ICT12367

การใช้กรอบงานสำหรับการพัฒนาเว็บแอปพลิเคชัน เพื่อความมั่นคงปลอดภัย

Chapter 7

MVT และ URL & View

คำสั่งที่ใช้ในการสร้างโปรเจกต์ Django

- Django เป็น Web Framework ของ Python สามารถใช้สร้างโปรเจกต์และแอปพลิเคชัน คำสั่งที่ใช้ในการสร้างโปรเจกต์ Django มีดังนี้:
  - สร้างโปรเจกต์ Django ใช้คำสั่ง:

Django-admin startproject <ชื่อโปรเจกต์>

  - ตัวอย่าง:

```
django-admin startproject myproject
```

คำสั่งที่ใช้ในการสร้างโปรเจกต์ Django

❑ โครงสร้างที่ได้::

```
myproject/
|── manage.py              # ไฟล์หลักที่ใช้รันคำสั่ง Django
|── myproject/             # ไดเรกทอรีหลักของโปรเจกต์
|    |── __init__.py       # ทำให้โฟลเดอร์นี้เป็น Python Package
|    |── settings.py       # ไฟล์ตั้งค่าของ Django
|    |── urls.py           # กำหนด URL Routing
|    |── asgi.py           # สำหรับ ASGI Server
|    |── wsgi.py           # สำหรับ WSGI Server
```

- startproject จะสร้างโครงสร้างพื้นฐานของโปรเจกต์ Django

คำสั่งที่ใช้ในการสร้างโปรเจกต์ Django

❑ รันเซิร์ฟเวอร์ Django

- เมื่อต้องการทดสอบเว็บ ให้ใช้คำสั่ง:

Python manage.py runserver

- หลังจากรันเซิร์ฟเวอร์แล้ว จะเห็นข้อความประมาณนี้:
- เปิด เว็บเบราว์เซอร์ แล้วไปที่ http://127.0.0.1:8000/ เพื่อดูหน้าเว็บของ Django

```
Starting development server at http://127.0.0.1:8000/
```

คำสั่งที่เกี่ยวข้อง

คำสั่ง	คำอธิบาย
`pip install django`	ติดตั้ง Django
`django-admin startproject ชื่อโปรเจกต์`	สร้างโปรเจกต์ใหม่
`cd ชื่อโปรเจกต์`	เข้าไปในโฟลเดอร์โปรเจกต์
`python manage.py runserver`	รันเซิร์ฟเวอร์
`python manage.py startapp ชื่อแอป`	สร้างแอปใหม่
`python manage.py migrate`	ใช้การตั้งค่าฐานข้อมูล
`python manage.py createsuperuser`	สร้างผู้ดูแลระบบ (Admin)
`python manage.py makemigrations`	สร้างไฟล์ Migration สำหรับ Model
`python manage.py shell`	เข้าสู่ Django Shell

เบื้องต้นที่ควรรู้

- เข้าสู่โฟลเดอร์โปรเจกต์ ใช้คำสั่ง cd <ชื่อโปรเจกต์>
  - cd <ชื่อโปรเจกต์> เป็นคำสั่งของ Command Line (CLI) หรือ Terminal ซึ่งใช้สำหรับเปลี่ยนไดเรกทอรีไปยังโฟลเดอร์ที่ระบุ

```
cd<ชื่อโปรเจกต์>
```

- วิธีใช้ cd.. ใน Command Line หรือ Terminal
  - cd.. เป็นคำสั่งที่ใช้ใน Command Line เพื่อ ย้อนกลับไปยังไดเรกทอรีก่อนหน้า (Parent Directory)

```
cd..
```

คำสั่งที่เกี่ยวข้อง

คำสั่ง	ความหมาย	ใช้ที่ไหน
`cd..`	ย้อนกลับไป 1 ระดับ	Windows CMD, PowerShell
`cd ..`	ย้อนกลับไป 1 ระดับ	macOS, Linux Terminal
`os.chdir("..")`	ย้อนกลับไป 1 ระดับ	Python
`os.chdir("../..")`	ย้อนกลับไป 2 ระดับ	Python
`cd /`	ไปยังไดเรกทอรีราก (root)	macOS, Linux
`cd C:\Path\To\Folder`	ไปยังไดเรกทอรีที่ระบุ	Windows
`cd ~/Desktop`	ไปยัง Desktop	macOS, Linux

โครงสร้างโปรเจกต์

- manage.py คือไฟล์ script สำหรับรันคำสั่งต่างๆ ที่เกี่ยวข้องกับ Django เช่น Run Server, Modal & Migration
- __init__.py คือ initial ไฟล์หรือไฟล์เปล่าๆ มีไว้เก็บ Python Package เราสามารถเพิ่ม Script การทำงานเข้าไปในไฟล์นี้ได้
- settings.py คือไฟล์ที่ใช้สำหรับการตั้งค่าโปรเจค เช่น การตั้งค่า แอพ, เวลา, Path, ฐานข้อมูลที่ใช้เป็นต้น

