# Djangoで、CreateViewでPOST後に動作する、ModelFormやModelのバリデーションを試した

Django Python

Djangoで、CreateView + ModelForm + Modelを使った時に、

- バリデーションを書く場所
- バリデーションの実行順序

が気になりました。

後述の通り日本語の分かりやすい記事があったものの、実際に手を動かさないと理解できない部分もあったため、その時のメモを残します。

目次

- ソースコード

## 環境

- Windows10
- Python 3.5.1
- Django 1.9.2

## バリデーションを書く場所について

バリデーションの書き方は以下が参考になりました。
Validators - Writing validators | Django documentation | Django

また、ModelFormやModelのバリデーションを書く場所については、以下が参考になりました。
[django]フォームのバリデーションまとめ(to_python, clean, validate) - dackdive's blog

そこで今回は、上記に加えてFormのvalidatorsやModelについても試すことにしました。

### ModelForm

公式ドキュメントでは、以下に記載がありました。

- Creating forms from models - Validation on a ModelForm| Django documentation | Django
- Form and field validation | Django documentation | Django

#### 各フィールドごとのバリデーション

`django.forms` で用意されているFieldをそのまま使う場合は、

- `Form.clean_<field_name>()`
- ModelFormのフィールドでvalidatorsを指定 (例： `CharField(validators=[my_validator])` )

が使えます。

一方、 `django.forms` のFieldを拡張して使う場合は、

- `Form.clean_<field_name>()`
- ModelFormのフィールドでvalidatorsを指定 (例： `CharField(validators=[my_validator])` )

- `Field.to_python()`
- `Field.validate()`
- `Field.clean()`

が使えます。

#### 複数フィールドを組み合わせるバリデーション

- `Form.clean()`

### Model

ModelFormと同じようなメソッドが用意されています。
Model instance reference - Validating objects | Django documentation | Django

#### 各フィールドごとのバリデーション

`django.db.models` で用意されているFieldをそのまま使う場合は、

- Modelのフィールドでvalidatorsを指定 (例： `CharField(validators=[my_validator])` )

が使えます。

一方、 `django.db.models` のFieldを拡張して使う場合は、

- Modelのフィールドでvalidatorsを指定 (例： `CharField(validators=[my_validator])` )
- `Field.to_python()`
- `Field.validate()`
- `Field.clean()`

が使えます。

#### 複数フィールドを組み合わせるバリデーション

- `Model.clean()`

## バリデーションの呼び出し順について

呼び出し順については以下が参考になりました。ありがとうございました。

- DjangoのFormのcleanメソッドの呼び出し順 - 雑記
- Django の Form のバリデーションについて整理する - はてダ

今回はModelFormとModelを使った時の呼び出し順を知りたかったため、ModelFormやModelなどを拡張してprintデバッグを埋め込んでみました。

## 実装

### myapp/utils.py

printデバッグの時にインデントを付ける関数を用意しました。

```
INDENT = ' '*2

def print_with_indent(message, repeat):
    print('{nest}{message}'.format(nest=INDENT*repeat, message=message))
```

