CritiqueBrainz

Release 0.1

unknown

Aug 30, 2022

CONTENTS

Hello there! Thanks for your interest in CritiqueBrainz project. It is a repository for Creative Commons licensed reviews for music, books and other things. This project is based on data from MusicBrainz - open music encyclopedia, and BookBrainz - open book encyclopedia. Everyone - including you - can participate and contribute.

This is an open source project. Source code is available on GitHub.

CHAPTER

ONE

CONTENTS

1.1 Introduction

1.1.1 Setting up the server

You can set up CritiqueBrainz server using Docker. It requires minimum amount of configuration.

> **Warning:** Make sure you have the latest version of docker and docker-compose.

Configuration

First, you need to create custom configuration file. Copy the skeleton configuration:

```
$ cp custom_config.py.example custom_config.py
```

Then open `critiquebrainz/custom_config.py` in your favourite text editor and update configuration values as described next.

MusicBrainz login

To be able to log in using MusicBrainz accounts you need to set `MUSICBRAINZ_CLIENT_ID` and `MUSICBRAINZ_CLIENT_SECRET` OAuth values.

These values can be obtained from MusicBrainz after you register a new instance of CritiqueBrainz at https://musicbrainz.org/account/applications/register (you'll need a MusicBrainz account). In the `Callback URL` field type:

```
http://localhost:8200/login/musicbrainz/post
```

Note: By default, CritiqueBrainz will be available at http://localhost:8200. If you change the port or host that it is running on, you should set the Callback URL to the host that you configure.

After application has been registered, set `MUSICBRAINZ_CLIENT_ID` and `MUSICBRAINZ_CLIENT_SECRET` in your `custom_config.py` to the values that you see on the MusicBrainz website.

Spotfiy API authentication

To use the Spotify Web API, you need to set the `SPOTIFY_CLIENT_ID` and `SPOTIFY_CLIENT_SECRET` values. OAuth keys can be obtained after registering on the Spotify developer website.

After registering and logging into your Spotify account, head to https://developer.spotify.com/my-applications/ and then register your application following the instructions at https://developer.spotify.com/web-api/tutorial/#registering-your-application.

Finally, save the obtained `Client ID` and `Client Secret` fields in your `custom_config.py` fields `SPOTIFY_CLIENT_ID` and `SPOTIFY_CLIENT_SECRET` respectively.

Startup

We provide a helper tool, `develop.sh` to wrap docker-compose and to provide some shortcuts to common tasks.

Build all CritiqueBrainz services with:

```
$ ./develop.sh build
```

A MusicBrainz database is required for running CritiqueBrainz. The `mbdump.tar.bz2` is the core MusicBrainz archive which includes the tables for artist, release_group etc. The `mbdump-derived.tar.bz2` archive contains annotations, user tags and search indexes. These archives include all the data required for setting up an instance of CritiqueBrainz.

You can import the database dump by downloading and importing the data in a single command:

```
$ ./develop.sh run --rm musicbrainz_db
```

Note: You can also manually download the dumps and then import it:-

i. For this, you have to download the dumps `mbdump.tar.bz2` and `mbdump-derived.tar.bz2` from http://ftp.musicbrainz.org/pub/musicbrainz/data/fullexport/.

> **Warning:** Make sure that you get the latest dumps

ii. Set the environment variable `DUMPS_DIR` to the path of the folders containing the dumps. For example:

```
$ export DUMPS_DIR="/home/me/folder/with/dumps"
```

You can check that the variable `DUMPS_DIR` has been succesfully assigned or not by:

```
$ echo $DUMPS_DIR
```

This must display the path of your folder containing database dumps. The folder must contain at least the file `mbdump.tar.bz2`.

iii. Then import the database dumps by this command:

```
$ ./develop.sh run --rm -v $DUMPS_DIR:/home/musicbrainz/dumps \
-v $PWD/data/mbdata:/var/lib/postgresql/data/pgdata musicbrainz_db
```

Note: You can also use the smaller sample dumps available at http://ftp.musicbrainz.org/pub/musicbrainz/data/sample/ to set up the MusicBrainz database. However, note that these dumps are .tar.xz dumps while CritiqueBrainz

currently only supports import of .tar.bz2 dumps. So, a decompression of the sample dumps and recompression into .tar.bz2 dumps will be needed. This can be done using the following command

$ xzcat mbdump-sample.tar.xz | bzip2 > mbdump.tar.bz2

Warning: Keep in mind that this process is very time consuming, so make sure that you don't delete the `data/mbdata` directory by accident. Also make sure that you have about 25GB of free space to keep the MusicBrainz data.

