jsonToBatProject

0.2.0

Generated on Wed Feb 28 2024 22:07:52 for jsonToBatProject by Doxygen 1.9.8

Wed Feb 28 2024 22:07:52

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 json::JSONHandler::assignHideShell () const

: Error handling if not found

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 cliInterface Namespace Reference

Classes

- class CliHandler

6.3 json Namespace Reference

json namespace

Classes

- class JSONData
- class JSONHandler

  JSONHandler class.

6.3.1 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.

6.4 utils Namespace Reference

Namespace for utility functions.

Classes

- class StartupHandler

    Handles startup task for the application.

Variables

- static int verbose = 0

6.4.1 Detailed Description

Namespace for utility functions.

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

6.4.2 Variable Documentation

6.4.2.1 verbose

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

Definition at line 11 of file StartupHandler.cpp.

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()

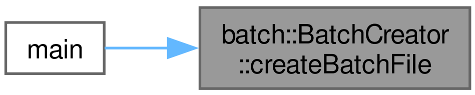
```
batch::BatchCreator::BatchCreator (
             std::shared_ptr< json::JSONData > jsonData )  [explicit]
```

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 clInterface::CliHandler Class Reference

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

7.2.1 Detailed Description

Definition at line 10 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 37 of file JSONData.cpp.

References commands.

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 46 of file JSONData.cpp.

References environmentVariables.

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 57 of file JSONData.cpp.

References 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 30 of file JSONData.cpp.

References application.

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 7 of file JSONData.cpp.

References 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.

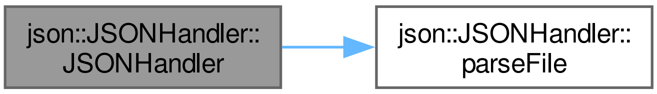
Parameters

filename	The filename to parse

Definition at line 9 of file JSONHandler.cpp.

References 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.

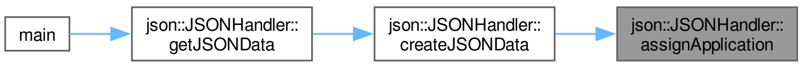
Note

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

Definition at line 47 of file JSONHandler.cpp.

References data, 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

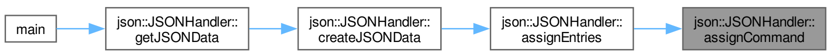
Note

Error handling is done in the JSONData object

Definition at line 69 of file JSONHandler.cpp.

References data.

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

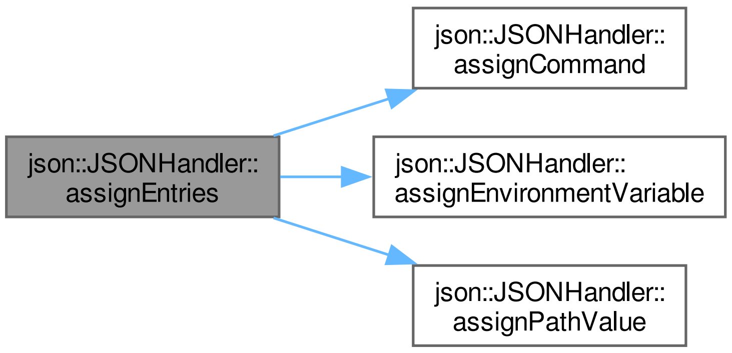
Note

Other error handling is done in the JSONData object

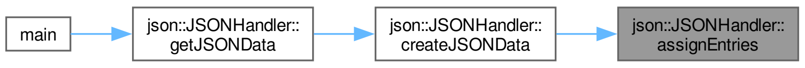
Definition at line 52 of file JSONHandler.cpp.

References assignCommand(), assignEnvironmentVariable(), assignPathValue(), 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

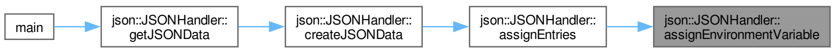
Note

Error handling is done in the JSONData object

Definition at line 74 of file JSONHandler.cpp.

References data.

Here is the caller graph for this function:


main
json::JSONHandler::getJSONData
json::JSONHandler::createJSONData
json::JSONHandler::assignEntries
json::JSONHandler::assignEnvironmentVariable

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

Todo : Error handling if not found

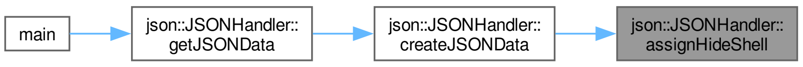
Note

: default to false

Definition at line 40 of file JSONHandler.cpp.

