libRevCurl

Introduction

libRevCurl is a revolution library that provides a wrapper around the curl command line tool*. It exposes many of the configuration parameters of curl, and allows monitoring of the progress of uploads/downloads. It also allows the simultaneous execution of multiple instances of curl, thus allowing for the kind of parallel communications that some modern web-services require (it was written to facilitate the use of the various amazon web-services). It is currently limited to the http and https protocols, but I intend to add support for ftp and other protocols in later versions.

Using libRevCurl involves creating instances ("curls"), and setting the various properties of those instances, and then executing the instances, and finally deleting the instances.

This is a very simple example of using libRevCurl:

```
--------------------------------------------------------------------------

put curl.new() into tCurl -- tCurl now holds the id of the new curl instance
curl.setUrl tCurl, "http://www.google.com"
curl.execute tCurl
wait until curl.done(tCurl) with messages
put curl.response(tCurl)
curl.cleanup tCurl

--------------------------------------------------------------------------
```

Generally, such a function might be better (more efficiently) done using libUrl, but it serves as a basic example.

libRevCurl allows a couple of things that libUrl can't do - http PUT, http DELETE, http HEAD, and uploading from files on disc. Also, when the curl instances run, they run as processes separate from Revolution, allowing your application to remain responsive and able to carry out other tasks (including starting more curl instances).

However, for the ease of consistency in my own applications, I've included some simple handlers that do these simple operations:

curl.get("http://www.google.com") will return exactly the same data as the function above, but takes care of setting up, execution and cleanup.

curl.post("http://someserver.com, pPostData) provides a similarly simple function for posting to a server.

A slightly more complex example. This is an http PUT request. Obviously, this will only work with servers that support the PUT method (not many do, though the Amazon S3 storage service is one that requires it).

```
put curl.new() into tCurl
curl.setUrl tCurl, “http://someServer.com/fileUploads/newFile”
curl.setMethod tCurl, “PUT”
curl.setInputFile “/Users/marksmith/Desktop/fileToUpload”
curl.execute tCurl
repeat until curl.done(tCurl)
  wait 300 millisecs with messages
  put curl.prog(tCurl) into tPercent
  set the thumbpos of scrollbar “sb1” to tPercent
end repeat
get curl.status(tCurl)
if it is “error” then
  put “error:” && it
else if it is “cancelled” then
  put “cancelled”
else
  put “success:” && it
end if
curl.cleanup tCurl
```

----

*Curl comes as part of the normal distribution on Macintosh OS X, and nearly all normal distributions of Linux. It does not come as standard on Windows, but binaries are available, and can be used with this library without problem. See the “curl.setCurlPath” entry in the dictionary. You can download windows binaries from here: http://curl.haxx.se/download.html

----

Mark Smith - August 2008