

## Django - Rest API Integration :-

- In django we developed some projects like CRUD operation projects if any third party application wants to communicate our <sup>CRUD</sup> application then we are saying, sorry I am not developing any REST APIs, so your unable to communicate with my application, from your application
- Then this third party application user can communicate with other django application
- same like one more java application person, also wants to communicate with our application, bcz of no REST APIs this java person also go to another application for communicating.
- like this we are saying if any third party persons want to communicate our application then open the browser and send request to my application directly, then I will return response to you successfully

### Drawback :-

- so 3rd party persons can't use our applications, bcz of no APIs
- so, our application is not getting the more popularity

### How to Overcome :-

- to overcome this problem we need to provide APIs for our application, so that all third party applications can successfully interacted with our application, then finally our application getting more demanding

### How to add Rest API features to our Django :-

for our django application purpose, we have models.py, forms.py, views.py, applevel urls.py majorly. so don't disturb this application files for our APIs developing

Step-1: To add Rest API to existing django, we need to install django Rest-framework module

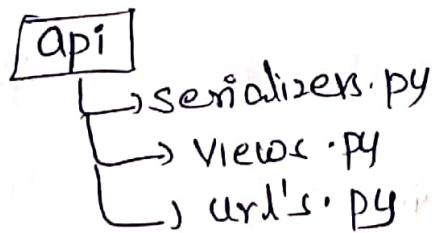
```
pip install djangorestframework
```

Step-2: Add rest-framework to default application inside INSTALLED App section in settings.py file of django project

```
INSTALLED_APPS = [  
    .....  
    'appname',  
    'rest-framework',  
]
```

Step-3: create one folder as 'api' inside our django application name. to maintaining  
Inside this 'api' folder, we need to store all our API related files

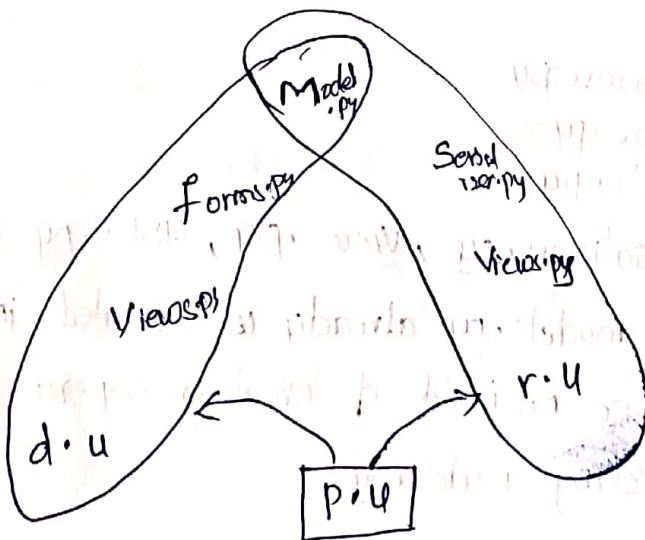
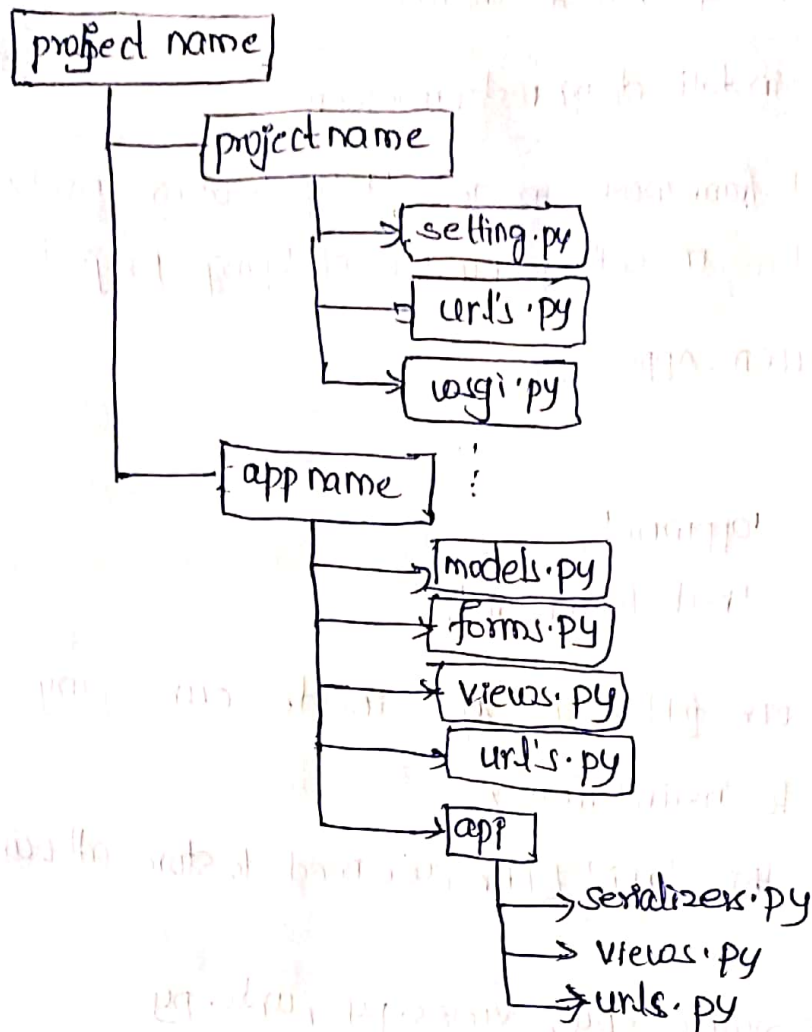
Eg:- serializers.py, views.py, urls.py



