Day2 (Django template 그리고 urls)

자료	Django
구분	Django
과목	

오늘은 Django 에서 템플릿(template) 그리고 유알엘(urls) 에 대해서 더 살펴 볼 것이다.

장고에서의 템플릿의 역할은 무엇인가? “데이터를 표현하는 것과 관련된 곳이다.”

오늘은 장고 템플릿 랭귀지 DTL (Django Template Language) 라는 것을 이용해서 html 문서에

‘동적컨텐츠’를 넣어볼 것이다.

DTL은 (장고 템플릿 언어) Django HTML에서 조건(if) 반복(for) 변수{ } 필터 기능을 제공해 준다.

DTL에는 여러가지의 기능이 있지만 우선적으로 많이 사용하는

Variable / Filters / Tags 그리고 주석(Comments)에 대한 설명을 한번 들어보자.

- Variable가 변수라는 것을 모르는 사람은 없을 것이다.
- FIlter는 언제 사용할까? FIlter는 변수가 화면에 보여지는 것을 살짝 살짝 바꿔서 보여줄 때 사용한다. 예를 들어 변수에 20이 들어가 있다고 가정하자. 이 변수 값을 10을 더한 값으로 화면에 보여 줄 때 사용한다. 화면에 변수 값에 10을 더한 값으로 보여주더라도 실제 변수의 값이 바뀐 것이 아닐 것이다.
- Tag는 무엇일까?

  Tag는 조건문(for) 반복문(if) 등의 기능들이 있는데 차후에 하나씩 살펴볼 것이다.

변수

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

```
from django.shortcuts import render

def index(requests):

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

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

우리가 render 함수를 returen 값으로 호출을 할때 세번째 인자로

`{'name':name}` 를 추가로 적어 주었다.

render 함수의 세번째 인자값 `{'name':name}` 를 index.html 파일로 넘겨 줄 것이라는 뜻이다.

HTML 파일로 넘어갈 인자값은 항상 딕셔너리의 형태로 값을 넘겨 준다.

Django에서는 인자값으로 넘어가는 딕셔너리를 context라고 한다.

자 그럼 tempates 에서 넘겨 받은 값을 사용하면 된다.

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

방금은 name 하나만 HTML 파일로 값을 넘겨 주었다면, 이번에는 여러개의 값을 HTML 파일로 넘겨보자.

```
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})
```

나는 함수 안에서 info 라는 이름의 하나의 딕셔너리에 여러개의 값을 넣었다.

그리고 context를 만들어서 HTML 파일로 넘겨 볼 것이다.

그리고 템플릿에서는 파이썬에서 딕셔너리를 사용 했던 것과 같이 . 을 사용해서 변수의 속성에 접근할 수 있다.

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

Filter

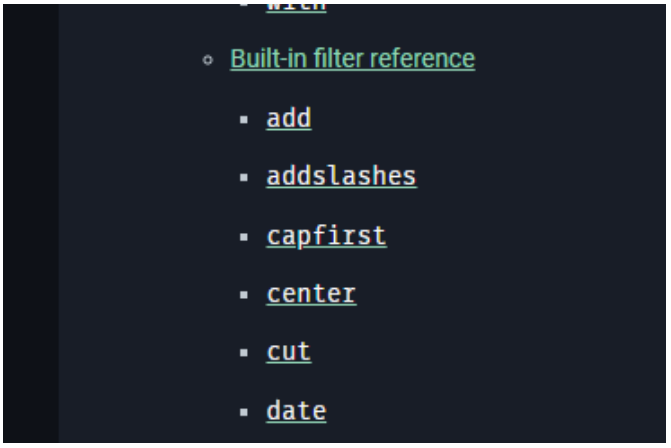
이번에는 filter 를 살펴보자.

필터는 {{ 변수 + (버티컬바 |) + 적용을 시킬 필터이름 }} 를 넣어주면 된다.

바로 template에 적용 해 보자.

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

필터에 적용시킬 수 있는 기능들은 정말 많다. 약 60개 이상의 built-in 필터들이 있으므로 공식문서를 통해서 그때 그때 필요한 필터를 찾아서 사용한다.



https://docs.djangoproject.com/en/5.2/ref/templates/builtins/

Tag

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 너의 나이가 부럽지 않아</p>
{% endif %}
```

