Flask_auth_service_mongo

Zinobe

Dec 21, 2020

CONTENTS:

Flask Auth Service Mongo is an Auth JWT implementation for Flask with Mongo. It provides useful services such as: Authentication, CRUD for User and Role managment.

CHAPTER

ONE

GET THE CODE

The source is available on GitLab.

1.1 Installation

Install Flask Auth Service Mongo is pretty simple. Here is a step by step plan to how do it.

You have to install the package in the virtual environment:

```
pip install flask-auth-service-mongo
```

1.2 Configuration

1.2.1 ENV vars

Define the following environment variables

- **(str) Key with which the token is generated** `SECRET_KEY=`
- **(bool) Turn the token whitelist on or off** `WHITE_LIST_TOKEN=`
- **(int) Minimum username length** `USERNAME_MIN_LENGTH=`
- **(int) Minimum password length** `PASSWORD_MIN_LENGTH=`
- **(int) Minutes in which the token will expire** `TOKEN_EXPIRE_MINUTES=`
- **(int) Length of the password generated in the reset** `RESET_PASSWORD_LEN_GENERATOR=`
- **(bool) Default value for auth.required(require_password_change)** `REQUIRE_PASSWORD_CHANGE=`

Example

```
USERNAME_MIN_LENGTH=5
PASSWORD_MIN_LENGTH=8
WHITE_LIST_TOKEN=0
SECRET_KEY='not-secret'
```

1.3 User Guide

Here you will learn how to use the package correctly and you will see useful examples

1.3.1 Access Control

This service provides a access control.

Paths

Check this example to how use auth.required.

is_authenticated

```
from flask import Blueprint
from flask_restplus import Api, Resource
from flask_auth_service_mongo import auth

view_admin = Blueprint('view_admin', __name__)
api = Api(view_admin)


@api.route('/blog')
class ApiLogout(Resource):
    @auth.required()
    def post(self):
        return {
            'message': 'Ok'
        }
```

POST /admin/blog

Example request:

```
POST /admin/blog HTTP/1.1
Host: example.com
Content-Type: application/json
Authorization: Bearer token_123_xD

{}
```

specific_role

```
from flask import Blueprint
from flask_restplus import Api, Resource
from flask_auth_service_mongo import auth

view_admin = Blueprint('view_admin', __name__)
api = Api(view_admin)
```

(continues on next page)

(continued from previous page)

```
@api.route('/blog')
class ApiLogout(Resource):
    @auth.required('custom_role')
    def post(self):
        return {
            'message': 'Ok'
        }
```

list_roles

```
from flask import Blueprint
from flask_restplus import Api, Resource
from flask_auth_service_mongo import auth

view_admin = Blueprint('view_admin', __name__)
api = Api(view_admin)


@api.route('/blog')
class ApiLogout(Resource):
    @auth.required(['custom_role', 'other_role'])
    def post(self):
        return {
            'message': 'Ok'
        }
```

Middleware Mutations Graphene

Check this example to how use MutationMiddleware.

```
import graphene
from flask import Blueprint
from flask_auth_service_mongos import MutationMiddleware

view_admin = Blueprint('view_admin', __name__)

my_schema = graphene.Schema()

my_access_control = [
    {'mutation': 'create_blog', 'roles': ['is_authenticated']},
    {'mutation': 'update_blog', 'roles': ['role_user', 'role_admin']},
    {'mutation': 'delete_blog', 'roles': ['role_admin']},
]

view_admin.add_url_rule(
    '/graphql',
    view_func=result_to_json(
        auth.required(role='admin')(
            GraphQLView.as_view(
                'graphql',
                schema=my_schema,
                graphiql=True,
                middleware=[
```

(continues on next page)

(continued from previous page)

```
                    MutationMiddleware(my_access_control)
                ]
            )
        )
    ),
    methods=['GET', 'POST']
)
```

1.3.2 Session

This service provides a Login and Logout.

