curlInterface

Simple Web Access

2.3.2

19 May 2023

Christopher Jefferson

Michael Young

Christopher Jefferson
Email: caj21@st-andrews.ac.uk
Homepage: http://caj.host.cs.st-andrews.ac.uk/
Address: School of Computer Science
University of St Andrews
Jack Cole Building, North Haugh
St Andrews, Fife, KY16 9SX
United Kingdom

Michael Young
Email: mct25@st-andrews.ac.uk
Homepage: http://mct25.host.cs.st-andrews.ac.uk/
Address: School of Computer Science
University of St Andrews
Jack Cole Building, North Haugh
St Andrews, Fife, KY16 9SX
United Kingdom

Contents

Chapter 1

Overview

CurlInterface allows a user to interact with http and https servers on the internet, using the 'curl' library. Pages can be downloaded from a URL, and http POST requests can be sent to the URL for processing.

1.1 Installing curlInterface

curlInterface requires the 'curl' library, available from `https://curl.haxx.se/`. Instructions for building and installing curl can be found at `https://curl.haxx.se/docs/install.html`, however in most systems curl can be installed from your OS's package manager.

1.1.1 Linux

- On Debian and Ubuntu, call: `apt-get install libcurl4-gnutls-dev`
- On Redhat and derivatives, call: `yum install curl-devel`

1.1.2 Cygwin

Install `libcurl-devel` from the cygwin package manager

1.1.3 macOS

curl is installed by default on Macs, but libcurl may be required.

- Homebrew: `brew install curl`
- Fink: `fink install libcurl4`
- MacPorts: `port install curl`

1.2 Functions

curlInterface currently provides the following functions for interacting with URLs:

1.2.1 DownloadURL

▷ DownloadURL(*URL[, opts]*) (function)

Returns: a record

Downloads a URL from the internet. *URL* should be a string describing the address, and should start with either "http://" or "https://". For descriptions of the output and the additional argument *opts*, see CurlRequest (1.2.4).

Example

```
gap> r := DownloadURL("www.gap-system.org");;
gap> r.success;
true
gap> r.result{[1..50]};
"<?xml version=\"1.0\" encoding=\"utf-8\"?>\n\n<!DOCTYPE "
```

1.2.2 PostToURL

▷ PostToURL(*URL, str[, opts]*) (function)

Returns: a record

Sends an HTTP POST request to a URL on the internet. *URL* should be a string describing the address, and should start with either "http://" or "https://". *str* should be the string which will be sent to the server as a POST request. For descriptions of the output and the additional argument *opts*, see CurlRequest (1.2.4).

Example

```
gap> r := PostToURL("www.httpbin.org/post", "animal=tiger");;
gap> r.success;
true
gap> r.result{[51..100]};
"\"form\": {\n    \"animal\": \"tiger\"\n  }, \n  \"headers\":"
```

1.2.3 DeleteURL

▷ DeleteURL(*URL[, opts]*) (function)

Returns: a record

Attempts to delete a file on the internet, by sending an HTTP DELETE request to the given URL. *URL* should be a string describing the address to be deleted, and should start with either "http://" or "https://". For descriptions of the output and the additional argument *opts*, see CurlRequest (1.2.4).

Example

```
gap> r := DeleteURL("www.google.com");;
gap> r.success;
true
gap> r.result{[1471..1540]};
"<p>The request method <code>DELETE</code> is inappropriate for the URL"
```

1.2.4 CurlRequest

▷ CurlRequest(*URL, type, out_string[, opts]*) (function)

Returns: a record

Sends an HTTP request of type *type* to a URL on the internet. *URL*, *type*, and *out_string* should all be strings: *URL* is the URL of the server (which should start with "http://" or "https://"), *type* is the type of HTTP request (e.g. "GET"), and *out_string* is the message, if any, to send to the server (in requests such as GET this will be ignored).

An optional fourth argument *opts* may be included, which should be a record specifying additional options for the request. The following options are supported:

- `verifyCert`: a boolean describing whether to verify HTTPS certificates (corresponds to the curl options `CURLOPT_SSL_VERIFYPEER` and `CURLOPT_SSL_VERIFYHOST`, the default is `true` for both);
- `verbose`: a boolean describing whether to print extra information to the screen (corresponds to the curl option `CURLOPT_VERBOSE`, the default is `false`);
- `followRedirect`: a boolean describing whether to follow redirection to another URL (corresponds to the curl option `CURLOPT_FOLLOWLOCATION`, the default is `true`);
- `failOnError`: a boolean describing whether to regard 404 (and other 4xx) status codes as error (corresponds to the curl option `CURLOPT_FAILONERROR`, the default is `false`).

As output, this function returns a record containing some of the following components, which describe the outcome of the request:

- `success`: a boolean describing whether the request was successfully received by the server;
- `result`: body of the information sent by the server (only if `success = true`);
- `error`: human-readable string saying what went wrong (only if `success = false`).

Most of the standard HTTP request types should work, but currently only body information is returned. To see headers, consider using the `verbose` option. For convenience, dedicated functions exist for the following request types:

- `DownloadURL` (1.2.1) for GET requests;
- `PostToURL` (1.2.2) for POST requests;
- `DeleteURL` (1.2.3) for DELETE requests.

Example

```
gap> r := CurlRequest("https://www.google.com",
>                     "HEAD",
>                     "",
>                     rec(verifyCert := false));
rec( result := "", success := true )
gap> r := CurlRequest("www.httpbin.org/post", "POST", "animal=tiger");;
gap> r.success;
true
gap> r.result{[51..100]};
"\"form\": {\n    \"animal\": \"tiger\"\n  }, \n  \"headers\":"
```

Index