- for도 한번 써보자.

```
from django.shortcuts import render


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 너의 나이가 부럽지 않아</p>
{% endif %}

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

DTL은 여기까지만 살펴보자.

template 상속

이번에는 template 상속에 대해서 보자.

상속의 개념에는 "부모 템플릿"과 "자식 템플릿"이 존재한다. 우리는 상속을 언제 사용하게 될까?

여러개의 자식 템플릿에서 반복적으로 또는 공통적으로 사용할 코드들은 부모 템플릿에서 기본 구조로 정의 한다.

그리고 자식 템플릿은 부모템플릿의 내용을 상속을 받아서 사용한다면, 자식템플릿에서의 의미없는 반복을 크게 줄일 수 있을 것이다.

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@5.3.3/dist/css/bootstrap
  </head>
  <body>
    <h1>**********</h1>
    {% block content %}

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

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.
  </body>
</html>
```

(부트스트랩 CDN은 부트스트랩 사이트에 들어가서 직접 CDN을 복사 해 주세요)

base.html 파일 내용을 “그.대.로.” 자식 컨텐츠에게 상속을 할 것이다.

`{% block content %}`와 `{% endblock %}` 사이 공백(content) 부분에

자식 컨텐츠의 내용으로 “쏙!” 들어갈 것이다.

그다음 index.html 파일을 자식 컨텐츠로 사용해 보자.

```
{% 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 %}


{% endblock content %}
```

extends 를 이용해서 자식 템플릿이 부모 템플릿을 확장할 것이다. 즉, 상속을 받을 것이다.

그리고 block content를 이용해서 부모 컨텐츠에서의 내용이 index.html 파일에 적용이 될 것이다.

저장 후 확인해 보자.

여기서 주의 할 점은 base.html 파일로 부터 상속을 받겠다고 명시한 부분인

`{% extends 'articles/base.html' %}` 을 index.html 파일의 가장 상단(최 상단)에 작성해야 한다는 점이다.

음.. 그런데 갑자기 궁금하다.

Django는 어떻게 template을 척척 알아서 찾아서 일을 처리를 해 주는 것일까?

프로젝트의 설정을 총괄하는 settings.py 로 한번 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' 의 값이 True 라고 되어 있는데 이것이 의미하는 것은 다음과 같다.

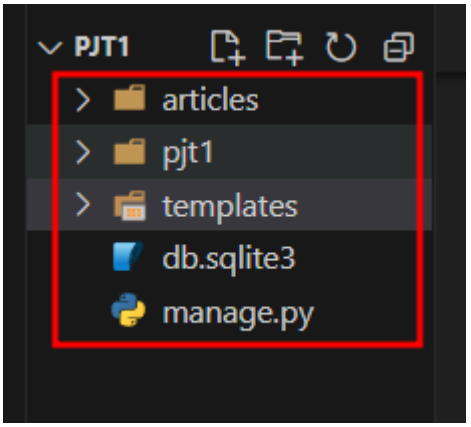
Django야~ 'APP_DIRS' 앱 안에 있는 폴더들을 살펴보면 template라는 폴더가 있는데 거기를 뒤적뒤적 해서 index.html이라는 탬플릿을 찾아줄래?
라는 옵션이 켜져 있기 때문이다.

저 옵션값을 True로 해 놓았기 때문에 Django 프레임워크 내부에서 필요한 template를 알아서 잘 찾아 주는 것이다.

