My Project

Generated by Doxygen 1.9.8

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:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

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 documented files with brief descriptions:

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 **CurlPtr** = 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.

Warning

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

Note

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.

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:

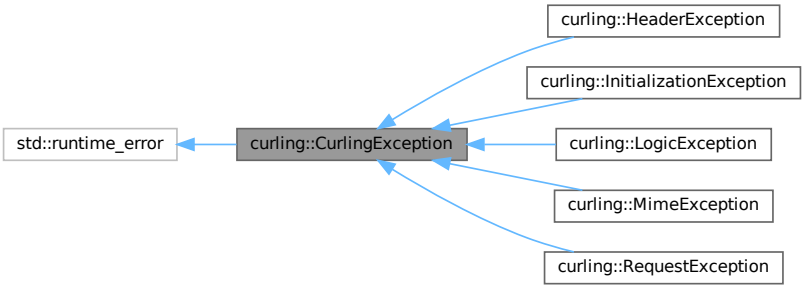
- include/curling.hpp

7.2 curling::CurlingException Class Reference

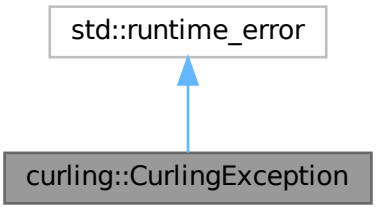
Base exception class for Curling errors.

`#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:

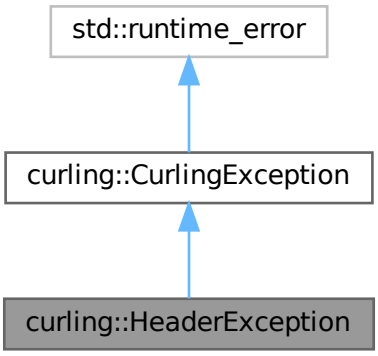
- include/curling.hpp

7.5 curling::HeaderException Class Reference

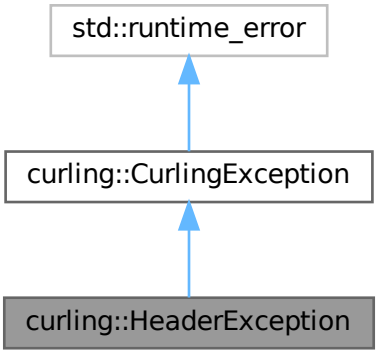
Thrown when header operations fail.

`#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:

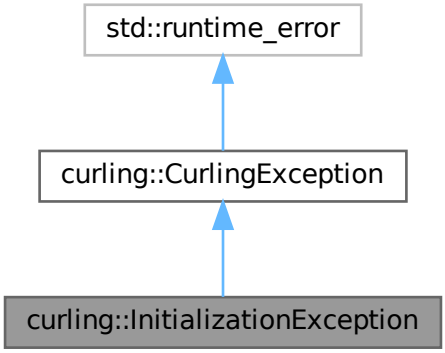
- include/curling.hpp

7.6 curling::InitializationException Class Reference

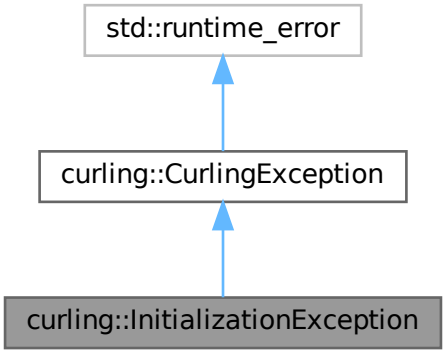
Thrown when curl initialization fails.