A BookBrainz database is also required for running CritiqueBrainz. You can import the database dump by downloading and importing the data in a single command:

```
$ ./develop.sh run --rm critiquebrainz bash scripts/download-import-bookbrainz-dump.sh
```

Next, initialize the CritiqueBrainz database:

```
$ ./develop.sh run --rm critiquebrainz python3 manage.py init_db
```

You will also need to run some commands to build the static assets (javascript and css files) for the first run:

```
$ ./develop.sh run --rm static_builder npm install
$ ./develop.sh run --rm static_builder npm run pre-dev
```

Then you can start all the services:

```
$ ./develop.sh up
```

Visit CritiqueBrainz at `http://localhost:8200` in your browser.

Note: CB Runs on 8200. change line x if you want it somewhere else.

Importing data dump

We provide daily data dumps from https://critiquebrainz.org that include reviews and most of the data associated with them. If you want to import that into your own installation, download the archives from http://ftp.musicbrainz.org/pub/musicbrainz/critiquebrainz/dump/ (you'll need to get the base archive `cbdump.tar.bz2` and the reviews `cbdump-reviews-all.tar.bz2`) and use `python3 manage.py dump import` command. First you need to import base archive and then the one that contains reviews. For example:

```
$ ./develop.sh run --rm critiquebrainz python3 manage.py dump import cbdump.tar.bz2
$ ./develop.sh run --rm critiquebrainz python3 manage.py dump import cbdump-reviews-all.
↪tar.bz2
```

Keep in mind that CritiqueBrainz only supports importing into an empty database. This should work if you just ran `init_db` command.

1.1.2 Testing

To test the web server run:

```
$ ./develop.sh test up --build
```

1.1.3 Modifying strings

CritiqueBrainz supports interface translation. If you add or modify strings that will be displayed to users, then you need to wrap them in one of two functions: `gettext()` or `ngettext()`.

Before committing changes don't forget to extract all strings into `messages.pot`:

```
$ python3 manage.py update_strings
```

For more info see *Translation*.

1.2 API

CritqueBrainz provides an API which you can use to interact with content on the website. There's also an *OAuth* protocol implementation which you can build your applications on top of.

1.2.1 Endpoint reference

CritiqueBrainz provides various endpoints that can be used to interact with the data. Web API uses JSON format.

Root URL: `https://critiquebrainz.org/ws/1`

Below you will find description of all available endpoints.

Reviews

`GET /review/languages`

Get list of supported review languages (language codes from ISO 639-1).

Example Request:

```
$ curl https://critiquebrainz.org/ws/1/review/languages \
       -X GET
```

Example Response:

```
{
  "languages": [
    "aa",
    "ab",
    "af",
    "ak",
    "yo",
    "za",
    "zh",
```

(continues on next page)

(continued from previous page)

```
    "zu"
  ]
}
```

Response Headers

- Content-Type – *application/json*

GET /review/

Get list of reviews.

Request Example:

```
$ curl "https://critiquebrainz.org/ws/1/review/?limit=1&offset=50" \
       -X GET
```

Response Example:

```
{
  "count": 9197,
  "limit": 1,
  "offset": 50,
  "reviews": [
    {
      "created": "Fri, 16 May 2008 00:00:00 GMT",
      "edits": 0,
      "entity_id": "09259937-6477-3959-8b10-af1cbaea8e6e",
      "entity_type": "release_group",
      "id": "c807d0b4-0dd0-43fe-a7c4-d29bb61f389e",
      "language": "en",
      "last_updated": "Fri, 16 May 2008 00:00:00 GMT",
      "license": {
        "full_name": "Creative Commons Attribution-NonCommercial-ShareAlike 3.0␣
↪Unported",
        "id": "CC BY-NC-SA 3.0",
        "info_url": "https://creativecommons.org/licenses/by-nc-sa/3.0/"
      },
      "popularity": 0,
      "source": "BBC",
      "source_url": "http://www.bbc.co.uk/music/reviews/vh54",
      "text": "TEXT CONTENT OF REVIEW",
      "rating": 5,
      "user": {
        "created": "Wed, 07 May 2014 16:20:47 GMT",
        "display_name": "Jenny Nelson",
        "id": "3bf3fe0c-6db2-4746-bcf1-f39912113852",
        "karma": 0,
        "user_type": "Noob"
      },
      "votes": {
        "positive": 0,
        "negative": 0
      }
```

