Django Form

구분	Django

- 우리는 지금까지 HTML form, input 태그를 통해 사용자로부터 데이터를 받았음
- 현재 우리 Django 서버는 들어오는 요청을 모두 수용하고 있는데, 이러한 요청 중에는

  비정상적인 혹은 악의적인 요청이 있다는 것을 생각해야 함
- 이처럼 사용자가 입력한 데이터가 우리가 원하는 데이터 형식이 맞는지에 대한 유효성 검증 반드시 필요
  - 이러한 유효성 검증은 많은 부가적인 것들을 고려해서 구현해야 하는데, 이는 개발 생산성을 늦출뿐더러 쉽지 않은 작업임
- Django Form은 이 과정에서 과중한 작업과 반복 코드를 줄여줌으로써 훨씬 쉽게 유효성 검증을 진행할 수 있도록 만들어 줌

Form에 대한 Django의 역할

- Form은 Django의 유효성 검사 도구 중 하나로 외부의 악의적 공격 및 데이터 손상에 대한 중요한 방어 수단
- Django는 Form과 관련한 유효성 검사를 단순화하고 자동화할 수 있는 기능을 제공하여, 개발자가 직접 작성하는 코드보다 더 안전하고 빠르게 수행하는 코드를 작성할 수 있다.
  - 개발자가 필요한 핵심 부분만 집중할 수 있도록 돕는 프레임워크의 특성

Django Form Class

Django는 Form에 관련된 작업의 세 부분을 처리

1. 렌더링을 위한 데이터 준비 및 재구성
2. 데이터에 대한 HTML forms 생성
3. 클라이언트로부터 받은 데이터 수신 및 처리

From Class 선언

- Form Class를 선언하는 것은 Model Class를 선언하는 것과 비슷하다.
    - 비슷한 이름의 필드 타입을 많이 가지고 있다. (이름만 같을 뿐 같은 필드는 아님!!)
- Model과 마찬가지로 상속을 통해 선언
    - forms 라이브러리의 Form 클래스를 상속 받음
- 앱 폴더에 `forsm.py`를 생성 후 ArticleForm Class 선언
- form에는 model field와 달리 TextField가 존재하지 않음
    - 모델의 TextField처럼 사용하려면 widgets 사용

```
# articles/forms.py

from django import forms

class ArticleForm(forms.Form):
     title = forms.CharField(max_length=10)
     content = forms.CharField(widget=forms.Textarea)
```

view 업데이트

```
from .forms import ArticleForm

def create(request):
    if request.method == 'POST':
        title = request.POST.get('title')
        content = request.POST.get('content')
        article = Article(title=title, content=content)
        article.save()
        return redirect('articles:detail', article.pk)
    else:
        form = ArticleForm()
        context = {'form':form }
        return render(request, 'articles/create.html', context)
```

create 템플릿 업데이트

```
{% extends 'base.html' %}

{% block content %}
  <h1>글작성</h1>
  <hr>
```

```
  <form action="{% url 'articles:create' %}" method="POST">
    {% csrf_token %}
    {{ form.as_p }}
    <input type="submit">
  </form>
{% endblock content %}
```

From rendering options

- <label> & <input> 쌍에 대한 3가지 출력 옵션

1. `as_p()`
   - 각 필드가 단락(<p>태그)으로 감싸져서 렌더링
2. `as_ul()`
   - 각 필드가 목록 항목(<ul>태그)으로 감싸져서 렌더링
   - 태그는 직접 작성해야 함
3. `as_table()`
   - 각 필드가 테이블(<tr> 태그)행으로 감싸져서 렌더링

Django의 2가지 HTML input 요소 표현

Form fields

- 입력에 대한 유효성 검사 로직을 처리
- 템플릿에서 직접 사용됨
- 예 : `forms.CharField()`

Widgets

- 웹페이지의 HTML input 요소 렌더링을 담당
  - 단순히 input 요소의 보여지는 부분을 변경하는 것으로 유효성 검증과는 관계 없음
- Widgets은 반드시 form fields에 할당됨
- 예 : `forms.CharField(widget=forms.Textarea)`
- 다양한 built-in 위젯 확인하기
  - https://docs.djangoproject.com/ko/3.2/ref/forms/widgets/#built-in-widgets