1. Serializer

-Serialization​

-Serializer class

def get\_fieldname():

...

-Routing -> View -> Go to the database to get objects or QuerySet -> Convert the serializer class into a list, dictionary, ordered dictionary -> JSON processing

ser = DepartSerializer(instance= object)

-Data verification

-Serializer class

-Route- >View->request.data->validate(serializer class)->operate(db,serializer class)

ser = DepartSerializer(data=request.data)

ser.validated\_data

models.Depart.objects.create(\*\*ser.validated\_data)

ser = DepartModelSerializer(data=request.data)

ser.validated\_data

ser.save()

1. Custom Serializer + Fields

2. Custom Serializer + Field (Built-in + Regular Expression)

3. Custom Serializer + Field (built-in + regular) + Field Hook + Global Hook

4. Custom ModelSerializer + extra\_kwargs + save (more, pop; less, save parameters)

5. Custom ModelSerializer + FK => Automatically obtain related data depart => depart\_id

6. Custom ModelSerializer + M2M => Automatically obtain related data ListField or DictField + hook

7. Return value of save

instance = ser.save()

instance.id

instance.name

instance.age

8. Serialization return

-Verify Serializer + Serialize Serializer

class UusView(APIView):

def post(self, request, \*args, \*\*kwargs):

ser = UusModelSerializer(data=request.data)

if ser.is\_valid():

instance = ser.save()

print(instance)

xx = UusModelSerializer(instance=instance)

return Response(xx.data)

else:

return Response(ser.errors)

-Verify Serializer

-Verify Serializer

- Checksum, 5 fields

-Serialization , 3 fields

read\_only=True , only used in serialization

write\_only=True is only used when validating

class UusView(APIView):

def post(self, request, \*args, \*\*kwargs):

ser = UusModelSerializer(data=request.data)

if ser.is\_valid():

ser.save()

return Response(ser.data)

else:

return Response(ser.errors)

-Combination​

Create a user: {"user":"","password"}

-validate (serializer class)

-Database object = Connection data storage

- Let's talk about the newly added data return

Call (serializer class) again to serialize the newly added data database object.

{"user":"","password"}

{"id":1,"user":"","password"} + default generated data

The demand is here: write a serialization class to create a user.

Provided: {"name":"x3","age":11,"gender":1}

Returns: {"id":1, "name":"x3", "gender":"男",}

class NbUserInfo(models.Model):

name = models.CharField(verbose\_name=" name", max\_length=32)

age = models.IntegerField(verbose\_name=" age")

gender = models.SmallIntegerField(verbose\_name=" Gender", choices=((1, "Male"), (2, "Female")))

2. Case: Blog system

About the comments:

- Certified components

-Where to write

Single view:

class CommentView(APIView):

def get(self, request, blog\_id):

""" Comment list, logged in in order"""

class CreateComment(APIView):

authentication\_classes = ["..."]

def post(self):

pass

Comments List + Create Comment:

class CommentView(APIView):

authentication\_classes = ["..."] # None request.user = None

def get(self, request, blog\_id):

""" Comment list, logged in in order"""

def post(self,):

""" You must be logged in to create a comment"""

Task:

Pagination

- Blog List

- Comments list

view:

- APIView , 10 public

- ... , 5

- .. 5

- class(..):20

class UserView(ModelViewSet):

..

..