그런데 나중에 개발의 규모가 커지면 우리의 프로젝트 안에는 지금과 같이 1개의 앱만 존재하는 것이 아니라 여러개의 앱이 존재 할 것이다.

따라서 우리는 프로젝트 안에 있는 모든 앱에서 상속받을 html 파일을 생성하기 위해서 프로젝트 폴더에 `templates` 이름의 폴더를 새로 생성을 할 것이다.
그리고 프로젝트 안에 있는 모든 앱의 html 파일에 상속 받아서 사용 할 수 있도록 만들어 보자.

실습해보자.
지금부터 전역으로 모든 앱의 html 문서에 적용시킬 base.html 을 밖으로 꺼낼 것이다.



1. 먼저, 프로젝트 폴더에 templates 폴더를 하나 만들어 보자.

   폴더 위치를 주의하자. articles앱의 상위폴더인 프로젝트 폴더에 templates 폴더를 만드는 것이다.

2. 그곳에 아까 만들었던 base.html 파일을 templates 폴더로 드래그 해서 옮겨주자.
3. 프로젝트에 존재하는 모든 앱에서 상속받을 파일의 경로를 적어 줄 것이다.
   a. settings.py 파일의 'DIRS': [] 에 작성한다.

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

(이제부터는 폴더 이름이 꼭 templates가 아니여도 되지만!... 관례상 templates로 해두자.)

Django가 전역으로 만들어 놓은 base.html 템플릿을 찾을 때는

'APP_DIRS' 을 통해서 각 앱들의 templates 디렉토리만 뒤적뒤적 하지 말고,

앱 폴더 밖에 있는 다른 폴더도 찾아볼래?

라고 이야기 해 줄 수 있는 부분이 'DIRS': [] 부분 이라고 보면 된다.

[참고]

자 그런데.. BASE_DIR 이것은 왜 적어줄까? ~~~(프레임워크 사용법이 그러하니까 그렇

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 ㅍ

즉 프로젝트의 가장 최상단 경로를 BASE_DIR로 정의를 하겠다!! 라고 써 있다.

자 그다음에 articles 앱에 들어가서 index.html 파일이 상속 받을 부모의 경로를 수정해 주자.

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

****

Hi KEVIN

나이는 : 23

이름을 소문자로 : kevin

나는 부럽지가 않아

- red
- black
- white

****

지금까지 실습한 것이 템플릿 상속이다 .

자 그럼 복습 겸 한번 더 실습을 해보자.

앱을 하나 더 추가로 만들어 보자.

pages 라는 앱을 만들 것이다. (이 과정은 숙지해야 한다!)

1. $ python manage.py startapp pages

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

3. urls.py에서
   from articles import views as articles_views
   from pages import views as pages_views

(as 를 이용해서 별칭을 안붙이면 views가 articles views인지 pages 의 views인지 인

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

---

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

6. pages 앱 안에 templates 폴더만들고 > 그 안에 pages 폴더를 만든 후
   index.html 파일을 생성한다.

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

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

7. 서버 실행 $ python manage.py runserver

이게 template 상속 끝이다. 간단하다 😊

Variable Routing

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 가정하자.

그런데.. 유저가 1000명이면 1000명의 url을 직접 다 urls.py 에 직접 지정을 해 줄 수는 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.

www.민호.com/profile/{변수값}

프로젝트의 urls.py에 들어가 보자.

pages 경로를 조금 수정해 주자.

```
from django.contrib import admin
from django.urls import path
from articles import views as articles_views
from pages import views as pages_views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('articles/',articles_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 pages 앱의 views.py 에서 이름을 받아주자.

index 함수에 name 이라는 매개변수를 만들어 url을 통해서 전달받는 값을 인자로 받을 것이다.

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)
```

