쟝고 자습 0830

날짜	@2022년 8월 30일
과정	django

- 가상환경 세팅

```
$ python -m venv [가상환경이름]
$ source [가상환경이름]/Scripts/activate
```

- 가상환경에 장고 설치

```
$ pip install django==3.2.13
```

- 패키지 목록 생성

```
$ pip freeze > requirements.txt
```

- 프로젝트 생성

```
$ django-admin startproject firstpjt .
```

- 서버 실행

```
$ python manage.py runserver
```

  - 서버를 끄고 수정을 해야 안전한듯

- 애플리케이션 생성

```
$ python manage.py startapp articles
```

  - 생성후에는 프로젝트폴더안 setting 내용 중 INSTALLED_APPS = 안에
    - ‘생성 앱 이름’, 적어줌 반드시 콤마까지

> 💡 반드시 앱 “생성 후 등록”

- 생성후 프로젝트폴더안 urls 내용 맨위에 from 앱이름 import views 해줘야함
  - 그러고 나서 urlpatterns에 path 적어줌 ex)

```
path('index/', views.index),
```

- 앱의 views에서 path에 쓴 함수를 선언하여 request를 받게한다.

```
def index(request):
    return render(request, 'index.html')
```

- render(request, template_name, context)
  - request : 응답을 생성하는 데 사용되는 요청 객체
  - template_name : 템플릿의 전체 이름 또는 템플릿 이름의 경로
  - context : 템플릿에서 사용할 데이터 (딕셔너리 타입)

개발 흐름

> URL → VIEW → TEMPLATE 순으로 작성한다. (Django Framework 의 작동 순서)

- 추가설정

```
# settings.py

LANGUAGE_CODE = 'ko-kr'
TIME_ZONE = 'Asia/Seoul'
```

- Templates 생성(실제 내용을 보여주는데 사용되는 파일)
  - app폴더 안에다가 templates 폴더를 만든다.
  - 그 templates 폴더 안에다가 view에서 함수에 쓴 html 파일을 만드는 것이다.

DTL 쟝고 템플릿 랭귀지(HTML안에 사용)

1. Variable

```
{{ variable }}
```

   - render의 세번째 인자로서 딕셔너리 형태로 넘겨준다. 여기서 key가 template에서 사용가능한 변수명이 된다.

2. Filters

```
{{ variable|filter }}
```

   - 표시할 변수를 수정할 때 사용

3. Tags

```
{% tag %}
```

   - 출력 텍스트를 만들거나, 반복 또는 논리를 수행하여 제어흐름을 만드는 등 변수보다 복잡한 일 수행
   - 일부 태그는 시작과 종료 태그가 필요

4. Comments

```
{#  #}
```

   - 쟝고 템플릿에서 라인의 주석을 표현
   - 한줄 주석에만 가능
   - 여러 줄 주석은

```
{% comment %}
  여러 줄 주석
{% endcomment %}
```

Variable

- Context 데이터가 1개

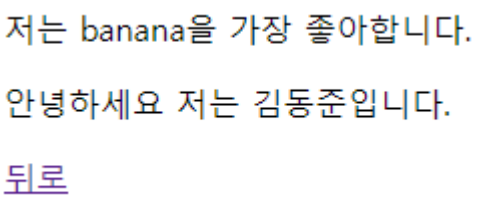
```
def greeting(request):
    return render(request, 'greeting.html', {'name':'김동준'})
```

- Context 데이터가 여러개

```
def greeting(request):
    foods = ['apple', 'banana', 'coconut',]
    info = {
        'name': '김동준'
    }
    context = {
        'foods': foods,
        'info': info,
    }
    return render(request, 'greeting.html', context)
```

```
<body>
  <p>저는 {{foods.1}}을 가장 좋아합니다.</p>
  <p>안녕하세요 저는 {{info.name}}입니다.</p>

  <a href="/index/">뒤로</a>
</body>
```



