ModelSerializer

参考 https://www.django-rest-framework.org/api-guide/serializers/#modelserializer

ModelSerializer是Serializer的子类除了具有Serializer的功能外：

- 根据Model类，自动生成字段
- 为序列化器自动生成校验器
- 为序列化器提供了简单的create()和update()方法实现，所以可以直接save()

序列化器

需要解决2个问题：

- 依照那个Model类
- 需要哪些字段

```
from rest_framework import serializers
from .models import Employee

class EmpSerializer(serializers.ModelSerializer):
    class Meta:
        model = Employee
        fields = '__all__'

print('~' * 30)
print(EmpSerializer())
print('~' * 30)
```

```
EmpSerializer():
    emp_no = IntegerField(label='工号', max_value=2147483647, min_value=-2147483648, validators=[<UniqueValidator(queryset=Employee.objects.all())>])
    birth_date = DateField(label='生日')
    first_name = CharField(label='名', max_length=14)
    last_name = CharField(label='姓', max_length=16)
    gender = ChoiceField(choices=[(1, '男'), (2, '女')], label='性别', validators=[<django.core.validators.MinValueValidator object>, <django.core.validators.MaxValueValidator object>])
    hire_date = DateField()
```

```
class EmpSerializer(serializers.ModelSerializer):
    class Meta:
        model = Employee
        fields = ['emp_no', 'first_name', 'last_name'] # 指定字段
```

```
class EmpSerializer(serializers.ModelSerializer):
    class Meta:
        model = Employee
        exclude = ['gender'] # 排除某些字段
```

```python
class EmpSerializer(serializers.ModelSerializer):
    class Meta:
        model = Employee
        fields = '__all__'
        read_only_fields = ['hire_date'] # 单独指定readonly字段
```

```python
class EmpSerializer(serializers.ModelSerializer):
    class Meta:
        model = Employee
        fields = '__all__'
        read_only_fields = ['hire_date'] # 单独指定readonly字段
        extra_kwargs = {'gender': {'write_only': True}}
        # 额外字段选项定义，gender为write_only
```

序列化

```python
class EmpSerializer(serializers.ModelSerializer):
    class Meta:
        model = Employee
        fields = '__all__'
```

```python
import os
import django

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'salary.settings')
django.setup(set_prefix=False)
# 所有测试代码，都要在上面4行之下

from employee.models import Employee
from employee.serializers import EmpSerializer

emgr = Employee.objects
emp = emgr.get(pk=10010)
serializer = EmpSerializer(emp)
print(serializer.data)
```

```
{'emp_no': 10010, 'birth_date': '1963-06-01', 'first_name': 'Duangkaew',
'last_name': 'Piveteau', 'gender': 2, 'hire_date': '1989-08-24'}
```

```python
emgr = Employee.objects
emps = emgr.filter(pk__gt=10018)
serializer = EmpSerializer(emps, many=True) # 多个
print(serializer.data)
```

```
[OrderedDict([('emp_no', 10019), ('birth_date', '1953-01-23'), ('first_name', 'Lillian'), ('last_name', 'Haddadi'), ('gender', 1), ('hire_date', '1999-04-30')]), OrderedDict([('emp_no', 10020), ('birth_date', '1952-12-24'), ('first_name', 'Mayuko'), ('last_name', 'Warwick'), ('gender', 1), ('hire_date', '1991-01-26')]), OrderedDict([('emp_no', 10021), ('birth_date', '1963-06-01'), ('first_name', 'si'), ('last_name', 'li'), ('gender', 1), ('hire_date', '1989-08-24')])]
```

反序列化

ModelSerializer提供了create、update方法。

```python
import os
import django

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'salary.settings')
django.setup(set_prefix=False)
# 所有测试代码，都要在上面4行之下

from employee.models import Employee
from employee.serializers import EmpSerializer

data = {
    'emp_no': 10022, 'birth_date': '1963-06-01',
    'first_name': 'wu', 'last_name': 'wang',
    'gender': 1, 'hire_date': '1989-08-24',
} # 注意，这是字典数据是data，不是Employee的实例

# 只有data，是新增
serializer = EmpSerializer(data=data)
serializer.is_valid(True)
x = serializer.save() # 返回持久化的实例
print(type(x), x)
```

```python
emgr = Employee.objects
emp = emgr.get(pk=10022) # 查实例

data = {
    'emp_no': 10022, 'birth_date': '1963-06-01',
    'first_name': '三', 'last_name': '张',
    'gender': 1, 'hire_date': '1989-08-24',
} # 注意，这是字典数据是data，不是Employee的实例

# 有实例，有data，是更新
serializer = EmpSerializer(emp, data=data)
serializer.is_valid(True)
x = serializer.save() # 更新
print(type(x), x) # 返回实例
```

