

My Project

Generated by Doxygen 1.9.8

1 Curling: Modern C++ libcurl Wrapper	1
1.1 Features	1
1.2 Example	1
2 Namespace Index	3
2.1 Namespace List	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 File Index	9
5.1 File List	9
6 Namespace Documentation	11
6.1 curling Namespace Reference	11
6.1.1 Detailed Description	12
6.2 curling::detail Namespace Reference	12
6.2.1 Detailed Description	12
7 Class Documentation	13
7.1 curling::CurlHandleDeleter Struct Reference	13
7.1.1 Detailed Description	13
7.2 curling::CurlingException Class Reference	13
7.2.1 Detailed Description	14
7.3 curling::CurlMimeDeleter Struct Reference	14
7.4 curling::CurlSlistDeleter Struct Reference	14
7.5 curling::HeaderException Class Reference	15
7.5.1 Detailed Description	16
7.6 curling::InitializationException Class Reference	16
7.6.1 Detailed Description	17
7.7 curling::LogicException Class Reference	17
7.7.1 Detailed Description	18
7.8 curling::MimeException Class Reference	18
7.8.1 Detailed Description	19
7.9 curling::Request Class Reference	20
7.9.1 Detailed Description	21
7.9.2 Member Enumeration Documentation	22
7.9.2.1 HttpVersion	22
7.9.2.2 Method	22
7.9.3 Constructor & Destructor Documentation	22
7.9.3.1 Request()	22

7.9.4 Member Function Documentation	23
7.9.4.1 addArg()	23
7.9.4.2 addFormField()	23
7.9.4.3 addFormFile()	24
7.9.4.4 addHeader()	24
7.9.4.5 downloadToFile()	24
7.9.4.6 enableVerbose()	25
7.9.4.7 send()	25
7.9.4.8 setAuthToken()	25
7.9.4.9 setBody()	26
7.9.4.10 setConnectTimeout()	26
7.9.4.11 setCookiePath()	26
7.9.4.12 setFollowRedirects()	27
7.9.4.13 setHttpAuth()	27
7.9.4.14 setHttpAuthMethod()	27
7.9.4.15 setMethod()	28
7.9.4.16 setProgressCallback()	28
7.9.4.17 setProxy()	28
7.9.4.18 setProxyAuth()	29
7.9.4.19 setProxyAuthMethod()	29
7.9.4.20 setRawOption()	29
7.9.4.21 setTimeout()	30
7.9.4.22 setURL()	30
7.9.4.23 setUserAgent()	30
7.10 curling::RequestException Class Reference	31
7.10.1 Detailed Description	32
7.11 curling::Response Struct Reference	32
7.11.1 Detailed Description	32
8 File Documentation	33
8.1 curling.hpp	33
Index	37

Chapter 1

Curling: Modern C++ libcurl Wrapper

Curling is a lightweight, header-only C++17 wrapper around libcurl for making HTTP requests with a modern design and safe resource handling.

1.1 Features

- RAII and smart-pointer-based resource management
- MIME support for file uploads
- Fluent API for intuitive chaining
- Proxy and authentication support
- Persistent cookie management

1.2 Example

```
curling::Request req;  
req.setMethod(curling::Request::Method::POST)  
  .setURL("https://example.com")  
  .addHeader("Content-Type: application/json")  
  .setBody(R"({"key": "value"})");  
curling::Response res = req.send();  
std::cout << res.toString();
```


Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

curling	11
curling::detail	
Util/helper section	12

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

curling::CurlHandleDeleter	13
curling::CurlMimeDeleter	14
curling::CurlSlistDeleter	14
curling::Request	20
curling::Response	32
std::runtime_error	
curling::CurlingException	13
curling::HeaderException	15
curling::InitializationException	16
curling::LogicException	17
curling::MimeException	18
curling::RequestException	31

Chapter 4

Class Index

4.1 Class List

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

curling::CurlHandleDeleter	13
curling::CurlingException	
Base exception class for Curling errors	13
curling::CurlMimeDeleter	14
curling::CurlSlistDeleter	14
curling::HeaderException	
Thrown when header operations fail	15
curling::InitializationException	
Thrown when curl initialization fails	16
curling::LogicException	
Thrown when library logic prohibits an operation	17
curling::MimeException	
Thrown when MIME operations fail	18
curling::Request	
Provides a fluent wrapper for HTTP requests via libcurl	20
curling::RequestException	
Thrown when a request operation fails	31
curling::Response	
Represents an HTTP response	32

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

include/ curling.hpp	33
--	----

Chapter 6

Namespace Documentation

6.1 curling Namespace Reference

Namespaces

- namespace [detail](#)
Util/helper section.

Classes

- struct [CurlHandleDeleter](#)
- class [CurlingException](#)
Base exception class for Curling errors.
- struct [CurlMimeDeleter](#)
- struct [CurlSlistDeleter](#)
- class [HeaderException](#)
Thrown when header operations fail.
- class [InitializationException](#)
Thrown when curl initialization fails.
- class [LogicException](#)
Thrown when library logic prohibits an operation.
- class [MimeException](#)
Thrown when MIME operations fail.
- class [Request](#)
Provides a fluent wrapper for HTTP requests via libcurl.
- class [RequestException](#)
Thrown when a request operation fails.
- struct [Response](#)
Represents an HTTP response.