References data, and root.

Here is the caller graph for this function:


main
json::JSONHandler::getJSONData
json::JSONHandler::createJSONData
json::JSONHandler::assignHideShell

7.4.3.6 assignOutputFile()

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

Assigns the output file to the JSONData object.

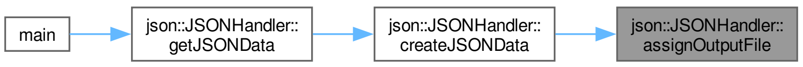
Note

Error handling is done in the JSONData object

Definition at line 35 of file JSONHandler.cpp.

References data, and root.

Here is the caller graph for this function:


main
json::JSONHandler::getJSONData
json::JSONHandler::createJSONData
json::JSONHandler::assignOutputFile

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

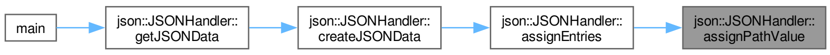
Note

Error handling is done in the JSONData object

Definition at line 80 of file JSONHandler.cpp.

References data.

Here is the caller graph for this function:


main
json::JSONHandler::getJSONData
json::JSONHandler::createJSONData
json::JSONHandler::assignEntries
json::JSONHandler::assignPathValue

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.

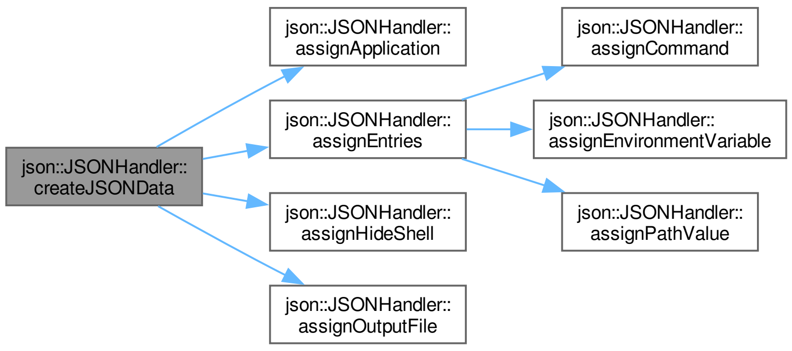
Returns

std::shared_ptr<JSONData> The JSONData object

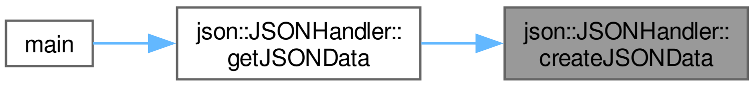
Definition at line 26 of file JSONHandler.cpp.

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

Here is the call graph for this function:


json::JSONHandler:: createJSONData
json::JSONHandler:: assignApplication
json::JSONHandler:: assignEntries
json::JSONHandler:: assignHideShell
json::JSONHandler:: assignOutputFile
json::JSONHandler:: assignCommand
json::JSONHandler:: assignEnvironmentVariable
json::JSONHandler:: assignPathValue

Here is the caller graph for this function:


main
json::JSONHandler:: getJSONData
json::JSONHandler:: createJSONData

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

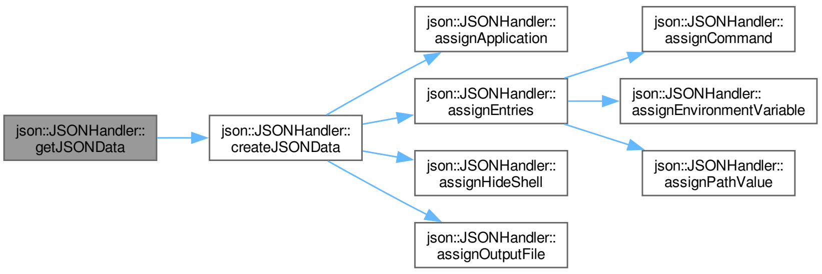
Returns

std::shared_ptr<JSONData> The JSONData object

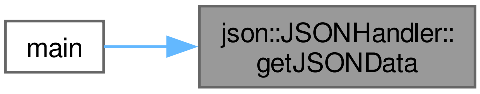
Definition at line 22 of file JSONHandler.cpp.

References createJSONData().

