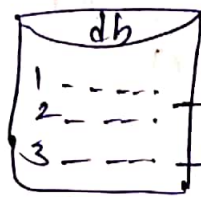


# Security

- \* If we want to get all records from database we will send non-id based url then successfully we are getting data from database.
- \* Same like all CRUD operations behaviour, if we want to do by using id and non-id based url's we can do successfully

For example:-



create new  
update existing 1  
deleting existing 1

By using this url : `http://127.0.0.1:8000/lapi/emp`, by using this url anybody can interact with our database

- \* Here, any person can get our data, may delete or modify our data, it means anything they may do unnecessarily on our database by using our api's

Bcz till now we are developing the api's without having

security

How to overcome:-

As a REST API developer we need to provide security to our API's to control the 3rd party persons, without doing the unwanted actions

To providing security to our ~~API's~~ database we are using the 'authentication' and 'authorization' mechanisms

What is Authentication:-

The process of validating the user credentials is called as Authentication.

for example :-

→ If we want to gmail account, it requires user name and password

→ If you are an employee in any software company, then security guard will ask you access card, without it he may not allow you into organisation. This is nothing but Authentication

→ Most of the times we are checking the authentication by using username & password only. Sometimes by using tokens also we will check if the user is valid or not

DRF (Django Rest Framework) inbuilt authentication classes:-

→ DRF provides "authentication" module for providing the Authentication (user valid or not) to our API's. This module provides several classes like below

1. BasicAuthentication } → inbuilt

2. SessionAuthentication }

3. JWT (Json Web Token) authentication → 3<sup>rd</sup> party class

4. TokenAuthentication } → inbuilt

→ we will import like below

from rest\_framework.authentication import BasicAuthentication



## Authorization :-

The process of validating the access permissions of user is called Authorization, means here user what can do or what can't do on our database (user roles)

for example :-

- If we have any one bank account by telling our account number, account person name...etc. we may get our balance details if we want, but not others account
- If we are asking bank employees about any others bank account details, then they will not tell others info bcz we are not authorized person

NOTE :- First we have authentication then only we have to perform authorization

Q : Is every authentication person is an authorization person or not ?

sol :- NO, every authentication person, may or may not be having all permissions to do something (50-50 chances)

Q : Is every authorization person is an authentication person or not?

sol :- Yes, every authorization person <sup>should</sup> be authentication person it means without user name & password no one can do something (100% chance)

## DRF Authorization (permission) classes :-

Django rest framework provides several inbuilt permission classes to providing or checking the user permissions, what they can do or what can't do purpose

- It is providing "permissions" module, this module providing several permissions classes to checking the user roles

- for example :-
1. Allow Any
  2. IsAuthenticated
  3. IsAdminUser
  4. IsAuthenticated Or Read Only

### Allow Any :-

- If we are using allow any permission ~~per~~ class in our project then by default all persons can have all <sup>permissions</sup> ~~persons~~ to access our API's, here without username and password also any body can do CRUD operations on our database
- Allow Any is default class

### IsAuthenticated :-

- By using IsAuthenticated permission class only authenticated person can access our API's. Here valid person can do all CRUD operations by using our API on our database
- If user is not authenticated (invalid username & password) then it returns some exception like '401-unauthorized access'

### IsAdminUser :-

- By using IsAdminUser permission class, if the user is admin user then only this user allowed to perform CRUD operations, by using our API
- Even though user having valid user name & password, if not admin person then it returns exception like 401 unauthorized like you don't have permission to do this operation

### IsAuthenticated Or Read Only :-

- By using this class, without user name & password also any person can read the data only possible because of ~~Read~~ Read only



→ without username & password he is trying to do remaining CURD operations not possible - only authenticated persons can do all CURD operations on our API

→ we can import the permission classes like below

```
from rest-framework.permissions import Authenticated
```

Different ways to providing security :-

→ By using DRF we can provide security in two levels

1. project level (settings.py) (or) Global security
2. ~~local level~~ (2. class level (views.py) local security)

→ for project level (we are providing Global) security purpose, open settings.py and create one dictionary object variable which name as 'REST\_FRAMEWORK' with required authentication and permission classes

→ Take Project level security code :

```
REST_FRAMEWORK = {
```

```
    'DEFAULT_AUTHENTICATION_CLASSES':
```

```
        ('rest-framework.authentication.BasicAuthentication',)
```

```
    'DEFAULT_PERMISSION_CLASSES':
```

```
        ('rest-framework.permissions.IsAuthenticated',),
```

```
}
```