NOTE:- For our API's serializers.py, views.py, urls.py generally we are using, but models.py already we created in our django application, so no need of creating separately 'models' we will reuse existing models.py

Step-4:

# Django - to - RestAPI files structure



## NOTE:-

All rest Api files, via Api folder, we are calling



## Project:

create a project to performing the CRUD operations by using Django

step-1: project name: django-to-restapi-cbv-project

step-2: application name: employees

step-3: data base name: django-rest-cbv-db

step-4: open settings.py file and add our 'app' name, and  
and 'rest-framework' inside INSTALLED APPS section  
and configure the database details

step-5: open models.py

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

```
from django.urls import reverse
```

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

```
    eno = models.IntegerField()
```

```
    ename = models.CharField(max_length=30)
```

```
    esal = models.DecimalField(max_digits=10, decimal_places=2)
```

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

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

```
        return self.ename
```

```
    def get_absolute_url(self):
```

```
        return reverse('employee_list')
```

step-6: open views.py

```
from django.shortcuts import render
```

```
from django.views import generic
```

```
from models import employee
```

```

class EmployeeListView(generic.ListView):
    model = Employee
    # template_name = 'employee_list.html'
    # context_object_name = 'employee_list'

class EmployeeDetailView(generic.DetailView):
    model = Employee
    # template_name = 'employee_detail.html'
    # context_object_name = 'object or employee'

class EmployeeCreateView(generic.CreateView):
    model = Employee
    fields = '__all__'

    # template_name = 'employee_form.html'
    # context_object_name = 'form'

class EmployeeUpdateView(generic.UpdateView):
    model = Employee
    fields = '__all__'
    # template_name = 'employee_form.html'
    # context_object_name = 'form'

class EmployeeDeleteView(generic.DeleteView):
    model = Employee
    success_url = '/employee/'
    # template_name = 'employee_confirm_delete.html'
    # context_object_name = 'object'

```

```

from django.http import HttpResponse
import requests

def contactListView(request):
    response = requests.get('http://127.0.0.1:8888/api/emp/')

```

```
return HttpResponse ('<h1><font color= 'blue'>' + response.text +  
'</font>' + '<br><font color= 'red'> status  
code is:' + str(response.status_code) +  
'<br> Data Displayed </h1>')
```

step-7:

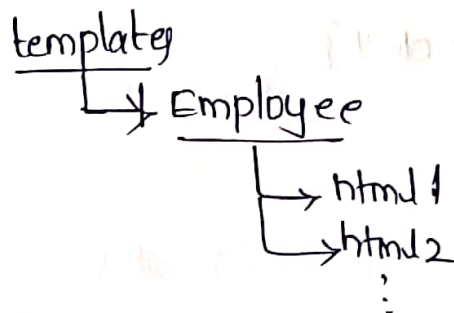
open admin.py

```
from django.contrib import admin
```

```
from models import Employee
```

```
admin.site.register (Employee)
```

step-8:- create a sub folder in the templates file, for our  
'html' files, the folder name may be ~~app~~ with  
application, bcz of easily understandable.



open base.html and write the code

<html>

<head>

[copy bootstrap cdn lines]

<title> Employee Records </title>

<style>

h1 { color : red;

background-color : yellow;

font-size : 30px;

padding : 20px;

}



```

    btn: hover {
        background-color: yellow;
        color: red;
    }
}
th, td {
    padding-bottom: 15px;
}
</style>
</head>
<body style = 'background-color: yellowgreen">
    <div class = 'container'>
        <h1> Welcome to Class Based View CRUD Operations </h1>
        <hr style = 'border-color: red; border-width: 10px'">
            { % block content % }
        { % endblock % }
    </div>
</body>
</html>

```

pen employee-confirm-delete.html and write the code

```

{ % extends 'employees/base.html' % }

```

```

{ % block content % }

```

```

<h4> Are you sure you want to delete this object </h4>

```

```

<form method = 'post'>

```

```

    { % csrf-token % }

```

```

    <input type = 'submit' value = 'delete' class = 'btn btn-danger'>

```

```

    <a href = "{ % url 'employee-list' % }" class = 'btn btn-primary'>

```

```

    </form>

```

```

        cancel </a>

```

```

{ % endblock % }

```

open employee\_detail.html

{ % extends 'employees/base.html' % }

{ % block content % }

<h2> Display Employee Detail </h2>

<h2> <ol>

<li> Employee Number is: {{ object.eno }} </li>

<li> Employee Name is: {{ object.ename }} </li>

<li> Employee salary is: {{ object.esal }} </li>

<li> Employee Address is: {{ object.eaddr }} </li>

</ol> </h2>

<a href="{% url 'employee\_list' %}" class='btn btn-primary'>

Goto Back </a>

{ % end block % }

open employee\_form.html

{ % extends 'employees/base.html' % }

{ % block content % }

<form method='post'>

{ % csrf-token % }

<table>

{{ form }}

</table>

<input type='submit' value='post' style='margin-left:

100px;'>

</form>

{ % end block % }



open employee\_list.html

§ f extends 'employees/base.html' v.3

§ f. block content v.3

<h2> Display all Employees </h2>

§ f. if employee\_list v.3

<table class='table table-striped table-bordered table-dark'>

<thead>

<tr>

<th> ENo </th>

<th> Name </th>

<th> salary </th>

<th> Address </th>

<th> Actions </th>

</tr>

</thead>

§ f. for emp in employee\_list v.3

<tbody>

<tr>

<td> {{ emp.eno }} </td>

<td> {{ emp.ename }} </td>

<td> {{ emp.esal }} </td>

<td> {{ emp.eaddr }} </td>

<td>

<a href='{{url 'employee\_detail' emp.id v.3}}' class='

btn btn-info">Detail </a>

<a href='{{url 'employee\_update' emp.id v.3}}' class='btn btn-success'>Edit </a>

<a href='{{url 'employee\_delete' emp.id v.3}}' class='btn btn-danger'>Delete </a>

</td>

</tr>

</tbody>

§ f. end for v.3

</table>

```

<h2 align="center">
< a href="{% url 'employee-create' %}"
class="btn btn-primary btn-lg"> Create New Employee </a>
{% else %}
<h1> No data available </h1>
{% end if %}
{% endblock %}

```

→ To integrate django with restapi, we have to create a folder inside the 'application' and write all the Rest-API related code inside the folder