Typedefs

- using **CurlIPtr** = std::unique_ptr< CURL, [CurlHandleDeleter](#) >
- using **CurlSlistPtr** = std::unique_ptr< curl_slist, [CurlSlistDeleter](#) >
- using **CurlMimePtr** = std::unique_ptr< curl_mime, [CurlMimeDeleter](#) >

Functions

- `std::string version ()`

Variables

- `constexpr int version_major = 1`
- `constexpr int version_minor = 1`
- `constexpr int version_patch = 0`

6.1.1 Detailed Description

Note

If you use `Method::MIME` (multipart POST), you must reset the [Request](#) before switching to another method. Attempting to change it afterward throws `logic_error`.

Curling internally manages `curl_global_init()` and `curl_global_cleanup()` using `std::once_flag`. You don't need to do this manually.

Header keys in [Response::headers](#) are stored in lowercase to support case-insensitive lookup.

By default, cookies are persisted in "cookies.txt". Override this via `setCookiePath()`.

Calling `enableVerbose(true)` enables libcurl's verbose output to `stderr`, which is useful for debugging.

Warning

This class is not thread-safe. Do not share a [Request](#) instance across threads. Each thread should use its own [Request](#) object.

6.2 curling::detail Namespace Reference

Util/helper section.

Functions

- `void trim (std::string &s)`
- `void toLowerCase (std::string &s)`
- `int ProgressCallbackBridge (void *clientp, curl_off_t dltotal, curl_off_t dlnow, curl_off_t ultotal, curl_off_t ulnow)`

6.2.1 Detailed Description

Util/helper section.

Note

meant for internal library use only

Chapter 7

Class Documentation

7.1 `curling::CurlHandleDeleter` Struct Reference

```
#include <curling.hpp>
```

Public Member Functions

- void **operator()** (CURL *h) const noexcept

7.1.1 Detailed Description

SAFETY: RAII deleters for CURL handles

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

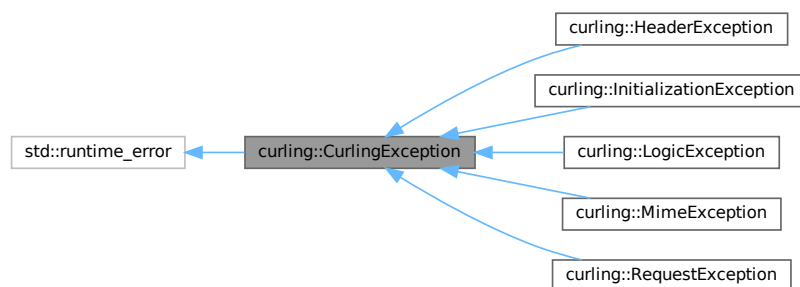
- include/curling.hpp

7.2 `curling::CurlingException` Class Reference

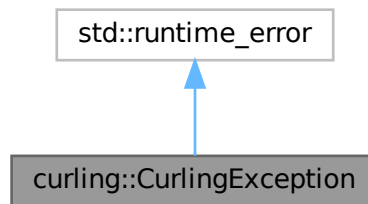
Base exception class for Curling errors.

```
#include <curling.hpp>
```

Inheritance diagram for `curling::CurlingException`:



Collaboration diagram for `curling::CurlingException`:



Public Member Functions

- **CurlingException** (const std::string &msg)

7.2.1 Detailed Description

Base exception class for Curling errors.

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

- include/curling.hpp

7.3 curling::CurlMimeDeleter Struct Reference

Public Member Functions

- void **operator()** (curl_mime *m) const noexcept

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

- include/curling.hpp

7.4 curling::CurlSlistDeleter Struct Reference

Public Member Functions

- void **operator()** (curl_slist *l) const noexcept

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

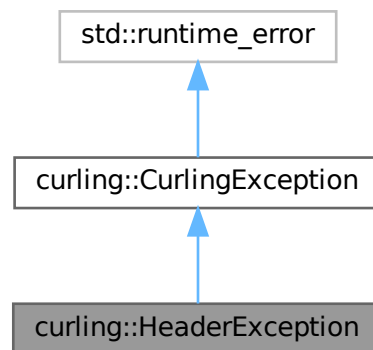
- include/curling.hpp

7.5 curling::HeaderException Class Reference

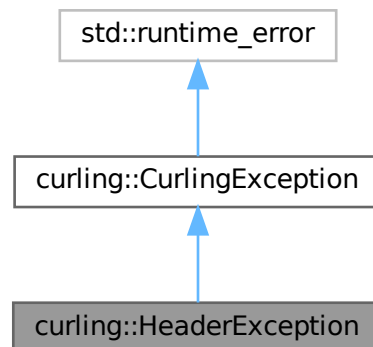
Thrown when header operations fail.

```
#include <curling.hpp>
```

Inheritance diagram for curling::HeaderException:



Collaboration diagram for curling::HeaderException:



Public Member Functions

- **HeaderException** (const std::string &msg)

Public Member Functions inherited from [curling::CurlingException](#)

- **CurlingException** (const std::string &msg)

7.5.1 Detailed Description

Thrown when header operations fail.

