

Exception

1.0

Generated by Doxygen 1.5.8

Sat Feb 6 20:35:37 2010

Contents

| | | |
|----------|--|-----------|
| 1 | Exception documentation | 1 |
| 1.1 | Introduction | 1 |
| 1.2 | Author | 1 |
| 1.3 | Installation | 1 |
| 1.4 | Example | 2 |
| 2 | Todo List | 3 |
| 3 | Data Structure Index | 5 |
| 3.1 | Data Structures | 5 |
| 4 | File Index | 7 |
| 4.1 | File List | 7 |
| 5 | Data Structure Documentation | 9 |
| 5.1 | Exception Class Reference | 9 |
| 5.1.1 | Detailed Description | 9 |
| 5.1.2 | Constructor & Destructor Documentation | 10 |
| 5.1.2.1 | Exception | 10 |
| 5.1.2.2 | Exception | 10 |
| 5.1.2.3 | ~Exception | 11 |
| 5.1.3 | Member Function Documentation | 11 |
| 5.1.3.1 | getError | 11 |
| 5.1.3.2 | printStacktrace | 11 |
| 5.1.3.3 | what | 11 |
| 6 | File Documentation | 13 |
| 6.1 | Exception File Reference | 13 |
| 6.2 | Exception.cpp File Reference | 14 |

Chapter 1

Exception documentation

1.1 Introduction

[Exception](#) is a C++ class inheriting from the standard exception class and implements a stacktrace option. [Exception](#) comes as a shared library and must be installed via the GNU rule of three.

1.2 Author

Markus Mazurczak <coding@markus-mazurczak.de> Please send me any comments or Bug-reports.

1.3 Installation

Unpack the `Exception_[VERSION].tar` and

- `./configure`
- `make`
- `make install`

To list all available configure options type

- `./configure --help`

To clean all build files type

- `make clean`

To uninstall `libexception` type

- `make uninstall`

By default, `make install` will install `libexception` to `/usr/local/lib` and the [Exception](#) header to `/usr/local/include`. To install the library and header to `/home/foobar/libtest` type:

- `./configure --prefix=/home/foobar/libtest`

this will result in `/home/foobar/libtest/lib/libexception.*` and `/home/foobar/libtest/include/Exception`

If you're done with coding and you are about to compile your program using the [Exception](#) class you have to link the exception library via

- `-lexception`

1.4 Example

File: test.cpp

```
#include <iostream>
#include <string>
#include <Exception>

class foo {
public:
    void bar(void) throw(Exception) {
        std::cout << "In bar" << std::endl;
        throw Exception("Running into some serious exception");
    };
};

int main() {
    foo *a = new foo();
    try {
        a->bar();
    } catch(Exception &obj) {
        obj.printStackTrace();
    }
    return(1);
}
```

Compile the example prog via

- `g++ test.cpp -lexception -o test`

Chapter 2

Todo List

Global `Exception::Exception(std::string msg)` Implementing option to specify maximal stacktracing depth

Global `Exception::Exception(const char *msg)` Implementing option to specify maximal stacktracing depth

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

| | |
|--|---|
| Exception (Exception class implementing a stacktrace function) | 9 |
|--|---|

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

| | |
|---|----|
| Exception | 13 |
| Exception.cpp | 14 |

Chapter 5

Data Structure Documentation

5.1 Exception Class Reference

[Exception](#) class implementing a stacktrace function.

```
#include <Exception>
```

Inherits `std::exception`.

Public Member Functions

- [Exception](#) (std::string msg)
Constructor awaiting your error message as a std::string.
- [Exception](#) (const char *msg)
Constructor specially for using standard exception handling like `bad_alloc` or similar.
- virtual [~Exception](#) () throw ()
Virtual standard destructor who must be disallowed to throw exceptions.
- virtual std::string [getError](#) (void) const throw ()
Returning the last error message and itself is disallowed to throw a exception.
- virtual const char * [what](#) (void) const throw ()
Polymorphed what function from standard C++ exception class returning the last error message and itself is disallowed to throw a exception.
- void [printStackTrace](#) (void) const throw ()
Printing out a stacktrace and itself is disallowed to throw a exception.

5.1.1 Detailed Description

[Exception](#) class implementing a stacktrace function.

Author:

Markus Mazurczak

Version:

1.0

Date:

2010

Definition at line 80 of file Exception.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Exception::Exception (std::string *msg*)

Constructor awaiting your error message as a std::string.

Parameters:*msg* Error description**Todo**

Implementing option to specify maximal stacktracing depth

Definition at line 3 of file Exception.cpp.

```
3                                     {
4     _msg = msg;
5     _traced = backtrace(_stacktrace, 64);
6 }
```

5.1.2.2 Exception::Exception (const char * *msg*)

Constructor specially for using standard exception handling like bad_alloc or similiar.

Parameters:**msg* Char pointer pointing to the error description**Todo**

Implementing option to specify maximal stacktracing depth

Definition at line 8 of file Exception.cpp.

```
8                                     {
9     _msg = msg;
10    _traced = backtrace(_stacktrace, 64);
11 }
```

5.1.2.3 Exception::~~Exception () throw () [inline, virtual]

Virtual standard destructor who must be disallowed to throw exceptions.

Definition at line 101 of file Exception.

```
101 {};
```

5.1.3 Member Function Documentation**5.1.3.1 std::string Exception::getError (void) const throw ()** [virtual]

Returning the last error message and itself is disallowed to throw a exception.

Returns:

std::string

Definition at line 101 of file Exception.

5.1.3.2 void Exception::printStackTrace (void) const throw ()

Printing out a stacktrace and itself is disallowed to throw a exception.

Definition at line 21 of file Exception.cpp.

```
21                                     {
22     Dl_info dlinfo;
23     const char *symname;
24     char *demangled;
25     int status;
26
27     if(!_traced) {
28         std::cerr << "Error: No stacktrace available" << std::endl;
29     } else {
30         for(int i = 0; i < (_traced - 1); i++) {
31             if(!dladdr(_stacktrace[i], &dlinfo)) {
32                 continue;
33             }
34             symname = dlinfo.dli_sname;
35             demangled = abi::__cxa_demangle(symname, NULL, 0, &status);
36             if(status == 0 && demangled) {
37                 symname = demangled;
38             }
39             std::cerr << dlinfo.dli_fname << " -- " << symname << std::endl;
40             if(demangled) {
41                 free(demangled);
42             }
43         }
44     }
45 }
```

5.1.3.3 const char * Exception::what (void) const throw () [virtual]

Polymorphed what function from standard C++ exception class returning the last error message and itself is disallowed to throw a exception.

Returns:

const char *

Definition at line 17 of file Exception.cpp.

```
17                                     {  
18     return(_msg.c_str());  
19 }
```

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

- [Exception](#)
- [Exception.cpp](#)

Chapter 6

File Documentation

6.1 Exception File Reference

```
#include <iostream>
#include <string>
#include <exception>
#include <execinfo.h>
#include <dlfcn.h>
#include <cxxabi.h>
#include <stdlib.h>
```

Data Structures

- class [Exception](#)
Exception class implementing a stacktrace function.

6.2 Exception.cpp File Reference

```
#include "Exception"
```

Index

- ~Exception
 - Exception, [10](#)
- Exception, [9](#), [13](#)
 - ~Exception, [10](#)
 - Exception, [10](#)
 - getError, [11](#)
 - printStackTrace, [11](#)
 - what, [11](#)
- Exception.cpp, [14](#)
- getError
 - Exception, [11](#)
- printStackTrace
 - Exception, [11](#)
- what
 - Exception, [11](#)