→ To providing class level security, open views.py and import required authentication and permission classes and represent these classes inside user defined 'view class' by using some predefined fields

```
from rest-framework.authentication import BasicAuthentication
```

```
from rest-framework.permissions import IsAuthenticated
```

```
class EmployeeViewData(MVS):
```

query set = - - - - -

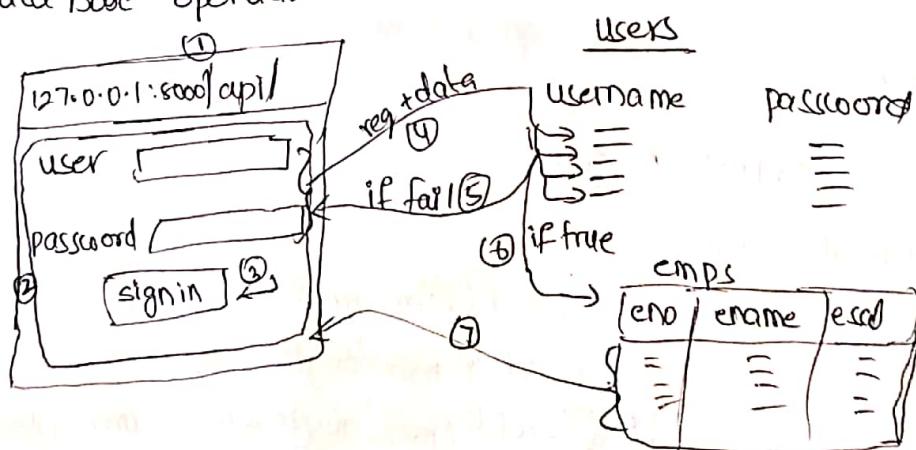
serializer - class = - - - - -

authentication - classes = (BasicAuthentication,)

permission - classes = (IsAuthenticated,)

### Basic Authentication :-

- By using BasicAuthentication model, when users enter any url then immediately it returns one default login page, to identifying the user credentials valid or not
- this login page accepting the user name and password to validating the user is valid or not
- the application has to verify the user credentials, if they are not valid then it returns same login page to take one more time user credentials for valid data.
- Here, when the user entered credentials request goes to django users table along with data and checking all, that user is existing or not
- If the user is valid then it redirected to 'api' access for getting the data base operations



\* create a project to provide security to our api's by using Basic Authentication

step-1: project name : BasicAuthentication-project

step-2: app name : BasicAuthentication-app

step-3: Data base name: Basicdb

step-4: configure database in settings.py and add our application name and rest-framework in applications in settings.py

step-6: (open models.py and write the below-code) provide BasicAuthentication settings inside 'INSTALLED\_APPS' section

step-5: create security settings inside settings.py file by using dictionary, REST-FRAMEWORK object

```
REST_FRAMEWORK = {
```

```
    'DEFAULT_AUTHENTICATION_CLASSES':
```

```
        ('rest_framework_authentication.BasicAuthentication',),
```

```
    'DEFAULT_PERMISSION_CLASSES':
```

```
        ('rest_framework.permissions.IsAuthenticated',),
```

```
}
```

step-6: models.py

```
from django.db import models
```

```
class Emp(models.Model):
```

```
    empid = models.IntegerField(primary_key=True)
```

```
    empname = models.CharField(max_length=100)
```

```
    salary = models.DecimalIntegerField(max_digits=10, decimal_places=2)
```

```
    def __str__(self):
```

```
        return self.empname
```



7. open admin.py

from django.contrib import admin

from models import Emp

class EmpAdmin (admin. Model Admin):

list\_display = ['empid', 'empname', 'salary']

admin. site. register (Emp, EmpAdmin)

step-8:- open serializers.py

from models import Emp

from rest\_framework import <sup>ers</sup> ~~Serializers~~

class Emp~~Serializer~~ (Serializers. Model Serializer)

class Meta:

model = Emp

fields = '\_\_all\_\_'

step-9: open views.py

from django import views

from models import ~~model~~ Emp

from ~~Emp~~ Serializers import Emp~~Serializer~~

from rest\_framework import <sup>viewsets</sup> ~~Views~~ sets

class Emp\_Viewset (viewsets. Model Viewset):

queryset = Emp. objects. all()

serializer\_class = Emp~~Serializer~~

step-10: open project level urls.py

from django.urls import path, include

from django.contrib import admin



url patterns = [

path ('admin/', admin.site.urls)

path ('api/', include ('Basic-Authentication-app.urls'))

]

step-11: open applevel urls.py

from django.conf.urls import url

from .views import EmpView

from rest\_framework.routers import DefaultRouter

router = DefaultRouter()

router.register ('emp', EmpViewSet)

url patterns = [

url (r'', include (router.urls))