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

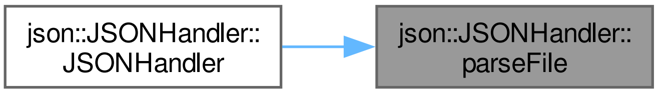
filename	The filename to parse

Returns

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

Definition at line 13 of file JSONHandler.cpp.

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::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)

7.5.1 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.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 StartupHandler() `[1/2]`

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

Copy constructor (deleted)

This class should not be instantiated.

7.5.2.2 StartupHandler() `[2/2]`

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

Constructor (private)

This class should not be instantiated.

7.5.3 Member Function Documentation

7.5.3.1 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.

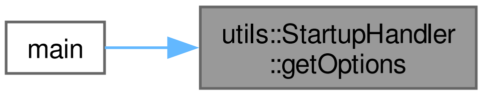
Exceptions

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

Definition at line 20 of file StartupHandler.cpp.

References utils::verbose.

Here is the caller graph for this function:



7.5.3.2 initEasyLogging()

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

Initialize easylogging.

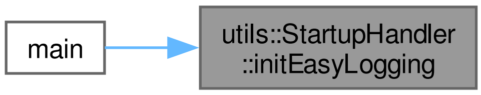
This function initializes easylogging with the configuration file "$SOURCE/conf/easylogging.conf".

Todo • Improve easylogging configuration

Bug Easylogging conf only recognized when running application from source dir

Definition at line 13 of file StartupHandler.cpp.

Here is the caller graph for this function:



7.5.3.3 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

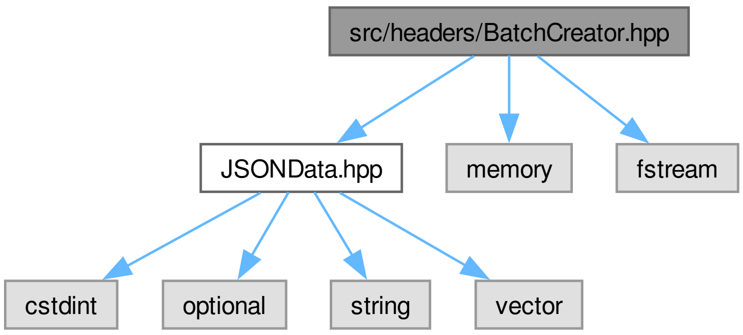
Chapter 8

File Documentation

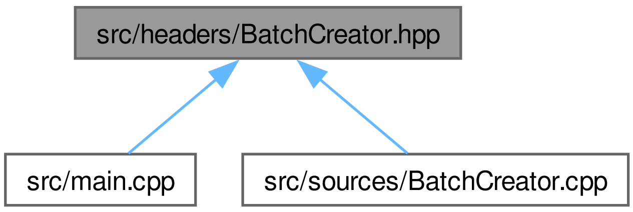
8.1 src/headers/BatchCreator.hpp File Reference

```
#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

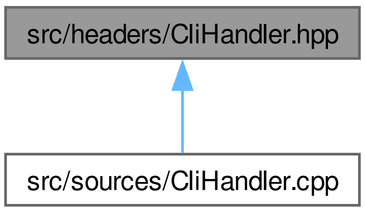
8.2 BatchCreator.hpp

Go to the documentation of this file.

```
#ifndef BATCHCREATOR_HPP
#define BATCHCREATOR_HPP

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

namespace batch {
    class BatchCreator {
    public:
        explicit BatchCreator(std::shared_ptr<json::JSONData> jsonData);

        std::shared_ptr<std::ofstream> createBatchFile();

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

        void writeHideShellStart();

        void writeHideShellEnd();

        void writeShell();

        void writeCommands();

        void writeEnvironmentVariables();

        void writePathValue();

        void writeApplication();


    };
} // namespace batch


#endif // BATCHCREATOR_HPP
```

8.3 src/headers/CliHandler.hpp File Reference

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



Classes

- class clInterface::CliHandler

Namespaces

- namespace clInterface

8.4 CliHandler.hpp

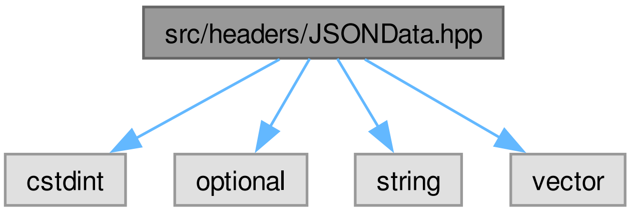
Go to the documentation of this file.