`#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:

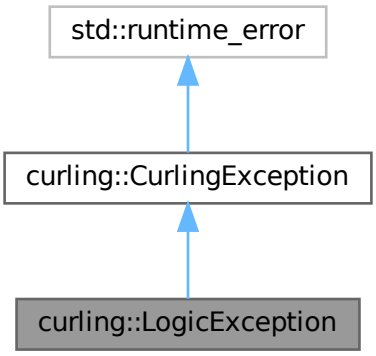
- include/curling.hpp

7.7 curling::LogicException Class Reference

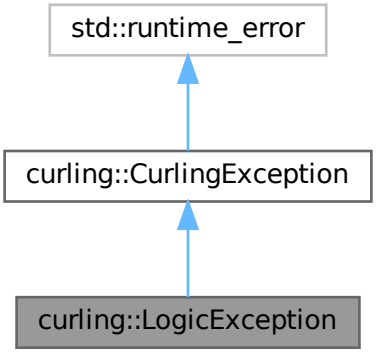
Thrown when library logic prohibits an operation.

`#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:

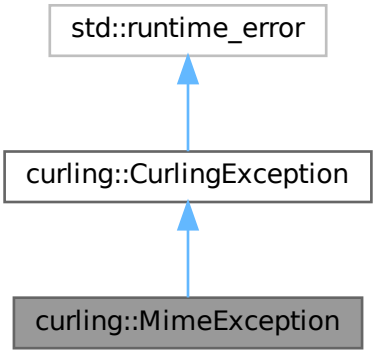
- include/curling.hpp

7.8 curling::MimeException Class Reference

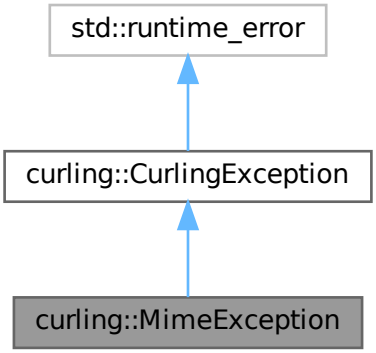
Thrown when MIME operations fail.

`#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)

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

InitializationException	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

key	Parameter name.
value	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

fieldName	Field name.
value	Field value.

Returns

∗this

Exceptions

MimeException	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

fieldName	Field name.
filePath	Path to file on disk.

Returns

∗this

Exceptions

MimeException	on internal curl errors.

7.9.4.4 addHeader()

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

Adds a custom HTTP header.

Parameters

header	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

path	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

enabled	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

token	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

body	Request body content.

Returns

*this

7.9.4.10 setConnectTimeout()

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

Sets connection timeout (in seconds).

Parameters

seconds	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

path	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

follow	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

username	Username.
password	Password.

Returns

*this

7.9.4.14 setHttpAuthMethod()

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

Sets HTTP authentication scheme.

Parameters

method	Authentication method.

Returns

*this

7.9.4.15 setMethod()

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

Sets the HTTP method for the request.

Parameters

m	Enum value for HTTP method.

Returns

∗this

7.9.4.16 setProgressCallback()

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

Sets the progress callback function.

Parameters

cb	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

url	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

username	Proxy username.
password	Proxy password.

Returns

∗this

7.9.4.19 setProxyAuthMethod()

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

Sets proxy authentication scheme.

Parameters

method	Authentication method.

Returns

∗this

7.9.4.20 setTimeout()

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

Sets a timeout for the request (in seconds).

Parameters

seconds	Timeout in seconds.

Returns

∗this

7.9.4.21 setURL()

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

Sets the request URL.

Parameters

url	URL to fetch.

Returns

*this

7.9.4.22 setUserAgent()

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

Sets the User-Agent header.

Parameters

userAgent	Agent string.

Returns

*this

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

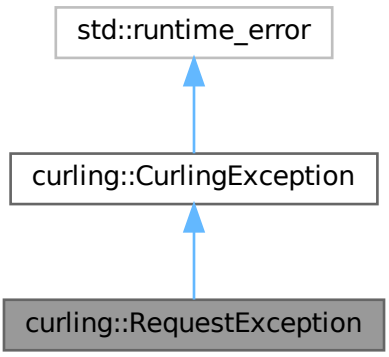
- include/curling.hpp

7.10 curling::RequestException Class Reference

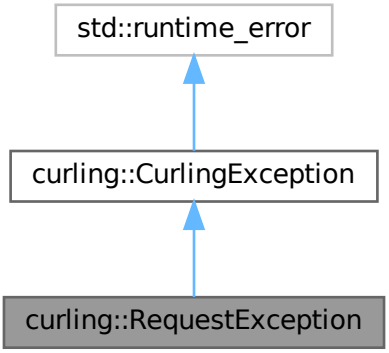
Thrown when a request operation fails.

