Qué es?

cURL es una librería de funciones para conectar con servidores para trabajar con ellos. El trabajo se realiza con formato URL. Es decir, sirve para realizar acciones sobre archivos que hay en URLs de Internet, soportando los protocolos más comunes, como http, ftp, https, etc.

-Es una herramienta multiplataforma.

-Soporta los protocolos FTP, FTPS, HTTP, HTTPS, TFTP, SCP, SFTP, Telnet, DICT, FILE y LDAP, entre otros.

-Es un proyecto de software consistente en una biblioteca (libcurl) y un intérprete de comandos (curl) orientado a la transferencia de archivos.

-Es una herramienta válida para simular las acciones de usuarios en un navegador web.

-LibcURL es la biblioteca/API correspondiente que los usuarios pueden incorporar en sus programas donde cURL actúa como un envoltorio (wrapper) aislado para la biblioteca LibcURL.7​ LibcURL se usa para proveer capacidades de transferencia de URL a numerosas aplicaciones, tanto libres y open source como privativas. La biblioteca "libcurl" puede ser usada desde más de 30 lenguajes distintos.

Primer ejemplo

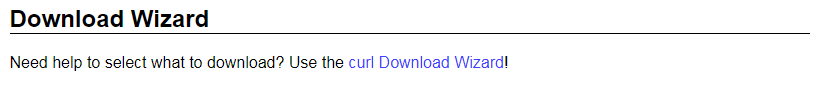
curl -X GET 'http://samples.openweathermap.org/data/2.5/forecast?id=524901&appid=b1b15e88fa797225412429c1c50c122a1'

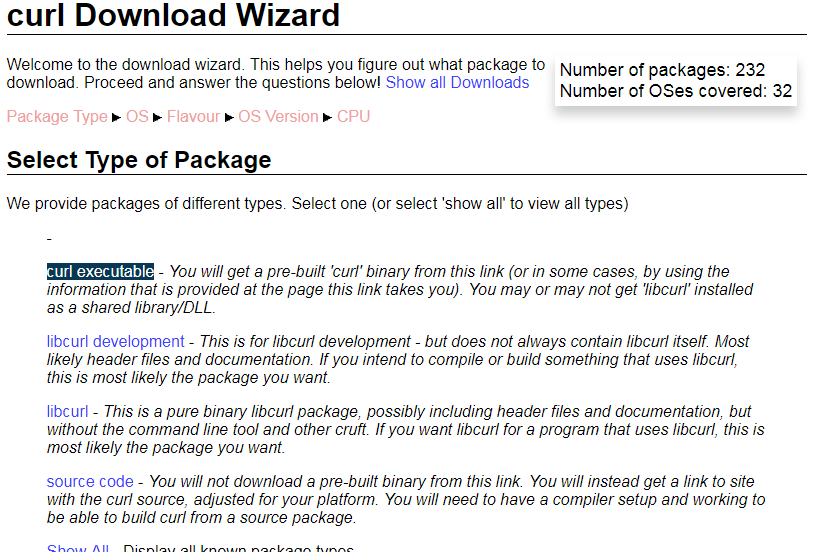
Instalación

Página de descarga:

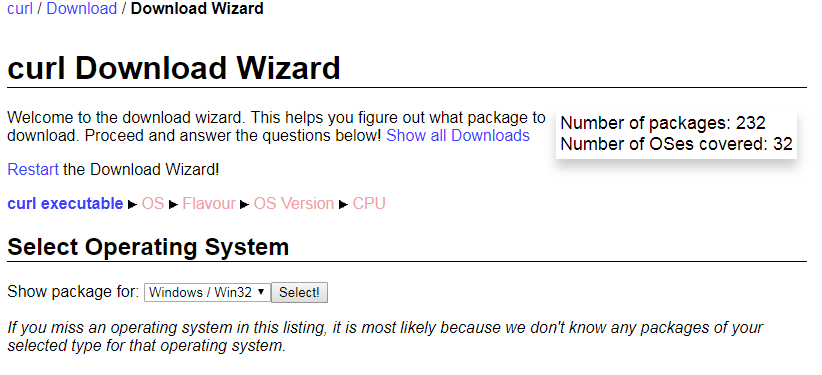
https://curl.haxx.se/download.html

Elegir la versión deseada o instalar desde el wizard

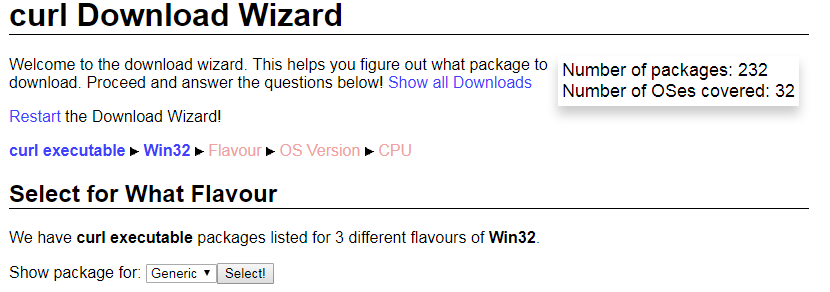




Seleccionar el sistema operativo:



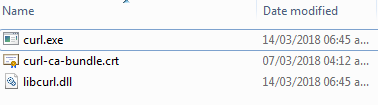
Seleccionar el tipo de paquete



Descomprimir

Ir a la carpeta descomprimida, subcarpeta bin

C:\instaladores\curl-7.59.0-win32-mingw\bin



Abrir una ventana de comandos.

