ShopAppstoreLib - PHP Documentation

Release 0.1

DreamCommerce SA

January 26, 2016

Contents

This is an API/functional documentation of shop-appstore-lib. Here, you'll find an useful information on installation and usage of this library.

Contents:

CHAPTER 1

Library

In order to create a shop application in easy way, we created a SDK, which provides functions allowing payment handling and communications with resources.

1.1 Configuration

Library needs these parameters to be configured:

- Shop URL - each REST-ful API request needs to specify an entry point - a shop URL
- Application ID
- Secret

The configuration is done by specifying values in client object constructor:

```
$client = new \DreamCommerce\Client(
    'https://myshop.example.com', 'application ID', 'secret'
);
```

1.2 Debugging

SDK raises errors using exceptions:

- *DreamCommerce\Exception\ClientException* - informs about errors occurred within client library, eg. invalid token or credentials
- *DreamCommerce\Exception\ResourceException* - connected with particular resources, eg. invalid parameters

1.2.1 Debug mode

SDK allows to activate debug mode. Its purpose is to log all requests, responses and headers.

The debugging may be enabled by defining `DREAMCOMMERCE_DEBUG` constant.

value	meaning
false	debug mode is disabled
true	debug mode is enabled

You can define the constant in your source code:

```
define('DREAMCOMMERCE_DEBUG', true);
```

1.2.2 Logging messages

SDK allows to log all messages to stream.

The log file may be set by defining `DREAMCOMMERCE_LOG_FILE` constant.

value	meaning
false	logging is disabled
string	file path or stream (i.e. `"php://stdout"`, `"logs/application.log"`)

You can define the constant in your source code:

```
define('DREAMCOMMERCE_LOG_FILE', "php://stdout");
```

Messages can be passed to simple logger class using multiple priorities:

```
\DreamCommerce\Logger::debug("debug message");
\DreamCommerce\Logger::info("informational message");
\DreamCommerce\Logger::notice("notice message");
\DreamCommerce\Logger::warning("warning message");
\DreamCommerce\Logger::error("error message");
\DreamCommerce\Logger::critical("critical message");
\DreamCommerce\Logger::alert("alert message");
\DreamCommerce\Logger::emergency("emergency message");
```

Debug messages are filtered if debug mode is disabled.

1.2.3 Catching exceptions

A code example using exceptions handling:

```
try{
    $client = new \DreamCommerce\Client(
        'https://myshop.example.com', 'Application ID', 'Secret'
    );

    $client->setAccessToken('SHOP TOKEN');

    // fetch collection/object
    $product = new \DreamCommerce\Resource\Product($client);
    $list = $product->get();

    foreach($list as $item){
        //...
    }

} catch (\DreamCommerce\Exception\ClientException $ex) {
    // client error
    \DreamCommerce\Logger::error($ex);
} catch (\DreamCommerce\Exception\ResourceException $ex) {
    // resource error
    \DreamCommerce\Logger::error($ex);
}
```

Each exception lets to access an exception of lower layer, eg. HTTP response. Simply use standard exception's method `getPrevious` on every exception.

```
try{

    // ...

} catch (\DreamCommerce\Exception\ClientException $ex) {
    \DreamCommerce\Logger::error(sprintf("Client error: %s", $ex->getMessage()));

    $prev = $ex->getPrevious();

    if($prev instanceof \DreamCommerce\Exception\HttpException){
        \DreamCommerce\Logger::error(sprintf("HTTP error: %s", $prev->getMessage()));

        if($prev->getCode() == \DreamCommerce\Exception\HttpException::QUOTA_EXCEEDED){
            \DreamCommerce\Logger::warning("Quota exceeded");
        }
    }

} catch (\DreamCommerce\Exception\ResourceException $ex) {
    \DreamCommerce\Logger::error(sprintf("Resource error: %s", $ex->getMessage()));
}
```

In order to directly access error message, it's possible to use static method *DreamCommerce\Client::getError*.

```
try{

    // ...

} catch (\DreamCommerce\Exception\ClientException $ex) {
    \DreamCommerce\Logger::error(sprintf("Client error: %s", Client::getError($ex)));
} catch (\DreamCommerce\Exception\ResourceException $ex) {
    \DreamCommerce\Logger::error(sprintf("Resource error: %s", Client::getError($ex)));
}
```

1.3 Examples

1.3.1 Configuration

```
return array(
    /*
     * Application ID generated by AppStore
     */
    'appId' => '',

    /*
     * Secret generated by AppStore
     */
    'appSecret' => '',

    /*
     * AppStore Secret generated by AppStore
     */
    'appstoreSecret' => '',
```

```
    /*
     * Enable debug mode or not
     */
    'debug' => false,

    /*
     * Path to log file or empty to disable logging
     */
    'logFile' => "logs/application.log",


    /*
     * timezone of the application
     *
     * Value is passed to date_default_timezone_set function
     */
    'timezone' => 'Europe/Warsaw',

    'php' => array(
        /*
         * This determines whether errors should be printed to the screen as
         * part of the output or if they should be hidden from the user
         *
         * Value is passed to ini_set function
         */
        'display_errors' => 'off'
    )
);
```

1.3.2 Bootstrapping

```
// force utf-8 as primary encoding
if (PHP_VERSION_ID < 50600) {
    mb_internal_encoding('utf-8');
} else {
    ini_set('default_charset', 'utf-8');
}

// internal autoloader
spl_autoload_register(function($class){
    $class = str_replace('\\', '/', $class);
    require 'src/'.$class.'.php';
});

// composer autoloader - you can enable it on by uncommenting this line:
//require 'vendor/autoload.php';

// use config from previous example
$config = require __DIR__. '/Config.php';

// various PHP configuration values
date_default_timezone_set($config['timezone']);
ini_set('display_errors', $config['php']['display_errors']);

// check debug mode options
$debug = false;
```

```
if(isset($config['debug'])){
    if($config['debug']){
        $debug = true;
    }
}
// check environment variable
if(getenv('DREAMCOMMERCE_DEBUG')){
    $debug = true;
}
define("DREAMCOMMERCE_DEBUG", $debug);

// log errors to stdout by default
$logFile = "php://stdout";
if(isset($config['logFile'])){
    if($config['logFile']){
        $logFile = $config['logFile'];
    }else{
        $config['logFile'] = false;
    }
}
define("DREAMCOMMERCE_LOG_FILE", $logFile);

return $config;
```

