Haedrian Documentation

Release 0.1

audakel, jroweboy

December 13, 2015

Contents

1 haedrian package **1**
- 1.1 Subpackages . . . 1
- 1.2 Submodules . . . 8
- 1.3 haedrian.admin module . . . 8
- 1.4 haedrian.admin_dashboard module . . . 9
- 1.5 haedrian.admin_menu module . . . 9
- 1.6 haedrian.forms module . . . 9
- 1.7 haedrian.serializers module . . . 10
- 1.8 haedrian.tasks module . . . 10
- 1.9 haedrian.urls module . . . 10
- 1.10 haedrian.views module . . . 10
- 1.11 Module contents . . . 10

2 apiv1 package **11**
- 2.1 Subpackages . . . 11
- 2.2 Submodules . . . 13
- 2.3 apiv1.admin module . . . 13
- 2.4 apiv1.btc_exchange_rate module . . . 13
- 2.5 apiv1.dummy_data module . . . 13
- 2.6 apiv1.models module . . . 13
- 2.7 apiv1.serializers module . . . 14
- 2.8 apiv1.tasks module . . . 15
- 2.9 apiv1.tests module . . . 15
- 2.10 apiv1.urls module . . . 15
- 2.11 apiv1.views module . . . 15
- 2.12 Module contents . . . 16

3 sms package **17**
- 3.1 Subpackages . . . 17
- 3.2 Submodules . . . 17
- 3.3 sms.admin module . . . 17
- 3.4 sms.app module . . . 17
- 3.5 sms.models module . . . 19
- 3.6 sms.sms_verify module . . . 19
- 3.7 sms.strings module . . . 20
- 3.8 sms.tasks module . . . 20
- 3.9 sms.telerivet_backend module . . . 20
- 3.10 sms.tests module . . . 20

3.11 sms.urls module 21
3.12 sms.views module 21
3.13 Module contents 21

4 Indices and tables **23**

Python Module Index **25**

CHAPTER 1

haedrian package

1.1 Subpackages

1.1.1 haedrian.google package

Submodules

haedrian.google.lang module

Valid languages to be used in Google Places API calls. @author: sam@slimkrazy.com

haedrian.google.places module

A simple wrapper around the 'experimental' Google Places API, documented here: http://code.google.com/apis/maps/documentation/places/. This library also makes use of the v3 Maps API for geocoding. Prerequisites: A Google API key with Places activated against it. Please check the Google API console, here: http://code.google.com/apis/console NOTE: Please ensure that you read the Google terms of service (labelled 'Limits and Requirements' on the documentation url) prior to using this library in a production environment. @author: sam@slimkrazy.com

`haedrian.google.places.`**`geocode_location`**(*location*, *sensor=False*)

Converts a human-readable location to lat-lng. Returns a dict with lat and lng keys. keyword arguments: location – A human-readable location, e.g 'London, England' sensor – Boolean flag denoting if the location came from a device using

its' location sensor (default False)

raises: GooglePlacesError – if the geocoder fails to find a location.

exception `haedrian.google.places.`**`GooglePlacesError`**

Bases: `exceptions.Exception`

exception `haedrian.google.places.`**`GooglePlacesAttributeError`**

Bases: `exceptions.AttributeError`

Exception thrown when a detailed property is unavailable. A search query from the places API returns only a summary of the Place. in order to get full details, a further API call must be made using the place_id. This exception will be thrown when a property made available by only the detailed API call is looked up against the summary object. An explicit call to get_details() must be made on the summary object in order to convert a summary object to a detailed object.

class haedrian.google.places.**GooglePlaces**(*api_key*)

Bases: object

A wrapper around the Google Places Query API.

BASE_URL = 'https://maps.googleapis.com/maps/api'

PLACE_URL = 'https://maps.googleapis.com/maps/api/place'

GEOCODE_API_URL = 'https://maps.googleapis.com/maps/api/geocode/json?'

RADAR_SEARCH_API_URL = 'https://maps.googleapis.com/maps/api/place/radarsearch/json?'

NEARBY_SEARCH_API_URL = 'https://maps.googleapis.com/maps/api/place/nearbysearch/json?'

TEXT_SEARCH_API_URL = 'https://maps.googleapis.com/maps/api/place/textsearch/json?'

AUTOCOMPLETE_API_URL = 'https://maps.googleapis.com/maps/api/place/autocomplete/json?'

DETAIL_API_URL = 'https://maps.googleapis.com/maps/api/place/details/json?'

CHECKIN_API_URL = 'https://maps.googleapis.com/maps/api/place/check-in/json?sensor=%s&key=%s'

ADD_API_URL = 'https://maps.googleapis.com/maps/api/place/add/json?sensor=%s&key=%s'

DELETE_API_URL = 'https://maps.googleapis.com/maps/api/place/delete/json?sensor=%s&key=%s'

PHOTO_API_URL = 'https://maps.googleapis.com/maps/api/place/photo?'

MAXIMUM_SEARCH_RADIUS = 50000

RESPONSE_STATUS_OK = 'OK'

RESPONSE_STATUS_ZERO_RESULTS = 'ZERO_RESULTS'

query(***kwargs*)

nearby_search(*language='en'*, *keyword=None*, *location=None*, *lat_lng=None*, *name=None*, *radius=3200*, *rankby='prominence'*, *sensor=False*, *types=[]*)

Perform a nearby search using the Google Places API. One of either location or lat_lng are required, the rest of the keyword arguments are optional. keyword arguments: keyword – A term to be matched against all available fields, including

but not limited to name, type, and address (default None)

location – A human readable location, e.g 'London, England' (default None)

language – The language code, indicating in which language the results should be returned, if possible. (default lang.ENGLISH)

lat_lng – A dict containing the following keys: lat, lng (default None)

name – A term to be matched against the names of the Places. Results will be restricted to those containing the passed name value. (default None)

radius – The radius (in meters) around the location/lat_lng to restrict the search to. The maximum is 50000 meters. (default 3200)

rankby – Specifies the order in which results are listed : ranking.PROMINENCE (default) or ranking.DISTANCE (imply no radius argument).

sensor – Indicates whether or not the Place request came from a device using a location sensor (default False).

types – An optional list of types, restricting the results to Places (default []).

text_search(*query*, *language='en'*, *lat_lng=None*, *radius=3200*, *types=[]*, *location=None*)
Perform a text search using the Google Places API. Only the query kwarg is required, the rest of the keyword arguments are optional. keyword arguments: lat_lng – A dict containing the following keys: lat, lng

