python-shop-sdk Documentation

Release 0.1

Arne Simon => email::[arne.simon@slice-dice.de]

March 28, 2014

CONTENTS

Contents:

CHAPTER

ONE

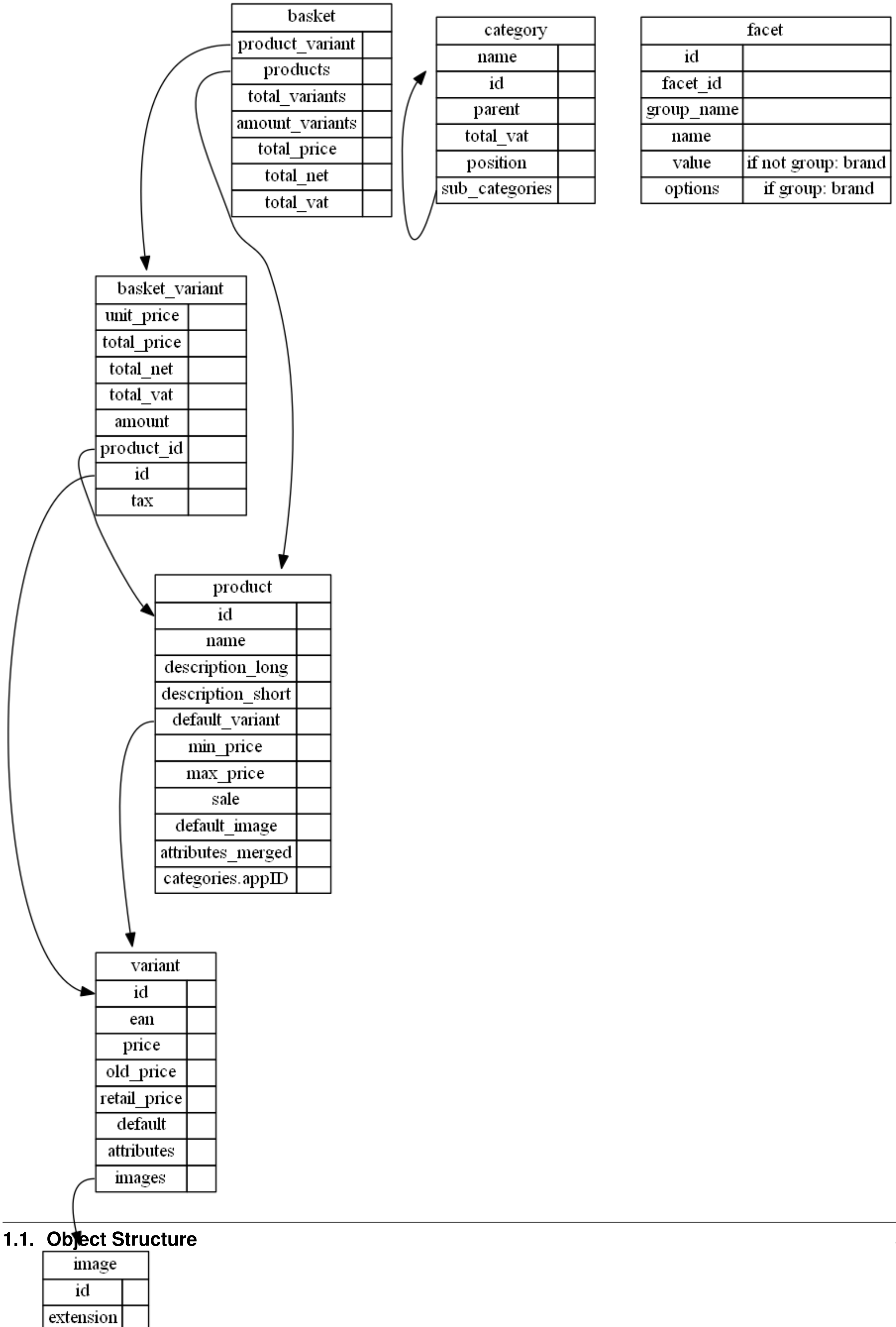
COLLINS

Author Arne Simon => email::[arne_simon@gmx.de]

This module provieds two wrappers around the Collins-Shop-API.

