Clase para usar cURL en PHP

En los artículos anteriores hemos aprendido a usar cURL con PHP. Por supuesto, hay una gran cantidad de opciones que no hemos tratado. Es lógico. Son tantas que, prácticamente, necesitaríamos un libro para analizarlas todas detalladamente. Sin embargo, con las que ya conocemos podemos sacarle mucho partido a cURL.
Y, claro, ahora vamos a lo cómodo. Nos creamos una clase, que ya la tendremos y nos servirá para cualquier proyecto. Después, mostraremos un ejemplo de uso de esa clase, instanciándola en un objeto, para comunicar nuestra aplicación con un servidor remoto.

LA CLASE Curl_Connect

El código de la clase que podemos usar para enviar ficheros a un servidor remoto, o recibirlos del mismo, usando cURL es Curl_Connect.class.php, con el siguiente listado:

```
<?php
/**
 * Connect by cURL with
 * FTP with Implicit SSL/TLS
 * or Explicit SFTP.
 *
 * Simple wrapper for cURL functions to transfer a file.
 */
class Curl_Connect {
 private $ch;
 private $url;
 public function __construct($protocol, $implicit, $server, $port = 990, $username, $password, $remote_path = '', $passive_mode = false) {
  // check for blank protocol
  if (!$protocol)
   throw new Exception('Protocol is blank.');
  // check for blank username
  if (!$username)
   throw new Exception('Username is blank.');
  // don't check for blank password (highly-questionable use case, but still)
  // check for blank server
  if (!$server)
   throw new Exception('FTP Server is blank.');
  // check for blank port
  if (!$port)
   throw new Exception ('FTP Port is blank.');
  // set host/initial path
  $this->url = $protocol.'://'.$server.'/'.$remote_path;
  // setup connection
  $this->ch = curl_init();
  // check for successful connection
  if (!$this->ch)
   throw new Exception('Could not initialize cURL.');
```

```
 // connection options
 $options = [
  CURLOPT_USERPWD        => $username.':'.$password,
  CURLOPT_PORT           => $port,
  CURLOPT_TIMEOUT        => 30,
  CURLOPT_FTP_SSL        => CURLFTPSSL_ALL, // require SSL For both control and data connections
  CURLOPT_FTPSSLAUTH     => CURLFTPAUTH_DEFAULT, // let cURL choose the FTP authentication method (either SSL or TLS)
 ];
 // cURL FTP enables passive mode by default, so disable it by enabling the PORT command and allowing cURL to select the IP address for the data connection
 if (!$passive_mode)
  $options[CURLOPT_FTPPORT] = '-';
 // If implicit mode
 if ($implicit) {
  $options[CURLOPT_SSL_VERIFYPEER] = false;
  $options[CURLOPT_SSL_VERIFYHOST] = false;
 } else { // No implicit mode
  $options[CURLOPT_SSL_VERIFYPEER] = true;
  $options[CURLOPT_SSL_VERIFYHOST] = true;
 }

 // set connection options, use foreach so useful errors can be caught instead of a generic "cannot set options" error with curl_setopt_array()
 foreach ($options as $option_name => $option_value) {
  if (!curl_setopt($this->ch, $option_name, $option_value))
   throw new Exception(sprintf('Could not set cURL option: %s', $option_name));
 }
      }

public function upload($remote_file, $local_file) {
 curl_setopt($this->ch, CURLOPT_UPLOAD, true);
 curl_setopt($this->ch, CURLOPT_RETURNTRANSFER, false);
 // set file name
 if (!curl_setopt($this->ch, CURLOPT_URL, $this->url.$remote_file))
  throw new Exception ("Could not set cURL file name: $remote_file");
 // open memory stream for writing
 $stream = fopen('php://temp', 'w+');
 // check for valid stream handle
 if (!$stream)
  throw new Exception('Could not open php://temp for writing.');
 // write file into the temporary stream
 fwrite($stream, file_get_contents($local_file));
 // rewind the stream pointer
 rewind($stream);
 // set the file to be uploaded
 if (!curl_setopt($this->ch, CURLOPT_INFILE, $stream))
  throw new Exception("Could not load file $remote_file");
 // upload file
 if (!curl_exec($this->ch))
```

```
  throw new Exception(sprintf('Could not upload file. cURL Error: [%s] - %s', curl_errno($this->ch), curl_error($this->ch)));
 // close the stream handle
 fclose($stream);
}
public function download($remote_file, $local_file)
{
 curl_setopt($this->ch, CURLOPT_UPLOAD, false);
 curl_setopt($this->ch, CURLOPT_RETURNTRANSFER, true);
 // set file name
 if (!curl_setopt($this->ch, CURLOPT_URL, $this->url.$remote_file))
  throw new Exception("Could not set cURL file name: $remote_file");
 $content = curl_exec($this->ch);
 $file_handle = fopen($local_file, "w+");
 fputs($file_handle, $content);
 fclose($file_handle);
}

public function __destruct() {
 @curl_close($this->ch);
}
}
```

No vamos a entrar en muchos detalles aquí sobre el funcionamiento de cada método, porque todo lo que contienen se ha explicado en los artículos previos de esta serie, por lo que sería repetirnos. La clase se basa en un manejador de cURL (llamado $ch por curl handle). Emplea cuatro métodos: un constructor y un destructor que son de los llamados métodos mágicos de PHP (puedes leer más sobre métodos mágicos en este enlace). Como ya sabes, el constructor se ejecuta al crear un objeto, y el destructor cuando ese objeto ha terminado su ciclo de vida. Además, tiene un método para enviar ficheros a un servidor remoto, y otro para descargarlos a la ubicación de nuestra aplicación.

USANDO LA CLASE

Esta es la parte que nos interesa. En primer lugar debemos empezar, claro está, por incluir la clase en nuestro código, así:

```
include ('Curl_Connect.class.php');
```

Si está en otra ruta distinta, deberemos especificarla, por supuesto.

El siguiente paso es definir las variables que vamos a pasar al constructor, siguiendo la firma del mismo, y crear un objeto para usar cURL, así:

```
$protocol = 'ftp'; // El protocolo de comunicación$implicit = true; // Usamos FTP implícito sobre TSL.$server = '245.245.245.245'; // IP del servidor remoto$port = 7693; // Puerto de datos del servidor remoto$username = 'user'; // El nombre de usuario con el que conectar al servidor remoto$password = 'password'; // La contraseña de acceso al servidor remoto$path = 'upload/'; // Ruta del servidor donde leeremos o escribiremos$pasive = true; // Un servidor pasivo
$curl_object = new Curl_Connect($protocol, $implicit, $server, $port, $username, $password, $path, $pasive);
```

Para enviar un archivo local a nuestra aplicación, al servidor remoto, usamos el método upload(). Este recibe dos argumentos. El primero es el nombre con el que grabaremos el fichero en remoto, y el segundo es el nombre con el que tenemos el fichero en nuestra aplicación. Lo usamos así:

```
$curl_object->upload('p1_remote.txt', 'p1_local.txt');
```

Por el contrario, para descargar un archivo desde el servidor remoto al local usaremos el método download(). La firma de este método es la misma que la del anterior, así:

```
$curl_object->download('p1_remote.txt', 'p1_local.txt');
```

Y ya lo tenemos. Más fácil, imposible.