pages > index.html로 가서

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

{% block content %}

<h1>pages 앱의 test 입니다. </h1>
<h2>{{name}}님 반갑습니다. </h2>

<a href="/articles/"> 아티클스로 넘어가기</a>
```

```
{% endblock content %}
```

a 태그를 이용해서 해당 프로젝트의 다른 앱으로 넘어가는 코드도 추가해 보았다.

서버를 켜서 /pages/최민호 라고 적어 보았다.

****

pages 앱의 test 입니다.

최민호님 반갑습니다.

아티클스로 넘어가기

****

==========

APP URL Mapping

이번에는 APP URL Mapping 하는것을 해보자.

APP URL Mapping은 무엇일까? 만약에 우리가 프로젝트를 개발하는데 있어서

프로젝트 안에 여러 개의 앱이 존재 한다고 가정하자. 그렇다 보면

모든 앱의 경로를 "프로젝트" 폴더 안에 있는 urls.py" 에 모두 적으면

경로가 너무 많이 적히게 됨으로써 코드의 가독성도 떨어지고

프로젝트의 유지보수 측면에서도 좋지 않을 것이다.

그래서 우리가 지금 할 것은

프로젝트 폴더 안에 있는 urls.py 파일 안에 프로젝트 모든 앱들의 경로를

한곳에 몰아서 작성할 것이 아니라,

각각의 앱에 , 각각의 urls.py 파일을 만들어서 경로를 분산 시켜 줄 것이다.

즉, 경로(path)를 프로젝트의 urls.py에 모두 다 몰아서 작성하는 것이 아니라

각 앱에 urls.py 파일을 새로 생성하여 경로(path)를 분리 시켜 줄 것이다.

예를들면, 1차적으로 프로젝트의 urls.py에서 요청을 받으면

어.. 너는 articles 앱의 urls.py 파일에서 처리하면 되고

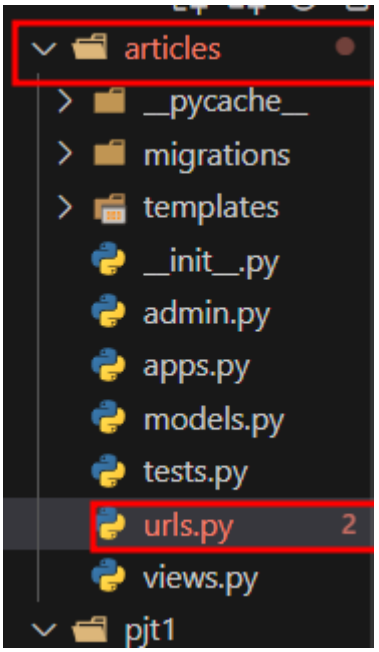
어.. 너는 pages 앱의 urls.py 파일에서 처리하면 될 것 같아..라고 경로(path)를

각 앱에서 처리할 수 있도록 할 것이다.

그리고 나서 각 앱의 urls.py 파일에서 각자의 views.py 에 있는 함수로 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 pages앱의 urls.py 파일에 [복붙] 해보자.

그 다음 필요 없는 것을 다 지워보면 위의 코드만 남을 것이다.

즉, 남은 코드가 바로 urls.py 파일의 기본 구조다.

다시, 프로젝트 폴더의 urls.py 파일로 가보자.

```
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>/',pages_views.test),
]
```

자 우리가 하고자 하는 것은 클라이언트로 부터 요청이 들어오면

프로젝트의 urls.py에서 각 앱의 urls로 일감을 뿌려 줄 것이다.

그래서 프로젝트의 urls.py로 가서 코드를 조금 수정 할 것이다.