- A thin python wrapper which takes Python dict's and list's and returns the result as the same.
- EasyCollins, which is a more convient layer of abstraction of the API as an object herachie which caches results and query results if there are needed.

1.1 Object Structure



class `collins.Collins`(*config*)

Bases: `object`

An interface to the Collins API.

This is thin warper around the Collins API. All functions return the JSON responses as Python List and Dictonarys.

Parameters config – A Config instance or a file name to a JSON config file.

`autocomplete`(*searchword*, *limit=None*, *types=None*)

Parameters

- **searchword** (*str*) – The abbriviation.
- **categories** (*list*) – against which types should be autocompleted
- **limit** (*int*) – the amount of items returned per selected type

> **Attention:** In the documentation stands **autocomplete** but the real Tag is **autocompletion**!

`basketadd`(*sessionid*, *products*)

The request "basket add" does change the Items available in the basket. A set to 0 or less leads to a deletion. Only the variants in the items list will be changed. If a variant is changed or added an internal availability check is done and error codes are returned.

Parameters

- **sessionid** (*str*) – identification of the basket -> user, user -> basket
- **products** (*list*) – is the array for the product variants

product variant dict

fieldname	meaning
id	the product variant id
command	either add for the + operator (even -1 is possible) or set for a specific quantity. If the end amount is <= 0 the product is removed from basket with an result code.
amount	the amount to add or set

`basketget`(*sessionid*)

This returns the current basket for a User / Session ID. The basket belongs to a specific app id and session id, another app can have the same session id.

Parameters sessionid (*str*) – identification of the basket -> user, user -> basket

`category`(*ids*)

You are able to retrieve single categories.

Parameters ids (*list*) – List of category ids.

Example

```
collins.category([16077])
```

```
{
    "16077": {
        "active": true,
        "position": 1,
        "name": "Damen",
        "parent": 0,
        "id": 16077
    }
}
```

`categorytree`(*max_depth=None*)

The request category tree returns a tree of categories of a specified max depth for your app id.

Parameters **max_depth** (*int*) – max depth of your category tree counted from root

Example

```
collins.categorytree()
```

```
[
    {
        "name": "Damen",
        "parent": null,
        "sub_categories": [
            {
                "name": "Bekleidung",
                "parent": 16077,
                "sub_categories": [
                    {
                        "name": "Oberteile",
                        "parent": 16078,
                        "sub_categories": [
                            {
                                "name": "Fr\u00fchlingslooks",
                                "parent": 16079,
                                "sub_categories": [],
                                "active": false,
                                "position": 1,
                                "id": 23882
                            }
                        ]
                    }
                ]
            }
        ]
    }
]
```

`facets`(*group_ids=None*, *limit=None*, *offset=None*)

This returns a list of available facet groups or a facets of a group.

Parameters

- **group_ids** (*list*) – get only these group ids, if empty get all group ids which belong to me
- **limit** (*int*) – limit the per page items
- **offset** (*int*) – offset for paging through the items

Example

```
collins.facets([Constants.FACET_CUPSIZE])
```

```
{
    "facet": [
        {
            "id": 4,
            "group_name": "cupsize",
            "name": "D",
            "value": "D",
            "facet_id": 96
        },
        {
            "id": 4,
            "group_name": "cupsize",
            "name": "A",
            "value": "A",
            "facet_id": 93
        },
        {
            "id": 4,
            "group_name": "cupsize",
            "name": "B",
            "value": "B",
            "facet_id": 94
        },
        {
            "id": 4,
            "group_name": "cupsize",
            "name": "C",
            "value": "C",
            "facet_id": 95
        }
    ],
    "hits": 4
}
```

facettypes()

This query returns a list of facet groups available.

Example

```
collins.facettypes()
```

```
[0, 2, 1, 6, 172, 206, 173, 194, 175, 204, 5, 189, 180, 231, 187, 190, 211, 181, 247]
```

getorder(*orderid*)

Through this query you could get a order which was created for/through your app. This is limited to a configured timeframe and to your app.