1.3.3 REST GET

```
use DreamCommerce\Client;
use DreamCommerce\Exception\ClientException;
use DreamCommerce\Exception\ResourceException;

$config = require 'bootstrap.php';

try {
    // set custom retries count
    // it will throw HttpException if the limit is too low
    \DreamCommerce\Http::setRetryLimit(2);
    $client = new Client(
        'https://myshop.example.com', $config['appId'], $config['appSecret']
    );
    $client->setAccessToken('INSERT TOKEN HERE');

    $resource = new \DreamCommerce\Resource\Product($client);
    // or
    $resource = $client->products;

    // particular object, with ID=1
    $result = $resource->get(1);

    // list of objects
    $result = $resource->get();

    // list of objects (page 3) with filtering/limiting:
    $result = $resource->filters(array('translations.name' => array('=', 'laptop')))->page(3)->limit

    printf("Found: %d\n", $result->count);
    printf("Page: %d of %d\n", $result->page, $result->pages);
```

```
    printf("Iterating over products:\n");
    foreach ($result as $i) {
        printf("ID #%d\n", $i->product_id);
        // or - for your convenience:
        //printf("ID #%d\n", $i['product_id']);
    }
} catch (ClientException $ex) {
    \DreamCommerce\Logger::error("An error occurred during the Client initialization: ".Client::getEr
} catch (ResourceException $ex) {
    \DreamCommerce\Logger::error("An error occurred during Resource access: ".Client::getError($ex));
}
```

1.3.4 REST POST

```
use DreamCommerce\Client;
use DreamCommerce\Exception\ClientException;
use DreamCommerce\Exception\ResourceException;

$config = require 'bootstrap.php';

try {
    // set custom retries count
    // it will throw HttpException if the limit is too low
    \DreamCommerce\Http::setRetryLimit(2);
    $client = new Client(
        'https://myshop.example.com', $config['appId'], $config['appSecret']
    );
    $client->setAccessToken('INSERT TOKEN HERE');

    $resource = new \DreamCommerce\Resource\Producer($client);
    // or
    $resource = $client->producers;

    $insertedId = $resource->post(array(
        'name' => 'Awesome Manufacturer!',
        'web' => 'http://example.org'
    ));

    // or:
    $data = new stdClass();
    $data->name = 'Awesome Manufacturer!';
    $data->web = 'http://example.org';
    $insertedId = $resource->post($data);


} catch (ClientException $ex) {
    \DreamCommerce\Logger::error("An error occurred during the Client initialization: ".Client::getEr
} catch (ResourceException $ex) {
    \DreamCommerce\Logger::error("An error occurred during Resource access: ".Client::getError($ex));
}
```

1.3.5 REST PUT

```
use DreamCommerce\Client;
use DreamCommerce\Exception\ClientException;
```

```
use DreamCommerce\Exception\ResourceException;

$config = require 'bootstrap.php';

try {
    // set custom retries count
    // it will throw HttpException if the limit is too low
    \DreamCommerce\Http::setRetryLimit(2);
    $client = new Client(
        'https://myshop.example.com', $config['appId'], $config['appSecret']
    );
    $client->setAccessToken('INSERT TOKEN HERE');

    $resource = new \DreamCommerce\Resource\Producer($client);
    // or
    $resource = $client->producers;

    $insertedId = $resource->put(2, array(
        'name' => 'Awesome Manufacturer!'
    ));

    \DreamCommerce\Logger::info("Object modified");

} catch (ClientException $ex) {
    \DreamCommerce\Logger::error("An error occurred during the Client initialization: ".Client::getEr
} catch (ResourceException $ex) {
    \DreamCommerce\Logger::error("An error occurred during Resource access: ".Client::getError($ex));
}
```

1.3.6 REST DELETE

```
use DreamCommerce\Client;
use DreamCommerce\Exception\ClientException;
use DreamCommerce\Exception\ResourceException;

$config = require 'bootstrap.php';

try {
    // set custom retries count
    // it will throw HttpException if the limit is too low
    \DreamCommerce\Http::setRetryLimit(2);
    $client = new Client(
        'https://myshop.example.com', $config['appId'], $config['appSecret']
    );
    $client->setAccessToken('INSERT TOKEN HERE');

    $resource = new \DreamCommerce\Resource\Producer($client);
    // or
    $resource = $client->producers;

    $result = $resource->delete(41);
    \DreamCommerce\Logger::info("An object was successfully deleted");

} catch (ClientException $ex) {
    \DreamCommerce\Logger::error("An error occurred during the Client initialization: ".Client::getEr
} catch (ResourceException $ex) {
```

```
    \DreamCommerce\Logger::error("An error occurred during Resource access: ".Client::getError($ex));
}
```

1.3.7 Token refreshing

```
use DreamCommerce\Exception\ClientException;

$config = require 'bootstrap.php';
try {
    $c = new \DreamCommerce\Client('https://myshop.example.com', $config['appId'], $config['appSecret
    $token = $c->refreshToken('INSERT TOKEN HERE');

    \DreamCommerce\Logger::info("Token has been successfully refreshed");
} catch (ClientException $ex) {
    \DreamCommerce\Logger::error("An error occurred during the Client request: ".$ex->getMessage());
}
```

1.4 Installation

1.4.1 requirements

This library requires following aspects of interpreter to be satisfied:

- PHP 5.3 or higher
- SSL support enabled
- `allow_url_fopen` flag set to `on`
- compiled JSON extension (in the most of cases, available out-of-the-box)

1.4.2 installation

There are two ways to get and install the SDK into application:

- using Composer - in project directory call `composer require dreamcommerce/shop-appstore-lib`
- manual archive extraction - download an archive and extract its contents into project directory

1.5 Typical tasks

1.5.1 Handling billing system events

```
$handler = $this->handler = new Handler(
    'https://myshop.example.com', 'application ID', 'Secret', 'AppStore Secret'
);

$handler->subscribe('install', 'installHandler');   // function name
// $handler->subscribe('install', $installHandler);   // lambda
// $handler->subscribe('install', array($this, 'installHandler'));   // object method

//$handler->subscribe('billing_install', array($this, 'billingInstallHandler'));
```

```
//$handler->subscribe('billing_subscription', array($this, 'billingSubscriptionHandler'));
//$handler->subscribe('uninstall', array($this, 'uninstallHandler'));
```

Passed callback will be executed as an action handler.

