auth Documentation

Release stable

Sep 27, 2017

Contents

1 What is Auth? 3

2 requirements 5

3 Installation 7

4 Show me an example 9

5 Authirization Methods 11

6 RESTful API 13

7 RESTful API helpers 15

8 API Methods 17

9 Deployment 19

10 Dockerizing 21

11 Copyright 23

12 Documentation 25

13 Unit Tests and Coverage 27

14 To DO 29

RESTful, Simple Authorization system with ZERO configuration.

CHAPTER 1

What is Auth?

Auth is a module that makes authorization simple and also scalable and powerful. It also has a beautiful RESTful API for use in micro-service architectures and platforms. It is originally desinged to use in Appido, a scalable media market in Iran.

It supports Python2.6+ and if you have a mongodb backbone, you need ZERO configurations steps. Just type `auth-server` and press enter!

I use Travis and Codecov to keep myself honest.

CHAPTER 2

requirements

You need to access to **mongodb**. If you are using a remote mongodb, provide these environment variables:

`MONGO_HOST` and `MONGO_PORT`

CHAPTER 3

Installation

```
pip install auth
```

CHAPTER 4

Show me an example

ok, lets image you have two users, **Jack** and **Sara**. Sara can cook and Jack can dance. Both can laugh.

You also need to choose a secret key for your application. Because you may want to use Auth in various tools and each must have a secret key for seperating their scope.

```
my_secret_key = "pleaSeDoN0tKillMyC_at"
from auth import Authorization
cas = Authorization(my_secret_key)
```

Now, Lets add 3 groups, Cookers, Dancers and Laughers. Remember that groups are Roles. So when we create a group, indeed we create a role:

```
cas.add_group('cookers')
cas.add_group('dancers')
cas.add_group('laughers')
```

Ok, great. You have 3 groups and you need to authorize them to do special things.

```
cas.add_permission('cookers', 'cook')
cas.add_permission('dancers', 'dance')
cas.add_permission('laughers', 'laugh')
```

Good. You let cookers to cook and dancers to dance etc... The final part is to set memberships for Sara and Jack:

```
cas.add_membership('sara', 'cookers')
cas.add_membership('sara', 'laughers')
cas.add_membership('jack', 'dancers')
cas.add_membership('jack', 'laughers')
```

That's all we need. Now lets ensure that jack can dance:

```
if cas.user_has_permission('jack', 'dance'):
    print('YES!!! Jack can dance.')
```

CHAPTER 5

Authirization Methods

use pydoc to see all methods:

```
pydoc auth.Authorization
```

CHAPTER 6

RESTful API

Lets run the server on port 4000:

```python
from auth import api, serve
serve('localhost', 4000, api)
```

Or, from version 0.1.2+ you can use this command:

```
auth-server
```

Simple! Authorization server is ready to use.

You can use it via simple curl or using mighty Requests module. So in you remote application, you can do something like this:

```python
import requests
secret_key = "pleaSeDoN0tKillMyC_at"
auth_api = "http://127.0.0.1:4000/api"
```

Lets create admin group:

```python
requests.post(auth_api+'/role/'+secret_key+'/admin')
```

And lets make Jack an admin:

```python
requests.post(auth_api+'/permission/'+secret_key+'/jack/admin')
```

And finally let's check if Sara still can cook:

```python
requests.get(auth_api+'/has_permission/'+secret_key+'/sara/cook')
```

CHAPTER 7

RESTful API helpers

auth comes with a helper class that makes your life easy.

```
from auth.client import Client
service = Client('srv201', 'http://192.168.99.100:4000')
print(service)
service.get_roles()
service.add_role(role='admin')
```

CHAPTER 8

API Methods

```
pydoc auth.CAS.REST.service
```

- `/ping` [GET]

  Ping API, useful for your monitoring tools
- `/api/membership/{KEY}/{user}/{role}` [GET/POST/DELETE]

  Adding, removing and getting membership information.
- `/api/permission/{KEY}/{role}/{name}` [GET/POST/DELETE]

  Adding, removing and getting permissions
- `/api/has_permission/{KEY}/{user}/{name}` [GET]

  Getting user permission info
- `/api/role/{KEY}/{role}` [GET/POST/DELETE]

  Adding, removing and getting roles
- `/api/which_roles_can/{KEY}/{name}` [GET]

  For example: Which roles can send_mail?
- `/api/which_users_can/{KEY}/{name}` [GET]

  For example: Which users can send_mail?
- `/api/user_permissions/{KEY}/{user}` [GET]

  Get all permissions that a user has
- `/api/role_permissions/{KEY}/{role}` [GET]

  Get all permissions that a role has
- `/api/user_roles/{KEY}/{user}` [GET]

  Get roles that user assinged to

- `/api/roles/{KEY}` [GET]

  Get all available roles

CHAPTER 9

Deployment

Deploying Auth module in production environment is easy:

```
gunicorn auth:api
```

CHAPTER 10

Dockerizing

It's simple:

```
docker build -t python/auth-server https://raw.githubusercontent.com/ourway/auth/
↪master/Dockerfile
docker run --name=auth -e MONGO_HOST='192.168.99.100' -p 4000:4000 -d --
↪restart=always --link=mongodb-server python/auth-server
```

CHAPTER 11

Copyright

- Farsheed Ashouri @

CHAPTER 12

Documentation

Feel free to dig into source code. If you think you can improve the documentation, please do so and send me a pull request.

CHAPTER 13

Unit Tests and Coverage

I am trying to add tests as much as I can, but still there are areas that need improvement.

CHAPTER 14

To DO

- Add Authentication features
- Improve Code Coverage