Parameters **orderid** (*int*) – this is the order id to get info about

initiateorder(*sessionid*, *sucess_url*, *cancel_url=None*, *error_url=None*)

At this request you initiate a order to a basket. This should be done if a user wants to go to the checkout.

Parameters

- **sessionid** (*str*) – identification of the basket -> user, user -> basket (see basket_get, basket_add)
- **sucess_url** (*str*) – this is a callback url if the order was successfully created. (see checkout api)
- **cancel_url** (*str*) – this is a callback url if the order was canceled. (see checkout api)
- **error_url** (*str*) – this is a callback url if the order throwed exceptions (see checkout api)

livevariant (*ids*)

This does return the live information about the product variant. This is as "live" as possible. And could differ vs. a "product search" or "product" query.

Parameters **ids** (*list*) – array of product variant id

products (*ids*, *fields=None*)

Here you get a detail view of a product or a list of products returned by its ids.

Parameters **ids** (*list*) – array of product variant id

Example

```
collins.products([227838, 287677])
```

```
{
    "pageHash": "4ae38022-8ddd-4f15-9654-4ea0156c33f0",
    "ids": {
        "287677": {
            "active": true,
            "styles": [
                {
                    "active": true,
                    "id": 287685,
                    "style_key": "m01_233966406",
                    "name": "Badeshorts, Buffalo"
                },
                {
                    "active": false,
                    "id": 358863,
                    "style_key": "m01_233966406",
                    "name": "Badeshorts, Buffalo"
                }
            ],
            "id": 287677,
            "style_key": "m01_233966406",
            "name": "Badeshorts, Buffalo"
        },
        "227838": {
            "active": true,
            "styles": [
                {
                    "active": true,
                    "id": 227854,
                    "style_key": "m01_255897631",
                    "name": "Badeshort Herren"
                }
            ],
            "id": 227838,
            "style_key": "m01_255897631",
            "name": "Badeshort Herren"
        }
    }
}
```

productsearch(*sessionid*, *filter=None*, *result=None*)

This is the main query for retrieving products for your app. Lists of products will be returned which are filtered. In a search you dont get back inactive set products. The response can contain the available facets and their product count and categories for the filter set.

There are two types of facets:

1. **is a pre defined and always available set of facets**
   (a) price range (amazon like see)
   (b) sale facet how many products are sale and how many not
   (c) categories
2. dynamic facets which occur through the products itself. If a product has one the these attributes you are able to query for these attributes. These facets are always just ids. The name -> id, id -> name resolution works through facets call. To get all the facet groups you can filter or are in your result set see "Facet types"

Note: To get facet types see `collins.Constants`

Parameters

- **sessionid** (*str*) – the session_id of the frontend customer
- **filter** (*dict*) – object of filter information, these filters do change the subset of products
- **result** (*dict*) – object of result information, these properties change the order and the appearance of the subset

filter dict

fieldname	meaning
categories	array of category ids to include the products from
sale	true => only sale products false => no sale products null => both
prices	is an object with properties from and to which lead to range filter on the price set as eurocent
searchword	the word which filters every product
facets	object of attributes every key is an array of facet id values to filter by them

result dict

field-name	meaning
sort	how to sort the products
price	geta back a range of attributes and their product counts also some statistical information in eurocent.
sale	product count of products which are in sale
facets	object of which facet get back what count of facets
categories	set if get or not back the categories in which are products and the count of products
limit	how many products to get back
offset	the offset where to start to get the products from

result.facets dict

field-name	meaning
_all	is an object with property limit which holds an integer to get the amount of attributes, sort by occurrences in products if this is set, this is the default for ALL attributes.
[0-9]+	is an object with property limit which holds an integer to get the amount of attributes of this attribute group id, sort by occurrences in products. this overwrites even the _all field.

Example

```
# i belive we search shorts now o.O
collins.productsearch(TEST_SESSION_ID, filter={"categories":[16354]})
```

```
{
    "product_count": 78,
    "pageHash": "6e63cadb-5505-47b7-b36d-643311292a22",
    "facets": {},
    "products": [
        {
            "id": 227838,
            "name": "Badeshort Herren"
        }
    ]
}
```

send (*cmd*, *obj*)

Sends a Pyhton structure of dict's and list's as raw JSON to collins and returns a Python structure of dict's and list's from the JSON answer.

