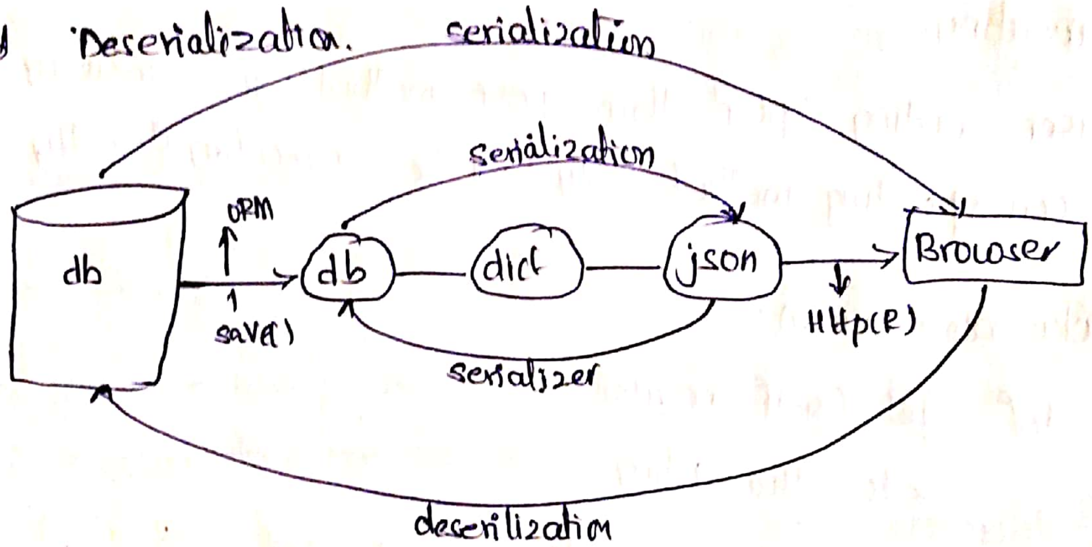
Serilization :-

converting the object from one form into ~~anto~~ another form is called serialization.

the process of converting the querysets (or) database instances into python dict type and then converting this dict data into JSON data type is called as serialization

Deserialization :-

The process of converting JSON data type into python dict type and then converted into database instances (or) querysets format is called Deserialization.



Q How many ways we will perform serilization:-

We are performing 3 ways

1. By using python inbuilt module JSON methods(,

```
json.dumps(data
```

2. By using Django's serilize() method

```
from.django.core.serializers import serialize
```

3. By using rest-frame work serialization module

```
from rest-framework import serializers
```

Installation of django Rest framework

If you want to use Rest Api in our existing django project, we need to install one 3rd party module.

```
cmd > pip install djangorestframework ==3.8
```

Now add rest_framework default application Inside installed app section in settings.py file

```
INSTALLED_APPS = [
            ......,
            ......,
            'appname',
            'rest_framework',  → # add this one in every project
        ]
```

Now all restapi, modules we can import from this rest_framework application

```
from rest_framework import required_modules
```

Django follows MVT structure, But rest Api follows 'MVS' structure

M → model
V → view
T → Template
$\downarrow$
(presentation)

$$\begin{tikzcd} & \text{(presentation)} \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \\ \text{html} & \text{css} & \text{JS} \end{tikzcd}$$

M → model
V → view
S → serilization
$\downarrow$
carrying data
$\downarrow$
(json)

* A traditional Django application needs a dedicated url; view, templates to translate information from the database to browser (or) between to db

* But in django rest framework we need $\boxed{\text{url}}$ $\boxed{\text{view}}$ and $\boxed{\text{serializer}}$ concept but not template concept

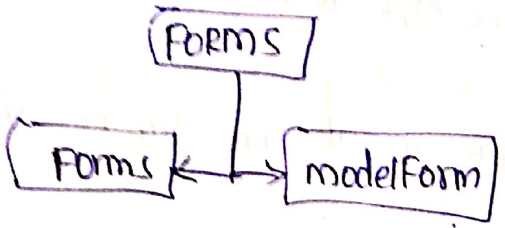
→ url controls access to API endpoints

→ views controls the logic of the data being sent

→ the ~~serializer~~ serializer performs the magic of converting our information into a format suitable for transmission over the internet like JSON

→ A normal webpage require HTML, css and JS. But our API's only sending data in the JSON format. so, no HTML and No css just we need only data.

