Project Name Documentation

Release 0.1

December 01, 2016

Contents

This documentation is out of date, please refer to the repository Phraseanet-PHP-SDK <https://github.com/alchemy-fr/Phraseanet-PHP-SDK

CHAPTER 1

Introduction

Phraseanet PHP SDK is an object oriented PHP library that gives you access to the your Phraseanet ressources from your PHP application/website.

CHAPTER 2

Design

This library has been strongly inspired by Doctrine ORM design and sits on top of *Phraseanet API<https://docs.phraseanet.com/Devel>*.

The aim is to let you build your application/mashup as if you were connected to a database.

This library provide a set of entities mapped to the logical structure of the API and a set of related repositories to fetch this entities in various ways.

This is a **read only** library ; current version only allow read from the API.

CHAPTER 3

Installation

We rely on composer to use this library. If you do no still use composer for your project, you can start with this `composer.json` at the root of your project:

```
{
    "require": {
        "phraseanet/php-sdk": "~0.2"
    }
}
```

Install composer :

```
# Install composer
curl -s http://getcomposer.org/installer | php
# Upgrade your install
php composer.phar install
```

You now just have to autoload the library to use it :

```
<?php
require 'vendor/autoload.php';
```

This is a very short intro to composer. If you ever experience an issue or want to know more about composer, you will find help on their website http://getcomposer.org/.

CHAPTER 4

The SDK

To use the SDK, you will need an **Entity Manager**. This EntityManager, provides methods to access repositories, delegating all work to an HTTP client.

Most of the time, you'll only have to set your credentials to this client. If you want to customize it (use a configuration for your credentials, add Memcached or Redis cache...) see the dedicated section.

4.1 Phraseanet SDK client

The Client object is the heart of the SDK, it performs HTTP requests and handles authentication flow as well.

Get a Phraseanet client

For the following example we will use Guzzle Http Client Adapter which use Guzzle http://guzzlephp.org/index.html, highly customizable library.

```php
<?php
use PhraseanetSDK\Client;
use PhraseanetSDK\HttpAdapter\Guzzle as GuzzleAdapter;

$HttpAdapter = GuzzleAdapter::create();
$HttpAdapter->setBaseUrl('http://url-to-phraseanet.net/');

$client = new Client('Your API Key', 'Your API Secret', $HttpAdapter);
```

Note: You have to create an application in Phraseanet to get credentials. This application will be created in "Account" => "Applications".

4.1.1 Authentication

All requests performed on Phraseanet API are authenticated requests.

Phraseanet API implements OAuth2.0 http://oauth.net/2/ authentication flow, this means you MUST provide an oAuth ACCESS_TOKEN to request the API otherwise you will get a `PhraseanetSDK\Exception\UnauthorizedException`.

There are two ways to use the SDK in your application. Either you will use the same token for every request, either you will use a custom token using the whole oAuth2 authentication flow.

If you want to use this second way, please read the dedicated article in the recipes.

Use developer Token

```
<?php
$client->setAccessToken('YOUR_ACCESS_TOKEN');
```

Full authentication flow

If you do not want to use the developer token and prefer the oAuth2 full authentication flow, read the dedicated doc in the recipes.

4.2 Entity Manager

Once you have a client, you'll have access to the `EntityManager` ; it is the central access point to Phraseanet ressources.

```
<?php
use PhraseanetSDK\EntityManager;

$entityManager = new EntityManager($client);

// retrieve a collection of all feeds available
$feeds = $entityManager->getRepository('Feed')->findAll();
```

4.2.1 Repositories

Repositories are access point to entities. To get a repository instance use the Entity Manager

```
<?php
// $em is an Entity Manager instance
$recordRepository = $em->getRepository('Record');
// return the 20 latest records
$recordRepository->find(0, 20);

$basketRepository = $em->getRepository('Basket');
// return all the active baskets of the user
$basketRepository->findAll();
```

Repositories methods depend of the repository. Here is the list of all available repositories and a link to the related API documentation.

- Basket repository
- BasketElement repository
- Caption repository
- Databox repository
- DataboxCollection repository
- DataboxDocumentStructure repository
- DataboxStatus repository
- Entry repository
- Feed repository

- Metadatas repository
- Quarantine repository
- Record repository
- RecordStatus repository
- Subdef repository

4.2.2 Entities

Entities are ressources with identity. Their identity has a conceptual meaning inside the Phraseanet API Domain.

Since the SDK is a read only library, you will never have the need to create a new entity object but just getting them trought the API.

```
<?php
// $em is an Entity Manager instance
$recordRepository = $em->getRepository('Record');

// return the 20 latest records as record entities
$records = $recordRepository->find(0, 20);

foreach($records as $record) {
    // $record is an entity

    // return the title of the record
    $record->getTitle();

    // return a Subdef entity corresponding to the thumbnail
    $thumbnail = $record->getThumbnail();

    if($thumbnail) {
        $url = $thumbnail->getPermalink()->getUrl();
    }
}
```

Here is a complete list of all entities provided by the API and a link to their API doc.

