Ask Company

리액트와 함께 장고 시작하기 / 장고 DRF

Serializer를 통한 유효성 검사 및 저장

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

Serializer의 생성자

Serializer는 Django Form와 컨셉/사용법이 유사하나, 생성자 차이

```python
# django/forms/forms.py
class BaseModelForm:
    def __init__(self, data=None, files=None, auto_id='id_%s', prefix=None,
                 initial=None, error_class=ErrorList, label_suffix=None,
                 empty_permitted=False, instance=None, use_required_attribute=None):

class Form(BaseForm):
    pass
```

```python
# rest_framework/serializers.py

class BaseSerializer(Field):
    def __init__(self, instance=None, data=empty, **kwargs):

class Serializer(BaseSerializer):
    pass
```

data= 인자가 주어지면

.is_valid()가 호출되고 나서야 ...

1. .initial_data 필드에 접근할 수 있고
2. .validated_data를 통해 유효성 검증에 통과한 값들이 .save() 시에 사용됩니다.
3. .errors → 유효성 검증 수행 후에 오류 내역
4. .data → 유효성 검증 후에, 갱신된 인스턴스에 대한 필드값 사전

serializer.save(**kwargs) 호출을 할 때

1. DB에 저장한 관련 instance를 리턴
2. .validated_data와 kwargs사전을 합친 데이터를
    1. .update 함수 / .create 함수를 통해 관련 필드에 값을 할당하고, DB로의 저장을 시도
    2. .update() : self.instance 인자를 지정했을 때
    3. .create() : self.instance 인자를 지정하지 않았을 때

from rest_framework.validators import ...

값에 대한 유효성 검사를 수행하는 호출 가능한 객체

DRF에서는 유일성 체크를 도와주는 Validators 제공

1. UniqueValidator : 지정 1개 필드가 지정 QuerySet 범위에서의 유일성 여부 체크
2. UniqueTogetherValidator : UniqueValidator의 다수 필드 버전
3. BaseUniqueForValidator
4. UniqueForDateValidator (BaseUniqueValidator) : 지정 날짜 범위에서 유일성 여부 체크
5. UniqueForMonthValidator (BaseUniqueValidator) : 지정 월 범위에서 유일성 여부 체크
6. UniqueForYearValidator (BaseUniqueValidator) : 지정 년 범위에서 유일성 여부 체크

https://github.com/encode/django-rest-framework/blob/master/rest_framework/validators.py

UniqueValidator

모델 필드에 unique=True를 지정하면, 자동 지정

- qureyset (필수)
- message : 유효성 검사 실패 시의 에러 메세지
- lookup : 디폴트 'exact'

Serializer 필드에 직접 validators 지정 가능

```
from rest_framework.validators import UniqueValidator

slug = SlugField(
    max_length=100,
    validators=[UniqueValidator(queryset=BlogPost.objects.all())]
)
```

UniqueTogetherValidator

모델 Meta에 unique_together이 지정된다면, 자동 지정

- queryset (필수)
- fields (필수)
- message

```python
from rest_framework.validators import UniqueTogetherValidator

class ExampleSerializer(serializers.Serializer):
    # ...
    class Meta:
        validators = [
            UniqueTogetherValidator(
                queryset=ToDoItem.objects.all(),
                fields=['list', 'position']
            )
        ]
```

UniqueForDateValidator / Month / Year

- queryset (필수)
- field (필수)
- date_field (필수)
- message

```python
from rest_framework.validators import UniqueForYearValidator

class ExampleSerializer(serializers.Serializer):
    # ...
    class Meta:
        validators = [
            UniqueForYearValidator(
                queryset=BlogPostItem.objects.all(),
                field='slug',
                date_field='published'
            )
        ]
```

유효성 검사에 실패하면 ValidationError 예외 발생

필히 rest_framework.exceptions.ValidationError 사용

장고 기본에서는 django.forms.exceptions.ValidationError

```
class ValidationError(APIException):
    status_code = status.HTTP_400_BAD_REQUEST
    default_detail = _('Invalid input.')
    default_code = 'invalid'

    def __init__(self, detail=None, code=None):
        if detail is None:
            detail = self.default_detail
        if code is None:
            code = self.default_code

        if not isinstance(detail, dict) and not isinstance(detail, list):
            detail = [detail]

        self.detail = _get_error_details(detail, code)
```

https://github.com/encode/django-rest-framework/blob/3.10.1/rest_framework/exceptions.py#L138

Serializer에의 유효성 검사

필드 정의 시에 validators 지정하거나, 클래스 Meta.validators 지정

Field Level 검사 : 유효성 검사 및 값 변환

```
class PostSerializer(serializers.Serializer):
    title = serializers.CharField(max_length=100)

    def validate_title(self, value):
        if 'django' not in value:
            raise ValidationError('제목에 필히 django가 포함되어야합니다.')
        return value
```

Object Level 검사 : 유효성 검사 및 값 변환

```
class PostSerializer(serializers.Serializer):
    title = serializers.CharField(max_length=100)

    def validate(self, data):
        if 'django' not in data['title']:
            raise ValidationError('제목에 필히 django가 포함되어야합니다.')
        return data
```

DB로의 반영과 Mixins의 perform_ 계열 함수

APIView의 create/update/destroy 멤버함수에서 실질적인 DB처리 로직은

perform_create(serializer)/perform_update(serializer)/perform_destroy(instance)를 통해 이뤄집니다.

```
class CreateModelMixin:
    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()

    def get_success_headers(self, data):
        try:
            return {'Location': str(data[api_settings.URL_FIELD_NAME])}
        except (TypeError, KeyError):
            return {}
```

create 시에 추가로 저장할 필드가 있다면?

모델에 ip 필드가 있고, 유저의 아이피들 저장하고 싶다면?

```
def perform_create(self, serializer):
    serializer.save(ip=self.request.META['REMOTE_ADDR'])
```

인생은 짧습니다.
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

-

Ask Company