django-obm

Release 0.3.5

Alexander Polishchuk

Jun 26, 2020

CONTENTS

CHAPTER

ONE

RATIONALE

There are a lot of projects that need a cryptocurrency payment system under the hood for transactions sending/receiving, unique addresses creation, fee estimating and other blockchain interactions. Each of them have to implement their own service for that propose due to lack of opensource product, that could satisfy their needs. This project aims to provide such functionality and facilitate the implementation of such a microservice.

CHAPTER

TWO

RESOURCES

- Documentation: https://django-obm.readthedocs.io

CHAPTER

THREE

INSTALLATION

See Installation for complete instructions.

```
pip install django-obm
```

CHAPTER

FOUR

REQUIREMENTS

- Python 3.8 or higher.
- bitcoin-core node

CHAPTER

FIVE

FEATURES

- BTC (bitcoin-core) support
- sending/receiving transactions and confirmation
- unique addresses creation
- fee estimating
- REST API for actions above

5.1 Future features

- support of: ETH, ETC, DASH, BCHABC, BCHSV, LTC, ZEC, XEM, XRP, etc.
- `django_obm.wallet` app witch help in implementation of multi cryptocurrency wallet

CHAPTER

SIX

IS DJANGO-OBM PRODUCTION READY?

The project is now under active development. Use at your own risk.

CHAPTER

SEVEN

EXAMPLE

You can find the example project in this repo example folder.

CHAPTER

EIGHT

CONTRIBUTING

See CONTRIBUTING.md for instructions.

CHAPTER

NINE

SUPPORT THE DEVELOPER

9.1 Sponsors

Special thanks for Swapzilla project that paid me part of the development.

You can also become the sponsor and get priority development of the features you require. Just contact me.

9.2 Buy me a beer

```
BTC 179B1vJ8LvAQ2r9ABNhp6kDE2yQZfm1Ng3
```

CHAPTER

TEN

TABLE OF CONTENTS

10.1 Installation

10.1.1 Python package

```
pip install django-obm
```

10.1.2 Django

Add packages in `INSTALLED_APPS` in your `settings.py`.

```
INSTALLED_APPS = [
    ...
    'django.contrib.auth',
    'django.contrib.admin',
    'django.contrib.contenttypes',

    'django_obm',
]
```

If you need the REST API for `django_obm` models, update your `urls.py`.

```
urlpatterns = [
    ...
    url(r'^obm/', include('django_obm.urls')),
    ...
]
```

10.1.3 Post-Installation

Migrate database

In your Django root execute the command below to create your database tables:

```
python manage.py migrate
```

Install cryptocurrency nodes

`django-obm` interact with blockchains through cryptocurrency nodes. You should install them and allow RPC access. Configuration example for each supported node is in example project.

Now only following nodes are being supported by the framework:

- Bitcoin: bitcoin-core

10.2 Quickstart

Note: This guide assume that you have installed and configured bitcoin-core node. See *Install cryptocurrency nodes* for instructions.

This guide will walk you through the basics of creating simple bitcoin payment system that can receive and send transactions, create addresses, and estimate fees.

10.2.1 Creating currency and node objects

`django-obm` store configuration for specific node in database. There are two ways to create them.

1. Managemant command

Open `settings.py` and define `BLOCKCHAIN_NODES_INITIAL_CONFIG` setting. It maps on fields of `django_obm.models.Node` and related to it `django_obm.models.Currency` models.

```
BLOCKCHAIN_NODES_INITIAL_CONFIG = [
    {
        'currency': {
            'name': 'BTC',
            'min_confirmations': 2,
        },
        'name': 'bitcoin-core',
        'is_default': True,
        'rpc_username': 'rpcuser',
        'rpc_password': '************',
        'rpc_host': 'localhost',
        'rpc_port': 18332,
    },
]
```

To apply the config on database execute command bellow in your Django root:

```
$ python manage.py init_nodes
<Currency: BTC> created successfully.
<Node: bitcoin-core> created successfully.
```

It's worth clarifying, that you can't create `Node` or `Currency` object if framework doesn't support corresponded cryptocurrency or node. To discover supported things you can use special connectors registry property.

```
>>> from django_obm import connectors
>>> connectors.registry.available_currencies
{'BTC'}
>>> connectors.registry.available_nodes
{'bitcoin-core'}
```

2. Manual creation

Also it can be created in any place of your project then when you need it.

```
>>> from django_obm import models
>>> currency = models.Currency.objects.create(
...     name='BTC',
...     min_confirmations=2,
... )
>>> models.Node.objects.create(
...     name='bitcoin-core',
...     currency=currency,
...     is_default=True,
...     rpc_username='username',
...     rpc_password='password',
...     rpc_host='127.0.0.1',
...     rpc_port=18332,
... )
<Node: bitcoin-core>
```

10.2.2 Receive payments

There are method and daemon to fetch received transactions from nodes and write them into database. Each transaction will get status `tx.is_confirmed == True` if the conformations number greater than `tx.node.currency.min_conformations`, in our case it's 2.

Method

Now you are ready to receive payments. For fetch new received transaction call `models.Node` manager `process_receipts` method:

```
>>> models.Node.objects.process_receipts()
```

Daemon

Also you can use built-in daemon, that will do it by timer. Just execute `run_receipts_processing` django command.

```
python manage.py run_receipts_processing --frequency=120
```

It runs `process_receipts` `models.Node` manager method with specified frequency (defaults to 60 sec.). For defineing your own default frequency set `RECEIPTS_PROCESSING_DEFAULT_FREQUENCY` to needed value in `settings.py`.

The daemon has the `--once` option that allow to execute `process_receipts` only once, like regular command. It might be helpful if you wish to use some system-level (like systemd, crontab etc.) tool to accept payments.

10.2.3 Example

You can find the example in example project.

10.3 Configuration

Available settings:

BLOCKCHAIN_NODE_TIMEOUT (=3) Specifies the timeout for request to blockchain node.

BLOCKCHAIN_NODES_INITIAL_CONFIG (=[]) Specifies the initial database state for nodes and currencies related to them. It is a list of dicts that represents the `Node` object with nested `Currency` that look like below:

```
BLOCKCHAIN_NODES_INITIAL_CONFIG = [
    {
        'currency': {
            'name': 'BTC',
            'min_confirmations': 2,
        },
        'name': 'bitcoin-core',
        'is_default': True,
        'rpc_username': 'rpcuser',
        'rpc_password': '************',
        'rpc_host': 'localhost',
        'rpc_port': 18332,
    },
]
```

You can apply it on your database with `init_nodes` managemant command.

RECEIPTS_PROCESSING_DEFAULT_FREQUENCY (=60) Defines default receipts processing frequency for `run_receipts_processing` managemant command.

CHAPTER

ELEVEN

INDICES AND TABLES

- genindex
- modindex
- search