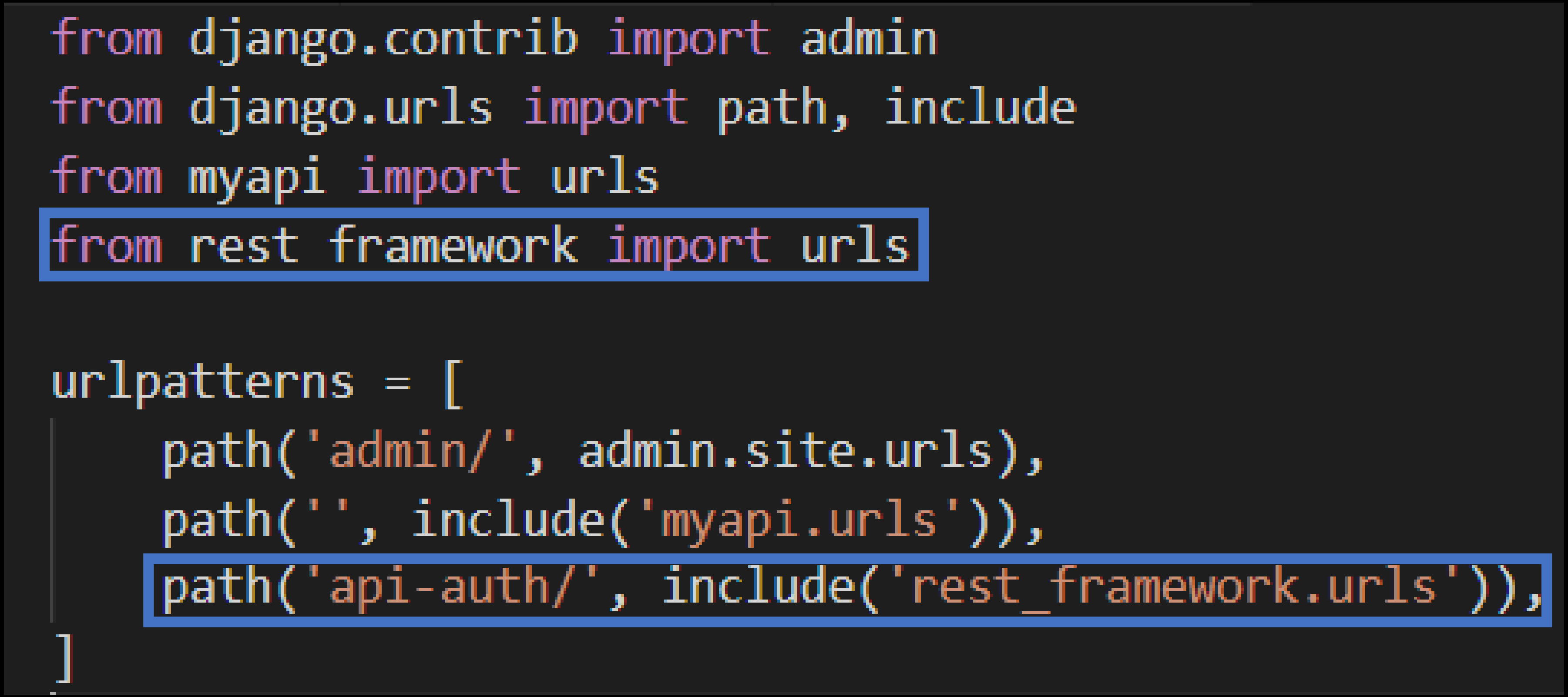
프로젝트 만들어보기

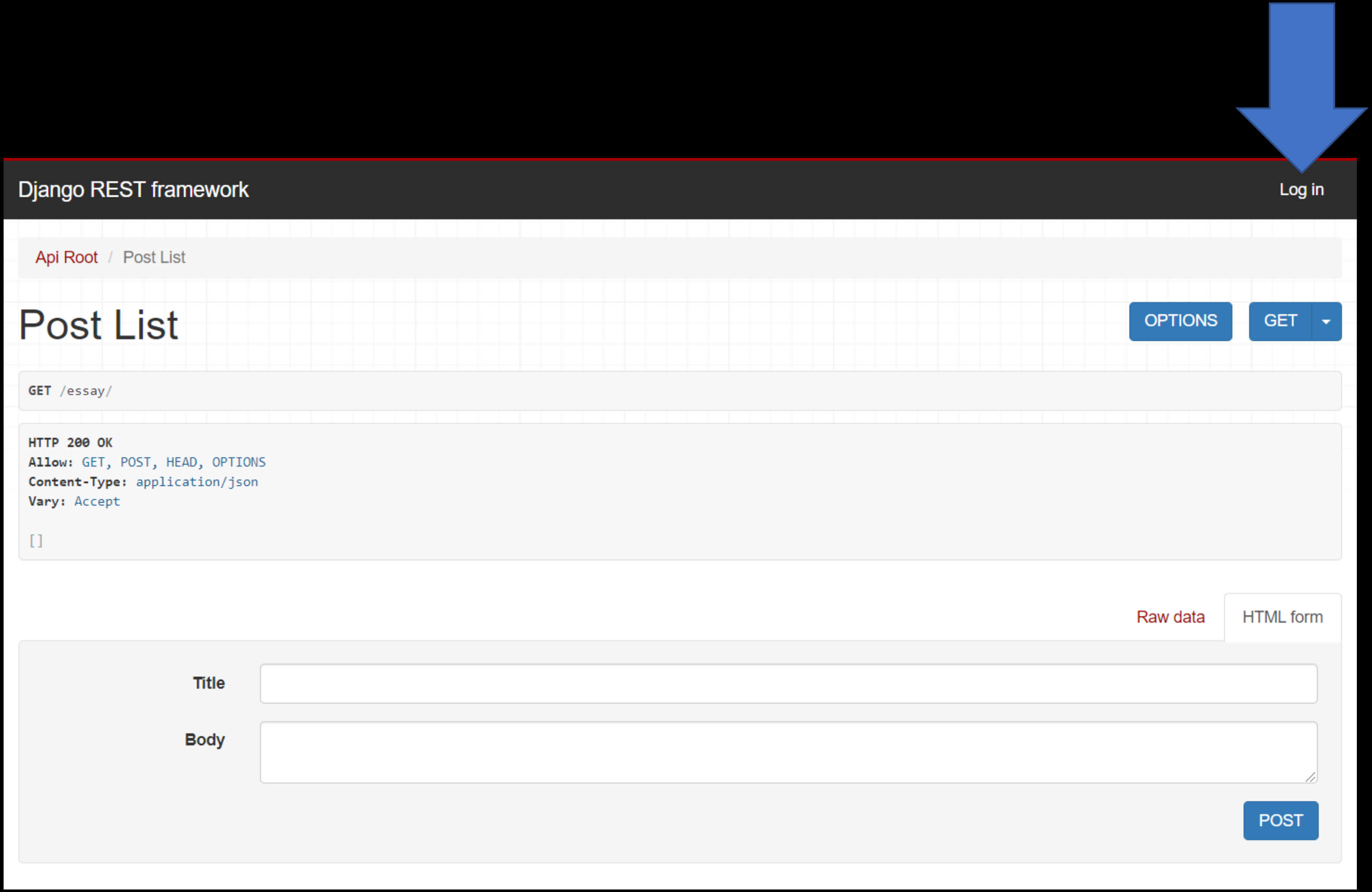
로그인 /로그아웃

위치 : prj/urls.py



```
from django.contrib import admin
from django.urls import path, include
from myapi import urls
from rest framework import urls

