When to use Serializer's create() and ModelViewset's perform_create()

Asked 8 years ago Modified 3 years, 2 months ago Viewed 71k times

150

I want to clarify the given documentation of Django-rest-framework regarding the creation of a model object. So far I have found that there are 3 approaches on how to handle such events.

1. The Serializer's `create()` method. Here is the documentation

```
class CommentSerializer(serializers.Serializer):

    def create(self, validated_data):
        return Comment.objects.create(**validated_data)
```

2. The ModelViewset `create()` method. Documentation

```
class AccountViewSet(viewsets.ModelViewSet):

    queryset = Account.objects.all()
    serializer_class = AccountSerializer
    permission_classes = [IsAccountAdminOrReadOnly]
```

3. The ModelViewset `perform_create()` method. Documentation

```
class SnippetViewSet(viewsets.ModelViewSet):

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

These three approaches are important depending on your application environment. But when do we need to use each `create() / perform_create()` function? On the other hand, I found some accounts that two create methods were called for a single post request the `ModelViewSet`'s `create()` and serializer's `create()`.

python django serialization django-rest-framework

Share Improve this question

Follow

edited Sep 9, 2021 at 4:18

Sabito

5,034 ● 9 ● 37 ● 65

asked Dec 12, 2016 at 3:59

Shift 'n Tab

9,415 ● 14 ● 78 ● 123

2 Answers

Sorted by: Highest score (default)

194

1. You would use `create(self, validated_data)` to add any extra details into the object before saving AND "prod" values into each model field just like `**validated_data` does. Ideally speaking, you want to do this form of "prodding" only in ONE location so the `create` method in your `CommentSerializer` is the best place. On top of this, you might want to also call external apis to create user accounts on their side just before saving your accounts into your own database. You should use this `create` function in conjunction with `ModelViewSet`. Always think - "Thin views, Thick serializers".

   Example:

   ```
   def create(self, validated_data):
       email = validated_data.get("email", None)
       validated.pop("email")
       # Now you have a clean valid email string
       # You might want to call an external API or modify another table
       # (eg. keep track of number of accounts registered.) or even
       # make changes to the email format.

       # Once you are done, create the instance with the validated data
       return models.YourModel.objects.create(email=email, **validated_data)
   ```

2. The `create(self, request, *args, **kwargs)` function in the `ModelViewSet` is defined in the `CreateModelMixin` class which is the parent of `ModelViewSet`. `CreateModelMixin`'s main functions are these:

   ```
   from rest_framework import status
   from rest_framework.response import Response


   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()
   ```

   As you can see, the above `create` function takes care of calling validation on your serializer and producing the correct response. The beauty behind this, is that you can now isolate your application logic and NOT concern yourself about the mundane and repetitive validation calls and handling response output :). This works quite well in conjuction with the `create(self, validated_data)` found in the serializer (where your specific application logic might reside).

3. Now you might ask, why do we have a separate `perform_create(self, serializer)` function with just one line of code!?!? Well, the main reason behind this is to allow customizeability when calling the `save` function. You might want to supply extra data before calling `save` **(like `serializer.save(owner=self.request.user)`** and if we didn't have `perform_create(self, serializer)`, you would have to override the

`create(self, request, *args, **kwargs)` and that just defeats the purpose of having mixins doing the heavy and boring work.

Share Improve this answer Follow

edited Sep 9, 2021 at 4:14

Sabito
5,034 ● 9 ● 37 ● 65

answered Dec 12, 2016 at 4:30

Apoorv Kansal
3,280 ● 6 ● 28 ● 38

1 Hi! Thanks for your sharing your knowledge! About the `create(self, validated_data)` in the serializer, it means that it focuses on data validation logic? and more over it can help return the given serializer's data back to the response right? – Shift 'n Tab Dec 12, 2016 at 4:50

1 No so at this point, you already have passed all your validation. I am talking about how you might want to customize the validated data just before it is saved into a database. I will make an example in my answer. – Apoorv Kansal Dec 12, 2016 at 4:52

1 No worries - just added an example to give more context. – Apoorv Kansal Dec 12, 2016 at 4:58

2 Then this code `return models.YourModel.objects.create(email=email, **validated_data)` will save the data from database right? – Shift 'n Tab Dec 12, 2016 at 5:01

2 So the `create` function in the serializer itself is **only** called when you do `serializer.save()`. In your `create(self, request)` function inside (`AccountViewSet`), you are not calling `serializer.save()` at all and therefore, the only instance creation is happening with this call: `Account.objects.create_user(**serializer.validated_data)` . – Apoorv Kansal Dec 12, 2016 at 6:16

10

While Apoorv's answer is correct and very detailed, here's a quick answer:

- Override `perform_create()` when you want to change the "behind-the-scenes" behavior of how your object is created. For example, performing some extra actions before or after the object is created.
- Override `create()` when you want to modify the response. For example, if you want to re-structure the response, add extra data, extra headers, etc.

Share Improve this answer Follow

answered Sep 30, 2021 at 15:36

Ben Davis
13.7k ● 11 ● 57 ● 72

7 Which create() method? The one that belongs to serializer or to ModelViewSet? – Haliax Dec 13, 2021 at 9:30