外键关系

员工和工资，是一对多关系

Model类

```python
from django.db import models

class Employee(models.Model):
    class Gender(models.IntegerChoices): # 枚举类型，限定取值范围
        MAN = 1, '男'
        FEMALE = 2, '女'
    class Meta:
        db_table = 'employees'
        verbose_name = '员工'
    emp_no = models.IntegerField(primary_key=True, verbose_name='工号')
    birth_date = models.DateField(verbose_name='生日')
    first_name = models.CharField(max_length=14, verbose_name='名')
    last_name = models.CharField(max_length=16, verbose_name='姓')
    gender = models.SmallIntegerField(verbose_name='性别',
choices=Gender.choices)
    hire_date = models.DateField()

class Salary(models.Model):
    class Meta:
        db_table = "salaries"
    #id = models.AutoField(primary_key=True) # 额外增加的主键，Django不支持联合主
键
    emp_no = models.ForeignKey(Employee, on_delete=models.CASCADE,
                               db_column='emp_no', related_name='salaries')
    from_date = models.DateField()
    salary = models.IntegerField(verbose_name='工资')
    to_date = models.DateField()
```

序列化器

```python
from rest_framework import serializers
from .models import Employee, Salary

class EmpSerializer(serializers.ModelSerializer):
    class Meta:
        model = Employee
        fields = '__all__'

class SalarySerializer(serializers.ModelSerializer):
    class Meta:
        model = Salary
        fields = '__all__'

print('~' * 30)
print(EmpSerializer())
print(SalarySerializer())
print('~' * 30)
```

序列化器如下

```
EmpSerializer():
```

```
    emp_no = IntegerField(label='工号', max_value=2147483647,
min_value=-2147483648, validators=
[<UniqueValidator(queryset=Employee.objects.all())>])
    birth_date = DateField(label='生日')
    first_name = CharField(label='名', max_length=14)
    last_name = CharField(label='姓', max_length=16)
    gender = ChoiceField(choices=[(1, '男'), (2, '女')], label='性别',
validators=[<django.core.validators.MinValueValidator object>,
<django.core.validators.MaxValueValidator object>])
    hire_date = DateField()


SalarySerializer():
    id = IntegerField(label='ID', read_only=True)
    from_date = DateField()
    salary = IntegerField(label='工资', max_value=2147483647,
min_value=-2147483648)
    to_date = DateField()
    emp_no = PrimaryKeyRelatedField(queryset=Employee.objects.all())
```

SalarySerializer有一个外键关联

序列化

1、各自独立查询

先查员工，再去查相关工资信息

```python
import os
import django

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'salary.settings')
django.setup(set_prefix=False)
# 所有测试代码，都要在上面4行之下

from employee.models import Employee
from employee.serializers import EmpSerializer, SalarySerializer

emgr = Employee.objects

emp = emgr.get(pk=10003)
print(EmpSerializer(emp).data)
print(SalarySerializer(emp.salaries.all(), many=True).data)
```

2、关联主键

参考 https://www.django-rest-framework.org/api-guide/relations/#primarykeyrelatedfield