- Basket
- BasketElement
- BasketValidationChoice
- BasketValidationParticipant
- Databox
- DataboxCollection
- DataboxDocumentStructure
- DataboxStatus
- Feed
- FeedEntry
- FeedEntryItem
- Metadatas

- Permalink
- Quarantine
- QuarantineSession
- Query
- QuerySuggestion
- Record
- RecordCaption
- Subdef
- Technical

4.3 Fetching datas from API

Now you have your entity manager, it is very easy to fetch datas. You have to get the repository type that relies on the data type you want.

Let's say you want records, then you need the Record Repository.

Look at the Record Repository API, and see that there are three ways to retrieve records :

- findById
- find
- search

Some of the repository methods will fetch one entity and some others will retrieve a collection of entities.

Fetching one element

```
<?php
// $em is an Entity Manager instance
$recordRepository = $em->getRepository('Record');

$databoxId = 1;
$recordId = 234;

// Fetch one record identified by its databox id ans its own id
// Return a Record Entity object
$record = recordRepository->findById($databoxId, $recordId);
```

Fetching a collection of elements

```
<?php
// $em is an Entity Manager instance
$recordRepository = $em->getRepository('Record');

$offsetStart = 1;
$perPage = 20;

// Fetch 20 records
// Return a Doctrine Array Collection object
$records = recordRepository->find($offsetStart, $perPage);

foreach($records as $record) {
```

```
    echo $record->getTitle() . "\n";
}
```

4.4 Lazy Loading

Whenever you have an entity instance at hand, you can traverse and use any associations of that entity to retrieve the associated objects.

```
<?php
// Fetch one record
$record = $recordRepository->findById(1, 87);

// Will execute a request to the api to load the record status
$status = $record->getStatus();

// performs another request to get subdefs
foreach($record->getSubdefs() as $subdef) {
    echo "Sub definition " . $subdef->getName() . " has URL " . $subdef->getPermalink()->getUrl() . "
}
```

Note: Try to avoid lazy loading in a loop unless you have some cache implementation to reduce the number of requests.

4.5 Recipes

You'll find usefull recipes in our Recipes

4.5.1 Recipes

How to check if you are connected to the API ?

There is not a dedicated method to test if you are actually connected to the API.

However if you want to test if the connection can be established you must perform a dummy request to the remote instance and check if the response is ok.

```
<?php
use PhraseanetSDK\EntityManager;
use PhraseanetSDK\Client;
use PhraseanetSDK\HttpAdapter\Guzzle as GuzzleAdapter;
use PhraseanetSDK\Exception\UnauthorizedException;

$HttpAdapter = GuzzleAdapter::create();
$HttpAdapter->setBaseUrl('http://url-to-phraseanet.net/');

$client = new Client($apikey, $apiSecret, $HttpAdapter);
$client->setAccessToken($token);

$em = new EntityManager($client);
```

```
$databoxRepository = $em->getRepository('Databox');

try {
  $databoxRepository->findAll();
} catch (UnauthorizedException $e) {
      // Connection is not valid, handle it
} catch (\Exception $e) {
      // Something else went wrong
}
```

Retrieve the last twenty Feed entries

This code retrieves the 20 latest feed entries and print some informations about it.

```
<?php
use PhraseanetSDK\EntityManager;
use PhraseanetSDK\Client;
use PhraseanetSDK\HttpAdapter\Guzzle as GuzzleAdapter;

$HttpAdapter = GuzzleAdapter::create();
$HttpAdapter->setBaseUrl('http://url-to-phraseanet.net/');

$client = new Client($apikey, $apiSecret, $HttpAdapter);
$client->setAccessToken($developerToken);

$em = new EntityManager($client);

$entries = $em->getRepository('Entry')->findInAggregatedFeed(0, 20);

foreach($entries as $entry) {
    $output = "======================\n";
    $output .= $entry->getAuthorName() . "\n";
    $output .= $entry->getTitle() . "\n";
    $output .= $entry->getSubTitle() . "\n";
    $output .= $entry->getCreatedOn()->format('d/m/Y H:i:s') . "\n";
}
```

Search for records

The following code search for records

```
<?php
use PhraseanetSDK\EntityManager;
use PhraseanetSDK\Client;
use PhraseanetSDK\HttpAdapter\Guzzle as GuzzleAdapter;

$HttpAdapter = GuzzleAdapter::create();
$HttpAdapter->setBaseUrl('http://url-to-phraseanet.net/');

$client = new Client($apikey, $apiSecret, $HttpAdapter);
$client->setAccessToken($token);
$_SESSION['token_phrasea'] = $token;


$em = new EntityManager($client);
```

```
$recordRepository = $em->getRepository('Record');

$query = $recordRepository->search(array(
    'query' => 'animals',
    'offset_start' => 0,
    'per_page' => 20,
    'bases' => array(1, 4),
    'record_type' => 'image'
));

echo $query->getTotalResults() . " items found in " . $query->getQueryTime() . " seconds\n";

foreach($query->getResults() as $record) {
    $output = "======================\n";
    $output .= $record->getTitle() . "\n";
    $output .= $record->getOriginalName() . "\n";
}
```