(continues on next page)

(continued from previous page)

```
    }
  ]
}
```

Query Parameters

- **entity_id** – UUID of an entity to retrieve reviews for **(optional)**
- **entity_type** – `musicbrainz` (for reviews about MusicBrainz entities) or `bookbrainz` (for reviews about bookbrainz entities) or one of the following entity types *critiquebrainz.db.review.ENTITY_TYPES* **(optional)**
- **user_id** – user's UUID **(optional)**
- **sort** – `popularity` or `published_on` **(optional)**. Defaults to `published_on`
- **sort_order** – `asc` or `desc` **(optional)**. Defaults to `desc`
- **limit** – results limit, min is 0, max is 50, default is 50 **(optional)**
- **offset** – result offset, default is 0 **(optional)**
- **language** – language code (ISO 639-1) **(optional)**
- **review_type** – `review` or `rating`. If set, only return reviews which have a text review, or a rating **(optional)**
- **include_metadata** – `true` or `false`. Include metadata of the entity **(optional)**

NOTE: If entity_id is provided, then additional top-level item "average_rating" which includes the average of all ratings for this entity and the total count of these ratings is also included.

Response Headers

- Content-Type – *application/json*

`POST /review/`

Publish a review.

OAuth scope: review

Request Headers

- Content-Type – *application/json*

JSON Parameters

- **entity_id** (*uuid*) – UUID of the entity that is being reviewed
- **entity_type** (*string*) – One of the supported reviewable entities. 'release_group' or 'event' etc.
- **text** (*string*) – Text part of review, min length is 25, max is 5000 **(optional)**
- **rating** (*integer*) – Rating part of review, min is 1, max is 5 **(optional)**
- **license_choice** (*string*) – license ID
- **language** (*string*) – language code (ISO 639-1), default is `en` **(optional)**
- **is_draft** (*boolean*) – whether the review should be saved as a draft or not, default is `False` **(optional)**

NOTE: You must provide some text or rating for the review.

Response Headers

- Content-Type – *application/json*

GET /review/(uuid: *review_id*)/revisions/(int: *rev*)

Get a particular revisions of review with a specified UUID.

Request Example:

```
$ curl https://critiquebrainz.org/ws/1/review/b7575c23-13d5-4adc-ac09-2f55a647d3de/
↪revisions/1 \
       -X GET
```

Response Example:

```
{
  "revision": {
    "id": 1,
    "review_id": "b7575c23-13d5-4adc-ac09-2f55a647d3de",
    "text": "TEXT CONTENT OF REVIEW",
    "rating": 5,
    "timestamp": "Tue, 10 Aug 2010 00:00:00 GMT",
    "votes_negative": 0,
    "votes_positive": 0
  }
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – review not found

Response Headers

- Content-Type – *application/json*

GET /review/(uuid: *review_id*)/revisions

Get revisions of review with a specified UUID.

Request Example:

```
$ curl https://critiquebrainz.org/ws/1/review/b7575c23-13d5-4adc-ac09-2f55a647d3de/
↪revisions \
       -X GET
```

Response Example:

```
{
  "revisions": [
    {
      "id": 1,
      "review_id": "b7575c23-13d5-4adc-ac09-2f55a647d3de",
      "text": "TEXT CONTENT OF REVIEW",
      "rating": 5,
      "timestamp": "Tue, 10 Aug 2010 00:00:00 GMT",
```

(continues on next page)

(continued from previous page)

```
      "votes_negative": 0,
      "votes_positive": 0
    }
  ]
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – review not found

Response Headers

- Content-Type – *application/json*

POST /review/(uuid: *review_id*)/report

Create spam report for a specified review.