โครงสร้างโปรเจกต์

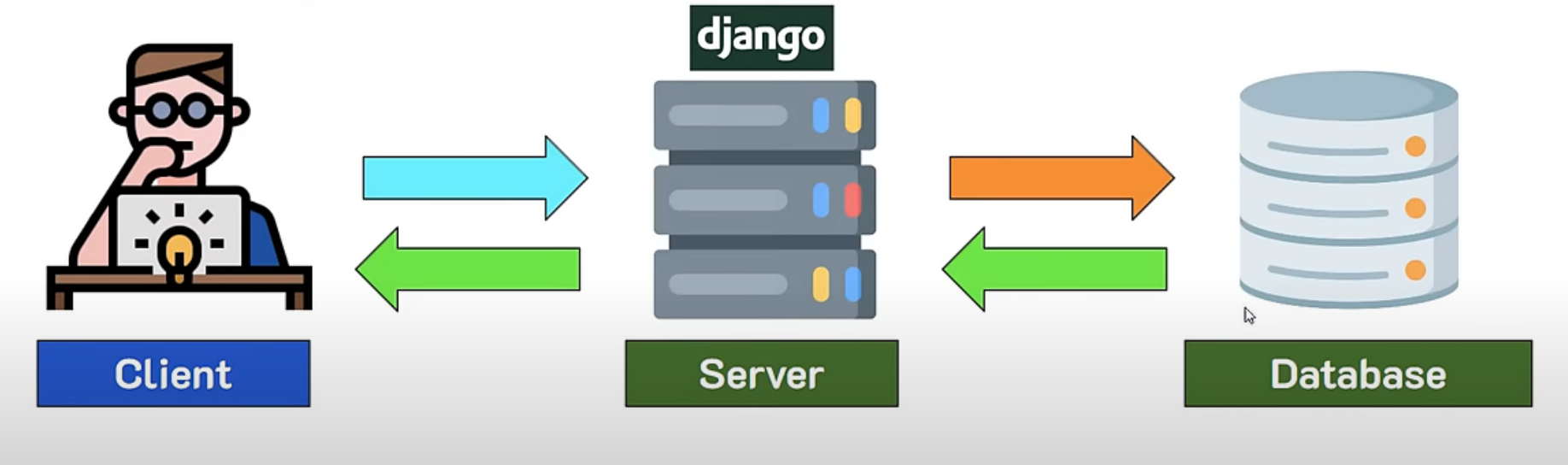
- urls.py คือไฟล์ที่ใช้เก็บการ routing ของ HTTP request หรือ เรียกอีกอย่างว่าการกำหนด urlpattern ของ Django project
- wsgi.py คือไฟล์ที่ใช้เก็บข้อมูลโปรเจคสำหรับการ Deployment

รู้จักกับ MVT (Model-View-Template)

- Modal (M) คือส่วนที่เก็บข้อมูลของ Application
- View (V) คือส่วนประมวลผลคำสั่งหรือข้อมูลต่างๆ โดยควบคุมการทำงานระหว่าง Modal และ Template
- Template (T) คือหน้าตา Application เป็นส่วนที่ไว้ใช้แสดงผลข้อมูล ผลลัพธ์จากการประมวลผลข้อมูลในหน้าเว็บร่วมกับ HTML

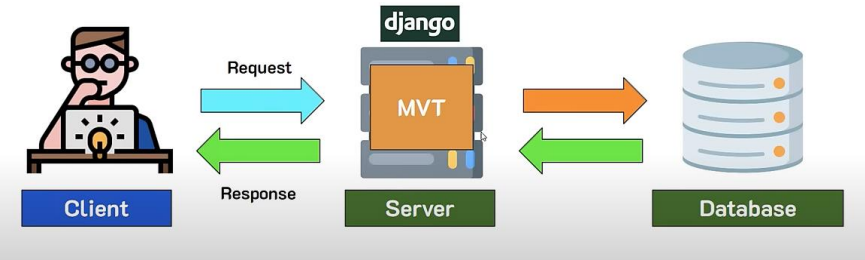
รู้จักกับ MVT (Model-View-Template)

Django ใช้ MVT (Model-View-Template) เป็นโครงสร้างหลักในการพัฒนาเว็บแอปพลิเคชัน ครงสร้างการทำงานของ Django จะเห็นได้ว่า Django เป็น Web Framework ที่ทำงานในรูปแบบ Client-Server-Database โดยใช้ MVT Architecture (Model-View-Template)



รู้จักกับ MVT (Model-View-Template)

❑การทำงานของ Django



1. **Client (ผู้ใช้งาน)** ผู้ใช้ส่งคำขอ (Request) ผ่านเว็บเบราว์เซอร์ ไปยังเซิร์ฟเวอร์ของ Django
2. **Django Server (Backend)** Django Framework ทำหน้าที่เป็นตัวกลางระหว่าง Client และ Database เมื่อลูกค้าส่งคำขอ Django จะใช้ View (views.py) เพื่อตรวจสอบว่า Client ต้องการทำอะไร
3. **Database (ฐานข้อมูล)** เป็นที่เก็บข้อมูล Django ใช้ ORM (Object-Relational Mapping) ช่วยให้สามารถจัดการฐานข้อมูลผ่าน Python ได้โดยไม่ต้องใช้ SQL โดยตรง
4. **Django ตอบกลับ Client** หลังจากประมวลผลเสร็จ Django จะใช้ Template (HTML, CSS, JavaScript) เพื่อสร้างหน้าเว็บ

