My Project

Generated by Doxygen 1.9.8

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 curling::Request Class Reference

Handles HTTP requests using libcurl.

`#include <curling.hpp>`

Public Types

- enum class Method {
  GET , POST , PUT , DELETE_ ,
  MIME }

  Enumerates the supported HTTP methods.

Public Member Functions

- Request ()

  Constructor for the Request class.
- ~Request ()

  Destructor for the Request class.
- **Request** (const Request &)=delete
- Request & **operator=** (const Request &)=delete
- **Request** (Request &&)=delete
- Request & **operator=** (Request &&)=delete
- void setMethod (Method m)

  Sets the HTTP method for the request.
- void setURL (const std::string &URL)

  Sets the URL for the request.
- void setProxy (const std::string &URL)

  Sets the URL for the request.
- void setProxyAuth (const std::string &username, const std::string password)

  Sets the auth credentials for the proxy.
- void setProxyAuthMethod (long method)

  Sets the Auth Method used with proxies.
- void addArg (const std::string &key, const std::string &value)

Adds an argument to the query string of the request.

- void addHeader (const std::string &header)

  Adds a header to the request.
- void setBody (const std::string &body)

  Sets the body of the HTTP request, applicable for POST and PUT methods.
- Response **send** ()

  Sends the HTTP request using libcurl.
- void **reset** ()

  Resets the internal state for reuse of this Request instance.
- void setTimeout (long seconds)

  Sets timeout of request.
- void setFollowRedirects (bool follow)

  Sets follow redirects.
- void setConnectTimeout (long seconds)

  Sets timeout to connect.
- void setAuthToken (const std::string &token)

  Sets the header "Autorization: Bearer <token>".
- void setCookiePath (const std::string &path)

  Sets the cookie file path.
- void setUserAgent (const std::string &userAgent)

  Sets the User Agent header.
- void addFormField (const std::string &fieldName, const std::string &value)

  Adds a form field.
- void addFormFile (const std::string &fieldName, const std::string &filePath)

  Adds a form file.

3.1.1 Detailed Description

Handles HTTP requests using libcurl.

This class provides functionality to perform HTTP operations such as GET, POST, PUT, and DELETE. It manages the setup and execution of these requests with libcurl.

3.1.2 Member Enumeration Documentation

3.1.2.1 Method

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

Enumerates the supported HTTP methods.

Enumerator

GET	Represents an HTTP GET request.
POST	Represents an HTTP POST request.
PUT	Represents an HTTP PUT request.
DELETE↩_	Represents an HTTP DELETE request.
MIME	Represents an HTTP POST request but with multipart form.

3.1.3 Constructor & Destructor Documentation

3.1.3.1 Request()

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

Constructor for the Request class.

Initializes a new instance of the Request class and increments the instances counter.

3.1.3.2 ~Request()

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

Destructor for the Request class.

Decrements the instances counter and performs any necessary cleanup.

3.1.4 Member Function Documentation

3.1.4.1 addArg()

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

Adds an argument to the query string of the request.

Parameters

key	The arg key
value	The arg value

3.1.4.2 addFormField()

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

Adds a form field.

This method adds a form field for a multipart form post request

3.1.4.3 addFormFile()

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

Adds a form file.

This method add a file to upload during the multipart post request

3.1.4.4 addHeader()

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

Adds a header to the request.

Parameters

header	The header string to be added (e.g., "Content-Type: text/html").

3.1.4.5 setAuthToken()

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

Sets the header "Autorization: Bearer <token>".

This method sets the header with a given authorization token.

3.1.4.6 setBody()

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

Sets the body of the HTTP request, applicable for POST and PUT methods.

Parameters

body	The body content as a string.

3.1.4.7 setConnectTimeout()

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

Sets timeout to connect.

This method is to limit the amount of time per connection

3.1.4.8 setCookiePath()

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

Sets the cookie file path.

This method sets the name or path of the file that curl will write cookies into. Default "cookies.txt".

3.1.4.9 setFollowRedirects()

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

Sets follow redirects.

This method is to enable or disable follow redirects.

3.1.4.10 setMethod()

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

Sets the HTTP method for the request.

Parameters

m	The method to set (e.g., Method::GET, Method::POST).

3.1.4.11 setProxy()

```
void curling::Request::setProxy (
            const std::string & URL )
```

Sets the URL for the request.