(default None)

location – A human readable location, e.g 'London, England' (default None)

radius – The radius (in meters) around the location/lat_lng to restrict the search to. The maximum is 50000 meters. (default 3200)

query – The text string on which to search, for example: "Restaurant in New York".

types – An optional list of types, restricting the results to Places (default []).

autocomplete(*input*, *lat_lng=None*, *location=None*, *radius=3200*, *language='en'*, *types=None*, *components=[]*)
Perform an autocomplete search using the Google Places API. Only the input kwarg is required, the rest of the keyword arguments are optional. keyword arguments: input – The text string on which to search, for example:

"Hattie B's".

lat_lng – A dict containing the following keys: lat, lng (default None)

location – A human readable location, e.g 'London, England' (default None)

radius – The radius (in meters) around the location to which the search is to be restricted. The maximum is 50000 meters. (default 3200)

language – The language code, indicating in which language the results should be returned, if possible. (default lang.ENGLISH)

types – A type to search against. See *types.py* "autocomplete types" for complete list https://developers.google.com/places/documentation/autocomplete#place_types.

components – An optional grouping of places to which you would like to restrict your results. An array containing one or more tuples of: * country: matches a country name or a two letter ISO 3166-1 country code. eg: [('country','US')]

radar_search(*sensor=False*, *keyword=None*, *name=None*, *language='en'*, *lat_lng=None*, *opennow=False*, *radius=3200*, *types=[]*, *location=None*)
Perform a radar search using the Google Places API. One of lat_lng or location are required, the rest of the keyword arguments are optional. keyword arguments: keyword – A term to be matched against all available fields, including

but not limited to name, type, and address (default None)

name – A term to be matched against the names of Places. Results will be restricted to those containing the passed name value.

language – The language code, indicating in which language the results should be returned, if possible. (default lang.ENGLISH)

lat_lng – A dict containing the following keys: lat, lng (default None)

location – A human readable location, e.g 'London, England' (default None)

radius – The radius (in meters) around the location/lat_lng to restrict the search to. The maximum is 50000 meters. (default 3200)

opennow – Returns only those Places that are open for business at the time the query is sent. (default False)

sensor – Indicates whether or not the Place request came from a device using a location sensor (default False).

types – An optional list of types, restricting the results to Places (default []).

checkin (*place_id*, *sensor=False*)

Checks in a user to a place. keyword arguments: place_id – The unique Google identifier for the relevant place. sensor – Boolean flag denoting if the location came from a

device using its location sensor (default False).

get_place (*place_id*, *sensor=False*, *language='en'*)

Gets a detailed place object. keyword arguments: place_id – The unique Google identifier for the required place. sensor – Boolean flag denoting if the location came from a

device using its' location sensor (default False).

language – The language code, indicating in which language the results should be returned, if possible. (default lang.ENGLISH)

add_place (***kwargs*)

Adds a place to the Google Places database. On a successful request, this method will return a dict containing the the new Place's place_id and id in keys 'place_id' and 'id' respectively. keyword arguments: name – The full text name of the Place. Limited to 255

characters.

lat_lng – A dict containing the following keys: lat, lng. accuracy – The accuracy of the location signal on which this request

is based, expressed in meters.

types – The category in which this Place belongs. Only one type can currently be specified for a Place. A string or single element list may be passed in.

language – The language in which the Place's name is being reported. (defaults 'en').

sensor – Boolean flag denoting if the location came from a device using its location sensor (default False).

delete_place (*place_id*, *sensor=False*)

Deletes a place from the Google Places database. keyword arguments: place_id – The textual identifier that uniquely identifies this

Place, returned from a Place Search request.

sensor – Boolean flag denoting if the location came from a device using its location sensor (default False).

request_params

api_key

sensor

haedrian.google.ranking module

Valid place search rankings to be optionally used in Google Place query api calls. @author: sam@slimkrazy.com

Module contents

1.1.2 haedrian.migrations package

Submodules

haedrian.migrations.0001_initial module

class haedrian.migrations.0001_initial.**Migration**(*name*, *app_label*)
Bases: django.db.migrations.migration.Migration

dependencies = [(u'auth', u'__first__'), (u'auth', u'0006_require_contenttypes_0002'), (u'apiv1', u'0001_initial')]

operations = [<CreateModel fields=[(u'id', <django.db.models.fields.AutoField>), (u'name', <django.db.models.fields.C

Module contents

1.1.3 haedrian.wallets package

Submodules

haedrian.wallets.coins_ph module

class haedrian.wallets.coins_ph.**CoinsPhWallet**(*user*)
Bases: *haedrian.wallets.wallet.BaseWallet*

verify_buy(*data*)

buy(*kwargs*)

buy_history(*kwargs*)

get_transfer_history(*id*)

get_history(*kwargs*)

get_wallet_info()

get_pending_balance()

send_to_user(*amount_btc*, *address*)

send(*receiver*, *currency*, *amount_btc*, *amount_local*, *target_address*)

get_balance()

get_address()

get_crypto_routes()

get_exchanges(*kwargs*)

get_exchange_fees(*kwargs*)

get_exchange_types()

get_extra_wallet_info()

create_wallet(*user*, *data*)

haedrian.wallets.coins_ph.**get_correct_wallet_number**()

haedrian.wallets.coins_ph.**make_oauth_request**(*url*, *user*, *body={}*, *put=False*, *headers=''*, *content_type=True*, *get_params={}*)

haedrian.wallets.coins_ph.**make_hmac_request**(*url*, *body=''*)
Make a HMAC request to coins

haedrian.wallets.coins_ph.**get_user_token**(*user*)

haedrian.wallets.gem module

class haedrian.wallets.gem.**GemWallet**(*user*)
Bases: *haedrian.wallets.wallet.BaseWallet*

get_wallet_info()

get_pending_balance()

send_to_user(*user*, *amount_btc*)

send(*address*, *amount_btc*)

get_balance(*user*)

get_address()

haedrian.wallets.gem.**create_app_user**(*email*, *password*)

haedrian.wallets.gem.**create_sms_user**(*phone*)

class haedrian.wallets.gem.**GemBackend**
Bases: object

authenticate(*username*, *password*, ***kwargs*)

haedrian.wallets.test_wallet module

class haedrian.wallets.test_wallet.**TestWallet**(*user*)
Bases: *haedrian.wallets.wallet.BaseWallet*