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('myapi.urls')),
    path('api-auth/', include('rest_framework.urls')),
]
```

로그인 /로그아웃



Post List

```
GET /essay/
```

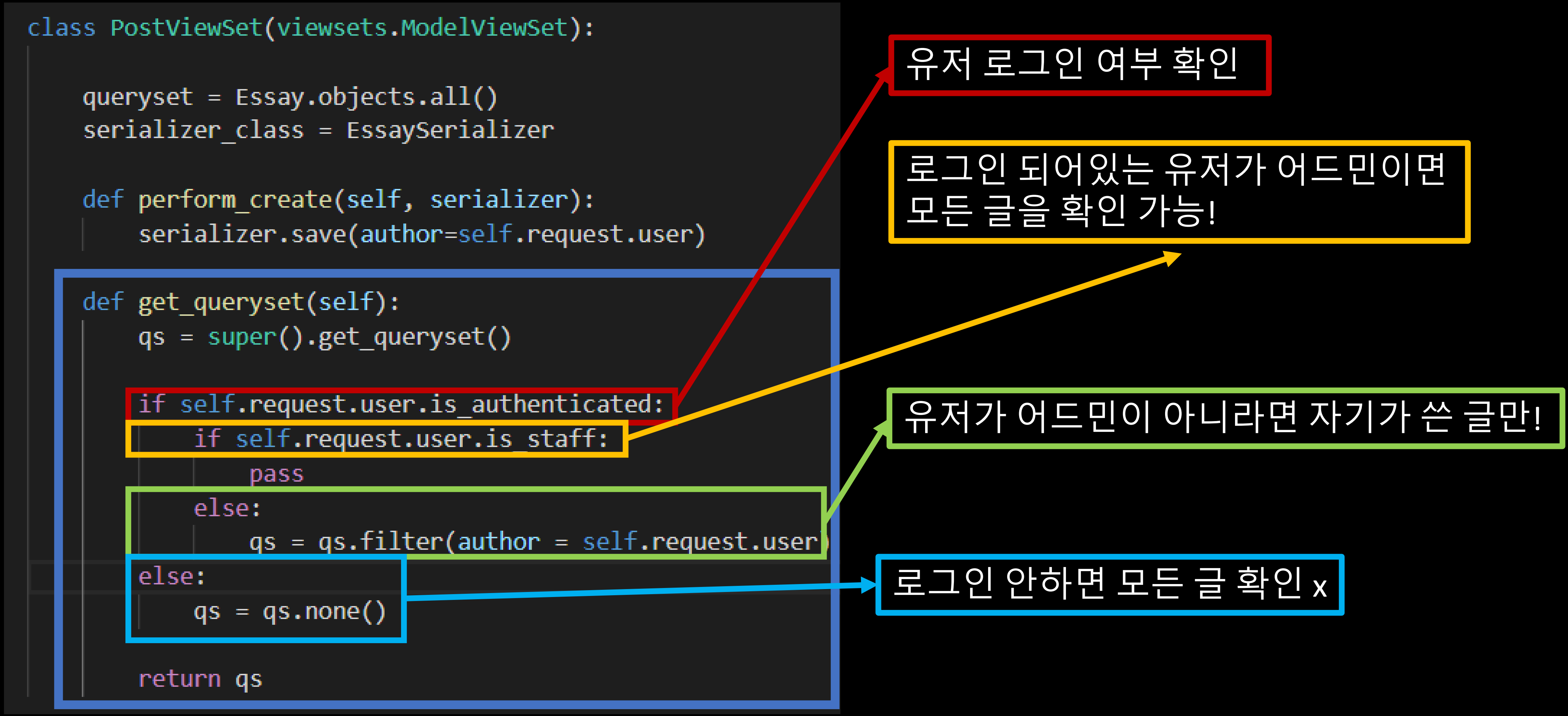
```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[]
```