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

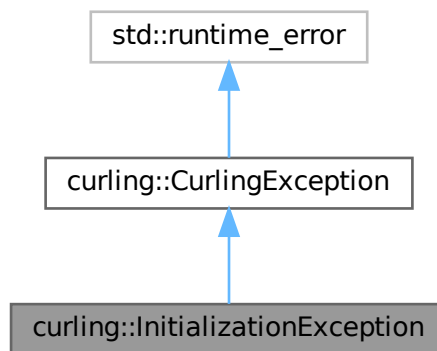
- include/curling.hpp

7.6 curling::InitializationException Class Reference

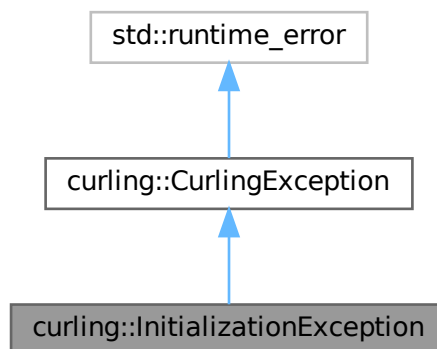
Thrown when curl initialization fails.

```
#include <curling.hpp>
```

Inheritance diagram for curling::InitializationException:



Collaboration diagram for curling::InitializationException:



Public Member Functions

- **InitializationException** (const std::string &msg)

Public Member Functions inherited from [curling::CurlingException](#)

- **CurlingException** (const std::string &msg)

7.6.1 Detailed Description

Thrown when curl initialization fails.

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

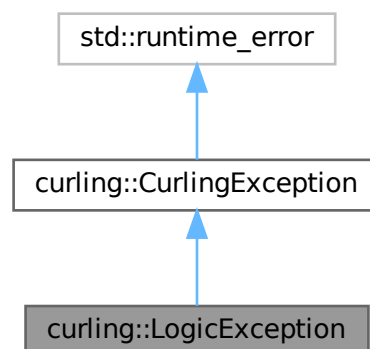
- include/curling.hpp

7.7 curling::LogicException Class Reference

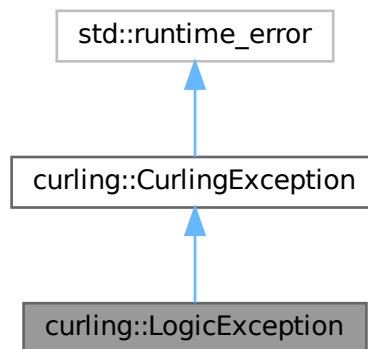
Thrown when library logic prohibits an operation.

```
#include <curling.hpp>
```

Inheritance diagram for curling::LogicException:



Collaboration diagram for `curling::LogicException`:



Public Member Functions

- **LogicException** (const std::string &msg)

Public Member Functions inherited from [curling::CurlingException](#)

- **CurlingException** (const std::string &msg)

7.7.1 Detailed Description

Thrown when library logic prohibits an operation.

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

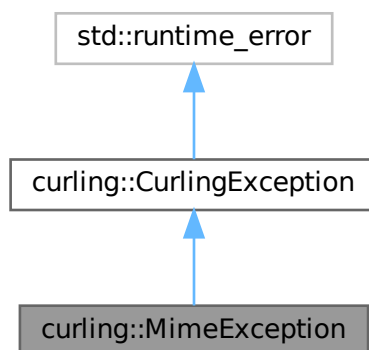
- `include/curling.hpp`

7.8 `curling::MimeException` Class Reference

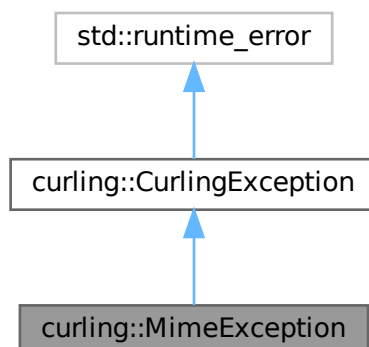
Thrown when MIME operations fail.

```
#include <curling.hpp>
```

Inheritance diagram for curling::MimeException:



Collaboration diagram for curling::MimeException:



Public Member Functions

- **MimeException** (const std::string &msg)

Public Member Functions inherited from [curling::CurlingException](#)

- **CurlingException** (const std::string &msg)

7.8.1 Detailed Description

Thrown when MIME operations fail.