```
//
// Created by simon on 28.02.24.
//

#ifndef JSONTOBATCH_CLIHANDLER_HPP
#define JSONTOBATCH_CLIHANDLER_HPP

namespace clInterface {

    class CliHandler {

    };

} // clInterface

#endif //JSONTOBATCH_CLIHANDLER_HPP
```

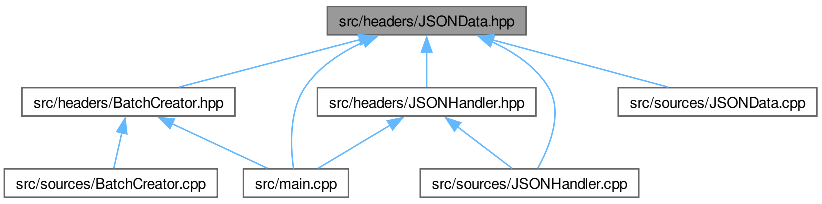
8.5 src/headers/JSONData.hpp File Reference

```
#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:


src/headers/JSONData.hpp
src/headers/BatchCreator.hpp
src/headers/JSONHandler.hpp
src/sources/JSONData.cpp
src/sources/BatchCreator.cpp
src/main.cpp
src/sources/JSONHandler.cpp

Classes

- class json::JSONData

Namespaces

- namespace json

  json namespace

8.6 JSONData.hpp

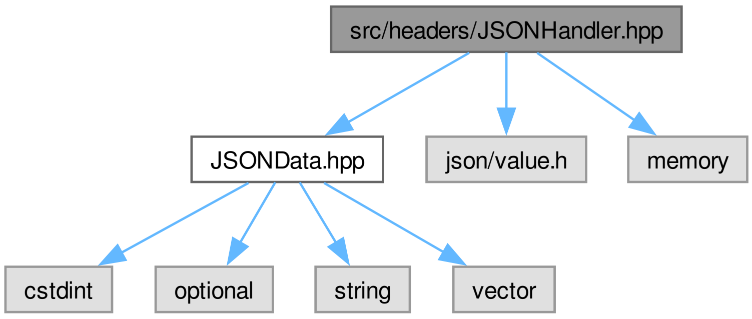
Go to the documentation of this file.

```
#ifndef JSONDATA_HPP
#define JSONDATA_HPP

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

namespace json {
    class JSONData {
    public:
        void setOutputFile(std::string &newOutputfile);

        void setHideShell(bool newHideShell) {
            this->hideShell = newHideShell;
        }

        void setApplication(const std::string &newApplication);

        void addCommand(const std::string &command);

        void addEnvironmentVariable(const std::string &name,
                                    const std::string &value);

        void addPathValue(const std::string &pathValue);

        [[nodiscard]] const std::string &getOutputFile() const {
            return outputfile;
        }

        [[nodiscard]] bool getHideShell() const {
            return hideShell;
        }

        [[nodiscard]] const std::optional<std::string> &getApplication() const {
            return application;
        }

        [[nodiscard]] const std::vector<std::string> &getCommands() const {
            return commands;
        }

        [[nodiscard]] const std::vector<std::tuple<std::string, std::string»
        &getEnvironmentVariables() const {
            return environmentVariables;
        }

        [[nodiscard]] const std::vector<std::string> &getPathValues() const {
            return pathValues;
        }

    private:
        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;
        const static int8_t suffixLength = 4;
    };
} // namespace json

#endif // JSONDATA_HPP
```

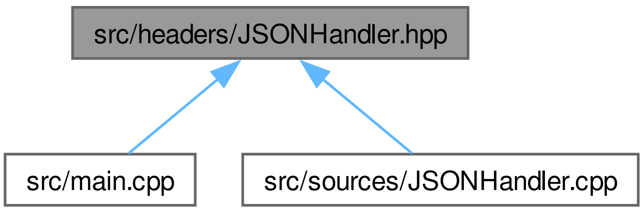
8.7 src/headers/JSONHandler.hpp File Reference

```
#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.

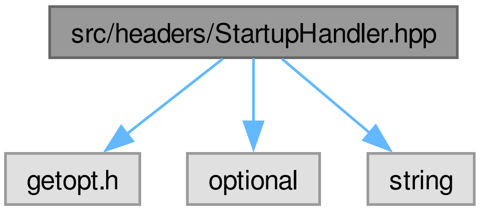
```
#ifndef JSONHANDLER_HPP
#define JSONHANDLER_HPP

