Ask Company

리액트와 함께 장고 시작하기 / 장고 DRF

mixins 상속을 통한 APIView

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

DRF에서 지원하는 mixins

CreateModelMixin

ListModelMixin

RetrieveModelMixin

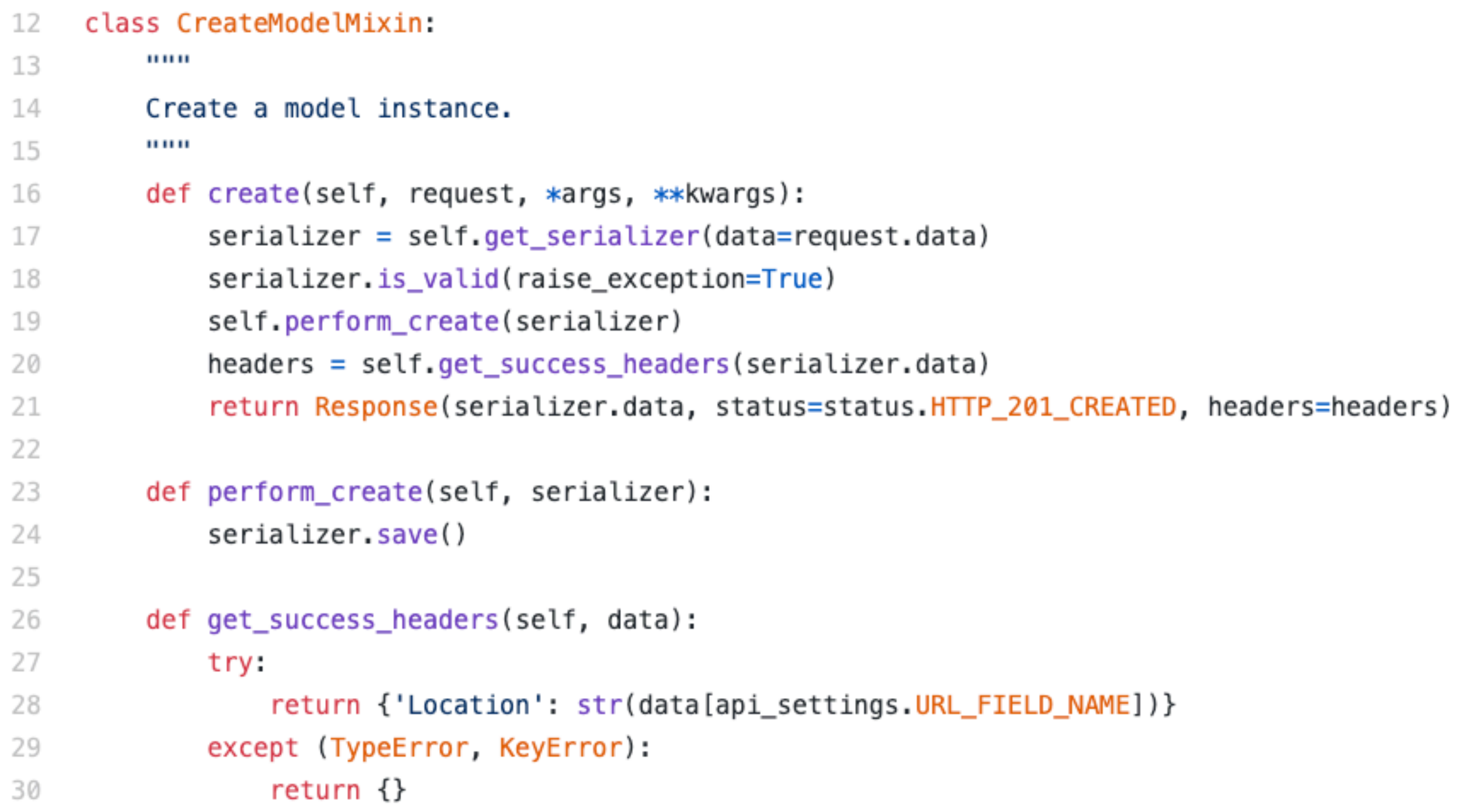
UpdateModelMixin

DestroyModelMixin

필요에 의해 다양한 믹스인을 만드실 수 있어요.

https://github.com/encode/django-rest-framework/blob/3.11.0/rest_framework/mixins.py