OAuth scope: vote

Response Headers

- Content-Type – *application/json*

GET /review/(uuid: *review_id*)/vote

Get your vote for a specified review.

Request Example:

```
$ curl "https://critiquebrainz.org/ws/1/review/9cb11424-d070-4ac1-8771-a8703ae5cccd/
↪vote" \
       -X GET \
       -H "Authorization: Bearer <access token>"
```

Response Example:

```
{
  "vote": {
    "vote": true,
    "voted_at": "Thu, 22 Dec 2016 11:49:56 GMT"
  }
}
```

OAuth scope: vote

Response Headers

- Content-Type – *application/json*

PUT /review/(uuid: *review_id*)/vote

Set your vote for a specified review.

OAuth scope: vote

Request Example:

```
$ curl "https://critiquebrainz.org/ws/1/review/9cb11424-d070-4ac1-8771-a8703ae5cccd/
↪vote" \
       -X PUT \
       -H "Content-type: application/json" \
       -H "Authorization: Bearer <access token>" \
       -d '{"vote":true}'
```

Response Example:

```
{
  "message": "Request processed successfully"
}
```

JSON Parameters

- **vote** (*boolean*) – `true` if upvote, `false` if downvote

NOTE: Voting on reviews without text is not allowed.

Status Codes

- 200 OK – success
- 400 Bad Request – invalid request (see source)
- 403 Forbidden – daily vote limit exceeded
- 404 Not Found – review not found

Response Headers

- Content-Type – *application/json*

DELETE /review/(uuid: *review_id*)/vote

Delete your vote for a specified review.

OAuth scope: vote

Request Example:

```
$ curl "https://critiquebrainz.org/ws/1/review/9cb11424-d070-4ac1-8771-a8703ae5cccd/
↪vote" \
       -X DELETE \
       -H "Authorization: Bearer <access token>"
```

Response Example:

```
{
  "message": "Request processed successfully"
}
```

Response Headers

- Content-Type – *application/json*

GET /review/(uuid: *review_id*)

Get review with a specified UUID.

Request Example:

```
$ curl https://critiquebrainz.org/ws/1/review/b7575c23-13d5-4adc-ac09-2f55a647d3de \
       -X GET
```

Response Example:


```
{
  "review": {
    "created": "Tue, 10 Aug 2010 00:00:00 GMT",
    "edits": 0,
    "entity_id": "03e0a99c-3530-4e64-8f50-6592325c2082",
    "entity_type": "release_group",
    "id": "b7575c23-13d5-4adc-ac09-2f55a647d3de",
    "language": "en",
    "last_updated": "Tue, 10 Aug 2010 00:00:00 GMT",
    "license": {
      "full_name": "Creative Commons Attribution-NonCommercial-ShareAlike 3.0␣
↪Unported",
      "id": "CC BY-NC-SA 3.0",
      "info_url": "https://creativecommons.org/licenses/by-nc-sa/3.0/"
    },
    "popularity": 0,
    "source": "BBC",
    "source_url": "http://www.bbc.co.uk/music/reviews/3vfd",
    "text": "TEXT CONTENT OF REVIEW",
    "rating": 5,
    "user": {
      "created": "Wed, 07 May 2014 14:55:23 GMT",
      "display_name": "Paul Clarke",
      "id": "f5857a65-1eb1-4574-8843-ae6195de16fa",
      "karma": 0,
      "user_type": "Noob"
    },
    "votes": {
      "positive": 0,
      "negative": 0
    }
  }
}
```


Status Codes

- 200 OK – no error
- 404 Not Found – review not found

Response Headers

- Content-Type – *application/json*

`DELETE /review/(uuid: review_id)`

Delete review with a specified UUID.

OAuth scope: review

Request Example:

```
$ curl "https://critiquebrainz.org/ws/1/review/9cb11424-d070-4ac1-8771-a8703ae5cccd
↪" \
       -X DELETE \
       -H "Authorization: Bearer <access token>"
```

Response Example:

```
{
  "message": "Request processed successfully"
}
```

Status Codes

- 200 OK – success
- 403 Forbidden – access denied
- 404 Not Found – review not found

Response Headers

- Content-Type – *application/json*

`POST /review/(uuid: review_id)`

Update review with a specified UUID.

