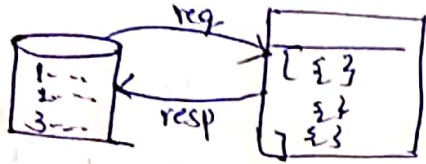


## PAGINATION:-

Some times our database may containing few records, less no: of records, then by send non-id based url for getting all records from database, we are not getting any problem to display all records at a time on the screen



Sometimes our database may have hundreds of resources then when we are sending request for displaying all records at a time on the screen is getting someone difficult. we are getting a vertical scroll bar to see all records, we need to move top to bottom.

If huge no: of records are available in database, then highly recommended those resources divided into across the multiple pages which is nothing but pagination concept.

### Definition:-

The process of splitting the large no: of database records into multiple pages wise is called pagination. If no: of records are very high then we will go for pagination.

django rest framework provides several pagination class to implement the pagination concept in our existing project.

### for example:-

1. page number pagination (PNP)
2. limit+offset pagination (LOP)
3. Cursor pagination (CP)

these all classes available in 'PAGINATION' module

We can import like below

```
from rest_framework.pagination import PageNumberPagination
```

pagination we can provide like two variations

1. Global pagination

2. Local pagination

Global pagination:-

Global pagination means inside settings.py file, we create one dictionary object which name as REST\_FRAMEWORK

for ex:-

```
REST_FRAMEWORK = {
```

```
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.  
    PageNumberPagination',
```

```
    'PAGE_SIZE': 10
```

```
}
```

→ here ~~page~~ page-size representing for every page how many records we want to display

Query String parameter:-

Query parameters are a defined set of parameters attached to the end of url

They are the extension of the url that are used to help the define specific content (or) actions based on the data being Passed. By

By using the querystring parameters end user can give input dynamically with required information and they will get response dynamically based on the input

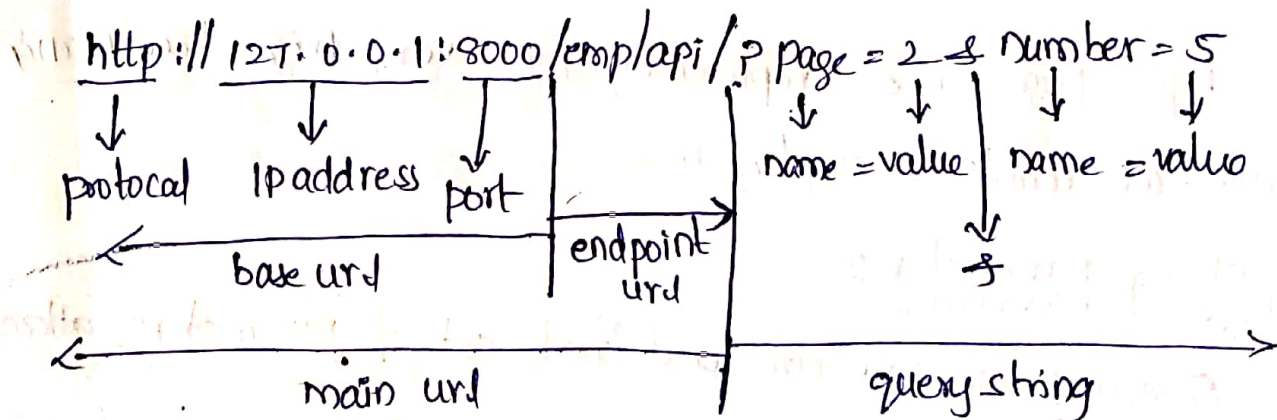


- Query string is separated by "?" symbol with main url
- After Question mark(?) if any url is there, then we are called that url as query string
- Querystring containing 'queryname' & 'query value' separated by "=" symbol, both name & value we are called query pair, then multiple query pairs are separated by "&" symbol

-----url-----? page=2 & number=5

URL ? name1=value1 & name2=value2

- Before '?' if any url is there, that is called as main/actual url
- Again this main 'url' containing two parts 'base url' and 'end point' url
- Again "baseurl" containing multiple parts like 'http' 'ip address', 'port' number



### local pagination :-

- It means, inside `views.py` we will provide the pagination settings
- Here we are importing required pagination class and then inside user view by using 'pagination\_class' field we are representing required class

inside views.py we add like below

from rest\_pagination

from rest\_framework.pagination import PageNumberPagination

```
class EmployeeAPIView (generics.ListAPIView):
```

```
    queryset = -----
```

```
    serializer_class = -----
```

```
    pagination_class = PageNumberPagination
```

\* Generally all serialization code we created in a separate file like serializers.py. same like all pagination logics will work in a separate class file like "pagination.py" and here create user defined pagination with required overriding logics

\* After completing this <sup>user</sup> pagination class then import this class inside

views.py

\* This pagination.py create inside application name

```
from rest_framework.pagination import PageNumberPagination
```

```
class UserPagination (PageNumberPagination):
```

```
    page_size = 10
```

```
    -----  
    -----  
    -----
```

↓  
pagination.py

views.py

```
from .pagination import UserPagination
```

```
class classname (ModelViewSet):
```

```
    queryset = -----
```

```
    serializer_class = -----
```

```
    pagination_class = UserPagination
```



→ To creating user defined values and using instead predefined values in a querystring parameters then we will create user defined user defined pagination class

pagenumber pagination :-

By using pnp we can represent which page we want to open and each page how many records wants to display

→ In this pagenumber pagination, page-size is a mandatory value which represents how many records we want to display by default.

→ Here 'page' is the default value which represents, which page we want to display by default it display's first page

\* Here 'last' is the default value of page attribute, which represents directly 'last page'.

url...?page = last

\* By using "page-query-param" we can represent our own page attribute for displaying the which page we want

page-query-param = 'my<sup>my page</sup>page' → override the 'page' attribute

eg: ...url...?my<sup>my page</sup>page = 2 → which opens second page

\* By using "page-size-query-param", we can represent our attribute name by using this, we will override this default page-size behavior by using this value we can display how many records we want <sup>override size</sup>

eg: ...url...?my<sup>my page</sup>page = 3 & number = 10 <sup>page-size-query-param = 'number'</sup>

Here, we will open directly 3<sup>rd</sup> page with 'Ten' records

...url...?number=7  $\rightarrow$  by default it first page with 7 records

\* By using 'last-page-strings' we can override default value of last

last-page-strings = ('end page', )

Eg:- ...url...?mypage='endpage' & number=5  $\rightarrow$  it opens last page with 5 records

...url...?mypage='endpage'  $\rightarrow$  it open last page with ~~5~~ records default page-size values displaying

\* Max-page-size  $\rightarrow$  it represents for each page how many records we want to display

Eg:- max-page-size=10

...url...?mypage=2 & number=7  $\rightarrow$  it opens 2nd page with 7 records

...url...?mypage=2 & number=15  $\rightarrow$  it open 2nd page with 10 records but not 15 bcz of max-page-size=10

Σ "count": 18,

"next": "url?mypage=5&number=2",

"previous": "url?mypage=3&number=2",

"results": [ Σ "empid": 70,

"empname": 'Bhumra',

"email": "bhumra@gmail.com",

"salary": "7000.00"

},

⋮

'previous' represents our current page related back page

'results'  $\rightarrow$  it represents the list of data base records based on size

?mypage=4 & number=2, here 'count' represents total number of records in data base

'next' represents our current page related forward page



## Limit Offset Pagination :-

- \* By using this class we can display how many records we want and from which index we want
- \* Here 'limit' represents "total no. of records" that we want to display
- \* offset represents from which index onwards, the records want to displayed (it gives some range of records)

\* Eg:- url...?limit=20 & offset=10

→ Here 10th index onwards 20 records we will display

→ To creating the user defined pagination by extending LimitOffset pagination, then create like below

from rest\_framework.pagination import LimitOffsetPagination

class my\_pagination(LimitOffsetPagination):

default\_limit = 15

→ Here default limit is representing how many records we can display by default, it is mandatory field, offset by default starts from '0'

→ By using 'limit-query-param' we can override default 'limit' parameter

eg:- limit-query-param = 'mylimit'

→ By using 'offset-query-param' we can override default 'offset' parameter

eg:- offset-query-param = 'myoffset'

→ 'max\_limit' parameter is used for representing maximum no. of records to display for a request

Eg:-  $\{127.0.0.1:8000/api/?mylimit=13 \& myoffset=20\}$  -> by using override methods  
Here index 20 onwards 13 records are displayed if available

Eg:-  $\{127.0.0.1:8000/api/?mylimit=25 \& offset=10\}$

Here index 10 from 10th index onwards, maximum we are getting 25 records only but not '25', if max limit = 20 records

Eg:-  $\{127.0.0.1:8000/api/?mylimit=15 \& myoffset=10\}$

Here from 10th index onwards, we will get total 15 records but not 20 records

### Cursor pagination :-

By using the cursor pagination we will get all database records and we will get sorting order also. By default sorting order is ascending order by using created column

→ Here 'ordering' field is used for sorting the all records based on the required column

→ By default we are getting order is 'Ascending order'

Eg:- ordering = 'salary'

→ Here, salary based we are getting all records order wise, if we want to get in 'descending order' then take prefix as '-' (minus) before the column

Eg:- ordering = '-salary' → for descending order

→ Here salary is a column which should be available in our database table (or) model fields

→ By using page\_size field we can represent how many records we want to display



- if page-size is not provided, then total no. of records displayed
- if ordering = required column name, we are not providing then by default it takes column name as '-created'

eg:- ordering = '-created'

- if ~~order~~ created column not available in our model, then we will get one exception called 'fieldError', it shows keyword 'create' noavailable and available column choices are (column name)

Example:- create a django rest api project to performing the db operation by using pagination concept

- step-1: project name: pagination-project
- step-2: app name: pagination-app
- step-3: db name: pagination-db
- step-4: open settings.py and configure 'db' in and add our app name, rest-framework in 'INSTALLED\_APP' section

step-5: open models.py and write the below code

```
from django.db import models

class Emp(models.Model):
    emp_id = models.IntegerField(primary_key=True)
    ename = models.CharField(max_length=100)
    email = models.EmailField(max_length=100)
    esal = models.DecimalField(max_digits=10, decimal_places=2)
```

step-6:- open serializers.py

```
from models import emp
from django rest-framework import serialization
class EmpSerialization(serializers.ModelSerializer):
```

class Meta:

model = Emp

fields = '\_\_all\_\_' ('empid', 'ename', 'email', 'esal']

step-6: create pagination.py under application name and create the user pagination

from rest-framework: ~~import~~ pagination import (PageNumberPagination, LimitOffsetPagination, CursorPagination)

class mypagination (PageNumberPagination):

page\_size = 3

page\_query\_param = 'my page'

page\_size\_query\_param = 'number'

max\_page\_size = 5

last\_page\_strings = ('end pages',)

class mypagination (LimitOffsetPagination):

default\_limit = 2

limit\_query\_param = 'mylimit'

offset\_query\_param = 'myoffset'

class Mypagination (CursorPagination):

ordering = '-salary'

# ordering = '-empname'

page\_size = 4

step-7: open views.py and write below code

from django.shortcuts import render

from pagination.App.models import emp

from .serialization import empserializer

from rest-framework import viewsets



from .pagination import MyPagination

class EmpViewSet (viewsets.ModelViewSet):

queryset = Emp.objects.all()

serializer\_class = EmpSerializer

pagination\_class = MyPagination

Step-8:- open urls.py and write the below code

from django.conf.urls import url, include

from pagination\_App import views

from rest\_framework.routers import DefaultRouter

router = DefaultRouter()

router.register('emp-viewset', views.EmpViewSet)

urlpatterns = [

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

]

Step-9:- Run makemigrations, Migrate and runserver

Step-10: click on Ip address