Parameters

URL	The URL of the Proxy, a string.

3.1.4.12 setProxyAuth()

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

Sets the auth credentials for the proxy.

Parameters

username	
password	

3.1.4.13 setProxyAuthMethod()

```
void curling::Request::setProxyAuthMethod (
```

```
            long method )
```

Sets the Auth Method used with proxies.

Parameters

method	

3.1.4.14 setTimeout()

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

Sets timeout of request.

This method is to limit the amount of time per request

3.1.4.15 setURL()

```
void curling::Request::setURL (
            const std::string & URL )
```

Sets the URL for the request.

Parameters

URL	The URL string to be used in the HTTP request.

3.1.4.16 setUserAgent()

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

Sets the User Agent header.

This method sets the user agent header that will be sent with the request

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

- include/curling.hpp

3.2 curling::Response Struct Reference

Represents an HTTP response.

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

Public Attributes

- long **httpCode**

  The HTTP status code received in the response.
- std::string **body**

  The body content of the HTTP response as a string.
- std::map< std::string, std::vector< std::string > > headers

  A map to store HTTP headers from the response.

3.2.1 Detailed Description

Represents an HTTP response.

This structure holds details of an HTTP response, including the status code, body content, and headers.

3.2.2 Member Data Documentation

3.2.2.1 headers

```
std::map<std::string, std::vector<std::string> > curling::Response::headers
```

A map to store HTTP headers from the response.

Each header is stored with its name as the key and the corresponding value is stored into a vector, as there can be many headers with same key.

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

- include/curling.hpp

Chapter 4

File Documentation

4.1 curling.hpp

```
00001 #ifndef CURLING_HPP
00002 #define CURLING_HPP
00003
00004 #include <iostream>
00005 #include <string>
00006 #include <map>
00007 #include <vector>
00008 #include <sstream>
00009 #include <atomic>
00010 #include <mutex>
00011 #include <stdexcept>
00012 #include <algorithm>
00013 #include <cctype>
00014 #include <locale>
00015 #include <curl/curl.h>
00016
00017 namespace curling {
00018
00026 struct Response {
00027     long httpCode;
00028
00029     std::string body;
00030
00037     std::map<std::string, std::vector<std::string>> headers;
00038 };
00039
00040
00048 class Request {
00049 public:
00054     enum class Method {
00055         GET,
00056         POST,
00057         PUT,
00058         DELETE_,
00059         MIME,
00060     };
00061
00068     Request();
00069
00075     ~Request();
00076
00077     Request(const Request&) = delete;
00078     Request& operator=(const Request&) = delete;
00079     Request(Request&&) = delete;
00080     Request& operator=(Request&&) = delete;
00081
00087     void setMethod(Method m);
00088
00094     void setURL(const std::string& URL);
00095
00101     void setProxy(const std::string& URL);
00102
00109     void setProxyAuth(const std::string& username, const std::string password);
00110
00111
00117     void setProxyAuthMethod(long method);
00118
00126     void addArg(const std::string& key, const std::string& value);
```

```
00127
00133     void addHeader(const std::string& header);
00134
00140     void setBody(const std::string& body);
00141
00145     Response send();
00146
00150     void reset();
00151
00152
00158     void setTimeout(long seconds);
00159
00165     void setFollowRedirects(bool follow);
00166
00167
00173     void setConnectTimeout(long seconds);
00174
00180     void setAuthToken(const std::string& token);
00181
00187     void setCookiePath(const std::string& path);
00188
00189
00195     void setUserAgent(const std::string& userAgent);
00196
00197
00203     void addFormField(const std::string& fieldName, const std::string & value);
00204
00210     void addFormFile(const std::string& fieldName, const std::string & filePath);
00211
00212
00213 private:
00214     Method method;
00215     CURL* curlHandle;
00216     struct curl_slist* list;
00217     std::string url, args, body, cookieFile, cookieJar;
00218     curl_mime* mime = nullptr;
00219
00235     static size_t WriteCallback(void* contents, size_t size, size_t nmemb, void* userp);
00236
00251     static size_t HeaderCallback(char* buffer, size_t size, size_t nitems, void* userdata);
00252
00259     void clean();
00260
00267     void updateURL();
00268
00274     static void trim(std::string & s);
00275
00276 };
00277
00278 } // namespace curling
00279
00280 #endif // CURLING_HPP
```

Index