CreateModelMixin



```python
class CreateModelMixin:
    """
    Create a model instance.
    """
    def create(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        self.perform_create(serializer)
        headers = self.get_success_headers(serializer.data)
        return Response(serializer.data, status=status.HTTP_201_CREATED, headers=headers)

    def perform_create(self, serializer):
        serializer.save()

    def get_success_headers(self, data):
        try:
            return {'Location': str(data[api_settings.URL_FIELD_NAME])}
        except (TypeError, KeyError):
            return {}
```

https://github.com/encode/django-rest-framework/blob/3.11.0/rest_framework/mixins.py#L12

예시) method별 로직 연결 (1/2)

```python
from rest_framework import generics
from rest_framework import mixins

class PostListAPIView(mixins.ListModelMixin, mixins.CreateModelMixin,
                      generics.GenericAPIView):
    queryset = Post.objects.all()
    serializer_class = PostSerializer

    def get(self, request, *args, **kwargs):
        return self.list(request, *args, **kwargs)

    def post(self, request, *args, **kwargs):
        return self.create(request, *args, **kwargs)
```

매번 직접 연결을 할려고 하니
번거롭습니다. ☹

예시) method별 로직 연결 (2/2)

```python
from rest_framework import generics
from rest_framework import mixins

class PostDetailAPIView(mixins.RetrieveModelMixin, mixins.UpdateModelMixin,
                        mixins.DestroyModelMixin, generics.GenericAPIView):
    queryset = Post.objects.all()
    serializer_class = PostSerializer

    def get(self, request, *args, **kwargs):
        return self.retrieve(request, *args, **kwargs)

    def put(self, request, *args, **kwargs):
        return self.update(request, *args, **kwargs)

    def delete(self, request, *args, **kwargs):
        return self.destroy(request, *args, **kwargs)
```

매번 직접 연결을 할려고 하니
번거롭습니다. ☹

여러 generics APIView (모두 GenericsAPIView 상속)

- generics.CreateAPIView : post → create
- generics.ListAPIView : get → list
- generics.RetrieveAPIView : get → retrieve
- generics.DestroyAPIView : delete → destroy
- generics.UpdateAPIView : put → update, patch → partial_update
- generics.ListCreateAPIView : get → list, post → create
- generics.RetrieveUpdateAPIView : get → retrieve, put → update, patch → partial_update
- generics.RetrieveDestroyAPIView : get → retrieve, delete → destroy
- generics.RetrieveUpdateDestroyAPIView
  : get → retrieve, put → update, patch → partial_update, delete → destroy

https://github.com/encode/django-rest-framework/blob/3.11.0/rest_framework/generics.py

generic APIView 활용 (1/2)

```
from rest_framework import generics

class PostListAPIView(generics.ListCreateAPIView):
    queryset = Post.objects.all()
    serializer_class = PostSerializer
```

So Simple !!!

```
# rest_framework/generics.py
class ListCreateAPIView(mixins.ListModelMixin,
                        mixins.CreateModelMixin,
                        GenericAPIView):

    def get(self, request, *args, **kwargs):
        return self.list(request, *args, **kwargs)

    def post(self, request, *args, **kwargs):
        return self.create(request, *args, **kwargs)
```

generic APIView 활용 (2/2)

```python
from rest_framework import generics

class PostDetailAPIView(generics.RetrieveUpdateDestroyAPIView):
    queryset = Post.objects.all()
    serializer_class = PostSerializer
```

```python
# rest_framework/generics.py
class RetrieveUpdateDestroyAPIView(mixins.RetrieveModelMixin,
                                   mixins.UpdateModelMixin, mixins.DestroyModelMixin,
                                   GenericAPIView):

    def get(self, request, *args, **kwargs):
        return self.retrieve(request, *args, **kwargs)

    def put(self, request, *args, **kwargs):
        return self.update(request, *args, **kwargs)

    def patch(self, request, *args, **kwargs):
        return self.partial_update(request, *args, **kwargs)

    def delete(self, request, *args, **kwargs):
        return self.destroy(request, *args, **kwargs)
```

다양한 View의 구현 방법 → 중복을 줄이기

상황에 맞춰 다양한 방법으로 View를 구현하시어, 생산성을 극대화시켜주세요.

다양한 ViewSet (다음 에피소드)

다양한 APIView, @api_view

다양한 장고 View

인생은 짧습니다.
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

-

Ask Company