JWT process

JWT authentication is not inbuilt authentication of Django rest framework we have to install seperately 3rd party modules

→ There are several third party packages are available to implement JWT authentication for DRF

1. djangorestframework-JWT
2. djangorest-framework-simplejWT

How to implement JWT authentication in our project :-

step-1: we have to install 'django rest framework-jwt' module by using pip command

```
pip install djangorestframework-jwt
```

step-2: Importing the predefined jwt views from jwt application inside urls.py

```
from rest_framework_jwt.views import obtain_jwt_token, ~~obtain~~ verify_jwt_token
```

step-3: create some url patterns to perform access token access token and verify token in urls.py

```
urlpatterns = [
    ----  -----
  url(r'^api-token-auth/', obtain_jwt_token),
  url(r'^api-token-verify/', verify_jwt_token),
]
```

Enabling JWT Authentication class in Views.py

```
from rest-framework_jwt.authentication import JSONWebTokenAuthentication
class Emp_Viewset(ModelViewSet):
    queryset = employee.objects.all()
    serializer_class = EmployeeSerializer
```

permission_classes = (IsAuthenticated,)

authentication_classes = (JSONWebTokenAuthentication,)

How jwt work :

The jwt is just an authorization token that should be included in every request, this authentication associated with the following token management terminology

1. **Access token** :-

this token can used to access our API end point, the default expire time is 5 minutes, if we want to change this expiration ~~time~~, then we can change according to our requirements.

for this we need to write overriding logics in settings.py file by using "JWT_AUTH" dictionary object

After accessing the JWT token value when we are sending this token value along with APIs, then inside headers select key as 'Authorization' and value as 'prefix JWT, suffix JWT' value

To overriding default JWT values, then add settings like below

```
import datetime
JWT_AUTH = {

         { "JWT_AUTH_HEADER_PREFIX": 'JWT',
default  {  "JWT_EXPIRATION_DELTA": datetime.timedelta(seconds=300),

                          (or)

         { "JWT_AUTH_HEADER_PREFIX": 'SRI',
our own  {  "JWT_EXPIRATION_DELTA": datetime.timedelta(seconds=120),
```

We can generate access token by using following configurations

```
from rest-framework-jwt .views import obtain-jwt-token
urlpatterns = [

        url(r'^api-token-auth/', obtain-jwt-token),

    ]
```

Verify token :-

We can verify whether weather the token is expire or not by using the following url pattern

```
from rest-framework-jwt .views import verify-jwt-token
urlpatterns = [

        url(r'^api-token-verify/', verify-jwt-token),
    ]
```

→ passing a token to the verification end point will return 200 res response, if the token is valid, otherwise it will return 400 bad request, as well as an error identifying key why the token was invalid

NOTE:- This tokens and expiration time concept is required to provide security. JWT token hold more information, then Tokens of tokenAuthentication

JWT testing :-

To access the API to getting all records

select GET : http:- get url .... in url path section

click on headers and enter key as authorization and value as JWT <Jwt token value> ↵, then we will get output success

```
[ {    "empid" : 30,
       "empname" : "SRINIVAS"
   }
      :
  ]
```

→ After 5 min ~~w~~ if we ~~are~~ are trying to access the API for data with same access key value then we are getting exception like <u>401</u> unauthorized

<u>status</u>

```
{ "detail": "signature has expired"
}
```

→ for verification token, we will give input like below

```
{
  "token": "JWT token value".
}
```

then we will get response 200 OK

```
{ "token": "JWT token value",
}.
```

→ If response is failed, it means token value expired, then we will get 400 bad request

```
{ "non-fields-errors": [ "signature has expired'
                       ]
```

<u>project</u>:-

1. project name :- JWT-project
2. application name: JWT-app
3. database name: JWT-db

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

```
JWT-App.apps.JwtAppConfig
ROOT_URLCONF = 'JWT-Project.urls'
import datatime
JWT-AUTH = {
        'JWT_AUTH-HEADER-PREFIX': 'SRI',
        'JWT_EXPIRATION-DELTA': datetime.timedelta(seconds=120),
  }
```

open models.py

```
from django.db import models
class Employee (models.Model):
    empid = models. IntegerField(primary_key = True)
    empname = models. CharField (max_length = 100)
    salary = models. DecimalField (max_digit = 10, decimal_places=2)
    location = models. CharField (max_length = 100)
```

open serializers.py

```
from rest_framework.serializers import ModelSerializer
from .models import Employee
class EmployeeSerializer (ModelSerializer):
        class Meta:
                model = Employee
                fields = '__all__'
```

open views.py

```
from django.shortcuts import render
from .models import Employee
from .serializers import EmployeeSerializer
from rest_framework.viewsets import ModelViewSet
from rest_framework.~~viewsets~~ ~~import~~ permissions import
                        IsAuthenticated, IsAdminUser
from rest_framework_jwt.authentication import JSONWebTokenAuthentication
class Emp_ViewSet (ModelViewSet):
    queryset = Employee.objects.all()
    serializer_class = EmployeeSerializer
    permission_classes = (IsAuthenticated)
    authentication_classes = (JSONWebTokenAuthentication)
```

open admin.py

```
from django.contrib import admin
from .models import Employee
class EmployeeAdmin(admin.ModelAdmin):
      list_display = ['empid', 'empname', 'salary', 'location']
admin.site.register(Employee, EmployeeAdmin)
```

open urls.py

```
from django.contrib import admin
from django.urls import path ,include
from django.conf.urls import url
from JWT_App import views
from rest_framework.routers import DefaultRouter
router = DefaultRouter()
router.register('emp' ,views.Emp_Viewset)
from rest_framework_jwt.views import obtain_jwt_token, Verify_jwt_token
urlpatterns =[
    path('admin' ,admin.site.urls),
    path('api/', include (router.urls)).
    url(r'^api-token-auth/',.obtain_jwt_token),
    url (r'^api-token-verify/', Verify_jwt_token)
 ]
```