Django

0.start

django

- Python Web Framework

- MVT pattern

- Open source

* homepage

0.start

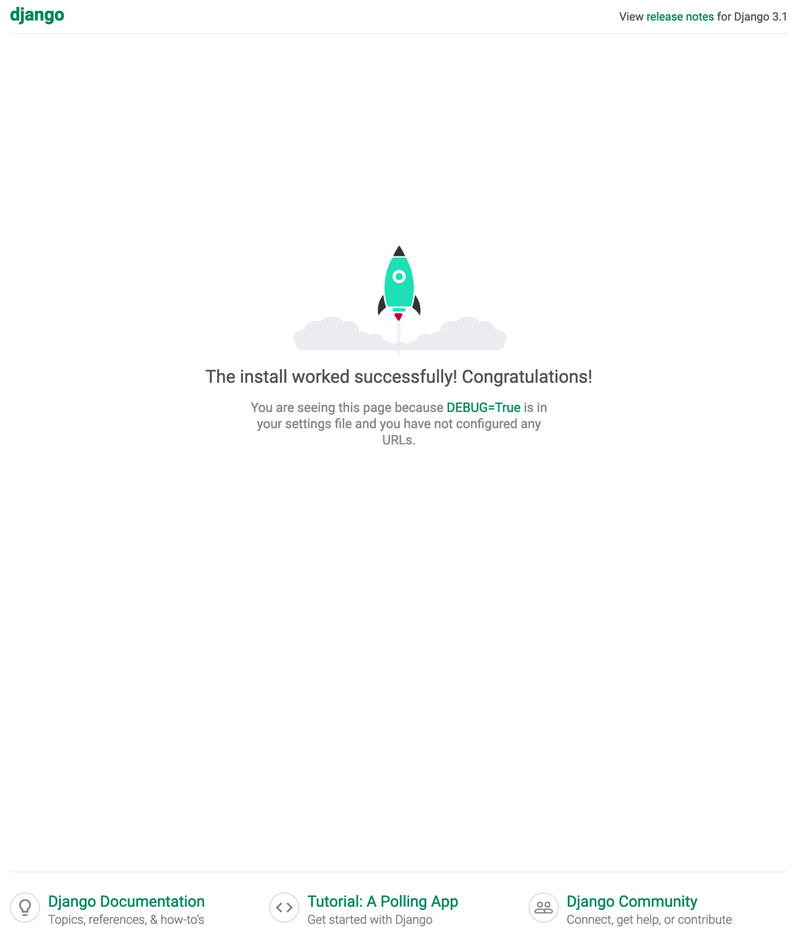
install

```
pip install django

django-admin startproject mysite

python manage.py runserver

# http://127.0.0.1:8000/
```



0.start

기본 구조

manage.py

- project 관리

- 주요 명령어
  startapp
  runserver
  migrate

```
mysite/
    manage.py
    mysite/
        __init__.py
        asgi.py
        settings.py
        urls.py
        wsgi.py
```

0.start

기본 구조

asgi.py

- Asynchronous Server Gateway Interface

- Web Server, Framework, WAS 의
  동기/비동기 통신을 지원하는 python interface

- django 3.0 부터 지원

```
mysite/
    manage.py
    mysite/
        __init__.py
        asgi.py
        settings.py
        urls.py
        wsgi.py
```

0.start

기본 구조

settings.py

- application 환경 설정

- 주요 설정
  DEBUG
  ALLOWED_HOSTS
  INSTALLED_APPS
  TEMPLATES
  DATABASES
  STATIC_URL

```
mysite/
    manage.py
    mysite/
        __init__.py
        asgi.py
        settings.py
        urls.py
        wsgi.py
```

0.start

기본 구조

urls.py

- URLconf (url configuration)

- url ⟷ function mapping

- 정규표현식으로 간단하게 표현 가능

```
mysite/
    manage.py
    mysite/
        __init__.py
        asgi.py
        settings.py
        urls.py
        wsgi.py
```

