



CI Python Linter

```
1 from django.apps import AppConfig
2
3
4 class ProfilesConfig(AppConfig):
5     default_auto_field = "django.db.models.BigAutoField"
6     name = "profiles"
7 |
```

Settings:



Results:

All clear, no errors found

```

1 from django.db import models
2 from django.db.models.signals import post_save
3 from django.contrib.auth.models import User
4
5
6 class Profile(models.Model):
7     """
8     The model for the profiles.
9     """
10     owner = models.OneToOneField(User, on_delete=models.CASCADE)
11     created_at = models.DateTimeField(auto_now_add=True)
12     updated_at = models.DateTimeField(auto_now=True)
13     name = models.CharField(max_length=255, blank=True)
14     contact = models.TextField(blank=True)
15     image = models.ImageField(
16         upload_to='images/', default='../profile_picture.jpg'
17     )
18
19     class Meta:
20         ordering = ['-created_at']
21
22     def __str__(self):
23         return f'{self.owner}'s profile"
24
25
26 def create_profile(sender, instance, created, **kwargs):
27     if created:
28         Profile.objects.create(owner=instance)
29
30
31 post_save.connect(create_profile, sender=User)
32

```

Settings:



Results:

All clear, no errors found

```

1 from users.models import Profile
2 from followers.models import Follower
3
4
5
6 class ProfileSerializer(serializers.ModelSerializer):
7     """
8     Serializer for the Profile model.
9     """
10    owner = serializers.ReadOnlyField(source='owner.username')
11    is_owner = serializers.SerializerMethodField()
12    following_id = serializers.SerializerMethodField()
13    posts_count = serializers.ReadOnlyField()
14    followers_count = serializers.ReadOnlyField()
15    following_count = serializers.ReadOnlyField()
16
17    def get_is_owner(self, obj):
18        request = self.context['request']
19        return request.user == obj.owner
20
21    def get_following_id(self, obj):
22        user = self.context['request'].user
23        if user.is_authenticated:
24            following = Follower.objects.filter(
25                owner=user, followed=obj.owner
26            ).first()
27            return following.id if following else None
28        return None
29
30    class Meta:
31        model = Profile
32        fields = (
33            'id', 'owner', 'created_at', 'updated_at', 'name',
34            'content', 'image', 'is_owner', 'following_id',
35            'posts_count', 'followers_count', 'following_count',
36        )

```

Settings:



Results:

All clear, no errors found

```

1 from django.conf import settings
2 from django.conf.urls import url
3
4 urlpatterns = [
5     url(r'^$', views.index, name='index'),
6     url(r'^about/$', views.about, name='about'),
7 ]

```

Settings:



Results:

All clear, no errors four

```

14 queryset = Profile.objects.annotate(
15     posts_count=Count('owner_post', distinct=True),
16     followers_count=Count('owner_followed', distinct=True),
17     following_count=Count('owner_following', distinct=True)
18 ).order_by('-created_at')
19 serializer_class = ProfileSerializer
20 filter_backends = (
21     filters.OrderingFilter,
22     DjangoFilterBackend,
23 )
24 filterset_fields = (
25     'owner_following_followed_profile',
26     'owner_followed_owner_profile',
27 )
28 ordering_fields = (
29     'posts_count',
30     'followers_count',
31     'following_count',
32     'owner_following_created_at',
33     'owner_followed_created_at',
34 )

```

```

37 class ProfileDetail(generic.RetrieveUpdateDestroyView):
38     """
39     Retrieve or update a profile if you're the owner.
40     """
41     permission_classes = [IsOwnerOrReadOnly]
42     queryset = Profile.objects.annotate(
43         posts_count=Count('owner_post', distinct=True),
44         followers_count=Count('owner_followed', distinct=True),
45         following_count=Count('owner_following', distinct=True)
46     ).order_by('-created_at')
47     serializer_class = ProfileSerializer

```

Settings:



Results:

All clear, no errors found