curl简介

13331067 韩龙粤

概述

cURL是一个利用URL语法在命令行下工作的文件传输工具，1997年首次发行。它支持文件上传和下载，所以是综合传输工具，但按传统，习惯称cURL为下载工具。cURL还包含了用于程序开发的libcurl。

cURL支持的通讯协议
有 `FTP` 、 `FTPS` 、 `HTTP` 、 `HTTPS` 、 `TFTP` 、 `SFTP` 、 `Gopher` 、 `SCP` 、 `Telnet` 、 `DICT` 、 `FILE` 、 `LDAP` 、 `LDAPS` 和 `RTSP` 。

curl同样支持 `SSL证书` ， `HTTP POST` ， `HTTP PUT` ， `FTP上传` ， `基于表单的HTTP上传` 等及其他特性。

curl命令行常见用法

INTERACTING WITH PROTOCOLS

Getting a HTML page from a server is as simple as putting a HTTP URL as first cURL argument.

```
curl http://www.nytimes.com
```

cURL makes a `GET` request to the specified URL and brings back the results into the command line. This output can be saved to a file using `-O` parameter (`-o` if you want to use your own name for the file).

```
curl -O http://www.google.com/logos/2012/end_of_the_mayan_calendar-993005-hp.jpg
```

Getting a file from a FTP server is also easy:

```
curl ftp://username:password@ftp.server:21/path/to/file.tar.gz
```

There are two handy parameters, especially useful for debugging:

- `-i` or `--include` which puts headers data returned by the server in the output
- `-v` or `--verbose` which includes in the output both, headers data sent to the server and returned by the server

SENDING DATA TO THE SERVER

`-d` or `--data` parameter allows to specify data to send to the HTTP server. It simulates a form submission. Invoking cURL with this parameter will make a `POST` request (instead of default `GET`). cURL will set `Content-Type` as `application/x-www-form-urlencoded` automatically. Multiple fields should be separated by `&` or specified with a separate `-d` parameter for each field.

```
curl -d 'name=zaiste' -d 'age=17' http://server/
curl -d 'user[name]=zaiste' http://server/
```

It is also possible to send data in JSON format. `Content-Type` has to be explicitly set to `application/json` as it is used by the server to decide how to handle the incoming data.

```
curl -d '{"user": {"name": "zaiste"}}' -H "Content-Type: application/json" http://server/
```

For convenience, the data can be also read from a file or from the standard input:

```
curl -d @sample-data.txt http://server/widgets
curl -d @- http://server/widgets
{ "name": "Widget 1" }
^D
```

We could also specify the `Accept` header which is used to tell the server what content types we accept as a client. The server will respond with `Content-Type` that will inform us about the content type of the returned data. As a result, `Content-Type` header is used to specify the type of data being sent both, from and to the client.

RESTFUL INTERACTION

cURL comes in handy when interacting with RESTful APIs.

Let's start by creating a simple resource from a JSON. As mentioned earlier, `POST` request override default GET when using -d parameter.

```
curl -H "Content-Type: application/json" \
-d '{"name": "Widget 1", "desc":"This is widget 1", "amount":"17"}' \
http://express-mongoose-coffee.herokuapp.com/widgets
{
    "__v": 0,
    "name": "Widget 1",
    "desc": "This is widget 1",
    "amount": 17,
    "_id": "50f2fcc18270670200000001",
    "created_at": "2012-11-10T21:26:07.333Z"
}
```

For a resource update we will specify `PUT` method with `-X` or `--request` parameter.

```
curl -X PUT \
     -H "Content-Type: application/json" \
     -d '{"amount": "21"}' \
     http://express-mongoose-coffee.herokuapp.com/widgets/50f2fcc18270670200000001
```

It is also possible to simulate `PUT` method with `POST` by using a special `X-HTTP-Method-Override` header.

```
curl -H "X-HTTP-Method-Override: PUT" \
     -H "Content-Type: application/json" \
     -d '{"amount": "9999"}'  \
     http://express-mongoose-coffee.herokuapp.com/widgets/50f2fcc18270670200000001
```

Finally we can delete the resource using DELETE method.

```
curl -X DELETE  \
     http://express-mongoose-coffee.herokuapp.com/widgets/50f2fcc18270670200000001
```

Similar to `PUT` , we can ask the server to override a `POST` request if setting `DELETE` explicitly is not possible.

```
curl -H "X-HTTP-Method-Override: DELETE" \
     -H "Content-Type: application/json" \
     -d '{"amount": "9999"}'  \
     http://express-mongoose-coffee.herokuapp.com/widgets/50f2fcc18270670200000001
```

LOGIN WITH BASIC AUTH

HTTP Basic authentication is a simple way to secure web pages. If SSL is not used, the credentials are passed in plain text. cURL has `-u|--user` option that allows to login using that method.

```
curl -u username:password http://server/
```

LOGIN WITH COOKIE

cURL can be also used to login using a cookie. In the exampleb below, we simulate a form submission of login credentials, and we store returned cookies in a file with `-c` parameter. For following requests, we can then use that cookies (`-b` parameter) to authenticate. The process simulates how a browser handles the session.

```
curl -d "username=admin&password=admin" -c cookies http://server/login
curl -L -b cookies http://server/
```

Things may get a little tricker when CSRF validation is enabled. In such case it would be also necessary to extract CSRF token.

Reference:

1. wikipedia - curl
2. Introduction to curl