cHttpRequest

The cHttpRequest classes are a collection of classes to deal with HTTP requests. This is useful if your plugins or modules need to get data from an external service.

Overview

CONTENIDO is able to utilize cURL or PHP sockets to perform an HTTP request. The class cHttpRequest hides the differences between the two from the developer. You can let CONTENIDO choose which method to use:

```
$httpRequest = cHttpRequest::getHttpRequest("http://example.com/");
```

To perform the actual request you have to use the request function:

```
$output = $httpRequest->request();
```

The output variable now contains the body of the HTTP response.

POST Requests

Often it is required to send POST requests to other servers. You can do this by adding POST parameters to the object:

```
$httpRequest = cHttpRequest::getHttpRequest("http://example.com/");
$httpRequest->setPostParams(array(
        'somePostKey' => 'somePostValue'
))->request();
```

Every functions starting with "set" returns a reference to the object itself so that you can chain functions like shown above.

The request function will always use the POST method if any POST parameters were added to the object and GET otherwise. If you need a specific method you can use

```
$httpRequest->sendGetRequest();
```

or

```
$httpRequest->sendPostRequest();
```

All request functions have two boolean parameters. By default they return only the body of the response. If you do not care about the response and only want to know if the request was successful or not, you can use

```
// This turns off returning the header and will only return true if the server responded with HTTP status code
200
$httpRequest->request(false);
```

If you need to read the headers of the response yourself you can use

```
// The function will now return the complete response of the server without omitting the headers
$httpRequest->request(true, true);
```

Custom Headers

The class also supports adding custom HTTP headers. Be careful with this though, since some headers might make the server act in an unexpected way!

```
$httpRequest->setHeaders(array(
        'X-Some-Custom-HTTP-Header' => 'some value'
))->request();
```

Force the usage of cURL/sockets

Since both classes inherit from the base cHttpRequest classes you can use them directly in your code too.

```
$curlRequest = new cHttpRequestCurl('http://example.com/');
$curlRequest->request();
```

The public interface is at the same for both of them. The cHttpRequestCurl class has two extra methods though:

```
// Use this to specify your own options for the cURL channel
$curlRequest->setOpt(CURLOPT_SOME_CURL_OPTION, 'some curl value');

// Use this to get the cURL channel used by the class
$curl = $curlRequest->getCurl();
```

Be careful though! cURL might not be available on all systems, so it is better to use the cHttpRequest class itself.