Title

Body

필터링(내글만보기)

위치 : myapi/veiws.py



```
class PostViewSet(viewsets.ModelViewSet):

    queryset = Essay.objects.all()
    serializer_class = EssaySerializer

    def perform_create(self, serializer):
        serializer.save(author=self.request.user)

    def get_queryset(self):
        qs = super().get_queryset()

        if self.request.user.is_authenticated:
            if self.request.user.is_staff:
                pass
            else:
                qs = qs.filter(author = self.request.user)
        else:
            qs = qs.none()

        return qs
```

유저 로그인 여부 확인

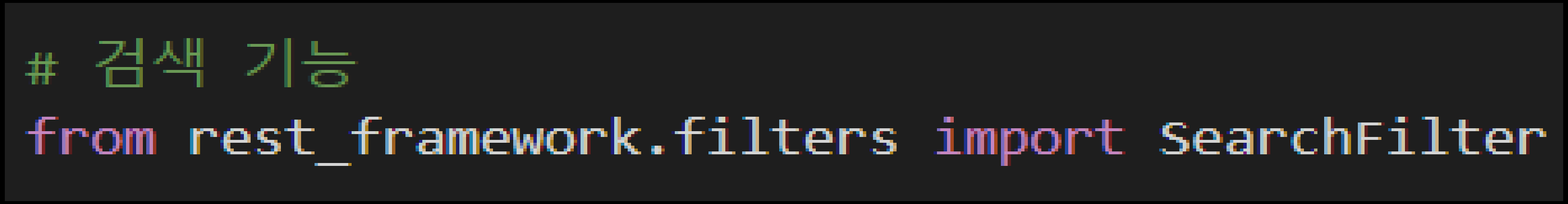
로그인 되어있는 유저가 어드민이면
모든 글을 확인 가능!

유저가 어드민이 아니라면 자기가 쓴 글만!

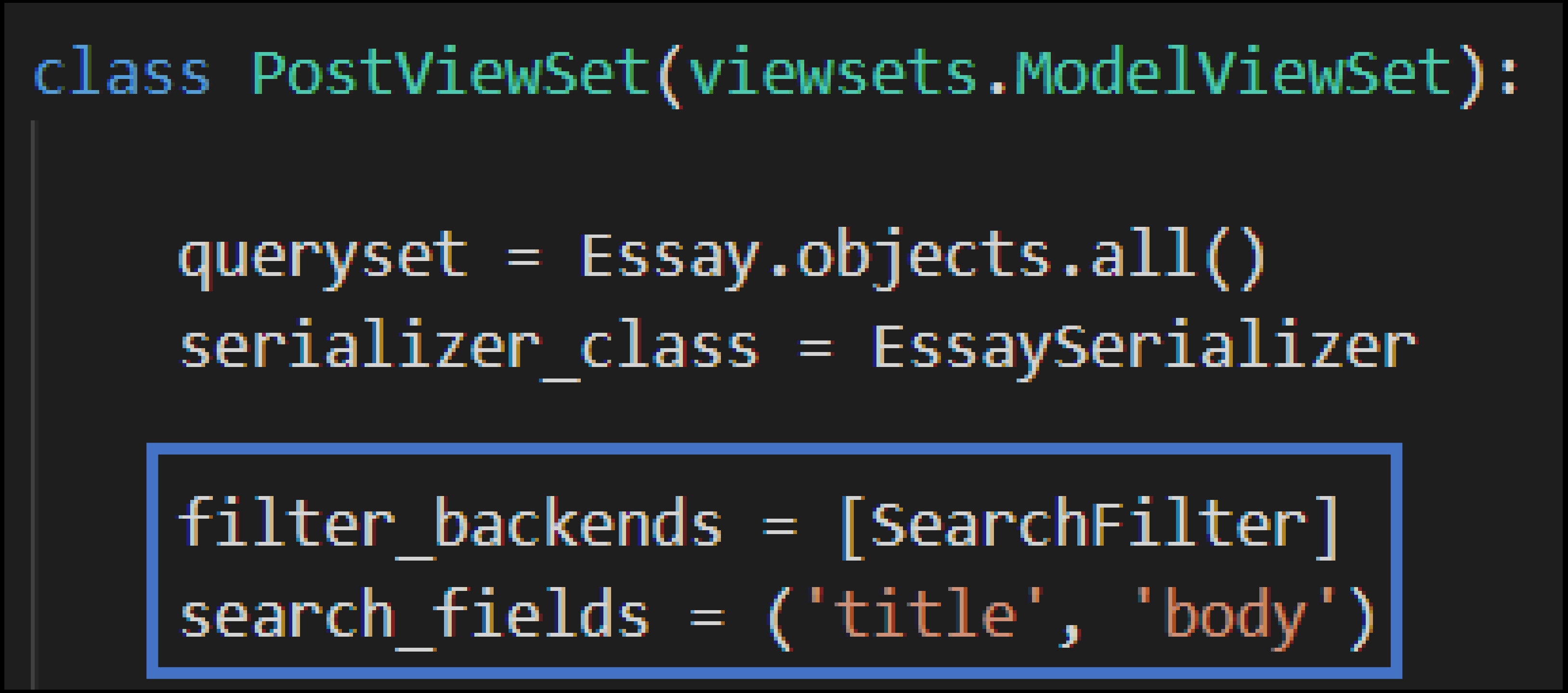
로그인 안하면 모든 글 확인 x

서칭(검색하기)

