

Topic: 14. DRF Pagination

Speaker: Personal / Notebook: API Development using Django Framework



Django Pagination:

[Documentation on Django Rest Framework Pagination](#)

REST framework includes support for customizable pagination styles. This allows you to modify how large result sets are split into individual pages of data.

The pagination API can support either:

- Pagination links that are provided as part of the content of the response.
- Pagination links that are included in response headers, such as `Content-Range` or `Link`.

The built-in styles currently all use links included as part of the content of the response. This style is more accessible when using the browsable API.

Pagination is only performed automatically if you're using the generic views or viewsets. If you're using a regular `APIView`, you'll need to call into the pagination API yourself to ensure you return a paginated response. See the source code for the `mixins.ListModelMixin` and `generics.GenericAPIView` classes for an example.

Pagination can be turned off by setting the pagination class to `None`.

Two most commonly used pagination:

`PageNumberPagination` - takes a `page_size` as a parameter and returns a response accordingly.

example: `/blogs/?page=10`

`LimitOffsetPagination` -

The `limit` parameter sets the number of items you want to see on a single page.

The `offset` parameter tells the API where to start fetching the items from.

example for a blog post of 100:

`/blogs/?limit=10&offset=0` * this means get the first 10 blog posts(items 1-10)

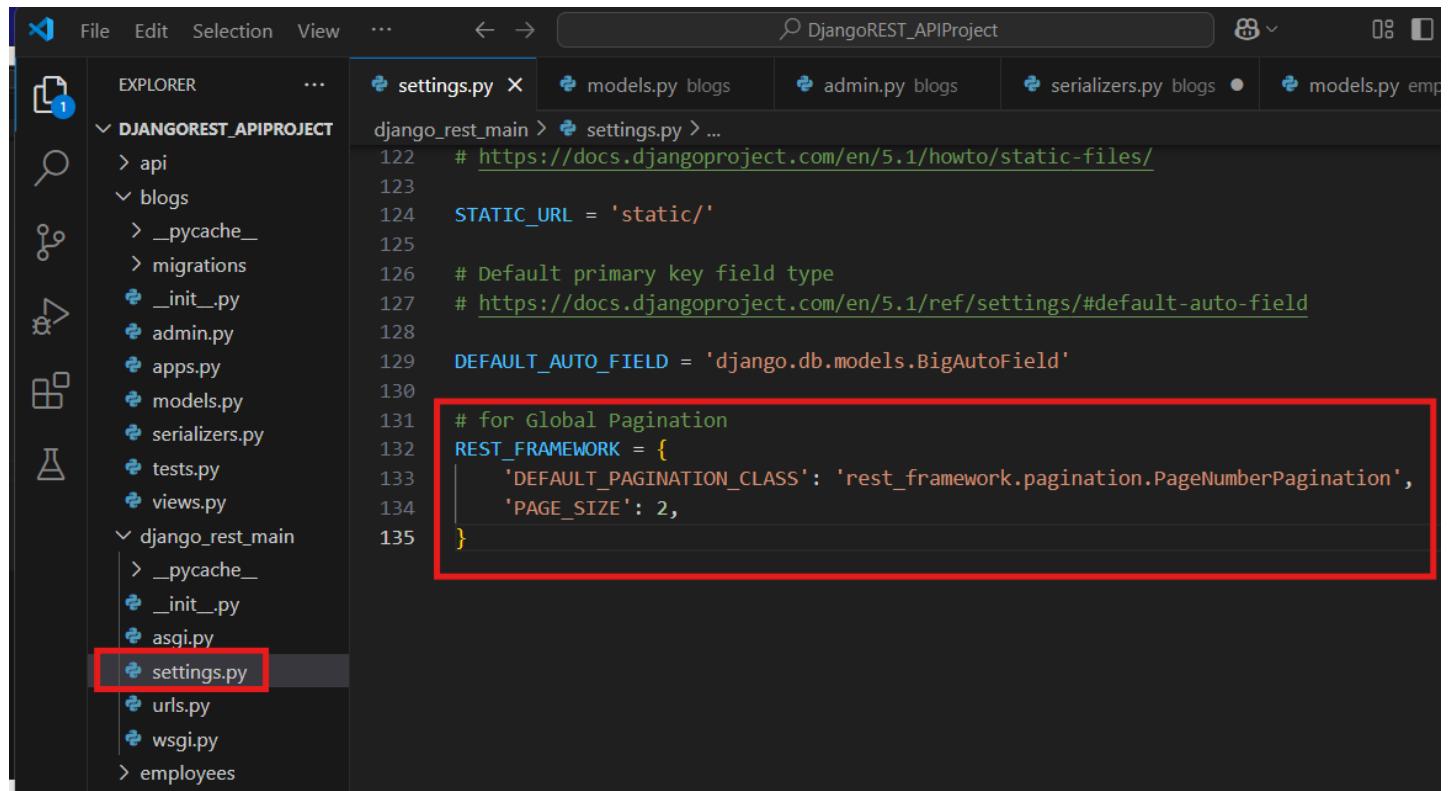
`/blogs/?limit=10&offset=10` * this means get the next 10 blog posts(items 11-20)

`/blogs/?limit=10&offset=90` * this means get the last 10 blog posts(items 91-100)

You can apply pagination using **Global Pagination** and using **Custom Pagination**.

Global Pagination:

1. Go to `SETTINGS.PY`, and add the `default_pagination_class`: This works only with `generics` and `viewsets`.



The screenshot shows a code editor with the file `settings.py` open. The `REST_FRAMEWORK` configuration is highlighted with a red box. The code is as follows:

```
122 # https://docs.djangoproject.com/en/5.1/howto/static-files/
123
124 STATIC_URL = 'static/'
125
126 # Default primary key field type
127 # https://docs.djangoproject.com/en/5.1/ref/settings/#default-auto-field
128
129 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
130
131 # for Global Pagination
132 REST_FRAMEWORK = {
133     'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',
134     'PAGE_SIZE': 2,
135 }
```

2. So, now when you reload your blog page:

BEFORE GENERAL PAGINATION, it lists all the blog posts on one page.

→ ⌂ ⓘ 127.0.0.1:8000/api/v1/blogs/ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂

Django REST framework

api_djangoadmin

Api Root / Blogs

Blogs

OPTIONS GET ▾

GET /api/v1/blogs/

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
  {
    "id": 2,
    "comments": [
      {
        "id": 4,
        "comment": "Interesting!",
        "blog": 2
      },
      {
        "id": 5,
        "comment": "I wonder when this can be a law in other states too.",
        "blog": 2
      }
    ],
    "blog_title": "Creating and Sharing Deceptive AI-Generated Media Is Now a Crime in New Jersey",
    "blog_body": "Creating and sharing deceptive media made with artificial intelligence is now a crime in New Jersey and open to lawsuit"
  },
  {
    "id": 3,
    "comments": [
      {
        "id": 6,
        "comment": "Interesting top dog bidders here!",
        "blog": 3
      },
      {
        "id": 8,
        "comment": "I want to see Microsoft there!",
        "blog": 3
      }
    ],
    "blog_title": "Bidders of Tik Tok",
    "blog_body": "Amazon, Blackstone, OnlyFans founder, Project Liberty, MrBeast, Perplexity AI are some of the bidders"
  }
]
```

AFTER GENERAL PAGINATION, it lists 2 blog posts per page and shows page numbers. There were a total of 5 blog posts saved in a database model, BLOGS.

Page 1:

→ ⌂ ⓘ 127.0.0.1:8000/api/v1/blogs/

Django REST framework api_djangoadmin

Api Root / Blogs

Blogs

OPTIONS GET

1 2 3 »

GET /api/v1/blogs/

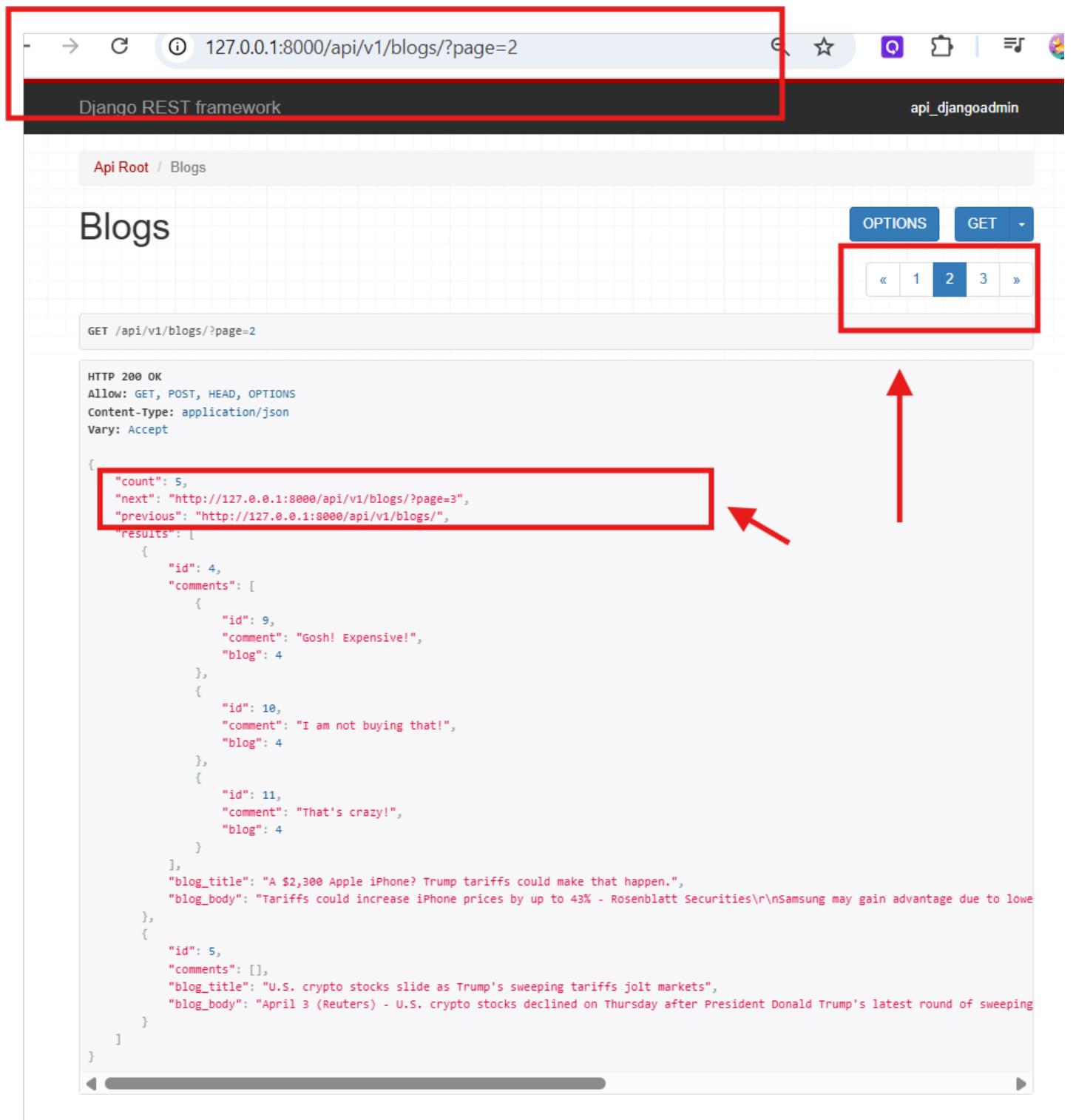
HTTP 200 OK

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{ "count": 5, "next": "http://127.0.0.1:8000/api/v1/blogs/?page=2", "previous": null, "results": [ { "id": 2, "comments": [ { "id": 4, "comment": "Interesting!", "blog": 2 }, { "id": 5, "comment": "I wonder when this can be a law in other states too.", "blog": 2 } ], "blog_title": "Creating and Sharing Deceptive AI-Generated Media Is Now a Crime in New Jersey", "blog_body": "Creating and sharing deceptive media made with artificial intelligence is now a crime in New Jersey and open to law" }, { "id": 3, "comments": [ { "id": 6, "comment": "Interesting top dog bidders here!", "blog": 3 }, { "id": 8, "comment": "I want to see Microsoft there!", "blog": 3 } ], "blog_title": "Bidders of Tik Tok", "blog_body": "Amazon, Blackstone, OnlyFans founder, Project Liberty, MrBeast , Perplexity AI are some of the bidders" } ] }
```



127.0.0.1:8000/api/v1/blogs/?page=2

Django REST framework api_djangoadmin

Api Root / Blogs

Blogs

OPTIONS GET

« 1 2 3 »

GET /api/v1/blogs/?page=2

HTTP 200 OK

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{ "count": 5, "next": "http://127.0.0.1:8000/api/v1/blogs/?page=3", "previous": "http://127.0.0.1:8000/api/v1/blogs/", "results": [ { "id": 4, "comments": [ { "id": 9, "comment": "Gosh! Expensive!", "blog": 4 }, { "id": 10, "comment": "I am not buying that!", "blog": 4 }, { "id": 11, "comment": "That's crazy!", "blog": 4 } ], "blog_title": "A $2,300 Apple iPhone? Trump tariffs could make that happen.", "blog_body": "Tariffs could increase iPhone prices by up to 43% - Rosenblatt Securities\nSamsung may gain advantage due to lowe }, { "id": 5, "comments": [], "blog_title": "U.S. crypto stocks slide as Trump's sweeping tariffs jolt markets", "blog_body": "April 3 (Reuters) - U.S. crypto stocks declined on Thursday after President Donald Trump's latest round of sweeping } ] }
```

3. Also, when we run our path for EMPLOYEES model, we see this pagination as well.

127.0.0.1:8000/api/v1/employees/

Django REST framework

api_djangoadmin

Api Root / Employee Viewset List

Employee Viewset List

OPTIONS GET

GET /api/v1/employees/

HTTP 200 OK

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{  
    "count": 3,  
    "next": "http://127.0.0.1:8000/api/v1/employees/?page=2",  
    "previous": null,  
    "results": [  
        {  
            "id": 1,  
            "emp_id": "EMP001",  
            "emp_name": "Rosilie",  
            "designation": "Software Developer"  
        },  
        {  
            "id": 6,  
            "emp_id": "EMP004",  
            "emp_name": "Arnel Zethus",  
            "designation": "AI Engineer"  
        }  
    ]  
}
```

Raw data

HTML form

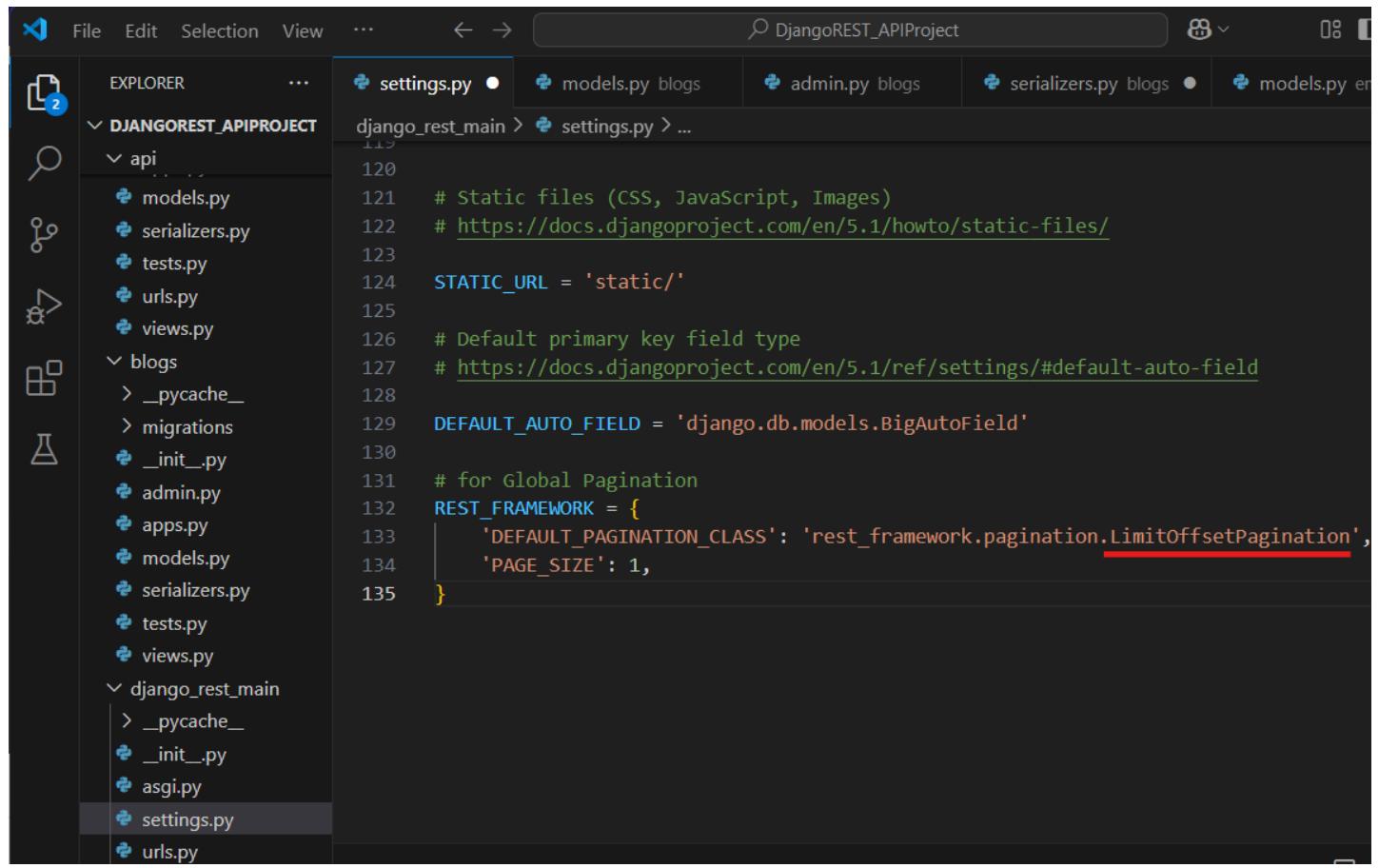
Emp id

Emp name

Designation

POST

4. Now if we use LimitOffsetPagination, we update our SETTINGS.PY:



The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, ...
- Search Bar:** DjangoREST_APIProject
- Toolbar:** Includes icons for file operations, search, and other project management.
- Left Sidebar (EXPLORER):** Shows the project structure:
 - DJANGOREST_APIPROJECT
 - api
 - models.py
 - serializers.py
 - tests.py
 - urls.py
 - views.py
 - blogs
 - _pycache_
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - serializers.py
 - tests.py
 - views.py
 - django_rest_main
 - _pycache_
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py- Right Panel (Code View):** Displays the content of `settings.py` with line numbers:

```
120
121 # Static files (CSS, JavaScript, Images)
122 # https://docs.djangoproject.com/en/5.1/howto/static-files/
123 STATIC_URL = 'static/'
124
125 # Default primary key field type
126 # https://docs.djangoproject.com/en/5.1/ref/settings/#default-auto-field
127
128 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
129
130 # for Global Pagination
131 REST_FRAMEWORK = {
132     'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.LimitOffsetPagination',
133     'PAGE_SIZE': 1,
134 }
```

Our blog pages shall look like this where only one blog post per page and it sets the offset to 1. If you click on NEXT, it shall display OFFSET=2 on your URL path.

127.0.0.1:8000/api/v1/blogs/?offset=1

Django REST framework

api_djangoadmin

Api Root / Blogs

Blogs

OPTIONS GET

« 1 2 3 4 5 »

GET /api/v1/blogs/?offset=1

HTTP 200 OK

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{ "count": 5, "next": "http://127.0.0.1:8000/api/v1/blogs/?limit=1&offset=2", "previous": "http://127.0.0.1:8000/api/v1/blogs/?limit=1", "results": [ { "id": 3, "comments": [ { "id": 6, "comment": "Interesting top dog bidders here!", "blog": 3 }, { "id": 8, "comment": "I want to see Microsoft there!", "blog": 3 } ], "blog_title": "Bidders of Tik Tok", "blog_body": "Amazon, Blackstone, OnlyFans founder, Project Liberty, MrBeast, Perplexity AI are some of the bidders" } ] }
```

Raw data HTML form

Blog title

Blog body

POST

If the offset is 2:

The screenshot shows a browser window with the URL `127.0.0.1:8000/api/v1/blogs/?limit=1&offset=2`. The page title is "Django REST framework" and the sidebar shows "api_djangoadmin". The main content is titled "Blogs". On the right, there are "OPTIONS" and "GET" buttons. Below them is a pagination bar with buttons for "«", "1", "2", "3" (which is highlighted in blue), "4", "5", and "»". A red box highlights the "3" button and the "next" URL in the JSON response. The JSON response shows a "count" of 5, a "next" URL, a "previous" URL, and a "results" array containing one blog entry with an ID of 4, comments from users 9, 10, and 11, and a blog body about iPhone tariffs.

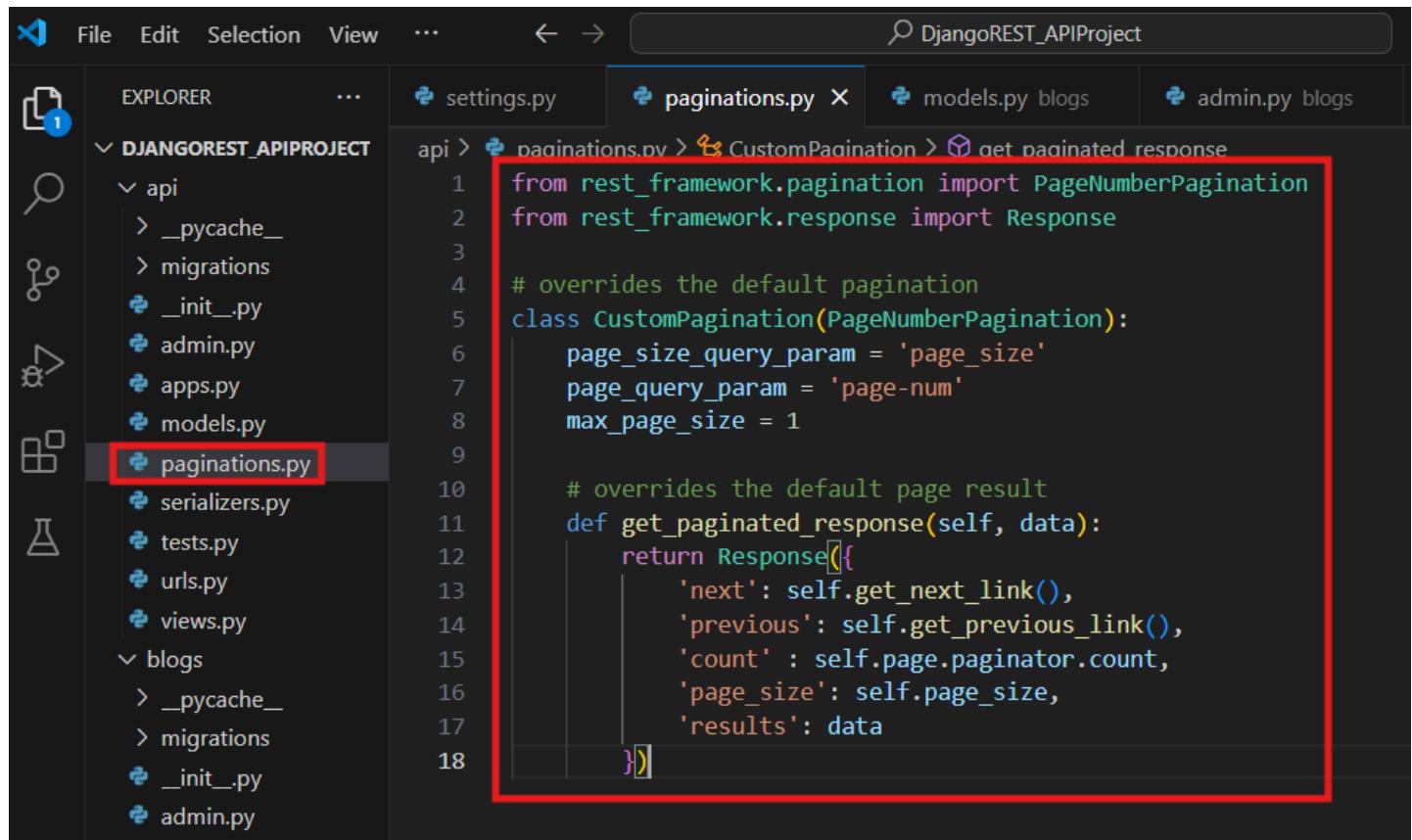
```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 5,
    "next": "http://127.0.0.1:8000/api/v1/blogs/?limit=1&offset=3",
    "previous": "http://127.0.0.1:8000/api/v1/blogs/?limit=1&offset=1",
    "results": [
        {
            "id": 4,
            "comments": [
                {
                    "id": 9,
                    "comment": "Gosh! Expensive!",
                    "blog": 4
                },
                {
                    "id": 10,
                    "comment": "I am not buying that!",
                    "blog": 4
                },
                {
                    "id": 11,
                    "comment": "That's crazy!",
                    "blog": 4
                }
            ],
            "blog_title": "A $2,300 Apple iPhone? Trump tariffs could make that happen.",
            "blog_body": "Tariffs could increase iPhone prices by up to 43% - Rosenblatt Securities\nSamsung may gain advantage due to low"
        }
    ]
}
```

Customized Pagination:

5. If you want only a certain model to have a certain pagination, you can customize the pagination for this model alone. So if we want Employee model to have a customized pagination, create a new file under API or under root directory PAGINATIONS.PY.

The class below will override the default class and method for pagination.



The screenshot shows a code editor with the following details:

- File Menu:** File, Edit, Selection, View, ...
- Search Bar:** DjangoREST_APIProject
- Explorer:** Shows the project structure under DJANGOREST_APIPROJECT, including api, blogs, and a paginations.py file (which is selected and highlighted with a red box).
- Code Editor:** The paginations.py file contains the following code:

```
from rest_framework.pagination import PageNumberPagination
from rest_framework.response import Response

# overrides the default pagination
class CustomPagination(PageNumberPagination):
    page_size_query_param = 'page_size'
    page_query_param = 'page-num'
    max_page_size = 1

    # overrides the default page result
    def get_paginated_response(self, data):
        return Response({
            'next': self.get_next_link(),
            'previous': self.get_previous_link(),
            'count': self.page.paginator.count,
            'page_size': self.page_size,
            'results': data
        })
```

6. So currently, EMPLOYEES page looks like this because, in SETTINGS.PY, we have the default_pagination_class.

127.0.0.1:8000/api/v1/employees/?offset=1

Django REST framework

api_djangoadmin

Api Root / Employee Viewset List

Employee Viewset List

OPTIONS GET ▾

« 1 2 3 »

GET /api/v1/employees/?offset=1

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 3,
    "next": "http://127.0.0.1:8000/api/v1/employees/?limit=1&offset=2",
    "previous": "http://127.0.0.1:8000/api/v1/employees/?limit=1",
    "results": [
        {
            "id": 6,
            "emp_id": "EMP004",
            "emp_name": "Arnel Zethus",
            "designation": "AI Engineer"
        }
    ]
}
```

Raw data HTML form

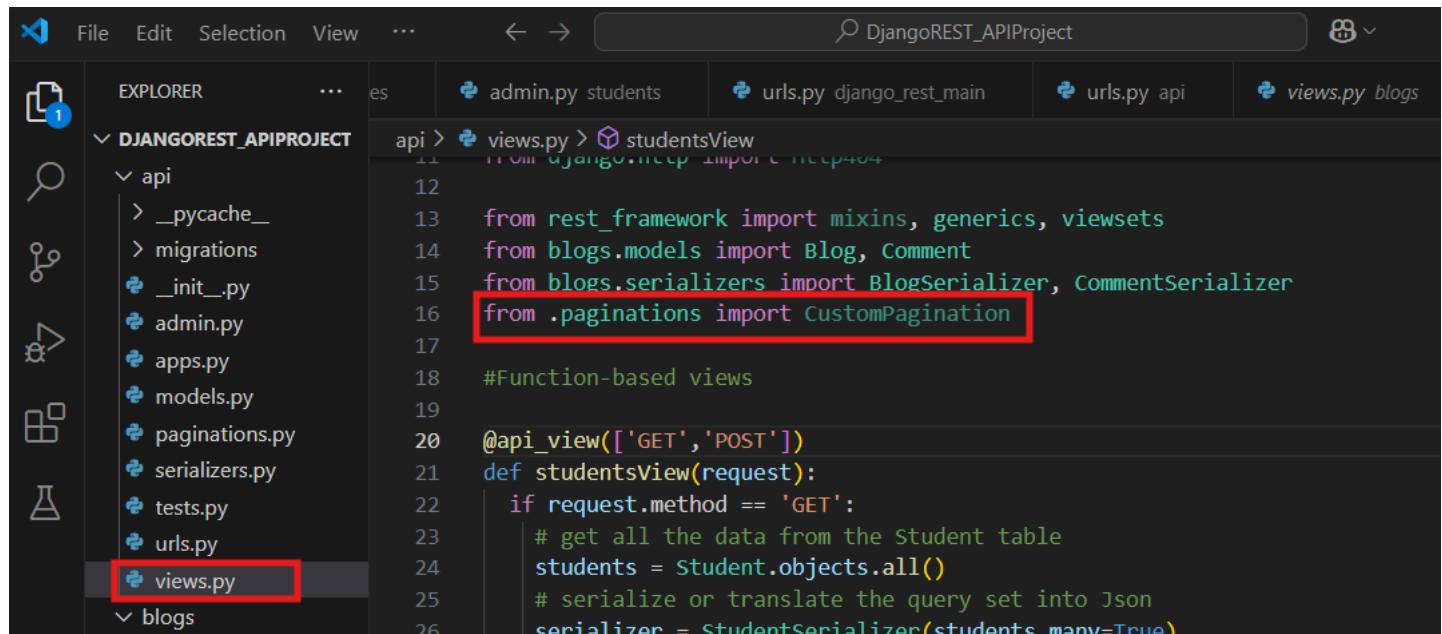
Emp id

Emp name

Designation

POST

So to use the customized pagination, PAGINATIONS.PY, we import this in VIEWS.PY:



```
File Edit Selection View ... ← → 🔍 DjangoREST_APIProject 🌐
```