โครงสร้าง MVT ใน Django

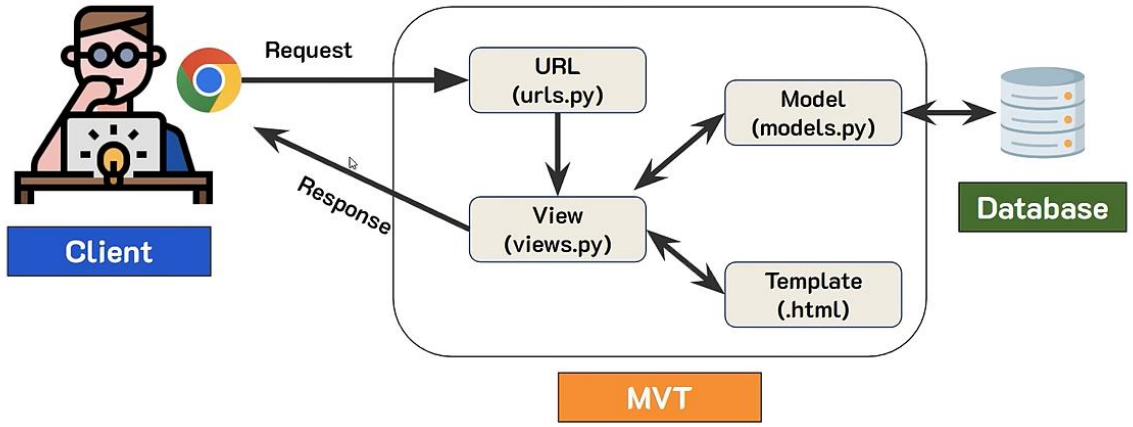
- MVT เป็นสถาปัตยกรรมที่ช่วยให้พัฒนาเว็บได้ง่ายขึ้น โดยแบ่งเป็น 3 ส่วน:

องค์ประกอบ	ทำหน้าที่อะไร?	ตัวอย่างไฟล์
Model	จัดการข้อมูลและฐานข้อมูล	`models.py`
View	จัดการตรรกะของโปรแกรมและการประมวลผลข้อมูล	`views.py`
Template	แสดงผลข้อมูลให้กับผู้ใช้ (Frontend)	`templates/*.html`

กระบวนการทำงานของ Django เมื่อมีการขอข้อมูล

1. Client เปิดเว็บ http://127.0.0.1:8000/
2. คำขอจะถูกส่งไปยัง View (views.py)
3. View จะดึงข้อมูลจาก Model (models.py)
4. ข้อมูลถูกส่งไปยัง Template (home.html)
5. Django แปลงข้อมูลเป็น HTML และส่งกลับไปยัง Client
6. Client เห็นข้อมูลที่ถูกแสดงผลบนหน้าเว็บ

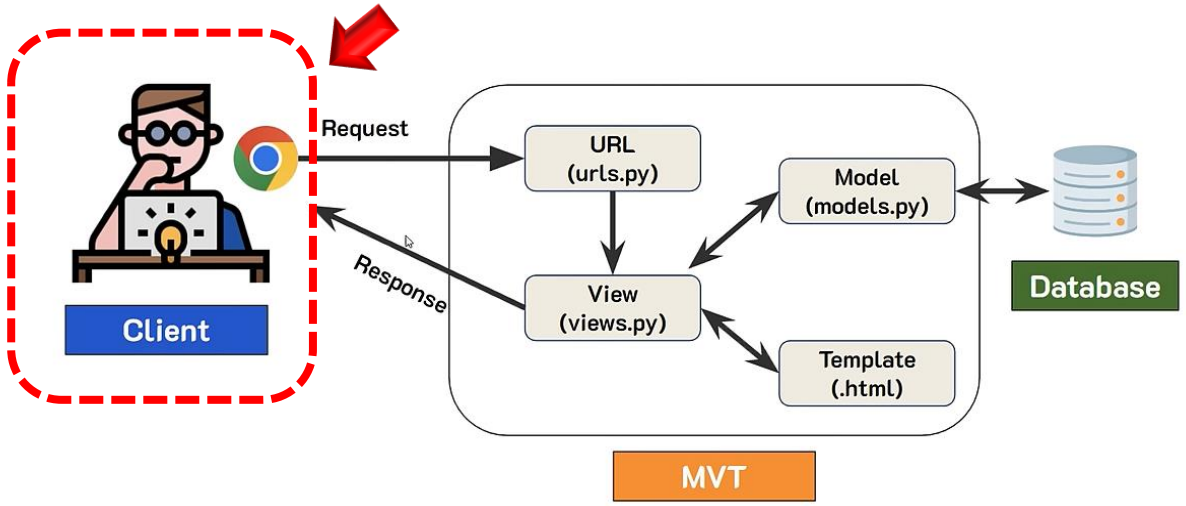
รู้จักกับ MVT (Model-View-Template)



การทำงานของ Django Web Framework ซึ่งใช้ MVT Architecture (Model-View-Template) ในการประมวลผลคำขอ (Request) จาก Client และตอบกลับ (Response) โดยแยกหน้าที่ของแต่ละส่วนออกจากกันอย่างชัดเจน

รู้จักกับ MVT (Model-View-Template)

