[Tutorial 1: Serialization](https://www.django-rest-framework.org/tutorial/1-serialization/#tutorial-1-serialization)

One nice property that serializers have is that you can inspect all the fields in a serializer instance, by printing its representation. Open the Django shell with python manage.py shell, then try the following:

from snippets.serializers import SnippetSerializer

serializer = SnippetSerializer()

print(repr(serializer))

# SnippetSerializer():

# id = IntegerField(label='ID', read\_only=True)

# title = CharField(allow\_blank=True, max\_length=100, required=False)

# code = CharField(style={'base\_template': 'textarea.html'})

# linenos = BooleanField(required=False)

# language = ChoiceField(choices=[('Clipper', 'FoxPro'), ('Cucumber', 'Gherkin'), ('RobotFramework', 'RobotFramework'), ('abap', 'ABAP'), ('ada', 'Ada')...

# style = ChoiceField(choices=[('autumn', 'autumn'), ('borland', 'borland'), ('bw', 'bw'), ('colorful', 'colorful')...

It's important to remember that ModelSerializer classes don't do anything particularly magical, they are simply a shortcut for creating serializer classes:

1. An automatically determined set of fields.
2. Simple default implementations for the create() and update() methods.

[Tutorial 2: Requests and Responses](https://www.django-rest-framework.org/tutorial/2-requests-and-responses/#tutorial-2-requests-and-responses)

REST framework also introduces a Response object, which is a type of TemplateResponse that takes unrendered content and uses content negotiation to determine the correct content type to return to the client.

知道下面这个是怎么回事

- urlpatterns = format\_suffix\_patterns(urlpatterns)

- path('snippets/<int:pk>', views.snippet\_detail)

- def get(self, request, format=None)

[Tutorial 3: Class-based Views](https://www.django-rest-framework.org/tutorial/3-class-based-views/#tutorial-3-class-based-views)

任务一，在一个类中写好增删改查

class SnippetDetail(APIView):

"""

Retrieve, update or delete a snippet instance.

"""

def get\_object(self, pk):

try:

return Snippet.objects.get(pk=pk)

except Snippet.DoesNotExist:

raise Http404

def get(self, request, pk, format=None):

snippet = self.get\_object(pk)

serializer = SnippetSerializer(snippet)

return Response(serializer.data)

def put(self, request, pk, format=None):

snippet = self.get\_object(pk)

serializer = SnippetSerializer(snippet, data=request.data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

def delete(self, request, pk, format=None):

snippet = self.get\_object(pk)

snippet.delete()

return Response(status=status.HTTP\_204\_NO\_CONTENT)