위치 : myapi/veiws.py

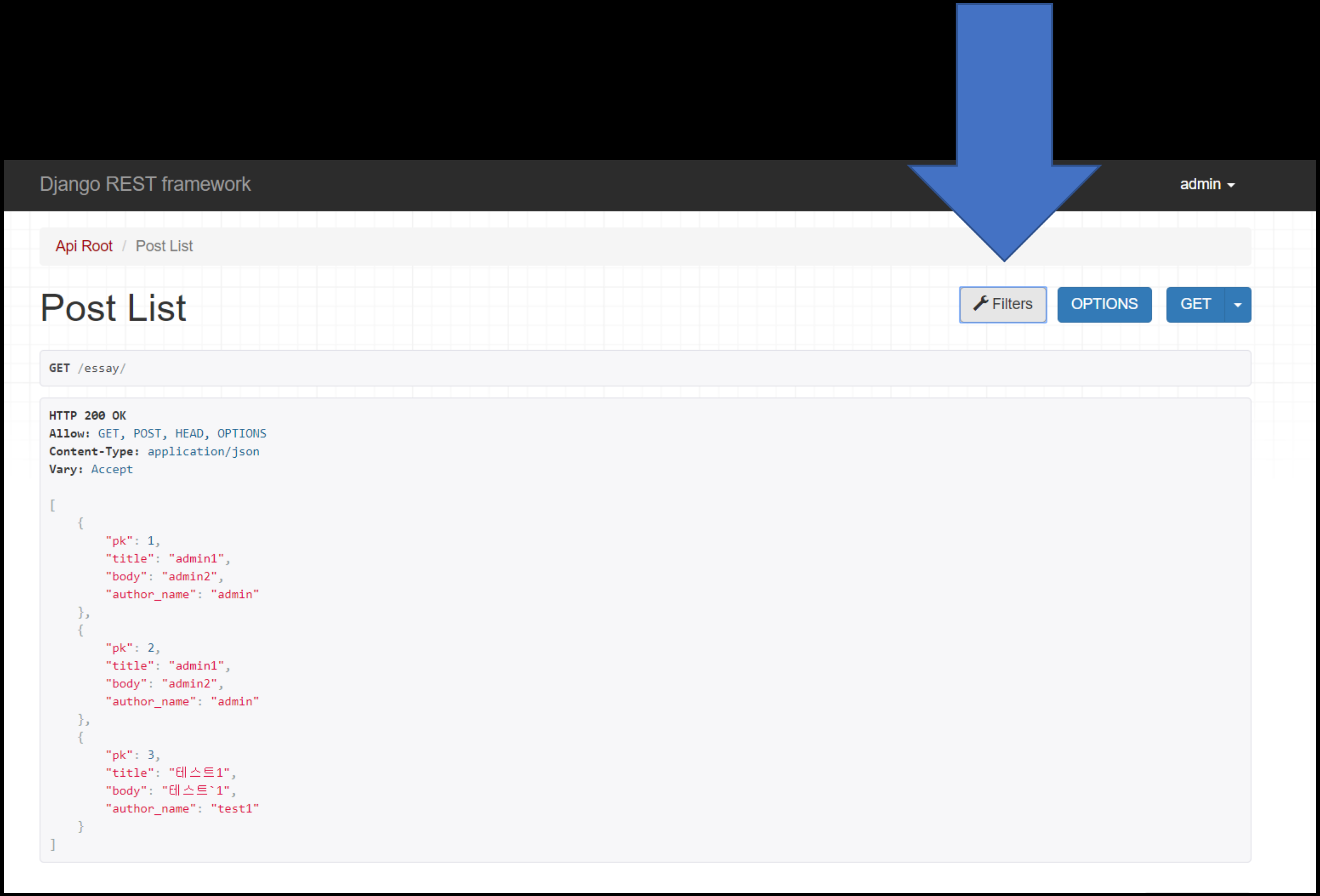


```python
# 검색 기능
from rest_framework.filters import SearchFilter
```



```python
class PostViewSet(viewsets.ModelViewSet):

    queryset = Essay.objects.all()
    serializer_class = EssaySerializer

    filter_backends = [SearchFilter]
    search_fields = ('title', 'body')
```

서칭(검색하기)



