Django

0.Install


```
pip install django

django-admin startproject mysite

python manage.py runserver

# http://127.0.0.1:8000
```

0.Install

구조

```
mysite
|-- manage.py
`-- mysite
    |-- asgi.py
    |-- settings.py
    |-- urls.py
    `-- wsgi.py
```

project 관리

주요 명령어

- startapp
- runserver
- makemigrations
- migrate

0.Install


```
mysite
|-- manage.py
`-- mysite
    |-- asgi.py
    |-- settings.py
    |-- urls.py
    `-- wsgi.py
```

Web Server Gateway Interface

WebServer, WAS의 통신 지원

wsgi → asgi 로 대체

0.Install


```
mysite
|-- manage.py
`-- mysite
    |-- asgi.py
    |-- settings.py
    |-- urls.py
    `-- wsgi.py
```

Asynchronous Server Gateway Interface

WebServer, WAS의 동기/비동기 통신 지원

(django 3.0 이상)

0.Install

구조

```
mysite
|-- manage.py
`-- mysite
    |-- asgi.py
    |-- settings.py
    |-- urls.py
    `-- wsgi.py
```

project 환경 설정

주요 설정

- ALLOW_HOSTS
- INSTALLED_APPS
- TEMPLATES
- DATABASES
- STATIC_URL

0.Install


```
mysite
|-- manage.py
`-- mysite
    |-- asgi.py
    |-- settings.py
    |-- urls.py
    `-- wsgi.py
```

URLConf (URL Configuration)

url ⟷ function mapping

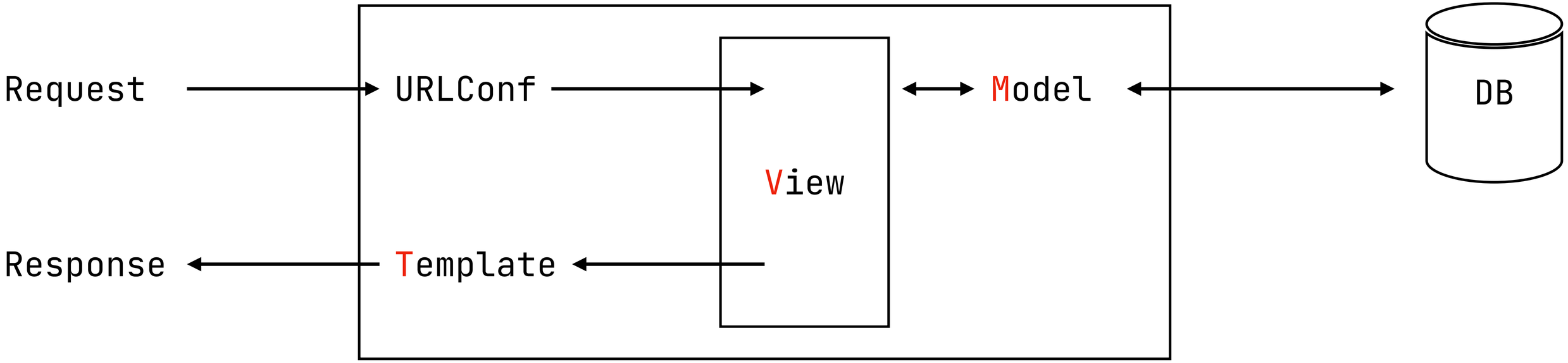
요청에 맞는 작업 호출

1.개념

Model : DataBase 연동 (ORM)