OAuth scope: review

JSON Parameters

- **`text`** (*string*) – Text part of review, min length is 25, max is 5000 **(optional)**
- **`rating`** (*integer*) – Rating part of review, min is 1, max is 5 **(optional)**

NOTE: Please provide only those parameters which need to be updated

Status Codes

- 200 OK – success
- 400 Bad Request – invalid request
- 403 Forbidden – access denied
- 404 Not Found – review not found

Response Headers

- Content-Type – *application/json*

`GET /reviews/`

Get a list of reviews with specified UUIDs.

Hidden reviews are omitted from the response. UUIDs which are not valid result in an error. UUIDs which are not review ids are omitted from the response.

The returned data is a dictionary where each key is the review id. This UUID is formatted in the canonical UUID representation and may be different from the UUID that was passed in as the query parameter. A mapping of user-provided UUIDs to canonical representation is provided in the `review_id_mapping` element of the response.

Request Example:

```
$ curl https://critiquebrainz.org/ws/1/reviews?review_ids=B7575C23-13D5-4ADC-AC09-
↪2F55A647D3DE,e4364ed2-a5db-4427-8456-ea7604b499ef \
       -X GET
```

Response Example:

```
{
  "reviews": {
    "b7575c23-13d5-4adc-ac09-2f55a647d3de": {
      "created": "Tue, 10 Aug 2010 00:00:00 GMT",
      "edits": 0,
      "entity_id": "03e0a99c-3530-4e64-8f50-6592325c2082",
      "entity_type": "release_group",
      "id": "b7575c23-13d5-4adc-ac09-2f55a647d3de",
      "language": "en",
      "last_updated": "Tue, 10 Aug 2010 00:00:00 GMT",
      "license": {
        "full_name": "Creative Commons Attribution-NonCommercial-ShareAlike 3.0␣
↪Unported",
        "id": "CC BY-NC-SA 3.0",
        "info_url": "https://creativecommons.org/licenses/by-nc-sa/3.0/"
      },
      "popularity": 0,
      "source": "BBC",
      "source_url": "http://www.bbc.co.uk/music/reviews/3vfd",
      "text": "TEXT CONTENT OF REVIEW",
      "rating": 5,
      "user": {
        "created": "Wed, 07 May 2014 14:55:23 GMT",
        "display_name": "Paul Clarke",
        "id": "f5857a65-1eb1-4574-8843-ae6195de16fa",
        "karma": 0,
        "user_type": "Noob"
      },
      "votes": {
        "positive": 0,
        "negative": 0
      }
    },
    -- more reviews here --
  },
  "review_id_mapping": {
    "B7575C23-13D5-4ADC-AC09-2F55A647D3DE": "b7575c23-13d5-4adc-ac09-2f55a647d3de"
  }
}
```

Status Codes

- 200 OK – no error
- 400 Bad Request – Too many review ids were requested, or a provided review id isn't a valid UUID
- 503 Service Unavailable – Query string is too long

Query Parameters

- **review_ids** – a comma-separated list of review IDs to retrieve.

Response Headers

- Content-Type – *application/json*

Users

GET /user/me/applications

Get your applications.

Request Example:

```
$ curl "https://critiquebrainz.org/ws/1/user/me/applications" \
        -X GET \
        -H "Authorization: Bearer <access token>"
```

Response Example:

```
{
  "applications": [
    {
      "website": "https://your-website.com",
      "user_id": "your-unique-user-id",
      "name": "Name of your Application",
      "redirect_uri": "https://your-call-back.com/uri",
      "client_id": "your Oauth client ID",
      "client_secret": "your super-secret Oauth client secret",
      "desc": "Application description set by you."
    }
  ]
}
```

Response Headers

- Content-Type – *application/json*

GET /user/me/reviews

Get your reviews.

Response Headers

- Content-Type – *application/json*

GET /user/me/tokens

Get your OAuth tokens.