### ModelForm

```
from django import forms
from .models import Article
from .utils import print_with_indent

def dummy_validator_for_custom_field(value):
    '''ArticleFormのform_onlyフィールドで指定するvalidators用'''
    print_with_indent('Form.custom_field.validator', 5)

class CustomFormField(forms.CharField):
    '''ArticleFormのform_onlyフィールド用'''
    def to_python(self, value):
        print_with_indent('Form.custom_field.to_python()', 5)

        if len(value) == 2:
            # 挙動を確認するためのバリデーション
            raise forms.ValidationError('CustomFormField.to_pythonで長さ2文字はダメ')
        return super(CustomFormField, self).to_python(value)

    def validate(self, value):
        print_with_indent('Form.custom_field.validate()', 5)
        return super(CustomFormField, self).validate(value)

    def clean(self, value):
```

```
        print_with_indent('Form.custom_field.clean()', 4)
        return super(CustomFormField, self).clean(value)


class ArticleForm(forms.ModelForm):
    form_only = CustomFormField(required=False,
                                validators=[dummy_validator_for_custom_field])

    def is_valid(self):
        print_with_indent('Form.is_valid()', 0)
        return super(ArticleForm, self).is_valid()

    def full_clean(self):
        print_with_indent('Form.full_clean()', 2)
        return super(ArticleForm, self).full_clean()

    def _clean_fields(self):
        print_with_indent('Form._clean_fields()', 3)
        return super(ArticleForm, self)._clean_fields()

    def _clean_form(self):
        print_with_indent('Form._clean_form()', 3)
        return super(ArticleForm, self)._clean_form()

    def _post_clean(self):
        print_with_indent('Form._post_clean()', 3)
        return super(ArticleForm, self)._post_clean()

    def clean(self):
        print_with_indent('Form.clean()', 4)

        cleaned_data = super(ArticleForm, self).clean()
        if cleaned_data.get('lines') == 10:
            raise forms.ValidationError('Form.clean()でlinesが10だとダメ')
        return cleaned_data

    def clean_title(self):
        print_with_indent('Form.clean_title()', 4)
        title = self.cleaned_data['title']
        if len(title) == 1:
            raise forms.ValidationError('clean_titleで長さ1文字はダメ')
        return title

    def clean_lines(self):
        print_with_indent('Form.clean_lines()', 4)
        return self.cleaned_data['lines']

    def clean_form_only(self):
        print_with_indent('Form.clean_form_only()', 4)
```

```
        form_only = self.cleaned_data['form_only']
        if len(form_only) == 3:
            # 挙動を確認するためのバリデーション
            raise forms.ValidationError('clean_form_onlyで長さ3文字はダメ')
        return form_only

    @property
    def errors(self):
        print_with_indent('Form.errors', 1)
        return super(ArticleForm, self).errors

    def save(self, commit=True):
        print_with_indent('Form.save()', 0)
        return super(ArticleForm, self).save()

    class Meta:
        model = Article
        fields = '__all__'
```

### Model

```
from django.db import models
from django.core.exceptions import ValidationError
from .utils import print_with_indent

def dummy_validator_for_custom_field(value):
    '''Articleのtitleフィールドで指定するvalidators用'''
    print_with_indent('Model.custom_field.validator', 7)

def dummy_validator_for_lines(value):
    '''Articleのlineフィールドで指定するvalidators用'''
    print_with_indent('Model.lines.validator', 6)

class CustomModelField(models.CharField):
    '''Articleのtitleフィールド用'''
    def to_python(self, value):
        print_with_indent('Model.custom_field.to_python()', 7)

        if len(value) == 4:
            # 挙動を確認するためのバリデーション
            raise ValidationError('CustomModelField.to_pythonで長さ4文字はダメ')
        return super(CustomModelField, self).to_python(value)

    def validate(self, value, model_instance):
        print_with_indent('Model.custom_field.validate()', 7)
        return super(CustomModelField, self).validate(value, model_instance)
```

```
    def clean(self, value, model_instance):
        print_with_indent('Model.custom_field.clean()', 6)
        return super(CustomModelField, self).clean(value, model_instance)


class Article(models.Model):
    title = CustomModelField('Title', max_length=255,
                             validators=[dummy_validator_for_custom_field])
    lines = models.DecimalField('Lines', max_digits=10, decimal_places=0,
                                validators=[dummy_validator_for_lines])

    def full_clean(self, exclude=None, validate_unique=True):
        print_with_indent('Model.full_clean()', 4)
        return super(Article, self).full_clean(exclude, validate_unique)

    def clean_fields(self, exclude=None):
        print_with_indent('Model.clean_fields()', 5)
        return super(Article, self).clean_fields(exclude)

    def clean(self):
        print_with_indent('Model.clean()', 5)
        return super(Article, self).clean()

    def validate_unique(self, exclude=None):
        print_with_indent('Model.validate_unique()', 5)
        return super(Article, self).validate_unique(exclude)

    def save(self, force_insert=False, force_update=False, using=None, update_fields=None):
        print_with_indent('Model.save()', 2)
        return super(Article, self).save(force_insert, force_update, using, update_fields)
```

## バリデーションエラーなしの場合の動作確認

特にエラーがない場合、以下のような順番で呼ばれました。

```
Form.is_valid()
  Form.errors
    Form.full_clean()
      Form._clean_fields()
        Form.clean_title()
        Form.clean_lines()
        Form.custom_field.clean()
          Form.custom_field.to_python()
          Form.custom_field.validate()
          Form.custom_field.validator
```

```
        Form.clean_form_only()
      Form._clean_form()
        Form.clean()
      Form._post_clean()
        Model.full_clean()
          Model.clean_fields()
            Model.custom_field.clean()
              Model.custom_field.to_python()
              Model.custom_field.validate()
              Model.custom_field.validator
            Model.lines.validator
          Model.clean()
          Model.validate_unique()
Form.save()
  Form.errors
    Model.save()
              Model.custom_field.to_python()
```