1.5.2 Getting OAuth token

Token exchange is performed with the authorization key got during the application install.

The most convenient way is to exchange this code during the install. In billing system entry point it's enough to use:

```
$client = new \DreamCommerce\Client(
    'https://myshop.example.com', 'application ID', 'Secret'
);

$token = $client->getToken('AUTH CODE');

// $token is an object with access_token, refresh_token, expires_in
```

The best is to store gathered tokens into the database - *access_token* is required every time an application gets access to the shop.

1.5.3 Refreshing the token

In case the token gets expired (look at: `expires_in`) or in case it's invalidated, it's possible to refresh it:

```
$client = new \DreamCommerce\Client(
    'https://myshop.example.com', 'Application ID', 'Secret'
);

$token = $client->refreshToken('REFRESH TOKEN');

// $token is an object with access_token, refresh_token, expires_in
```

This object has equal information to the example above.

1.5.4 Performing REST-ful request

With a valid token, it's possible to perform request to the shop according to the API documentation:

```
$client = new \DreamCommerce\Client(
    'https://myshop.example.com', 'Application ID', 'Secret'
);

$client->setAccessToken('SHOP TOKEN');

// getting collection/object
$product = new \DreamCommerce\Resource\Product($client);
$list = $product->get();

foreach($list as $item){
    //...
}

// object update
```

```
$product->put(ID, array(...));

// create object
$product->post(array(...));

// delete object
$product->delete(ID);
```

CHAPTER 2

Classes list

2.1 Client

class `DreamCommerce\Client`

A client library allowing to perform REST-ful requests.

2.1.1 static methods

static `DreamCommerce\Client::getError` (*Exception $exception*)

Returns an error message of specified exception, layer-independent.

Parameters

- **$exception** (*Exception*) – an exception which is a source for error message

Return type string

Returns error message

2.1.2 methods

`DreamCommerce\Client::__construct` (*$entrypoint*, *$clientId*, *$clientSecret*)

constructor

Parameters

- **$exception** (*string*) – in case of webapi/rest is not a part of URL, it will be automatically appended
- **$clientId** (*string*) – string application ID
- **$clientSecret** (*string*) – application secret code (generated upon App Store application registration)

`DreamCommerce\Client::__get` (*$resource*)

Returns resource object by name

Parameters

- **$resource** (*string*) – resource name

Return type resource

`DreamCommerce\Client::`**`getToken`**`(`*$authCode*`)`

Gets token using the AuthCode.

Parameters

- **`$authCode`** (*`string`*) – authorization code, passed during installation

Return type array

Returns an array with tokens

`DreamCommerce\Client::`**`refreshToken`**`(`*$refreshToken*`)`

Refreshes token

Parameters

- **`$refreshToken`** (*`string`*) – refresh token, passed during token getting/exchange

Return type array

Returns an array with new tokens

`DreamCommerce\Client::`**`request`**`(`*$res*, *$method*, *$objectPath = null*, *$data =*[ ], *$query =*[ ]`)`

Performs a request to the server

Parameters

- **`$res`** (*`Resource`*) – resource to perform request on
- **`$method`** (*`string`*) – `GET/POST/PUT/DELETE`
- **`$objectPath`** (*`null|string|array`*) – resource path, eg. `object/1/something`. It can be also an array - then class will automatically glue it with `/`.
- **`$data`** (*`array`*) – data to pass with request
- **`$query`** (*`array`*) – query string

Returns request response

Return type mixed

`DreamCommerce\Client::`**`setAccessToken`**`(`*$token*`)`

Sets an access token for current script execution. Called automatically upon exchange/refreshing of token.

Parameters

- **`$token`** (*`string`*) – token

2.2 Exception

class `DreamCommerce\`**`Exception`**

2.2.1 ClientException

class `DreamCommerce\Exception\`**`ClientException`**

An exception raised upon `DreamCommerce\Client` library error.

constants

API_ERROR an server API error occured - check error message

ENTRYPOINT_URL_INVALID invalid shop URL

METHOD_NOT_SUPPORTED tried to perform an unsupported method (other than `GET/POST/PUT/DELETE`)

UNKNOWN_ERROR unknown error

2.2.2 HandlerException

class `DreamCommerce\Exception\`**`HandlerException`**

An exception raised upon billing system responder error.

constants

ACTION_HANDLER_NOT_EXISTS action handler doesn't exist

ACTION_NOT_EXISTS tried to handle non-existing action

ACTION_NOT_SPECIFIED an action to handle has not been specified

CLIENT_INITIALIZATION_FAILED an error occurred upon client class initialization - probably invalid data/tokens provided

HASH_FAILED packet checksum is invalid

INCORRECT_HANDLER_SPECIFIED handler for action is invalid

PAYLOAD_EMPTY server returned a response with no data

2.2.3 HttpException

class `DreamCommerce\Exception\`**`HttpException`**

An exception raised upon problem in communication layer (HTTP protocol error).

constants

LIMIT_TOO_LOW specified request limit is too low

METHOD_NOT_SUPPORTED tried to perform an unsupported HTTP method

QUOTA_EXCEEDED requests quota exceeded

REQUEST_FAILED a request failed (eg. I/O)

methods

`DreamCommerce\Exception\HttpException::`**`__construct`**([*$message* = ''[, *$code* = *0*[, *$previous* = *null*[, *$headers* = *array()*[, *$response* = '']]]]])

Method instantiates exception.

