Configuring Flask-JWT

Flask-JWT adds JWT functionality to Flask in an easy to use manner. It gives you a lot of functionality out of the box, but sometimes we want to modify some of the configuration. This document walks through how to:

- Change the authentication endpoint (by default, `/auth`);
- Change the token expiration time (by default, `5 minutes`);
- Change the authentication key name (by default, `username`);
- Change the authentication response body (by default, only contains `access_token`).

In addition, it covers how to retrieve the currently logged in user from any of our Flask app endpoints.

This tutorial assumes that you've followed the lectures and have set up Flask-JWT already! If you haven't done so yet, check out Section 5 of the Udemy course.

Before we begin

First, let's take a look at what we already have here. In our app.py file, we should already set up the JWT using the below code:

```
from flask_jwt import JWT
from security import authenticate, identity

jwt = JWT(app, authenticate, identity)  # /auth
```

And in our security.py file, we should have something like this:

```
from werkzeug.security import safe_str_cmp
from models.user import UserModel

def authenticate(username, password):
    user = UserModel.find_by_username(username)
    if user and safe_str_cmp(user.password, password):
        return user

def identity(payload):
    user_id = payload['identity']
    return UserModel.find_by_id(user_id)
```

Configuration

Authentication URL

If we want to change the url to the authentication endpoint, for instance, we want to use /login instead of /auth, we can do something like this:

```
app.config['JWT_AUTH_URL_RULE'] = '/login'
jwt = JWT(app, authenticate, identity)
```

Important: We added the second line of code to emphasize that we must change the JWT authentication URL first, before creating the JWT instance. Otherwise, our configuration won't take effect.

However, it is only required for configuring the auth URL, the following confurations will still take effect after requesting the JWT instance.

Token Expiration Time

```
# config JWT to expire within half an hour
app.config['JWT_EXPIRATION_DELTA'] = timedelta(seconds=1800)
```

Authentication Key Name

```
# config JWT auth key name to be 'email' instead of default
'username'
app.config['JWT_AUTH_USERNAME_KEY'] = 'email'
```

Authentication Response Handler

Sometimes we may want to include more information in the authentication response body, not just the access_token. For example, we may also want to include the user's ID in the response body. In this case, we can do something like this:

```
# customize JWT auth response, include user_id in response
body
from flask import jsonify
from flask_jwt import JWT

from security import authenticate, identity as
identity_function
jwt = JWT(app, authenticate, identity_function)

@jwt.auth_response_handler
def customized_response_handler(access_token, identity):
    return jsonify({
                        'access_token':
access_token.decode('utf-8'),
                        'user_id': identity.id
                    })
```

Remember that the identity should be what you've returned by the authenticate() function, and in our sample, it is a UserModel object which contains a field id. Make sure to only access valid fields in your identity model!

Moreover, it is generally not recommended to include information that is encrypted in the access_token since it may introduce security issues.

Error Handler

By default, Flask-JWT raises JWTError when an error occurs within any of the handlers (e.g. during authentication, identity, or creating the response). In some cases we may want to customize what our Flask app does when such an error occurs. We can do it this way:

```
# customize JWT auth response, include user_id in response
body
from flask import jsonify
from flask_jwt import JWT

from security import authenticate, identity as
identity_function
jwt = JWT(app, authenticate, identity_function)

@jwt.error_handler
def customized_error_handler(error):
    return jsonify({
                        'message': error.description,
                        'code': error.status_code
                    }), error.status_code
```

More

Retrieving User From Token

Another frequently asked question is: **how can I get the user's identity from an access token (JWT)**? Since in some cases, we not only want to guarantee that only our users can access an endpoint, but we may want to access the user's data as well.

For example, if you want to restrict the access to a certain user group, not for every user. In this case, you can do something like this:

```
from flask_jwt import jwt_required, current_identity


class User(Resource):

    @jwt_required()
    def get(self):   # view all users
        user = current_identity
        # then implement admin auth method
        # ...
```

Now this endpoint is protected by JWT. And you have access to the identity of the user who is interacting with this endpoint using current_identity from JWT.