Post List

```
GET /essay/
```

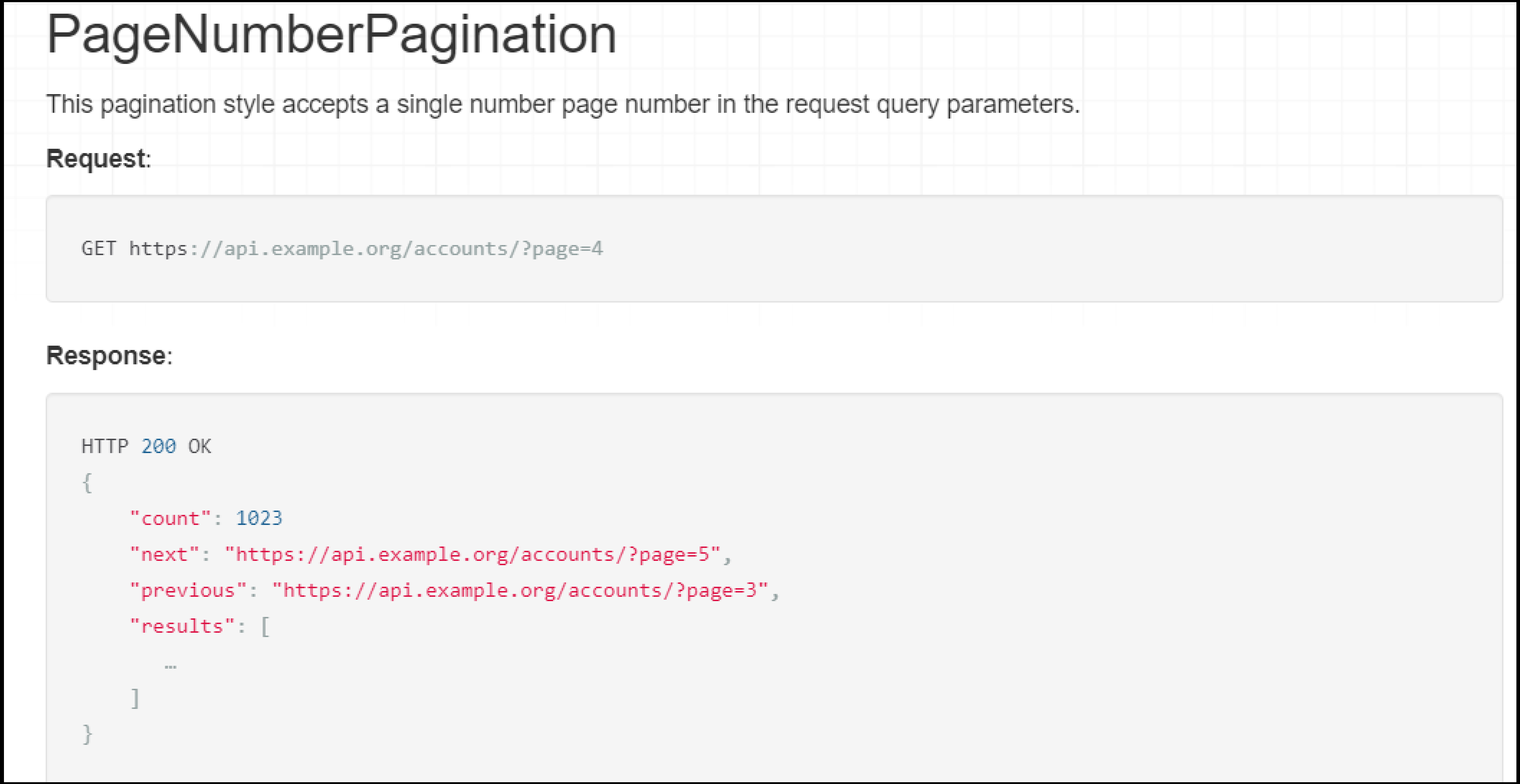
```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "pk": 1,
        "title": "admin1",
        "body": "admin2",
        "author_name": "admin"
    },
    {
        "pk": 2,
        "title": "admin1",
        "body": "admin2",
        "author_name": "admin"
    },
    {
        "pk": 3,
        "title": "테스트1",
        "body": "테스트`1",
        "author_name": "test1"
    }
]
```

페이지네이션

1. PageNumberPagination
2. LimitOffetPagination
3. CursorPagination
4. CustomizedPagination

페이지네이션

1.PageNumberPagination



PageNumberPagination

This pagination style accepts a single number page number in the request query parameters.

Request:

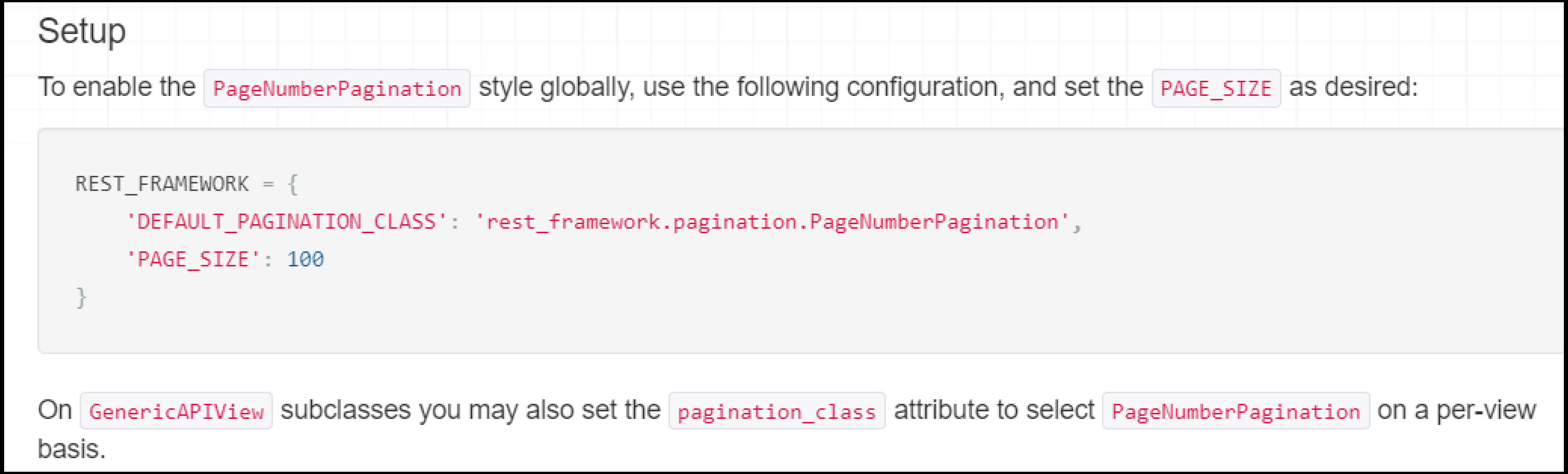
```
GET https://api.example.org/accounts/?page=4
```

