KAZAKH BRITISH TECHNICAL UNIVERSITY

BACKEND FRAMEWORK (DJANGO)

Lesson 7

DRF Serializers

rest_framework.Serializer

1. Serializing objects
2. create(self, validated_data)
3. update(self, instance, validated_data)
4. save()

Validation

1. serializer.**is_valid()** - while deserializing data —(True, False)
2. serializer**.errors**
3. serializer.is_valid(**raise_exception**=True)

Field level validation

1. validate_*<field_name>*(field_value)
2. return validated value
3. raise serializers.ValidationError if any

Object level validation

1. validate(data)
2. return validated value
3. raise serializers.ValidationError if any

Validators

```
def rating_range_validation(value):
  if not (1 <= value <= 5):
    raise serializers.ValidationError('Invalid rating value')
```

rating = IntegerField(**validators=[rating_range_validation]**)

ModelSerializer

The **ModelSerializer** class is the same as a regular **Serializer** class, except that:

1. It will automatically generate a set of fields for you, based on the model.
2. It will automatically generate validators for the serializer, such as unique_together validators.
3. It includes simple default implementations of **.create()** and **.update()**.

```
class TaskSerializer(serializers.ModelSerializer):
  class Meta:
    model = Task
    fields = (‘id’, ‘title’, ‘description’, ‘created_at’,)

    # fields = ‘__all__'
```

ModelSerializer

1. fields or exclude
2. read_only_fields

```
class TaskSerializer(serializers.ModelSerializer):
  class Meta:
    model = Task
    fields = (‘id’, ‘title’, ‘description’, ‘created_at’,)

    # fields = ‘__all__'
```

Nested Serializers

BaseSerializer

1. .data
2. .is_valid()
3. .validated_data
4. .errors
5. .save()
6. .create() / .update()
7. .to_representation() / .to_internal_value()

Serializer Inheritance

Questions?