**REST API (Django Rest Framework-DRF)**

**Session-9**

**DRF api\_view:**

* api\_view allow us to define functions that match standard http methods like GET, POST, PUT ,PATCH etc.
* The main functionality of api\_view decorator is which takes a list of HTTP methods that your view should respond.
* api\_view will simplify the view logic.

**resquest.data :**

* request. **data returns the parsed content of the request body**. This is similar to the standard request. It includes all parsed content.

**Response:**

* Response objects are initialized with data, which should consist of native python primitives.

**Headers:**

* HTTP Headers are an important part of the API request and response as they represent the meta-data associated with the API request and response.
* Headers carry information for: Request and Response Body

**Ex with CRUD api\_view using function based view:**

**Settings.py:**

INSTALLED\_APPS = [  
 **'django.contrib.admin'**,  
 **'django.contrib.auth'**,  
 **'django.contrib.contenttypes'**,  
 **'django.contrib.sessions'**,  
 **'django.contrib.messages'**,  
 **'django.contrib.staticfiles'**,  
 **'myapp.apps.MyappConfig'**,  
 **'rest\_framework'**,  
  
 **'myapp2'**,  
]

**Models.py:**

**from** django.db **import** models  
  
*# Create your models here.***class** Trainer(models.Model):  
 name=models.CharField(max\_length=20)  
 address=models.CharField(max\_length=20)  
 mail = models.CharField(max\_length=20)  
 age = models.IntegerField()

**admin.py:**

**from** django.contrib **import** admin  
**from** myapp2.models **import** Trainer  
*# Register your models here.*@admin.register(Trainer)  
**class** TrainerAdmin(admin.ModelAdmin):  
 list\_display = [**'id'**,**'name'**,**'address'**,**'mail'**,**'age'**]

**Now go to terminal and then type the following commands**

* Python manage.py makemigrations
* Python manage.py migrate
* Python manage.py createsuperuser
* Now run server: python manage.py runserver
* Now go to browser: <http://127.0.0.1:8000/admin>
* Insert few records

Create a new python file with the name serializers.py.

**serializer.py:**

**from** rest\_framework **import** serializers  
**from** myapp2.models **import** Trainer  
  
**class** TrainerSerializer(serializers.ModelSerializer):  
 **class** Meta:  
 model=Trainer  
 fields=[**'id'**,**'name'**,**'address'**,**'mail'**,**'age'**]

**views.py:**

**from** django.shortcuts **import** render  
**from** rest\_framework.decorators **import** api\_view  
**from** rest\_framework.response **import** Response  
**from** myapp2.models **import** Trainer  
**from** myapp2.serializers **import** TrainerSerializer  
  
*# Create your views here.*@api\_view([**'GET'**,**'POST'**,**'PUT'**,**'DELETE'**])  
**def** trainer\_api(request):  
 **if** request.method == **'GET'**:  
 id = request.data.get(**'id'**)  
 **if** id **is not None**:  
 tr=Trainer.objects.get(id=id)  
 serializer=TrainerSerializer(tr)  
 **return** Response(serializer.data)  
 tr=Trainer.objects.all()  
 serializer=TrainerSerializer(tr,many=**True**)  
 **return** Response(serializer.data)

**urls.py :**

**from** django.contrib **import** admin  
**from** django.urls **import** path  
**from** myapp2 **import** views  
  
urlpatterns = [  
 path(**'admin/'**, admin.site.urls),  
 path(**'trainer/'**,views.trainer\_api),  
]

Create a new python file with the name test.py inside the application.

**test.py:**

**import** requests  
**import** json  
  
URL=**" http://127.0.0.1:8000/trainer/"  
  
def** get\_record(id=**None**):  
 data={}  
 **if** id **is not None**:  
 data={**'id'**:id}  
 jsondata=json.dumps(data)  
  
 headers={**'content-Type'**:**'application/json'**}  
 r=requests.get(url=URL,headers=headers,data=jsondata)  
 data=r.json()  
 print(data)  
  
*#get\_record(1)*get\_record()