get_exchange_fees()

get_exchange_types()

create_wallet(*email*, *password*)

get_wallet_info()

get_pending_balance()

send_to_user(*user*, *amount_btc*)

get_address()

get_balance(*user*)

send(*receiver*, *amount_local*, *target_address*)

haedrian.wallets.wallet module

class haedrian.wallets.wallet.**BaseWallet**(*user*)
Bases: object

verify_buy()
Return a unique BTC address for this wallet

buy_history(*kwargs*)
Return a unique BTC address for this wallet

get_address()
Return a unique BTC address for this wallet

get_transfer_history()
Return transaction history for this wallet

get_history()
Return transaction history for this wallet

create_wallet(*email*, *password*)
Make a new node on wallet provider backend

get_wallet_info()
Return a unique handle, ID, or other type of address that given wallet provider uses

get_balance(*user*)
Fetch the latest and most updated balance information in BTC

get_pending_balance()
Fetch the latest and most updated pending balance information in BTC

get_exchange_fees(*kwargs*)
Fetch the latest and most updated pending balance information in BTC

get_exchange_types()
Fetch the latest and most updated pending balance information in BTC

buy()
Buy BTC

send(*receiver*, *amount_local*, *target_address*)
Send Bitcoins amount from this wallet to the address provided :param address - the address to send the BTC to :param amount_local - Amount to send in Local

send_to_user(*user*, *amount_btc*)
Send Bitcoins amount from this wallet to the user provided Each wallet is responsible for finding the appropriate handle or address to send to

:param user - the Haedrian User to send the BTC to :param amount_btc - Amount to send in BTC :exception UserNotFound if the wallet cannot find the user to send to

Module contents

1.2 Submodules

1.3 haedrian.admin module

class haedrian.admin.**PersonAdmin**(*model*, *admin_site*)
Bases: django.contrib.admin.options.ModelAdmin

list_display = ('name', 'country', 'reason')

media

class haedrian.admin.**UserDataAdmin**(*model*, *admin_site*)
Bases: django.contrib.admin.options.ModelAdmin

list_display = ('user', 'phone', 'credit_score', 'default_currency')

media

class haedrian.admin.**PendingDepositAdmin**(*model*, *admin_site*)
Bases: django.contrib.admin.options.ModelAdmin

list_display = ('user', 'time', 'amount', 'user_confirmed', 'exchange_confirmed', 'expired')

media

class haedrian.admin.**ExchangeRatesAdmin**(*model*, *admin_site*)
Bases: django.contrib.admin.options.ModelAdmin

list_display = ('provider', 'code_from', 'code_to', 'buy', 'sell', 'date')

media

class haedrian.admin.**TransactionAdmin**(*model*, *admin_site*)
Bases: django.contrib.admin.options.ModelAdmin

list_display = ('sender', 'receiver', 'amount_local', 'amount_local_currency', 'payment_confirmed', 'date_modified'

media

class haedrian.admin.**WalletAdmin**(*model*, *admin_site*)
Bases: django.contrib.admin.options.ModelAdmin

list_display = ('user', 'type', 'provider_wallet_id', 'access_token', 'refresh_token', 'currency', 'blockchain_address'

media

1.4 haedrian.admin_dashboard module

1.5 haedrian.admin_menu module

1.6 haedrian.forms module

class haedrian.forms.**BetaApplicantForm**(*data=None*, *files=None*, *auto_id=u'id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*)

Bases: django.forms.models.ModelForm

class Meta

model
alias of BetaApplicant

fields = ['name', 'email', 'country', 'reason']

BetaApplicantForm.**base_fields = OrderedDict([('name', <django.forms.fields.CharField object at 0x7f674adf83**

BetaApplicantForm.**declared_fields = OrderedDict()**

BetaApplicantForm.**media**

class haedrian.forms.**EmailUserForm**(*data=None*, *files=None*, *auto_id=u'id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*)

Bases: django.contrib.auth.forms.UserCreationForm

class Meta

model
alias of User

fields = ('username', 'email', 'password1', 'password2')

EmailUserForm.**save**(*commit=True*)

EmailUserForm.**base_fields = OrderedDict([('username', <django.forms.fields.CharField object at 0x7f674adf879**

EmailUserForm.**declared_fields = OrderedDict([('password1', <django.forms.fields.CharField object at 0x7f674**

EmailUserForm.**media**

class haedrian.forms.**NewUserForm**(**args*, ***kwargs*)

Bases: django.forms.models.ModelForm

class Meta

model
alias of UserData

fields = ('phone', 'country', 'organization', 'org_id')

labels = {'organization': u'Select the organization you are connected to', 'org_id': u'ID number'}

NewUserForm.**clean_phone**()

NewUserForm.**clean_org_id**()

NewUserForm.**clean**()

NewUserForm.**base_fields = OrderedDict([('phone', <django.forms.fields.CharField object at 0x7f674adf8890>), ('c**

NewUserForm.**declared_fields = OrderedDict()**

NewUserForm.**media**

.————————- automodule:: haedrian.models

members

undoc-members

show-inheritance

1.7 haedrian.serializers module

class haedrian.serializers.**CoinsphBuySerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)

Bases: rest_framework.serializers.Serializer

update(*instance*, *validated_data*)

create(*validated_data*)

1.8 haedrian.tasks module

class haedrian.tasks.**ExchangeProvider**(*name*, *url*, *known_currencies*)

Bases: object

fetch()

class haedrian.tasks.**CoinsphExchange**

Bases: *haedrian.tasks.ExchangeProvider*

fetch()

class haedrian.tasks.**YahooScraper**

Bases: *haedrian.tasks.ExchangeProvider*

fetch()

1.9 haedrian.urls module

1.10 haedrian.views module

haedrian.views.**index**(*request*)

haedrian.views.**login**(*request*, **args*, ***kwargs*)

1.11 Module contents

CHAPTER 2

apiv1 package

2.1 Subpackages

2.1.1 apiv1.external package

Submodules

apiv1.external.mifosx module

apiv1.external.mifosx.**mifosx_loan**(*user*)

apiv1.external.mifosx.**mifosx_auth**(*base_url*, *username='mifos'*, *password='password'*, *tenant='default'*)

apiv1.external.mifosx.**mifosx_api**(*endpoint*, *user=''*, *app=''*, *method='GET'*, *params={}*, *body=None*, *token=None*, *tenant='default'*)