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

namespace json {

    class JSONHandler {
    public:
        explicit JSONHandler(const std::string &filename);

        std::shared_ptr<JSONData> getJSONData();

    private:
        [[nodiscard]] std::shared_ptr<Json::Value> parseFile(const std::string &filename) const;

        void assignOutputFile() const;

        void assignHideShell() const;

        void assignApplication() const;

        void assignEntries() const;

        void assignCommand(const Json::Value &entry) const;

        void assignEnvironmentVariable(const Json::Value &entry) const;

        void assignPathValue(const Json::Value &entry) const;

        std::shared_ptr<JSONData> createJSONData();

        std::shared_ptr<Json::Value> root;
        std::shared_ptr<JSONData> data;
    };
} // namespace json

#endif // JSONHANDLER_HPP
```

8.9 src/headers/StartupHandler.hpp File Reference

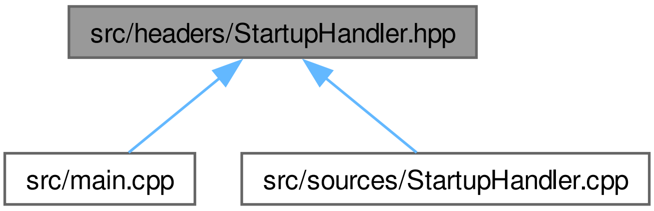
```
#include <getopt.h>
#include <optional>
```

`#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.

8.10 StartupHandler.hpp

Go to the documentation of this file.

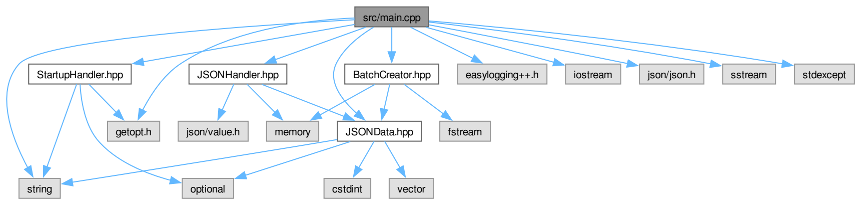
```
#ifndef STARTUPHANDLER_HPP
#define STARTUPHANDLER_HPP

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

```

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

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

        StartupHandler(const StartupHandler &) = delete;

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

    private:
        StartupHandler() = default;


    };
} // namespace utils
#endif // STARTUPHANDLER_HPP
```

8.11 src/main.cpp File Reference

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

Include dependency graph for main.cpp:



Functions

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

  Main function.

8.11.1 Function Documentation

8.11.1.1 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.

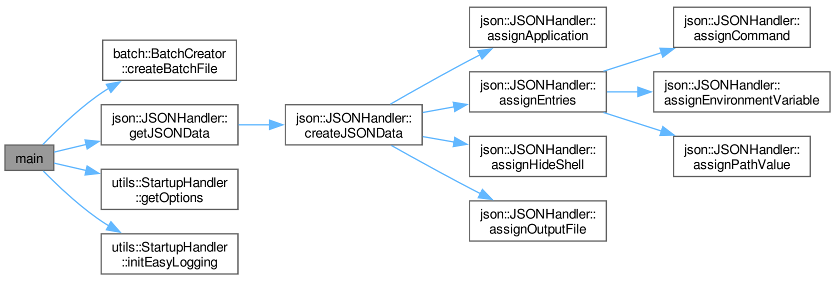
Note

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

Definition at line 29 of file main.cpp.

References batch::BatchCreator::createBatchFile(), json::JSONHandler::getJSONData(), utils::StartupHandler::getOptions(), and utils::StartupHandler::initEasyLogging().

Here is the call graph for this function:


main
batch::BatchCreator ::createBatchFile
json::JSONHandler:: getJSONData
utils::StartupHandler ::getOptions
utils::StartupHandler ::initEasyLogging
json::JSONHandler:: createJSONData
json::JSONHandler:: assignApplication
json::JSONHandler:: assignEntries
json::JSONHandler:: assignHideShell
json::JSONHandler:: assignOutputFile
json::JSONHandler:: assignCommand
json::JSONHandler:: assignEnvironmentVariable
json::JSONHandler:: assignPathValue

8.12 main.cpp

Go to the documentation of this file.