## バリデーションエラーありの場合の動作確認

バリデーションエラーとなった場合、特に

- 最初のForm.clean_<filed_name>でエラー
- Form.CustomFormFieldのうち、最初のバリデーションでエラー
- Formにしかないフィールドのバリデーションでエラー
- Form全体のバリデーション(Form.clean())でエラー
- Model.CustomFormFieldのうち、最初のバリデーションでエラー

の場合に、どの処理が実行されるか気になったため、以下の通り試してみました。

### 最初のForm.clean_<filed_name>でエラー

エラーが発生したところで `Form.errors` が呼ばれますが、それ以外は最後までバリデーションが呼ばれています。

```
Form.is_valid()
  Form.errors
    Form.full_clean()
      Form._clean_fields()
        Form.clean_title()
  Form.errors
        Form.clean_lines()
        Form.custom_field.clean()
```

```
          Form.custom_field.to_python()
          Form.custom_field.validate()
          Form.custom_field.validator
        Form.clean_form_only()
      Form._clean_form()
        Form.clean()
      Form._post_clean()
        Model.full_clean()
          Model.clean_fields()
            Model.lines.validator
          Model.clean()
          Model.validate_unique()
```

### Form.CustomFormFieldのうち、最初のバリデーションでエラー

今までと異なり、

- Form.custom_field.validate()
- Form.custom_field.validator

が呼ばれませんでした。

```
Form.is_valid()
  Form.errors
    Form.full_clean()
      Form._clean_fields()
        Form.clean_title()
        Form.clean_lines()
        Form.custom_field.clean()
          Form.custom_field.to_python()
  Form.errors
      Form._clean_form()
        Form.clean()
      Form._post_clean()
        Model.full_clean()
          Model.clean_fields()
            Model.custom_field.clean()
              Model.custom_field.to_python()
              Model.custom_field.validate()
              Model.custom_field.validator
            Model.lines.validator
          Model.clean()
          Model.validate_unique()
```

### Formにしかないフィールドのバリデーションでエラー

エラーが発生したところで `Form.errors` が呼ばれますが、それ以外は最後までバリデーションが呼ばれています。

```
Form.is_valid()
  Form.errors
    Form.full_clean()
      Form._clean_fields()
        Form.clean_title()
        Form.clean_lines()
        Form.custom_field.clean()
          Form.custom_field.to_python()
          Form.custom_field.validate()
          Form.custom_field.validator
        Form.clean_form_only()
  Form.errors
      Form._clean_form()
        Form.clean()
      Form._post_clean()
        Model.full_clean()
          Model.clean_fields()
            Model.custom_field.clean()
              Model.custom_field.to_python()
              Model.custom_field.validate()
              Model.custom_field.validator
            Model.lines.validator
          Model.clean()
          Model.validate_unique()
```

### Form全体のバリデーション(Form.clean())でエラー

エラーが発生したところで `Form.errors` が呼ばれますが、それ以外は最後までバリデーションが呼ばれています。

```
Form.is_valid()
  Form.errors
    Form.full_clean()
      Form._clean_fields()
        Form.clean_title()
        Form.clean_lines()
        Form.custom_field.clean()
          Form.custom_field.to_python()
          Form.custom_field.validate()
          Form.custom_field.validator
```

```
        Form.clean_form_only()
      Form._clean_form()
        Form.clean()
  Form.errors
      Form._post_clean()
        Model.full_clean()
          Model.clean_fields()
            Model.custom_field.clean()
              Model.custom_field.to_python()
              Model.custom_field.validate()
              Model.custom_field.validator
            Model.lines.validator
          Model.clean()
          Model.validate_unique()
```

### Model.CustomFormFieldのうち、最初のバリデーションでエラー

Formと同様、

- Model.custom_field.validate()
- Model.custom_field.validator

が呼ばれませんでした。

```
Form.is_valid()
  Form.errors
    Form.full_clean()
      Form._clean_fields()
        Form.clean_title()
        Form.clean_lines()
        Form.custom_field.clean()
          Form.custom_field.to_python()
          Form.custom_field.validate()
          Form.custom_field.validator
        Form.clean_form_only()
      Form._clean_form()
        Form.clean()
      Form._post_clean()
        Model.full_clean()
          Model.clean_fields()
            Model.custom_field.clean()
              Model.custom_field.to_python()
            Model.lines.validator
          Model.clean()
  Form.errors
          Model.validate_unique()
```