```
from django.contrib import admin
from django.urls import path, include
from articles import views as articles_views

urlpatterns = [
    path('admin/', admin.site.urls),
```

```
    path('articles/', articles_views.index),
    path('pages/', include('pages.urls')),
]
```

```python
from django.contrib import admin
from django.urls import path, include  # include 추가
from articles import views as articles_views
# from pages import views as pages_views 삭제!!!

urlpatterns = [
    path('admin/', admin.site.urls),
    path('articles/', articles_views.index),
    path('pages/', include('pages.urls')),  # 수정
]
```

include() 라는 함수는 프로젝트 내부 앱들의 URL을 참조 할 수 있도록 맵핑을 도와주는 함수다.

그래서 클라이언트로 부터 요청이 들어올 때 pages 앱에 관한 요청이 들어오면

그때에는 include 함수 안에 명시해 놓은 대로 pages 앱 안의 urls 파일에서 처리를 해!!

라는 뜻이다.

따라서 http://127.0.0.1:8000/pages/ 이후에 들어오는 주소를 pages 앱에 있는 urls.py 파일에서 처리를 할 것이다.

다시 pages 앱의 urls.py 파일로 넘어가보자.

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

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

]
```

위의 코드처럼 import 문 하나를 추가해 주자.

path에는 pages/ 이후에 들어오는 주소는 이제 여기서 처리할 것이다.

서버 켜서 확인해 보자.

http://127.0.0.1:8000/pages/greeting/최민호/

도전미션

스스로 도전해보자.

pages 앱의 url path를 따로 분리 시켰다.

이번에는 articles 앱의 url path를 분리 시키고자 한다.

articles앱에 urls.py 파일을 생성 후 http://127.0.0.1:8000/articles/samsung01 경로를 통해서

아티클 앱의 views.py 파일 안에 있는 index 함수를 호출해 보자.

아래 정답을 보지 말고 직접 도전해 보자.

[정답]

```
# firstpjt / urls.py

from django.contrib import admin
from django.urls import path, include


urlpatterns = [
    path('admin/', admin.site.urls),
    path('articles/', include('articles.urls')),
    path('pages/', include('pages.urls')),
]

----------------------------------------------------------------
```

```
# articles / urls.py

from django.urls import path
from . import views

urlpatterns = [
    path('samsung01/', views.index)
]
```

자, 도전미션을 잘 수행 했다면 다른 개념 하나만 더 추가 해보자.

URL namespace

“namespace” 라는 것을 볼 것이다. 우리가 앞서 pages 앱의 index.html 파일에

```
<a href="/articles/">아티클스로 넘어가기</a>
```

이렇게 프로젝트 파일 안에 있는 urls.py 파일에 있는 경로를 하드코딩의 방식으로 적어 보았다.

