Day2 (django template 그리고 urls)

자료	Django
구분	Django

오늘은 template 그리고 urls에 대해서 더 살펴보자.
장고 템플릿은 뭐하는곳이다? 데이터를 표현과 관련된 곳이다.
오늘은 장고 DTL (django template language)를 이용해서 html문서에 '동적컨텐츠'를 넣어볼 것이다.

DTL은 조건 반복 변수 필터 기능을 제공해 준다.
조건 반복문이 된다고.. 코드가 파이썬 느낌이 난다는 이유로
python이 HTML에 포함된 것이라고 생각 할 수도 있는데
절대 아니다. DTL은 python이랑 상관이 없는 장고템플릿언어 다!

DTL에는 여러 기능이 있지만
variable
filters
tags 그리고
주석(comments)을 보자.

variable가 변수라는 것을 모르는 사람은 없을 것이다. 변수는 필터랑 같이 보겠다.
그럼 filter는 무엇인가?
변수가 보여지는 것을 살짝 바꿔서 보여줄 때 사용한다.
예를 들어 변수에 20이 들어가 있다고 한다면.. 이 변수 값을 10을 더한 값으로 보여 줄 때 사용한다.
10을 더한 값으로 보여주더라도 변수의 값은 바뀐 것이 아니다.

tag는 조건 반복 등.. 여러가지 기능들이 있는데 많이 사용 하는 것 위주로 조금 이따가 하나 씩 살펴볼 것이다.

[잠깐 tip] ctrl + j 를 누르면 터미널 창을 열고 닫을 수 있다.
[잠깐 tip] 참고로 views.py 에 render 함수를 컨트롤 클릭을 하면 함수 내부를 확인할 수 있다.

def render(request, template_name, context=None, content_type=None, status=None, using=None):

첫번째, 두번째, 세번째 매개변수를 확인해 보면
request 그다음 템플릿이름 그다음에 콘텍스트를 인자값으로
넘겨 주면 되는구나 를 확인할 수 있다.

변수부터 살펴보자.
view 함수에서 template에 변수 값을 넘겨줌으로써 template를 동적으로
만들어 보겠다.

이 콘텍스트는 key value 형태 즉, 딕셔너리를 이용해서 값을 넘겨 줍니다.

다시말해 render 함수의 세번째 인자로 딕셔너리 형태로 변수를 넘겨줄 것이다.

```
from django.shortcuts import render

# Create your views here.
def index(requests):

  name='kevin'
  return render(requests, 'articles/index.html',{'name':name})
```

name 이라는 key 그리고 kevin의 변수 name을 value로 써주었다.

이렇게 딕셔너리의 형태로 views 에서 templates 로 변수 값을 넘겼다.

자 그럼 tempaltes 에서 넘겨 받은 값을 사용하면 됩니다.

```
<h1>Hi {{name}}</h1>
```

이런 것을 하는 것이 바로 변수의 사용이다.

이번에는 name만 넘기는 것이 아니라 info라는 딕셔너리를 만들어서 넘겨 볼 것이다.

```
from django.shortcuts import render

# Create your views here.
def index(requests):
  info = {
    'name':'KEVIN',
    'age':21,
  }
  name='kevin'
  return render(requests, 'articles/index.html',{'info':info})
```

그 다음 나이만 가져다 가 쓸 것이라면?

```
<h1>Hi {{info.name}}</h1>
<h2>나이는 : {{info.age}}</h2>
```

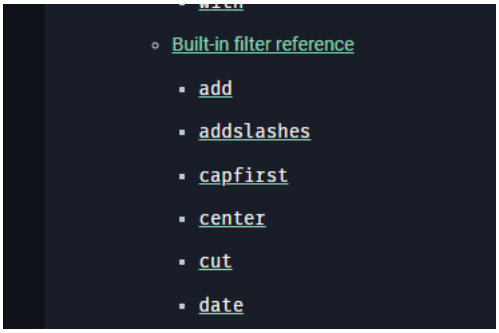
이번엔 filter다.

필터는 {{ 변수 버티컬바 | 그리고 적용시킬 필터}} 를 넣으면 된다.

