TorneLIB\Module\Network\NetWrapper

Welcome to the MODULE_CURL replacement. Yes, this is the generic module that replaces MODULE_CURL which will become very clear if you take a look at the backwards compatible class MODULE_CURL directly in the codebase. It is recommended that you drop the use of MODULE_CURL as soon as possible as it is just there to help 6.0-users to stay alive. The backward compatibility is not entirely safe either, but if you prefer to stay with an autoselected wrapper for communication, this is the way. Autotests are documented in several places - here or here but if you're more comfortable with written examples here, you can stay.

Most of the wrappers are built the same way with a wrapper interface. This also makes it possible to build your own drivers. The NetWrapper is where you should start build your own drivers if you want to go that way. The wrappers included in the core set may be enough to get started for real, but if you want to do it a harder way, you can for example build a driver that runs through NetWrapper - but with support for Guzzle or Zend. This document does not yet cover those parts.

Getting started

All wrappers is built around a request method. In version 6.0, each request method has its own method: doGet for GET-requests, doPost for POST-requests, doDelete for DELETE, and so on. In 6.1 we stay with the request method. NetWrapper and MODULE_CURL however supports the old do-methods, even if it is highly recommended to not use those methods. The difference between this module and the others is that NetWrapper finds a preferred driver where curl has priority. If curl is unavailable, NetWrapper will automatically try to run with internal streams by default.

I've created a method in netWrapperTest to demonstrate, that will be left over there for you to test with:

Simple GET Request

```
$wrapper = new NetWrapper();
$wrapper->request('https://ipv4.netcurl.org/');
$parsed = $wrapper->getParsed();
static::assertNotEmpty(filter_var($parsed->ip, FILTER_VALIDATE_IP));
```

In this example, we want to get data from https://ipv4.netcurl.org which is a page that returns the content of your request as a json object. To get the json object correctly formatted so you don't have to parse it yourself we use getParsed()-method. Feel free to test here. If you explicitly need to test IPv6, you can use https://ipv6.netcurl.org for this.

Doing it on a simple row

A big difference between 6.0 and 6.1 is the chaining, which did not exist in the old versions as it supported PHP 5.3 in the "good old (horrible) times". Almost everything in netcurl 6.1 is chained, so the above method could be pushed into one simple row, like this:

The oneliner

```
 $parsed = (new NetWrapper())->request(sprintf('https://ipv4.netcurl.org/'))->getParsed();
static::assertNotEmpty(filter_var($parsed->ip, FILTER_VALIDATE_IP));
```