直接获取所有**关联pk**，利用`serializers.PrimaryKeyRelatedField`来关联

```python
from rest_framework import serializers
from .models import Employee, Salary

class EmpSerializer(serializers.ModelSerializer):
    class Meta:
```

```
        model = Employee
        fields = '__all__'
    # salaries必须和Model类中的属性对应
    salaries = serializers.PrimaryKeyRelatedField(many=True, read_only=True)
    # salaries =
serializers.PrimaryKeyRelatedField(queryset=Salary.objects.all(), many=True)

class SalarySerializer(serializers.ModelSerializer):
    class Meta:
        model = Salary
        fields = '__all__'
```

```
from employee.models import Employee
from employee.serializers import EmpSerializer, SalarySerializer

emgr = Employee.objects

emp = emgr.get(pk=10003)
print(EmpSerializer(emp).data)
```

```
{'emp_no': 10003, 'salaries': [24, 25, 26, 27, 28, 29, 30], 'birth_date':
'1959-12-03', 'first_name': 'Parto', 'last_name': 'Bamford', 'gender': 1,
'hire_date': '1986-08-28'}
```

3、关联字符串表达

参考 https://www.django-rest-framework.org/api-guide/relations/#stringrelatedfield

直接获取所有关联对象的字符串表达，需要用到 `__str__`

```
from rest_framework import serializers
from .models import Employee, Salary

class EmpSerializer(serializers.ModelSerializer):
    class Meta:
        model = Employee
        fields = '__all__'
    # salaries必须和Model类中的属性对应
    salaries = serializers.StringRelatedField(many=True)

class SalarySerializer(serializers.ModelSerializer):
    class Meta:
        model = Salary
        fields = '__all__'
```

```
from employee.models import Employee
from employee.serializers import EmpSerializer, SalarySerializer

emgr = Employee.objects

emp = emgr.get(pk=10003)
print(EmpSerializer(emp).data) # 使用的是Salary模型类的__str__
```

```
{'emp_no': 10003, 'salaries': ['<S 24, 10003, 40006>', '<S 25, 10003,
43616>', '<S 26, 10003, 43466>', '<S 27, 10003, 43636>', '<S 28, 10003,
43478>', '<S 29, 10003, 43699>', '<S 30, 10003, 43311>'], 'birth_date':
'1959-12-03', 'first_name': 'Parto', 'last_name': 'Bamford', 'gender': 1,
'hire_date': '1986-08-28'}
```

4、关联对象

参考 https://www.django-rest-framework.org/api-guide/relations/#nested-relationships

如果需要更加完整的关联数据，可以使用关联对象的方式，要利用对方的序列化器

```python
from rest_framework import serializers
from .models import Employee, Salary

# 颠倒一下序列化器的定义次序
class SalarySerializer(serializers.ModelSerializer):
    class Meta:
        model = Salary
        fields = '__all__'

class EmpSerializer(serializers.ModelSerializer):
    class Meta:
        model = Employee
        fields = '__all__'
    # salaries必须和Model类中的属性对应
    salaries = SalarySerializer(many=True, read_only=True)
```

```python
from employee.models import Employee
from employee.serializers import EmpSerializer, SalarySerializer

emgr = Employee.objects
emp = emgr.get(pk=10003)

import json
print(json.dumps(EmpSerializer(emp).data))
```

返回的数据序列化成json格式如下

```
{
    "emp_no":10003,
    "salaries":[
        {
            "id":24,
            "from_date":"1995-12-03",
            "salary":40006,
            "to_date":"1996-12-02",
            "emp_no":10003
        },
        {
            "id":25,
            "from_date":"1996-12-02",
            "salary":43616,
            "to_date":"1997-12-02",
```

```
            "emp_no":10003
        },
        {
            "id":26,
            "from_date":"1997-12-02",
            "salary":43466,
            "to_date":"1998-12-02",
            "emp_no":10003
        },
        {
            "id":27,
            "from_date":"1998-12-02",
            "salary":43636,
            "to_date":"1999-12-02",
            "emp_no":10003
        },
        {
            "id":28,
            "from_date":"1999-12-02",
            "salary":43478,
            "to_date":"2000-12-01",
            "emp_no":10003
        },
        {
            "id":29,
            "from_date":"2000-12-01",
            "salary":43699,
            "to_date":"2001-12-01",
            "emp_no":10003
        },
        {
            "id":30,
            "from_date":"2001-12-01",
            "salary":43311,
            "to_date":"9999-01-01",
            "emp_no":10003
        }
    ],
    "birth_date":"1959-12-03",
    "first_name":"Parto",
    "last_name":"Bamford",
    "gender":1,
    "hire_date":"1986-08-28"
}
```