Response:

```
HTTP 200 OK
{
    "count": 1023
    "next": "https://api.example.org/accounts/?page=5",
    "previous": "https://api.example.org/accounts/?page=3",
    "results": [
       …
    ]
}
```

페이지네이션

1.PageNumberPagination



Setup

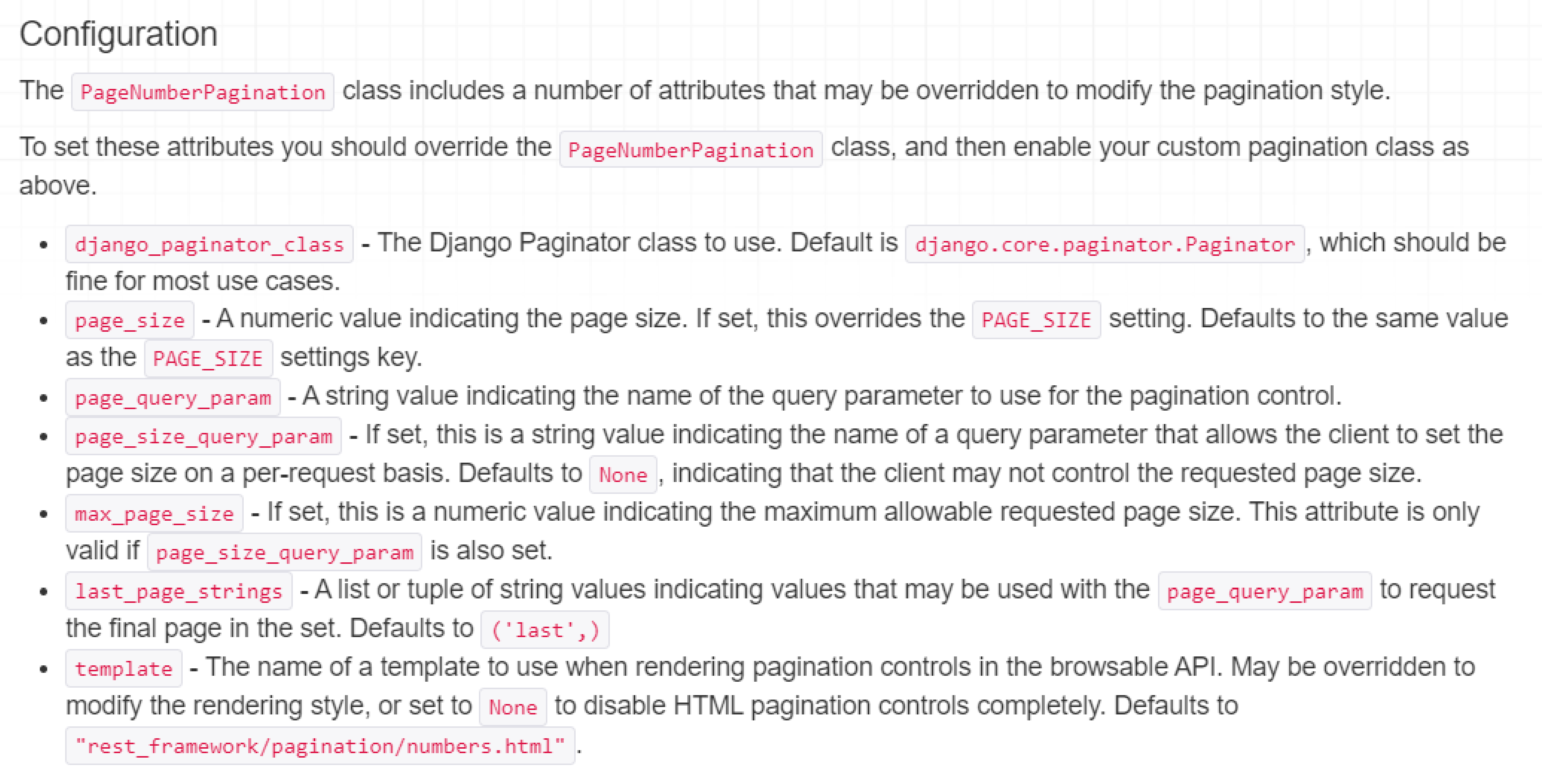
To enable the `PageNumberPagination` style globally, use the following configuration, and set the `PAGE_SIZE` as desired:

```
REST_FRAMEWORK = {
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 100
}
```

On `GenericAPIView` subclasses you may also set the `pagination_class` attribute to select `PageNumberPagination` on a per-view basis.

페이지네이션

1.PageNumberPagination



Configuration

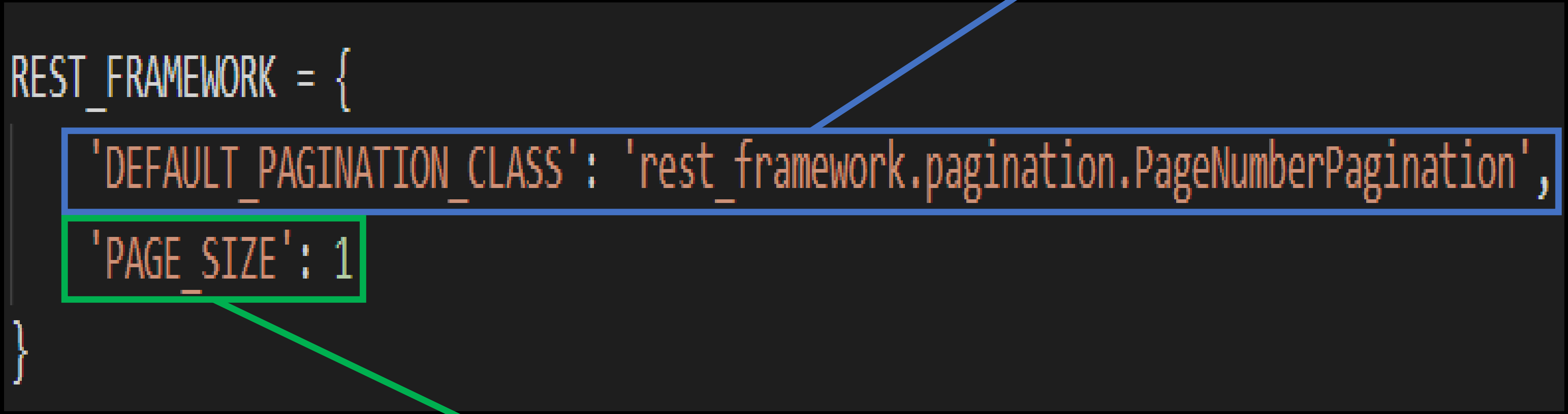
The `PageNumberPagination` class includes a number of attributes that may be overridden to modify the pagination style.