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

- include/curling.hpp

7.9 curling::Request Class Reference

Provides a fluent wrapper for HTTP requests via libcurl.

```
#include <curling.hpp>
```

Public Types

- enum class [Method](#) {
[GET](#) , [POST](#) , [PUT](#) , [DEL](#) ,
[PATCH](#) , [HEAD](#) , [MIME](#) }
Supported HTTP methods.
- enum class [AuthMethod](#) { **BASIC** = CURLAUTH_BASIC , **NTLM** = CURLAUTH_NTLM , **DIGEST** = CURLAUTH_DIGEST }
HTTP authentication schemes.
- enum class [HttpVersion](#) { [DEFAULT](#) , [HTTP_1_1](#) , [HTTP_2](#) , [HTTP_3](#) }
Specifies the HTTP version to be used for the request.
- using **ProgressCallback** = std::function< bool(curl_off_t dltotal, curl_off_t dlnow, curl_off_t ultotal, curl_off_t ulnow)>

Public Member Functions

- [Request](#) ()
Constructor initializes curl global state.
- [~Request](#) () noexcept
Destructor cleans up curl state if last instance.
- **Request** ([Request](#) &&) noexcept
- [Request](#) & **operator=** ([Request](#) &&) noexcept
- **Request** (const [Request](#) &)=delete
- [Request](#) & **operator=** (const [Request](#) &)=delete
- [Request](#) & **setProgressCallback** (ProgressCallback cb)
Sets the progress callback function.
- [Request](#) & **setMethod** ([Method](#) m)
Sets the HTTP method for the request.
- [Request](#) & **setURL** (const std::string &url)
Sets the request URL.
- [Request](#) & **setProxy** (const std::string &url)
Enables proxy usage.
- [Request](#) & **setProxyAuth** (const std::string &username, const std::string &password)
Sets credentials for proxy authentication.
- [Request](#) & **setProxyAuthMethod** ([AuthMethod](#) method)
Sets proxy authentication scheme.
- [Request](#) & **setHttpAuth** (const std::string &username, const std::string &password)
Sets credentials for HTTP auth (Basic/Digest/NTLM).
- [Request](#) & **setHttpAuthMethod** ([AuthMethod](#) method)
Sets HTTP authentication scheme.
- [Request](#) & **addArg** (const std::string &key, const std::string &value)
Adds a query parameter to the URL.
- [Request](#) & **addHeader** (const std::string &header)
Adds a custom HTTP header.

- [Request](#) & [setBody](#) (const std::string &body)
Sets the body of the request (for POST/PUT/PATCH).
- [Request](#) & [downloadToFile](#) (const std::string &path)
Enables download streaming to a file.
- [Request](#) & [setTimeout](#) (long seconds)
Sets a timeout for the request (in seconds).
- [Request](#) & [setConnectTimeout](#) (long seconds)
Sets connection timeout (in seconds).
- [Request](#) & [setFollowRedirects](#) (bool follow)
Enables or disables automatic redirect-following.
- [Request](#) & [setAuthToken](#) (const std::string &token)
Adds a Bearer token for Authorization header.
- [Request](#) & [setCookiePath](#) (const std::string &path)
Overrides default cookie file for persistence.
- [Request](#) & [setUserAgent](#) (const std::string &userAgent)
Sets the User-Agent header.
- [Request](#) & [addFormField](#) (const std::string &fieldName, const std::string &value)
Adds a field to multipart/form-data.
- [Request](#) & [addFormFile](#) (const std::string &fieldName, const std::string &filePath)
Adds a file to multipart upload.
- [Request](#) & [enableVerbose](#) (bool enabled=true)
Enables or disables libcurl verbose output.
- [Response](#) [send](#) ()
Executes the HTTP request.
- void [reset](#) ()
Resets internal state to allow reuse.
- [Request](#) & [setHttpVersion](#) ([HttpVersion](#) version)
Set the HTTP protocol version (http1.1, 2 or 3)
- template<typename T >
 [Request](#) & [setRawOption](#) (CURLOption opt, T value)
Low level access to define curl options.

Friends

- int [detail::ProgressCallbackBridge](#) (void *clientp, curl_off_t dltotal, curl_off_t dlnow, curl_off_t ultotal, curl_off_t ulnow)

7.9.1 Detailed Description

Provides a fluent wrapper for HTTP requests via libcurl.

7.9.2 Member Enumeration Documentation

7.9.2.1 HttpVersion

```
enum class curling::Request::HttpVersion [strong]
```

Specifies the HTTP version to be used for the request.

Allows explicit control over the HTTP version that libcurl should use when performing a request. If the selected version is not supported by the libcurl build or the server, fallback behavior may occur.

Note

If you request a version not supported by libcurl at runtime, Curling will throw a [LogicException](#) during configuration. If the server doesn't support the version (e.g., HTTP/2), libcurl may fall back to an older version without error.

Enumerator

DEFAULT	Let libcurl automatically negotiate the best supported HTTP version.
HTTP_1↔ _1	Force HTTP/1.1 for all requests.
HTTP_2	Force HTTP/2 (requires libcurl built with nghttp2 support).
HTTP_3	Force HTTP/3 (requires libcurl built with HTTP/3 support, e.g. with quiche or ngtcp2).

7.9.2.2 Method

```
enum class curling::Request::Method [strong]
```

Supported HTTP methods.

Enumerator

GET	Standard GET.
POST	Standard POST.
PUT	PUT.
DEL	DELETE (named DEL to avoid macro clash)
PATCH	PATCH.
HEAD	HEAD request just headers.
MIME	Multipart/form-data POST.

