Authentication and Authorization:

The APIs, which are developed up to this can be accessed by every one.

By using ENDPOINT, any one can create a new resource,can modify and delete any

existing resource. It causes security problems. To provide Security for our API, we should

go for Authentication and Authorization.

Authentication:

The process of validating user is called authentication. Most of the times we can perform

authentication by using username and password combination or by using tokens etc

DRF provides several inbuilt authentication mechanisms

1) Basic Authentication

2) Session Authentication

3) Token Authentication

4) JWT(Json window Token) Authentication

etc

Note: By using DRF, we can implement our own custom authentication mechanism also

Authorization:

The process of validating access permissions of user is called authorization.

DRF provides several permission classes for authorization.

1) AllowAny

2) IsAuthenticated

3) IsAdminUser

4) IsAuthenticatedOrReadOnly

5) DjangoModelPermissions

6) DjangoModelPermissionsOrAnonReadOnly

etc

Note: After authentication we have to perform authorization.

Token based Authentication:

 Every request will be authenticated by using Token, which is nothing but Token

Authentication.

 TokenAuthentication is best suitable for native desktop clients and mobile clients.

 To implement TokenAuthentication, We have to use 3rd party application

'rest\_framework.authtoken',which is responsible to generate and validate required

tokens. This application is the part of DRF

Steps to implement TokenAuthentication:

1) We have to include authtoken application in INSTALLED\_APPS list inside settings.py

file.

1) INSTALLED\_APPS = [

2) ....

3) 'rest\_framework',

4) 'rest\_framework.authtoken',

5) 'testapp'

6) ]

2) Perform migrations so that the required tables of authtoken application will be

created in the database.

The table name is Tokens.

3) We can generate Tokens in the backend from admin interface by selecting required

user.

4) User also can send a request to authtoken application to generate token explicitly. For

this auth application url-pattern we have to configure in urls.py file.

1) from rest\_framework.authtoken import views

2) urlpatterns = [

3) .....

4) url(r'^get-api-token/', views.obtain\_auth\_token,name='get-api-token'),

5) ]

We can send request to this authtoken application to get token as follows

http POST http://127.0.0.1:8000/get-api-token/ username="naresh"

password="naresh123"

 authtoken application will validate this username and password.After valiation, it will

check is any token already generated for this user or not. If it is already generated then

return existing token from Tokens table.

 If it is not yet generated, then authtoken application will generate Token and save in

Tokens table and then send that token to the client.

Note: From the postman also,we can send the request. But username and password we

have to provide in Body section.

Enabling Authentication and Authorization (Permissions) for

View Class:

 Upto this just we tested authtoken application to generate and store Tokens

We have to enable authentication and authorization for our view classes either locally

OR globally.

Enabling Locally:

Our application may contain several view classes. If we want to enable authentication and

authorization for a particular view class then we have to use this local approach.

1) from rest\_framework.authentication import TokenAuthentication

2) from rest\_framework.permissions import IsAuthenticated

3) class EmployeeCRUDCBV(ModelViewSet):

4) ...

5) authentication\_classes=[TokenAuthentication,]

6) permission\_classes=[IsAuthenticated,]

Note: Now,if we want to access ENDPOINT compulsory we should send Token, otherwise

we will get 401 Unauthorized error response.

1) D:\>http http://127.0.0.1:8000/api/

2) HTTP/1.0 401 Unauthorized

3) Allow: GET, POST, HEAD, OPTIONS

4) Content-Length: 58

5) Content-Type: application/json

6) Date: Mon, 21 Jan 2019 11:43:07 GMT

7) Server: WSGIServer/0.2 CPython/3.6.5

8) Vary: Accept

9) WWW-Authenticate: Token

10) X-Frame-Options: SAMEORIGIN

11)

12){

13) "detail": "Authentication credentials were not provided."

14)}

How to Send the Request with Token:

1) D:\>http http://localhost:8000/api/ "authorization:Token 3639020972

202cc1d25114ab4a5f54e6078184a4"

2)

3) HTTP/1.0 200 OK

4) Allow: GET, POST, HEAD, OPTIONS

5) Content-Length: 136

6) Content-Type: application/json

7) Date: Mon, 21 Jan 2019 11:45:13 GMT

8) Server: WSGIServer/0.2 CPython/3.6.5

9) Vary: Accept

10) X-Frame-Options: SAMEORIGIN

11)

12) [

13) {

14) "eaddr": "Mumbai",

15) "ename": "ramesh",

16) "eno": 100,

17) "esal": 1000.0,

18) "id": 1

19) },

20) {

21) "eaddr": "Hyderabad",

22) "ename": "naresh",

23) "eno": 200,

24) "esal": 2000.0,

25) "id": 2

26) }

27) ]

Enabling Globally:

 If we want to enable authentication and authorization for all view classes, we have to

use this approach.