* REST API serializer is same like django form and model



→ By using serializer class we need to write all model fields with required datatypes it increasing more lines of code

→ By using 'model serializer' we writing less code and it providing more internal code it deals with model instance

* serializers code we will create a seperate python file which name as $\boxed{\text{serializers.py}}$

* create this above file inside our application name same like django $\boxed{\text{forms.py}}$

<u>Model class</u>:-

```
        class Emp (models.Model)
            eno= models.IntegerField()
            ename = models.charField(max_length=100)
            esal  = models.IntegerField()
```

* TO creating the user defined serializer then write below code

```
from rest-framework import serializer
class Empserializer (serializers.Serializer)
      eno = serializers.IntegerField()
      ename = Serializers.CharField(max_length=100)
      esal = serializers.IntegerField()
```

* TO creating the user serializer class by model serializer class

step-1 :-

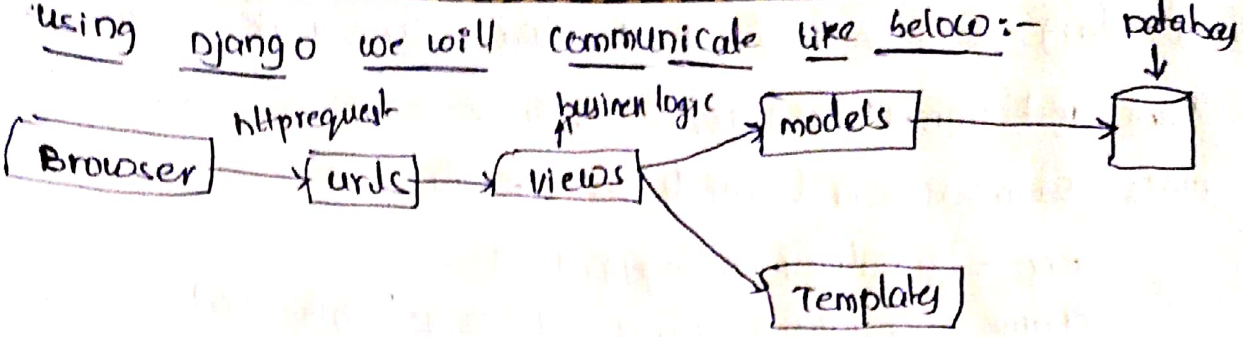
1. importing the serializers module
2. importing the userdefined model class
3. creating the user defined serializer class by extending the predefined model serializer class
4. creating the meta class as Nested class of our's user defined serializer class

Representing our user defined model class name by using model field respresenting all our model fields by using fields attribute

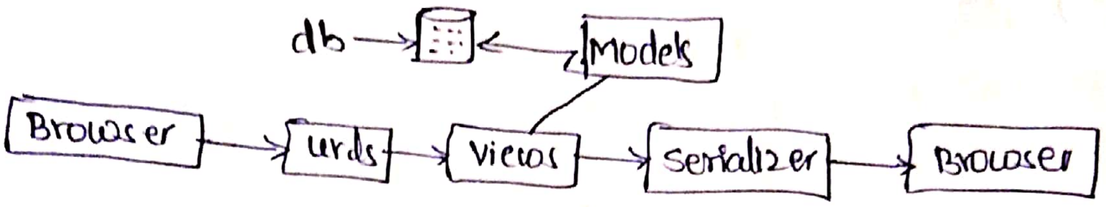
Model serializer :-

```
1. from rest-framework import serializer
2. from .models import emp
3. class empserializer (serializers.ModelSerializer)
4.     class Meta :
             model = Emp
             fields = "__all__"
```

using Django we will communicate like below:-



using RestApi we will communicate like below:-



To developing the Rest Api programs for performing the database operations we are going to use (http methods) like

GET PUT POST DELETE

To performing the CURD operations on our database using http methods we will perform 5 operations these all operations dividing like a 2 sections

1. Non id based operations (or) Non primary key
2. id based operations (or) primary key

→ Getting all and records and Creating new records is comes under non id based operations

→ Getting single record updating single record and deleting single record are comes under ID (or) primary key based records

→ To performing the non-ID based operations we will create one Non ID based class which containing required operations for

ex:-

```
class NonIdbased(....):
    def get(self,request):
        ......   Get logic
    def post(self,request):
        ......   post logics
```

→ To performing the id based operations create one ID based class which containing required operations

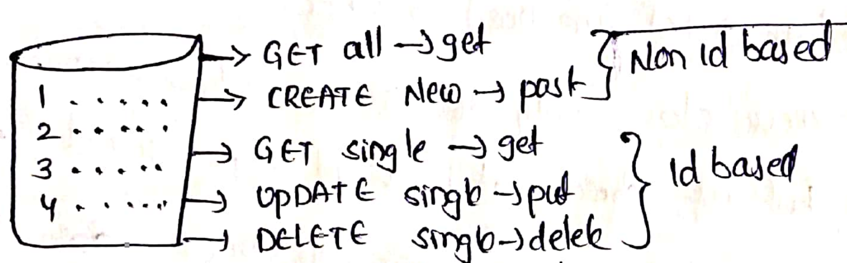
```
class IDbasedclass (....):
    def get (self, request, ID):
        ----- GET logics
    def put (....):
        ----- update logics
    def delete(....):
        ----- delete logics
```

→ To executing Non ID based class create one Non ID based url

http://127.0.0.1:8000/api/emp/1

→ To executing ID based operations create one ID based url

http://127.0.0.1:8000/api/emp/2



```
class CN(....):
    def get(...):
        -----
    def post(...):
        -----
```

```
class CN2(....):
    def get(....):
        -----
    def put (....):
        -----
    def delete (....):
        -----
```