00082   this->data->addCommand(command);
00083 }
00084
00085 void JSONHandler::assignEnvironmentVariable(const Json::Value &entry) const {
00086   LOG_INFO << "Assigning environment variable...\n";
00087   std::string key = entry.get("key", "").asString();
00088   std::string value = entry.get("value", "").asString();
00089   this->data->addEnvironmentVariable(key, value);
00090 }
00091
00092 void JSONHandler::assignPathValue(const Json::Value &entry) const {
00093   LOG_INFO << "Assigning path value...\n";
00094   std::string pathValue = entry.get("path", "").asString();
00095   this->data->addPathValue(pathValue);
00096 }
00097 } // namespace json
```

8.25 src/sources/LogAndOut.cpp File Reference

```
#include "LogAndOut.hpp"
#include "CliHandler.hpp"
#include "Verbose.hpp"
```

Include dependency graph for LogAndOut.cpp:



Namespaces

- namespace utils

  Namespace for utility functions.

8.26 LogAndOut.cpp

Go to the documentation of this file.

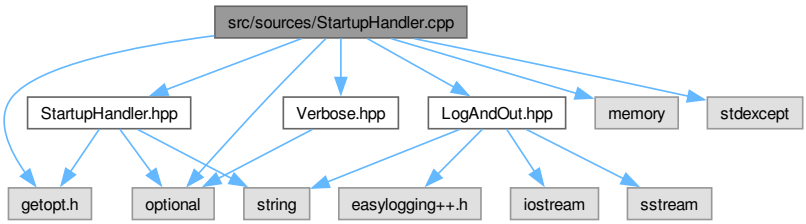
```
00001 #include "LogAndOut.hpp"
00002 #include "CliHandler.hpp"
00003 #include "Verbose.hpp"
00004
```

```
00005 namespace utils {
00006 LogAndOut::~LogAndOut() {
00007   switch (this->level) {
00008   case LogLevel::OUT:
00009     std::cout « this->buffer.str();
00010     LOG(INFO) « this->prefix « this->buffer.str();
00011     break;
00012   case LogLevel::INFO:
00013     if (VerboseHandler::getInstance().isVerbose()) {
00014       std::cout « this->prefix « this->buffer.str();
00015     }
00016     LOG(INFO) « this->prefix « this->buffer.str();
00017     break;
00018   case LogLevel::WARNING:
00019     std::cout « cli::YELLOW_FG « this->buffer.str() « cli::RESET;
00020     LOG(WARNING) « this->prefix « this->buffer.str();
00021     break;
00022   case LogLevel::ERROR:
00023     std::cerr « cli::ERROR « this->prefix « this->buffer.str() « cli::RESET;
00024     LOG(ERROR) « this->prefix « this->buffer.str();
00025     break;
00026   case LogLevel::FATAL:
00027     std::cerr « cli::BLACK_FG « cli::RED_BG « this->prefix
00028               « this->buffer.str() « cli::RESET;
00029     LOG(FATAL) « this->prefix « this->buffer.str();
00030     break;
00031   case LogLevel::DEBUG:
00032     std::cout « cli::ITALIC « cli::CYAN_FG « this->prefix
00033               « this->buffer.str() « cli::RESET;
00034     LOG(DEBUG) « this->prefix « this->buffer.str();
00035     break;
00036   }
00037 }
00038 LogAndOut &LogAndOut::operator«(Manipulator manipulator) {
00039   manipulator(std::cout);
00040   this->buffer « manipulator;
00041   return *this;
00042 }
00043 } // namespace utils
```

8.27 src/sources/StartupHandler.cpp File Reference

```
#include "StartupHandler.hpp"
#include "LogAndOut.hpp"
#include "Verbose.hpp"
#include <getopt.h>
#include <memory>
#include <optional>
#include <stdexcept>
```

Include dependency graph for StartupHandler.cpp:



Namespaces

- namespace utils

  Namespace for utility functions.

8.28 StartupHandler.cpp

Go to the documentation of this file.

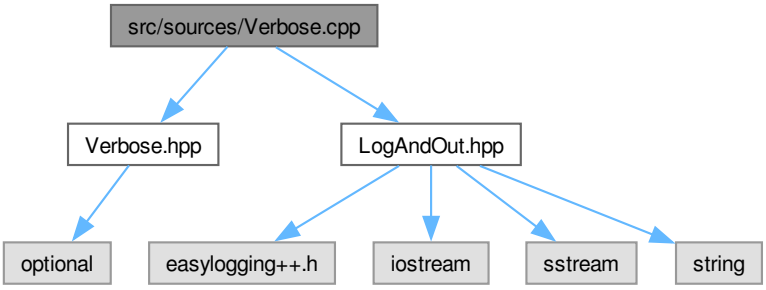
```
00001 #include "StartupHandler.hpp"
00002 #include "LogAndOut.hpp"
00003 #include "Verbose.hpp"
00004
00005 #include <getopt.h>
00006 #include <memory>
00007 #include <optional>
00008 #include <stdexcept>
00009
00010 namespace utils {
00011
00012 void StartupHandler::initEasyLogging() {
00013   el::Configurations conf("conf/easylogging.conf");
00014   el::Loggers::reconfigureLogger("default", conf);
00015   el::Loggers::reconfigureAllLoggers(conf);
00016   LOG_INFO « "Easylogging++ initialized\n";
00017 }
00018
00019 std::optional<std::string> StartupHandler::getOptions(int argc, char *argv[]) {
00020   LOG_INFO « "Parsing options...\n";
00021   int verbose = 0;
00022   static const struct option long_options[] = {
00023       /* These options set a flag. */
00024       {"verbose", no_argument, &verbose, 1},
00025       {"brief", no_argument, &verbose, 0},
00026       {"help", no_argument, nullptr, 'h'},
00027       {"version", no_argument, nullptr, 'V'},
00028       {"test", required_argument, nullptr, 0},
00029       nullptr};
00030
00031   do {
00032     int optIndex = -1;
00033     std::unique_ptr<struct option> opt = nullptr;
00034     auto result = getopt_long(argc, argv, "hV", long_options, &optIndex);
00035
00036     if (result == -1) {
00037       break;
00038     }
00039
00040     switch (result) {
00041     case '?':
00042       LOG_INFO « "Unknown option given\n";
00043       break;
00044
00045     case 'h':
00046       LOG_INFO « "Help option given\n";
00047       break;
00048
00049     case 'V':
00050       LOG_INFO « "Version option given\n";
00051       std::cout « "long V\n";
00052
00053     case 0:
00054       LOG_INFO « "Option without short version given...\n";
00055       opt = std::make_unique<struct option>(long_options[optIndex]);
00056       LOG_INFO « "Option " « opt->name « " given\n";
00057
00058       if (opt->has_arg == required_argument) {
00059         LOG_INFO « "Argument: " « optarg « "\n";
00060       }
00061       break;
00062     default:
00063       utils::log(utils::LogLevel::FATAL) « "Default case reached\n";
00064       break;
00065     }
00066   } while (true);
00067
00068   LOG_INFO « "Parsing options done\n";
00069   LOG_INFO « "Setting verbose flag...\n";
00070   VerboseHandler::getInstance().setVerbose(verbose);
00071
00072   LOG_INFO « "Parsing other arguments ...\n";
00073   std::optional<std::string> filename = {};
00074   while (optind < argc) {
00075     if (filename.has_value()) {
00076       LOG_ERROR « "Only one filename can be given right now!\n";
00077       throw std::invalid_argument("Only one filename can be given!\n");
00078     }
00079
00080     LOG_INFO « "Filename set to: " « argv[optind] « "\n";
00081     filename = std::string(argv[optind++]);
00082   }
```

```
00083
00084   return filename;
00085 }
00086 } // namespace utils
```

8.29 src/sources/Verbose.cpp File Reference

```
#include "Verbose.hpp"
#include "LogAndOut.hpp"
```

Include dependency graph for Verbose.cpp:



Namespaces

- namespace utils

  Namespace for utility functions.

8.30 Verbose.cpp

Go to the documentation of this file.

```
00001 #include "Verbose.hpp"
00002 #include "LogAndOut.hpp"
00003
00004 namespace utils {
00005 bool VerboseHandler::isVerbose() const { return this->verboseFlag; }
00006
00007 void VerboseHandler::setVerbose(bool verbose) {
00008   LOG_INFO << "Setting verbose flag to: " << verbose;
00009   this->verboseFlag = verbose;
00010 }
00011 } // namespace utils
```

Index