- > ☐ Django-To-Restapi-CRV-Project
- > ☐ employees → application name
  - > ☐ api
    - ☐ serializers.py
    - ☐ urls.py
    - ☐ views.py

open serializers.py

```
from rest-framework import Serializers
```

```
from employees.models import Employee
```

```
class EmployeeSerializer(serializers.ModelSerializer):
```

```
    class Meta:
```

```
        model = Employee
```

```
        fields = '__all__'
```

open views.py

```
from employees.models import Employee
from employees.api.serializers import EmployeeSerializer
from rest_framework import Viewsets
class EmployeeModelViewSet (Viewsets.ModelViewSet):
    queryset = Employee.objects.all()
    serializer_class = EmployeeSerializer
```

open urls.py

```
from django.urls import path, include
from employees.api import views
from rest_framework.routers import DefaultRouter
router = DefaultRouter()
router.register('emp', views.EmployeeModelViewSet)
urlpatterns = [
    path('', include(router.urls))
]
```



Partner.py

import requests

url = 'http://127.0.0.1:8833/api/emp/'

# testing restapi url

response = requests.get('http://127.0.0.1:8833/api/emp/')

# creating new record

payload = {

    'leno': 30,

    'ename': 'Srinivas',

    'esal': 35000,

    'leaddr': 'KPHB'

}

# updating single record

payload = {

    'leno': 30,

    'ename': 'srini',

    'esal': 45000,

    'leaddr': 'maitrivaram'

}

# handling the response to display required messages

if response.status\_code == 200:

    try:

        print(response.json())

        print('Display all records successfully!')

    except:

        print("status code is :", response.status\_code)

        print("you are not getting json data from provider")

elif response.status\_code == 201:

    print("status code is" , response.status\_code)

```
print(response.json())  
print('Record created successfully.')
```

```
elif response.status_code == 204:  
    print('status code is:', response.status_code)  
    print('Record deleted successfully')
```

```
elif response.status_code == 400:  
    print('status code is:', response.status_code)  
    print('Record not created successfully, please send valid type data.')
```

```
elif response.status_code == 404:  
    print('status code is:', response.status_code)  
    print('Record not found')
```

```
elif response.status_code == 500500:  
    print('status code is:', response.status_code)  
    print('server side problems occurs')
```

```
elif response.status_code == 403:  
    print('method not hitting, csrf token missing')
```

```
else:  
    print('status code is:', response.status_code)  
    print('some thing wrong')
```

```

from django.http import HttpResponseRedirect
import json
import requests

# getting all records from partner app
def ContactListView(request):
    response = requests.get('http://127.0.0.1:9944/api/emp/')
    if response.status_code == 200:
        context = { 'contact_list': json.loads(response.text) }
        return render(request, 'contacts/contact-list.html', context)

# getting id based record from partner app
def ContactDetailView(request, pk):
    response = requests.get('http://127.0.0.1:9944/api/emp/' + str(pk))
    if response.status_code == 200:
        context = { 'contact': json.loads(response.text) }
        return render(request, 'contacts/contact-detail.html', context)
    else:
        context = { 'noncontact': json.loads(response.text) }
        return render(request, 'contacts/contact-detail.html', context)

# deleting id based record from partner app
def ContactDeleteView(request, pk):
    response = requests.delete('http://127.0.0.1:9944/api/emp/' + str(pk))
    if response.status_code == 204:
        context = { 'contact': 'Record deleted successfully' }
        return render(request, 'contacts/contact-confirm-delete.html', context)
    else:
        context = { 'noncontact': 'Record not available to delete' }
        return render(request, 'contacts/contact-confirm-delete.html', context)

```



from .models import Employee

from .forms import ContactForm

def ContactCreateView(request):

if request.method == 'POST':

form = ContactForm(request.POST)

if form.is\_valid():

eid = request.POST.get('eid', '')

ename = request.POST.get('ename', '')

eemail = request.POST.get('eemail', '')

econtact = request.POST.get('econtact', '')

payload = {

'eid': eid,

'ename': ename,

'eemail': eemail,

'econtact': econtact

}

response = requests.post('http://127.0.0.1:9944/api/empl/',  
data=payload)

if response.status\_code == 201:

context = {'created': json.loads(response.text)}

return render(request, 'contacts/create.html', context)

else:

context = {'not created': json.loads(response.text)}

return render(request, 'contacts/create.html', context)

else:

return HttpResponse('data not save')

else:

form = ContactForm()

return render(request, 'contacts/contact\_form.html',

{ 'form': form })

open base.html

<html>

<head>

[copy cdn lines]