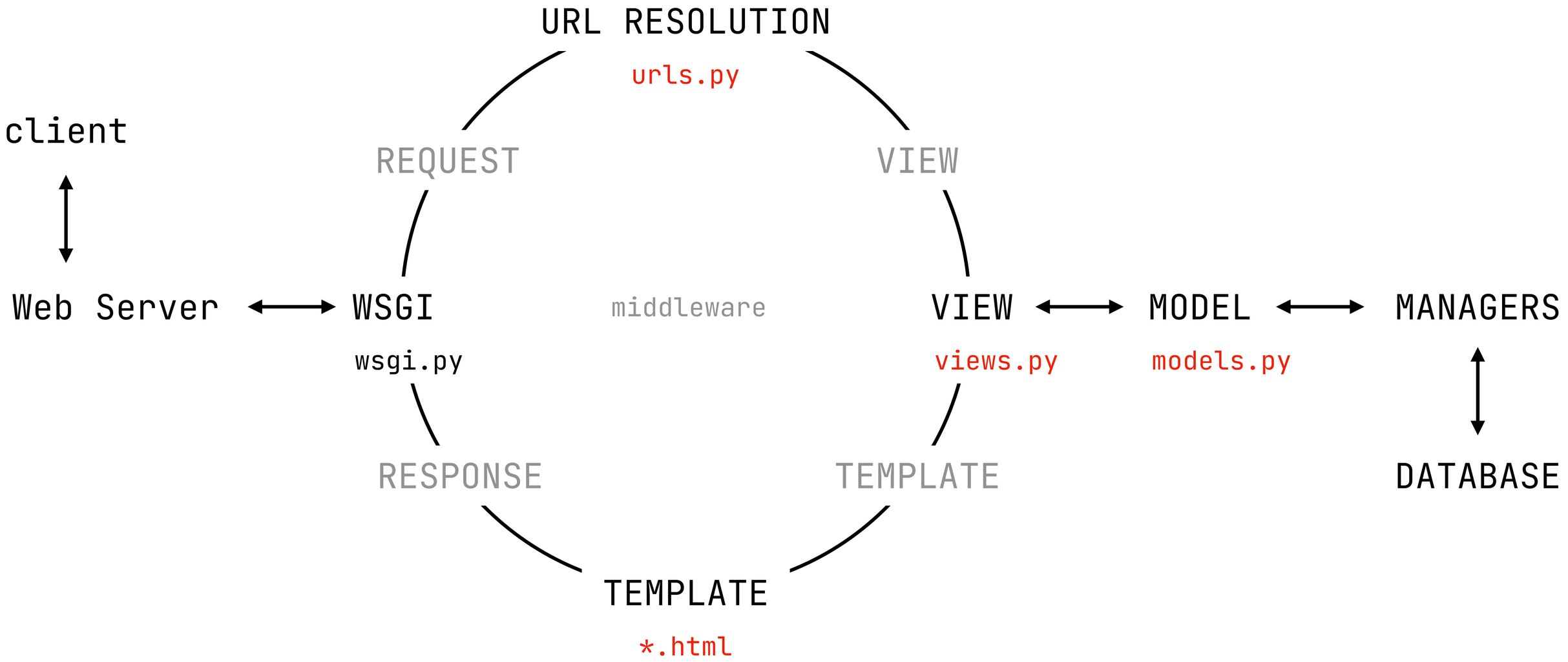
View : Data 구성 (Business Logic)

Template : Data 표현 (presentation)



1.개념

request-response cycle



2.문법

urls.py

path 함수를 사용해 url ⟷ function (views.py) mapping

```
path('mysite/hello', views.hello)
```

path(<변수명>), path(<type:변수명>) 형태로 parameter 표현 가능

```
path('mysite/greetings/<name>', views.greetings)
path('mysite/greetings/<name>/<int:age>', views.greetings)
```

re_path 함수를 사용해 정규식 형태로 mappging

```
re_path(r'^mytest/', include('mytest.urls'))
```

2.문법

views.py

Request를 받아 화면을 구성하여 Response

```
def hello(request):
    return HttpResponse(“<h1>Hello, World!</h1>”)
```

render, redirect, HttpResponse, JsonResponse 등의 함수를 사용하여 return

```
def greetings(request, name):
    return render(request, ‘greeting.html’, {“name”: name})
```

data를 template에 전달하여 화면을 구성 (rendering)

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

2.문법

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

2.문법


```
{% for %}

{% endfor %}
```

```
{% for i in number %}
  <p>{{ i }}</p>
{% endfor %}
```

```
{% if %}

{% elif %}

{% else %}

{% endif %}
```

```
{% if i == 1 %}
  <p>{{ i }}</p>
{% elif i == 2 %}
  <p>second</p>
{% else %}
  <p>3~</p>
{% endif %}
```

2.문법

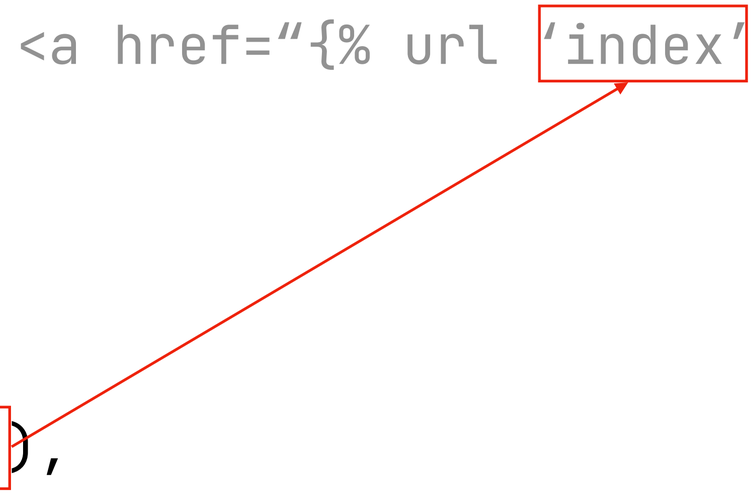
templates

```
{% url %}
```

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

urls.py

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



2.문법


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

* sqlites3 / postgrsql / mysql / oracle 사용 가능

2.문법

django 가 지원하는 RDBMS 추상화 API

Table = Model class를 상속받는 class

Column = models.fieldType

```
django.db.models.Model

class MyBoard(Model):
    myname = models.CharField()
```

3.frameworks

installed app

django.contrib.admin	database web으로 제어
django.contrib.auth	login / logout
django.contrib.contenttype	models 사용 관련
django.contrib.sessions	session 사용 관련
django.contrib.messages	request에 message 담아서 사용
django.contrib.staticfiles	static file (css, js, ...)