데이터의 흐름 v2(애플리케이션이 2개있을때 예시)

127.0.0.1:8000/pages/index/

브라우저에 /pages/index로 접근이 오면 ->

1."프로젝트의" urls.py로 접근

2.path() 함수 호출 path함수는 매개변수 (3개까지 가질 수 있음)

ex)

```
urlpatterns = [
    path('admin/', admin.site.urls),
    # 무언가 추가
    path('articles/', views.index),
    path('hello/', views.hello, name="hello"),
    path(),
]
```

2-1 첫번째 매개변수 "들어올 URL"

```
urlpatterns = [
    path('admin/', admin.site.urls),
    # 무언가 추가
    path('articles/', views.index),
    path('hello/', views.hello, name="hello"),
    path('index'),
]
```

2-2 /index/ 접근왔을때, 호출해줄 함수를 적어주어야함.. 그러나 함수를 urls.py에 작성할 수 x

views.py에 작성해서, import

```
def index():
  pass
```

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

urlpatterns = [
    path('admin/', admin.site.urls),
    # 무언가 추가
    path('articles/', views.index),
    path('hello/', views.hello, name="hello"),
    path('index/', views.index),
]
```

구분자 추가

```
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),
    # 무언가 추가
    path('articles/', articles_views.index),
    path('hello/', articles_views.hello, name="hello"),
    path('index/', pages_views.index),
]
```

views함수 마저 구현

```
def index(request):
  return render()
```

views함수들은 http랜더를 리턴해야함

render(함수는 매개변수 3개 가질 수 있다)

render(request -> 고정 대신 , "html 파일명", context -> dic자료구조)

```
from django.shortcuts import render

# Create your views here.
def index(request):
  return render(request, "")
```

작성할 html 파일이 없기때문에 application에 templates 폴더 생성후 -> html 파일 생성

```
from django.shortcuts import render

# Create your views here.
def index(request):
  return render(request, "index.html")
```

Application URL mapping

등장 이유 (목적) :

```
from django.contrib import admin
from django.urls import path
from articles import views

urlpatterns = [
    path('admin/', admin.site.urls),
    # 무언가 추가
    path('articles/', views.index),
    path('hello/', views.hello, name="hello"),
]
```

"프로젝트" 폴더의 "urls.py" -> url들이 여러개가 등장 -> 해결하기 위해 등장

방법 : **각 application마다 urls.py를 만들어서 분리**

1. 애플리케이션이 최소 2개이상 (python manage.py startapp [애플리케이션 이름])
2. 각 애플리케이션마다 urls.py 생성
3. 애플리케이션의 urls.py에다가 프로젝트의 urls.py 내용을 복사

   ex)

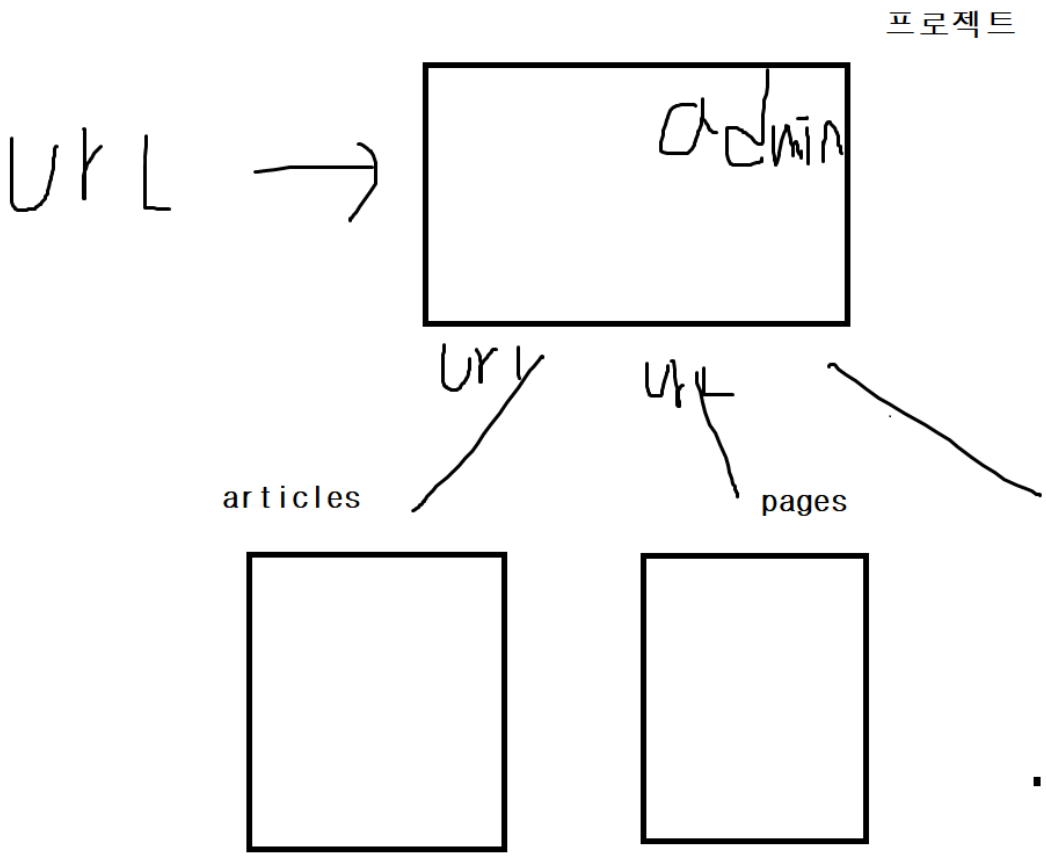
```
from django.contrib import admin
from django.urls import path
from articles import views