Note: See documentation for possible query parameters https://docs.phraseanet.com/en/Devel/

Retrieve all validation basket

```
<?php
use PhraseanetSDK\EntityManager;

$em = new EntityManager($myClient);

$basketRepository = $em->getRepository('Basket');

$baskets = $basketRepository->findAll();

foreach($query->getResults()->filter(function($basket){
    return $baket->isValidationBasket();
}) as $basket) {
    $output = "======================\n";
    $output .= $basket->getName() . "\n";
    $output .= $record->getDescription() . "\n";
}
```

Note: ArrayCollection object provides many useful function take a look Doctrine\Common\Collections\ArrayCollection

oAuth2 Authentication Flow

How to get a token from the API ?

Phraseanet API only supports 'Token Grant Type'.

With this grant type you redirect the user to an authorization page on Phraseanet, and your script is called back once the end-user authorized your API key to access the Phraseanet service on its behalf.

Authorization page

```php
<?php

$client->setGrantType(Client::GRANT_TYPE_AUTHORIZATION, array('redirect_uri' => 'YOUR_REDIRECT_URI'));

// output the authentication url to the end user
echo $client->getAuthorizationUrl();
```

Note: In case your authorization page is the same that your callback page

Callback page

```php
<?php
use Symfony\Component\HttpFoundation\Request;
use PhraseanetSDK\Exception\AuthenticationException;
use PhraseanetSDK\Exception\RuntimeException;

$request = Request::createFromGlobals();

// retrieve the access token from current request
try {
    $client->retrieveAccessToken($request);
} catch (AuthenticationException $e) {
    // Something went wrong during the authentication flow
} catch (RuntimeException $e) {
    // Something went wrong for obscur reasons during the retrieval of the token
}
```

Note: ACCESS_TOKEN does not expire. So once you have an ACCESS_TOKEN associated to your current user, you can manage user's token with your own storage system on top of the library or you can just extends the PhraseanetSDKClient object an override the *getAccessToken* and *setAccessToken* method. See the next example to store token.

Store clients token in session

In some case you would probably store clients token in the session or database. SDK provide a StoreInterface for that : Let's store our token in `session`.

```php
<?php

namespace Acme\Application\Phrasea

use PhraseanetSDK\Authentication\StoreInterface;

class SessionStore implements StoreInterface
{
    protected $token;

    public function __construct()
    {
        $this->initSession();
    }

    public function saveToken($token)
```

```
    {
        $this->token = $token;
    }

    public function getToken()
    {
        return $this->token;
    }

    private function initSession()
    {
        if ( ! session_id()) {
            session_start();
        }

        $this->token = &$_SESSION['phrasea_oauth_token'];
    }
}
```

Usage

```
<?php

use Acme\Application\Phrasea\SessionStore;
use PhraseanetSDK\HttpAdapter\Guzzle as GuzzleAdapter;
use PhraseanetSDK\Client;

$HttpAdapter = GuzzleAdapter::create();
$HttpAdapter->setBaseUrl('http://url-to-phraseanet.net/');

$client = new Client('Your API Key', 'Your API Secret', $HttpAdapter);
$client->setTokenStore(new SessionStore());

if(null !== $client->getAccessToken()) {
    //user is still authenticated
} else {
    //force user to authenticate by providing the clicking authorization url
    echo $client->getAuthorizationUrl();
}
```

4.6 Handling Exceptions

The PHP SDK throws 3 different types of exception :

- `PhraseanetSDK\Exception\Runtime` is thrown when something went wrong most of the time you can get more informations by getting the previous exception.
- `PhraseanetSDK\Exception\UnauthorizedException` which is thrown when request is not authenticated
- `PhraseanetSDK\Exception\NotFoundException` which is thrown when the requested ressource can not be found

All these Exception implements `\PhraseanetSDK\Exception\ExceptionInterface` so you can catch any of these exceptions by catching this exception interface.

CHAPTER 5

Report a bug

If you experience an issue, please report it in our issue tracker. Before reporting an issue, please be sure that it is not already reported by browsing open issues.

When reporting, please give us information to reproduce it by giving your platform (Linux / MacOS / Windows) and its version, the version of PHP you use (the output of `php --version`)

CHAPTER 6

Ask for a feature

We would be glad you ask for a feature ! Feel free to add a feature request in the issues manager on GitHub !

CHAPTER 7

Contribute

You find a bug and resolved it ? You added a feature and want to share ? You found a typo in this doc and fixed it ? Feel free to send a Pull Request on GitHub, we will be glad to merge your code.

CHAPTER 8

Run tests

Phraseanet-PHP-SDK relies on PHPUnit for unit tests. To run tests on your system, ensure you have PHPUnit installed, and, at the root of the project, execute it :

```
phpunit
```

CHAPTER 9

About

Phraseanet-PHP-SDK has been written by the Alchemy Dev Team for Phraseanet, our DAM software. Try it, it's awesome !

CHAPTER 10

License

Phraseanet-PHP-SDK is licensed under the MIT License