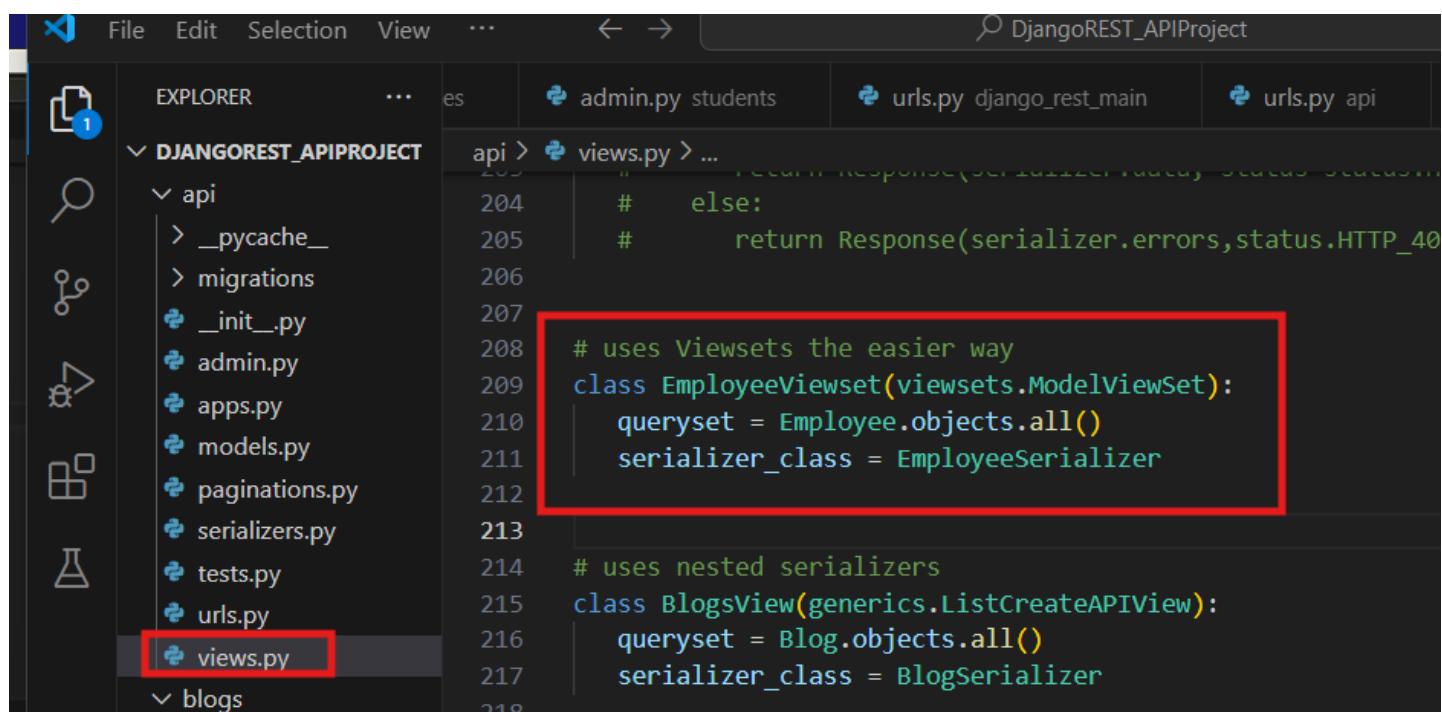
EXPLORER

DJANGOREST_APIPROJECT

- api
 - __pycache__
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - paginations.py
 - serializers.py
 - tests.py
 - urls.py
 - views.py**
- blogs

```
api > views.py > studentsView
11     from django.http import JsonResponse
12
13     from rest_framework import mixins, generics, viewsets
14     from blogs.models import Blog, Comment
15     from blogs.serializers import BlogSerializer, CommentSerializer
16     from .paginations import CustomPagination
17
18     #Function-based views
19
20     @api_view(['GET', 'POST'])
21     def studentsView(request):
22         if request.method == 'GET':
23             # get all the data from the Student table
24             students = Student.objects.all()
25             # serialize or translate the query set into Json
26             serializer = StudentSerializer(students, many=True)
```

Update the EmployeeView as:



```
File Edit Selection View ... ← → 🔍 DjangoREST_APIProject 🌐
```

EXPLORER

DJANGOREST_APIPROJECT

- api
 - __pycache__
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - paginations.py
 - serializers.py
 - tests.py
 - urls.py
 - views.py**
- blogs

```
api > views.py > ...
203     "return Response(serializer.data, status=status.HTTP_200_OK)
204     # else:
205     #     return Response(serializer.errors, status.HTTP_400_BAD_REQUEST)
206
207     # uses Viewsets the easier way
208     class EmployeeViewset(viewsets.ModelViewSet):
209         queryset = Employee.objects.all()
210         serializer_class = EmployeeSerializer
211
212     # uses nested serializers
213     class BlogsView(generics.ListCreateAPIView):
214         queryset = Blog.objects.all()
215         serializer_class = BlogSerializer
216
217     # uses nested serializers
218     class CommentView(generics.ListCreateAPIView):
```

TO:

File Edit Selection View ... ← → DjangoREST_APIProject

EXPLORER ... es admin.py students urls.py django_rest_main urls.py api views.py blogs

DJANGOREST_APIPROJECT

- api
 - __pycache__
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - paginations.py
 - serializers.py
 - tests.py
 - urls.py
 - views.py
- blogs

```

203     #     else:
204     #         return Response(serializer.errors, status.HTTP_400_BAD_REQUEST)
205
206
207
208     # uses Viewsets the easier way
209     class EmployeeViewSet(viewsets.ModelViewSet):
210         queryset = Employee.objects.all()
211         serializer_class = EmployeeSerializer
212
213         # uses the customized pagination instead of the default class
214         pagination_class = CustomPagination
215
216
217     # uses nested serializers
218     class BlogsView(generics.ListCreateAPIView):
219         queryset = Blog.objects.all()

```

This will result to:

127.0.0.1:8000/api/v1/employees/?page-num=3

Django REST framework api_djangoadmin

Api Root / Employee Viewset List

Employee Viewset List

OPTIONS GET

« 1 2 3 »

GET /api/v1/employees/?page-num=3

HTTP 200 OK

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "next": null,
  "previous": "http://127.0.0.1:8000/api/v1/employees/?page-num=2",
  "count": 3,
  "page_size": 1,
  "results": [
    {
      "id": 7,
      "emp_id": "EMP002",
      "emp_name": "Ziggy DartVader",
      "designation": "Web Designer"
    }
  ]
}
```

Raw data HTML form

Emp id:

Emp name:

Designation:

POST