 We have to add the following lines inside settings.py file.

1) REST\_FRAMEWORK={

2) 'DEFAULT\_AUTHENTICATION\_CLASSES':('rest\_framework.authentication.TokenAut

hentication',),

3) 'DEFAULT\_PERMISSION\_CLASSES':('rest\_framework.permissions.IsAuthenticated',)

4) }

Q) What is use of 'AllowAny' Permission Class?

 AllowAny is the default value for the permission class.

 If we configure permission classes globally then applicable for all view classes. If we

don't want authorization for a particular class then we should use this 'AllowAny'

permission Class.

1) from rest\_framework.authentication import TokenAuthentication

2) from rest\_framework.permissions import AllowAny

3) class EmployeeCRUDCBV(ModelViewSet):

4) queryset=Employee.objects.all()

5) serializer\_class=EmployeeSerializer

6) authentication\_classes=[TokenAuthentication,]

7) permission\_classes=[AllowAny,]

To access this end point now authentication and authorization is not required. Any one

can access.

Various Possible Permission Classes:

DRF provides the following pre defined permission classes

1) AllowAny

2) IsAuthenticated

3) IsAdminUser

4) IsAuthenticatedOrReadOnly

5) DjangoModelPermissions

6) DjangoModelPermissionsOrAnonReadOnly

1) AllowAny:

 The AllowAny permission class will allow unrestricted access irrespective of whether

request is authenticated or not.

 This is default value for permission-class. It is very helpful to allow unrestricted access

for a particular view class if global settings are enabled.

2) IsAuthenticated:

 The IsAuthenticated permission class will deny permissions to any unauthorized user.

ie only authenticated users are allowed to access endpoint.

 This permission is suitable, if we want our API to be accessible by only registered

users.

Note:

We can send Token in postman inside Headers Section

Key: Authorization

Value: Token 3639020972202cc1d25114ab4a5f54e6078184a4

3) IsAdminUser:

 If we use IsAdminUser permission class then only AdminUser is allowed to access.i.e

the users where is\_staff property is True.

 This type of permission is best suitable if we want our API to be accessible by only

trusted administrators.

 If the user is not admin and if he is trying to access endpoint then we will get 403

status code error response saying:

{

"detail": "You do not have permission to perform this action."

}

4) IsAuthenticatedOrReadOnly:

To perform read operation (safe methods:GET,HEAD,OPTIONS) authentication is not

required. But for the remaining operations (POST,PUT,PATCH,DELETE) authentication

must be required.

 If any person is allowed to perform read operation and only registered users are

allowed to perform write operation then we should go for this permission class.

 Eg: In IRCTC application, to get trains information (read operation) registration is not

required. But to book tickets (write operation) login must be required.

5) DjangoModelPermissions:

 This is the most powerful permission class. Authorization will be granted iff user is

authenticated and has the relevant model permissions.

 DjangoModelPermissions = Authentication + Model Permissions.

 If the user is not authenticated(we are not providing token) then we will get 401

Unauthorized error message saying

{

"detail": "Authentication credentials were not provided."

}

If we are providing Token (authenticated) but not having model permissions then we can

perform only GET operation. But to perform POST,PUT,PATCH,DELETE compulsory model

permissions must be required,otherwise we will get 403 Forbidden error message saying

{

"detail": "You do not have permission to perform this action."

}

How to provide Model Permissions:

To perform POST operation the required model permission is 'add'

To perform PUT,PATCH operations the required model permission is 'change'

To pderform DELETE operation the required model permission is 'delete'

We have to provide these model permissions in admin interface under User permissions:

testapp | employee | Can change employee

testapp | employee | Can add employee

testapp | employee | Can delete employee

Note: DjangoModelPermissions class is more powerful and we have complete control on

permissions.

6) DjangoModelPermissionsOrAnonReadOnly:

It is exactly same as DjangoModelPermissions class except that it allows

unauthenticated users to have read-only access to the API

1) from rest\_framework.viewsets import ModelViewSet

2) from testapp.models import Employee

3) from testapp.serializers import EmployeeSerializer

4) from rest\_framework.authentication import TokenAuthentication

5) from rest\_framework.permissions import IsAuthenticated,AllowAny,IsAdminUser,I

sAuthenticatedOrReadOnly,DjangoModelPermissions,DjangoModelPermissionsOr

AnonReadOnly

6) class EmployeeCRUDCBV(ModelViewSet):

7) queryset=Employee.objects.all()

8) serializer\_class=EmployeeSerializer

9) authentication\_classes=[TokenAuthentication,]

10) permission\_classes=[DjangoModelPermissionsOrAnonReadOnly,]