하지만.. 방금 과 같이 개발 과정 중에 articles 앱의 URL이

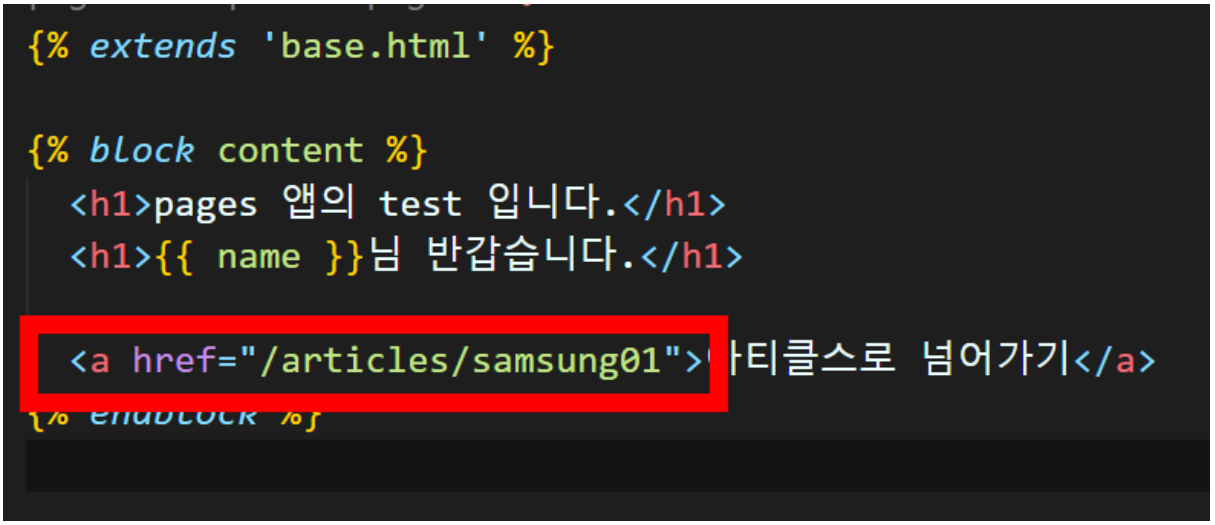
http://127.0.0.1:8000/articles/ 에서

http://127.0.0.1:8000/articles/samsung01 로 URL이 수정 되었다.

그렇다면 /articles/ 라는 URL을 사용했던 곳을 직접 찾아가 해당 경로(path)를

/articles/samsung01/ 로 URL을 수정을 해줘야 할 것이다.

pages 앱에서 index.html 파일로 이동해서 아래와 같이 코드를 수정하자.



만약에 다른 앱에서도 /articles/ 라고 url경로를 직접 하드코딩을 해 주었을 경우

개발 중간에 url이 바뀌게 되면 일일이 해당 `/articles/` url을 사용한 곳을 찾아 다니면서 직접 하나씩

`/articles/samsung01/` 으로 다 바꿔줘야 하는 불편함이 발생 할 수 있다.

실제로 프로젝트 URL 리팩토링은 자주 발생한다. 개발 중간에 URL을 수정하는 경우는

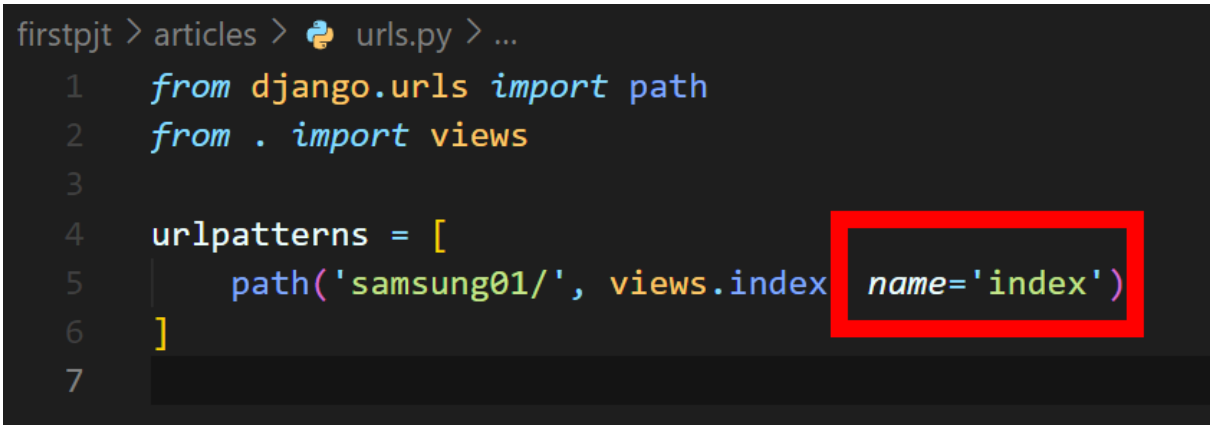
자주 발생한다. 따라서 앞으로는 a 태그 등을 사용해서 이동하고자 하는 경로를 적어 줘야 할 때에는

/articles/ 이렇게 하드코딩을 하는 것은 좋은 아이디어는 아닐 것이다.