Request Example:

```
$ curl "https://critiquebrainz.org/ws/1/user/me/tokens" \
        -X GET \
        -H "Authorization: Bearer <access token>"
```

Response Example:

```
{
  "tokens": [
    {
      "scopes": "user",
      "client": {
        "website": "https://your-website.com",
        "user_id": "your-unique-user-id",
        "name": "Name of your Application",
        "redirect_uri": "https://your-call-back.com/uri",
        "client_id": "your Oauth client ID",
        "client_secret": "your super-secret Oauth client secret",
        "desc": "Application description set by you."
      },
      "refresh_token": "refresh token generated for your Oauth authorization code"
    }
  ]
}
```

Response Headers

- Content-Type – *application/json*

GET /user/me

Get your profile information.

Request Example:

```
$ curl "https://critiquebrainz.org/ws/1/user/me" \
       -X GET \
       -H "Authorization: Bearer <access token>"
```

Response Example:

```
{
  "user": {
    "display_name": "your_display_name",
    "created": "Fri, 02 Dec 2016 19:02:47 GMT",
    "user_type": "Noob",
    "email": "your_email_id",
    "karma": 0,
    "musicbrainz_username": "username/id associated with musicbrainz",
    "id": "your-unique-user-id"
  }
}
```

Query Parameters

- **inc** – includes

Response Headers

- Content-Type – *application/json*

`POST /user/me`

Modify your profile.

OAuth scope: user

Request Headers

- Content-Type – *application/json*

JSON Parameters

- **`display_name`** (*string*) – Display name **(optional)**
- **`email`** (*string*) – Email address **(optional)**

Response Headers

- Content-Type – *application/json*

`DELETE /user/me`

Delete your profile.

OAuth scope: user

Request Example:

```
$ curl "https://critiquebrainz.org/ws/1/user/me" \
       -X DELETE \
       -H "Authorization: Bearer <access token>"
```

Response Example:

```
{
  "message": "Request processed successfully"
}
```

Response Headers

- Content-Type – *application/json*

`GET /user/`

Get list of users.

Request Example:

```
$ curl "https://critiquebrainz.org/ws/1/user/?offset=10&limit=3" \
       -X GET
```

Response Example:

```
{
  "count": 925,
  "limit": 3,
  "offset": 10,
  "users": [
    {
      "created": "Wed, 07 May 2014 14:46:58 GMT",
      "display_name": "Display Name",
      "id": "b291a99b-7bb0-4531-ba45-f6cfb4d944de",
```

(continues on next page)

(continued from previous page)


```
      "karma": 0,
      "user_type": "Noob"
    },
    {
      "created": "Wed, 07 May 2014 14:46:59 GMT",
      "display_name": "Name comes here",
      "id": "a52e1629-a516-43c2-855f-bb195aeb2a33",
      "karma": 3,
      "user_type": "Noob"
    },
    {
      "created": "Wed, 07 May 2014 14:47:00 GMT",
      "display_name": "Display Name",
      "id": "1fb36917-d4d3-411b-82c4-901d949e17b8",
      "karma": 0,
      "user_type": "Noob"
    }
  ]
}
```


Query Parameters

- **limit** – results limit, min is 0, max is 50, default is 50 **(optional)**
- **offset** – result offset, default is 0 **(optional)**

Response Headers

- Content-Type – *application/json*

GET /user/(uuid: *user_id*)

Get profile of a user with a specified UUID.

Request Example:

```
$ curl https://critiquebrainz.org/ws/1/user/ae5a003f-292c-497e-afbd-8076e9626f2e \
       -X GET
```

Response Example:


```
{
  "user": {
    "created": "Wed, 07 May 2014 14:47:03 GMT",
    "display_name": "User's Name comes here",
    "id": "ae5a003f-292c-497e-afbd-8076e9626f2e",
    "karma": 0,
    "user_type": "Noob"
  }
}
```


Response Headers

- Content-Type – *application/json*

OAuth

See *OAuth documentation* for more info.

POST /oauth/token

OAuth 2.0 token endpoint.