Make a request to the Mifosx API to get the rest of the client info that we need

apiv1.external.urls module

apiv1.external.views module

class apiv1.external.views.**WhitelistPermission**

Bases: rest_framework.permissions.BasePermission

Check to see if this is an approved IP

has_permission(*request*, *view*)

class apiv1.external.views.**CreateUser**(***kwargs*)

Bases: rest_framework.views.APIView

Manages Users that were created for external organizations

authentication_classes = []

permission_classes = []

get(*request*)

post(*request*)

Module contents

2.1.2 apiv1.internal package

Submodules

apiv1.internal.repayment module

apiv1.internal.utils module

apiv1.internal.utils.**format_currency_display**(*in_currency*, *out_currency*, *amount*, *for_sms=False*)

apiv1.internal.utils.**calculate_fees**(*currency*, *amount_local*)

apiv1.internal.views module

apiv1.internal.views.**create_account**(*request*)

apiv1.internal.views.**format_sms_amounts**(*number*)

apiv1.internal.views_tasks module

apiv1.internal.views_tasks.**get_temp_wallet**(*user*)

Module contents

2.1.3 apiv1.migrations package

Submodules

apiv1.migrations.0001_initial module

class apiv1.migrations.0001_initial.**Migration**(*name*, *app_label*)

Bases: django.db.migrations.migration.Migration

dependencies = [(u'auth', u'__first__')]

operations = [<CreateModel fields=[(u'id', <django.db.models.fields.AutoField>), (u'currency_code', <django.db.mod

apiv1.migrations.0002_supportedcurrencies module

apiv1.migrations.0003_verifygroup_currency module

apiv1.migrations.0004_auto_20150709_1031 module

apiv1.migrations.0005_auto_20150709_1052 module

apiv1.migrations.0006_auto_20150710_1041 module

apiv1.migrations.0007_auto_20150710_1223 module

apiv1.migrations.0008_auto_20150715_1334 module

apiv1.migrations.0009_auto_20150731_1225 module

apiv1.migrations.0010_verifygroup_send_confirmed module

Module contents

2.2 Submodules

2.3 apiv1.admin module

2.4 apiv1.btc_exchange_rate module

class apiv1.btc_exchange_rate.**ExchangeBackend**
Bases: money.exchange.BackendBase

base

rate(*currency*)

quotation(*origin*, *target*)

2.5 apiv1.dummy_data module

2.6 apiv1.models module

class apiv1.models.**SupportedCurrencies**(*id*, *currency_code*, *full_name*)
Bases: django.db.models.base.Model

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception SupportedCurrencies.**MultipleObjectsReturned**
Bases: django.core.exceptions.MultipleObjectsReturned

SupportedCurrencies.**objects** = <django.db.models.manager.Manager object>

class apiv1.models.**VerifyGroup**(*id*, *group_id*, *size*, *created*, *buy_order_id*, *buy_confirmed*, *send_confirmed*, *total_payment*, *currency*, *created_by*)
Bases: django.db.models.base.Model

created_by

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception VerifyGroup.**MultipleObjectsReturned**
Bases: django.core.exceptions.MultipleObjectsReturned

VerifyGroup.**get_next_by_created**(**moreargs*, ***morekwargs*)

VerifyGroup.**get_previous_by_created**(**moreargs*, ***morekwargs*)

VerifyGroup.**objects = <django.db.models.manager.Manager object>**

VerifyGroup.**transaction_set**

VerifyGroup.**verifyperson_set**

class apiv1.models.**VerifyPerson**(*id*, *group*, *mifos_id*, *phone*, *amount*, *confirmed*)
Bases: django.db.models.base.Model

group

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception VerifyPerson.**MultipleObjectsReturned**
Bases: django.core.exceptions.MultipleObjectsReturned

VerifyPerson.**objects = <django.db.models.manager.Manager object>**

2.7 apiv1.serializers module

class apiv1.serializers.**PlacesSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)
Bases: rest_framework.serializers.Serializer

class apiv1.serializers.**UserSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)
Bases: rest_framework.serializers.Serializer

create(*validated_data*)

update(*instance*, *validated_data*)

phone = (CharField(),)

country = <django_countries.fields.CountryField>

class apiv1.serializers.**SendSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)
Bases: rest_framework.serializers.Serializer

create(*validated_data*)

update(*instance*, *validated_data*)

class apiv1.serializers.**ExchangeWorkerSerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)

Bases: rest_framework.serializers.Serializer

create(*validated_data*)

update(*instance*, *validated_data*)

class apiv1.serializers.**CurrencySerializer**(*instance=None*, *data=<class rest_framework.fields.empty>*, ***kwargs*)

Bases: rest_framework.serializers.Serializer

create(*validated_data*)

update(*instance*, *validated_data*)

2.8 apiv1.tasks module

2.9 apiv1.tests module

class apiv1.tests.**AccountTests**(*methodName='runTest'*)

Bases: rest_framework.test.APITestCase

classmethod setUpClass()

Create the haedrian user and setup the mifos account for our test user

classmethod tearDownClass()

test_create_account()

test_get_history()

class apiv1.tests.**WalletTests**(*methodName='runTest'*)

Bases: rest_framework.test.APITestCase

test_send()

2.10 apiv1.urls module

2.11 apiv1.views module

class apiv1.views.**OncePerDayUserThrottle**

Bases: rest_framework.throttling.UserRateThrottle

rate = '10/day'

exception apiv1.views.**ServiceUnavailable**(*detail=None*)

Bases: rest_framework.exceptions.APIException

status_code = 503

default_detail = 'Service temporarily unavailable, try again later.'