`#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 #pragma once
00002
00003 /*
00004  * Copyright (c) 2025 Paul Caron
00005  *
00006  * This file is part of Curling - a modern C++ wrapper for libcurl.
00007  *
00008  * Licensed under the MIT License. You may obtain a copy of the license at
00009  * https://opensource.org/licenses/MIT
00010  */
00011
00012
00068 #include <string>
00069 #include <map>
00070 #include <vector>
00071 #include <sstream>
00072 #include <mutex>
00073 #include <stdexcept>
00074 #include <algorithm>
00075 #include <cctype>
00076 #include <memory>
00077 #include <functional>
00078 #include <curl/curl.h>
00079
00080 namespace curling {
00081
00082 inline constexpr int version_major = 1;
00083 inline constexpr int version_minor = 1;
00084 inline constexpr int version_patch = 0;
00085
00086 std::string version();
00087
00092 namespace detail {
00093
00094 inline void trim(std::string& s) {
00095     s.erase(s.begin(), std::find_if(s.begin(), s.end(), [](unsigned char ch){
00096         return !std::isspace(ch);
00097     }));
00098     s.erase(std::find_if(s.rbegin(), s.rend(), [](unsigned char ch){
00099         return !std::isspace(ch);
00100     }).base(), s.end());
00101 }
00102
00103 inline void toLowerCase(std::string& s) {
00104     std::transform(s.begin(), s.end(), s.begin(),
00105                    [](unsigned char c) { return std::tolower(c); });
00106 }
00107
00108 inline int ProgressCallbackBridge(void* clientp, curl_off_t dltotal, curl_off_t dlnow,
00109                                   curl_off_t ultotal, curl_off_t ulnow);
00110
00111
00112 }//detail end
00113
00118 class CurlingException : public std::runtime_error {
00119 public:
00120     explicit CurlingException(const std::string& msg) : std::runtime_error(msg) {}
00121 };
```

```
00122
00126 class InitializationException : public CurlingException {
00127 public:
00128     explicit InitializationException(const std::string& msg) : CurlingException(msg) {}
00129 };
00130
00134 class RequestException : public CurlingException {
00135 public:
00136     explicit RequestException(const std::string& msg) : CurlingException(msg) {}
00137 };
00138
00142 class HeaderException : public CurlingException {
00143 public:
00144     explicit HeaderException(const std::string& msg) : CurlingException(msg) {}
00145 };
00146
00150 class MimeException : public CurlingException {
00151 public:
00152     explicit MimeException(const std::string& msg) : CurlingException(msg) {}
00153 };
00154
00158 class LogicException : public CurlingException {
00159 public:
00160     explicit LogicException(const std::string& msg) : CurlingException(msg) {}
00161 };
00162
00164 struct CurlHandleDeleter { void operator()(CURL* h) const noexcept { if (h) curl_easy_cleanup(h); }};
00165 struct CurlSlistDeleter { void operator()(curl_slist* l) const noexcept { if (l)
      curl_slist_free_all(l); }};
00166 struct CurlMimeDeleter { void operator()(curl_mime* m) const noexcept { if (m) curl_mime_free(m); }};
00167
00168 using CurlPtr = std::unique_ptr<CURL, CurlHandleDeleter>;
00169 using CurlSlistPtr = std::unique_ptr<curl_slist, CurlSlistDeleter>;
00170 using CurlMimePtr = std::unique_ptr<curl_mime, CurlMimeDeleter>;
00171
00178 struct Response {
00179     long httpCode;
00180     std::string body;
00181     std::map<std::string, std::vector<std::string» headers;
00182
00183     std::string toString() const {
00184         std::ostringstream oss;
00185         oss « "status: " « httpCode « "\nbody:\n" « body « "\nheaders:\n";
00186         for (auto const& h : headers) {
00187             oss « h.first « ": ";
00188             for (auto const& v : h.second) oss « v « " ";
00189             oss « "\n";
00190         }
00191         return oss.str();
00192     }
00193     std::vector<std::string> getHeader(const std::string& key) const {
00194         std::string lowered = key;
00195         detail::toLowerCase(lowered);
00196         auto it = headers.find(lowered);
00197         return (it != headers.end()) ? it->second : std::vector<std::string>{};
00198     }
00199 };
00200
00205 class Request {
00206 public:
00207     using ProgressCallback = std::function<bool(curl_off_t dltotal, curl_off_t dlnow,
00208                                                 curl_off_t ultotal, curl_off_t ulnow)>;
00209
00214     enum class Method {
00215         GET,
00216         POST,
00217         PUT,
00218         DEL,
00219         PATCH,
00220         HEAD,
00221         MIME
00222     };
00223
00228     enum class AuthMethod {
00229         BASIC = CURLAUTH_BASIC,
00230         NTLM = CURLAUTH_NTLM,
00231         DIGEST = CURLAUTH_DIGEST
00232     };
00233
00247     enum class HttpVersion {
00248         DEFAULT,
00249         HTTP_1_1,
00250         HTTP_2,
00251         HTTP_3
00252     };
00253
00258     Request();
```