바로 template을 보자.

```
<h1>Hi {{info.name}}</h1>
<h2>나이는 : {{info.age|add:2}}</h2>
<h3>이름을 소문자로 : {{info.name|lower}}</h3>
```

약 60개 이상의 빌트인 필터가 있으니까 공식문서를 확인하세요



https://docs.djangoproject.com/en/4.2/ref/templates/builtins/

이번에는 tag다. 태그도 연습해 보자

if 한 다음에 tab 키를 눌러보자.

```
<h1>Hi {{info.name}}</h1>
<h2>나이는 : {{info.age|add:2}}</h2>
<h3>이름을 소문자로 : {{info.name|lower}}</h3>

{% if info.age == 21 %}
  <p>나는 부럽지가 않아</p>
{% endif %}
```

for도 한번 써보자.

```
from django.shortcuts import render

# Create your views here.
def index(requests):
  info = {
    'name':'KEVIN',
    'age':21,
    'color':['red','black','white']
  }
  name='kevin'

  return render(requests, 'articles/index.html',{'info':info})
```

```
<h1>Hi {{info.name}}</h1>
<h2>나이는 : {{info.age|add:2}}</h2>
<h3>이름을 소문자로 : {{info.name|lower}}</h3>

{% if info.age == 21 %}
  <p>나는 부럽지가 않아</p>
{% endif %}

{% for color in info.color %}
  <li>{{color}}</li>
{% endfor %}

<p>내가 좋아하는 컬러는 {{info.color.0}} 입니다.</p>
```

여기까지 DTL을 보았다.

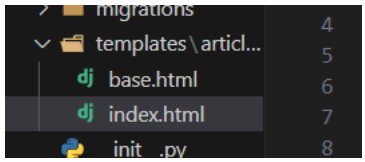
==========

이번에는 template 상속에 대해서 보자. 상속에는 부모와 자식 템플릿이 존재한다.

예를 먼저 들어보면 부모 템플릿에서 기본 구조를 잡아두고

여러 개의 자식 템플릿에서 공통적인 부분이 아닌 일정 부분만 바꿔서 사용하는 경우
상속을 사용한다.

templates > articles > 폴더에 base.html 파일을 하나 만들어 보자



그리고 base.html에는 공통적으로 사용할 코드를 적어줄 것이다.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-4bw+/aepP/YC
</head>
<body>
  <h1>********************************</h1>
  {% block content %}

  {% endblock content %}
  <h1>********************************</h1>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-HwwvtgBNo3bZJJLYd8oVXjr
</body>
</html>
```

base.html 파일 내용 그대로 자식 컨텐츠에게 상속을 주는데
block 부분은 자식컨텐츠의 내용으로 싹 들어갈 것이다.

그다음 index.html을 자식 컨텐츠로 사용해 보자.
extends 를 이용해서 자식템플릿이 부모 템플릿을 확장할 것이다. 즉, 상속을 받을 것이다. 그리고
block 을 넣어서 부모 컨텐츠의 block에 들어갈 내용을 채워보자.

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

{% block content %}

<h1>Hi {{info.name}}</h1>
<h2>나이는 : {{info.age|add:2}}</h2>
<h3>이름을 소문자로 : {{info.name|lower}}</h3>

{% if info.age == 21 %}
  <p>나는 부럽지가 않아</p>
{% endif %}

{% for color in info.color %}
  <li>{{color}}</li>
{% endfor %}

<p>내가 좋아하는 컬러는 {{info.color.0}} 입니다.</p>

{% endblock content %}
```

저장 후 확인해 보자

음.. 아니 그런데 갑자기 궁금하다. django는 어떻게 template을 찾아서 알아서 처리를 해 주는 것일까?
프로젝트의 설정을 총괄하는 settings.py로 가보자.

django가 템플릿을 알아서 처리해 주는 것에 대한 비밀을 'APP_DIRS'에 있다.

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
```

'APP_DIRS' 값이 true 라고 되어 있는데 이것이 의미 하는것은!

django야 'APP_DIRS' 앱 디렉토리 안쪽에
template가 있는데 거기 한번 뒤적뒤적 해서 탬플릿 한번 찾아줄래? 라는 옵션이 켜져 있는것이다.
그래서 장고가 알아서 template를 잘 찾는다.

그런데.. 나중에 개발을 하다보면… 앱이 1개가 아니라 여러개를 만들어서 개발을 할 것이다.
그럼 base.html 파일이 각각의 앱마다 존재 한다면???

users
articles 앱에,,, 각각의 앱의 템플릿에 부트스트랩 코드를 2개나 넣을 필요가 있을까? 중복으로 할 필요가 없네!! 그렇다면
base.html 파일을 앱 안에 넣는것이 아니라 프로젝트 밖으로 뺄 수 있을 것이다.

그리고 장고가 템플릿을 찾을때
'APP_DIRS' → 앱 디렉토리만 뒤적뒤적 하지 말고
다른 디렉토리도 찾아볼래?
라고 하는 부분이 'DIRS': [], 라고 보면 된다.

그래서 앱 밖에 프로젝트에 template 폴더를 만들고
부모 base.html 파일을 만들고 나서….
 'DIRS': [] 에다가 템플릿들이 들어가 있는 경로를 적어 줄 것이다.

[오늘의 tip] ctrl + w를 누르면 창이 하나씩 닫힐 것이다.

자 그래서 지금 할 것은 base.html을 밖으로 꺼낼 것이다.
프로젝트 폴더에 templates 폴더를 하나 만들어 보자
(이제는 이름이 꼭 tamplates가 아니여도 되지만… 관례상 템플리츠로 하자.)

그리고 base.html파일을 템플릿츠로 드래그 해서 move 해주자.
그 다음, 'DIRS': [] 이곳에

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR/'templates'],
        'APP_DIRS': True,
```

라고 수정을 해보자. BASE_DIR 이건 뭐지???