To set these attributes you should override the `PageNumberPagination` class, and then enable your custom pagination class as above.

- `django_paginator_class` - The Django Paginator class to use. Default is `django.core.paginator.Paginator`, which should be fine for most use cases.
- `page_size` - A numeric value indicating the page size. If set, this overrides the `PAGE_SIZE` setting. Defaults to the same value as the `PAGE_SIZE` settings key.
- `page_query_param` - A string value indicating the name of the query parameter to use for the pagination control.
- `page_size_query_param` - If set, this is a string value indicating the name of a query parameter that allows the client to set the page size on a per-request basis. Defaults to `None`, indicating that the client may not control the requested page size.
- `max_page_size` - If set, this is a numeric value indicating the maximum allowable requested page size. This attribute is only valid if `page_size_query_param` is also set.
- `last_page_strings` - A list or tuple of string values indicating values that may be used with the `page_query_param` to request the final page in the set. Defaults to `('last',)`
- `template` - The name of a template to use when rendering pagination controls in the browsable API. May be overridden to modify the rendering style, or set to `None` to disable HTML pagination controls completely. Defaults to `"rest_framework/pagination/numbers.html"`.

페이지네이션

위치 : prj/settings.py



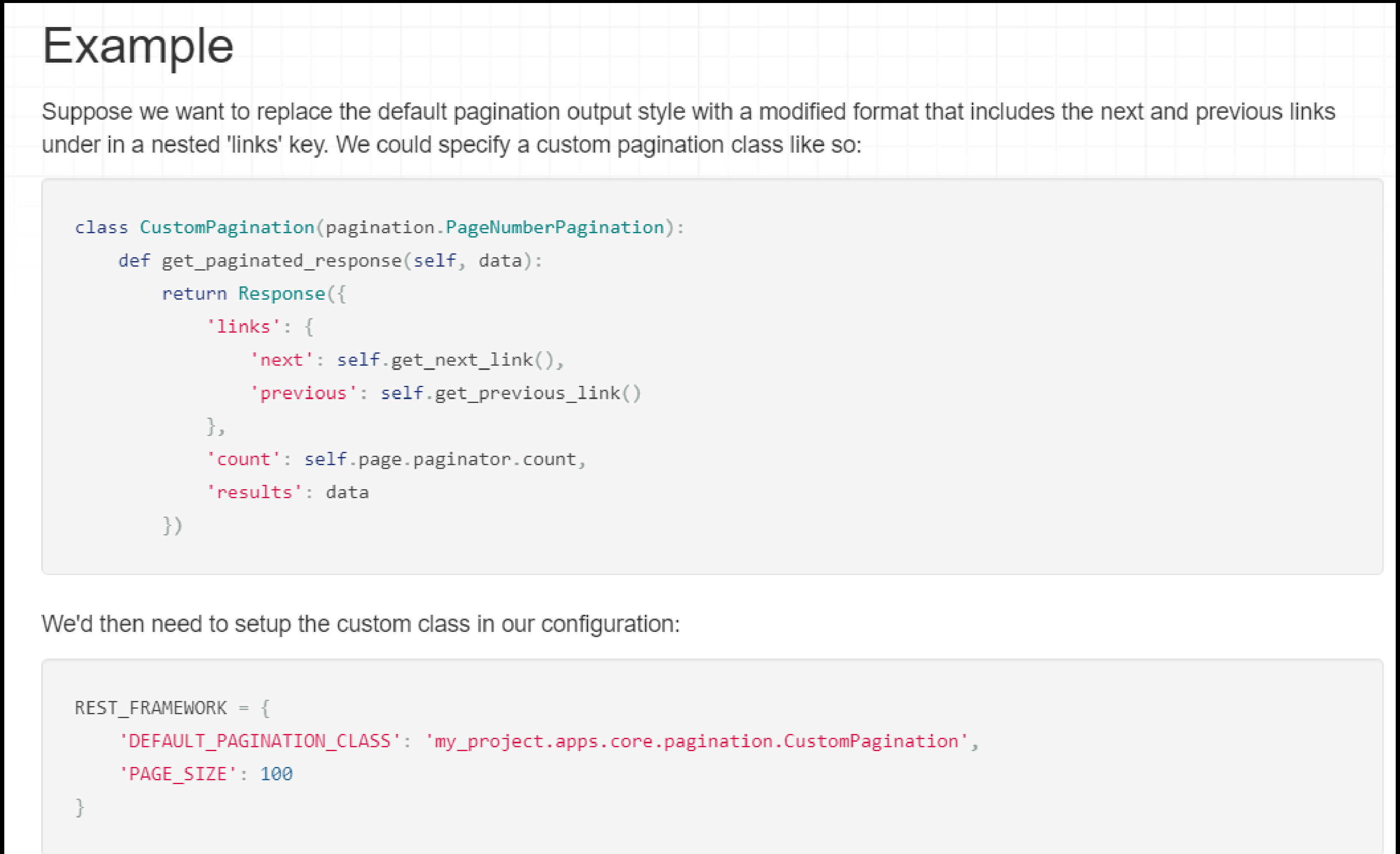
```
REST_FRAMEWORK = {
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 1
}
```

페이지넘버 페이지네이션을 사용하겠다.