0.start

기본 구조

wsgi.py

- Web Server Gateway Interface

- Web Server, WAS 의 통신을 지원하는
  python interface

* wsgi → asgi
  비동기 통신 지원이 어려워서 asgi로 대체

```
mysite/
    manage.py
    mysite/
        __init__.py
        asgi.py
        settings.py
        urls.py
        wsgi.py
```

1.MVT

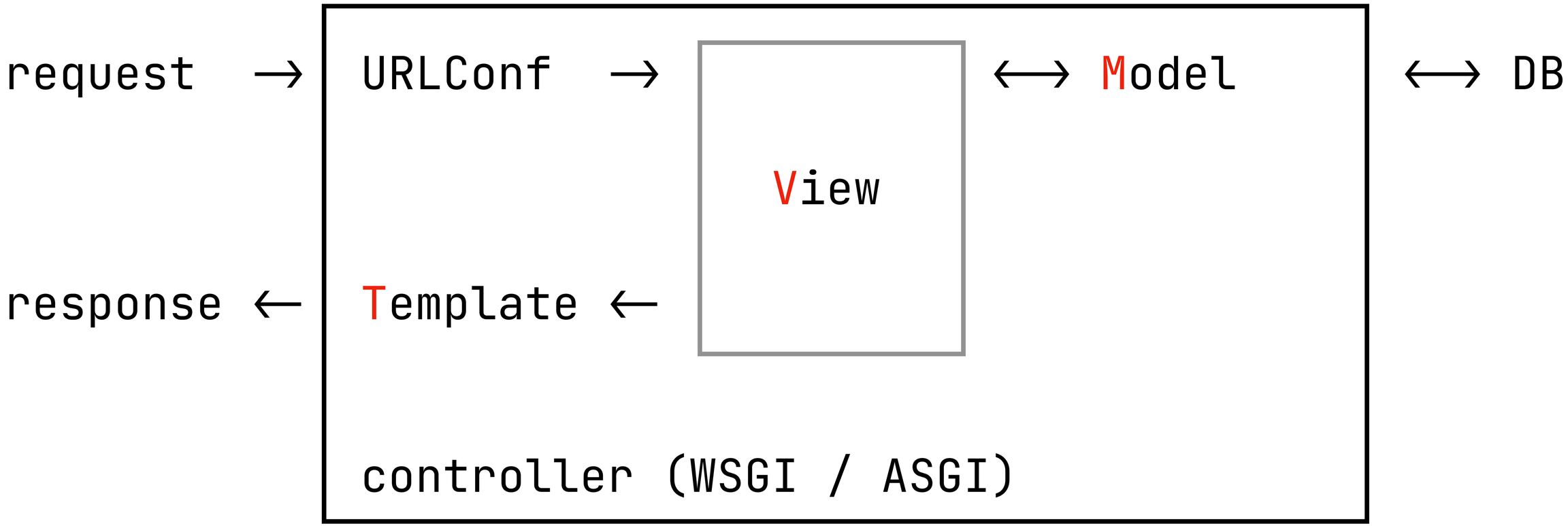
정의

- Model DataBase 연동 (orm)
- View Data 구성 (business logic)
- Template Data 표현 (presentation layer)

