django-cleverreach Documentation

Release 5.1.0

Simon Bächler

April 02, 2015

Contents

1 api Package **3**

1.1 `api` Package 3

1.2 `api.v5_1` Module 3

2 cleverreach Package **5**

2.1 `utils` Module 5

2.2 Subpackages 5

3 Installation **9**

4 Usage **11**

5 Indices and tables **13**

Python Module Index **15**

Contents:

CHAPTER 1

api Package

1.1 `api` Package

1.2 `api.v5_1` Module

You have to define a group (in the cleverreach admin panel) for each language. The form code is optional, we use the first form if it is not provided.

The Cleverreach API requires suds: https://fedorahosted.org/suds/

You have to define the following parameters in your settings.py:

CLEVERREACH = {'api_key': '<API KEY>',}

API documentation is at http://api.cleverreach.com/soap/doc/5.0/

class `cleverreach.api.v5_1.`**`Client`**
Bases: `object`

`forms_activation_mail`(*form_id*, *email*, *doidata=None*)
Will send the activation mail to the given email. You will have to manualy add the receiver first with "receiver.add" or use an existing one. If the user allready is activated, status will return an error.

`forms_get_code`(*form_id*)
Returns the HTML code for the given embedded form. form_id – the id of the form (not the list!)

`forms_get_list`(*list_id*)
Returns a list of available forms for the given group. Forms are object with the properties [id, name, description]

`group_clear`(*list_id*)
truncates the contents of a the Group

Warning: This may have heavy impact on statistics and campaigns since every related data (receivers, orders, events) will removed.

`group_get_list`()
Returns a list of group classes of the form:

```
(group){
   id = 108907
   name = "test1"
   last_mailing = 1335887342
   last_changed = 1335886187
   count = 84
```

```
    inactive_count = 0
    total_count = 84
  }
```

The dict keys are actually object properties.

query_data (*method*, **args*, ***kwargs*)

receiver_add (*list_id*, *receiver*)
Adds a new single receiver

This function tries to add a single receiver. If the receiver allready exists, the operation will Fail. Use receiver_update in that case. The default form only accepts email, registered, activated, source and attributes. To add more fields you have to add them as attributes. Make sure the keys are the same as the name of the fields in the form. (Check with get_by_email) Attribute keys may only contain lowercase a-z and 0-9.

receiver_delete (*list_id*, *email*)
Deletes an email out of an group.

receiver_get_by_email (*list_id*, *email*, *level=1*)
Gets userdetails based on given readout level. Possible levels (bit whise):

```
000 (0) > Basic readout with (de)activation dates
001 (1) > including attributes (if available)
010 (2) > including Events (if available)
100 (4) > including Orders (if available)
```

receiver_set_active (*list_id*, *email*)
Deactivates a given receiver/email The receiver wont receive anymore mailings from the system. This sets/overwrites the deactivation date with the current date.

receiver_set_inactive (*list_id*, *email*)
Deactivates a given receiver/email The receiver wont receive anymore mailings from the system. This sets/overwrites the deactivation date with the current date.

cleverreach Package

2.1 `utils` Module

Helper functions for Django.

`cleverreach.utils.`**`insert_new_user`**(*user*, *list_id*, *activated=None*, *sendmail=True*, *form_id=None*, *attrs=None*, *client=None*)

Adds a new single receiver. If the email address already exists, an error is raised.

Keyword arguments:

- user – Can be a dict (such as cleaned_data) or an object (such as request.user).
- list_id – The receiver group id
- **activated – (True/False) If the new address is already activated or not (double opt-in).** If activated is not set or None, the cleverreach default settings are used.
- sendmail – boolean, if True, an activation email will be sent out by cleverreach.
- **form_id – (optional) The id of the form used for the list. If this is not** provided, the first available form is used.
- **attrs – (optional) This needs to be a list in the form ['first_name', 'last_name'] and** the attribute must exist on the user object/dict.
- **client – (optional) If an instance of the api.Client exists already, you can** pass it to the function. Otherwise a new instance is created.

2.2 Subpackages

2.2.1 api Package

`api` Package

`api.v5_1` Module

You have to define a group (in the cleverreach admin panel) for each language. The form code is optional, we use the first form if it is not provided.

The Cleverreach API requires suds: https://fedorahosted.org/suds/

You have to define the following parameters in your settings.py:

CLEVERREACH = {'api_key': '<API KEY>',}

API documentation is at http://api.cleverreach.com/soap/doc/5.0/

class `cleverreach.api.v5_1.`**`Client`**

Bases: `object`

`forms_activation_mail`(*form_id*, *email*, *doidata=None*)

Will send the activation mail to the given email. You will have to manualy add the receiver first with "receiver.add" or use an existing one. If the user allready is activated, status will return an error.