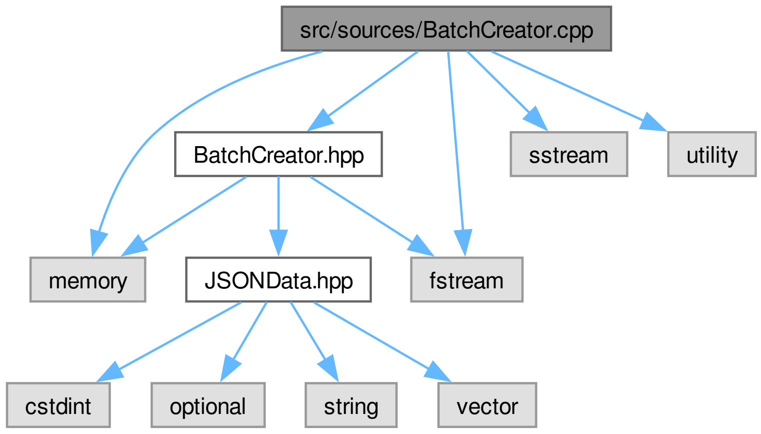
```
00001 #include "JSONData.hpp"
00002 #include "StartupHandler.hpp"
00003 #include "JSONHandler.hpp"
00004 #include "BatchCreator.hpp"
00005
00006 #include <easylogging++.h>
00007 #include <getopt.h>
00008 #include <iostream>
00009 #include <json/json.h>
00010 #include <sstream>
00011 #include <stdexcept>
00012 #include <string>
00013
00015 INITIALIZE_EASYLOGGINGPP
00016
00029 int main(int argc, char *argv[]) {
00030     std::cout « "Starting Application..." « std::endl;
00031     utils::StartupHandler::initEasyLogging();
00032
00033     if (argc <= 1) {
00034         LOG(WARNING) « "No arguments provided, exiting!";
00035         std::cout « "No arguments provided, exiting!\n";
00036         return 1;
00037     }
00038
00039     std::optional<std::string> filename;
00040
```

```
    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;

    // Initialize the JSONHandler with the file(name)
    json::JSONHandler jsonHandler(filename.value());
    // Get a JSONData object from the JSONHandler
    std::shared_ptr<json::JSONData> jsonData = jsonHandler.getJSONData();
    // Print the outputfile as a test
    std::cout « "Outputfile: " « jsonData->getOutputFile() « std::endl;

    batch::BatchCreator batchCreator(jsonData);
    std::shared_ptr<std::ofstream> batchFile = batchCreator.createBatchFile();
    batchFile->close();

    LOG(INFO) « "Application exiting!";
    return 0;
}
```

8.13 src/sources/BatchCreator.cpp File Reference

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

Include dependency graph for BatchCreator.cpp:



8.14 BatchCreator.cpp

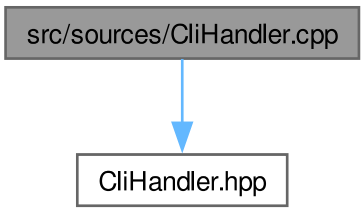
Go to the documentation of this file.

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

namespace batch {

    BatchCreator::BatchCreator(std::shared_ptr<json::JSONData> jsonData)
            : jsonData(std::move(jsonData)) {}

    std::shared_ptr<std::ofstream> BatchCreator::createBatchFile() {
        this->batchFile = std::make_shared<std::ofstream>();
        this->batchFile->open(this->jsonData->getOutputFile());
        if (this->jsonData->getHideShell()) {
            this->writeHideShellStart();
        }
        this->writeShell();
        this->writeCommands();
        this->writeEnvironmentVariables();
        this->writePathValue();
        this->writeApplication();
        if (this->jsonData->getHideShell()) {
            this->writeHideShellEnd();
        }
        return this->batchFile;
    }

    void BatchCreator::writeHideShellStart() { *this->batchFile « "@ECHO OFF\n"; }

    void BatchCreator::writeHideShellEnd() { *this->batchFile « "@ECHO ON\n"; }

    void BatchCreator::writeShell() {
        if (this->jsonData->getHideShell()) {
            *this->batchFile « R"(START C:\Windows\System32\cmd.exe /C ")";
            return;
        }
        *this->batchFile « R"(START C:\Windows\System32\cmd.exe /K ")";
    }