한 페이지에 몇 개의 게시물을 노출하겠다!

페이지네이션

4. CustomizePagination



Example

Suppose we want to replace the default pagination output style with a modified format that includes the next and previous links under in a nested 'links' key. We could specify a custom pagination class like so:

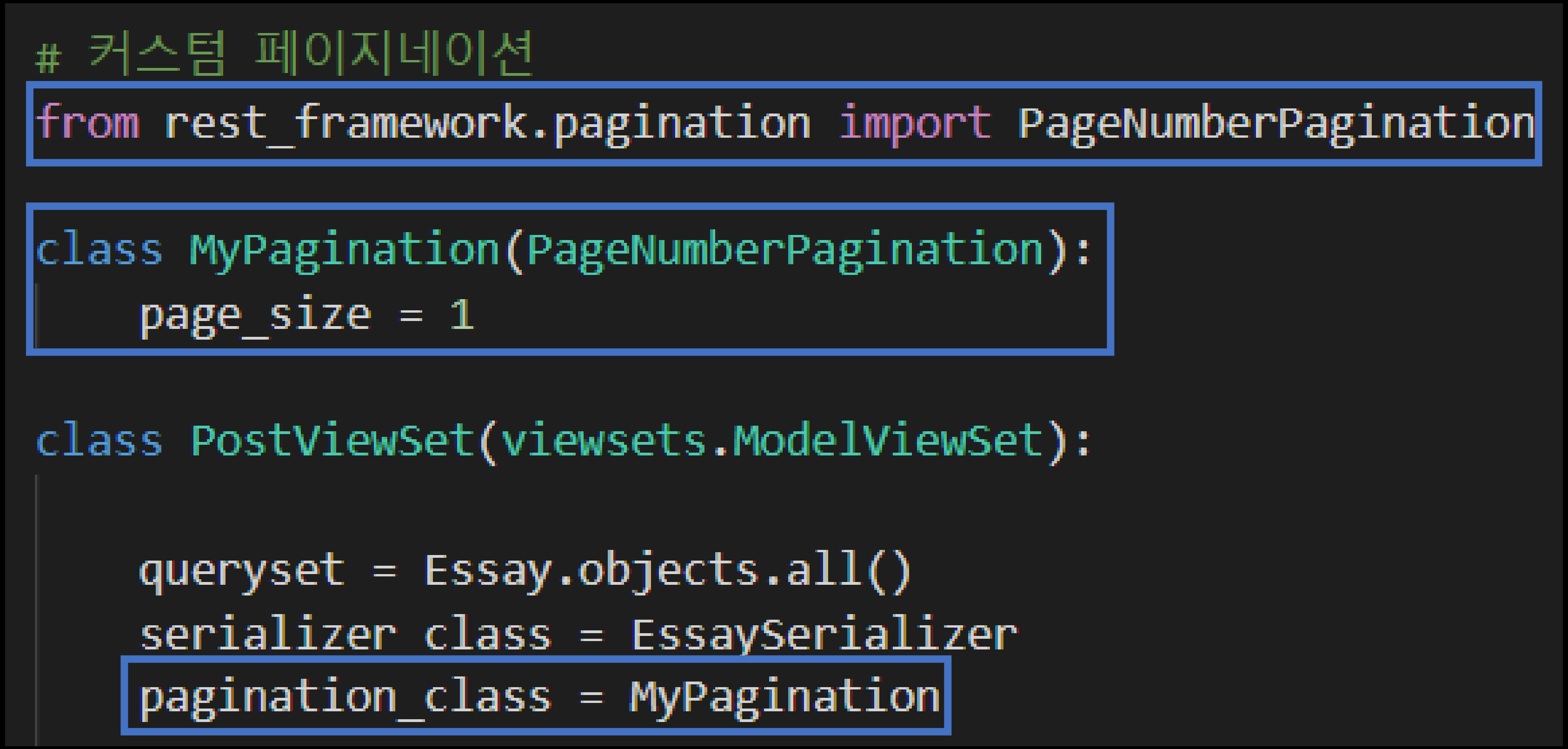
```
class CustomPagination(pagination.PageNumberPagination):
    def get_paginated_response(self, data):
        return Response({
            'links': {
                'next': self.get_next_link(),
                'previous': self.get_previous_link()
            },
            'count': self.page.paginator.count,
            'results': data
        })
```

We'd then need to setup the custom class in our configuration:

```
REST_FRAMEWORK = {
    'DEFAULT_PAGINATION_CLASS': 'my_project.apps.core.pagination.CustomPagination',
    'PAGE_SIZE': 100
}
```

페이지네이션

위치 : myapi/views.py



```python
# 커스텀 페이지네이션
from rest_framework.pagination import PageNumberPagination

class MyPagination(PageNumberPagination):
    page_size = 1

class PostViewSet(viewsets.ModelViewSet):

    queryset = Essay.objects.all()
    serializer_class = EssaySerializer
    pagination_class = MyPagination
```