Parameters

- **$message** (*string*) – exception message
- **$code** (*integer*) – error code
- **$previous** (*Exception|null*) – a handle to the previous exception
- **$headers** (*array*) – an array with response headers

DreamCommerce\Exception\HttpException::**getHeaders**()

Returns an array with headers returned upon exception raising.

Return type array

Returns headers

DreamCommerce\Exception\HttpException::**getResponse**()

Returns a raw server response that caused an exception.

Return type string

Returns response

2.2.4 ResourceException

class DreamCommerce\Exception**ResourceException**

An exception raised upon problems with data manipulation, eg. invalid parameters.

constants

MALFORMED_RESPONSE server response is not parseable

CLIENT_ERROR client error library occurred (you can reach client exception using `getPrevious()` method)

LIMIT_BEYOND_RANGE a limit of maximum simultaneous connections is incorrectly specified

FILTERS_NOT_SPECIFIED empty filters specified

ORDER_NOT_SUPPORTED tried to sort data by non-existing/not supported field

INVALID_PAGE specified results page is beyond the pages count

2.3 Handler

class DreamCommerce**Handler**

It's an object for handling automatic messages from AppStore. It's event-based which allows to easily bind many handlers to the particular event (eg. for installation handler, it's possible to bind database storing, cache purging, etc).

2.3.1 methods

DreamCommerce\Handler::**__construct**(*$entrypoint*, *$clientId*, *$secret*, *$appStoreSecret*)

Class constructor

Parameters

- **$entrypoint** (*string*) – shop URL - in case of webapi/rest is not a part of URL, it will be automatically appended

- **$clientId** (*string*) – application ID
- **$secret** (*string*) – API key
- **$appStoreSecret** (*string*) – secret code

`DreamCommerce\Handler::`**`subscribe`**`(`*$event*, *$handler*`)`
Subscribes a handler to the chosen event.

Parameters

- **$event** (*string*) – event type for handling by handler
- **$handler** (*Callable*) – a handler to call when subscribed event is fired. `$handler` will be called with one argument of event params. If handler returns false value, event propagation is stopped.

Available `$event` values:

- `install` - an application is being installed in shop
- `uninstall` - an application is being uninstalled
- `billing_install` - installation payment
- `billing_subscription` - subscription payment

`DreamCommerce\Handler::`**`unsubscribe`**`(`*$event*, *$handler = null*`)`
Unsubscribes from event handling.

Parameters

- **$event** (*string*) – event type for handling by handler
- **$handler** (*null|Callable*) – if is `null`, all handlers are unbound. Specifying particular handler, leaves alone all except chosen.

Available `$event` values:

- `install` - an application is being installed in shop
- `uninstall` - an application is being uninstalled
- `billing_install` - installation payment
- `billing_subscription` - subscription payment

2.4 Http

class `DreamCommerce\`**`Http`**

A class performing HTTP requests.

Each of implemented methods returns following data set:

```
[
    'headers' => [
        'Content-Type' => 'application/json'
    ],
    'data' => <\ArrayObject|string>
]
```

2.4.1 static methods

static `DreamCommerce\Http::`**`instance`**

Returns a class instance

Returns class instance

Return type Http

static `DreamCommerce\Http::`**`setRetryLimit`**(*$num*)

Sets retries count if requests quota is exceeded.

Parameters

- **`$num`** (*`integer`*) – retries limit

2.4.2 methods

`DreamCommerce\Http::`**`delete`**(*$url*[, *$query = array()*[, *$headers = array()*]])

Performs HTTP DELETE.

Parameters

- **`$url`** (*`string`*) – URL
- **`$query`** (*`array`*) – query parameters (URL query string, after question mark)
- **`$headers`** (*`array`*) – additional headers to sent within request

Return type array

Returns see: *structure*

`DreamCommerce\Http::`**`get`**(*$url*[, *$query = array()*[, *$headers = array()*]])

Performs HTTP GET.

Parameters

- **`$url`** (*`string`*) – URL
- **`$query`** (*`array`*) – query parameters (URL query string, after question mark)
- **`$headers`** (*`array`*) – additional headers to sent within request

Return type array

Returns see: *structure*

`DreamCommerce\Http::`**`post`**(*$url*[, *$body = array()*[, *$query = array()*[, *$headers = array()*]]])

Performs HTTP POST.

Parameters

- **`$url`** (*`string`*) – URL
- **`$body`** (*`string`*) – request body
- **`$query`** (*`array`*) – query parameters (URL query string, after question mark)
- **`$headers`** (*`array`*) – additional headers to sent within request

Return type mixed

Returns see: *structure*

`DreamCommerce\Http::put` (*$url*[, *$body = array()*[, *$query = array()*[, *$headers = array()*]]])
Performs HTTP PUT.

Parameters

- **$url** (*string*) – URL
- **$body** (*string*) – request body
- **$query** (*array*) – query parameters (URL query string, after question mark)
- **$headers** (*array*) – additional headers to sent within request

Return type mixed

Returns see: *structure*

2.5 Logger

class `DreamCommerce\Logger`

A class performing simple messages logging.

2.5.1 static methods

static `DreamCommerce\Logger::__callStatic` (*$name*, *$args*)
calls static log method using $name as priority name and $args[0] as message

Parameters

- **$name** (*string*) – magic method name used as priority name
- **$args** (*array*) – arguments passed to magic method, $args[0] is treated as log message

Messages can be passed to simple logger class using multiple priorities:

```
\DreamCommerce\Logger::debug("debug message");
\DreamCommerce\Logger::info("informational message");
\DreamCommerce\Logger::notice("notice message");
\DreamCommerce\Logger::warning("warning message");
\DreamCommerce\Logger::error("error message");
\DreamCommerce\Logger::critical("critical message");
\DreamCommerce\Logger::alert("alert message");
\DreamCommerce\Logger::emergency("emergency message");
```

2.5.2 methods

`DreamCommerce\Logger::log` (*$message*, *$lvl = self::DEBUG*)
Logs message to defined stream.