2.12 Module contents

CHAPTER 3

sms package

3.1 Subpackages

3.1.1 sms.migrations package

Submodules

sms.migrations.0001_initial module

class sms.migrations.0001_initial.**Migration**(*name*, *app_label*)
Bases: django.db.migrations.migration.Migration

dependencies = [(u'auth', u'__first__')]

operations = [<CreateModel fields=[(u'id', <django.db.models.fields.AutoField>), (u'from_number', <django.db.mode

Module contents

3.2 Submodules

3.3 sms.admin module

3.4 sms.app module

class sms.app.**SMSapplication**(*router*)
Bases: rapidsms.apps.base.AppBase

handle(*msg*)
For testing - to get a new msg obj into our test file try:

fileObject = open("sms/sms_msg.txt",'wb') pickle.dump(msg, fileObject) fileObject.close()

except Exception as e: print (str(e))

sms.app.**sms_send**(*msg*, *parts*)
Send money to someone

Parameters

- **msg** – A full telerivet message object
- **parts** – A list of words in the SMS

Returns A SMS response that the message has been sent

sms.app.**sms_help**(*msg*)
Send a help message

Parameters **msg** – A full telerivet message object

Returns A SMS response with full command usage list

sms.app.**format_sms_amounts**(*number*)

sms.app.**sms_balance**(*msg*, *user*)

sms.app.**save_message**(*msg*)

sms.app.**verify**(*msg*)
Verify that the cell number is associated with a user in our db :param msg: A full telerivet message object :returns: True or false

sms.app.**sms_whoami**(*msg*)
Check what the username is :param msg: A full telerivet message object :returns: Username associated with the phone number

sms.app.**sms_where**(*msg*, *parts*, *user*)

sms.app.**sms_repay**(*msg*, *parts*, *user*)
Start the deposit process with a buy order :param msg: A full telerivet message object :param parts: ex. 'Repay 12' - text message to repay 12 PHP :returns: Instructions on how to deposit at your chosen deposit location

sms.app.**sms_done**(*msg*, *parts*, *user*)
Mark your deposit as paid after you have deposited the money :param msg: A full telerivet message object :param parts: 'done' :returns: A message letting you know the status of the deposit

sms.app.**get_deposit_types**(*user*, *location=''*)
Returns a list of deposit locations :param location: format option :returns: A list of deposit locations

sms.app.**frmt_db_lctn**(*location*)
Formats the database location into nicer, more readable style :param location: sms deposit location :returns: Formated sms deposit location

sms.app.**sms_location**(*msg*, *parts*, *user*)
Allows you to see either your current deposit location, or a list of avaiable deposit locations :param parts: either 'location' or 'location 2' where the number is what deposit location you want :returns: Either a list of locations or a confirmation of change

sms.app.**sms_id**(*msg*, *parts*, *user*)
Allows users to update / input their MFI and MFI ID :param parts: 'id test 23' sets your MFI to 'test' and your id to '23' :returns: If it found a corresponding user in Mifos it will return a message with details about the name, office, loan and wallet balance status

sms.app.**sms_info**(*msg*, *parts*, *user*)
Allows users to see all the basic info we have for them :param parts: 'info' :returns: Message with name, mfi, id, and username

3.5 sms.models module

class sms.models.**Message**(*id*, *from_number*, *to_number*, *message_body*, *created*, *from_city*, *from_country*)
Bases: django.db.models.base.Model

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception Message.**MultipleObjectsReturned**
Bases: django.core.exceptions.MultipleObjectsReturned

Message.**get_next_by_created**(**moreargs*, ***morekwargs*)

Message.**get_previous_by_created**(**moreargs*, ***morekwargs*)

Message.**objects = <django.db.models.manager.Manager object>**

class sms.models.**Signup**(*id*, *phone_number*, *user_handle*)
Bases: django.db.models.base.Model

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception Signup.**MultipleObjectsReturned**
Bases: django.core.exceptions.MultipleObjectsReturned

Signup.**objects = <django.db.models.manager.Manager object>**

class sms.models.**SmsDepositor**(*id*, *phone_number*)
Bases: django.db.models.base.Model

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception SmsDepositor.**MultipleObjectsReturned**
Bases: django.core.exceptions.MultipleObjectsReturned

SmsDepositor.**objects = <django.db.models.manager.Manager object>**

class sms.models.**PendingDeposit**(*id*, *time*, *order_id*, *user_confirmed*, *exchange_confirmed*, *expired*, *amount*, *currency*, *user*)
Bases: django.db.models.base.Model

user

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception PendingDeposit.**MultipleObjectsReturned**
Bases: django.core.exceptions.MultipleObjectsReturned

PendingDeposit.**get_next_by_time**(**moreargs*, ***morekwargs*)

PendingDeposit.**get_previous_by_time**(**moreargs*, ***morekwargs*)

PendingDeposit.**objects = <django.db.models.manager.Manager object>**

3.6 sms.sms_verify module

sms.sms_verify.**verify_sender**(*msg*, *parts*)

sms.sms_verify.**check_number_exist**(*msg*)

sms.sms_verify.**currently_signing_up**(*msg*)

sms.sms_verify.**check_handle_exist**(*msg_handle*)

sms.sms_verify.**create_handle**(*msg*)

sms.sms_verify.**sms_create_user**(*username*, *msg*)

3.7 sms.strings module

sms.strings.**rando**(*type*)

3.8 sms.tasks module

3.9 sms.telerivet_backend module

class sms.telerivet_backend.**TelerivetBackendView**(***kwargs*)
Bases: rapidsms.backends.http.views.GenericHttpBackendView

params = {'text_name': 'message', 'identity_name': 'phone'}

3.10 sms.tests module

sms.tests.**setUpModule**()

class sms.tests.**SMSTestCase**(*methodName='runTest'*)
Bases: django.test.testcases.TestCase

test_user = 'mankey'

phone = '+639420023895'

incoming_msg(*command*, *phone=''*)

process_msg(*command*, *phone='+639420023895'*)

setUp()

test_simple_sms()

test_location()

test_deposit()

test_wallet()

test_model()

test_signup()

3.11 sms.urls module

3.12 sms.views module

sms.views.**receive_text_message**(*request*)

3.13 Module contents

Contents:

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

a

apiv1, 16
apiv1.admin, 13
apiv1.btc_exchange_rate, 13
apiv1.dummy_data, 13
apiv1.external, 12
apiv1.external.mifosx, 11
apiv1.external.urls, 11
apiv1.external.views, 11
apiv1.internal, 12
apiv1.internal.repayment, 12
apiv1.internal.utils, 12
apiv1.internal.views, 12
apiv1.internal.views_tasks, 12
apiv1.migrations, 13
apiv1.migrations.0001_initial, 12
apiv1.models, 13
apiv1.serializers, 14
apiv1.tasks, 15
apiv1.tests, 15
apiv1.urls, 15
apiv1.views, 15

h

haedrian, 10
haedrian.admin, 8
haedrian.forms, 9
haedrian.google, 5
haedrian.google.lang, 1
haedrian.google.places, 1
haedrian.google.ranking, 5
haedrian.migrations, 5
haedrian.migrations.0001_initial, 5
haedrian.serializers, 10
haedrian.tasks, 10
haedrian.urls, 10
haedrian.views, 10
haedrian.wallets, 8
haedrian.wallets.coins_ph, 5
haedrian.wallets.gem, 6
haedrian.wallets.test_wallet, 6
haedrian.wallets.wallet, 7

s

sms.migrations, 17
sms.migrations.0001_initial, 17

A

AccountTests (class in apiv1.tests), 15
ADD_API_URL (haedrian.google.places.GooglePlaces attribute), 2
add_place() (haedrian.google.places.GooglePlaces method), 4
api_key (haedrian.google.places.GooglePlaces attribute), 4
apiv1 (module), 16
apiv1.admin (module), 13
apiv1.btc_exchange_rate (module), 13
apiv1.dummy_data (module), 13
apiv1.external (module), 12
apiv1.external.mifosx (module), 11
apiv1.external.urls (module), 11
apiv1.external.views (module), 11
apiv1.internal (module), 12
apiv1.internal.repayment (module), 12
apiv1.internal.utils (module), 12
apiv1.internal.views (module), 12
apiv1.internal.views_tasks (module), 12
apiv1.migrations (module), 13
apiv1.migrations.0001_initial (module), 12
apiv1.models (module), 13
apiv1.serializers (module), 14
apiv1.tasks (module), 15
apiv1.tests (module), 15
apiv1.urls (module), 15
apiv1.views (module), 15
authenticate() (haedrian.wallets.gem.GemBackend method), 6
authentication_classes (apiv1.external.views.CreateUser attribute), 11
autocomplete() (haedrian.google.places.GooglePlaces method), 3
AUTOCOMPLETE_API_URL (haedrian.google.places.GooglePlaces attribute), 2

B

base (apiv1.btc_exchange_rate.ExchangeBackend attribute), 13
base_fields (haedrian.forms.BetaApplicantForm attribute), 9
base_fields (haedrian.forms.EmailUserForm attribute), 9
base_fields (haedrian.forms.NewUserForm attribute), 10
BASE_URL (haedrian.google.places.GooglePlaces attribute), 2
BaseWallet (class in haedrian.wallets.wallet), 7
BetaApplicantForm (class in haedrian.forms), 9
BetaApplicantForm.Meta (class in haedrian.forms), 9
buy() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5
buy() (haedrian.wallets.wallet.BaseWallet method), 7
buy_history() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5
buy_history() (haedrian.wallets.wallet.BaseWallet method), 7

C

calculate_fees() (in module apiv1.internal.utils), 12
check_handle_exist() (in module sms.sms_verify), 20
check_number_exist() (in module sms.sms_verify), 19
checkin() (haedrian.google.places.GooglePlaces method), 4
CHECKIN_API_URL (haedrian.google.places.GooglePlaces attribute), 2
clean() (haedrian.forms.NewUserForm method), 10
clean_org_id() (haedrian.forms.NewUserForm method), 9
clean_phone() (haedrian.forms.NewUserForm method), 9
CoinsphBuySerializer (class in haedrian.serializers), 10
CoinsphExchange (class in haedrian.tasks), 10
CoinsPhWallet (class in haedrian.wallets.coins_ph), 5
country (apiv1.serializers.UserSerializer attribute), 14
create() (apiv1.serializers.CurrencySerializer method), 15
create() (apiv1.serializers.ExchangeWorkerSerializer method), 15
create() (apiv1.serializers.SendSerializer method), 14

create() (apiv1.serializers.UserSerializer method), 14
create() (haedrian.serializers.CoinsphBuySerializer method), 10
create_account() (in module apiv1.internal.views), 12
create_app_user() (in module haedrian.wallets.gem), 6
create_handle() (in module sms.sms_verify), 20
create_sms_user() (in module haedrian.wallets.gem), 6
create_wallet() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5
create_wallet() (haedrian.wallets.test_wallet.TestWallet method), 6
create_wallet() (haedrian.wallets.wallet.BaseWallet method), 7
created_by (apiv1.models.VerifyGroup attribute), 14
CreateUser (class in apiv1.external.views), 11
CurrencySerializer (class in apiv1.serializers), 15
currently_signing_up() (in module sms.sms_verify), 20

D

declared_fields (haedrian.forms.BetaApplicantForm attribute), 9
declared_fields (haedrian.forms.EmailUserForm attribute), 9
declared_fields (haedrian.forms.NewUserForm attribute), 10
default_detail (apiv1.views.ServiceUnavailable attribute), 15
DELETE_API_URL (haedrian.google.places.GooglePlaces attribute), 2
delete_place() (haedrian.google.places.GooglePlaces method), 4
dependencies (apiv1.migrations.0001_initial.Migration attribute), 12
dependencies (haedrian.migrations.0001_initial.Migration attribute), 5
dependencies (sms.migrations.0001_initial.Migration attribute), 17
DETAIL_API_URL (haedrian.google.places.GooglePlaces attribute), 2

E

EmailUserForm (class in haedrian.forms), 9
EmailUserForm.Meta (class in haedrian.forms), 9
ExchangeBackend (class in apiv1.btc_exchange_rate), 13
ExchangeProvider (class in haedrian.tasks), 10
ExchangeRatesAdmin (class in haedrian.admin), 8
ExchangeWorkerSerializer (class in apiv1.serializers), 14

F

fetch() (haedrian.tasks.CoinsphExchange method), 10
fetch() (haedrian.tasks.ExchangeProvider method), 10
fetch() (haedrian.tasks.YahooScraper method), 10
fields (haedrian.forms.BetaApplicantForm.Meta attribute), 9
fields (haedrian.forms.EmailUserForm.Meta attribute), 9
fields (haedrian.forms.NewUserForm.Meta attribute), 9
format_currency_display() (in module apiv1.internal.utils), 12
format_sms_amounts() (in module apiv1.internal.views), 12
format_sms_amounts() (in module sms.app), 18
frmt_db_lctn() (in module sms.app), 18