    void BatchCreator::writeCommands() {
        for (const auto &command: this->jsonData->getCommands()) {
            *this->batchFile « "CALL " « command « " && ^\n";
        }
    }

    void BatchCreator::writeEnvironmentVariables() {
        for (const auto &envVar: this->jsonData->getEnvironmentVariables()) {
            *this->batchFile « "SET " « std::get<0>(envVar) « "="
                             « std::get<1>(envVar) « " && ^\n";
        }
    }

    void BatchCreator::writePathValue() {
        std::stringstream additionalPaths;
        for (const auto &pathValue: this->jsonData->getPathValues()) {
            additionalPaths « pathValue « ";";
        }
        *this->batchFile « "SET PATH=%PATH%;" « additionalPaths.str();
    }

    void BatchCreator::writeApplication() {
        if (!this->jsonData->getApplication().has_value()) {
            *this->batchFile « "\"\n";
            return;
        }
        *this->batchFile « " && ^\n"
                         « this->jsonData->getApplication().value()
                         « "\"\n";
    }
} // namespace batch
```

8.15 src/sources/CliHandler.cpp File Reference

```
#include "CliHandler.hpp"
```

Include dependency graph for CliHandler.cpp:



Namespaces

- namespace clInterface

8.16 CliHandler.cpp

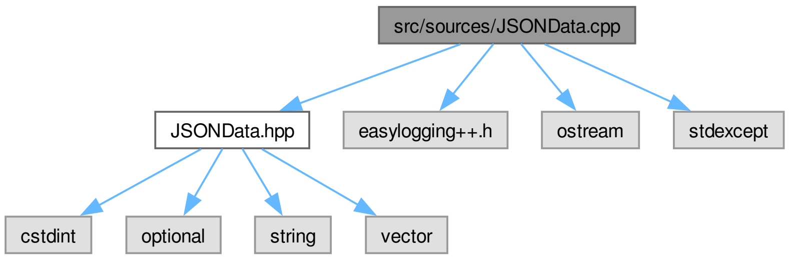
Go to the documentation of this file.

```
//
// Created by simon on 28.02.24.
//

#include "CliHandler.hpp"

namespace clInterface {
} // clInterface
```

8.17 src/sources/JSONData.cpp File Reference

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

Include dependency graph for JSONData.cpp:



Namespaces

- namespace json

  json namespace

8.18 JSONData.cpp

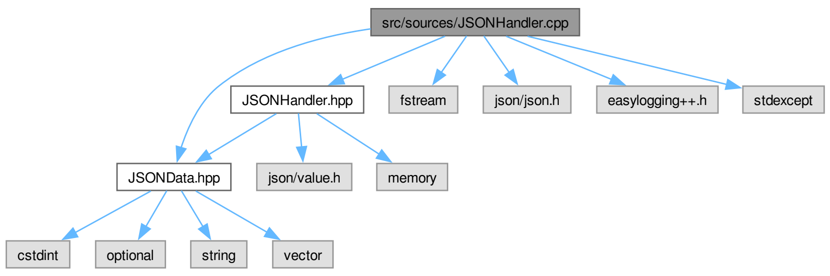
Go to the documentation of this file.

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

namespace json {
    void JSONData::setOutputFile(std::string &newOutputfile) {
        if (newOutputfile.empty()) {
            LOG(ERROR) « "Tried to set empty outputfile!";
            throw std::invalid_argument("Outputfile cannot be empty");
        }

        if (!this->outputfile.empty()) {
            LOG(ERROR) « "Outputfile already set!";
            throw std::invalid_argument("Outputfile already set");
        }

        if (newOutputfile.find(".bat") == std::string::npos ||
            newOutputfile.find(".bat") != newOutputfile.size() - JSONData::suffixLength) {
            newOutputfile += ".bat";
            std::cerr « "Outputfile does not have .bat suffix, adding it now: "
                      « newOutputfile « std::endl;
            LOG(WARNING) « "Outputfile does not have .bat suffix, adding it now: "
                         « newOutputfile;
        }

        this->outputfile = newOutputfile;
    }

    void JSONData::setApplication(const std::string &newApplication) {
        if (newApplication.empty()) {
            return;
        }
        this->application.emplace(newApplication);
    }

    void JSONData::addCommand(const std::string &command) {
        if (command.empty()) {
            LOG(ERROR) « "Tried to add emoty command to data object!";
            throw std::invalid_argument("Command cannot be empty");
        }

        this->commands.push_back(command);
    }

    void JSONData::addEnvironmentVariable(const std::string &name,
                                          const std::string &value) {
        if (name.empty() || value.empty()) {
            LOG(ERROR) « "Tried to add invalid environment variable to data object!";
            LOG(INFO) « "Envirement variables have to have a name and a value!";
            throw std::invalid_argument("Name and value cannot be empty");
        }

        this->environmentVariables.emplace_back(name, value);
    }

    void JSONData::addPathValue(const std::string &pathValue) {
        if (pathValue.empty()) {
            LOG(ERROR) « "Tried to add empty path value to data object!";
            throw std::invalid_argument("Path value cannot be empty");
        }

        this->pathValues.push_back(pathValue);
    }
} // namespace json
```