Parameters

- **$message** (*string*) – log message
- **$lvl** (*string*) – priority

The stream can be set by defining `DREAMCOMMERCE_LOG_FILE` constant

value	meaning
false	logging is disabled
string	file path or stream (i.e. `php://stdout`, `logs/application.log`)

You can define the constant in your source code:

```
define('DREAMCOMMERCE_LOG_FILE', "php://stdout");
```

By default, all the messages are added with debug priority. All those messages are by default filtered out, due to disabled debug mode. Debugging may be enabled by defining `DREAMCOMMERCE_DEBUG` constant.

value	meaning
false	debug mode is disabled
true	debug mode is enabled

You can define the constant in your source code:

```
define('DREAMCOMMERCE_DEBUG', true);
```

2.6 Resource

class `DreamCommerce\`**`Resource`**

Represents particular resource.

2.6.1 resources

AboutPage

Check: Resource.

AdditionalField

Check: Resource.

constants

`FIELD_TYPE_TEXT` text input

`FIELD_TYPE_CHECKBOX` checkbox field

`FIELD_TYPE_SELECT` select (drop down)

`FIELD_TYPE_FILE` file input

`FIELD_TYPE_HIDDEN` hidden field

`LOCATE_USER` place field in user context

`LOCATE_USER_ACCOUNT` place field in user account panel

`LOCATE_USER_REGISTRATION` place field in user registration

`LOCATE_ORDER` place field in order

LOCATE_ORDER_WITH_REGISTRATION place field when user is being registered upon order

LOCATE_ORDER_WITHOUT_REGISTRATION place field when user is not being registered upon order

LOCATE_ORDER_LOGGED_ON_USER place field when user is logged in upon order

LOCATE_CONTACT_FORM place field in contact form

ApplicationLock

Check: Resource.

ApplicationConfig

Check: Resource.

constants

CONFIG_BLOG_COMMENTS_ENABLE Enable blog comments

CONFIG_BLOG_COMMENTS_FOR_USERS Only registered users are allowed to post blog comments

CONFIG_BLOG_COMMENTS_MODERATION Is blog comments moderation enabled - boolean

CONFIG_BLOG_DOWNLOAD_FOR_USERS Allow to download blog attached files - boolean

CONFIG_BLOG_ITEMS_PER_PAGE Blog items per page count

CONFIG_COMMENT_ENABLE Enable product comments - boolean

CONFIG_COMMENT_FOR_USERS Only registered users are allowed to post comments - boolean

CONFIG_COMMENT_MODERATION Is comments moderation enabled - boolean

CONFIG_DEFAULT_CURRENCY_ID Default currency identifier - integer

CONFIG_DEFAULT_CURRENCY_NAME Default currency name - ISO 4217 (“PLN”) - string

CONFIG_DEFAULT_LANGUAGE_ID Default shop language identifier - integer

CONFIG_DEFAULT_LANGUAGE_NAME Default shop language name - language_REGION format (eg. “pl_PL”) - string

CONFIG_DIGITAL_PRODUCT_LINK_EXPIRATION_TIME Product link expiration time (days) - integer

CONFIG_DIGITAL_PRODUCT_NUMBER_OF_DOWNLOADS Maximum downloads number per user - integer

CONFIG_DIGITAL_PRODUCT_UNLOCKING_STATUS_ID Status identifier which sends email with files - integer

CONFIG_LOCALE_DEFAULT_WEIGHT Shop weight unit:

- KILOGRAM
- GRAM
- LBS

CONFIG_LOYALTY_ENABLE Is loyalty program enabled - boolean

CONFIG_PRODUCT_DEFAULTS_ACTIVE Is product active in default - boolean

CONFIG_PRODUCT_DEFAULTS_AVAILABILITY_ID Default availability identifier - integer

CONFIG_PRODUCT_DEFAULTS_CODE Default product code generating method, available values:

- 1 - no method
- 2 - following ID
- 3 - random

CONFIG_PRODUCT_DEFAULTS_DELIVERY_ID Default product delivery identifier - integer

CONFIG_PRODUCT_DEFAULTS_ORDER Default ordering factor value - integer

CONFIG_PRODUCT_DEFAULTS_STOCK Default stock value - integer

CONFIG_PRODUCT_DEFAULTS_TAX_ID Default tax value identifier - integer

CONFIG_PRODUCT_DEFAULTS_UNIT_ID Default measurement unit identifier - integer

CONFIG_PRODUCT_DEFAULTS_WARN_LEVEL Default stock value warning level - integer

CONFIG_PRODUCT_DEFAULTS_WEIGHT Default product weight - float

CONFIG_PRODUCT_NOTIFIES_ENABLE Notify on product availabilities - boolean

CONFIG_PRODUCT_SEARCH_ALL_TOKENS Only for product name search - require exact phrase

CONFIG_PRODUCT_SEARCH_CODE Only for product details search - include product code

CONFIG_PRODUCT_SEARCH_DESCRIPTION Only for product details search - include product description - boolean

CONFIG_PRODUCT_SEARCH_SHORT_DESCRIPTION Only for product details search - include product short description

CONFIG_PRODUCT_SEARCH_TYPE Product search mode:

- 1 - only in product name
- 2 - in product name and other details

CONFIG_REGISTRATION_CONFIRM Require registration confirmation - boolean

CONFIG_REGISTRATION_ENABLE Enable user registration - boolean

CONFIG_REGISTRATION_LOGIN_TO_SEE_PRICE Only signed in users see price - boolean

CONFIG_REGISTRATION_REQUIRE_ADDRESS New users address requirements:

- 0 - only email address and password
- 1 - full address details

CONFIG_SHIPPING_ADD_PAYMENT_COST_TO_FREE_SHIPPING Force payment price addition even if free shipping is present - boolean

CONFIG_SHIPPING_VOLUMETRIC_WEIGHT_ENABLE Enable volumetric weight - boolean

CONFIG_SHOPPING_ALLOW_OVERSELLING Allow to sell more products than stock value - boolean

CONFIG_SHOPPING_ALLOW_PRODUCT_DIFFERENT_CURRENCY Allow to set currency per product - boolean

CONFIG_SHOPPING_ALLOW_TO_BU_NOT_REG Allow buying without registration - boolean

CONFIG_SHOPPING_BASKET_ADDING Actions performed upon product adding, available values:

- 1 - refresh page and do not redirect to the basket
- 2 - refresh page and perform redirection to the basket
- 3 - do not refresh page, show confirmation message

CONFIG_SHOPPING_BESTSELLER_ALGORITHM Bestseller calculation algorithm:

- 1 - most orders count
- 2 - most orders amount

CONFIG_SHOPPING_BESTSELLER_DAYS Only for automatic mode - days count when product is marked as bestseller, available values:

- 7 - last 7 days
- 30 - last 30 days
- 90 - last 90 days
- 0 - lifetime

CONFIG_SHOPPING_BESTSELLER_MODE Marking as "bestseller" mode:

- 0 - manual
- 1 - automatic, based on users orders

CONFIG_SHOPPING_CONFIRM_ORDER Require order confirmation - boolean

CONFIG_SHOPPING_MIN_ORDER_VALUE Minimal order value - float

CONFIG_SHOPPING_MIN_PROD_QUANTITY Minimal product quantity - float

CONFIG_SHOPPING_NEWPRODUCTS_DAYS Automatic mode - number of days after product creation it will be marked as "new" - integer

CONFIG_SHOPPING_NEWPRODUCTS_MODE Products marking as "new" mode, available values:

- 0 - manual
- 1 - automatic, based on product creation date

CONFIG_SHOPPING_OFF Disable shopping - boolean

CONFIG_SHOPPING_ORDER_VIA_TOKEN Share order via link - boolean

CONFIG_SHOPPING_PARCEL_CREATE_STATUS_ID Order status after parcel is created - integer|null

CONFIG_SHOPPING_PARCEL_SEND_STATUS_ID Order status after parcel is sent - integer|null

CONFIG_SHOPPING_PRICE_COMPARISON_FIELD Field which products are identified by - for price comparison websites, available values:

- code - product code
- additional_isbn - ISBN code
- additional_kgo - KGO price
- additional_bloz7 - BLOZ7 code
- additional_bloz12 - BLOZ12 code

CONFIG_SHOPPING_PRICE_LEVELS Defined price levels (1-3) - integer

CONFIG_SHOPPING_PRODUCTS_ALLOW_ZERO Allow to buy zero-priced products - boolean

CONFIG_SHOPPING_SAVE_BASKET Save basket contents - boolean

CONFIG_SHOPPING_SHIPPING_EXTRA_STEP Show shipping and payment, available values:

- 0 - show in basket
- 1 - show as separated step

CONFIG_SHOPPING_UPDATE_STOCK_ON_BUY Update stock values on buy - boolean

CONFIG_SHOP_ADDRESS_1 Shop address line 1 - string

CONFIG_SHOP_ADDRESS_2 Shop address line 2 - string

CONFIG_SHOP_CITY Shop city - string

CONFIG_SHOP_COMPANY_NAME Company name - string

CONFIG_SHOP_COUNTRY Shop country code - string

CONFIG_SHOP_EMAIL Shop mail e-mail address - string

CONFIG_SHOP_FULL_ADDRESS Shop full address - string

CONFIG_SHOP_NAME Full shop name - string

CONFIG_SHOP_OFF Is shop disabled - boolean

CONFIG_SHOP_PHONE Shop phone number - string

CONFIG_SHOP_PROVINCE Shop province - string

CONFIG_SHOP_REGON Company identification number - integer

CONFIG_SHOP_TAX_ID Tax identifier - integer

CONFIG_SHOP_TRADE Shop trade - string

CONFIG_SHOP_TRADE_CODE Shop trade code, available values:

- children
- books_and_multimedia
- clothes
- computers
- delicatessen
- gifts_and_accessories
- health_and_beauty
- hobby
- home_and_garden
- automotive
- consumer_electronics
- sport_and_travel
- other

CONFIG_SHOP_URL Shop URL - string

CONFIG_SHOP_ZIP_CODE Shop postcode - string

ApplicationVersion

Check: Resource.

AttributeGroup

Check: Resource.

Attribute

Check: Resource.

constants

LOCATE_USER locate field in user context

LOCATE_USER_ACCOUNT locate field in user account panel

LOCATE_USER_REGISTRATION locate field in user registration form

LOCATE_ORDER locate field in order context

LOCATE_ORDER_WITH_REGISTRATION locate field in order for anonymous user requested account registration

LOCATE_ORDER_WITHOUT_REGISTRATION locate field in order for anonymous user

LOCATE_ORDER_LOGGED_ON_USER locate field in order for logged on user

LOCATE_CONTACT_FORM locate field in contact form

Auction

Check: Resource.

constants

SALES_FORMAT_BIDDING auction is bid-based

SALES_FORMAT_IMMEDIATE “buy now”

SALES_FORMAT_SHOP treat auction just like a shop

AuctionHouse

Check: Resource.

AuctionOrder

Check: Resource.

Availability

Check: Resource.

Category

Check: Resource.

CategoriesTree

Check: Resource.

Currency

Check: Resource.

DashboardActivity

Check: Resource.

DashboardStat

Check: Resource.

Delivery

Check: Resource.

Gauge

Check: Resource.

GeolocationCountry

Check: Resource.

GeolocationRegion

Check: Resource.

Language

Check: Resource.

Metafield

Check: Resource.

constants

TYPE_INT type of integer

TYPE_FLOAT type of float

TYPE_STRING type of string

TYPE_BLOB type of binary data

MetafieldValue

Check: Resource.

ObjectMtime

Check: Resource.

OptionGroup

Check: Resource.

