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 setProxy (const std::string &URL)

  Sets the URL for the request.
- 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 setConnectTimeout (long seconds)

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

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

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 & 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 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 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.

3.1.4.4 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.

3.1.4.5 setConnectTimeout()

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

Sets timeout to connect.

This method is to limit the amount of time per connection

3.1.4.6 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).

3.1.4.7 setProxy()

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

Sets the URL for the request.

Parameters

URL	The URL of the Proxy, a string.

3.1.4.8 setTimeout()

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

Sets timeout of request.

This method is to limit the amount of time per request

3.1.4.9 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

```
#ifndef CURLING_HPP
#define CURLING_HPP

#include <iostream>
#include <string>
#include <map>
#include <vector>
#include <sstream>
#include <atomic>
#include <mutex>
#include <stdexcept>
#include <algorithm>
#include <cctype>
#include <locale>
#include <curl/curl.h>

namespace curling {

struct Response {
    long httpCode;

    std::string body;

    std::map<std::string, std::vector<std::string» headers;
};


class Request {
public:
    enum class Method {
        GET,
        POST,
        PUT,
        DELETE
    };

    Request();

    ~Request();

    Request(const Request&) = delete;
    Request& operator=(const Request&) = delete;
    Request(Request&&) = delete;
    Request& operator=(Request&&) = delete;

    void setMethod(Method m);

    void setURL(const std::string& URL);

    void setProxy(const std::string& URL);

    void addArg(const std::string& key, const std::string& value);

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

    void setBody(const std::string& body);

    Response send();

    void reset();


    void setTimeout(long seconds);


    void setConnectTimeout(long seconds);

    void setAuthToken(const std::string& token);

private:
    Method method;
    CURL* curlHandle;
    struct curl_slist* list;
    std::string url, args, body, cookieFile, cookieJar;

    static size_t WriteCallback(void* contents, size_t size, size_t nmemb, void* userp);

    static size_t HeaderCallback(char* buffer, size_t size, size_t nitems, void* userdata);

    void clean();

    void updateURL();

    static void trim(std::string & s);

};

} // namespace curling

#endif // CURLING_HPP
```

Index