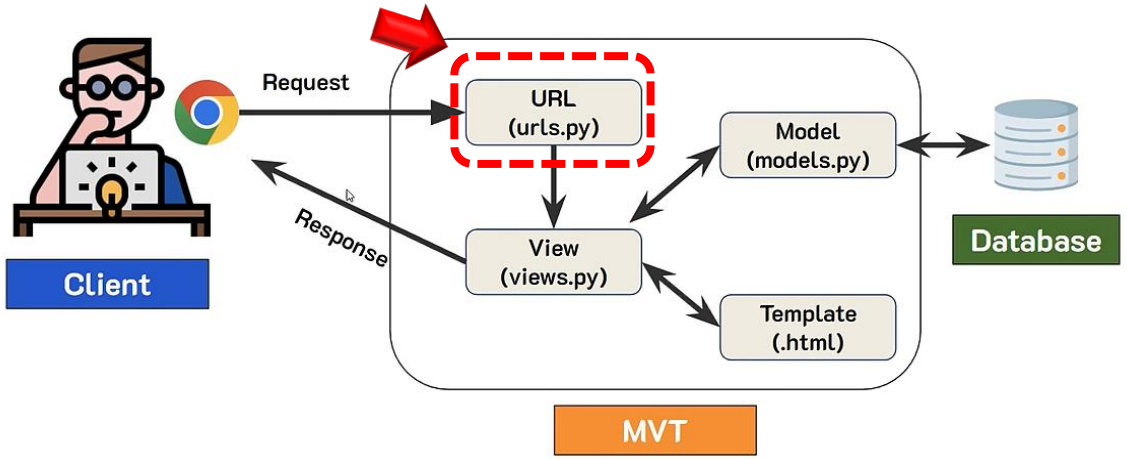
❑ กระบวนการทำงานของ MVT ใน Django



1. Client ส่งคำขอ (Request)
   - ผู้ใช้ (Client) ใช้ เว็บเบราว์เซอร์ (เช่น Google Chrome) ส่งคำขอ HTTP ไปยังเซิร์ฟเวอร์ของ Django
   - เช่น เปิดหน้าเว็บ http://127.0.0.1:8000/home/

รู้จักกับ MVT (Model-View-Template)

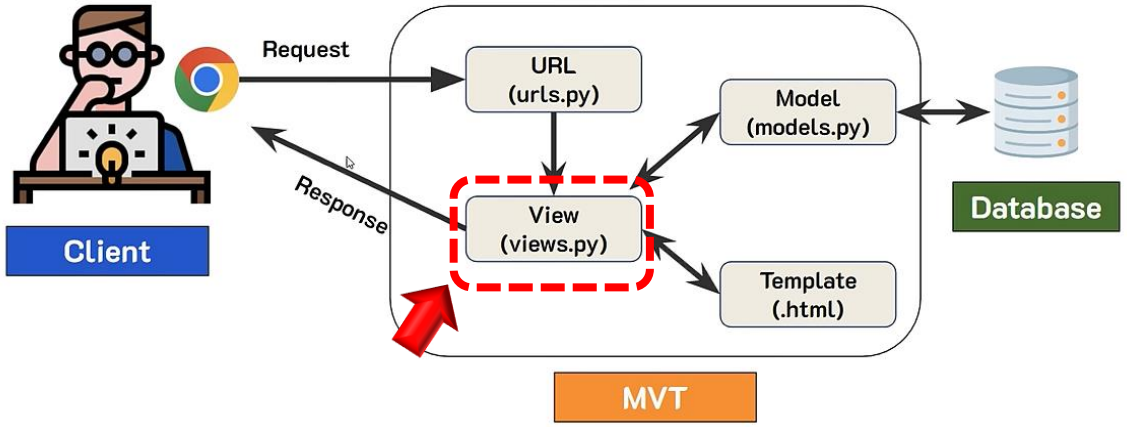
❑ กระบวนการทำงานของ MVT ใน Django



2. URL Mapping (urls.py)
   - คำขอที่เข้ามาจะถูกตรวจสอบใน urls.py
   - Django จะตรวจสอบว่า URL ที่ร้องขอมีฟังก์ชัน View ใดที่ต้องประมวลผล

รู้จักกับ MVT (Model-View-Template)

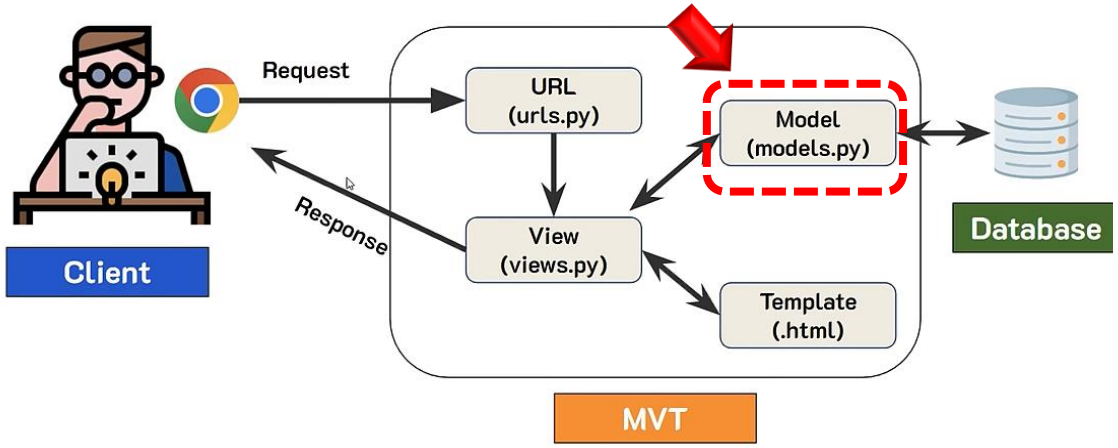
❑ กระบวนการทำงานของ MVT ใน Django



3. View (views.py) ประมวลผลคำขอ
   - views.py เป็นจุดที่ประมวลผลตรรกะหลักของแอป
   - View อาจทำสิ่งต่อไปนี้:
     - ดึงข้อมูลจาก Model (ฐานข้อมูล)
     - ส่งข้อมูลไปยัง Template (HTML)
     - ประมวลผลและส่งข้อมูลกลับไปให้ผู้ใช้

รู้จักกับ MVT (Model-View-Template)

❑ กระบวนการทำงานของ MVT ใน Django

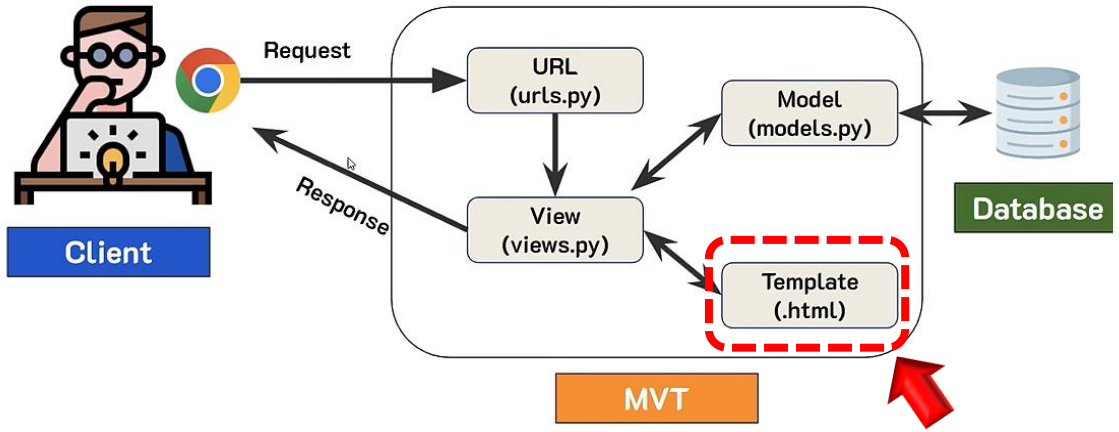


4. Model (models.py) จัดการฐานข้อมูล

- หาก View ต้องการข้อมูลจากฐานข้อมูล จะเรียกใช้ Model
- Model ใน Django ใช้ ORM (Object-Relational Mapping) ช่วยให้จัดการฐานข้อมูลโดยใช้ Python แทน SQL
- Model ดึงข้อมูลจาก Database และส่งกลับไปยัง View

รู้จักกับ MVT (Model-View-Template)

❑ กระบวนการทำงานของ MVT ใน Django

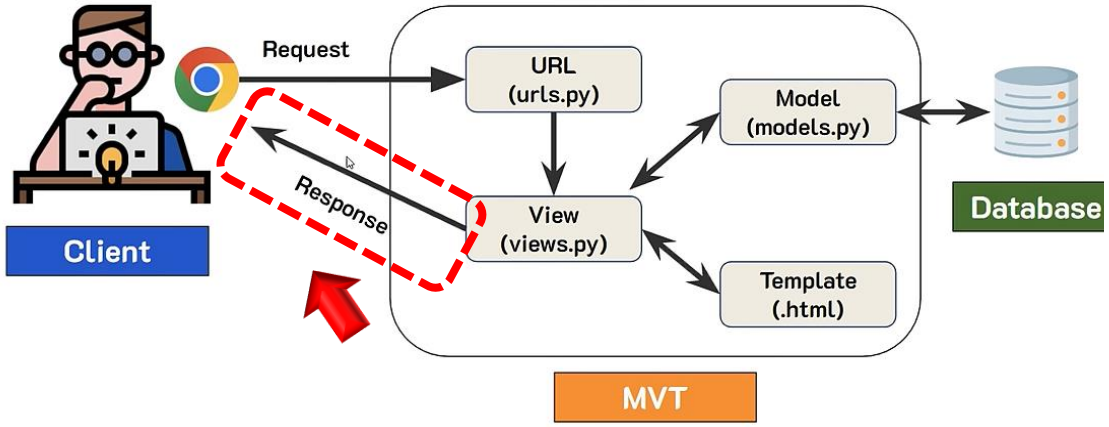


5. Template (.html) สร้างหน้าเว็บ

- View จะใช้ Template (.html) เพื่อนำข้อมูลที่ดึงจากฐานข้อมูลมาแสดงผลให้กับผู้ใช้
- Django ใช้ Django Template Language (DTL) ช่วยจัดการแสดงผลข้อมูล

รู้จักกับ MVT (Model-View-Template)

❑ กระบวนการทำงานของ MVT ใน Django



6. Django ส่ง Response กลับไปยัง Client

- Template ที่ได้รับการประมวลผลแล้วจะถูกส่งกลับเป็น HTML ไปยังเบราว์เซอร์ของผู้ใช้ผู้ใช้
- จะเห็นหน้าเว็บที่มีข้อมูลที่ร้องขอ

สรุป MVT ใน Django

องค์ประกอบ	ทำหน้าที่อะไร?	ตัวอย่างไฟล์
Model	จัดการฐานข้อมูล	`models.py`
View	ควบคุมตรรกะของแอปและดึงข้อมูล	`views.py`
Template	แสดงผลข้อมูลให้ผู้ใช้	`.html`
URL Mapping	กำหนดเส้นทางของ URL	`urls.py`

การสร้างแอปพลิเคชันใน Django และโครงสร้างของ Project & App

การสร้างแอปพลิเคชันใน Django สามารถแบ่งองค์ประกอบออกเป็นส่วนย่อยๆ แล้วนำมาทำงานร่วมกันในภายหลังได้ เรียกว่า แอพ เพื่อจัดการเกี่ยวกับระบบย่อยต่างๆ ภายในโปรเจกต์ ซึ่งภาพรวมของระบบสามารถแบ่งการทำงานออกเป็น 2 ระดับ คือ