Option

Check: Resource.

constants

HTTP_ERROR_OPTION_CAN_NOT_MODIFY_REQUIRE it's not possible to change required flag for option with warehouse support

HTTP_ERROR_OPTION_CAN_NOT_MODIFY_TYPE it's not possible to change type of an existing option

TYPE_FILE option type file input

TYPE_TEXT option type text

TYPE_RADIO option type radio option

TYPE_SELECT option type select (drop down)

TYPE_CHECKBOX option type checkbox

TYPE_COLOR option type color

PRICE_TYPE_DECREASE option value decreases price

PRICE_TYPE_KEEP option value is unchanged

PRICE_TYPE_INCREASE option value increases price

PRICE_PERCENT option value is modified by percent

PRICE_AMOUNT option value is modified by value

OptionValue

Check: Resource.

constants

HTTP_ERROR_OPTION_CHILDREN_NOT_SUPPORTED option type doesn't support values management

PRICE_TYPE_DECREASE option value decreases price

PRICE_TYPE_KEEP option value keeps price unchanged

PRICE_TYPE_INCREASE option value increases price

PRICE_PERCENT price change by percent

PRICE_AMOUNT price change by value

OrderProduct

Check: Resource.

Order

Check: Resource.

constants

HTTP_ERROR_ORDER_COMBINED combined order has been detected

ORIGIN_SHOP order comes from shop

ORIGIN_FACEBOOK order comes from Facebook

ORIGIN_MOBILE order comes from mobile

ORIGIN_ALLEGRO order comes from Allegro

ORIGIN_WEBAPI order comes from WebAPI

FIELD_TYPE_TEXT order field type is text

FIELD_TYPE_CHECKBOX order field type is checkbox

FIELD_TYPE_SELECT order field type is select (drop down)

FIELD_SHOW_ORDER place field in order

FIELD_SHOW_REGISTERED place field if user is being registered

FIELD_SHOW_GUEST place field if user is not registered

FIELD_SHOW_SIGNED_IN place field if user is signed in

ADDRESS_TYPE_BILLING address is for billing purposes

ADDRESS_TYPE_DELIVERY address is for delivery purposes

Parcel

Check: Resource.

constants

HTTP_ERROR_PARCEL_CAN_NOT_MODIFY It's not possibly to modify shipped parcel except shipping code

HTTP_ERROR_PARCEL_IS_ALREADY_SENT Parcel has been already sent

ADDRESS_TYPE_BILLING address used for billing purposes

ADDRESS_TYPE_DELIVERY address used for delivery purposes

Payment

Check: Resource.

Producer

Check: Resource.

Product

Check: Resource.

ProductFile

Check: Resource.

ProductImage

Check: Resource.

ProductStock

Check: Resource.

constants

PRICE_TYPE_KEEP keep base price

PRICE_TYPE_NEW specify price for stock

PRICE_TYPE_INCREASE increase base price

PRICE_TYPE_DECREASE decrease base price

WEIGHT_TYPE_KEEP keep base weight

WEIGHT_TYPE_NEW specify weight for stock

WEIGHT_TYPE_INCREASE increase base weight

WEIGHT_TYPE_DECREASE decrease base weight

Shipping

Check: Resource.

Status

Check: Resource.

constants

TYPE_NEW status: new

TYPE_OPENED status: opened

TYPE_CLOSED status: closed

TYPE_UNREALIZED status: not completed

SubscriberGroup

Check: Resource.

Subscriber

Check: Resource.

Tax

Check: Resource.

Unit

Check: Resource.

UserAddress

Check: Resource.

UserGroup

Check: Resource.

User

Check: Resource.

constants

ORIGIN_SHOP user created via shop

ORIGIN_FACEBOOK user created via Facebook

ORIGIN_MOBILE user created via mobile

ORIGIN_ALLEGRO user created via Allegro

FIELD_TYPE_TEXT text field

FIELD_TYPE_CHECKBOX checkbox

FIELD_TYPE_SELECT select (drop down)

FIELD_SHOW_USER show field for user

FIELD_SHOW_CLIENT show field for client

FIELD_SHOW_REGISTRATION show field during registration

Webhook

Check: Resource.

constants

FORMAT_JSON webhook data encoded using JSON

FORMAT_XML webhook data encoded using XML

EVENT_ORDER_CREATE webhook bound to order create event

EVENT_ORDER_EDIT webhook bound to order edit event

EVENT_ORDER_PAID webhook bound to order is paid event

EVENT_ORDER_STATUS webhook bound to order status change event

EVENT_ORDER_DELETE webhook bound to order delete event

EVENT_CLIENT_CREATE webhook bound to client create event

EVENT_CLIENT_EDIT webhook bound to client edit event

EVENT_CLIENT_DELETE webhook bound to client delete event

EVENT_PRODUCT_CREATE webhook bound to product create event

EVENT_PRODUCT_EDIT webhook bound to product edit event

EVENT_PRODUCT_DELETE webhook bound to product delete event

EVENT_PARCEL_CREATE webhook bound to parcel create event

EVENT_PARCEL_DISPATCH webhook bound to parcel dispatch event

EVENT_PARCEL_DELETE webhook bound to parcel delete event

Zone

Check: Resource.

constants

ZONE_MODE_COUNTRIES zone division by countries

ZONE_MODE_REGIONS zone division by regions

ZONE_MODE_CODES zone supports post codes

resource	description
AboutPage	About page

Continued on next page

Table 2.1 – continued from previous page

resource	description
AdditionalField	Additional field
ApplicationLock	Administrator panel lock
ApplicationConfig	Application config
ApplicationVersion	Application config
AttributeGroup	Attributes group
Attribute	Attribute
Auction	Auction
AuctionHouse	Auction house
AuctionOrder	Auction order
Availability	Product availability
Category	Category
CategoriesTree	Category tree
Currency	Currency
DashboardActivity	Dashboard activity
DashboardStat	Sales stats
Delivery	Delivery
Gauge	Gauge
GeolocationCountry	Geolocation country
GeolocationRegion	Geolocation region
Language	Language
Metafield	Metafield
MetafieldValue	Metafield Value
ObjectMtime	Modification timestamp of object
OptionGroup	Option group
Option	Variant
OptionValue	Variant value
OrderProduct	Product of order
Order	Order
Parcel	Parcel
Payment	Payment method
Producer	Producer
Product	Product
ProductFile	Product file
ProductImage	Photo of product
ProductStock	Product stock
Shipping	Shipping method
Status	Order status
SubscriberGroup	Subscriber group
Subscriber	Subscriber
Tax	Tax value
Unit	Unit of measurement
UserAddress	Shipping address
UserGroup	User group
User	User
Webhook	Webhook
Zone	Zone

2.6.2 static methods

static `DreamCommerce\Resource::factory`(*$client*, *$name*)

instantiates new Resource object

Parameters

- **`$client`** (*`Client`*) – handle to the client library instance
- **`$name`** (*`string`*) – name of resource

Return type Resource

```
\DreamCommerce\Resource::factory($client, "product");
```

2.6.3 methods

`DreamCommerce\Resource::delete`([*$id = null*])

Deletes an object by ID.