Login

Check this example to how use the Login

```
from flask import Blueprint
from flask_restplus import Api, Resource
from flask_auth_service_mongo import api_rest

view_admin = Blueprint('view_admin', __name__)
api = Api(view_admin)


@api.route('/login')
class ApiLogin(Resource):
    def post(self):
        return api_rest.login(role='admin')
```

POST /admin/login

Example request:

```
POST /admin/login HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "username": "username",
    "password": "password"
}
```

Example response Ok:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "message": "ok",
    "data": {
        "change_password": true,
        "token_type": "Bearer",
        "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJleHAiOjE1ODQxMTMzNjUsImlhdCI6MTU4NDEwOTc2NSwic3ViIjoiNWU2NmE5NzZlYmM3NDY5YzZlYjg
↪",
```

(continues on next page)

(continued from previous page)

```
        "refresh_token":
↪"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ93OTViIiwidXVpZCI6IjBhNGU1Z",
        "expires_in": 60,
        "role": "role"
    }
}
```

Status Codes

- 200 OK – If **change_password == true** the user needs to change the password.

  expires_in time in minutes the token expires.

Example response Error:

```
HTTP/1.1 400 BAD REQUEST
Content-Type: application/json

{
    "message": "bad_request"
}
```

Logout

Check this example to how use Logout

```
from flask import Blueprint
from flask_restplus import Api, Resource
from flask_auth_service_mongo import api_rest, auth

view_admin = Blueprint('view_admin', __name__)
api = Api(view_admin)


@api.route('/logout')
class ApiLogout(Resource):
    @auth.required(role='admin')
    def post(self):
        return api_rest.logout()
```

POST /admin/logout

Example request:

```
POST /admin/logout HTTP/1.1
Host: example.com
Content-Type: application/json
Authorization: Bearer token_123_xD

{}
```

Example response Ok:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

(continues on next page)

(continued from previous page)

```
{
    "message": "ok"
}
```

Note: Remember to register your blueprint module in your `create_app()`.

Example:

```
def create_app():
    app = Flask(__name__)
    ...

    from views.admin import view_admin
    app.register_blueprint(view_admin, url_prefix='/admin')

    return app
```

Update password

Update password of current user

POST /admin/graphql

GraphQL: Mutation Reset Password User

Example request:

```
POST /admin/graphql HTTP/1.1
Host: example.com
Content-Type: application/json
Authorization: Bearer token_123_xD
```

```
mutation {
    update_password (input: {
        current_password: "current"
        new_password: "new_pass"
        password_confirmed: "new_pass"
    }) {
        ok
    }
}
```

Example response Ok:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "data": {
        "update_password": {
            "ok": true
        }
    }
}
```

1.3.3 CRUD User

This package provides the Create, Read, Update and Delete options. It's ready to integrate with `graphene`.

Schema

A simple example to how to define the schema for `user`

```
import graphene
from graphene.relay import Node
from graphene_mongo import MongoengineConnectionField
from flask_auth_service_mongo import schema as auth_schema


class QueryAdmin(graphene.ObjectType):
    node = Node.Field()
    user = MongoengineConnectionField(auth_schema.User)
    role = MongoengineConnectionField(auth_schema.Role)


class MutationAdmin(graphene.ObjectType):
    create_user = auth_schema.CreateUser.Field()
    update_user = auth_schema.UpdateUser.Field()
    delete_user = auth_schema.DeleteUser.Field()


schemaAdmin = graphene.Schema(
    query=QueryAdmin,
    mutation=MutationAdmin,
    auto_camelcase=False
)
```

View

A simple example to how to define `view`

```
from flask import Blueprint
from flask_graphql import GraphQLView
from utils.decorators import result_to_json
from schemas import schemaAdmin


view_admin = Blueprint('view_admin', __name__)


view_admin.add_url_rule(
    '/graphql',
    view_func=result_to_json(
        auth.required(role='admin')(
            GraphQLView.as_view(
                'graphql',
                schema=schemaAdmin,
                graphiql=True
            )
        )
```