7.9.3 Constructor & Destructor Documentation

7.9.3.1 Request()

```
curling::Request::Request ( )
```

Constructor initializes curl global state.

Exceptions

<i>InitializationException</i>	if initialization fails.
--	--------------------------

7.9.4 Member Function Documentation

7.9.4.1 addArg()

```
Request & curling::Request::addArg (
    const std::string & key,
    const std::string & value )
```

Adds a query parameter to the URL.

Parameters

<i>key</i>	Parameter name.
<i>value</i>	Parameter value.

Returns

*this

7.9.4.2 addFormField()

```
Request & curling::Request::addFormField (
    const std::string & fieldName,
    const std::string & value )
```

Adds a field to multipart/form-data.

Parameters

<i>fieldName</i>	Field name.
<i>value</i>	Field value.

Returns

*this

Exceptions

<i>MimeException</i>	on internal curl errors.
--------------------------------------	--------------------------

7.9.4.3 addFormFile()

```
Request & curling::Request::addFormFile (
    const std::string & fieldName,
    const std::string & filePath )
```

Adds a file to multipart upload.

Parameters

<i>fieldName</i>	Field name.
<i>filePath</i>	Path to file on disk.

Returns

*this

Exceptions

<i>MimeException</i>	on internal curl errors.
--------------------------------------	--------------------------

7.9.4.4 addHeader()

```
Request & curling::Request::addHeader (
    const std::string & header )
```

Adds a custom HTTP header.

Parameters

<i>header</i>	A full header line, e.g. "Accept: application/json".
---------------	--

Returns

*this

7.9.4.5 downloadToFile()

```
Request & curling::Request::downloadToFile (
    const std::string & path )
```

Enables download streaming to a file.

Parameters

<i>path</i>	Local file path for saving response.
-------------	--------------------------------------

Returns

*this

7.9.4.6 enableVerbose()

```
Request & curling::Request::enableVerbose (
    bool enabled = true )
```

Enables or disables libcurl verbose output.

Parameters

<i>enabled</i>	True to enable verbose mode.
----------------	------------------------------

Returns

*this

7.9.4.7 send()

```
Response curling::Request::send ( )
```

Executes the HTTP request.

Returns

[Response](#) object with status, body, headers.

Exceptions

RequestException	on failure.
----------------------------------	-------------

7.9.4.8 setAuthToken()

```
Request & curling::Request::setAuthToken (
    const std::string & token )
```

Adds a Bearer token for Authorization header.

Parameters

<i>token</i>	Bearer token string.
--------------	----------------------

Returns

*this

7.9.4.9 setBody()

```
Request & curling::Request::setBody (
    const std::string & body )
```

Sets the body of the request (for POST/PUT/PATCH).

Parameters

<i>body</i>	Request body content.
-------------	-----------------------

Returns

*this

7.9.4.10 setConnectTimeout()

```
Request & curling::Request::setConnectTimeout (
    long seconds )
```

Sets connection timeout (in seconds).

Parameters

<i>seconds</i>	Timeout in seconds.
----------------	---------------------

Returns

*this

7.9.4.11 setCookiePath()

```
Request & curling::Request::setCookiePath (
    const std::string & path )
```

Overrides default cookie file for persistence.

Parameters

<i>path</i>	File path for storing cookies.
-------------	--------------------------------

Returns

*this

7.9.4.12 setFollowRedirects()

```
Request & curling::Request::setFollowRedirects (
    bool follow )
```

Enables or disables automatic redirect-following.

Parameters

<i>follow</i>	True to follow redirects.
---------------	---------------------------

Returns

*this

7.9.4.13 setHttpAuth()

```
Request & curling::Request::setHttpAuth (
    const std::string & username,
    const std::string & password )
```

Sets credentials for HTTP auth (Basic/Digest/NTLM).

Parameters

<i>username</i>	Username.
<i>password</i>	Password.

Returns

*this

7.9.4.14 setHttpAuthMethod()

```
Request & curling::Request::setHttpAuthMethod (
    AuthMethod method )
```

Sets HTTP authentication scheme.

Parameters

<i>method</i>	Authentication method.
---------------	------------------------

Returns

*this

7.9.4.15 setMethod()

```
Request & curling::Request::setMethod (
    Method m )
```

Sets the HTTP method for the request.

Parameters

<i>m</i>	Enum value for HTTP method.
----------	-----------------------------

Returns

*this

7.9.4.16 setProgressCallback()

```
Request & curling::Request::setProgressCallback (
    ProgressCallback cb )
```

Sets the progress callback function.

Parameters

<i>cb</i>	Callback receiving download/upload progress. Return true to abort.
-----------	--

Returns

*this

7.9.4.17 setProxy()

```
Request & curling::Request::setProxy (
    const std::string & url )
```

Enables proxy usage.

Parameters

<i>url</i>	Proxy URL.
------------	------------

Returns

*this

7.9.4.18 setProxyAuth()

```
Request & curling::Request::setProxyAuth (
    const std::string & username,
    const std::string & password )
```

Sets credentials for proxy authentication.

Parameters

<i>username</i>	Proxy username.
<i>password</i>	Proxy password.