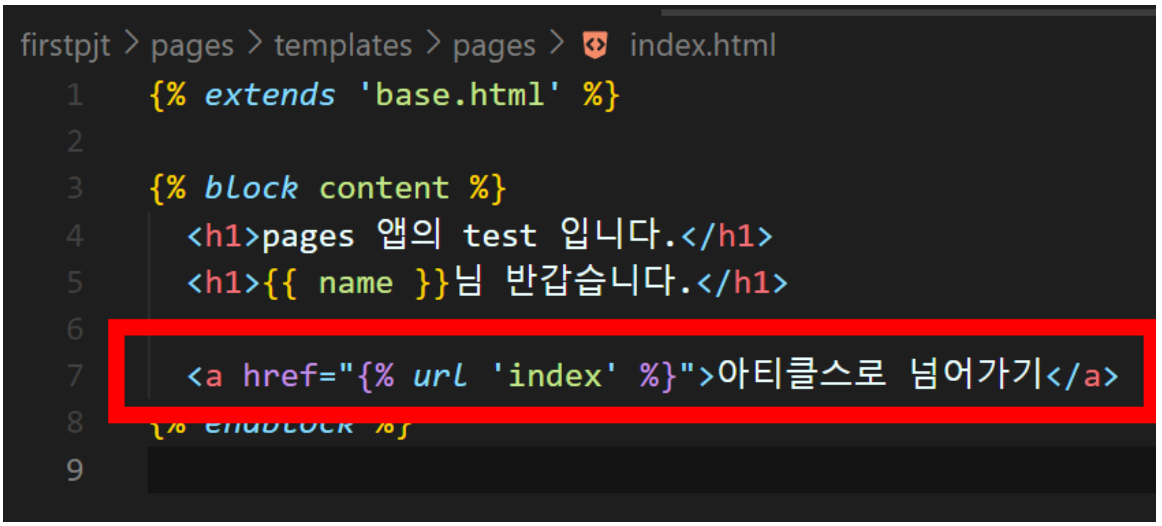
따라서 우리는, 각 앱의 urls.py 파일에 URL별칭을 부여해서, 나중에 개발 도중에 URL이 바뀌더라도

해당 URL을 일일이 찾아 다니면서 바꾸는 수고를 덜고자 한다.

articles 앱의 urls.py 파일에 코드를 추가해 보자.



그리고 pages 앱의 index.html 파일도 수정해 보자.



링크 잘 작동 되는지 확인해 보자.

****

Hi KEVIN

나이는 : 23

이름을 소문자로 : kevin

나는 부럽지가 않아

- red
- black
- white

****

자, 이렇게 경로의 name 속성을 부여한다면 개발도중에 URL이 바뀌더라도

해당 URL을 사용한 곳을 일일이 찾아 다니면서 바꿀 필요가 없어져 버린다. (굳)

우리는 앱이 articles만 있는 것이 아니다. pages 앱의 urls.py에 있는 경로도 URL 별칭을 붙여보자.

위와 같이 pages 앱의 urls.py 파일에 있는 path에 URL 별칭을 붙여 주었다. 그런데 여기서 문제가 발생한다.

articles 앱에서 사용한 URL 별칭과 pages앱에서 사용한 URL 별칭 모두 index 라고 정의를 하였다.

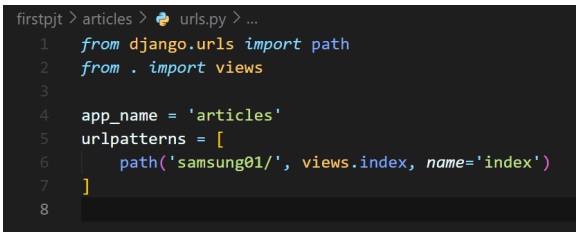
(공교롭게도 두 앱 모두 views.py 에 index 라는 함수가 존재한다. ^^;;)

그러면 차후에 URL대신 URL별칭으로 index 라고 사용을 하면.. 해당 index는 articles앱의 index 인지

혹은 pages앱의 index인지 Django가 구분을 하지 못 할 것이다.

따라서, 여기서 우리는 추가적으로 각 앱의 urls.py 파일에 이름을 부여할 것이다.