```
00259
00263     ~Request() noexcept;
00264
00265     Request(Request&&) noexcept;
00266     Request& operator=(Request&&) noexcept;
00267
00268     Request(const Request&) = delete;
00269     Request& operator=(const Request&) = delete;
00270
00276     Request& setProgressCallback(ProgressCallback cb);
00277
00283     Request& setMethod(Method m);
00284
00290     Request& setURL(const std::string& url);
00291
00297     Request& setProxy(const std::string& url);
00298
00305     Request& setProxyAuth(const std::string& username, const std::string& password);
00306
00312     Request& setProxyAuthMethod(AuthMethod method);
00313
00320     Request& setHttpAuth(const std::string& username, const std::string& password);
00321
00327     Request& setHttpAuthMethod(AuthMethod method);
00328
00335     Request& addArg(const std::string& key, const std::string& value);
00336
00342     Request& addHeader(const std::string& header);
00343
00349     Request& setBody(const std::string& body);
00350
00356     Request& downloadToFile(const std::string& path);
00357
00363     Request& setTimeout(long seconds);
00364
00370     Request& setConnectTimeout(long seconds);
00371
00377     Request& setFollowRedirects(bool follow);
00378
00384     Request& setAuthToken(const std::string& token);
00385
00391     Request& setCookiePath(const std::string& path);
00392
00398     Request& setUserAgent(const std::string& userAgent);
00399
00407     Request& addFormField(const std::string& fieldName, const std::string& value);
00408
00416     Request& addFormFile(const std::string& fieldName, const std::string& filePath);
00417
00423     Request& enableVerbose(bool enabled = true);
00424
00430     Response send();
00431
00435     void reset();
00436
00437     Request& setHttpVersion(HttpVersion version);
00438
00439     friend int detail::ProgressCallbackBridge(void* clientp, curl_off_t dltotal, curl_off_t dlnow,
00440                                               curl_off_t ultotal, curl_off_t ulnow);
00441
00442
00443 private:
00444     Method method;
00445     CurlPtr curlHandle;
00446     CurlSlistPtr list;//headers;
00447     std::string url, args, body, cookieFile, cookieJar;
00448     CurlMimePtr mime;
00449     std::string downloadFilePath;
00450     ProgressCallback progressCallback;
00451     HttpVersion httpVersion = HttpVersion::DEFAULT;
00452
00453     void clean() noexcept;
00454     void updateURL();
00455 };
00456
00457
00458
00459 } // namespace curling
00460
```

Index