urlpatterns = [
    path('admin/', admin.site.urls),
    # 무언가 추가
    path('articles/', views.index),
    path('hello/', views.hello, name="hello"),
]
```

4. 필요 없는것 제거

   4-1 필요없다의 기준 (애플리케이션 이기때문에) admin 필요 없음 (아래 그림 참고)

   4-2 프로젝트의 코드를 복사해온 것 이기때문에, 경로가 다름(본인으로 변경 -> .)



ex)

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

urlpatterns = [
    path('articles/', views.index),
    path('hello/', views.hello, name="hello"),
]
```

app2 ex)

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

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

프로젝트 url 변경 (include 가져와서 매핑)

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

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

Name space

templates의 이름이 "같게"되면, 방문할 수 없는 문제가 발생

-> index 변수, isValid함수 -> 중복되기 쉽다

-> urls.py에 app_name 속성을 사용해서 해결

ex)

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


app_name = 'articles'
urlpatterns = [
    path('articles/', views.index),
    path('hello/', views.hello, name="hello"),
]
```

링크를 걸때 "어디의 : 파일명"

```
<a href="{% url 'pages:index' %}">두번째 앱으로 이동</a>
```

templates폴더안에, 애플리케이션명과 이름을 같게 폴더를 생성후, html파일들은 안으로 옮김

```
def index(reqeust):
  return render(reqeust, 'articles/index.html')
```

애플리케이션이 **아무리 많아져도** 같은 이름의 파일명과 경로를 사용할 수 있다

정적파일

(항상 고정된 파일) ,고정되어있는 css

1. settings.py -> INSTALLED_APPS에 django.contrib.staticfiles 있는지 확인

```
INSTALLED_APPS = [
    'articles',
    'pages',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

2. settings.py -> STATIC_URL 있는지 확인
3. 사용방법 : {% static 경로 %}
4. static 경로에 정적 파일을 저장

추가 설정)

STATICFILES_DIRS -> 추가적인 정적 파일 경로 목록을 정의

STATIC_ROOT -> collectstatic이라는 배포를 위해 파일을 수집하는 절대 경로(배포)

정적파일들 확인하기

settings.py

```
STATIC_URL = '/static/'

STATIC_ROOT = BASE_DIR / 'staticfiles'
```

-> 명령어 입력

```
python manage.py collectstatic
```

직접 static 파일 넣기

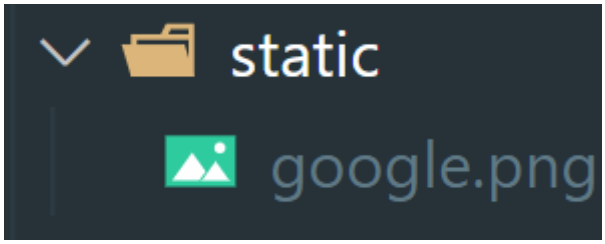
1. settings.py에 STATICFILES_DIRS넣기

```
STATIC_URL = '/static/'

STATIC_ROOT = BASE_DIR / 'staticfiles'

STATICFILES_DIRS = [
    BASE_DIR / 'static'
]
```

2.BASE_DIR에 STATIC 폴더생성 후 이미지 넣기



3. 애플리케이션으로 이동해서 load static 및 코드 작성하기

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

{% block content %}

<img src="{% static 'google.png' %}">

{% comment %} <a href="{% url 'hello' %}"> test로 이동 </a> {% endcomment %}
<a href="{% url 'pages:index' %}">두번째 앱으로 이동</a>
{% endblock  %}
```