[이를 URL 네임스페이스를 부여한다 라고 한다.]



두 앱의 urls.py 파일에

app_name 을 추가 해 주었다.

자 이제

pages 앱의 index.ht ml 파일로 가서 경로를 수정해 줄 것인데

app name과 URL별칭을 사용하여 a 태그에 URL 경로를 적어줄 것이다.

```
firstpjt > pages > templates > pages > index.html
1  {% extends 'base.html' %}
2
3  {% block content %}
4    <h1>pages 앱의 test 입니다.</h1>
5    <h1>{{ name }}님 반갑습니다.</h1>
6
7    <a href="{% url 'articles:index' %}">아티클스로 넘어가기</a>
8  {% endblock %}
9
```

자 이번엔 추가로

articles 앱의 index.html 파일로 가서 경로를 추가해 보자.

```
firstpjt > articles > templates > articles > index.html
3  {% block content %}
8    {% if info.age == 21 %}
9      <p>나는 부럽지가 않아</p>
10   {% endif %}
11
12   {% for color in info.color %}
13     <li>{{ color }}</li>
14   {% endfor %}
15
16   <a href="{% url 'pages:index' info.name %}">페이지스로 넘어가기</a>
17 {% endblock %}
18
```

'pages:index' 다음에 info.name 을 통해서 해당 URL로 인자값도 같이 전달 하였다.

info 는 무엇일까??

info는 views.py 에서 넘겨준 info 라고 정의한 context 이다.

따라서 info.name이라고 하였으니 kevin이 출력이 되야 할 것이다.

실행 결과는 다음과 같다.

****

Hi KEVIN

나이는 : 23

이름을 소문자로 : kevin

나는 부럽지가 않아

- red
- black
- white

페이지스로 넘어가기

****

****

pages 앱의 test 입니다.

KEVIN님 반갑습니다.

아티클스로 넘어가기

****

Form tag 관련 연습은 라이브를 통해서 꼭 복습을 잘 하도록 하자.

==========

[혼동 금지]

- Variavle routing

  URL 주소를 통해서 변수값을 입력 받으며 그 값을

  view함수를 통해 templete에서 입력받은 값을 사용

  방법:

  1. urls.py에서 /<str:name> 넣기
  2. view 함수에서 매개변수로 인자값 받기

- Form tag

  input 태그를 통해서 변수값을 입력 받으면 그 값을

  view함수를 통해 template에서 해당 값을 사용

실습 해보자. 라이브를 복습 했다는 가정 하게 간단하게 살펴 보겠다.

throw-catch 실습이다.

articles 앱에서 form태그를 통해서 입력받은 값을

pages 앱에서 출력해보자.

articles > templates > index.html

```
{% extends "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 %}

<a href="{% url "pages:index" info.name %}"> 페이지스로 넘어가기</a>

<h1>Throw</h1>
<form action="{% url "pages:index" info.name %}" method="GET">
  <label for="minho"> Throw를 연습하자. </label>
  <input type="text" id="minho" name="msg">
  <input type="submit">
</form>

{% endblock content %}
```

pages > views.py

```
from django.shortcuts import render

# Create your views here.
def index(request,name):
    message=request.GET.get('msg')
    context={
        'name':name,
        'message':message
    }
    return render(request, 'pages/index.html',context)
```

pages > templates > index.html

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

{% block content %}

<h1>pages 앱의 test 입니다. </h1>
<h2>{{name}}님 반갑습니다. </h2>

<a href="{% url "articles:index" %}"> 아티클스로 넘어가기</a>

<h3>{{message}} 를 받았습니다. </h3>


{% endblock content %}
```

끝~~ 쉽다. 앞으로 Django를 학습하려면 오늘 수업 내용을 완벽히 이해해야 할 것이다.

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

13기 광주2반 화이팅!! 긴장감을 늦추지 말고 전진합니다 😊