Posicionarse en la carpeta donde se encuentra curl.exe

|  |
| --- |
| Curl --version |
| C:\instaladores\curl-7.59.0-win32-mingw\bin>curl --version  curl 7.59.0 (i386-pc-win32) libcurl/7.59.0 OpenSSL/1.1.0h (WinSSL) zlib/1.2.11 b  rotli/1.0.3 WinIDN libssh2/1.8.0 nghttp2/1.31.0  Release-Date: 2018-03-14  Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s  rtsp scp sftp smb smbs smtp smtps telnet tftp  Features: AsynchDNS IDN IPv6 Largefile SSPI Kerberos SPNEGO NTLM SSL libz brotli  TLS-SRP HTTP2 HTTPS-proxy MultiSSL |

Utilización de cURL en el intérprete de comandos

1. Lectura y guardado de URL's

- Lectura simple de una página web:

curl http://www.google.es

- Lectura de una página web segura:

curl https://www.secure-site.com

- Guardado de un página web en un fichero. Utilizaremos la opción -o seguida del nombre del fichero con el que guardaremos el recurso:

curl -o prueba.html http://www.google.es

- Utilizar la autenticación HTTP:

curl -u user:pass https://www.secure-site.com

- Hay ocasiones en las que una página nos redirige automáticamente a otra. Por defecto cURL no sigue esas redirecciones por lo que tendremos que indicárselo:

curl -L http://www.ejemplo.com/

- Por último, podemos ver la respuesta completa a una petición, que incluye una descripción tanto de la entrada como de la salida de datos del servicio. Esto lo haremos mediante la opción --trace-ascii seguido del nombre del fichero donde guardaremos la traza. Por ejemplo con curl --trace-ascii fichero\_salida http://www.google.com crearía un fichero con el siguiente contenido:

== Info: About to connect() to www.google.com port 80 (#0)

== Info: Trying 173.194.78.147...

== Info: connected

== Info: Connected to www.google.com (173.194.78.147) port 80 (#0)

=> Send header, 78 bytes (0x4e)

0000: GET / HTTP/1.1

0010: User-Agent: curl/7.27.0

0029: Host: www.google.com

003f: Accept: \*/\*

004c:

== Info: additional stuff not fine transfer.c:1037: 0 0

== Info: HTTP 1.1 or later with persistent connection, pipelining supported

<= Recv header, 20 bytes (0x14)

0000: HTTP/1.1 302 Found

<= Recv header, 33 bytes (0x21)

0000: Location: http://www.google.es/

<= Recv header, 24 bytes (0x18)

0000: Cache-Control: private

<= Recv header, 40 bytes (0x28)

0000: Content-Type: text/html; charset=UTF-8

<= Recv header, 157 bytes (0x9d)

0000: Set-Cookie: PREF=ID=e25aaccb54054ef0:FF=0:TM=1369215269:LM=13692

0040: 15269:S=83ZlwudqItjjuG1V; expires=Fri, 22-May-2015 09:34:29 GMT;

0080: path=/; domain=.google.com

<= Recv header, 226 bytes (0xe2)

0000: Set-Cookie: ...

2. Peticiones GET

El tipo básico de petición es una petición GET. Por lo que es un buen comienzo para demostrar la sintaxis básica de cURL.

Evidentemente, se puede hacer peticiones GET con parámetros en la URL (query strings).

http://www.ejemplo.com/pages.php?id=35

- Se puede hacer petición a un rango de páginas, del mismo dominio, gracias a las expresiones regulares. En el siguiente ejemplo, se van a guardar las páginas de la 1 a 12 en sus correspondientes ficheros pages[1-12].html:

curl -o pages#1.html http://ejemplo.com/pages.php?id=[1-12]

- Evidentemente si el servicio RESTful tiene reglas de reescritura para sus URL's, podemos hacer peticiones GET sin usar parámetros en la URL:

curl http://localhost/login\_restful/Api.php?peticion=usuarios //con parametros en la url

curl http://localhost/login\_restful/usuarios //gracias a las reglas de reescritura

3. Peticiones POST

Podemos simular el envío de datos (simulando formulario) con dos opciones. Con la opción -d indicaremos el envío de variables y valores y con -F indicaremos la ruta de un fichero a subir.

- Parámetros simples:

curl -d "campo1=valor\_campo1&campo2=valor\_campo2 ..." url\_a\_donde\_hacemos\_el\_post.com

- Subida de ficheros:

curl -F "nombre\_del\_input=@path\_al\_file\_a\_subir" url\_a\_donde\_hacemos\_el\_post.com

Hay que darse cuenta que la arroba (@) delante del path (ruta) del fichero es obligatoria. Y nombre\_del\_input es el nombre que asignaríamos al campo de entrada del fichero en el formulario que simulamos: input name="nombre\_del\_input" type="file".

