Ask Company

리액트와 함께 장고 시작하기 / 장고 Forms

Form

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

Form

장고를 더욱 장고스럽게 만들어주는 주옥같은 Feature

주요 역할

입력폼 HTML 생성

입력폼 값에 대한 유효성 검증 (Validation) 및 값 변환

검증을 통과한 값들을 dict형태로 제공

```
# myapp/forms.py
from django import forms

class PostForm(forms.Form):
    title = forms.CharField()
    content = forms.CharField(widget=form.Textarea)
```

Django Style의 Form 처리 (1)

하나의 URL (하나의 View)에서 2가지 역할을 모두 수행

1. 빈 폼을 보여주는 역할과
2. 폼을 통해 입력된 값을 검증하고 저장하는 역할

Django Style의 Form 처리 (2)

GET 방식으로 요청받았을 때

New/Edit 입력폼을 보여줍니다.

POST 방식으로 요청받았을 때

데이터를 입력받아 (request.POST, request.FILES) 유효성 검증 수행

검증 성공 시 : 해당 데이터를 저장하고 SUCCESS URL로 이동

검증 실패 시 : 오류메세지와 함께 입력폼을 다시 보여줍니다.

```
def post_new(request):
    if request.method == 'POST':
        form = PostForm(request.POST, request.FILES)
        if form.is_valid():
            post = Post(**self.cleaned_data)
            post.save()
            return redirect(post)
    else:
        form = PostForm()

    return render(request, 'blog/post_form.html', {
        'form': form,
    })
```

#1) Form/ModelForm 클래스 정의

```
# myapp/forms.py
from django import forms

class PostForm(forms.Form):
    title = forms.CharField()
    content = forms.CharField(widget=form.Textarea)
```

#2) 필드 별로 유효성 검사 함수 추가 적용

Form의 경우

```
# myapp/forms.py
from django import forms

def min_length_3_validator(value):
    if len(value) < 3:
        raise forms.ValidationError('3글자 이상 입력해주세요.')

class PostForm(forms.Form):
    title = forms.CharField(validators=[min_length_3_validator])
    content = forms.CharField(widget=form.Textarea)
```

```
from django.core.validators import MinLengthValidator

min_length_3_validator = MinLengthValidator(3)
```

#2) 필드 별로 유효성 검사 함수 추가 적용

ModelForm의 경우

```
# myapp/models.py
from django import forms
from django import models

def min_length_3_validator(value):
    if len(value) < 3:
        raise forms.ValidationError('3글자 이상 입력해주세요.')

class Post(forms.Model):
    title = models.CharField(max_length=100, validators=[min_length_3_validator])
    content = models.TextField()

# myapp/forms.py
from django import forms
from .models import Post

class PostForm(forms.ModelForm):
    class Meta:
        model = Post
        fields = '__all__'
```

#3) View 함수 내에서 method에 따라 Form 객체 생성

if 조건체크를 POST에 대해 먼저 체크하는 것은 장고 스타일. GET요청은 Form 인스턴스 생성 이외에 특별한 루틴이 없어서인듯.

```
# myapp/views.py
from .forms import PostForm

if request.method == 'POST':
    # POST 요청일 때
    form = PostForm(request.POST, request.FILES)
else:
    # GET 요청일 때
    form = PostForm()
```

#4) POST 요청에 한해 입력값 유효성 검증

```
if request.method == 'POST':
    # POST인자는 request.POST와 request.FILES를 제공받음.
    form = PostForm(request.POST, request.FILES)

    # 인자로 받은 값에 대해서, 유효성 검증 수행
    if form.is_valid(): # 검증이 성공하면, True 리턴
        # 검증에 성공한 값들을 사전타입으로 제공받음.
        # 검증에 성공한 값을 제공받으면, Django Form의 역할은 여기까지 !!!
        # 필요에 따라, 이 값을 DB에 저장하기
        form.cleaned_data

        post = Post(**form.cleaned_data) # DB에 저장하기
        post.save()

        return redirect('/success_url/')
    else: # 검증에 실패하면, form.errors와 form.각필드.errors 에 오류정보를 저장
        form.errors
else: # GET 요청일 때
    form = PostForm()
    return render(request, 'myapp/form.html', {'form': form})
```

#5) 템플릿을 통해 HTML 폼 노출

1. GET요청일 때

   → 유저가 Form을 채우고 Submit → POST 요청

2. POST요청이지만 유효성 검증에서 실패했을 때

   Form 인스턴스를 통해 HTML폼 출력

   오류메세지도 있다면 같이 출력

   → 유저가 Form을 채우고 Submit → POST 재요청

```
<table>
    <form action="" method="post">
        {% csrf_token %}
        <table>{{ form.as_table }}</table>
        <input type="submit" />
    </form>
</table>
```

전체 코드

myapp/models.py

```
from django.db import models
from django.urls import reverse

class Post(models.Model):
    title = models.CharField(max_length=100)
    content = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def get_absolute_url(self):
        return reverse('myapp:post_detail', args=[self.pk])
```

myapp/forms.py

```python
from django import forms

class PostForm(forms.Form):
    title = forms.CharField()
    content = forms.CharField(widget=form.Textarea)

    # ModelForm.save 인터페이스를 흉내내어 구현
    def save(self, commit=True):
        post = Post(**form.cleaned_data)
        if commit:
            post.save()
        return post
```

```python
from django import forms
from .models import Post

class PostForm(forms.ModelForm):
    class Meta:
        model = Post
        fields = '__all__'
```

myapp/views.py

```
from django.shortcuts import render
from .forms import PostForm
from .models import Post

def post_new(request):
    if request.method == 'POST':
        form = PostForm(request.POST, request.FILES)
        if form.is_valid():
            post = form.save()
            return redirect(post)
    else:
        form = PostForm()
    return render(request, 'myapp/post_form.html', {
        'form': form,
    })
```

myapp/templates/myapp/post_form.html

```
<table>
    <form action="" method="post">
        {% csrf_token %}
        <table>{{ form.as_table }}</table>
        <input type="submit" />
    </form>
</table>
```

Form Fields

Model Fields 와 유사

Model Fields : Database Field 들을 파이썬 클래스화

Form Fields : HTML Form Field 들을 파이썬 클래스화

필드 종류

BooleanField, CharField, ChoiceField, DateField, DateTimeField, EmailField, FileField, ImageField, FloatField, IntegerField, RegexField 등

https://docs.djangoproject.com/en/3.0/ref/forms/fields/

Life is short.
You need Python and Django.

I will be your pacemaker.