8.19 src/sources/JSONHandler.cpp File Reference

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

Include dependency graph for JSONHandler.cpp:



Namespaces

- namespace json

  json namespace

8.20 JSONHandler.cpp

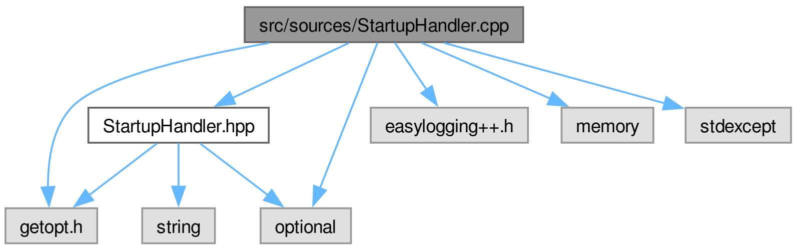
Go to the documentation of this file.

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

namespace json {
    JSONHandler::JSONHandler(const std::string &filename) {
        this->root = parseFile(filename);
    }

    std::shared_ptr<Json::Value> JSONHandler::parseFile(const std::string
                                                            &filename) const {
        std::ifstream file(filename);
        Json::Value newRoot;
        Json::Reader reader;
        reader.parse(file, newRoot);
        return std::make_shared<Json::Value>(newRoot);
    }

    std::shared_ptr<JSONData> JSONHandler::getJSONData() {
        return this->createJSONData();
    }

    std::shared_ptr<JSONData> JSONHandler::createJSONData() {
        this->data = std::make_shared<JSONData>();
        this->assignOutputFile();
        this->assignHideShell();
        this->assignApplication();
        this->assignEntries();
        return this->data;
    }

    void JSONHandler::assignOutputFile() const {
        std::string outputFile = this->root->get("outputfile", "").asString();
        this->data->setOutputFile(outputFile);
    }

    void JSONHandler::assignHideShell() const {
        bool hideShell = this->root->get("hideshell", false).asBool();
        this->data->setHideShell(hideShell);
    }

    void JSONHandler::assignApplication() const {
        std::string application = this->root->get("application", "").asString();
        this->data->setApplication(application);
    }

    void JSONHandler::assignEntries() const {
        for (const auto &entry: this->root->get("entries", "")) {
            std::string entryType = entry.get("type", "").asString();

            if (entryType == "EXE") {
                this->assignCommand(entry);
            } else if (entryType == "ENV") {
                this->assignEnvironmentVariable(entry);
            } else if (entryType == "PATH") {
                this->assignPathValue(entry);
            } else {
                LOG(ERROR) « "Unknown entry type";
                throw std::invalid_argument("Unknown entry type");
            }
        }
    }

    void JSONHandler::assignCommand(const Json::Value &entry) const {
        std::string command = entry.get("command", "").asString();
        this->data->addCommand(command);
    }

    void JSONHandler::assignEnvironmentVariable(const Json::Value &entry) const {
        std::string key = entry.get("key", "").asString();
        std::string value = entry.get("value", "").asString();
        this->data->addEnvironmentVariable(key, value);
    }

    void JSONHandler::assignPathValue(const Json::Value &entry) const {
        std::string pathValue = entry.get("path", "").asString();
        this->data->addPathValue(pathValue);
    }
} // namespace json
```

8.21 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

8.22 StartupHandler.cpp

Go to the documentation of this file.

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

#include <getopt.h>
#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