Parameters

- **`$id`** (*`integer`*) – object ID

Return type boolean

`DreamCommerce\Resource::filters`(*$filters*)

Filters records (GET only).

Chaining method - returns a handle to an object itself.

Parameters

- **`$filters`** (*`array`*) – an array of filters

Return type Resource

Returns chain

`DreamCommerce\Resource::get`([...])

Performs GET request.

Parameters

- **`..`** (*`mixed`*) – arguments - no argument: returns collection - single argument - an object with specified ID - more arguments - resource dependent

Return type ResourceList|ArrayObject|string

`DreamCommerce\Resource::getName`()

Returns a plural name of resource.

Return type string

`DreamCommerce\Resource::limit`(*$count*)

Restrict amount of fetched records of collection (GET only).

Chaining method - returns a handle to an object itself.

Parameters

- **`$count`** (*`integer`*) – count of records to fetch

Return type Resource

Returns chain

`DreamCommerce\Resource::`**`order`**(*$expr*)

Sorts records by specified criteria (GET only).

Chaining method - returns a handle to an object itself.

Parameters

- **`$expr`** (*`string`*) – sorting expression, eg. `field DESC`, `field ASC` or `+field`, `-field`

Return type Resource

Returns chain

`DreamCommerce\Resource::`**`page`**(*$page*)

Specifies results page number for fetching (GET only).

Chaining method - returns a handle to an object itself.

Parameters

- **`$page`** (*`integer`*) – number of page

Return type Resource

Returns chain

`DreamCommerce\Resource::`**`post`**([*$data = array()*])

Performs a POST request.

Chaining method - returns a handle to an object itself.

Parameters

- **`$data`** (*`array`*) – request body

Return type ArrayObject|string

`DreamCommerce\Resource::`**`put`**([*$id = null*[, *$data = array()*]])

Performs PUT request.

Chaining method - returns a handle to an object itself.

Parameters

- **`$id`** (*`null|integer`*) – ID of modified object
- **`$data`** (*`array`*) – request body

Return type ArrayObject|string

2.7 ResourceList

class `DreamCommerce\`**`ResourceList`**

Represents collection of resources, extends ArrayObject

2.7.1 methods

`DreamCommerce\ResourceList::`**`__construct`**(*$array = array(), $flags = ArrayObject::ARRAY_AS_PROPS*)

Creates a new class instance.

Parameters

- **`$array`** (*array*) – Objects in the collection
- **`$flags`** (*integer*) – flags for ArrayObject

Return type void

`DreamCommerce\ResourceList::`**`setCount`**`(`*$count*`)`
set number of all resources on requested list.

Parameters

- **`$count`** (*integer*) – number of all resources on the list

Return type void

`DreamCommerce\ResourceList::`**`getCount`**`()`
get number of all resources on requested list.

Return type integer

Returns number of objects

`DreamCommerce\ResourceList::`**`setPage`**`(`*$page*`)`
set current page number.

Parameters

- **`$page`** (*integer*) – current page number

Return type void

`DreamCommerce\ResourceList::`**`getPage`**`()`
get current page number.

Returns page number

Return type integer

`DreamCommerce\ResourceList::`**`setPageCount`**`(`*$count*`)`
set total page count.

Parameters

- **`$count`** (*integer*) – total page count

Return type void

`DreamCommerce\ResourceList::`**`getPageCount`**`()`
get total page count.

Return type integer

Returns pages count

CHAPTER 3

Application template

A template (boilerplate) allows to create an application in quickly way using the least amount of development.

It consists of a library and its sample implementation - billing system and main application.

3.1 Structure

The template allows to use simplified MVC pattern in order to separate particular application components.

Creating a new action step-by-step:

- To create new controller, create the class in `src/Controller/` (i.e. `src/Controller/Books.php`), that inherits from `ControllerAbstract`, and is defined in `Controller` namespace
- Create `myAction` method
- Create a view in directory `view` according to the schema: `controller-name/action-name.php` (i.e. `controller/my.php`)
- In order to pass a variable to the view, use following expression `$this['variable'] = 'value';` in action code. Then, this variable will be accessible like a regular variable within the view.

3.2 How to start

- Download the boilerplate
- *Install SDK*
- *Configure*

3.3 Install SDK

If you're using Composer to manage your packages and SDK, inside the project directory call `composer install`.

Alternatively, it's possible to manually install SDK. Just extract its contents into project directory.

3.4 Configure

In file `src/Config.php`:

```
return array(
    /*
     * Application ID generated by AppStore
     */
    'appId' => '',

    /*
     * Secret generated by AppStore
     */
    'appSecret' => '',

    /*
     * AppStore Secret generated by AppStore
     */
    'appstoreSecret' => '',


    'db' => array(
        /*
         * PDO DSN
         */
        'connection' => 'mysql:host=127.0.0.1;dbname=app',

        /*
         * PDO Database username
         */
        'user' => '',

        /*
         * PDO Database password
         */
        'pass' => ''
    ),

    /*
     * Enable debug mode or not
     */
    'debug' => false,

    /*
     * Path to log file or empty to disable logging
     */
    'logFile' => "logs/application.log",


    /*
     * timezone of the application
     *
     * Value is passed to date_default_timezone_set function
     */
    'timezone' => 'Europe/Warsaw',

    'php' => array(
        /*
```

```
         * This determines whether errors should be printed to the screen as
         * part of the output or if they should be hidden from the user
         *
         * Value is passed to ini_set function
         */
        'display_errors' => 'off'
    )
);
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

PHP Namespace Index

d

Index

Symbols

C

D

E

F

G

H

I

L

O

P

R

S

U