<style>

h1 { color: red;

background-color: yellow;

font-size: 30px;

padding: 20px;

}

• btn: hover {

background-color: yellow;

color: red;

}

h1, td {

padding-bottom: 15px;

}

</style>

</head>

<body style = "background-color: yellowgreen">

<div class = 'container'>

<h1> Welcome to class Based VECAS CURD operation </h1>

<hr style = 'border-color: red ; border-width: 10px'>

Σ % block content % }

Σ % end block % }

</div>

</body>

</html>

Open contact-confirm-delete.html

Σ % extends 'contacts/base.html' % }

Σ % block content % }

Σ % if contact % }

<h2> ΣΣ contact ΣΣ </h2>

Σ % else % }

<h2> ΣΣ nocontact ΣΣ </h2>

Σ % endif % }

Σ % endblock % }

Open contact-detail.html

Σ % extends 'contacts/base.html' % }

Σ % block content % }

Σ % if contact % }

<ol>

<li> primary key : ΣΣ contact.id ΣΣ </li>

<li> Employee Eid : ΣΣ contact.eid ΣΣ </li>

<li> Employee Name : ΣΣ contact.ename ΣΣ </li>

<li> Employee contact : ΣΣ contact.econtact ΣΣ </li>

</ol>

Σ % else % }

<h2> Record not available </h2>

Σ % endif % }

Σ % endblock % }



open contact-form.html

{ % extends 'contact/base.html' % }

{ % block content % }

<form method='post'>

{ % csrf\_token % }

<table>

{ % form % }

</table>

<input type='submit' value='post' class='btn btn-success'>

<input type='reset' value='clear' class='btn btn-danger'>

</form>

{ % end block % }

open contact-list.html

{ % extends 'contacts/base.html' % }

{ % block content % }

{ % if contact\_list % }

<table class='table table-dark'>

<thead>

<tr>

<th> Id </th>

<th> Eid </th>

<th> Ename </th>

<th> Email </th>

<th> Econtact </th>

<th> Actions </th>

</tr>

</thead>

{ % for row in contact\_list % }

<tbody>

<tr>

<td> {{ row.id }} </td>

<td> {{ row.eid }} </td>

```

<td> {{ row.name }} </td>
<td> {{ row.email }} </td>
<td> {{ row.contact }} </td>
<td>
<a href="/contacts/detail/{{ row.id }}" class="btn btn-primary">
    detail </a>
<a href="{{ url 'contacts-update' row.id }}" class="btn
    btn-success">update </a>
<a href="/contacts/delete/{{ row.id }}" class="btn btn-danger">
    delete </a>
</td>
</tr>
</tbody>
{% endfor %}
</table>
<h2 align="center">
<a href="{{ url 'contacts-create' }}" class="btn btn-success btn-lg">
    create New Record </a>
{% else %}
<h2> Data not available </h2>
{% endif %}
{% endblock %}

```

open create.html

```

{% extends 'contacts/base.html' %}
{% block content %}
    {% if created %}
        <h2> {{ created }} </h2>
    {% else %}
        <h2> {{ not_created }} </h2>
    {% endif %}
{% endblock %}

```

open project level urls.py

from django.contrib import admin

from django.urls import path, include

from employees import views

urlpatterns = [

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

# our django app cbv urls

path('employee/', views.EmployeeListView.as\_view(), name='employee-list'),

path('employee/<int:pk>/', views.EmployeeDetailView.as\_view(), name='employee-detail'),

path('employee/create/', views.EmployeeCreateView.as\_view(), name='employee-create'),

path('employee/<int:pk>/update/', views.EmployeeUpdateView.as\_view(), name='employee-update'),

path('employee/<int:pk>/delete/', views.EmployeeDeleteView(), name='employee-delete'),

# our django application rest-api urls

path('api/', include('employees-api.urls')),

# provider app communication urls

path('contacts/', views.ContactListView, name='contacts'),

path('contacts/detail/<int:pk>/', views.ContactDetailView, name='contact-detail'),

path('contacts/delete/<int:pk>/', views.ContactDeleteView, name='contacts-delete'),



path 'c:\contacts\create /', views: Contact CreateView, name='contacts-create')

2