`forms_get_code`(*form_id*)

Returns the HTML code for the given embedded form. form_id – the id of the form (not the list!)

`forms_get_list`(*list_id*)

Returns a list of available forms for the given group. Forms are object with the properties [id, name, description]

`group_clear`(*list_id*)

truncates the contents of a the Group

Warning: This may have heavy impact on statistics and campaigns since every related data (receivers, orders, events) will removed.

`group_get_list`()

Returns a list of group classes of the form:

```
(group){
   id = 108907
   name = "test1"
   last_mailing = 1335887342
   last_changed = 1335886187
   count = 84
   inactive_count = 0
   total_count = 84
 }
```

The dict keys are actually object properties.

`query_data`(*method*, **args*, ***kwargs*)

`receiver_add`(*list_id*, *receiver*)

Adds a new single receiver

This function tries to add a single receiver. If the receiver allready exists, the operation will Fail. Use receiver_update in that case. The default form only accepts email, registered, activated, source and attributes. To add more fields you have to add them as attributes. Make sure the keys are the same as the name of the fields in the form. (Check with get_by_email) Attribute keys may only contain lowercase a-z and 0-9.

`receiver_delete`(*list_id*, *email*)

Deletes an email out of an group.

`receiver_get_by_email`(*list_id*, *email*, *level=1*)

Gets userdetails based on given readout level. Possible levels (bit whise):

```
000 (0) > Basic readout with (de)activation dates
001 (1) > including attributes (if available)
010 (2) > including Events (if available)
100 (4) > including Orders (if available)
```

`receiver_set_active`(*list_id*, *email*)
Deactivates a given receiver/email The receiver wont receive anymore mailings from the system. This sets/overwrites the deactivation date with the current date.

`receiver_set_inactive`(*list_id*, *email*)
Deactivates a given receiver/email The receiver wont receive anymore mailings from the system. This sets/overwrites the deactivation date with the current date.

Installation

```
pip install django-cleverreach
```

Add this to your `settings.py`

```
CLEVERREACH = {'api_key': '<API KEY>',
               'raise_exceptions': DEBUG,
}
```

It's highly recommended to also put cleverreach list and form ids as well as the user parameters in there, so you have them if you need them. The easiest way to find the group id is by checking the URL of the group on the receiver groups page.

Django-cleverreach uses the `suds` module: https://fedorahosted.org/suds/

Usage

```python
import cleverreach.api
client = cleverreach.api.Client()  # This opens up a connection to cleverreach.
```

you can silence errors by setting `raise_exceptions` to False. In this case the module logs errors using the logger `cleverreach.api`.

The module throws Exceptions of type `cleverreach.CleverreachAPIException` and `suds.WebFault` in case of a network error.

The currently supported API version is 5.1.

There is a helper function `cleverreach.utils.insert_new_user()` which takes a User instance or a cleaned_data dictionary and sends it to cleverreach.

WARNING: cleverreach updates its APIs and turns old ones off without notice. Keep track of your installations and update this repository as needed.

Indices and tables

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

C

`cleverreach.api`, 5
`cleverreach.api.v5_1`, 5
`cleverreach.utils`, 5

Index

C

cleverreach.api (module), 3, 5
cleverreach.api.v5_1 (module), 3, 5
cleverreach.utils (module), 5
Client (class in cleverreach.api.v5_1), 3, 6

F

forms_activation_mail() (cleverreach.api.v5_1.Client method), 3, 6
forms_get_code() (cleverreach.api.v5_1.Client method), 3, 6
forms_get_list() (cleverreach.api.v5_1.Client method), 3, 6

G

group_clear() (cleverreach.api.v5_1.Client method), 3, 6
group_get_list() (cleverreach.api.v5_1.Client method), 3, 6

I

insert_new_user() (in module cleverreach.utils), 5

Q

query_data() (cleverreach.api.v5_1.Client method), 4, 6

R

receiver_add() (cleverreach.api.v5_1.Client method), 4, 6
receiver_delete() (cleverreach.api.v5_1.Client method), 4, 6
receiver_get_by_email() (cleverreach.api.v5_1.Client method), 4, 6
receiver_set_active() (cleverreach.api.v5_1.Client method), 4, 6
receiver_set_inactive() (cleverreach.api.v5_1.Client method), 4, 7