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 }

  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 addArg (const std::string &arg)

  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 setConnectTimeout (long seconds)

  Sets timeout to connect.

Static Public Attributes

- static std::atomic< int > **instances** = 0

  Static atomic counter to track the number of Request instances.

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.

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 & arg )
```

Adds an argument to the query string of the request.

Parameters

arg	The key-value pair formatted as a string (e.g., "key=value").

3.1.4.2 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").

3.1.4.3 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.

3.1.4.4 setConnectTimeout()

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

Sets timeout to connect.

This method is to limit the amount of time per connection

3.1.4.5 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).

3.1.4.6 setTimeout()

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

Sets timeout of request.

This method is to limit the amount of time per request

3.1.4.7 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.

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 <stdexcept>
00011 #include <algorithm>
00012 #include <cctype>
00013 #include <locale>
00014 #include <curl/curl.h>
00015
00016 namespace curling {
00017
00025 struct Response {
00026     long httpCode;
00027
00028     std::string body;
00029
00036     std::map<std::string, std::vector<std::string>> headers;
00037 };
00038
00039
00047 class Request {
00048 public:
00053     enum class Method {
00054         GET,
00055         POST,
00056         PUT,
00057         DELETE
00058     };
00059
00064     inline static std::atomic<int> instances = 0;
00065
00072     Request();
00073
00079     ~Request();
00080
00081     Request(const Request&) = delete;
00082     Request& operator=(const Request&) = delete;
00083     Request(Request&&) = delete;
00084     Request& operator=(Request&&) = delete;
00085
00091     void setMethod(Method m);
00092
00098     void setURL(const std::string& URL);
00099
00105     void addArg(const std::string& arg);
00106
00112     void addHeader(const std::string& header);
00113
00119     void setBody(const std::string& body);
00120
00124     Response send();
00125
```

```
00129     void reset();
00130
00131
00137     void setTimeout(long seconds);
00138
00139
00145     void setConnectTimeout(long seconds);
00146
00147 private:
00148     Method method;
00149     CURL* curlHandle;
00150     struct curl_slist* list;
00151     std::string url, args, body, cookieFile, cookieJar;
00152
00168     static size_t WriteCallback(void* contents, size_t size, size_t nmemb, void* userp);
00169
00184     static size_t HeaderCallback(char* buffer, size_t size, size_t nitems, void* userdata);
00185
00192     void clean();
00193
00200     void updateURL();
00201
00207     static void trim(std::string & s);
00208
00209 };
00210
00211 } // namespace curling
00212
00213 #endif // CURLING_HPP
```

Index