]

step-12: run makemigrations, migrate, create superuser commands  
and runserver command

step-13: execute api from browser

if credentials are valid

↓  
[ { "empid": 10,  
"empname": "Harish",  
"salary": "35000.00",  
},  
]

]

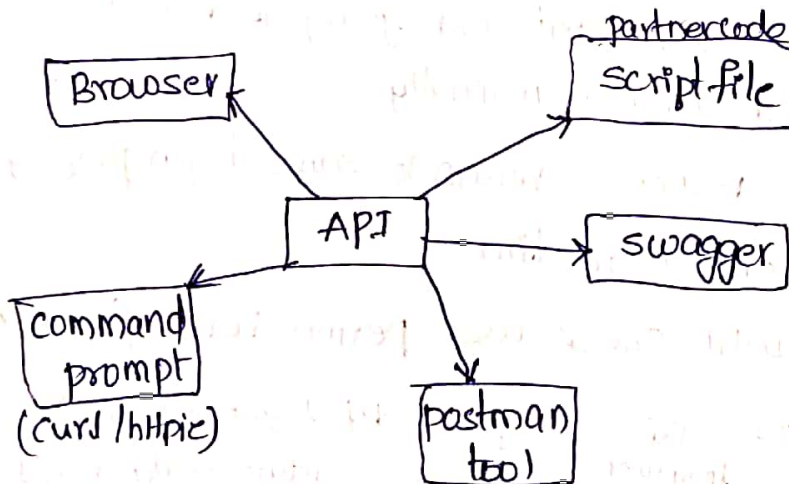
→ if credentials are not valid, we will get response like "401 Unauthorized"  
msg: Authentication details are not provided

## PASTMAN TOOL :-

- postman is a third party tool, we need to download this, use and install it
- it is a third party api testing tool, by using this we can test java api's, .Net api's, Django api's
- it is not common to any particular language related api's
- postman is always going to check end point url only, it is not checking this end point url is developed by using which language or platform
- it is a api independent tool, supported to all framework related APIs
- we will download this tool and install it

<https://www.getpostman.com/downloads/>

Q How many ways we are testing the APIs ?  
A our APIs we can test like multiple ways





## Basic Authentication Drawbacks:-

→ By

1. user name and password will send to the server in BASE64 format which is not secured
2. our Credentials are not going to store in the form of cookies, it is going in the form of request

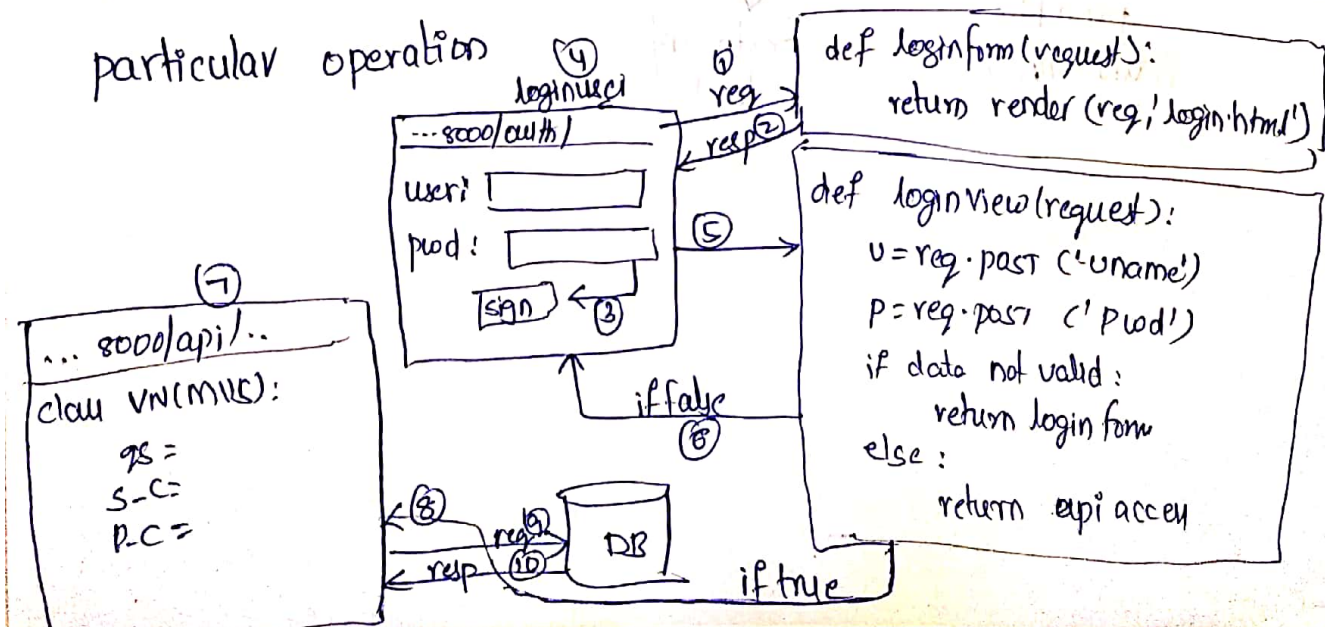
→ Using Basic Authentication we cannot customize look & feel of authentication form