Parameters

- **cmd** – The name of the command.
- **obj** – A python dict object which contains the request parameters.

suggest (*searchword*, *categories=None*, *limit=None*)

The suggest endpoint does what it says and suggest more terms. It takes a searchterm and suggests terms which occurance is really often in combination with these terms before.

Parameters

- **searchword** (*str*) – string of a searchterm to suggest more words for
- **categories** (*list*) – array of category ids
- **limit** (*int*) – amount of items to suggest

exception `collins.CollinsException`

Bases: `exceptions.Exception`

An exception in the collins module.

class `collins.Config`(*filename*)

Bases: `object`

The configuration of a collins api connection.

A config class has to have the following readable attributes:

- **entry_point_url:** The url for collins.
- **app_id** The application id.
- **app_password** The password for the corresponding application id.
- **agent** The name of the browser agent to fake.
- **authorization** The authorization header
- **image_url** A string as template for the image urls. As example http://cdn.mary-paul.de/product_images/{path}/{id}_{width}_{height}{extension}.
- **logging** A dictonary for logging.config.dictConfig.

Parameters **filename** (*str*) – The path to a JSON file which holds the configuration.

`imageurl`(*path*, *productid*, *width*, *height*, *extension*)

Returns the url for a image.

Parameters

- **path** – Part of the image path.
- **productid** –
- **width** – Width of the image.
- **height** – Height of the image.
- **extension** – The image extension. For example *'.jpg'*

class `collins.Constants`

Bases: `object`

Some contsants which are blatantly copied from the php-sdk.

> **Attention:**
> **The following constants are NOT in the PHP-SDK:**
> - FACET_CHANNEL
> - FACET_CARE_SYMBOL
> - FACET_CLOTHING_HATS_US

`API_ENVIRONMENT_LIVE` = **'live'**

`API_ENVIRONMENT_STAGE` = **'stage'**

`FACETS` = **set([0, 1, 194, 3, 4, 5, 6, 172, 2, 204, 173, 174, 175, 206, 195, 180, 181, 187, 189, 190])**

FACET_BRAND = 0

FACET_CARE_SYMBOL = 247

FACET_CHANNEL = 211

FACET_CLOTHING_HATS_US = 231

FACET_CLOTHING_MEN_BELTS_CM = 190

FACET_CLOTHING_MEN_DE = 187

FACET_CLOTHING_MEN_INCH = 189

FACET_CLOTHING_UNISEX_INCH = 174

FACET_CLOTHING_UNISEX_INT = 173

FACET_CLOTHING_UNISEX_ONESIZE = 204

FACET_CLOTHING_WOMEN_BELTS_CM = 181

FACET_CLOTHING_WOMEN_DE = 175

FACET_CLOTHING_WOMEN_INCH = 180

FACET_COLOR = 1

FACET_CUPSIZE = 4

FACET_DIMENSION3 = 6

FACET_GENDERAGE = 3

FACET_LENGTH = 5

FACET_SHOES_UNISEX_ADIDAS_EUR = 195

FACET_SHOES_UNISEX_EUR = 194

FACET_SIZE = 2

FACET_SIZE_CODE = 206

FACET_SIZE_RUN = 172

SORTS = set([’relevance’, ‘updated_date’, ‘price’, ‘most_viewed’, ‘created_date’])

SORT_CREATED = ‘created_date’

SORT_MOST_VIEWED = ‘most_viewed’

SORT_PRICE = ‘price’

SORT_RELEVANCE = ‘relevance’

SORT_UPDATED = ‘updated_date’

TYPES = set([’products’, ‘categories’])

TYPE_CATEGORIES = ‘categories’

TYPE_PRODUCTS = ‘products’

class collins.**EasyCollins**(*conf_or_filename*)
Bases: object

collins.**VERSION = ‘0.0’**
Version of the python shop SDK.

`collins.`**`check_sessionid`**`(`*sessionid*`)`

> **Attention:** We copied it from the php-sdk. collins seems to want have the session-id a minimum length of five characters. This is not tested or validated.

CHAPTER

TWO

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX