Backfeed Rest API Documentation

Release 0.1

Jelle Gerbrandy

May 19, 2016

Contents

This API is still in development (on https://github.com/Backfeed/backfeed-restapi).

CHAPTER 1

REST API for the backfeed-protocol

1.1 Installation

You need pip installed:

```
sudo apt-get install python-pip
```

You can now either directly install from the github repository:

```
pip install git+https://github.com/Backfeed/backfeed-restapi.git
```

1.2 Starting a server

First you need to create an settings file. You can download it from the git repository:

```
wget https://raw.githubusercontent.com/Backfeed/backfeed-restapi/master/development.ini
```

You can then start the server like this:

```
pserve development.ini
```

The default settings will use a database that runs completely in memory, so you will loose any changes after restarting the server.

1.3 Contributing

See CONTRIBUTING.rst

CHAPTER 2

Data Model

The basic objects of the protocol as `users`, `contributions` and `evaluations`.

2.1 Users

Users have `tokens`, a number representing the amount of *ownership* of a contract, and `reputation`, a number that represents the user reputation within the system.

```
{
    "id": 1,
    "reputation": 20.0,
    "reputation_normalized": 1.0,
    "tokens": 50.0,
    "total_reputation": 20.0
}
```

2.2 Contributions

A contribution has a `contributor` (which is the user that has made the contribution), a `score` which represents how much the contribution has been valued by the community, and `engaged_reputation`, a value between 0 and 1 that represents the sum of the reputation of the users that have evaluated the contribution.

```
{
    "contributor": {
        "id": 2,
        "reputation": 0.5,
        "tokens": 49.0
    },
    "id": 1,
    "stats": {
        "engaged_reputation": 0.0,
        "evaluations": {
            "0": {
                "reputation": 0.0
            },
            "1": {
                "reputation": 0.0
            }
        },
```

```
        "quality": 0.0,
        "score": 0.0
    },
    "type": "article"
}
```


2.3 Evaluations

An evaluation of a contribution has an `evaluator`, which is the user that has made the evaluation, the `contribution` that the evaluation pertains to, and a `value`.


```
{
    "contribution": {
        "contributor": {
            "id": 3,
            "reputation": 0.2506265664160401,
            "tokens": 49.0
        },
        "id": 2,
        "stats": {
            "engaged_reputation": 0.2481203007518797,
            "evaluations": {
                "0": {
                    "reputation": 0.0
                },
                "1": {
                    "reputation": 0.2481203007518797
                }
            },
            "quality": 0.2481203007518797,
            "score": 0.2481203007518797
        },
        "type": "article"
    },
    "evaluator": {
        "id": 4,
        "reputation": 19.8,
        "reputation_normalized": 0.2481203007518797,
        "tokens": 50.0,
        "total_reputation": 79.8
    },
    "id": 1,
    "value": 1.0
}
```

CHAPTER 3

API Documentation

All services have a `{contract}` as a parameter. This is the name of the contract. The actual effect of adding contributions and evaluations in terms of tokens and reputation of users will depend on this contract name.

3.1 User Collection service at /{contract}/users

Users

3.1.1 GET

Get a list of users

Response: json

3.1.2 POST

values in the body

- **reputation** (float) - (optional)
- **tokens** (float) - (optional)
- **referrer_id** (int) - (optional)

Create a new user.

Response: json

3.2 User Resource service at /{contract}/users/{id}

Users

3.2.1 GET

Get the user identified by `id`

Response: json

3.3 Contribution Collection service at /{contract}/contributions

Contributions

3.3.1 GET

values in the querystring

- **contributor_id** (int) - (optional)
- **order_by** (str) - (default: "-score")
- **limit** (int) - (default: 100)
- **start** (int) - (default: 0)

Get a list of contributions.

The parameter 'order_by' can take as its values:

- *score* order by score
- *-score* order descendingly, by score
- *time*: the time the contribution was added
- *-time*: last-added first

Response: json

3.3.2 POST

values in the body

- **contributor_id** (int)

Create a new contribution

Param contributor_id the id of the user that has made the contribution

Returns information about the new contribution

Response: json

3.4 Contribution Resource service at /{contract}/contributions/{id}

Contributions

3.4.1 GET

Get the contribution

Response: json

3.5 Evaluation Collection service at /{contract}/evaluations

Evaluations

3.5.1 GET

values in the querystring

- **contribution_id** (int) - (optional)
- **evaluator_id** (int) - (optional)

Get a list of users

Response: json

3.5.2 POST

values in the body

- **value** (int)
- **evaluator_id** (int)
- **contribution_id** (int)

Create a new evaluation.

Creating an evaluation will update tokens and reputation from the contributor, the evaluator, and previous evaluators.

Param evaluator_id required. The id of the user to that does the evaluation.

Param contribution_id required. The id of the contribution that is being evaluated

Param value required. This is a number - which values are accepted depends on the contract.

Returns information about the added evaluation

Response: json

3.6 Evaluation Resource service at /{contract}/evaluations/{id}

Evaluations

3.6.1 GET

Get evaluations from the contract

Response: json