## Session Authentication:-

By using Session Authentication we are going to validate user credentials, here we are developing our own authentication form to accept the user credentials. When user entered any credentials then we will capture these user input and going to users table and validating that user is existed or not manually

→ if not valid then response return to same login page, and to take user input one more time

→ if valid, then we will check user permissions for access doing a particular operation



### Authenticate() :-

- By using this `authenticate()`, user input (username, password) can validate and returns response
- If user validate it returns object, if not validate it returns `nonvalde`
- This method taking username and password and going to default user table and validating here that user is existing or not

### login() :-

- using `login()` method user is having particular permissions on db checking
- User not having permissions it returns exception, if user having the permission it unlock the database and can give the permission to access the api's for this user

\* Here `authenticate()`, is checking first user is authenticate user or not, if authenticated `login()` method checking permissions of user

\* This both methods available in default 'auth' application

```
from django.contrib.auth import authenticate, login
```

### Project :-

- create a project to provide security to APIs by using session authentication

[copy & paste from Basic Authentication] except `views.py` & project level `urls.py`

### open views.py

```
from django.shortcuts import render, redirect
from .models import Emp
from .serializers import empserializer
from rest_framework import viewsets
from rest_framework.authentication import SessionAuthentication
from rest_framework.permissions import IsAuthenticated, IsAdminUser
```



```
from django.contrib.auth import authenticate, login
```

```
def login_form(request):
```

```
    return render(request, 'login.html')
```

```
def login_user(request):
```

```
    username = request.post['uname']
```

```
    password = request.post['pwd']
```

```
    user = authenticate(username=username, password=password)
```

```
    if user is not None:
```

```
        login(request, user)
```

```
        return redirect('/api/')
```

```
    else:
```

```
        return redirect('/auth/')
```

```
class EmpViewSet(viewsets.ModelViewSet):
```

```
    queryset = Emp.objects.all()
```

```
    serializer_class = EmpSerializer
```

```
    authentication_classes = (SessionAuthentication)
```

```
    permission_classes = (IsAdminuser)
```

create login.html inside template folder

```
<html> <head> <title> Session Based project </title>  
</head>
```

```
<style>
```

```
body { background-color: bisque; }
```

```
form { padding: 10px;  
        margin: 10px;  
    }
```

```
h1 { padding: 10px; }
```

```
</style>
```

```

<body>
  <form action = '/loginuser/' method = 'post'>
    { '% csrf-token %' }
    <div> please provide your credentials here... </div>
    <table>
      <tr>
        <th> Username </th>
        <td> <input type='text' name = 'uname'> </td>
      </tr>
      <tr>
        <th> Password </th>
        <td> <input type = 'password' name = 'pwd'> </td>
      </tr>
      <tr>
        <th> </th>
        <td> <input type = 'submit' value = 'submit' > </td>
      </tr>
    </table>
  </body>
</head>

```

Note:- If form containing action attribute with some value then after submitting the form data, taking the form data and goes to particular destination represented by 'action' attribute. If 'action' not available then it goes to same view, where the request is coming and stored their.

open project level urls.py :-

```

from django.contrib import admin
from django.urls import path
from django.conf.urls import url, include
from Session-Authentication-app import views

urlpatterns = [
    path('auth/', views.login_form),

```



```

path ('login user/', views.login_user),
url (r'^api/', include ('session-Authentication-App.urls')),
path ('admin/', admin.site.urls),
]

```

→ Execute the runserver, migrate & makemigration commands

127.0.0.1:8000/auth/

Please provide the credentials.

username:

password:

→ enter username, password and click submit

127.0.0.1:8086/api/

API Root

```

{
  "emp_viewset": "http://127.0.0.1:8086/api/emp_viewset/"
}

```

\* click the above url then it redirect to router level url and it calls our user defined view

127.0.0.1:8086/api/emp\_viewset/

Emp Viewset2 list

```

[
  {
    "empid": 20,
    "empname": 'Virat',
    "salary": "40000.00"
  },
  ...
]

```