PartnerHTC Backend Document

Release 0.0.1

boot.AI

May 07, 2020

This is an API that manages Mini Mofa system using Python3, Flask, and SQLAlchemy.

1. Install Windows Server 2012 R2
2. Install Miniconda 3.7
3. Install packages required in requirements.txt
4. Install MySQL 5.7, Redis

1. The base endpoint is: http://27.72.196.104:5012
2. The backup endpoint is: http://27.72.196.105:5012
3. All endpoints return a JSON object.
4. All time and timestamp related fields are in seconds.

1. 2xx Request processed successfully
2. 5xx return codes are used for internal errors; the issue is on Backend side.

1. Configure project environment (Either A. Install Pycharm OR B. Create a Virtual Environment)
    1. Install Environment

    - Manually install packages to project interpreter (Pycharm -> Preferences -> Project -> Project Interpreter -> plus button on the lower left side of the package table) and apply changes OR type the command below on the activated virtual environment.

    ```
    conda activate
    conda create -n htcenv python=3.6
    conda activate htcenv
    pip install -r requirements.txt
    ```

2. Install MySQL, REDIS Database
    1. Search on the web on how to install MySQL in your OS
    2. Create database through piping

    ```
    mysql -u root < <Path to file>/create_db.sql
    * NOTE: depending on your mysql config, you need to provide your
    password if you have one
    ```

   3. Download and install redis via this link
   4. Restart Computer

3. Initialize and Populate Database
   1. Edit line 14 of settings.py and use the correct url to your mysql.

      ```
      'mysql://root:<password>@localhost/htc'
      ```

   2. Either run the line below.

      ```
      $ sh database_populator.sh
      ```

4. Run application:

   ```
   python manage.py
   ```

5. Refer to controller on how to test the code through curl or Postman

CHAPTER 1

Documentation for the Code

`app.api.v1.auth.`**`login`**`( )`

This is controller of the login api.

Request Body:

username: string, require
The username of the user. Max length accepted is 50 and minimum length is 1

password: string, require
The password of the user wanted to log in. Max length accepted is 50 and minimum length is 1

Returns:

access_token: string
your access token. you needed to save this to access to backend services. Please put access_token to Header Authorization: Bearer <accees_token>

force_change_password: boolean
When true. The user have force change password after login.

group: string
Current group of the user

list_permissions: list[string,]
Mapping action and resource user can access. For example create_user or get_users

login_failed_attempts: number
Number login failed of the current user.

logout_after_inactivate: number
Number in seconds. If user do not have any action in the period time. Use will be logged out

refresh_token: string
Token use to refresh expire time of the access token. Please put refresh_token to Header Authorization: Bearer <refresh_token>

Examples:

```
curl --location --request GET
'http://<sv_address>:5012/api/v1/users/4658df34-8630-11ea-b850-588a5a158009'
--header 'Authorization: Bearer <refresh_token>'
```

`app.api.v1.auth.`**`logout`**`( )`

Add token to blacklist :return:

`app.api.v1.auth.`**`refresh`**`( )`

This api use for refresh expire time of the access token. Please inject the refresh token in Authorization header

Args:

refresh_token : *string, require*
If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns:

access_token : new access token

- *Index*
- *Module Index*
- *Search Page*

Index