

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.

```
File Edit Selection View ... DjangoREST_APIProject
EXPLORER
DJANGOREST_APIPROJECT
  > api
  > 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
    > wsgi.py
  > employees

settings.py
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.

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:

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"
    }
  ]
}
```

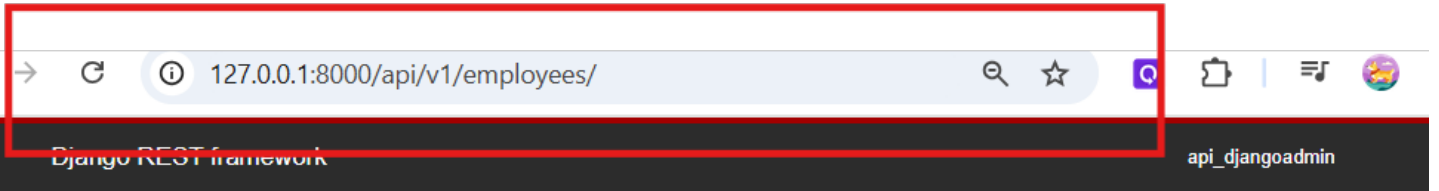
The screenshot shows a web browser at the URL `127.0.0.1:8000/api/v1/blogs/?page=2`. The page title is "Blogs" and the breadcrumb is "Api Root / Blogs". The interface includes "OPTIONS" and "GET" buttons. A pagination control shows pages 1, 2, and 3, with page 2 selected. The response headers are:

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

The JSON response body is:

```
{
  "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\r\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.



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	<input type="text"/>
Emp name	<input type="text"/>
Designation	<input type="text"/>

POST

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

The image shows a code editor window for a Django project named 'DjangoREST_APIProject'. The left sidebar shows the Explorer view with the following structure:

- EXPLORED 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

The main editor area shows the content of 'settings.py' with the following code:

```
119
120
121 # Static files (CSS, JavaScript, Images)
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.LimitOffsetPagination',
134     'PAGE_SIZE': 1,
135 }
```

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

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:

127.0.0.1:8000/api/v1/blogs/?limit=1&offset=2

api_djangoadmin

Api Root / Blogs

Blogs

OPTIONS GET

« 1 2 3 4 5 »

GET /api/v1/blogs/?limit=1&offset=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/?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\r\nSamsung may gain advantage due to lowe"
    }
  ]
}
```

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.

```
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
    __pycache__
    migrations
    __init__.py
    admin.py
settings.py paginations.py X models.py blogs admin.py blogs
api > paginations.py > CustomPagination > get paginated response
1 from rest_framework.pagination import PageNumberPagination
2 from rest_framework.response import Response
3
4 # overrides the default pagination
5 class CustomPagination(PageNumberPagination):
6     page_size_query_param = 'page_size'
7     page_query_param = 'page-num'
8     max_page_size = 1
9
10 # overrides the default page result
11 def get_paginated_response(self, data):
12     return Response({
13         'next': self.get_next_link(),
14         'previous': self.get_previous_link(),
15         'count': self.page.paginator.count,
16         'page_size': self.page_size,
17         'results': data
18     })
```

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:

The screenshot shows the Visual Studio Code editor interface for a Django REST API project. The Explorer sidebar on the left displays the project structure, with the `views.py` file in the `api` directory highlighted. The main editor window shows the code for `studentsView` in `views.py`. The following code is visible:

```
11 from django.http import Http404
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:

The screenshot shows the Visual Studio Code editor interface for a Django REST API project. The Explorer sidebar on the left displays the project structure, with the `views.py` file in the `api` directory highlighted. The main editor window shows the code for `EmployeeViewSet` and `BlogAPIView` in `views.py`. The following code is visible:

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

TO:

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

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