- Project Level คือ ระบบหลัก สามารถดำเนินการกับโปรเจกต์ได้โดยตรง ใช้จัดการการทำงานโดยรวมของเว็บแอปพลิเคชัน เช่น การตั้งค่าระบบ, URL หลัก, Middleware
- App level คือระบบย่อยเป็นการดำเนินการกับระบบย่อยต่างๆ ในโปรเจค เช่น ระบบสมาชิก ระบบหมวดหมู่ เป็นต้น

การสร้างโปรเจกต์และแอปพลิเคชัน ใน Django

1. สร้างโปรเจกต์ Django (Project Level) ก่อนอื่นต้องติดตั้ง Django (หากยังไม่ได้ติดตั้ง)

```
pip install django
```

- จากนั้นสร้างโปรเจกต์ใหม่โดยใช้คำสั่ง:

```
django-admin startproject myproject
```

การสร้างโปรเจกต์และแอปพลิเคชันใน Django

❑ ผลลัพธ์ที่ได้

```
myproject/
|— manage.py              # ใช้จัดการโปรเจกต์
|— myproject/             # ไดเรกทอรีหลักของโปรเจกต์
|    |— __init__.py
|    |— settings.py       # ตั้งค่าระบบ
|    |— urls.py           # จัดการเส้นทาง URL
|    |— asgi.py
|    |— wsgi.py
```

การสร้างโปรเจกต์และแอปพลิเคชัน ใน Django

2. **สร้างแอปพลิเคชัน (App Level)** โปรเจกต์ Django สามารถสร้างแอปได้หลายตัว ตัวอย่างเช่น ระบบสมาชิก (users), ระบบโพสต์ (posts), ระบบร้านค้า (store)

Python manage.py startapp <ชื่อแอพ>

```
python manage.py startapp users
```

การสร้างโปรเจกต์และแอปพลิเคชันใน Django

- โครงสร้างของแอป users

```
users/
├── migrations/          # ใช้เก็บการเปลี่ยนแปลงของฐานข้อมูล
├── __init__.py
├── admin.py             # จัดการใน Django Admin
├── apps.py              # ตั้งค่าแอป
├── models.py            # สร้างฐานข้อมูล
├── tests.py             # เขียน Unit Test
└── views.py             # ควบคุมการทำงานของแอป
```

การสร้างโปรเจกต์และแอปพลิเคชัน ใน Django

3.เพิ่มแอปเข้าไปในโปรเจกต์ (settings.py) หลังจากสร้างแอปแล้ว ต้องเพิ่มลงใน INSTALLED_APPS ของ settings.py

Settings.py

INSTALLED_APPS = [

‘ชื่อแอพ’

]

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'users',  # แอป users ที่สร้างขึ้นใหม่
]
```

การสร้างโปรเจกต์และแอปพลิเคชันใน Django

4. **กำหนด URL ให้แอป (urls.py)** สร้างไฟล์ urls.py ภายในแอป users:

```
touch users/urls.py
```

- จากนั้นเพิ่ม URL pattern ลงไป:

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

urlpatterns = [
    path('login/', views.login_view, name='login'),
    path('register/', views.register_view, name='register'),
]
```

การสร้างโปรเจกต์และแอปพลิเคชันใน Django

4. **กำหนด URL ให้แอป (urls.py)** สร้างไฟล์ urls.py ภายในแอป users:

   - จากนั้นต้องเพิ่ม URL ของแอปนี้ใน myproject/urls.py:

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('users/', include('users.urls')),  # เชื่อม URL ของแอป users
]
```

การสร้างโปรเจกต์และแอปพลิเคชันใน Django

5. สร้าง View สำหรับแอป

- ไปที่ users/views.py แล้วเพิ่มฟังก์ชันเพื่อจัดการการลงชื่อเข้าใช้และสมัครสมาชิก:

```python
from django.http import HttpResponse

def login_view(request):
    return HttpResponse("Login Page")

def register_view(request):
    return HttpResponse("Register Page")
```

การสร้างโปรเจกต์และแอปพลิเคชันใน Django

6. ทดสอบการรันโปรเจกต์

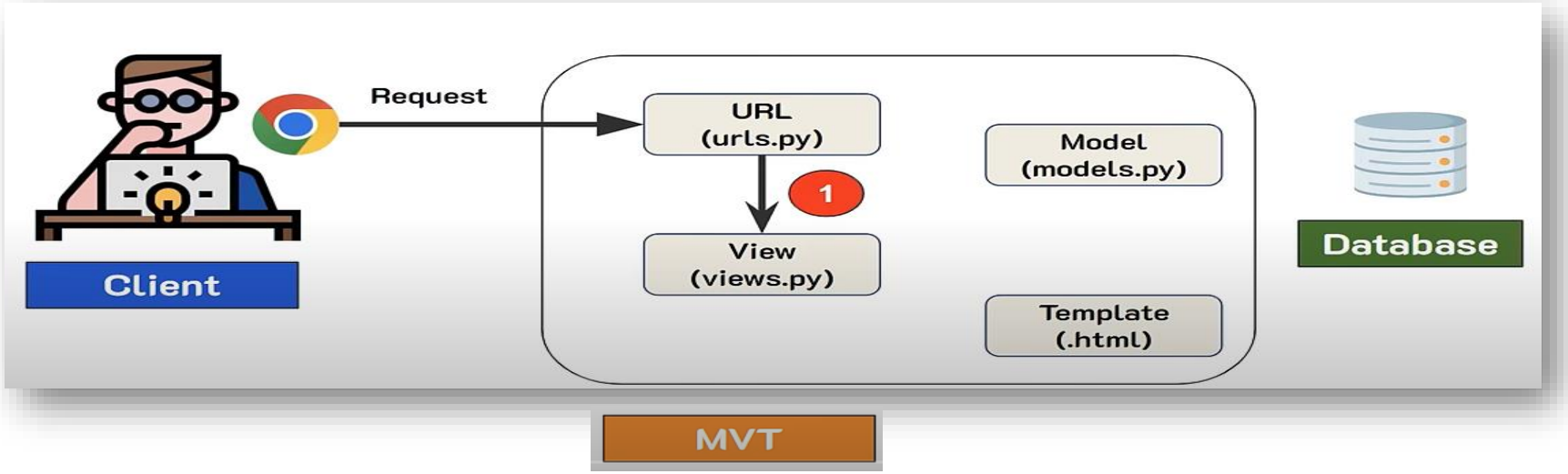
- รันเซิร์ฟเวอร์ด้วยคำสั่ง:

```
python manage.py runserver
```

- จากนั้นเปิด เว็บเบราว์เซอร์ แล้วไปที่:
  - http://127.0.0.1:8000/users/login/ → จะแสดงข้อความ "Login Page"
  - http://127.0.0.1:8000/users/register/ → จะแสดงข้อความ "Register Page"

URL & View

ใน Django Framework เมื่อ Client (ผู้ใช้) ส่ง Request มายังเซิร์ฟเวอร์ Django จะใช้ URL & View เพื่อควบคุมว่าแต่ละ Request ควรให้ข้อมูลอะไรกลับไป และควรประมวลผลอย่างไร



URL & View

❑ สรุปภาพรวมการทำงาน

องค์ประกอบ	ทำหน้าที่อะไร?	ไฟล์ที่เกี่ยวข้อง
Client	ผู้ใช้ส่ง Request ผ่านเบราว์เซอร์	-
URL Mapping	จับคู่ URL กับ View	`urls.py`
View	ควบคุมตรรกะของแอป และเรียก Model หรือ Template	`views.py`
Model (ถ้ามี)	จัดการฐานข้อมูล	`models.py`
Template	แสดงผล HTML ให้กับผู้ใช้	`.html`

URL & View

- URL คือส่วนที่ใช้ระบุเส้นทางในการรับส่งข้อมูล
- View คือ ศูนย์กลางสำหรับรับส่งข้อมูล โดยเชื่อมโยงการทำงานระหว่าง Modal และ template

URL & View

- โครงสร้าง URL (Uniform Resource Locator) ซึ่งใช้ระบุที่อยู่ของเว็บเพจหรือทรัพยากรต่าง ๆ บนอินเทอร์เน็ต โดยประกอบไปด้วย 3 ส่วนหลัก ได้แก่ Protocol, Domain และ Path

  - องค์ประกอบของ URL


```
https://www.example.com/project/computer
```

URL & View

- องค์ประกอบของ URL

1. **Protocol (โปรโตคอล)** เป็นส่วนแรกของ URL กำหนดรูปแบบการสื่อสารระหว่าง Client (เว็บเบราว์เซอร์) กับ Server

- ตัวอย่าง Protocol ที่ใช้บ่อย:
  - http:// → โปรโตคอลปกติ (ไม่เข้ารหัสข้อมูล)
  - https:// → โปรโตคอลที่ปลอดภัย (ข้อมูลถูกเข้ารหัส)

✅ เช่น:

- https://www.example.com → ใช้ HTTPS (ปลอดภัยกว่า HTTP)
- http://www.example.com → ใช้ HTTP (ข้อมูลไม่เข้ารหัส อาจไม่ปลอดภัย)

URL & View

- องค์ประกอบของ URL

2. **Domain (โดเมน)** คือที่อยู่ของเว็บไซต์หรือเซิร์ฟเวอร์ที่ให้บริการ

- โดเมนใช้สำหรับระบุเว็บเซิร์ฟเวอร์บนอินเทอร์เน็ต เช่น:
  - www.example.com
  - www.google.com
  - www.facebook.com

✅ ตัวอย่าง

- https://www.example.com → โดเมน example.com
- https://www.google.com → โดเมน google.com

URL & View

- องค์ประกอบของ URL

3. **Path (พาธ)** เป็นส่วนที่อยู่หลังโดเมน ใช้ระบุหน้าเว็บหรือไฟล์ที่ต้องการเข้าถึง

    - แสดงเป็นโครงสร้างไดเรกทอรี เช่น:
        - /project/computer
        - /news/article123
        - /blog/python-tutorial

✅ ตัวอย่าง

- https://www.example.com/project/computer → Path คือ /project/computer
- https://www.example.com/blog/python-tutorial → Path คือ /blog/python-tutorial

URL & View

- สรุปโครงสร้างของ URL

องค์ประกอบ	ความหมาย	ตัวอย่าง
Protocol	ระบุรูปแบบการสื่อสาร	`https://`
Domain	ที่อยู่ของเว็บไซต์	`www.example.com`
Path	ไดเรกทอรีหรือหน้าที่ต้องการเข้าถึง	`/project/computer`
Query String (ถ้ามี)	พารามิเตอร์เพิ่มเติม	`?q=django&page=2`
Fragment (ถ้ามี)	ระบุตำแหน่งบนหน้าเว็บ	`#section3`

URL & View

- ตัวอย่าง URL และการทำงาน

URL	อธิบาย
`https://www.google.com`	เปิดหน้าเว็บ Google
`https://www.example.com/blog`	เข้าสู่หน้า Blog
`https://www.shop.com/products?page=2`	เปิดหน้าที่ 2 ของสินค้าทั้งหมด
`https://news.com/article?id=123`	เปิดบทความที่มี ID = 123
`https://www.example.com/about#team`	เปิดหน้า About และเลื่อนไปยังส่วน Team

url.py

❑ ความสำคัญของโครงสร้าง URL ใน Django

ไฟล์สำหรับจัดการเส้นทางหรือพาธ(path) การทำงาน

องค์ประกอบของ urls.py

- Form Django.urls import path, include คือการนำเอาพาธ (url) มาใช้งาน
- Urlpatterns คือ การกำหนดกลุ่มรูปแบบพาธ(List)

url.py

❑ ตัวอย่าง URL Mapping

URL	View ที่เรียกใช้
`/`	`views.home`
`/about/`	`views.about`
`/blog/5/`	`views.blog_post(post_id=5)`

views.py

ใน Django Framework, views.py เป็นไฟล์ที่ใช้ในการ ประมวลผลคำขอ (Request) และส่งผลลัพธ์ (Response) กลับไปยังผู้ใช้

- View ทำหน้าที่เป็น ตัวกลาง ระหว่าง URL (urls.py) และ Template (.html) หรือ Model (models.py)
- ทุกครั้งที่ผู้ใช้เข้าถึงเว็บ Django URL จะเรียก View เพื่อกำหนดว่าจะตอบกลับอะไร

องค์ประกอบของ view.py