Returns

*this

7.9.4.19 setProxyAuthMethod()

```
Request & curling::Request::setProxyAuthMethod (
    AuthMethod method )
```

Sets proxy authentication scheme.

Parameters

<i>method</i>	Authentication method.
---------------	------------------------

Returns

*this

7.9.4.20 setRawOption()

```
template<typename T >
Request & curling::Request::setRawOption (
    CURLOPToption opt,
    T value ) [inline]
```

Low level access to define curl options.

Warning

Should be used with caution. Meant for the libcurl advanced users.

7.9.4.21 setTimeout()

```
Request & curling::Request::setTimeout (
    long seconds )
```

Sets a timeout for the request (in seconds).

Parameters

<i>seconds</i>	Timeout in seconds.
----------------	---------------------

Returns

*this

7.9.4.22 setURL()

```
Request & curling::Request::setURL (
    const std::string & url )
```

Sets the request URL.

Parameters

<i>url</i>	URL to fetch.
------------	---------------

Returns

*this

7.9.4.23 setUserAgent()

```
Request & curling::Request::setUserAgent (
    const std::string & userAgent )
```

Sets the User-Agent header.

Parameters

<i>userAgent</i>	Agent string.
------------------	---------------

Returns

*this

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

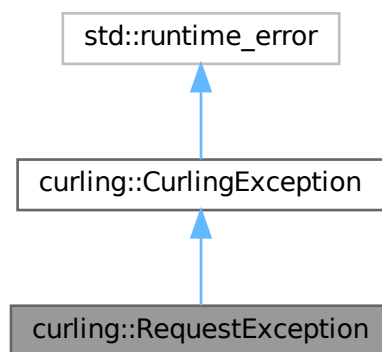
- include/curling.hpp

7.10 curling::RequestException Class Reference

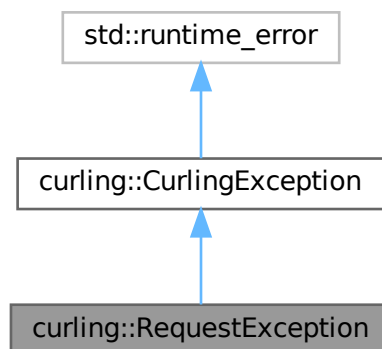
Thrown when a request operation fails.

```
#include <curling.hpp>
```

Inheritance diagram for curling::RequestException:



Collaboration diagram for curling::RequestException:



Public Member Functions

- **RequestException** (const std::string &msg)

Public Member Functions inherited from [curling::CurlingException](#)

- **CurlingException** (const std::string &msg)

7.10.1 Detailed Description

Thrown when a request operation fails.

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

- include/curling.hpp

7.11 curling::Response Struct Reference

Represents an HTTP response.

```
#include <curling.hpp>
```

Public Member Functions

- std::string **toString** () const
- std::vector< std::string > **getHeader** (const std::string &key) const

Public Attributes

- long **httpCode**
HTTP status code.
- std::string **body**
[Response](#) body.
- std::map< std::string, std::vector< std::string > > **headers**
Header map (key: lowercase).

7.11.1 Detailed Description

Represents an HTTP response.

Contains the HTTP status code, body, and headers.

