jsonToBatProject

Generated on Tue Feb 27 2024 15:40:56 for jsonToBatProject by Doxygen 1.9.8

Tue Feb 27 2024 15:40:56

1 README 1

1.1 README . . . 1

1.1.1 Precompiled . . . 2

2 Todo List 3

3 Namespace Index 5

3.1 Namespace List . . . 5

4 File Index 7

4.1 File List . . . 7

5 Namespace Documentation 9

5.1 WIP Namespace Reference . . . 9

5.1.1 Detailed Description . . . 9

5.1.2 Function Documentation . . . 9

5.1.2.1 exampleEasyLogging() . . . 9

6 File Documentation 11

6.1 README.md File Reference . . . 11

6.2 src/main.cpp File Reference . . . 11

6.2.1 Function Documentation . . . 11

6.2.1.1 main() . . . 11

6.3 main.cpp . . . 12

Index 13

Chapter 1

README

Doxygen Documentation

Sonar Cloud

1.1 README

Current workflows:

- build
  - build and test the application on:
    - windows with cl
    - ubunut with g++
    - ubuntu with clang++
- buildWithPrecompiled
  - Same as build but with the precompiled libraries
- CodeQl
  - Code security
- Doxygen Action
  - Generate Doxygen documentation
  - Deploys generated documentation to gh-pages
- Microsoft C++ Code Analysis
- pages-build-deployment
- SonarCloud
  - Static code analysis *For Scanning Alerts -> Security*

Regarding coding style (?):

- no classes in global namespace
- no "using NAMESPACE"
- 4 space indenting
- ? *setup astyle options?*

Git (?):

- no direct commits onto main (only via pull-requests)
-

Libraries

- jsoncpp
- Easyloggingpp
- Catch2

Libraries can be found in ./lib. They are subprojects and will be compiled when building the project for the first time. Alternatevly compiled versions can be found at ./lib/compiled. As is, this approach works on linux (gcc, clang) and Windows (Mingw). As steps found in the tutorial (checking for compiler in cmake) are not necessary.

1.1.1 Precompiled

By setting the flag `-DPRECOMPILED=ON` when initialising the cmake project, the precompiled versions of the libraries will be used. ***This does currently not work under windows***

Chapter 2

Todo List

Global main (int argc, char ∗argv[])

Github

- "Dev-Ops"
- Doxygen settings
- Template-Comment
- Template-Header-Comment

Global WIP::exampleEasyLogging ()

Configure easylogging properly

- outsource easylogging config
  - e.g. startup class?

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

WIP
Includes for test 9

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/main.cpp 11

Chapter 5

Namespace Documentation

5.1 WIP Namespace Reference

Includes for test.

Functions

- void exampleEasyLogging ()
  Example of how to use easylogging with a configuration file.

5.1.1 Detailed Description

Includes for test.

Namespace for work in progress.

Namespace I used for testing and trying out new things To be deleted

5.1.2 Function Documentation

5.1.2.1 exampleEasyLogging()

```
void WIP::exampleEasyLogging ( )
```

Example of how to use easylogging with a configuration file.

- This function is an example of how to use easylogging
- The configuration file is located in ../conf
- Before proper integration, config has to be done properly

  Todo

Definition at line 55 of file main.cpp.

Chapter 6

File Documentation

6.1 README.md File Reference

6.2 src/main.cpp File Reference

```
#include "easylogging++.h"
#include <iostream>
#include "catch2/catch_all.hpp"
#include "json/json.h"
```

Namespaces

- namespace WIP

  Includes for test.

Functions

- void WIP::exampleEasyLogging ()

  Example of how to use easylogging with a configuration file.
- int main (int argc, char ∗argv[ ])

  Main function.

6.2.1 Function Documentation

6.2.1.1 main()

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

Main function.

Codeconvention:

- Formatter: astyle

  Todo

Definition at line 26 of file main.cpp.

References WIP::exampleEasyLogging().

6.3 main.cpp

Go to the documentation of this file.

```
00001 #include "easylogging++.h"
00002 #include <iostream>
00003
00005 #include "catch2/catch_all.hpp"
00006 #include "json/json.h"
00007
00008 namespace WIP {
00009     void exampleEasyLogging();
00010 }
00011
00026 int main(int argc, char* argv[])
00027 {
00028     WIP::exampleEasyLogging();
00029     std::cout « "Hello, World!" « std::endl;
00030     return 0;
00031 }
00032
00033 INITIALIZE_EASYLOGGINGPP
00041 namespace WIP {
00055     void exampleEasyLogging()
00056     {
00057         el::Configurations conf("conf/easylogging.conf");
00058         el::Loggers::reconfigureLogger("default", conf);
00059         el::Loggers::reconfigureAllLoggers(conf);
00060         LOG(INFO) « "My first info log using default logger";
00061     }
00062 } // namespace WIP
```

Index

exampleEasyLogging
    WIP, 9

main
    main.cpp, 11
main.cpp
    main, 11

README, 1
README.md, 11

src/main.cpp, 11, 12

Todo List, 3

WIP, 9
    exampleEasyLogging, 9