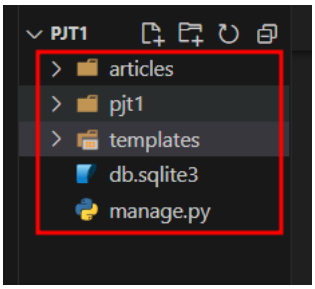
ctrl+click을 해보자.

BASE_DIR = Path(**file**).resolve().parent.parent

지금 file (settings.py) 파일에 해당하는 곳의

부모의 부모.... 그러니까 settings의 부모 (프로젝트이름폴더) 의 부모 (프로젝트 파일)

즉 프로젝트의 가장 최상단 경로를 BASE_DIR로 정의를 하겠다고 써 있다.



자 그다음에 index.html 파일의 부모경로를 수정해 주자.

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

이것이 템플릿 상속 입니다.

자 그럼 한번 더 실습을 해보자. 앱을 하나 더 만들어 보자.

pages 라는 앱을 만들 것이다.

```
$ python manage.py startapp pages

settings.py 로 가서
installed apps 에 'pages', 앱 등록

urls.py에서
from articles import views as article_views
from pages import views as pages_views

(별칭을 안붙이면 views가 article views인지 pages 의 views인지 인식못함)

그리고
path('articles/',article_views.index),
path('pages/',pages_views.index),          경로등록

--------------------------------------------------

views.py에서
def index(requests):
    return render(requests,'pages/index.html')   함수정의

pages 앱 안에 templates 폴더만들고
그 안에 pages 폴더 만들고 index.html 파일 생성하기
```

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

{% block  content%}
<h1> pages 앱의 test 입니다.</h1>
{% endblock content %}


$ python manage.py runserver
```

이게 상속 끝이다.

Variable Routing에 대해서 살펴보자.

variable 라우팅이란 url주소를 변수로 사용 하는 것을 의미한다. 예를 들어보자.

만약에 www.민호.com/profile/kevin 이라고 하면 kevin의 프로필 사진을 보여줘야 하고

만약에 www.민호.com/profile/kate 라고 하면 kate의 프로필 사진을 보여줘야 한다고 하자.

유저가 1000명이면 1000명의 url을 직접 다 지정해 줄 수 없다.

이때, url일부를 변수로 지정해서 view함수의 인자로 넘길 수 있다.

그래서 변수 값에 따라 “하나의 path” 에 여러 페이지를 연결 시킬 수 있다. 직접 보자.

프로젝트의 urls.py에 들어가 보자.

```
from django.contrib import admin
from django.urls import path
from articles import views as article_views
from users import views as users_views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('articles/',article_views.index),
    path('pages/<str:name>/', pages_views.index),

    url 요청을 받을때
    pages 다음에 문자열이 매칭이 되면
    pages_views.index에서 처리하라는 뜻이다.

    끝에 / 꼭 붙여줘라.
    Django url 끝에 / (트레일링슬러시)가 없다면
    자동으로 붙여 주는 것이 기본 설정이다.
    그래서 모든 주소가 / 로 끝나도록 되어있다.
]
```

그 다음에 users 앱의 views.py 에서 이름을 받아주자.

context 하나 만들어서 넘겨주자.

```
from django.shortcuts import render

# Create your views here.
def index(requests,name):

    context={
        'name':name
    }
    return render(requests,'pages/index.html',context)
```

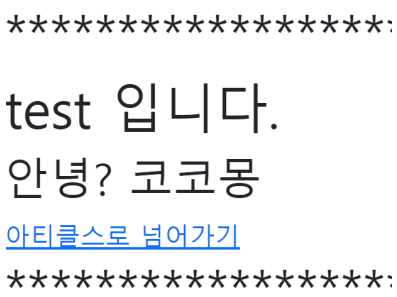
index.html로 가서

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

{% block  content%}
```

```
<h1>test 입니다.</h1>
<h2>안녕? {{name}}</h2>

<a href="/articles/"> 아티클스로 넘어가기 </a>

{% endblock content %}
```

서버키고

/pages/코코몽 이라고 해보자



==========

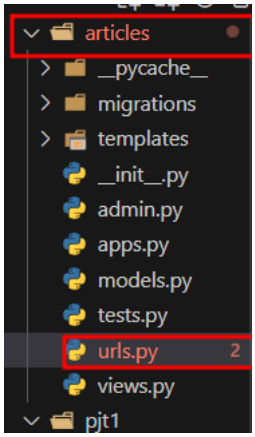
이번에는 APP URL Mapping 하는것을 해보자.