Form Parameters

- **string client_id** –
- **string client_secret** –
- **string redirect_uri** –
- **string grant_type** – `authorization_code` or `refresh_token`
- **string code** – (not required if grant_type is `refresh_token`)
- **string refresh_token** – (not required if grant_type is `authorization_code`)

Response Headers

- Content-Type – *application/json*

Constants

Constants that are relevant to using the API:

critiquebrainz.db.review.ENTITY_TYPES = ['event', 'place', 'release_group', 'work', 'artist', 'label', 'recording', 'bb_edition_group', 'bb_literary_work', 'bb_author', 'bb_series']

Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list. The argument must be an iterable if specified.

1.2.2 OAuth 2.0

Introduction

OAuth 2.0 is a protocol that lets you create applications that can request access different parts of user profiles. This page describes how to use OAuth 2.0 when accessing a server API from a web server application.

All developers need to register their application before getting started. A registered OAuth application is assigned a unique Client ID and Client Secret. The Client Secret should not be shared.

Authorization

Requesting authorization

The authorization process starts by redirecting the user to the authorization endpoint with a set of query string parameters describing the authorization request. The endpoint is located at `https://critiquebrainz.org/oauth/authorize` and is only available over HTTPS. HTTP connections are refused.

Parameters:

Parameter	Description
response_type	Desired grant type. Must be set to `code`.
client_id	Client ID assigned to your application.
redirect_uri	URL where clients should be redirected after authorization. It must match exactly the URL you entered when registering your application.
scope	*Optional.* Comma separated set of scopes. Identifies resources that your application will have access to. You should request only the scopes that your application needs. See *Scopes* for more info.
state	*Optional.* Random string that is used to protect against cross-site request forgery attacks. Server roundtrips this parameter, so your application receives the same value it sent.

Example:

```
https://critiquebrainz.org/oauth/authorize?
   response_type=code&
   scope=review,vote&
   redirect_uri=http%3A%2F%2Fwww.example.com.com%2Fcallback&
   client_id=yDDvwAzPUnoD8imvTpVm&
```

Handling server response

The response will be sent to the `redirect_url` specified in configuration of your application. If user approves access request, then the response will contain authorization code and `state` parameter (if it was included in the request):

```
https://www.example.com/callback?state=a35Bsw1koA3pM34&code=3lUq7v15Qqm9g8YcoUT31D
```

If the user does not approve the request, the response will contain an error message:

```
https://www.example.com/callback?state=a35Bsw1koA3pM34&error=access_denied
```

Access token

Exchanging authorization code for an access token

After your application receives an authorization code, it can send a *POST* request the token endpoint located at `https://critiquebrainz.org/ws/1/oauth/token`, to exchange the authorization code for an access token. As before, this endpoint is only available over HTTPS and HTTP requests will be refused. The request includes following parameters:

Parameter	Description
code	The authorization code returned from authorization request.
client_id	Client ID assigned to your application.
client_secret	Client secret assigned to your application.
redirect_uri	URL where response will be sent. Must match your application configuration.
grant_type	Must be set to `authorization_code`.

Token request might look like this:

```
POST /ws/1/oauth/token HTTP/1.1
Host: critiquebrainz.org
Content-Type: application/x-www-form-urlencoded

code=3lUq7v15Qqm9g8YcoUT31D&
client_id=yDDvwAzPUnoD8imvTpVm&
client_secret=AFjfpM7Ar1KtD0bnfV5InQ&
redirect_uri=http%3A%2F%2Fwww.example.com%2Fcallback&
grant_type=authorization_code
```

Successful response to this request will contain the following fields:

Field	Description
access_token	Access token that can be used to get data from CritiqueBrainz API
token_type	Type of returned access token. Will have `Bearer` value.
expires_in	The remaining lifetime of returned access token.
refresh_token	Token that can be used to obtain new access token. See below for more info about this.