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

- include/curling.hpp

Chapter 8

File Documentation

8.1 curling.hpp

```
00001 /*
00002  * Copyright (c) 2025 Paul Caron
00003  *
00004  * This file is part of Curling - a modern C++ wrapper for libcurl.
00005  *
00006  * Licensed under the MIT License. You may obtain a copy of the license at
00007  * https://opensource.org/licenses/MIT
00008  */
00009
00010
00066 #pragma once
00067 #include <string>
00068 #include <map>
00069 #include <vector>
00070 #include <sstream>
00071 #include <mutex>
00072 #include <stdexcept>
00073 #include <algorithm>
00074 #include <cctype>
00075 #include <memory>
00076 #include <functional>
00077 #include <curl/curl.h>
00078
00079 namespace curling {
00080
00081 inline constexpr int version_major = 1;
00082 inline constexpr int version_minor = 1;
00083 inline constexpr int version_patch = 0;
00084
00085 std::string version();
00086
00091 namespace detail {
00092
00093 inline void trim(std::string& s) {
00094     s.erase(s.begin(), std::find_if(s.begin(), s.end(), [](unsigned char ch){
00095         return !std::isspace(ch);
00096     }));
00097     s.erase(std::find_if(s.rbegin(), s.rend(), [](unsigned char ch){
00098         return !std::isspace(ch);
00099     }).base(), s.end());
00100 }
00101
00102 inline void toLowerCase(std::string& s) {
00103     std::transform(s.begin(), s.end(), s.begin(),
00104         [](unsigned char c) { return std::tolower(c); });
00105 }
00106
00107 int ProgressCallbackBridge(void* clientp, curl_off_t dltotal, curl_off_t dlnow,
00108     curl_off_t ultotal, curl_off_t ulnow);
00109
00110
00111 } // detail end
00112
00117 class CurlingException : public std::runtime_error {
00118 public:
00119     explicit CurlingException(const std::string& msg) : std::runtime_error(msg) {}
00120 };
00121
```

```

00125 class InitializationException : public CurlingException {
00126 public:
00127     explicit InitializationException(const std::string& msg) : CurlingException(msg) {}
00128 };
00129
00133 class RequestException : public CurlingException {
00134 public:
00135     explicit RequestException(const std::string& msg) : CurlingException(msg) {}
00136 };
00137
00141 class HeaderException : public CurlingException {
00142 public:
00143     explicit HeaderException(const std::string& msg) : CurlingException(msg) {}
00144 };
00145
00149 class MimeException : public CurlingException {
00150 public:
00151     explicit MimeException(const std::string& msg) : CurlingException(msg) {}
00152 };
00153
00157 class LogicException : public CurlingException {
00158 public:
00159     explicit LogicException(const std::string& msg) : CurlingException(msg) {}
00160 };
00161
00163 struct CurlHandleDeleter { void operator()(CURL* h) const noexcept { if (h) curl_easy_cleanup(h); }};
00164 struct CurlSlistDeleter { void operator()(curl_slist* l) const noexcept { if (l)
    curl_slist_free_all(l); }};
00165 struct CurlMimeDeleter { void operator()(curl_mime* m) const noexcept { if (m) curl_mime_free(m); }};
00166
00167 using CurlPtr = std::unique_ptr<CURL, CurlHandleDeleter>;
00168 using CurlSlistPtr = std::unique_ptr<curl_slist, CurlSlistDeleter>;
00169 using CurlMimePtr = std::unique_ptr<curl_mime, CurlMimeDeleter>;
00170
00177 struct Response {
00178     long httpCode;
00179     std::string body;
00180     std::map<std::string, std::vector<std::string>> headers;
00181
00182     std::string toString() const {
00183         std::ostringstream oss;
00184         oss << "status: " << httpCode << "\nbody:\n" << body << "\nheaders:\n";
00185         for (auto const& h : headers) {
00186             oss << h.first << ": ";
00187             for (auto const& v : h.second) oss << v << " ";
00188             oss << "\n";
00189         }
00190         return oss.str();
00191     }
00192     std::vector<std::string> getHeader(const std::string& key) const {
00193         std::string lowered = key;
00194         detail::toLowerCase(lowered);
00195         auto it = headers.find(lowered);
00196         return (it != headers.end()) ? it->second : std::vector<std::string>{};
00197     }
00198 };
00199
00204 class Request {
00205 public:
00206     using ProgressCallback = std::function<bool(curl_off_t dltotal, curl_off_t dlnow,
00207         curl_off_t ultotal, curl_off_t ulnow)>;
00208
00213     enum class Method {
00214         GET,
00215         POST,
00216         PUT,
00217         DEL,
00218         PATCH,
00219         HEAD,
00220         MIME
00221     };
00222
00227     enum class AuthMethod {
00228         BASIC = CURLAUTH_BASIC,
00229         NTLM = CURLAUTH_NTLM,
00230         DIGEST = CURLAUTH_DIGEST
00231     };
00232
00246     enum class HttpVersion {
00247         DEFAULT,
00248         HTTP_1_1,
00249         HTTP_2,
00250         HTTP_3
00251     };
00252
00257     Request();
00258

```

```

00262     ~Request() noexcept;
00263
00264     Request(Request&&) noexcept;
00265     Request& operator=(Request&&) noexcept;
00266
00267     Request(const Request&) = delete;
00268     Request& operator=(const Request&) = delete;
00269
00275     Request& setProgressCallback(ProgressCallback cb);
00276
00282     Request& setMethod(Method m);
00283
00289     Request& setURL(const std::string& url);
00290
00296     Request& setProxy(const std::string& url);
00297
00304     Request& setProxyAuth(const std::string& username, const std::string& password);
00305
00311     Request& setProxyAuthMethod(AuthMethod method);
00312
00319     Request& setHttpAuth(const std::string& username, const std::string& password);
00320
00326     Request& setHttpAuthMethod(AuthMethod method);
00327
00334     Request& addArg(const std::string& key, const std::string& value);
00335
00341     Request& addHeader(const std::string& header);
00342
00348     Request& setBody(const std::string& body);
00349
00355     Request& downloadToFile(const std::string& path);
00356
00362     Request& setTimeout(long seconds);
00363
00369     Request& setConnectTimeout(long seconds);
00370
00376     Request& setFollowRedirects(bool follow);
00377
00383     Request& setAuthToken(const std::string& token);
00384
00390     Request& setCookiePath(const std::string& path);
00391
00397     Request& setUserAgent(const std::string& userAgent);
00398
00406     Request& addFormField(const std::string& fieldName, const std::string& value);
00407
00415     Request& addFormFile(const std::string& fieldName, const std::string& filePath);
00416
00422     Request& enableVerbose(bool enabled = true);
00423
00429     Response send();
00430
00434     void reset();
00435
00439     Request& setHttpVersion(HttpVersion version);
00440
00446     template<typename T>
00447     Request& setRawOption(CURLOption opt, T value) {
00448         static_assert(std::is_pointer<T>::value || std::is_arithmetic<T>::value,
00449             "setRawOption only supports pointer or arithmetic types");
00450         curl_easy_setopt(curlHandle.get(), opt, value);
00451         return *this;
00452     }
00453
00454     friend int detail::ProgressCallbackBridge(void* clientp, curl_off_t dltotal, curl_off_t dlnow,
00455         curl_off_t ultotal, curl_off_t ulnow);
00456
00457 private:
00458     Method method;
00459     CurlPtr curlHandle;
00460     CurlSlistPtr list; //headers;
00461     std::string url, args, body, cookieFile, cookieJar;
00462     CurlMimePtr mime;
00463     std::string downloadFilePath;
00464     ProgressCallback progressCallback;
00465     HttpVersion httpVersion = HttpVersion::DEFAULT;
00466
00467     void clean() noexcept;
00468     void updateURL();
00469 };
00470
00471 static_assert(!std::is_copy_constructible_v<Request> && !std::is_copy_assignable_v<Request>,
00472     "curling::Request is not copyable: it is thread-unsafe and must not be shared between
00473     threads. One instance per thread.");
00474
00475 } // namespace curling