También podemos indicar un nombre específico para el fichero que queremos subir:

curl -F "nombre\_del\_input=@path\_al\_file\_a\_subir;filename=nombre\_del\_file" url\_a\_donde\_hacemos\_el\_post.com

- Envío de datos y ficheros.

Que pasa si queremos hacer una petición en la que se envíen tanto datos como un fichero. Podríamos pensar en una combinación de las opciones -d y -F. Pero eso ocasionaría un error ya que tienen diferentes content-types. Mientras que -F usa multipart/form-data para el envío Post; -d usa application/x-www-form-urlencoded. La solución es usar la opción -F varias veces.

curl -i -F "name=test" -F "filedata=@localfile.jpg" http://www.ejemplo.org/upload

La opción -i no es necesario usarla pero nos ayudará a mostrar la respuesta y sus cabeceras, lo que nos puede resultar útil. Por ejemplo curl -i htttp://www.google.com devolvería lo siguiente:

HTTP/1.1 302 Found

Location: http://www.google.es/

Cache-Control: private

Content-Type: text/html; charset=UTF-8

Set-Cookie: PREF=ID=0e54f1c8469...Wu; expires=Fri, 22-May-2015 09:39:28 GMT; path=/; domain=.google.com

Set-Cookie: NID=67=ll89Z...; expires=Thu, 21-Nov-2013 09:39:28 GMT; path=/; domain=.google.com; HttpOnly

P3P: CP="This is not a P3P policy! See http://www.google.com/support/accounts/bin/answer.py?hl=en&answer=151657 for more info."

Date: Wed, 22 May 2013 09:39:28 GMT

Server: gws

Content-Length: 218

X-XSS-Protection: 1; mode=block

X-Frame-Options: SAMEORIGIN

//Esto sería la respuesta propiamente dicha. Lo anterior es la cabecera de la respuesta

<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">

<TITLE>302 Moved</TITLE></HEAD><BODY>

<H1>302 Moved</H1>

The document has moved

<A HREF="http://www.google.es/">here</A>.

</BODY></HTML>

4. Peticiones PUT

PUT y DELETE son considerados 'custom-method', al contrario que GET y POST. Por lo tanto tenemos que indicarlo a la herramienta cURL con la opción -X:

curl -X PUT -d "telefono=1-800-999-9999" http://www.servidor.com/personas/persona/1

Recuerda que -d se utiliza para pasar parámetros. Pero solo para POST, PUT y DELETE.

5. Peticiones DELETE

curl -X DELETE http://www.servidor.com/personas/persona/1

curl -X DELETE http://www.servidor.com/personas.php?id=1

curl -X DELETE -d "id=1" http://www.servidor.com/borrarUsuario

6. Lectura de información

- Podemos mostrar únicamente las cabeceras de la respuesta y no la respuesta e sí. En esta ocasión no utilizaremos la opción -i, vista anteriormente, ya que mostraba ambas cosas.

curl --head http://www.google.com/

También se puede mostrar las cabeceras de cualquier recurso:

curl --head http://www.google.com/logo\_plain.jpg

- En ocasiones es útil indicar quien es el que solicita la petición al servidor y para ello podemos simular un campo 'referer' con la opción -e:

curl -e http://algun\_sition.com http://www.ejemplo.com/

- Además es posible indicar en una petición el tipo de cliente (campo user-agent) que hace dicha petición:

curl -A "Mozilla/5.0" http://www.ejemplo.com

7. FTP

Un cosa que tienes que tener en cuenta antes de hacer peticiones a un ftp privado, es que tanto el usuario como la contraseña se van a enviar en texto plano en las cabeceras, a no ser que el servidor admita SSL. Si es así, utiliza ftps en las peticiones.

- Listar un directorio de un ftp:

curl ftp://usuario:pass@ejemplo.com/directory/

También podemos usar la opción -u para introducir el usuario y la contraseña:

curl -u name:passwd ftp://ejemplo.com/full/path/to/file

- Subir un fichero a un ftp remoto

curl -T test.txt ftp://user:pwd@myserver/mydir/

//o también podemos subir varios ficheros a la vez

curl -T {test.txt,test2.txt} ftp://user:pwd@myserver/mydir/

-Descargar ficheros

Se pueden descargar datos de un ftp, como hemos visto antes en el apartado de descargas páginas, con la opción -o. Que indicará el nombre con el que guardaremos el recurso.

curl ftp://user:password@sitioejemplo.com/mp3/musica.zip -o musica.zip

Nota: si a todas las instrucciones añadimos la opción -v veremos los posibles errores cometidos.

Otros Ejemplos:

Probar nuestra api con curl: https://ajgallego.gitbooks.io/laravel-5/content/capitulo\_5\_curl.html

Probar api con protocolo OAuth - OpenID Connect: https://www.instagram.com/developer/authentication/