django-rest-allauth Documentation

Release latest

Nov 26, 2020

CONTENTS

django-rest-allauth is an authentication package for django

CHAPTER

ONE

QUICK START

1. **Install Dep:** pip install django-rest-framework django-cors-headers requests
2. Add "django_rest_allauth" to your INSTALLED_APPS setting like this:

```
INSTALLED_APPS = [
    ...
    'django_rest_allauth',
    'rest_framework',
    'rest_framework.authtoken',
    'corsheaders',
]
```

3. Include corsheaders in your middleware in settings.py:

```
MIDDLEWARE = [

    'corsheaders.middleware.CorsMiddleware',

    ...
]
```

4. Include DEFAULT_AUTHENTICATION_CLASSES and DEFAULT_PERMISSION_CLASSES in settings.py:

```
REST_FRAMEWORK = {

    'DEFAULT_AUTHENTICATION_CLASSES': (

        'rest_framework.authentication.TokenAuthentication',
    ),
    'DEFAULT_PERMISSION_CLASSES': (

        'rest_framework.permissions.IsAuthenticated','rest_framework.permissions.
↪AllowAny' )
}
```

5. Include the django_rest_allauth URLconf in your project urls.py like this:

```
path('django-rest-allauth/', include('django_rest_allauth.api.urls')),
```

6. **Set how you want your corsheaders in settings.py or whitelist your url::** CORS_ALLOW_CREDENTIALS = True

   CORS_ORIGIN_ALLOW_ALL = True

   SITE_ID = 1

7. Run `python manage.py makemigrations python manage.py migrate` to create the DjangoRestAllAuth models.
8. Start the development server and visit http://127.0.0.1:8000/admin/ to create a poll (you'll need the Admin app enabled).
9. Visit http://127.0.0.1:8000/django-rest-allauth/ to participate in the django_rest_allauth.

Url Endpoints This package uses token authentication

- token/createuser
- token/login
- token/getuser
- token/edituser
- token/changepassword
- token/resetpasswordcode
- token/resetpassword
- token/logout

createuser This is to create a user

Method: POST

Authorization: AllowAny

fields: { "email": "", "username": "", "password": "", "first_name": "", "last_name": "" }

Optional fields are username, first_name and last_name

login This is to login a user, it returns userdetails with token along

Method: POST

Authorization: AllowAny

fields: { "email": "", "username": "", "password": "", }

Optional either username or email can be used, or both. It returns response with token along with it for authentication

getuser This is to get user details, it returns an object with the user details

Method: GET

Authorization: Token

edituser This is to edituser details

Method: PATCH

Authorization: Token

fields: { "email": "", "username": "", "first_name": "", "last_name": "", }

All fields are optional, only input the field you want to change

changepassword This is to change user's password

Method: POST

Authorization: Token

fields: { "old_password": "", "new_password": "", }

If the old password is correct, it changes the user's password to the new one.

resetpasswordcode This will generate a code for the user and send back as response, the code can be sent to the user's email or sms, the next end point will be to accept the code and email

Method: POST

Authorization: AllowAny

fields: { "email": "", "resetcode": "" }

resetpassword This will accept the code, email and password, if it's correct, the password will be changed

Method: POST

Authorization: AllowAny

fields: { "email": "", "resetcode": "", "password": "" }

logout

This will delete the user's token

Method: POST

Authorization: AllowAny

authenticatesocialuser

To authenticate a user with social media (facebook and google)

Method: POST

Authorization: AllowAny

fields: { "provider": '', "token": "", "email": "", "username": "", "first_name": "", "last_name": "", "social_id": "", "profile_pic": "" } These fields are coming from google/facebook response.

provider field accepts 'Facebook' or 'Google' which ever provider being used.

token is the access_token returned from google or facebook.

social_id: for facebook it is "id" that is returned, for google, it is user_id.

optional fields are username, first_name, last_name and profile_pic.