G

GemBackend (class in haedrian.wallets.gem), 6
GemWallet (class in haedrian.wallets.gem), 6
GEOCODE_API_URL (haedrian.google.places.GooglePlaces attribute), 2
geocode_location() (in module haedrian.google.places), 1
get() (apiv1.external.views.CreateUser method), 11
get_address() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5
get_address() (haedrian.wallets.gem.GemWallet method), 6
get_address() (haedrian.wallets.test_wallet.TestWallet method), 6
get_address() (haedrian.wallets.wallet.BaseWallet method), 7
get_balance() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5
get_balance() (haedrian.wallets.gem.GemWallet method), 6
get_balance() (haedrian.wallets.test_wallet.TestWallet method), 6
get_balance() (haedrian.wallets.wallet.BaseWallet method), 7
get_correct_wallet_number() (in module haedrian.wallets.coins_ph), 6
get_crypto_routes() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5
get_deposit_types() (in module sms.app), 18
get_exchange_fees() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5
get_exchange_fees() (haedrian.wallets.test_wallet.TestWallet method), 6
get_exchange_fees() (haedrian.wallets.wallet.BaseWallet method), 7
get_exchange_types() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5
get_exchange_types() (haedrian.wallets.test_wallet.TestWallet method), 6
get_exchange_types() (haedrian.wallets.wallet.BaseWallet method), 7
get_exchanges() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5
get_extra_wallet_info() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5

get_history() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5
get_history() (haedrian.wallets.wallet.BaseWallet method), 7
get_next_by_created() (apiv1.models.VerifyGroup method), 14
get_next_by_created() (sms.models.Message method), 19
get_next_by_time() (sms.models.PendingDeposit method), 19
get_pending_balance() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5
get_pending_balance() (haedrian.wallets.gem.GemWallet method), 6
get_pending_balance() (haedrian.wallets.test_wallet.TestWallet method), 6
get_pending_balance() (haedrian.wallets.wallet.BaseWallet method), 7
get_place() (haedrian.google.places.GooglePlaces method), 4
get_previous_by_created() (apiv1.models.VerifyGroup method), 14
get_previous_by_created() (sms.models.Message method), 19
get_previous_by_time() (sms.models.PendingDeposit method), 19
get_temp_wallet() (in module apiv1.internal.views_tasks), 12
get_transfer_history() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5
get_transfer_history() (haedrian.wallets.wallet.BaseWallet method), 7
get_user_token() (in module haedrian.wallets.coins_ph), 6
get_wallet_info() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5
get_wallet_info() (haedrian.wallets.gem.GemWallet method), 6
get_wallet_info() (haedrian.wallets.test_wallet.TestWallet method), 6
get_wallet_info() (haedrian.wallets.wallet.BaseWallet method), 7
GooglePlaces (class in haedrian.google.places), 1
GooglePlacesAttributeError, 1
GooglePlacesError, 1
group (apiv1.models.VerifyPerson attribute), 14

H

haedrian (module), 10
haedrian.admin (module), 8
haedrian.forms (module), 9
haedrian.google (module), 5
haedrian.google.lang (module), 1
haedrian.google.places (module), 1
haedrian.google.ranking (module), 5
haedrian.migrations (module), 5
haedrian.migrations.0001_initial (module), 5
haedrian.serializers (module), 10
haedrian.tasks (module), 10
haedrian.urls (module), 10
haedrian.views (module), 10
haedrian.wallets (module), 8
haedrian.wallets.coins_ph (module), 5
haedrian.wallets.gem (module), 6
haedrian.wallets.test_wallet (module), 6
haedrian.wallets.wallet (module), 7
handle() (sms.app.SMSapplication method), 17
has_permission() (apiv1.external.views.WhitelistPermission method), 11

I

incoming_msg() (sms.tests.SMSTestCase method), 20
index() (in module haedrian.views), 10

L

labels (haedrian.forms.NewUserForm.Meta attribute), 9
list_display (haedrian.admin.ExchangeRatesAdmin attribute), 8
list_display (haedrian.admin.PendingDepositAdmin attribute), 8
list_display (haedrian.admin.PersonAdmin attribute), 8
list_display (haedrian.admin.TransactionAdmin attribute), 8
list_display (haedrian.admin.UserDataAdmin attribute), 8
list_display (haedrian.admin.WalletAdmin attribute), 8
login() (in module haedrian.views), 10

M

make_hmac_request() (in module haedrian.wallets.coins_ph), 6
make_oauth_request() (in module haedrian.wallets.coins_ph), 6
MAXIMUM_SEARCH_RADIUS (haedrian.google.places.GooglePlaces attribute), 2
media (haedrian.admin.ExchangeRatesAdmin attribute), 8
media (haedrian.admin.PendingDepositAdmin attribute), 8
media (haedrian.admin.PersonAdmin attribute), 8
media (haedrian.admin.TransactionAdmin attribute), 8
media (haedrian.admin.UserDataAdmin attribute), 8
media (haedrian.admin.WalletAdmin attribute), 8
media (haedrian.forms.BetaApplicantForm attribute), 9
media (haedrian.forms.EmailUserForm attribute), 9
media (haedrian.forms.NewUserForm attribute), 10
Message (class in sms.models), 19
Message.DoesNotExist, 19
Message.MultipleObjectsReturned, 19

mifosx_api() (in module apiv1.external.mifosx), 11
mifosx_auth() (in module apiv1.external.mifosx), 11
mifosx_loan() (in module apiv1.external.mifosx), 11
Migration (class in apiv1.migrations.0001_initial), 12
Migration (class in haedrian.migrations.0001_initial), 5
Migration (class in sms.migrations.0001_initial), 17
model (haedrian.forms.BetaApplicantForm.Meta attribute), 9
model (haedrian.forms.EmailUserForm.Meta attribute), 9
model (haedrian.forms.NewUserForm.Meta attribute), 9

N

nearby_search() (haedrian.google.places.GooglePlaces method), 2
NEARBY_SEARCH_API_URL (haedrian.google.places.GooglePlaces attribute), 2
NewUserForm (class in haedrian.forms), 9
NewUserForm.Meta (class in haedrian.forms), 9

O

objects (apiv1.models.SupportedCurrencies attribute), 13
objects (apiv1.models.VerifyGroup attribute), 14
objects (apiv1.models.VerifyPerson attribute), 14
objects (sms.models.Message attribute), 19
objects (sms.models.PendingDeposit attribute), 19
objects (sms.models.Signup attribute), 19
objects (sms.models.SmsDepositor attribute), 19
OncePerDayUserThrottle (class in apiv1.views), 15
operations (apiv1.migrations.0001_initial.Migration attribute), 12
operations (haedrian.migrations.0001_initial.Migration attribute), 5
operations (sms.migrations.0001_initial.Migration attribute), 17