```
Form Django.http import HttpResponse

Def index(request):

        return HttpResponse("ICT12367 SPU")
```

views.py

❑ การทำงานของ views.py

- รับคำขอ (Request) จากผู้ใช้ เช่น เปิดเว็บ http://127.0.0.1:8000/home/
- ประมวลผลข้อมูล (ถ้ามี) เช่น ดึงข้อมูลจาก Model (models.py)
- ส่งผลลัพธ์กลับไปยังผู้ใช้ โดยใช้ Template (.html) หรือส่ง JSON Response

ตัวอย่าง views.py

1 View แบบพื้นฐาน (HttpResponse)

📌 ส่งข้อความธรรมดากลับไปยังผู้ใช้

```python
from django.http import HttpResponse

def home_view(request):
    return HttpResponse("Welcome to Home Page!")
```

✅ เมื่อเปิด http://127.0.0.1:8000/home/ → จะแสดงข้อความ "Welcome to Home Page!"

URL & View

การตั้งค่า URL ในระดับโปรเจกต์ (urls.py) ของ Django เพื่อให้สามารถ เชื่อมโยงไปยัง URL ของแอปย่อย (App Level URL) โดยใช้ include()

ชื่อโปรเจค / urls.py ของโปรเจค

form Django.urls import path, include

urlpatterns = [

path(‘ ’,include(“ชื่อแอพ.urls”)),

]

```python
from django.urls import path, include

urlpatterns = [
    path('', include("ชื่อแอพ.urls")),  # เชื่อมโยงไปยัง urls.py ของแอป
]
```

URL & View

❑ ตัวอย่างใช้งานจริง (myproject/urls.py)

```python
from django.urls import path, include

urlpatterns = [
    path('', include("blog.urls")),  # โหลด urls.py ของแอป blog
]
```

- ✅ เมื่อเปิด http://127.0.0.1:8000/ Django จะไปที่ blog.urls ทันที

URL & View

❑ การกำหนด URL (urls.py) ในระดับแอปของ Django

ชื่อแอพ / urls.py

```
form ชื่อแอพ import views
form Django.urls import path
urlpatterns = [
        path(‘ ’,view.index),
]
```

```python
from django.urls import path
from ชื่อแอพ import views  # นำเข้า views.py

urlpatterns = [
    path('', views.index, name='index'),  # เชื่อมไปยังฟังก์ชัน index ใน views.py
]
```

URL & View

❑ ตัวอย่าง การสร้าง View (views.py)

```python
from django.http import HttpResponse

def index(request):
    return HttpResponse("Welcome to the Home Page of this App!")
```

✅ เมื่อเปิด http://127.0.0.1:8000/ จะแสดงข้อความ

```
Welcome to the Home Page of this App!
```

URL & View

❑ การเชื่อมโยง URL ของแอปไปยังโปรเจกต์

ต้องเพิ่ม URL ของแอปลงในโปรเจกต์หลัก (myproject/urls.py)

✅ ตัวอย่าง myproject/urls.py

```python
from django.urls import path, include

urlpatterns = [
    path('', include("ชื่อแอพ.urls")),  # โหลด urls.py ของแอป
]
```

✅ เมื่อเปิด http://127.0.0.1:8000/ Django จะโหลด urls.py ของแอป

URL & View

- การกำหนด View (views.py) ใน Django ใช้สำหรับ ประมวลผลคำขอ (Request) และส่งคำตอบ (Response) กลับไปยังผู้ใช้

ชื่อแอพ / view.py

```
form Django.http import HttpResponse   # แก้ไขการ import

Def index(request):

        return HttpResponse("ICT12367 SPU") # แสดงข้อความออกไป
```

URL & View

❑ การเชื่อมโยง views.py กับ urls.py

เพื่อให้ View (index()) ทำงานได้ ต้องกำหนด URL ที่จะเรียกใช้งาน

```python
from django.urls import path
from . import views  # นำเข้า views.py ของแอป

urlpatterns = [
    path('', views.index, name='index'),  # เชื่อมโยง URL หลักไปยัง views.index
]
```

✅ เมื่อเปิด http://127.0.0.1:8000/ → Django จะเรียก views.index และแสดง "ICT12367 SPU"


Q&A