* controller : django framework 자체

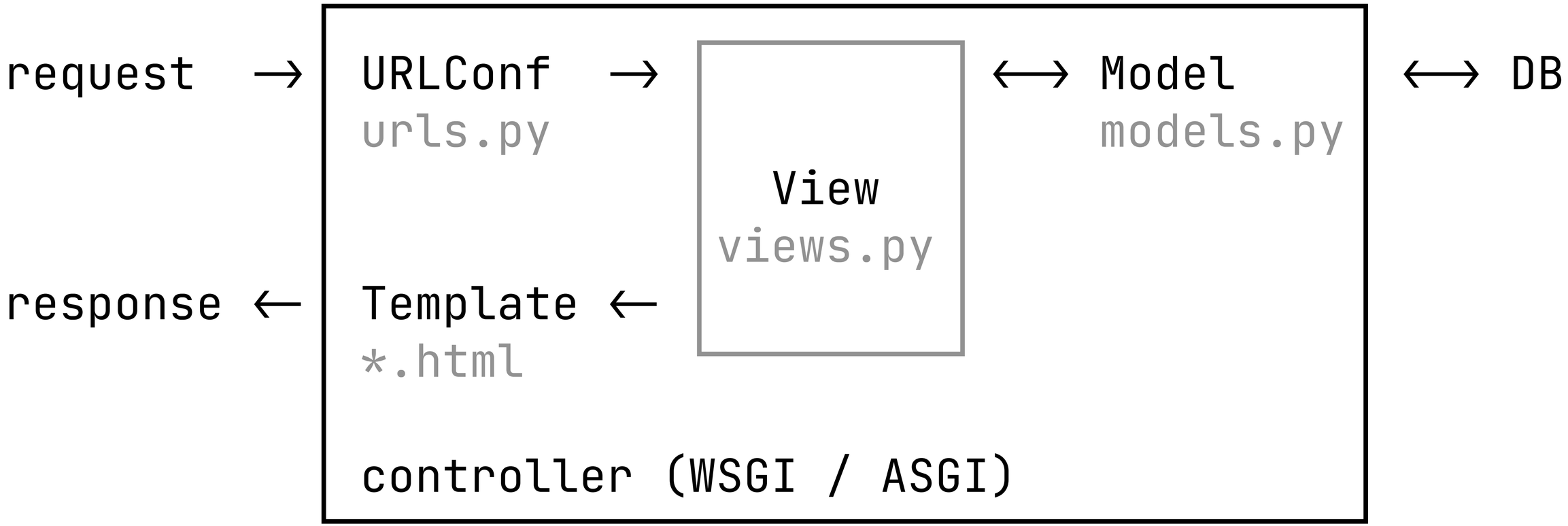
1.MVT

흐름


request →
URLConf →
View
⟷ Model
⟷ DB
response ←
Template ←
controller (WSGI / ASGI)