APP URL Mapping은 무엇이냐면… 만약에 우리가 프로젝트를 개발하는데 앱 (기능)도
다양하게 많이 있다고 가정하자. 그렇다 보면
모든 앱의 경로를 “프로젝트의 urls.py” 에 다 적어주면
코드 가독성도 떨어지고 유지보수 에도 좋지 않다.

그래서 우리가 지금 할 것은
프로젝트 폴더 안에 urls.py 파일 안에 모든 경로를 다 적어 줄 것이 아니라…
각각의 앱에 각각의 urls.py 를 만들어서 경로를 적어주는 것을 분리를 할 것이다.

1차적으로 프로젝트의 urls.py에서 요청을 받으면…
어.. 너는 articles의 urls.py에서 처리하면 되고
어.. 너는 users 의 urls.py에서 처리하면 될 것 같아… 라고 처리할 수 있을 것이다.
그리고 나서 각 앱의 urls.py에서 각 views.py 로 연결을 시켜 줄 것이다.

먼저 pages 앱 부터 보자.
pages 폴더 안에 urls.py 파일을 하나 만들어 보자.




```
from django.urls import path

urlpatterns = [

]
```

from django.urls import path는 갑자기 왜? 어디서 나왔는가??

프로젝트의 urls.py 파일을 복붙 해보자. 그 다음 필요 없는 것을 다 지워보자.

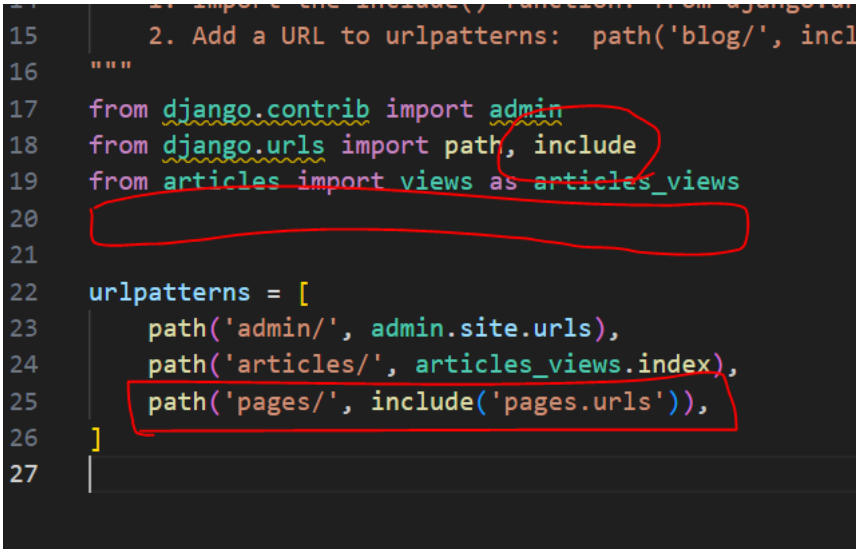
그러면 남는 것 … 위에 코드가 urls의 기본 구조다. !!!

```
"""
URL configuration for pjt1 project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from articles import views as article_views
from pages import views as pages_views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('articles/',article_views.index),
    path('pages/<str:name>/',users_views.test),
]
```

자 다시한번 말하면 클라이언트로 부터 요청이 들어오면

프로젝트의 urls에서 각 앱의 urls로 일감을 뿌려 줄 것이다.

그래서 프로젝트의 urls.py로 가보자.



include() 라는 함수는 프로젝트 내부 앱들의 URL을 참조 할 수 있도록 맵핑하는 함수다.
그래서 클라이언트로 부터 요청이 들어올 때 pages 로 들어오면… 그때는 include 함수 안
pages.urls에서 처리를 해!! 라는 뜻이다.

www.민호.com/users/ 이후에 들어오는 주소를 pages 앱에서의 urls.py에서 처리를 하자
다시 urls.py 앱으로 넘어가보자.

```
from django.urls import path
from . import views

urlpatterns = [
    path('greeting/<str:name>/',views.index)

]
```

import 문 하나 추가하고
path에는 pages를 지우자. pages/ 이후에 들어오는 주소는 여기서 처리할 것이기 때문이다.

서버 켜서 확인해 보자.
http://127.0.0.1:8000/users/greeting/콩순이/

하나더 추가 하자면… articles 앱 그리고 pages 앱 모두
index.html 파일이 있다. 그러면 나중에 어떤 index.html 파일인지
구분을 짓기 위해서 앱 네임을 추가할 수도 있다.

```
from django.urls import path
from . import views

app_name='pages'

urlpatterns = [
    path('greeting/<str:name>',views.index)
]
```

==========

Throw and catch는 라이브 방송을 보고 공부해보아라.