Filters

```
# app, views.py
import random

def dinners(request):
    foods = ['족발', '햄버거', '치킨', '초밥',]
    pick = random.choice(foods)
    context = {
        'pick': pick,
        'foods': foods,
```

```
    }
    return render(request, 'dinner.html', context)
```

```
<body>
  <p>{{ pick }} -> {{pick|length}}글자</p>
  <p>{{ foods|join:", "}}</p>

  <p>메뉴판</p>
  <ul>
    {% for food in foods %}
      <li>{{food}}</li>
    {% endfor %}
  </ul>

  <a href="/index/">뒤로</a>
</body>
```

초밥 -> 2글자

족발, 햄버거, 치킨, 초밥

메뉴판

- 족발
- 햄버거
- 치킨
- 초밥

뒤로

템플릿 상속

- {% extends '부모html' %} 태그 이용 (반드시 템플릿 최상단)
- {% block content %}{% endblock content %}
  - 하위 템플릿에서 재지정 할 수 있는 블록을 정의
  - 하위 템플릿이 채울 수 있는 공간
- base라는 이름의 스켈레톤 템플릿을 생성 (앱/템플릿/base.html)

```
{% extends 'base.html' %}
<!DOCTYPE html>
```

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  {% block content %}
    <h1>만나서 반가워요!</h1>
    <a href="/greeting/">greeting</a>
    <a href="/dinners/">dinners</a>
  {% endblock content %}
</body>
```

form

1. action
    - 입력 데이터가 전송될 URL을 지정
    - 데이터를 어디로 보낼 것인지 지정하는 것이며 이 값은 반드시 유효한 URL이어야 함
    - 만약 이속성을 지정하지 않으면 데이터는 현재 form이 있는 페이지의 URL로 보내짐
2. method
    - 데이터를 어떻게 보낼 것인지 정의
    - 입력 데이터의 HTTP request methods를 지정
    - GET 이나 POST 를 이용 → 명시적 표현을 위해 GET(대문자로 지정)
        - GET를 이용하면 URL의 변화를 볼 수 있다.
            - http://host:port/path?key=value&key=value

- 데이터 가져오기
    - 모든 요청 데이터는 view 함수의 첫번째 인자 request에 들어있다.

Variable routing

- 템플릿의 많은 부분이 중복 → 해결 : variable routing
- url 주소를 변수로 사용하는 것을 의미
- url의 일부를 변수로 지정하여 view 함수의 인자로 넘길 수 있음

- 즉, 변수 값에 따라 하나의 path()에 여러 페이지를 연결 시킬 수 있음
- 변수는 <> 에 정의하고 view 함수의 인자로 할당됨
- 기본 타입은 string이며 5가지 타입으로 명시할 수 있음

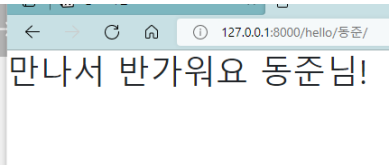
```python
# articles/views.py

def hello(request, name):
    context = {
        'name': name,
    }
    return render(request, 'hello.html', context)
```

```
<!-- articles/templates/hello.html -->

{% extends 'base.html' %}

{% block content %}
  <h1>만나서 반가워요 {{ name }}님!</h1>
{% endblock %}
```



App URL mapping

- 앱이 많아졌을 때 urls.py를 각 app에 매핑하는 방법
- 두번째 app을 생성하여 연습
- app의 view함수가 많아지면서 사용하는 path() 또한 많아지고,
- app 또한 더 많이 작성되기 때문에 프로젝트의 urls.py에서 모두 관리하는 것은 프로젝트 유지보수에 좋지 않음
- 메인 페이지의 주소가 바뀜
  - ~~~/index/ → ~~~/articles/index/

Naming URL patterns

- path 함수에 작성한 name을 사용하여 번거로움을 해소하는 방법
- html에서 {% url ‘name’ %} 와 urls에서 path(~~~ , name=’이름’) 을 이용하여 사용