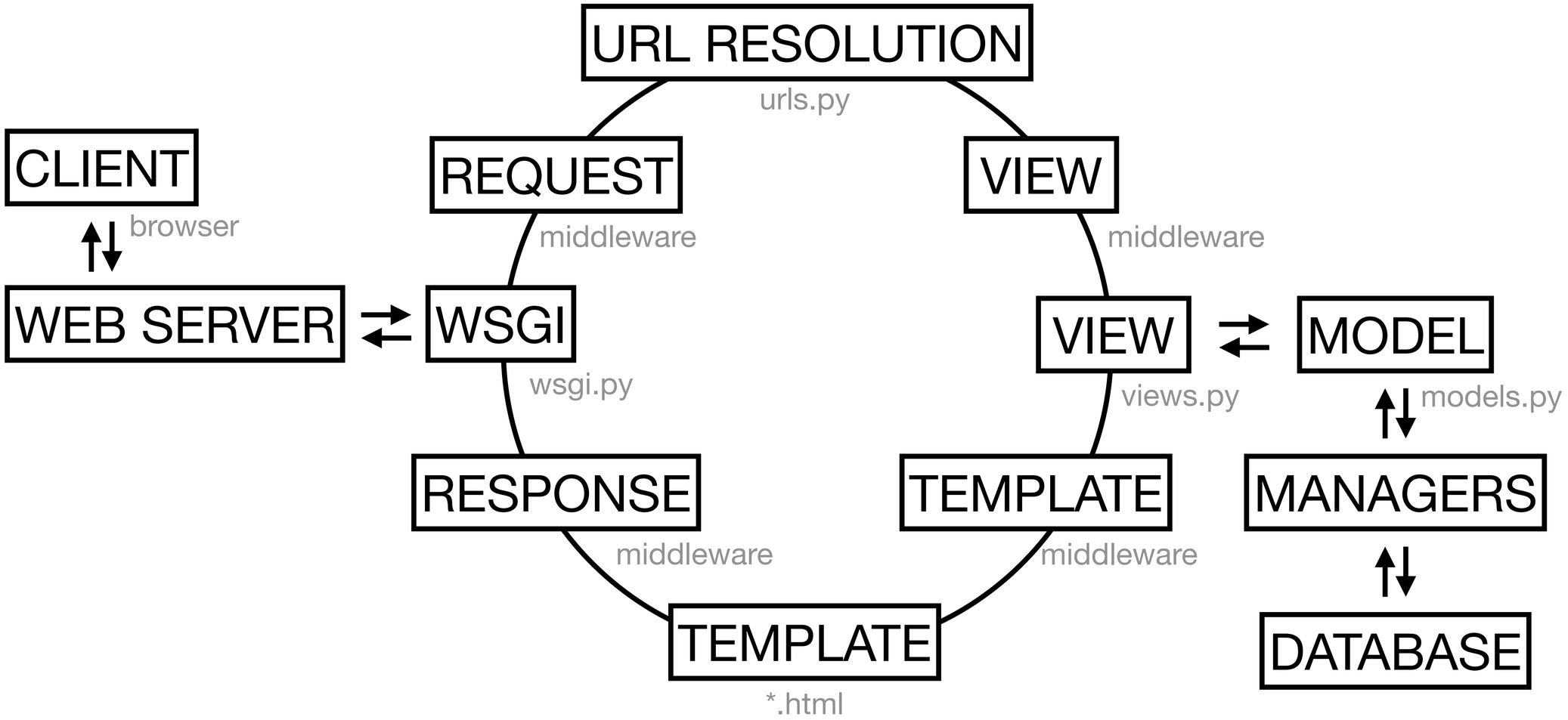
1.MVT

흐름


request
URLConf
urls.py
View
views.py
Model
models.py
DB
response
Template
*.html
controller (WSGI / ASGI)

1.MVT

request-response cycle



2.view

urls.py

- urlpatterns 에 URL 정보를 등록해 view 와 연결

- <변수명>, <type:변수명> 의 형태로 parameter 표현 가능
  path('bio/<username>/', views.bio, name='bio')

- re_path() 함수를 통해 정규식 형태로 url pattern 표현 가능
  re_path(r'^weblog/', include('blog.urls'))

2.view

views.py

- request 를 받아 화면을 구성하여 response

- render(), redirect(), HttpResponse, JsonResponse 등을 return

- 값을 template 에 전달하여 화면 구성 가능

3.templates

표현식

{{ 변수명 }}	Hello, {{name}}!
{{ iterable.index }}	{{lst.0}}
{{ object.key }}	{{dct.class}}
{{ value | filter }}	{{qclass|upper}}

3.templates

표현식

```
{% for %}

{% endfor %}
```

```
{% for i in number%}

  <p>{{i}}</p>

{% endfor %}
```

3.templates

표현식

```
{% if %}

{% elif %}

{% else %}

{% endif %}
```

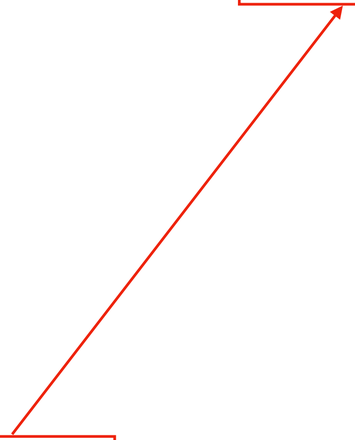
```
{% if i == 1%}
  <p>{{i}} is one</p>
{% elif i == 2 %}
  <p>{{i}} is two</p>
{% else %}
  <p>{{number}} is other</p>
{% endif%}
```

3.templates

표현식

```
{% url %}
```

```
<a href="{% url 'index' %}">index</a>
```



```
urls.py

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

4.model

settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

* sqlites3 / postgrsql / mysql / oracle 사용 가능

4.model

models.py

- django 가 지원하는 RDBMS 추상화 API
  django.db.models.Model

- Model class를 상속받은 class = Table
  models.fieldType = column

- fieldType
  IntegerField, CharField, DateTimefield, …

4.model

apps.py

- Application = django 에 포함되어 있는 구성 저장 및 감사 레지스트리
  = Python package
  (모델, 뷰, 템플릿, 템플릿 태그, 파일, URL, 미들웨어 등)

- AppConfig 를 상속받은 class 를 INSTALLED_APPS 에 추가해야 동작
  (apps.py) (settings.py)

- 구성 설정 및 내부 검사를 위한 메타데이터 인스턴스