### Djangoのソースコード確認

Djangoのソースコードでも呼び出し順を確認してみます。

| No | 場所 | メソッド | ファイル |
|---|---|---|---|
| 1 | CreateView | post() | django.views.generic.edit.py |
| 2 | Form | is_valid() | django.forms.forms.py |
| 3 | Form | errors | django.forms.form.py |
| 4 | Form | full_clean() | django.forms.form.py |
| 5 | Form | _clean_fields() | django.forms.forms.py |
| 6 | FormField | clean() | django.forms.fields.py |
| 7 | Form | _clean_form() | django.forms.forms.py |
| 8 | Form | clean() | django.forms.forms.py |
| 9 | Form | _post_clean() | django.forms.forms.py |
| 10 | ModelForm | _post_clean() | django.forms.models.py |
| 11 | Model | full_clean() | django.db.models.base.py |
| 12 | Model | clean_fields() | django.db.models.base.py |
| 13 | ModelField | clean() | django.db.models.fields.__init__.py |
| 14 | Model | clean() | django.db.models.base.py |
| 15 | CreateView | form_valid() | django.views.generic.edit.py |
| 16 | ModelForm | save() | django.forms.models.py |

## 動作確認に使用したテストコードについて

バリデーションエラーありの場合の動作確認について、ブラウザによる手動テストは手間でした。

そこで、テストコードで動作確認をしたため、その時のメモも残しておきます。

### テストコードからPOSTリクエストを出す方法について

Seleniumを使うのは大げさすぎるので、テストコードからDjangoのFormにPOSTする方法を調べたところ、`django.test.RequestFactory`を使うのが良さそうでした。

- python - How TDD can be applied to Django Class based Generic Views? - Stack Overflow
- Advanced testing topics - The request factory| Django documentation | Django

## パラメタライズドテストの書き方について

今回は省力化のために、パラメタライズドテスト(Parameterized Test)にてテストコードを書くことにしました。

Djangoで良い方法がないかを調べたところ、`py.test`を使う方法が紹介されていました。
Django Best Practice への道 #2

良さそうでしたが、`py.test`を追加で入れるのが気になりました。そのため、Djangoだけ使える方法がないかを探したところ、Python3.4以降で使えるようになった`subTest()`がありました。

- How to generate dynamic (parametrized) unit tests in python? - Stack Overflow
- 26.4.7. Distinguishing test iterations using subtests - 26.4. unittest — Unit testing framework — Python 3.5.1 documentation
- Python 3.4. subTest example - encukou/subTest

これが良さそうだったので、以下のようにテストコードを書きました。

**myapp/test.py**

```
from django.core.urlresolvers import reverse
from django.test import TestCase, RequestFactory
from . import views

class SimpleTest(TestCase):
    def setUp(self):
        self.factory = RequestFactory()

        self.tests = {
            '正常' : ['title', 1, 'form_only'],
            '最初のForm.clean_<filed_name>でエラー': ['t', 1, 'form_only'],
            'Form.CustomFormFieldのうち、最初のバリデーションでエラー': ['title', 1, 'fo'],
            'Formにしかないフィールドのバリデーションでエラー': ['title', 1, 'for'],
            'Form全体のバリデーション(Form.clean())でエラー': ['title', 10, 'form_only'],
            'Model.CustomFormFieldのうち、最初のバリデーションでエラー': ['titl', 1, 'form_only'],
        }

    def test_use_subtest(self):
        for test_name, (title, lines, form_only) in self.tests.items():
            with self.subTest(name=test_name):
                print('------------------\n{}\n------------------'.format(test_name))
                request = self.factory.post(reverse('my:article-create'))
                request.POST['title'] = title
                request.POST['lines'] = lines
                request.POST['form_only'] = form_only
```

```
        response = views.ArticleCreateView.as_view()(request)

        # 成功させないとログが出ないので、必ず成功するようにしている
        self.assertEqual(1,1)
```

あとは、`python manage.py test` でテストコードを実行して確認しました。