Response example:

```
{
  "access_token": "2YotnFZFEjr1zCsicMWpAA",
  "expires_in": 3600,
  "token_type": "Bearer",
  "refresh_token": "tGzv3JOkF0XG5Qx2TlKWIA"
}
```

Refreshing an access token

To obtain a new access token, your application needs to send POST request to `https://critiquebrainz.org/ws/1/oauth/token`. The request must include the following parameters:

Parameter	Description
refresh_token	The refresh token returned during the authorization code exchange.
client_id	Client ID assigned to your application.
client_secret	Client secret assigned to your application.
redirect_uri	URL where response will be sent. Must match your application configuration.
grant_type	Must be set to `refresh_token`

Request might look like this:

```
POST /ws/1/oauth/token HTTP/1.1
Host: critiquebrainz.org
Content-Type: application/x-www-form-urlencoded

refresh_token=tGzv3JOkF0XG5Qx2TlKWIA&
client_id=yDDvwAzPUnoD8imvTpVm&
client_secret=AFjfpM7Ar1KtD0bnfV5InQ&
redirect_uri=http%3A%2F%2Fwww.example.com%2Fcallback&
grant_type=refresh_token
```

As long as the user has not revoked the access granted to your application, you will receive response that will look like this:

```
{
  "access_token": "zIYanFZFEjr1zCsicMWpo6",
  "expires_in": 3600,
  "token_type": "Bearer",
  "refresh_token": "PUnoD8im10XG5QxGzv3JO1"
}
```

Scopes

Authorization requests have a limited scope. You should request only the scopes that your application necessarily needs. CritiqueBrainz provides the following scopes:

- `review` - Create and modify reviews.
- `vote` - Submit and delete votes on reviews.
- `user` - Modify profile info and delete profile.

1.3 Contributing

Apart from writing reviews, there are plenty of ways to contribute to the project. See CONTRIBUTING.md file in our GitHub repository for more info about that.

1.3.1 Structure

CritiqueBrainz project is separated into three main packages: data, frontend, and web service (ws). The data package is used to interact with the database. The frontend provides user-friendly interface that is available at https://critiquebrainz.org.

Here's an overview of the project structure:

1.4 Database

CritiqueBrainz uses PostgresSQL DBMS to save reviews and other data.

1.4.1 Entities

- Review
- Revision
- License
- User
- Vote
- SpamReport

- OAuthClient
- OAuthGrant
- OAuthToken

1.4.2 Schema diagram

1.5 Exporting data

You can create backups including various pieces of data that we store: reviews, revisions, users, and other stuff. Some parts include private data about users that is not meant to be shared.

1.5.1 Creating data dumps

Below you can find commands that can be used to create backups of different formats.

Complete database dump *(for PostgreSQL)*:

```
$ docker-compose -f docker/docker-compose.dev.yml run --rm critiquebrainz python3 manage.
↪py dump full_db
```

MusicBrainz-style dump public *(no private info)*:

```
$ docker-compose -f docker/docker-compose.dev.yml run --rm critiquebrainz python3 manage.
↪py dump public
```

JSON dump with all reviews *(no private info)*:

```
$ docker-compose -f docker/docker-compose.dev.yml run --rm critiquebrainz python3 manage.
↪py dump json
```

All commands have rotation feature which can be enabled by passing *-r* argument.

1.6 Translation

Our goal is to make CritiqueBrainz project accessible to readers and writers all over the world. You can post reviews in any language you want, but it is also important to provide good way to find these reviews. That's why we need your help!

Translation process is being done on Transifex platform. You can submit your translation suggestions there and collaborate with other translators. If you want to contribute translations, that's where you should to do it.

Our project page is located at https://www.transifex.com/projects/p/critiquebrainz/.

There are a couple of things you should know if you are trying to modify strings. See the info below.

1.6.1 Extracting strings

To extract strings into a Portable Object Template file (*messages.pot*) use command:

```
$ python3 manage.py extract_strings
```

1.6.2 Adding support for a new language

To add support for new language add its Transifex code into `SUPPORTED_LANGUAGES` list in default configuration file: `default_config.py`. After that you can pull translations from Transifex:

```
$ python3 manage.py pull_translations
```

You will need to create *.transifexrc* file that will look like:

```
[https://www.transifex.com]
hostname = https://www.transifex.com
username = <YOUR_EMAIL>
password = <YOUR_PASSWORD>
token =
```

More info about setup process is available at http://docs.transifex.com/developer/client/setup.

1.6.3 Additional info

CritiqueBrainz uses Flask-Babel extension. Its documentation is available at http://pythonhosted.org/Flask-Babel/.

HTTP ROUTING TABLE

/oauth

/review

/reviews

/user

INDEX

E