```

00476

Index

- addArg
 - [curling::Request, 23](#)
- addFormField
 - [curling::Request, 23](#)
- addFormFile
 - [curling::Request, 23](#)
- addHeader
 - [curling::Request, 24](#)
- curling, [11](#)
- Curling: Modern C++ libcurl Wrapper, [1](#)
- [curling::CurlHandleDeleter, 13](#)
- [curling::CurlingException, 13](#)
- [curling::CurlMimeDeleter, 14](#)
- [curling::CurlSlistDeleter, 14](#)
- [curling::detail, 12](#)
- [curling::HeaderException, 15](#)
- [curling::InitializationException, 16](#)
- [curling::LogicException, 17](#)
- [curling::MimeException, 18](#)
- [curling::Request, 20](#)
 - [addArg, 23](#)
 - [addFormField, 23](#)
 - [addFormFile, 23](#)
 - [addHeader, 24](#)
 - [DEFAULT, 22](#)
 - [DEL, 22](#)
 - [downloadToFile, 24](#)
 - [enableVerbose, 25](#)
 - [GET, 22](#)
 - [HEAD, 22](#)
 - [HTTP_1_1, 22](#)
 - [HTTP_2, 22](#)
 - [HTTP_3, 22](#)
 - [HttpVersion, 22](#)
 - [Method, 22](#)
 - [MIME, 22](#)
 - [PATCH, 22](#)
 - [POST, 22](#)
 - [PUT, 22](#)
 - [Request, 22](#)
 - [send, 25](#)
 - [setAuthToken, 25](#)
 - [setBody, 26](#)
 - [setConnectTimeout, 26](#)
 - [setCookiePath, 26](#)
 - [setFollowRedirects, 27](#)
 - [setHttpAuth, 27](#)
 - [setHttpAuthMethod, 27](#)
 - [setMethod, 28](#)
 - [setProgressCallback, 28](#)
 - [setProxy, 28](#)
 - [setProxyAuth, 29](#)
 - [setProxyAuthMethod, 29](#)
 - [setRawOption, 29](#)
 - [setTimeout, 29](#)
 - [setURL, 30](#)
 - [setUserAgent, 30](#)
- [curling::RequestException, 31](#)
- [curling::Response, 32](#)
- DEFAULT
 - [curling::Request, 22](#)
- DEL
 - [curling::Request, 22](#)
- downloadToFile
 - [curling::Request, 24](#)
- enableVerbose
 - [curling::Request, 25](#)
- GET
 - [curling::Request, 22](#)
- HEAD
 - [curling::Request, 22](#)
- HTTP_1_1
 - [curling::Request, 22](#)
- HTTP_2
 - [curling::Request, 22](#)
- HTTP_3
 - [curling::Request, 22](#)
- HttpVersion
 - [curling::Request, 22](#)
- [include/curling.hpp, 33](#)
- Method
 - [curling::Request, 22](#)
- MIME
 - [curling::Request, 22](#)
- PATCH
 - [curling::Request, 22](#)
- POST
 - [curling::Request, 22](#)
- PUT
 - [curling::Request, 22](#)
- Request
 - [curling::Request, 22](#)

send
 curling::Request, [25](#)
setAuthToken
 curling::Request, [25](#)
setBody
 curling::Request, [26](#)
setConnectTimeout
 curling::Request, [26](#)
setCookiePath
 curling::Request, [26](#)
setFollowRedirects
 curling::Request, [27](#)
setHttpAuth
 curling::Request, [27](#)
setHttpAuthMethod
 curling::Request, [27](#)
setMethod
 curling::Request, [28](#)
setProgressCallback
 curling::Request, [28](#)
setProxy
 curling::Request, [28](#)
setProxyAuth
 curling::Request, [29](#)
setProxyAuthMethod
 curling::Request, [29](#)
setRawOption
 curling::Request, [29](#)
setTimeout
 curling::Request, [29](#)
setURL
 curling::Request, [30](#)
setUserAgent
 curling::Request, [30](#)