P

params (sms.telerivet_backend.TelerivetBackendView attribute), 20
PendingDeposit (class in sms.models), 19
PendingDeposit.DoesNotExist, 19
PendingDeposit.MultipleObjectsReturned, 19
PendingDepositAdmin (class in haedrian.admin), 8
permission_classes (apiv1.external.views.CreateUser attribute), 11
PersonAdmin (class in haedrian.admin), 8
phone (apiv1.serializers.UserSerializer attribute), 14
phone (sms.tests.SMSTestCase attribute), 20
PHOTO_API_URL (haedrian.google.places.GooglePlaces attribute), 2
PLACE_URL (haedrian.google.places.GooglePlaces attribute), 2
PlacesSerializer (class in apiv1.serializers), 14
post() (apiv1.external.views.CreateUser method), 11
process_msg() (sms.tests.SMSTestCase method), 20

Q

query() (haedrian.google.places.GooglePlaces method), 2
quotation() (apiv1.btc_exchange_rate.ExchangeBackend method), 13

R

radar_search() (haedrian.google.places.GooglePlaces method), 3
RADAR_SEARCH_API_URL (haedrian.google.places.GooglePlaces attribute), 2
rando() (in module sms.strings), 20
rate (apiv1.views.OncePerDayUserThrottle attribute), 15
rate() (apiv1.btc_exchange_rate.ExchangeBackend method), 13
receive_text_message() (in module sms.views), 21
request_params (haedrian.google.places.GooglePlaces attribute), 4
RESPONSE_STATUS_OK (haedrian.google.places.GooglePlaces attribute), 2
RESPONSE_STATUS_ZERO_RESULTS (haedrian.google.places.GooglePlaces attribute), 2

S

save() (haedrian.forms.EmailUserForm method), 9
save_message() (in module sms.app), 18
send() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5
send() (haedrian.wallets.gem.GemWallet method), 6
send() (haedrian.wallets.test_wallet.TestWallet method), 6
send() (haedrian.wallets.wallet.BaseWallet method), 7
send_to_user() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5
send_to_user() (haedrian.wallets.gem.GemWallet method), 6
send_to_user() (haedrian.wallets.test_wallet.TestWallet method), 6
send_to_user() (haedrian.wallets.wallet.BaseWallet method), 7
SendSerializer (class in apiv1.serializers), 14
sensor (haedrian.google.places.GooglePlaces attribute), 4
ServiceUnavailable, 15
setUp() (sms.tests.SMSTestCase method), 20
setUpClass() (apiv1.tests.AccountTests class method), 15
setUpModule() (in module sms.tests), 20
Signup (class in sms.models), 19
Signup.DoesNotExist, 19
Signup.MultipleObjectsReturned, 19
sms (module), 21

sms.admin (module), 17
sms.app (module), 17
sms.migrations (module), 17
sms.migrations.0001_initial (module), 17
sms.models (module), 19
sms.sms_verify (module), 19
sms.strings (module), 20
sms.tasks (module), 20
sms.telerivet_backend (module), 20
sms.tests (module), 20
sms.urls (module), 21
sms.views (module), 21
sms_balance() (in module sms.app), 18
sms_create_user() (in module sms.sms_verify), 20
sms_done() (in module sms.app), 18
sms_help() (in module sms.app), 18
sms_id() (in module sms.app), 18
sms_info() (in module sms.app), 18
sms_location() (in module sms.app), 18
sms_repay() (in module sms.app), 18
sms_send() (in module sms.app), 17
sms_where() (in module sms.app), 18
sms_whoami() (in module sms.app), 18
SMSapplication (class in sms.app), 17
SmsDepositor (class in sms.models), 19
SmsDepositor.DoesNotExist, 19
SmsDepositor.MultipleObjectsReturned, 19
SMSTestCase (class in sms.tests), 20
status_code (apiv1.views.ServiceUnavailable attribute), 15
SupportedCurrencies (class in apiv1.models), 13
SupportedCurrencies.DoesNotExist, 13
SupportedCurrencies.MultipleObjectsReturned, 13

T

tearDownClass() (apiv1.tests.AccountTests class method), 15
TelerivetBackendView (class in sms.telerivet_backend), 20
test_create_account() (apiv1.tests.AccountTests method), 15
test_deposit() (sms.tests.SMSTestCase method), 20
test_get_history() (apiv1.tests.AccountTests method), 15
test_location() (sms.tests.SMSTestCase method), 20
test_model() (sms.tests.SMSTestCase method), 20
test_send() (apiv1.tests.WalletTests method), 15
test_signup() (sms.tests.SMSTestCase method), 20
test_simple_sms() (sms.tests.SMSTestCase method), 20
test_user (sms.tests.SMSTestCase attribute), 20
test_wallet() (sms.tests.SMSTestCase method), 20
TestWallet (class in haedrian.wallets.test_wallet), 6
text_search() (haedrian.google.places.GooglePlaces method), 2
TEXT_SEARCH_API_URL (haedrian.google.places.GooglePlaces attribute), 2
transaction_set (apiv1.models.VerifyGroup attribute), 14
TransactionAdmin (class in haedrian.admin), 8

U

update() (apiv1.serializers.CurrencySerializer method), 15
update() (apiv1.serializers.ExchangeWorkerSerializer method), 15
update() (apiv1.serializers.SendSerializer method), 14
update() (apiv1.serializers.UserSerializer method), 14
update() (haedrian.serializers.CoinsphBuySerializer method), 10
user (sms.models.PendingDeposit attribute), 19
UserDataAdmin (class in haedrian.admin), 8
UserSerializer (class in apiv1.serializers), 14

V

verify() (in module sms.app), 18
verify_buy() (haedrian.wallets.coins_ph.CoinsPhWallet method), 5
verify_buy() (haedrian.wallets.wallet.BaseWallet method), 7
verify_sender() (in module sms.sms_verify), 19
VerifyGroup (class in apiv1.models), 13
VerifyGroup.DoesNotExist, 14
VerifyGroup.MultipleObjectsReturned, 14
VerifyPerson (class in apiv1.models), 14
VerifyPerson.DoesNotExist, 14
VerifyPerson.MultipleObjectsReturned, 14
verifyperson_set (apiv1.models.VerifyGroup attribute), 14

W

WalletAdmin (class in haedrian.admin), 8
WalletTests (class in apiv1.tests), 15
WhitelistPermission (class in apiv1.external.views), 11

Y

YahooScraper (class in haedrian.tasks), 10