(continues on next page)

(continued from previous page)

```
    ),
    methods=['GET', 'POST']
)
```

Create

POST /admin/graphql

GraphQL: Mutation create User

Example request:

```
POST /admin/graphql HTTP/1.1
Host: example.com
Content-Type: application/json
Authorization: Bearer token_123_xD
```

```
mutation {
    create_user (input: {
        username: "userAdmin"
        password: "aP#D.o"
        password_confirmed: "aP#D.o"
        role: "admin"
    }) {
        user {
            id
        }
    }
}
```

Example response Ok:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "data": {
        "create_user": {
            "user": {
                "id": "VXNlcjo1ZTBhNjFkYmFmYzhhNDI4MWIyMmM2ZGY="
            }
        }
    }
}
```

Read

GET /admin/graphql
GraphQL: Query Users

Example request:

```
GET /admin/graphql HTTP/1.1
Host: example.com
Content-Type: application/json
Authorization: Bearer token_123_xD
```

```
{
    user {
        edges {
            node {
                id
                username
                change_password
                role {
                    name
                }
            }
        }
    }
}
```

Example response Ok:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "data": {
        "user": {
            "edges": [
                {
                    "node": {
                        "id": "VXNlcjo1ZTMzMzUyNWYxNzBjZmM5MmNkYTgwYmE=",
                        "username": "userAdmin",
                        "change_password": false,
                        "role": {
                            "name": "admin"
                        }
                    }
                }
            ]
        }
    }
}
```

Update

POST /admin/graphql

GraphQL: Mutation update User

Example request:

```
POST /admin/graphql HTTP/1.1
Host: example.com
Content-Type: application/json
Authorization: Bearer token
```

```
mutation {
    update_user (input: {
        id: "VXNlcjo1ZTY2OWNlMWZkMzRlMTJmMjQwMjk5OTk="
        change_password: false
    }) {
        user {
            id
            change_password
        }
    }
}
```

Example response Ok:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "data": {
        "update_user": {
            "user": {
                "id": "VXNlcjo1ZTY2OWNlMWZkMzRlMTJmMjQwMjk5OTk=",
                "change_password": false
            }
        }
    }
}
```

Delete

POST /admin/graphql

GraphQL: Mutation delete User

Example request:

```
POST /admin/graphql HTTP/1.1
Host: example.com
Content-Type: application/json
Authorization: Bearer token_123_xD
```

```
mutation {
    delete_user (input: {
        id: "VXNlcjo1ZTBhNjNlOWFmYzhhNDI4MWIyMmM2ZTA="
    }) {
```

(continues on next page)

(continued from previous page)

```
        ok
    }
}
```

Example response Ok:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "data": {
        "delete_user": {
            "ok": true
        }
    }
}
```

Reset password

Generate a random password for the user

POST /admin/graphql
GraphQL: Mutation Reset Password User

Example request:

```
POST /admin/graphql HTTP/1.1
Host: example.com
Content-Type: application/json
Authorization: Bearer token_123_xD
```

```
mutation {
    reset_password (input: {
        id: "VXNlcjo1ZTQwNDY1M2QxYzA5OGM1MTc3Y2M2OWQ="
    }) {
        password
    }
}
```

Example response Ok:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "data": {
        "reset_password": {
            "password": "abcd"
        }
    }
}
```

1.3.4 Commands

This service provides commands.

Register module

Register the blueprint module in your `create_app()`.

Example:

```
def create_app():
    app = Flask(__name__)
    ...

    from flask_auth_service_mongo import command_auth_mongo
    app.register_blueprint(command_auth_mongo)

    return app
```

All commands

Show all available commands:

```
flask flask_auth --help
```

Create Role

Create a Role:

```
flask flask_auth role-new --help
```

Create User

Create a User:

```
flask flask_auth user-new --help
```

Clear tokens

Clear all tokens of the WhitelistToken:

```
flask flask_auth clear-tokens --help
```

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

HTTP ROUTING TABLE

/admin